
Abjad Documentation

Release 2.8

Víctor Adán, Trevor Bača, Josiah Oberholtzer

April 16, 2012

CONTENTS

1	Abjad?	3
1.1	Abjad extends LilyPond	3
1.2	Abjad extends Python	3
1.3	What next?	4
1.4	Mailing lists	4
2	Installation	5
2.1	Abjad depends on Python	5
2.2	Abjad depends on LilyPond	5
2.3	Installing the current packaged version of Abjad with <code>easy_install</code>	5
2.4	Installing the current packaged version of Abjad from the Python Package Index	6
2.5	After install	6
2.6	Note for Linux users	6
3	Version history	7
3.1	Abjad 2.8	7
3.2	Abjad 2.7	13
3.3	Abjad 2.6	18
3.4	Abjad 2.5	22
3.5	Abjad 2.4	22
3.6	Abjad 2.3	23
3.7	Abjad 2.2	24
3.8	Abjad 2.1	24
3.9	Abjad 2.0	24
3.10	Abjad 1.1.1	26
3.11	Abjad 1.1.0	27
3.12	Abjad 1.0.1055	27
3.13	Abjad 1.0.1022	28
4	Bartók: <i>Mikrokosmos</i>	29
4.1	The score	29
4.2	The measures	29
4.3	The notes	30
4.4	The details	31
5	Ferneyhough: <i>Unsichtbare Farben</i>	35
5.1	The proportions	35
5.2	The transforms	35
5.3	The rhythms	36

5.4	The score	36
6	Ligeti: <i>Désordre</i>	39
6.1	The cell	40
6.2	The measure	41
6.3	The staff	42
6.4	The score	42
7	Mozart: <i>Musikalisches Würfelspiel</i>	45
7.1	The materials	45
7.2	The structure	51
7.3	The score	52
7.4	The document	54
8	Leaf, Container, Spanner, Mark	57
8.1	Example 1	58
8.2	Example 2	59
9	Working with component parentage	61
9.1	Improper parentage	61
9.2	Proper parentage	61
9.3	Parentage attributes	62
10	Working with threads	63
10.1	What is a thread?	63
10.2	What are threads for?	64
10.3	Coda	66
11	Understanding LilyPond grobs	69
11.1	Grobs control typography	69
11.2	Grobs can be overridden	69
11.3	Check the LilyPond docs	70
12	Understanding Abjad overrides	71
12.1	Grob-override component plug-ins	71
12.2	Grob proxies	71
12.3	Dot-chained override syntax	72
13	Time signature marks by example	73
14	Annotations	83
14.1	Creating annotations	83
14.2	Attaching annotations to a component	83
14.3	Getting the annotations attached to a component	83
14.4	Detaching annotations from a component one at a time	84
14.5	Detaching all annotations attached to a component at once	84
14.6	Inspecting the component to which an annotation is attached	84
14.7	Inspecting annotation name	84
14.8	Inspecting annotation value	84
15	Articulations	85
15.1	Creating articulations	85
15.2	Attaching articulations to a leaf	85
15.3	Attaching articulations to many notes and chords at once	85
15.4	Getting the articulations attached to a leaf	86
15.5	Detaching articulations from a leaf one at a time	86

15.6	Detaching all articulations attached to a leaf at once	86
15.7	Inspecting the leaf to which an articulation is attached	86
15.8	Understanding the interpreter display of an articulation that is not attached to a leaf	87
15.9	Understanding the interpreter display of an articulation that is attached to a leaf	87
15.10	Understanding the string representation of an articulation	88
15.11	Inspecting the LilyPond format of an articulation	88
15.12	Controlling whether an articulation appears above or below the staff	88
15.13	Getting and setting the name of an articulation	89
15.14	Copying articulations	89
15.15	Comparing articulations	90
15.16	Overriding attributes of the LilyPond script grob	90
16	Chords	91
16.1	Making chords from a LilyPond input string	91
16.2	Making chords from chromatic pitch numbers and duration	91
16.3	Getting all the written pitches of a chord at once	91
16.4	Getting the written pitches of a chord one at a time	91
16.5	Adding one pitch to a chord at a time	92
16.6	Adding many pitches to a chord at once	92
16.7	Deleting pitches from a chord	93
16.8	Formatting chords	93
16.9	Working with note heads	93
16.10	Working with empty chords	94
17	Containers	95
17.1	Creating containers	95
17.2	Inspecting music	95
17.3	Inspecting length	95
17.4	Inspecting duration	96
17.5	Adding one component to the end of a container	96
17.6	Adding many components to the end of a container	96
17.7	Finding the index of a component	96
17.8	Inserting a component by index	96
17.9	Removing a component by index	97
17.10	Removing a component by reference	97
17.11	Naming containers	97
17.12	Understanding { } and << >> in LilyPond	98
17.13	Understanding sequential and parallel containers	99
17.14	Changing sequential and parallel containers	100
17.15	Overriding containers	100
17.16	Overriding containers' contents	101
17.17	Removing container overrides	102
18	Durations	103
18.1	Introduction	103
18.2	Assignability	103
18.3	Prolation	104
18.4	Duration types	105
18.5	Duration initialization	108
18.6	LilyPond multipliers	109
18.7	Duration interfaces compared	110
19	Instrument marks	111
19.1	Creating instrument marks	111
19.2	Attaching instrument marks to a component	111

19.3	Getting the instrument mark attached to a component	111
19.4	Getting the instrument in effect for a component	112
19.5	Detaching instrument marks from a component one at a time	112
19.6	Detaching all instrument marks attached to a component at once	112
19.7	Inspecting the component to which an instrument mark is attached	113
19.8	Inspecting the instrument name of an instrument mark	113
19.9	Inspecting the short instrument name of an instrument mark	113
20	I/O	115
20.1	Reopening Abjad PDFs	115
20.2	Looking at LilyPond output	115
20.3	Looking at the LilyPond log	116
21	LilyPond command marks	117
21.1	Creating LilyPond command marks	117
21.2	Attaching LilyPond command marks to Abjad components	117
21.3	Getting the LilyPond command marks attached to an Abjad component	118
21.4	Detaching LilyPond command marks from components one at a time	118
21.5	Detaching all LilyPond command marks attached to a component at once	118
21.6	Inspecting the component to which a LilyPond command mark is attached	119
21.7	Getting and setting the command name of a LilyPond command mark	119
21.8	Copying LilyPond commands	119
21.9	Comparing LilyPond command marks	120
22	LilyPond comments	121
22.1	Creating LilyPond comments	121
22.2	Attaching LilyPond comments to leaves	121
22.3	Attaching LilyPond comments to containers	121
22.4	Getting the LilyPond comments attached to a component	122
22.5	Detaching LilyPond comments from a component one at a time	122
22.6	Detaching all LilyPond comments attached to a component at once	123
22.7	Inspecting the component to which a LilyPond comment is attached	123
22.8	Inspecting contents string of a LilyPond comment	123
23	LilyPond files	125
23.1	Making LilyPond files	125
23.2	Inspecting file output	125
23.3	Setting default paper size	125
23.4	Setting global staff size	126
24	Measures	127
24.1	Understanding measures in LilyPond	127
24.2	Understanding measures in Abjad	127
24.3	Creating measures	128
24.4	Working with dynamic measures	128
24.5	Adding music to dynamic measures	128
24.6	Removing music from dynamic measures	129
24.7	Setting the denominator of dynamic measures	129
24.8	Suppressing the meter of dynamic measures	129
24.9	Working with anonymous measures	129
24.10	Adding music to anonymous measures	130
24.11	Removing music from anonymous measures	130
25	Notes	131
25.1	Making notes from a string	131

25.2	Making notes from chromatic pitch number and duration	131
25.3	Getting the written pitch of notes	131
25.4	Changing the written pitch of notes	131
25.5	Getting the duration attributes of notes	132
25.6	Changing the written duration of notes	132
25.7	Overriding notes	133
25.8	Removing note overrides	134
26	Pitches	135
26.1	Creating pitches	135
26.2	Inspecting the name of a pitch	135
26.3	Inspecting the octave of a pitch	135
26.4	Working with pitch deviation	135
26.5	Sorting pitches	136
26.6	Comparing pitches	136
26.7	Converting one type of pitch to another	137
26.8	Converting pitches to pitch-classes	137
26.9	Copying pitches	137
26.10	Accidental abbreviations	138
26.11	Chromatic pitch numbers	138
26.12	Diatonic pitch numbers	139
26.13	Octave designation	140
26.14	Accidental spelling	140
27	Working with lists of numbers	143
28	Rests	145
28.1	Making rests from strings	145
28.2	Making rests from durations	145
28.3	Getting the duration attributes of rests	145
28.4	Changing the written duration of rests	146
29	Scores	147
29.1	Creating scores	147
29.2	Inspecting score music	147
29.3	Inspecting score length	147
29.4	Inspecting score duration	147
29.5	Adding one component to the bottom of a score	148
29.6	Finding the index of a score component	148
29.7	Removing a score component by index	148
29.8	Removing a score component by reference	149
29.9	Testing score containment	149
29.10	Naming scores	149
30	Spanners	151
30.1	Overriding spanners	151
30.2	Overriding the components to which spanners attach	152
30.3	Removing spanner overrides	152
31	Staves	155
31.1	Creating staves	155
31.2	Inspecting staff music	155
31.3	Inspecting staff length	155
31.4	Inspecting staff duration	155
31.5	Adding one component to the end of a staff	156

31.6	Adding many components to the end of a staff	156
31.7	Finding the index of a staff component	156
31.8	Removing a staff component by index	156
31.9	Removing a staff component by reference	157
31.10	Naming staves	157
31.11	Forcing context	157
32	Tuplets	159
32.1	Making a tuplet from a LilyPond input string	159
32.2	Making a tuplet from a list of other Abjad components	159
32.3	Understanding the interpreter display of a tuplet	159
32.4	Understanding the string representation of a tuplet	160
32.5	Inspecting the LilyPond format of a tuplet	160
32.6	Inspecting the music in a tuplet	160
32.7	Inspecting a tuplet’s leaves	160
32.8	Getting the length of a tuplet	161
32.9	Getting the duration attributes of a tuplet	161
32.10	Understanding rhythmic augmentation and diminution	161
32.11	Understanding binary and nonbinary tuplets	161
32.12	Adding one component to the end of a tuplet	162
32.13	Adding many components to the end of a tuplet	162
32.14	Finding the index of a component in a tuplet	162
32.15	Removing a tuplet component by index	162
32.16	Removing a tuplet component by reference	163
32.17	Overriding attributes of the LilyPond tuplet number grob	163
32.18	Overriding attributes of the LilyPond tuplet bracket grob	163
33	Voices	165
33.1	Making a voice from a LilyPond input string	165
33.2	Making a voice from a list of other Abjad components	165
33.3	Understanding the <code>repr</code> of a voice	165
33.4	Inspecting the LilyPond format of a voice	166
33.5	Inspecting the music in a voice	166
33.6	Inspecting a voice’s leaves	166
33.7	Getting the length of a voice	166
33.8	Getting the duration attributes of a voice	166
33.9	Adding one component to the end of a voice	167
33.10	Adding many components to the end of a voice	167
33.11	Finding the index of a component in a voice	167
33.12	Removing a voice component by index	168
33.13	Removing a voice component by reference	168
33.14	Naming voices	168
33.15	Changing the context of a voice	169
34	Codebase	171
34.1	How the Abjad codebase is laid out	171
34.2	Removing prebuilt versions of Abjad before you check out	171
34.3	Installing the development version	172
35	Docs	175
35.1	How the Abjad docs are laid out	175
35.2	Installing Sphinx	175
35.3	Removing old builds of the docs	176
35.4	Generating the Abjad API	176
35.5	Building the HTML docs	176

35.6	Building a PDF of the docs	177
35.7	Building a coverage report	178
35.8	Building other versions of the docs	179
35.9	Inserting images with <code>abjad-book</code>	179
35.10	Updating Sphinx	179
36	Tests	181
36.1	Automated regression?	181
36.2	Running the battery	181
36.3	Reading test output	182
36.4	Writing tests	182
36.5	Test files start with <code>test_</code>	182
36.6	Avoiding name conflicts	182
36.7	Updating <code>py.test</code>	182
36.8	Running <code>doctest</code> on the <code>tools</code> directory	183
37	Scripts	185
37.1	Searching the Abjad codebase with <code>abj-grep</code>	185
37.2	Removing old <code>*.pyc</code> files with <code>abj-rmpycs</code>	186
37.3	Updating your development copy of Abjad with <code>abj-update</code>	186
37.4	Counting lines of code with <code>count-source-lines</code>	186
37.5	Global search-and-replace with <code>replace-in-files</code>	186
37.6	Adding new development scripts	187
38	Using <code>abjad-book</code>	189
38.1	HTML with embedded Abjad	189
38.2	LaTeX with embedded Abjad	190
38.3	Using <code>abjad-book</code> on ReST documents	192
38.4	Using <code>[hide = True]</code>	192
39	Timing code	193
40	Profiling code	195
41	Memory consumption	197
42	Class attributes	199
43	Using slots	201
44	Coding standards	203
45	From Trevor and Víctor	207
46	Why MIDI is not enough	209
46.1	A very brief overview of MIDI	209
46.2	Limitations of MIDI from the point of view of score modeling	209
46.3	Written note durations vs. MIDI delta-times	210
46.4	Written note pitch vs. MIDI note-on	210
46.5	Conclusion	210
47	Why LilyPond is right for Abjad	211
47.1	Nested tuplets works out of the box	211
47.2	Broken tuplets work out of the box	211
47.3	Nonbinary meters work out of the box	212
47.4	Lilypond models the musical measure correctly	212

48 LilyPond template gallery	215
48.1 Default LilyPond layout	215
48.2 lagos.ly	215
48.3 oedo.ly	216
48.4 tangiers.ly	216
48.5 tirnaveni.ly	217
49 LilyPond text alignment	219
49.1 Default alignment	219
49.2 TextScript #'self-alignment-X	219
49.3 TextScript #'X-offset	220
50 Bibliography	221
51 Abjad API	223
51.1 Abjad API	223
Bibliography	2111
Index	2113

Abjad helps composers build up complex pieces of music notation in an iterative and incremental way. Use Abjad to create a symbolic representation of all the notes, rests, staves, tuplets, beams and slurs in any score. Because Abjad extends the Python programming language, you can use Abjad to make systematic changes to your music as you work. And because Abjad wraps the powerful LilyPond music notation package, you can use Abjad to control the typographic details of the symbols on the page.



Start here

ABJAD?

Abjad is an interactive software system designed to help composers build up complex pieces of music notation in an iterative and incremental way. Use Abjad to create a symbolic representation of all the notes, rests, staves, tuplets, beams and slurs in any score. Because Abjad extends the Python programming language, you can use Abjad to make systematic changes to your music as you work. And because Abjad wraps the powerful LilyPond music notation package, you can use Abjad to control the typographic details of the symbols on the page.

1.1 Abjad extends LilyPond

[LilyPond](#) is an open-source music notation package invented by Han-Wen Nienhuys and Jan Niewenhuizen and extended by an international team of developers and musicians. LilyPond differs from other music engraving programs in a number of ways. LilyPond separates musical content from page layout. LilyPond affords typographic control over almost everything. And LilyPond implements a powerfully correct model of the musical score.

You can start working with Abjad right away because Abjad creates LilyPond files for you automatically. But you will work with Abjad faster and more effectively if you understand the structure of the LilyPond files Abjad creates. For this reason we recommend new users spend a couple of days learning LilyPond first.

Start by reading about [text input](#) in LilyPond. Then work the [LilyPond tutorial](#). You can test your understanding of LilyPond by using the program to engrave of a Bach chorale. Use a grand staff and include slurs, fermatas and so on. Once you can engrave a chorale in LilyPond you'll understand the way Abjad works with LilyPond behind the scenes.

1.2 Abjad extends Python

[Python](#) is an open-source programming language invented by Guido van Rossum and further developed by a team of programmers working in many countries around the world. Python is used to provision servers, process text, develop distributed systems and do much more besides. The dynamic language and interpreter features of Python are similar to Ruby while the syntax of Python resembles C, C++ and Java.

To get the most out of Abjad you need to know (or learn) the basics of programming in Python. Abjad extends Python because it makes no sense to reinvent the wheel modern programming languages have developed to find, sort, store, model and encapsulate information. Abjad simply piggy-backs on the ways of doing these things that Python provides. So to use Abjad effectively you need to know the way these things are done in Python.

Start with the [Python tutorial](#). The tutorial is structured in 15 chapters and you should work through the first 12. This will take a day or two and you'll be able to use all the information you read in the Python tutorial in Abjad. If you're an experienced programmer you should skip chapters 1 - 3 but read 4 - 12. When you're done you can give yourself the equivalent of the chorale test suggested above. First open a file and define a couple of classes and functions in it. Then open a second file and write some code to first import and then do stuff with the classes and functions you

defined in the first file. Once you can easily do this without looking at the Python docs you'll be in a much better position to work with Abjad.

1.3 What next?

The most important parts of Abjad are the interlocking objects that structure the system. Read about the way Abjad models pitch, duration, leaves, containers, spanners and marks in the *Abjad reference manual*.

But note that important parts of the system are missing from the manual. The reason for this is that we completed the Abjad API months before we started the manual. This means that classes and functions you look up in the API may not yet be documented in the manual. The reference manual will eventually document all parts of the system. But until then check the API if the manual doesn't yet have what you need.

Once you understand the basics about how to work with Abjad you should spend some time with the *Abjad API*. The API documents all the functionality available in the system. Abjad comprises about 168,000 lines of code. About half of these implement the automated tests that check the correctness of Abjad. The rest of the code implements 39 packages comprising 221 classes and 1029 functions. All of these are documented in the API.

1.4 Mailing lists

As you begin working with Abjad please be in touch.

Questions, comments and contributions are welcomed from composers everywhere.

Questions or comments? Join the [abjad-user](#) list.

Want to contribute? Join the [abjad-devel](#) list.

INSTALLATION

2.1 Abjad depends on Python

You must have Python 2.5, 2.6 or 2.7 installed to run Abjad.

Abjad does not yet support the Python 3.x series of releases.

To check the version of Python installed on your computer type the following:

```
python --version
```

You can download different versions of Python at <http://www.python.org>.

2.2 Abjad depends on LilyPond

You must have LilyPond 2.12 or greater installed for Abjad to work properly.

You can download LilyPond at <http://www.lilypond.org>.

After you have installed LilyPond you should type the following to see if LilyPond is callable from your commandline:

```
lilypond --version
```

If LilyPond is not callable from your commandline you should add the location of the LilyPond executable to your PATH environment variable.

If you are new to working with the commandline you should use Google to get a basic introduction to editing your profile and setting environment variables.

2.3 Installing the current packaged version of Abjad with easy_install

There are different ways to install Python packages on your computer. One of the most direct ways is with `easy_install`.

If you have `easy_install` installed on your computer then you can install Abjad with this command:

```
sudo easy_install -U abjad
```

Python will install Abjad in the site packages directory on your computer and you'll be ready to start using the system.

If you do not have `easy_install` installed on your computer then you should follow the instructions below to install the current packaged version of Abjad from the Python Package Index.

2.4 Installing the current packaged version of Abjad from the Python Package Index

If you do not have `easy_install` installed on your computer you should follow these steps to install the current packaged version of Abjad from the Python Package Index:

1. Download the current release of Abjad from <http://pypi.python.org/pypi/Abjad>.
2. Unarchive the downloaded file. Under MacOS and Windows you can double click the archived file.
Under Linux execute the following command with `x.y` replaced by the current release of Abjad:

```
tar xzvf Abjad-x.y.tar.gz
```

3. Change into the directory created in step 2:

```
cd Abjad-x.y
```

4. Run the following under MacOS or Linux:

```
sudo python setup.py install
```

5. Or run this command under Windows after starting up a command shell with administrator privileges:

```
setup.py install
```

These commands will cause Python to install Abjad in your site packages directory. You'll then be ready to start using Abjad.

2.5 After install

When first run, Abjad creates an `.abjad` directory in your own `$HOME` directory. In `$HOME/.abjad` you will find the Abjad configuration file: `config.py`. Here you can tell Abjad about your preferred PDF file viewer, MIDI player, your preferred LilyPond language, etc. All relevant variables have defaults that you can change to suit your needs. In Linux, for example, you might want to set your `pdfviewer` to `evince` and your `MIDIplayer` to `tiMIDiTY`.

`config.py` is a regular Python file, so you should make sure the file follows Python syntax.

2.6 Note for Linux users

Abjad defaults to `xdg-open` to display PDF files using your default PDF viewer. Most Linux distributions now come with `xdg-utils` installed.

If you do not have `xdg-utils` installed, you can download it from <http://www.portland.freedsektop.org>.

Alternatively you can set the `pdfviewer` variable in `$HOME/.abjad/config` to your favorite PDF viewer.

VERSION HISTORY

3.1 Abjad 2.8

Released 2012-04-16. Built from r5418. Implements 306 public classes and 1037 functions totalling 178,000 lines of code.

Many documentation improvements appear in this release.

- A source link now accompanies all classes and functions in the API:

Source code for `abjad.tools.chordtools.arpeggiate_chord`

```
from abjad.tools.chordtools.Chord import Chord
from abjad.tools.decoratortools import requires

@requires(Chord)
def arpeggiate_chord(chord):
    '''.. versionadded:: 1.1

    Arpeggiate `chord`:

        abjad> chord = Chord("<c' d'' ef''>8")

    ::

        abjad> chordtools.arpeggiate_chord(chord)
        [Note("c'8"), Note("d''8"), Note("ef''8")]

    Arpeggiated notes inherit `chord` written duration.

    Arpeggiated notes do not inherit other `chord` attributes.

    Return list of newly constructed notes.

    .. versionchanged:: 2.0
        renamed ``chordtools.arpeggiate()`` to
        ``chordtools.arpeggiate_chord()``.

    from abjad.tools.notetools.Note import Note

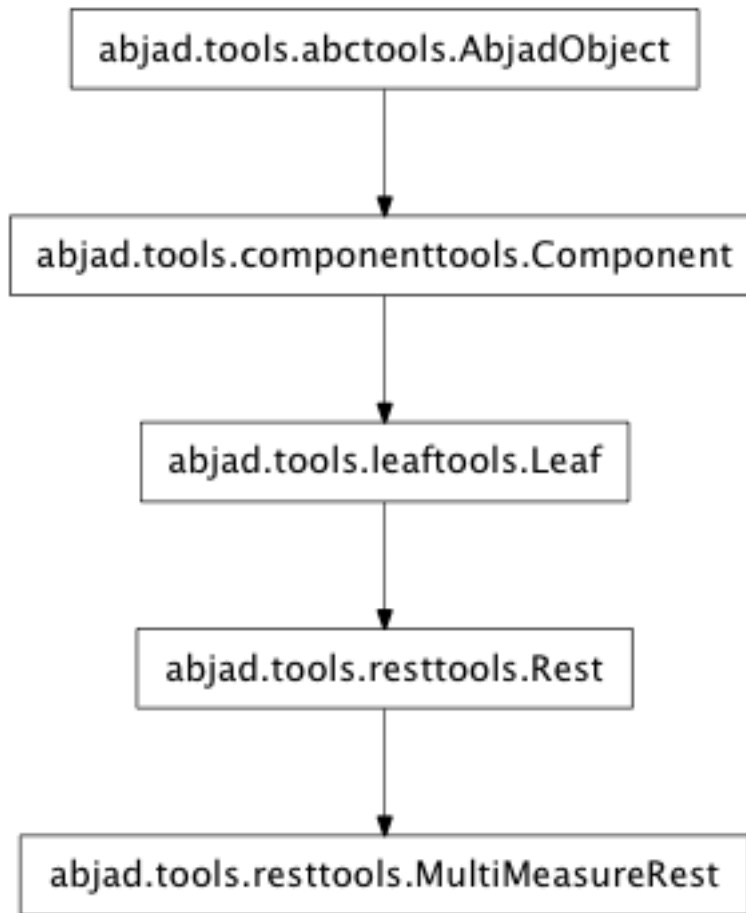
    result = []
    chord_written_duration = chord.written_duration
    for pitch in chord.written_pitches:
        result.append(Note(pitch, chord_written_duration))

    return result
```

[docs]

- All parts of the Abjad codebase are now viewable through the HTML version of the API.

- Inheritance diagrams now accompany all classes:



- Inherited attributes now appear in the API entry of each class.
- Added new `documentationtools` package:

```
documentationtools.APICrawler
documentationtools.AbjadAPIGenerator
documentationtools.ClassCrawler
documentationtools.ClassDocumenter
documentationtools.Documenter
documentationtools.FunctionCrawler
documentationtools.FunctionDocumenter
documentationtools.InheritanceGraph
documentationtools.ModuleCrawler
documentationtools.Pipe
```

The package houses custom code to build Abjad documentation.

Added the new `constrainttools` API.

- This release of the `constrainttools` package implements the following classes:

```
constrainttools.AbsoluteIndexConstraint
constrainttools.Domain
constrainttools.FixedLengthStreamSolver
constrainttools.GlobalConstraint
constrainttools.GlobalCountsConstraint
```

```

constrainttools.GlobalReferenceConstraint
constrainttools.RelativeCountsConstraint
constrainttools.RelativeIndexConstraint
constrainttools.VariableLengthStreamSolver

```

- **Example:**

```

abjad> from abjad.tools.constraintstools import *

abjad> domain = Domain([1, 2, 3, 4], 4)

abjad> all_unique = GlobalCountsConstraint(lambda x: all([y == 1 for y in x.values()]))
abjad> max_interval = RelativeIndexConstraint([0, 1], lambda x, y: abs(x - y) < 3)
abjad> solver = FiniteStreamSolver(domain, [all_unique, max_interval])

abjad> for solution in solver: print solution
...
(1, 2, 3, 4)
(1, 2, 4, 3)
(1, 3, 2, 4)
(1, 3, 4, 2)
(2, 1, 3, 4)
(2, 4, 3, 1)
(3, 1, 2, 4)
(3, 4, 2, 1)
(4, 2, 1, 3)
(4, 2, 3, 1)
(4, 3, 1, 2)
(4, 3, 2, 1)

```

- The `constrainttools` package is considered unstable and will be subject to changes in the next releases of Abjad.

Added octave-transposition mapping model.

- This version of the system contains the following classes:

```

pitchtools.OctaveTranspositionMapping
pitchtools.OctaveTranspositionMappingComponent
pitchtools.OctaveTranspositionMappingInventory

```

- Octave-transposition mappings specify a way to maybe pitches from one registral space to another.
- Use octave-transposition mappings as input to `pitchtools.transpose_chromatic_pitch_number_ty_octave_tr`

Many Abjad classes are now implemented as abstract base classes.

- Abstract base classes provide functionality to child subclasses.
- Abstract base classes can not be instantiated directly.
- The Abjad API now lists abstract classes and concrete classes separately.
- See <http://docs.python.org/library/abc.html> for a description of ABCs in Python.

Added the new `abctools` package to house abstract classes that are core to the Abjad object model.

- This version of the package contains the following classes:

```

abctools.AbjadObject
abctools.AttributeEqualityAbjadObject
abctools.ImmutableAbjadObject
abctools.SortableAttributeEqualityAbjadObject

```

- All Abjad classes now inherit from `AbjadObject`.

Added object inventories for several classes.

- This release contains inventories for the following classes:

```
contexttools.ClefMarkInventory
contexttools.TempoMarkInventory
instrumenttools.InstrumentInventory
markuptools.MarkupInventory
pitchtools.OctaveTranspositionMappingInventory
pitchtools.PitchRangeInventory
scoretools.PerformerInventory
```

- Object inventories model ordered collections of system objects.

Add the new `datastructuretools` package.

- This version of the package includes the following classes:

```
datastructuretools.Digraph
datastructuretools.ImmutableDictionary
datastructuretools.ObjectInventory
```

- Use `datastructuretools.Digraph` to detect cycles in any collection of hashable objects:

```
abjad> from abjad.tools.datastructuretools import Digraph

abjad> edges = [('a', 'b'), ('a', 'c'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')]
abjad> digraph = Digraph(edges)
abjad> digraph
Digraph(edges=[('a', 'c'), ('a', 'b'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')])

abjad> digraph.root_nodes
('a',)
abjad> digraph.terminal_nodes
('b', 'f')
abjad> digraph.cyclic_nodes
('c', 'd', 'e')
abjad> digraph.is_cyclic
True
```

- Use `datastructuretools.ObjectInventory` as the base class for an ordered collection of system objects.
- Object inventories inherit from `list` and are mutable.
- Object inventories extend `append()`, `extend()` and `__contains__()` to allow token input.

Added new `wellformednesstools` package.

- This version of the package implements the following classes:

```
wellformednesstools.BeamedQuarterNoteCheck
wellformednesstools.DiscontiguousSpannerCheck
wellformednesstools.DuplicateIdCheck
wellformednesstools.EmptyContainerCheck
wellformednesstools.IntermarkedHairpinCheck
wellformednesstools.MisduratedMeasureCheck
wellformednesstools.MisfilledMeasureCheck
wellformednesstools.MispitchedTieCheck
wellformednesstools.MisrepresentedFlagCheck
wellformednesstools.MissingParentCheck
```



```
wellformednesstools.NestedMeasureCheck
wellformednesstools.OverlappingBeamCheck
wellformednesstools.OverlappingGlissandoCheck
wellformednesstools.OverlappingOctavationCheck
wellformednesstools.ShortHairpinCheck
```

- The classes check different aspects of score well-formedness.
- To call these classes use `componenttools.is_well_formed_component()` or `componenttools.tabulate_well_formedness_violations_in_expr()`.

Added new `decoratortools` package.

- This version of the package contains only the `requires` decorator.
- The `requires` decorator will be used in later versions of Abjad to specify the input and output types of functions explicitly.
- This will help in the construction of function- and class-population tools.

Added new `scoretemplatetools` package.

- This version of the package implements the following classes:

```
scoretemplatetools.StringQuartetScoreTemplate
scoretemplatetools.TwoStaffPianoScoreTemplate
```

- Example:

```
abjad> from abjad.tools import scoretemplatetools

abjad> template = scoretemplatetools.StringQuartetScoreTemplate()
abjad> score = template()

abjad> score
Score-"String Quartet Score"<<1>>

abjad> f(score)
\context Score = "String Quartet Score" <<
  \context StaffGroup = "String Quartet Staff Group" <<
    \context Staff = "First Violin Staff" {
      \clef "treble"
      \context Voice = "First Violin Voice" {
      }
    }
    \context Staff = "Second Violin Voice" {
      \clef "treble"
    }
    \context Staff = "Viola Staff" {
      \clef "alto"
    }
    \context Staff = "Cello Staff" {
      \clef "bass"
    }
  >>
>>
```

- Class usage follows a two-step initialize-then-call pattern.

Added new `rhythmtreertools` package for parsing IRCAM-like RTM syntax.

- This version of the package implements the following function:

```
rhythmtreetools.parse_rtm_syntax.parse_rtm_syntax()
```

- **Example:**

```
abjad> from abjad.tools.rhythmtreetools import parse_rtm_syntax

abjad> rtm = '(1 (1 (1 (1 1)) 1))'
abjad> result = parse_rtm_syntax(rtm)
abjad> result
FixedDurationTuplet(1/4, [c'8, c'16, c'16, c'8])
```

- Use the `rhythmtreetools` package to turn nested lists of numbers into Abjad tuplets.

Added new `timetokentools` package.

- This version of the package contains the following concrete classes:

```
timetokentools.NoteFilledTimeTokenMaker
timetokentools.OutputBurnishedSignalFilledTimeTokenMaker
timetokentools.OutputIncisedNoteFilledTimeTokenMaker
timetokentools.OutputIncisedRestFilledTimeTokenMaker
timetokentools.RestFilledTimeTokenMaker
timetokentools.SignalFilledTimeTokenMaker
timetokentools.TokenBurnishedSignalFilledTimeTokenMaker
timetokentools.TokenIncisedNoteFilledTimeTokenMaker
timetokentools.TokenIncisedRestFilledTimeTokenMaker
```

- The `timetokentools` package implements a family of related rhythm-making classes.
- Class usage follows a two-step initialize-then-call pattern.

Added new classes to `instrumenttools`.

- Added human voice classes:

```
instrumenttools.BaritoneVoice
instrumenttools.BassVoice
instrumenttools.ContraltoVoice
instrumenttools.MezzoSopranoVoice
instrumenttools.SopranoVoice
instrumenttools.TenorVoice
```

Added new time-interval tree functionality:

- Extended `TimeIntervalTree` with the following public methods:

```
scale_by_rational()
scale_to_rational()
shift_by_rational()
shift_to_rational()
split_at_rationals()
```

- These methods allow time-interval trees to behave more similar to time-intervals.

All score components are now public.

- The following classes are now publically available for the first time:

```
componenttools.Component
contexttools.Context
leaftools.Leaf
```

Further new functionality:

- Added the `marktools.BendAfter` class to model LilyPond's `\bendAfter` command:

```
abjad> n = Note(0, 1)
abjad> marktools.BendAfter(8)(n)
BendAfter(8.0)(c'1)
abjad> f(n)
c'1 - \bendAfter #'8.0
```

- Added public pair property to `contexttools.TimeSignatureMark`:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 16))
abjad> time_signature.pair
(3, 16)
```

- Added `is_hairpin_token()` to `spannertools.HairpinSpanner` class.

Hairpin tokens are triples of the form `(x, y, z)` with dynamic tokens `x`, `y` and hairpin shape string `z`. For example `('p', '<', 'f')`.

- Added `resttools.replace_leaves_in_expr_with_rests()`.
- Added `leaftools.replace_leaves_in_expr_with_parallel_voices()`.
- Added `leaftools.replace_leaves_in_expr_with_named_parallel_voices()`.

Use the functions listed above to replace leaves in an expression with parallel voices containing copies of those leaves in both voices. This is useful for generating stemmed-glissandi structures.

- Added `contexttools.list_clef_names()`:

```
abjad> contexttools.list_clef_names()
['alto', 'baritone', 'bass', 'mezzosoprano', 'percussion', 'soprano', 'treble']
```

- Added `find-slots-implementation-inconsistencies` development script.

Changes to end-user functionality:

- Changed `intervaltreertools` to `timeintervaltools`.
- Changed `contexttools.Context.context` to `contexttools.Context.context_name`.
- Calling `bool(Container())` on empty containers now returns `false`. The previous behavior of the system was to return `true`. The new behavior better conforms to the Python iterable interface.
- Moved `abjad/docs/scr/make-abjad-api` to `abjad/scr/make-abjad-api`.

3.2 Abjad 2.7

Released 2012-02-27. Built from r5100. Implements 221 public classes and 1029 functions totalling 168,000 lines of code.

- Added `lilypondparsertools.LilyPondParser` class, which parses a subset of LilyPond input syntax:

```
abjad> from abjad.tools.lilypondparsertools import LilyPondParser
abjad> parser = LilyPondParser()
abjad> input = r"\new Staff { c'4 ( d'8 e' fs'2) \fermata }"
abjad> result = parser(input)
abjad> f(result)
\new Staff {
  c'4 (
  d'8
  e'8
```

```

    fs'2 -\fermata )
}

```

LilyPondParser defaults to English note names, but any of the other languages supported by LilyPond may be used:

```

abjad> parser = LilyPondParser('nederlands')
abjad> input = '{ c des e fis }'
abjad> result = parser(input)
abjad> f(result)
{
    c4
    df4
    e4
    fs4
}

```

Briefly, LilyPondParser understands theses aspects of LilyPond syntax:

- Notes, chords, rests, skips and multi-measure rests
- Durations, dots, and multipliers
- All pitchnames, and octave ticks
- Simple markup (i.e. `c'4 ^ "hello!"`)
- Most articulations
- Most spanners, including beams, slurs, phrasing slurs, ties, and glissandi
- Most context types via `\new` and `\context`, as well as context ids (i.e. `\new Staff = "foo" { }`)
- Variable assignment (i.e. `global = { \time 3/4 } \new Staff { \global }`)
- Many music functions: `- \acciaccatura - \appoggiatura - \bar - \breathe - \clef - \grace - \key - \transpose - \language - \makeClusters - \mark - \oneVoice - \relative - \skip - \slashedGrace - \time - \times - \transpose - \voiceOne, \voiceTwo, \voiceThree, \voiceFour`

LilyPondParser currently **DOES NOT** understand many other aspects of LilyPond syntax:

- `\markup`
 - `\book, \bookpart, \header, \layout, \midi and \paper`
 - `\repeat and \alternative`
 - Lyrics
 - `\chordmode, \drummode or \figuremode`
 - Property operations, such as `\override, \revert, \set, \unset, and \once`
 - Music functions which generate or extensively mutate musical structures
 - Embedded Scheme statements (anything beginning with `#`)
- Added `iotools.p()`, for conveniently parsing LilyPond syntax:

```

abjad> result = p(r"\new Staff { c'4 d e f }")
abjad> f(result)
\new Staff {
    c'4

```

```
d4
e4
f4
}
```

- Added `schemetools.Scheme`, as a more robust replacement for nearly all other `schemetools` classes:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> print Scheme(True).format
##t
abjad> print Scheme('a', 'list', 'of', 'strings').format
#(a list of strings)
abjad> print Scheme(('simulate', 'a', 'vector'), quoting="'#").format
##(simulate a vector)
abjad> print Scheme('a', ('nested', ('data', 'structure'))).format
#(a (nested (data structure)))
```

- Removed deprecated `schemetools` classes:

- `SchemeBoolean`
- `SchemeFunction`
- `SchemeNumber`
- `SchemeString`
- `SchemeVariable`

In all cases, simply use `schemetools.Scheme` instead.

- Reimplemented `MarkupCommand`.

The new implementation is initialized from a command-name, and a variable-size list of arguments. Arguments which are lists or tuples will be enclosed in curly-braces:

```
abjad> from abjad.tools.markuptools import MarkupCommand
abjad> bold = MarkupCommand('bold', ['two', 'words'])
abjad> rotate = MarkupCommand('rotate', 60, bold)
abjad> triangle = MarkupCommand('triangle', False)
abjad> concat = MarkupCommand('concat', ['one word', rotate, triangle])
abjad> print concat.format
\concat { #"one word" \rotate #60 \bold { two words } \triangle ##f }
```

- Added `contexttools.TempoMarkInventory`, which models an ordered list of tempo marks:

```
abjad> contexttools.TempoMarkInventory([('Andante', Duration(1, 8), 72), ('Allegro', Duration(1, 8), 120)])
TempoMarkInventory([TempoMark('Andante', Duration(1, 8), 72), TempoMark('Allegro', Duration(1, 8), 120)])
```

Inherits from `list`. Allows initialization, `append` and `extend` on tempo mark tokens.

- Added new `pitchtools.PitchRangeInventory` class.

The class acts as an ordered list of `PitchRange` objects.

The purpose of the class is to model something like palettes of different pitches available in all part of a score:

```
abjad> pitchtools.PitchRangeInventory(['[C3, C6]', '[C4, C6]'])
PitchRangeInventory([PitchRange('[C3, C6]'), PitchRange('[C4, C6]')])
```

The class inherits from `list`.

- Added `sequencetools.all_are_pairs()` predicate:

```
abjad> from abjad.tools.sequencetools import all_are_pairs
abjad> all_are_pairs([(1, 2), (3, 4), (5, 6)])
True
```

- Added `sequencetools.all_are_pairs_of_types()` predicate:

```
abjad> from abjad.tools.sequencetools import all_are_pairs_of_types
abjad> all_are_pairs_of_types([('a', 1.4), ('b', 2.3), ('c', 1.5)], str, float)
True
```

- Added `iotools.is_underscore_delimited_lowercase_file_name_with_extension()` string predicate:

```
abjad> iotools.is_underscore_delimited_lowercase_file_name_with_extension('foo_bar.blah')
True
```

- Added `iotools.is_underscore_delimited_file_name()` string predicate.

Returns true on any underscore-delimited lowercase string.

Also returns true on an underscore-delimited lowercase string terminated with an extension.

```
abjad> iotools.is_underscore_delimited_lowercase_file_name('foo_bar.py')
True
```

```
abjad> iotools.is_underscore_delimited_lowercase_file_name('foo_bar')
True
```

- Added `ImpreciseTempoError` to exceptions.
- Added `LilyPondParserError` to exceptions.
- Added `scr/devel/fix-test-cases`. The script is a two-line wrapper around the following other two scripts:

- `scr/devel/fix-test-case-names`
- `scr/devel/fix-test-case-numbers`

- Extended `Container` to use `LilyPondParser` to parse input strings.

- Extended `contexttools.InstrumentMark`, `scoretools.Performer` and `markuptools.Markup` with `__hash__` equality.

Now, if two instances compare equally (via `==`), their hashes also compare equally, allowing for more intuitive use of these classes as dictionary keys.

- Extended `contexttools.TempoMark` with textual indications and tempo ranges You may instantiate as normal, or in some new combinations:

```
abjad> from abjad.tools.contexttools import TempoMark
abjad> t = TempoMark('Langsam', Duration(1, 4), (52, 57))
abjad> t = TempoMark('Langsam')
abjad> t = TempoMark((1, 4), (52, 57))
```

In addition to its new read/write “textual_indication” attribute, `TempoMark` now also exposes a read-only “is_imprecise” property, which returns `True` if the mark cannot be expressed simply as `duration=units_per_minute`. Arithmetic operations on `TempoMarks` will now raise `ImpreciseTempoErrors` if any mark involved is imprecise.

- Extended tempo marks to be able to initialize from ‘tempo mark tokens’. A tempo mark token is a length-2 or length-3 tuple of tempo mark arguments.

- Extended tempo mark with `is_tempo_mark_token()` method:

```
abjad> tempo_mark = contexttools.TempoMark(Duration(1, 4), 72)
abjad> tempo_mark.is_tempo_mark_token((Duration(1, 4), 84))
True
```

- Extended case-testing `iotools` string predicates to allow digits.

Functions changed:

- `iotools.is_space_delimited_lowercase_string`
- `iotools.is_underscore_delimited_lowercase_file_name`
- `iotools.is_lowercamelcase_string`
- `iotools.is_uppercamelcase_string`
- `iotools.is_underscore_delimited_lowercase_string`
- `iotools.is_underscore_delimited_lowercase_file_name_with_extension`

- Extended `lilypondfiletools._NonattributedBlock` with `is_formatted_when_empty` read-write property. `lilypondfiletools.ScoreBlock` no longer formats when empty, by default.
- Extended `marktools.BarLine` with `format_slot` keyword.
- Extended `pitchtools.PitchRange` class with read-only `pitch_range_name` and `pitch_range_name_markup` attributes.
- Extended `scoretools.InstrumentationSpecifier` with read-only `performer_name_string` attribute.
- Extended all `spannertools.Beam-`, `Slur-` and `Hairpin-` related spanner classes, as well as `tietools.TieSpanner` with an optional `direction` keyword:

```
abjad> c = Container("c'4 d'4 e'4 f'4")
abjad> spanner = spannertools.SlurSpanner(c[:], 'up')
abjad> f(c)
{
    c'4 ^ (
    d'4
    e'4
    f'4 )
}
```

The direction options are exactly the same as for Articulation and Markup: `'up'`, `'^'`, `'down'`, `'_'`, `'neutral'`, `'-'` and `None`.

- Extended `tonalitytools.Scale` with `create_named_chromatic_pitch_set_in_pitch_range()` method.
- Changed `tuplettools.FixedDurationTuplet.multiplier` to return fraction instead of duration.
- Renamed attributes, methods and functions throughout `intervalreetools`:
 - `centroid` => `center` (except where a weighted mean is actually used)
 - `high` => `stop`
 - `high_min` => `earliest_stop`
 - `high_max` => `latest_stop`
 - `low` => `start`
 - `low_min` => `earliest_start`

- low_max => latest_start
- magnitude => duration

This both clarifies the API, and prevents shadowing of Python's builtin `min()` and `max()`.

- Renamed `marktools.Articulation.direction_string` => `marktools.Articulation.direction`.
- Renamed `markuptools.Markup.direction_string` => `'markuptools.Markup.direction`.
- Renamed `tuplettools.Tuplet.ratio` to `tuplettools.Tuplet.ratio_string`.
- Renamed `scr/devel/find-nonalphabetized-method-names` to `scr/devel/find-nonalphabetized-class-attributes`.
- Improved `scr/devel/find-nonalphabetized-methods`.
- Updated literature examples to match API changes.
- Removed ancient `stafftools.make_invisible_staff()`.
- Added `text_editor` key to user config dictionary (in `~/abjad/config.py`).
- Improved `__repr__` strings of `tonalitytools.Mode` and `tonalitytools.Scale`.
- `contexttools.TempoMark.__repr__` now shows `__repr__` version of duration instead of string version of duration.
- `scr/devel/abj-grp` no longer excludes lines of code that include the string `'svn'`.

3.3 Abjad 2.6

Released 2012-01-29. Built from r4979. Implements 197 public classes and 941 public functions totalling 153,000 lines of code.

- Added top-level `decorators` directory with `requires` decorator. The `requires` decorator renders the following two function definitions equivalent:

```
from abjad.tools.decoratortools import requires

@requires(int)
def foo(x):
    return x ** 2

def foo(x):
    assert isinstance(x, int)
    return x ** 2
```

- Added new classes to `scoretools`:

```
scoretools.InstrumentationSpecifier
scoretools.Performer
```

- Added `scoretools.list_performer_names()`:

```
abjad> for name in scoretools.list_performer_names()[:10]:
...     name
...
'accordionist'
'bassist'
'bassoonist'
'cellist'
```



```
'clarinetist'
'flutist'
'guitarist'
'harpist'
'harpsichordist'
'hornist'
```

- Added `scoretools.list_primary_performer_names()`.
- Added `measuretools.measure_to_one_line_input_string()`:

```
abjad> measure = Measure((3, 4), "c4 d4 e4")
```

```
abjad> measure
Measure(3/4, [c4, d4, e4])
```

```
abjad> measuretools.measure_to_one_line_input_string(measure)
"Measure((3, 4), 'c4 d4 e4')"
```

- Added new classes to `instrumenttools`:

```
SopraninoSaxophone
SopranoSaxophone
AltoSaxophone
TenorSaxophone
BaritoneSaxophone
BassSaxophone
ContrabassSaxophone
```

```
ClarinetInA
```

```
AltoTrombone
BassTrombone
```

```
Harpsichord
```

- Added known untuned percussion:

```
abjad> for name in instrumenttools.UntunedPercussion.known_untuned_percussion[:10]:
...     print name
...
agogô
anvil
bass drum
bongo drums
cabasa
cajón
castanets
caxixi
claves
conga drums
```

- Added `_Instrument.get_default_performer_name()`:

```
abjad> bassoon = instrumenttools.Bassoon()
```

```
abjad> bassoon.get_default_performer_name()
'bassoonist'
```

- Added `_Instrument.get_performer_names()`:

```
abjad> bassoon.get_performer_names()
['instrumentalist', 'reed player', 'double reed player', 'bassoonist']
```

- Added read / write `_Instrument.pitch_range`:

```
abjad> marimba.pitch_range = (-24, 36)
abjad> marimba.pitch_range
PitchRange(' [C2, C7]')
```

- Added read-only `_Instrument.traditional_pitch_range`:

```
abjad> marimba = instrumenttools.Marimba()
abjad> marimba.traditional_pitch_range
PitchRange(' [F2, C7]')
```

- Added `instrumenttools.list_instruments()`:

```
abjad> for instrument_name in instrumenttools.list_instrument_names()[:10]:
...     instrument_name
...
'accordion'
'alto flute'
'alto saxophone'
'alto trombone'
'clarinet in B-flat'
'baritone saxophone'
'bass clarinet'
'bass flute'
'bass saxophone'
'bass trombone'
```

- Added other functions to `instrumenttools`:

```
instrumenttools.list_primary_instrument_names()
instrumenttools.list_secondary_instrument_names()
```

- Added new class to `lilypondfiletools`:

```
ContextBlock
```

- Added `pitchtools.is_symbolic_pitch_range_string()`:

```
abjad> pitchtools.is_symbolic_pitch_range_string(' [A0, C8]')
True
```

- Added `pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name()`:

```
abjad> pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name('A#4')
"as' "
```

- Added `pitchtools.symbolic_accidental_string_to_alphabetic_accidental_string_abbreviation`:

```
abjad> pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string('tqs')
'#+'
```

- Added other new functions to `pitchtools`:

```
pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string()
pitchtools.is_symbolic_accidental_string()
pitchtools.is_pitch_class_octave_number_string()
```

- Added `iotools.string_to_strict_directory_name()`:

```
abjad> iotools.string_to_strict_directory_name('Déja vu')
'deja_vu'
```

- Added `iotools.strip_diacritics_from_binary_string()`:

```
abjad> binary_string = 'Dvořák'
abjad> iotools.strip_diacritics_from_binary_string(binary_string)
'Dvorak'
```

- Added other new functions to `iotools`:

```
iotools.capitalize_string_start()
iotools.is_space_delimited_lowercamelcase_string()
iotools.is_underscore_delimited_lowercamelcase_package_name()
iotools.is_underscore_delimited_lowercamelcase_string()
iotools.is_lowercamelcase_string()
iotools.is_uppercamelcase_string()
iotools.space_delimited_lowercase_to_uppercamelcase()
iotools.uppercamelcase_to_space_delimited_lowercase()
iotools.uppercamelcase_to_underscore_delimited_lowercase()
```

- Added new functions to `mathtools`:

```
mathtools.is_positive_integer_power_of_two()
mathtools.is_integer_equivalent_expr()
```

- Added sequence type-checking predicates:

```
chordtools.all_are_chords()
containertools.all_are_containers()
durationtools.all_are_duration_tokens()
durationtools.all_are_durations()
gracetools.all_are_grace_containers()
leaftools.all_are_leaves()
markuptools.all_are_markup()
measuretools.all_are_measures()
notetools.all_are_notes()
pitcharraytools.all_are_pitch_arrays()
pitchtools.all_are_named_chromatic_pitch_tokens()
resttools.all_are_rests()
scoretools.all_are_scores()
sievetools.all_are_residue_class_expressions()
skiptools.all_are_skips()
spannertools.all_are_spanners()
stafftools.all_are_staves()
tupletools.all_are_tuplets()
```

- Extended `NamedChromaticPitch` to allow initialization from pitch-class / octave number strings:

```
abjad> pitchtools.NamedChromaticPitch('C#2')
NamedChromaticPitch('cs,')
```

- Extended `PitchRange` to allow initialization from symbolic pitch range strings:

```
abjad> pitchtools.PitchRange('[A0, C8]')
PitchRange('[A0, C8]')
```

- Extended `PitchRange` to allow initialization from pitch-class / octave number strings:

```
abjad> pitchtools.PitchRange('A0', 'C8')
PitchRange('[A0, C8]')
```

- Extended `leaftools.is_bar_line_crossing_leaf()` to work when no explicit time signature mark is found.
- Extended Markup to be able to function as a top-level `LilyPondFile` element.
- Extended instruments with `is_primary` and `is_secondary` attributes.
- Extended instruments with `instrument_name` and `instrument_name_markup` attributes.
- Extended instruments with `short_instrument_name` and `short_instrument_name_markup` attributes.
- Extended `iotools.write_expr_to_ly()` and `iotools.write_expr_to_pdf()` with `'tagline'` keyword.
- Extended `replace-in-files` script to skip `.text`, `.ly` and `.txt` files.
- Renamed `Accidental.symbolic_string` to `Accidental.symbolic_accidental_string`.
- Renamed `Accidental.alphabetic_string` to `Accidental.alphabetic_accidental_abbreviation`.
- Fixed bug in `iotools.play()`.
- Fixed bug in `quantizationtools` regarding quantizing a stream of `QEvents` directly.

3.4 Abjad 2.5

Released 2011-09-22. Built from r4803.

- Added `get_leaf_in_expr_with_minimum_prolated_duration()` function to `leaftools`.
- Added `get_leaf_in_expr_with_maximum_prolated_duration()` function to `leaftools`.
- Added `are_relatively_prime()` function to `mathtools`.
- Added `CyclicTree` class to `sequencetools`.
- Added `get_next_n_nodes_at_level(n, level)` method to `sequencetools.Tree`.
- Extended `spanners` to sort by `repr`.
- Renamed `lilyfiletools` to `lilypondfiletools`.
- Renamed `lilyfiletools.LilyFile` to `lilypondfiletools.LilyPondFile`.
- Renamed `lilyfiletools.make_basic_lily_file()` to `lilypondfiletools.make_basic_lilypond_file`.

Note that the three renames change user syntax. Composers working with the `lilypondfiletools` module should update their score code.

3.5 Abjad 2.4

Released 2011-09-12. Built from r4769.

- Added Mozart Musikalisches Wuerfelspiel.

Ein Musikalisches Wuerfelspiel

W. A. Mozart (maybe?)



- Added new `Tree` class to `sequencetools` to work with sequences whose elements have been grouped into arbitrarily many levels of containment.
- Added new `BarLine` class to `marktools` package.
- Added new `HorizontalBracketSpanner` to `spannertools` package.
- Improved `schemetools.SchemePair` handling.
- Extended `LilyPondFile` blocks with double underscore-delimited attributes.

3.6 Abjad 2.3

Released 2011-09-04. Built from r4747.

Filled out the API for working with marks:

```
marktools.attach_articulations_to_components_in_expr()
marktools.detach_articulations_attached_to_component()
marktools.get_articulations_attached_to_component()
marktools.get_articulation_attached_to_component()
marktools.is_component_with_articulation_attached()
```

These five type of functions are now implemented for the following marks:

```
marktools.Annotation
marktools.Articulation
marktools.LilyPondCommandMark
marktools.LilyPondComment
marktools.StemTremolo
```

The same type of functions are likewise implemented for the following context marks:

```
contexttools.ClefMark
contexttools.DynamicMark
contexttools.InstrumentMark
contexttools.KeySignatureMark
contexttools.StaffChangeMark
contexttools.TempoMark
contexttools.TimeSignatureMark
```

- Extended `Container.extend()` to allow for LilyPond input strings. You can now say `container.extend("c'4 d'4 e'4 f'4")`.

- Added public `parent` attribute to all components. You can now say `note.parent`. The attribute is read-only.
- Added `cfgtools.list_package_dependency_version()`.
- Added `py.test` and Sphinx dependencies to the Abjad package.
- Added LilyPond command mark chapter to reference manual
- Renamed `cfgtools` to `configurationtools`.
- Renamed `durtools` to `durationtools`.
- Renamed `metertools` to `timesignaturetools`.
- Renamed `seqtools` to `sequencetools`.
- Renamed `Mark.attach_mark()` to `Mark.attach()`.
- Renamed `Mark.detach_mark()` to `Mark.detach()`.
- Renamed `marktools.Comment` to `marktools.LilyPondComment`. This matches `marktools.LilyPondCommandMark`.
- Removed `contexttools.TimeSignatureMark(3, 8)` initialization. You must now say `contexttools.TimeSignatureMark((3, 8))` instead. This parallels the initialization syntax for rests, skips and measures.

3.7 Abjad 2.2

Released 2011-08-30. Built from r4677.

- Added articulations chapter to reference manual.
- Reordered the way in which Abjad determines the value of the `HOME` environment variable.
- Updated `scr/devel/replace-in-files` to avoid image files.
- Updated `iotools.log()` to call operating-specific text editor.

3.8 Abjad 2.1

Released 2011-08-21. Built from r4655.

- Updated instrument mark `repr` to display target context when instrument mark is attached.
- Extended `scr/abj` and `scr/abjad` to display Abjad version and revision numbers on startup.

3.9 Abjad 2.0

Released 2011-08-17. Built from r4638.

Abjad 2.0 is the first public release of Abjad in more than two years. The new release of the system more than doubles the number of classes, functions and packages available in Abjad.

- The API has been cleaned up and completely reorganized. Features have been organized into a collection of 39 different libraries:

cfgtools/	instrumenttools/	mathtools/	resttools/	tempotools/
chordtools/	intervaltreertools/	measuretools/	schemetools/	threadtools/
componenttools/	iotools/	metertools/	scoretools/	tietools/
containertools/	layouttools/	musicxmltools/	seqtools/	tonalitytools/
contexttools/	leaftools/	notetools/	sievetools/	tuplettools/
durtools/	lilyfiletools/	pitcharraytools/	skiptools/	verticalitytools/
gracetools/	marktools/	pitchtools/	spannertools/	voicetools/
importtools/	markuptools/	quantizationtools/	stafftools/	

- The name of almost every function in the public API has been changed to better indication what the function does. While this has the effect of making Abjad 2.0 largely non-backwards compatible with code written in Abjad 1.x, the longer and much more explicit function names in Abjad 2.0 make code used to structure complex scores dramatically easier to maintain and understand.
- The `contexttools`, `instrumenttools`, `intervaltreertools`, `lilyfiletools`, `marktools`, `pitcharraytools`, `quantizationtools`, `sievetools`, `tonalitytools` and `verticalitytools` packages are completely new.
- The classes implemented in the `contexttools` and `marktools` packages provide an object-oriented interfaces to clefs, time signatures, key signatures, articulations, tempo marks and other symbols stuck to the outside of the hierarchical score tree. The classes implemented in `contexttools` and `marktools` model information outside the score tree much the way that the classes implemented in `spannertools` implement object-oriented interfaces to beams, brackets, hairpins, glissandi and other line-like symbols.
- The `instrumenttools` package provides an object-oriented model of most of the conventional instruments of the orchestra.
- The `intervaltreertools` package implements a custom way of working with chunks of score during composition.
- The `lilyfiletools` package implements an object-oriented interface to arbitrarily structured LilyPond input files.
- The `pitcharraytools` package implements an object-oriented way of composing with pitches, pitch-classes and other pitch-related objects independent of rhythmic context.
- The experimental `quantizationtools` package implements classes and functions for quantizing rhythmic events.
- The `sievetools` package implements an object-oriented interface to the basics of Xenakis's system of sieves.
- The `tonalitytools` package implements classes and methods to model the basics of functional harmonic analysis.
- The `verticalitytools` package provides vertical-moment-based iteration and analysis of any score.
- The `pitchtools` package has grown considerably in size and functionality. Classes now exist to model named and numbered chromatic pitches (and pitch-classes), named and numbered diatonic pitches (and pitch-classes), melodic and harmonic diatonic intervals (and interval-classes), melodic and harmonic chromatic intervals (and interval-classes), as well as ordered segments and unordered sets of these and related objects. The package contains dozens of functions to create, inspect, iterate, analyze and transpose these classes and their collections.
- The old `listtools` package has been renamed `seqtools`.
- Dozens of new functions for cutting, pasting, partitioning, breaking, arranging and reordering score components have been added to the system. See the new functions in `componenttools`, `containertools`, `leaftools`, `measuretools` and `scoretools` for details.
- The core component classes modeling notes, rests, chords, tuplets, measures, voices, staves and scores have been reimplemented to consume dramatically less memory, making it much easier to work with arrays of hundreds and thousands of components.

- Abjad core formatting logic has been optimized to make the formatting of scores with hundreds or thousands of events take much less time than before.
- The component duration interfaces have been replaced by more straightforward read-only component attributes.
- Added Ferneyhough Unsichbare Farben example.



3.10 Abjad 1.1.1

- More complete documentation.
- The configuration file `config` changed to pure Python `config.py`. The file now supports more settings previously read as environment variables. All user settings are now found in this file. Users no longer need to set environment variables.
- Some new classes
 - `_HistoryInterface`. Use the `_HistoryInterface` to apply attributes to any component in score that will be completely ignored by Abjad. Think of the `_HistoryInterface` as a private user namespace.

- `_NoteColumnInterface` to handle the LilyPond `NoteColumn` grob.
- `_SpanBarInterface`. See API for details.
- `InvisibleStaff()` staff.
- `Moment` utility class to model the Abjad representation of the LilyPond moment.
- New Spanners
 - `TempoProportional` spanner.
- More than a dozen new tools added.

3.11 Abjad 1.1.0

- Many structure transform tools added. See the `abjad.tools.*` in the *Abjad API* package.
- Construction, transformation, manipulation and all other tools now grouped cleanly into packages.
- New `abjad-book` application available. Use `abjad-book` to interpret Abjad code blocks embedded in HTML, LaTeX and reST documents.

3.12 Abjad 1.0.1055

Changes to the public interface:

- Abjad now models ties exclusively with the `Tie` spanner. The old `_TieInterface._set` attribute is now deprecated.
- You can no longer say `t.tie = True` or `t.tie = False`, for leaf `t`. You must structurally span `t` as `Tie(t)` instead.
- New public properties in `_SpannerReceptor`: `chain`, `parented`, `count`.
- New public helpers:
 - `construct.notes_curve()`
 - `durationtools.rationalize()`
 - `iterate.tie_chains()`
 - `list_helpers()`
 - `mathtools.interpolate_divide()`
 - `measuretools.concentrate()`
 - `measuretools.scale_and_remeter()`
 - `measuretools.spin()`
 - `play()`
- Grace note `append()` and `extend()` no longer throw errors.

3.13 Abjad 1.0.1022

- First public release of Abjad.

Examples

BARTÓK: *MIKROKOSMOS*

This example reconstructs the last five measures of Bartók's "Wandering" from *Mikrokosmos*, volume III. The end result is just a few measures long but covers the basic features you'll use most often in Abjad.

Here is what we want to end up with:



4.1 The score

We'll construct the fragment top-down from containers to notes. We could have done it the other way around but it will be easier to keep the big picture in mind this way. Later, you can rebuild the example bottom-up as an exercise.

First let's create an empty score with a pair of staves connected by a brace:

```
abjad> score = Score([])
abjad> piano_staff = scoretools.PianoStaff([])
abjad> upper_staff = Staff([])
abjad> lower_staff = Staff([])

abjad> piano_staff.append(upper_staff)
abjad> piano_staff.append(lower_staff)
abjad> score.append(piano_staff)
```

Here we create an empty score and assign it to the `score` variable. Then we create an empty piano staff assigned to the `piano_staff` variable and two empty staves assigned to the `upper_staff` and `lower_staff` variables. Finally, we append the two staves to the piano staff and the piano staff to the score.

4.2 The measures

Now let's add some measures to our score:

```

abjad> m1 = Measure((2, 4), [])
abjad> m2 = Measure((3, 4), [])
abjad> m3 = Measure((2, 4), [])
abjad> m4 = Measure((2, 4), [])
abjad> m5 = Measure((2, 4), [])

abjad> upper_measures = [m1, m2, m3, m4, m5]
abjad> lower_measures = componenttools.copy_components_and_covered_spanners(upper_measures)

abjad> upper_staff.extend(upper_measures)
abjad> lower_staff.extend(lower_measures)

```

The lower measures are copies of the upper measures.

Note that we add lists of measures to staves with `extend()`. This is because `extend()` is used for adding many objects to an iterable at once while `append()` is used to add only one object at a time.

4.3 The notes

Now let's add some notes. We begin with the upper staff:

```

abjad> upper_measures[0].extend([Note(i, (1, 8)) for i in [9, 7, 5, 4]])

abjad> upper_measures[1].extend(notetools.make_notes([2, 7, 5, 4, 2], [(1, 4)] + [(1, 8)] * 4))

abjad> notes = notetools.make_notes([0, 2, 4, 5, 4], [(1, 8), (1, 16), (1, 16), (1, 8), (1, 8)])
abjad> upper_measures[2].extend(notes)

abjad> upper_measures[3].append(Note("d'2"))

abjad> upper_measures[4].append(Note("d'2"))

```

Now let's add notes to the lower staff. This will be a more intricate process than that needed for the upper staff. We added notes directly to the measures of the upper staff. But this will not be possible for the lower staff because of the simultaneous voices the lower staff contains.

We add notes to the lower staff measure by measure:

```

abjad> main_voice_m1 = Voice("b4 d'8 c'8")
abjad> main_voice_m1.name = 'main_voice'
abjad> lower_measures[0].append(main_voice_m1)

abjad> main_voice_m2 = Voice("b8 a8 af4 c'8 bf8")
abjad> main_voice_m2.name = 'main_voice'
abjad> lower_measures[1].append(main_voice_m2)

abjad> main_voice_m3 = Voice("a8 g8 fs8 gl6 a16")
abjad> main_voice_m3.name = 'main_voice'
abjad> lower_measures[2].append(main_voice_m3)

```

Notice that we give the same name to the three voices contained in the first three measures of the lower staff.

It is in the last two measures of the lower staff where Bartók writes two voices at once. We'll name the second of these two voices the *appendix_voice*:

```

abjad> appendix_voice_m4 = Voice([Note("b2")])
abjad> appendix_voice_m4.name = 'appendix_voice'
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('voiceOne')
abjad> lilypond_command_mark.attach(appendix_voice_m4)

abjad> main_voice_m4 = Voice("b4 a4")
abjad> main_voice_m4.name = 'main_voice'
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('voiceTwo')
abjad> lilypond_command_mark.attach(main_voice_m4)

abjad> container = Container([appendix_voice_m4, main_voice_m4])
abjad> container.is_parallel = True
abjad> lower_measures[3].append(container)

```

The LilyPond `\voiceOne` and `\voiceTwo` commands determine the direction of the stems in different voices.

Note that we must put both voices in a parallel container because they occur at the same time in the score. We do this by creating an Abjad container and then setting the `is_parallel` attribute of the container to true.

We now do a similar thing for the last measure:

```

abjad> appendix_voice_m5 = Voice("b2")
abjad> appendix_voice_m5.name = 'appendix_voice'
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('voiceOne')
abjad> lilypond_command_mark.attach(appendix_voice_m5)

abjad> main_voice_m5 = Voice("g2")
abjad> main_voice_m5.name = 'main_voice'
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('voiceTwo')
abjad> lilypond_command_mark.attach(main_voice_m5)

abjad> container = Container([appendix_voice_m5, main_voice_m5])
abjad> container.is_parallel = True
abjad> lower_measures[4].append(container)

```

Here's our work so far:

```
abjad> show(score)
```



4.4 The details

Ok, let's add the details. First, notice that the bottom staff has a treble clef just like the top staff. Let's change that:

```
abjad> contexttools.ClefMark('bass')(lower_staff)
```

Now let's add dynamic marks. For the top staff, we'll add them to the first note of the first measure and the second note of the second measure. For the bottom staff, we'll add dynamic markings to the second note of the first measure and the fourth note of the second measure.

```
abjad> contexttools.DynamicMark('pp')(upper_measures[0][0])
abjad> contexttools.DynamicMark('mp')(upper_measures[1][1])
abjad> contexttools.DynamicMark('pp')(lower_measures[0][0][1])
abjad> contexttools.DynamicMark('mp')(lower_measures[1][0][3])
```

Let's add a double bar to the end of the piece:

```
abjad> bar_line = marktools.BarLine('.')
abjad> bar_line.attach(lower_staff.leaves[-1])
```

And see how things are coming out:

```
abjad> show(score)
```



Notice that the beams of the eighth and sixteenth notes appear as you would usually expect: grouped by beat. We get this for free thanks to LilyPond's default beaming algorithm. But this is not the way Bartók notated the beams. Let's set the beams as Bartók did with some crossing the bar lines:

```
abjad> spannertools.BeamSpanner(upper_measures[0])
abjad> spannertools.BeamSpanner(lower_staff.leaves[1:5])
abjad> spannertools.BeamSpanner(lower_staff.leaves[6:10])

abjad> show(score)
```



Now some slurs:

```
abjad> spannertools.SlurSpanner(upper_staff.leaves[0:5])
abjad> spannertools.SlurSpanner(upper_staff.leaves[5:])
abjad> spannertools.SlurSpanner(lower_staff.leaves[1:6])
abjad> spannertools.SlurSpanner(lower_staff.leaves[6:13] + (main_voice_m4, main_voice_m5))
```

Hairpins:

```
abjad> spannertools.CrescendoSpanner(upper_staff.leaves[-7:-2])
abjad> spannertools.DecrescendoSpanner(upper_staff.leaves[-2:])
```

A ritardando marking above the last seven notes of the upper staff:

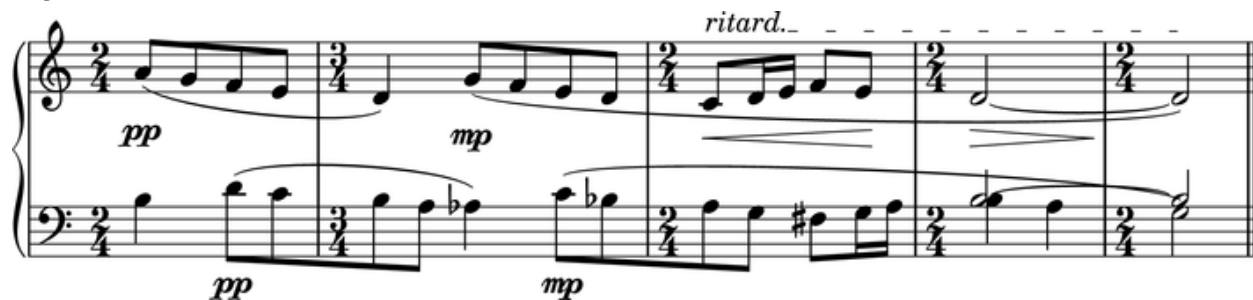
```
abjad> text_spanner = spannertools.TextSpanner(upper_staff.leaves[-7:])
abjad> text_spanner.override.text_spanner.bound_details__left__text = markuptools.Markup('ritard.')
```

And ties connecting the last two notes in each staff:

```
abjad> tietools.TieSpanner(upper_staff[-2:])  
abjad> tietools.TieSpanner([appendix_voice_m4[0], appendix_voice_m5[0]])
```

The final result:

```
abjad> show(score)
```



FERNEYHOUGH: *UNSICHTBARE FARBEN*

Mikhail Malt analyzes the rhythmic materials of Ferneyhough’s *Unsichtbare Farben* in *The OM Composer’s Book 2*.

Malt explains that Ferneyhough used OpenMusic to create an “exhaustive catalogue of rhythmic cells” such that:

1. They are subdivided into two pulses, with proportions from 1/1 to 1/11.
2. The second pulse is subdivided successively by 1, 2, 3, 4, 5 and 6.

Let’s recreate Malt’s results in Abjad.

5.1 The proportions

First we define proportions:

```
abjad> proportions = [(1, n) for n in range(1, 11 + 1)]

abjad> proportions
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 11)]
```

5.2 The transforms

Then we make aliases to give shorter names to two functions with long names:

```
abjad> make_tuplet = tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_encourage_c
abjad> tie_chain_to_tuplet = tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_c
```

And then define a helper function:

```
def divide_tuplet(tuplet, n):
    last_tie_chain = tietools.get_tie_chain(tuplet[-1])
    proportions = n * [1]
    new = tie_chain_to_tuplet(last_tie_chain, proportions)
    return new
```

5.3 The rhythms

We set the duration of each tuplet equal to a quarter note:

```
abjad> duration = Fraction(1, 4)
```

And then we make the rhythms:

```
music = []
for proportion in proportions:
    tuplets = []
    for n in range(1, 6 + 1):
        tuplet = make_tuplet(duration, proportion)
        divide_tuplet(tuplet, n)
        tuplets.append(tuplet)
    music.extend(tuplets)
```

5.4 The score

Finally we make the score:

```
abjad> staff = stafftools.RhythmicStaff(music)
abjad> score = Score([staff])
abjad> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)
```

Configure containers:

```
abjad> contexttools.TimeSignatureMark((1, 4))(staff)
abjad> score.override.bar_number.transparent = True
abjad> score.set.proportional_notation_duration = schemetools.SchemeMoment(1, 56)
abjad> score.set.tuplet_full_length = True
abjad> score.override.spacing_spanner.uniform_stretching = True
abjad> score.override.spacing_spanner.strict_note_spacing = True
abjad> score.override.tuplet_bracket.padding = 2
abjad> score.override.tuplet_bracket.staff_padding = 4
abjad> score.override.tuplet_number.text = schemetools.SchemeFunction('tuplet-number::calc-fraction-t')
abjad> score.override.time_signature.stencil = False
abjad> score.override.bar_line.stencil = False
```

Import layout tools:

```
abjad> from abjad.tools import layouttools
```

Configure the LilyPond file:

```
abjad> lilypond_file.default_paper_size = '11x17', 'portrait'
abjad> lilypond_file.global_staff_size = 12
abjad> lilypond_file.layout_block.indent = 0
abjad> lilypond_file.layout_block.ragged_right = True
abjad> lilypond_file.paper_block.ragged_bottom = True
abjad> lilypond_file.paper_block.system_system_spacing = layouttools.make_spacing_vector(0, 0, 8, 0)
```

And show the result:

```
abjad> show(lilypond_file)
```

The musical score consists of 10 staves, each containing a series of notes connected by lines. The notes are labeled with ratios, such as 3:2, 5:4, 7:4, 9:8, 6:5, 4:3, 11:8, 12:11, 18:11, and 15:11. The ratios are written above the lines, and some are written below the lines. The music is organized into measures, with some measures containing multiple notes connected by lines. The ratios are written above the lines, and some are written below the lines.

LIGETI: *DÉSORDRE*

This example demonstrates the power of exploiting redundancy to model musical structure. The piece that concerns us here is Ligeti’s *Désordre*: the first piano study from Book I. Specifically, we will focus on modeling the first section of the piece:

[illegible]

The redundancy is immediately evident in the repeating pattern found in both staves. The pattern is hierarchical. At the smallest level we have what we will here call a *cell*:

There are two of these cells per measure. Notice that the cells are strictly contained within the measure (i.e., there are no cells crossing a bar line). So, the next level in the hierarchy is the measure. Notice that the measure sizes (the meters) change and that these changes occur independently for each staff, so that each staff carries it's own sequence of measures. Thus, the staff is the next level in the hierarchy. Finally there's the piano staff, which is composed of the right hand and left hand staves.

In what follows we will model this structure in this order (*cell*, *measure*, *staff*, *piano staff*), from bottom to top.

6.1 The cell

Before plunging into the code, observe the following characteristic of the *cell*:

1. It is composed of two layers: the top one which is an octave “chord” and the bottom one which is a straight eighth note run.
2. The total duration of the *cell* can vary, and is always the sum of the eight note runs.
3. The eight note runs are always stem down while the octave “chord” is always stem up.
4. The eight note runs are always beamed together and slurred, and the first two notes always have the dynamic markings ‘f’ ‘p’.

The two “layers” of the *cell* we will model with two Voices inside a parallel Container. The top Voice will hold the octave “chord” while the lower Voice will hold the eighth note run. First the eighth notes:

```
abjad> pitches = [1, 2, 3]
abjad> notes = notetools.make_notes(pitches, [(1, 8)])
abjad> spannertools.BeamSpanner(notes)
abjad> spannertools.SlurSpanner(notes)
abjad> contexttools.DynamicMark('f')(notes[0])
abjad> contexttools.DynamicMark('p')(notes[1])

abjad> voice_lower = Voice(notes)
abjad> voice_lower.name = 'rh_lower'
abjad> marktools.LilyPondCommandMark('voiceTwo')(voice_lower)
```

The notes belonging to the eighth note run are first beamed and slurred. Then we add the dynamic marks to the first two notes, and finally we put them inside a Voice. After naming the voice we number it 2 so that the stems of the notes point down.

Now we construct the octave:

```
abjad> import math
abjad> n = int(math.ceil(len(pitches) / 2.))
abjad> chord = Chord([pitches[0], pitches[0] + 12], (n, 8))
abjad> marktools.Articulation('>')(chord)

abjad> voice_higher = Voice([chord])
abjad> voice_higher.name = 'rh_higher'
abjad> marktools.LilyPondCommandMark('voiceOne')(voice_higher)
```

The duration of the chord is half the duration of the running eighth notes if the duration of the running notes is divisible by two. Otherwise the duration of the chord is the next integer greater than this half. We add the articulation marking and finally add the Chord to a Voice, to which we set the number to 1, forcing the stem to always point up.

Finally we combine the two voices in a parallel Container:

```
abjad> p = Container([voice_lower, voice_higher])
abjad> p.is_parallel = True
```

This results in the complete *Désordre cell*:

```
LilyPond file written to 'desordre-cell.ly' ...
```



Because this *cell* appears over and over again, we want to reuse this code to generate any number of these *cells*. We here encapsulate it in a function that will take only a list of pitches:

```
def desordre_cell(pitches):
    '''The function constructs and returns a *Désordre cell*.
    - 'pitches' is a list of numbers or, more generally, pitch tokens.
    '''
    notes = [Note(p, (1, 8)) for p in pitches]
    spannertools.BeamSpanner(notes)
    spannertools.SlurSpanner(notes)
    contexttools.DynamicMark('f')(notes[0])
    contexttools.DynamicMark('p')(notes[1])
    v_lower = Voice(notes)
    v_lower.name = 'rh_lower'
    marktools.LilyPondCommandMark('voiceTwo')(v_lower)

    n = int(math.ceil(len(pitches) / 2.))
    chord = Chord([pitches[0], pitches[0] + 12], (n, 8))
    marktools.Articulation('>')(chord)
    v_higher = Voice([chord])
    v_higher.name = 'rh_higher'
    marktools.LilyPondCommandMark('voiceOne')(v_higher)
    p = Container([v_lower, v_higher])
    p.is_parallel = True
    # make all 1/8 beats breakable
    for n in v_lower.leaves[:-1]:
        marktools.BarLine('')(n)
    return p
```

Now we can call this function to create any number of *cells*. That was actually the hardest part of reconstructing the opening of Ligeti's *Désordre*. Because the repetition of patters occurs also at the level of measures and staves, we will now define functions to create these other higher level constructs.

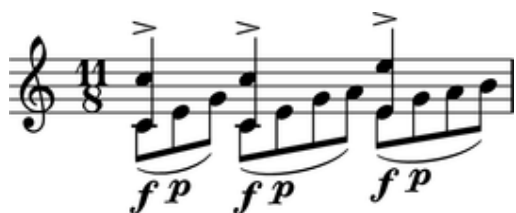
6.2 The measure

We define a function to create a measure from a list of lists of numbers:

```
def measure_build(pitches):
    '''Constructs a measure composed of *Désordre cells*.
    - 'pitches' is a list of lists of number (e.g., [[1, 2, 3], [2, 3, 4]])
    The function returns a DynamicMeasure.
    '''
    result = measuretools.DynamicMeasure([ ])
    for seq in pitches:
        result.append(desordre_cell(seq))
    # make denominator 8
    if contexttools.get_effective_time_signature(result).denominator == 1:
        result.denominator = 8
    return result
```

The function is very simple. It simply creates a `DynamicMeasure` and then populates it with *cells* that are created internally with the function previously defined. The function takes a list *pitches* which is actually a list of lists of pitches (e.g., `[[1, 2, 3], [2, 3, 4]]`). The list of lists of pitches is iterated to create each of the *cells* to be appended to the `DynamicMeasures`. We could have defined the function to take ready made *cells* directly, but we are building the hierarchy of functions so that we can pass simple lists of lists of numbers to generate the full structure. To construct a Ligeti measure we would call the function like so:

```
abjad> measure = measure_build([[0, 4, 7], [0, 4, 7, 9], [4, 7, 9, 11]])
LilyPond file written to 'desordre-measure.ly' ...
abjad> show(Staff([measure]))
```



6.3 The staff

Now we move up to the next level, the staff:

```
def staff_build(pitches):
    '''Returns a Staff containing DynamicMeasures.'''
    result = Staff([])
    for seq in pitches:
        measure = measure_build(seq)
        result.append(measure)
    return result
```

The function again takes a plain list as argument. The list must be a list of lists (for measures) of lists (for cells) of pitches. The function simply constructs the Ligeti measures internally by calling our previously defined function and puts them inside a `Staff`. As with measures, we can now create full measure sequences with this new function:

```
abjad> pitches = [[[-1, 4, 5], [-1, 4, 5, 7, 9]], [[0, 7, 9], [-1, 4, 5, 7, 9]]]
abjad> staff = staff_build(pitches)
LilyPond file written to 'desordre-staff.ly' ...
abjad> show(staff)
```



6.4 The score

Finally a function that will generate the whole opening section of the piece *Désordre*:

```
def desordre_build(pitches):
    '''Returns a complete PianoStaff with Ligeti music!'''
    assert len(pitches) == 2
```



```

piano = PianoStaff([])
# build the music...
for hand in pitches:
    seq = staff_build(hand)
    piano.append(seq)
# set clef and key signature to left hand staff...
piano[1].clef.forced = stafftools.Clef('bass')
piano[1].key_signature.forced = tonalitytools.KeySignature('b', 'major')
return piano

```

The function creates a `PianoStaff`, constructs `Staves` with Ligeti music and appends these to the empty `PianoStaff`. Finally it sets the clef and key signature of the lower staff to match the original score. The argument of the function is a list of length 2, depth 3. The first element in the list corresponds to the upper staff, the second to the lower staff.

The final result:

```

abjad> top = [[[-1, 4, 5], [-1, 4, 5, 7, 9]], [[0, 7, 9], [-1, 4, 5, 7, 9]], [[2, 4, 5, 7, 9], [0, 5, 7, 9]], [[-1, 4, 5, 7, 9], [0, 5, 7, 9]]]
abjad> bottom = [[[-9, -4, -2], [-9, -4, -2, 1, 3]], [[-6, -2, 1], [-9, -4, -2, 1, 3]], [[-4, -2, 1], [-9, -4, -2, 1, 3]], [[-4, -2, 1], [-9, -4, -2, 1, 3]]]
abjad>
abjad> desordre = desordre_build([top, bottom])
LilyPond file written to 'desordre-final.ly' ...
abjad> show(desordre)

```



Now that we have the redundant aspect of the piece compactly expressed and encapsulated, we can play around with it by changing the sequence of pitches.

Note: In order for each staff to carry its own sequence of independent measure changes, LilyPond requires some special setting up prior to rendering. Specifically, one must move the *Timing_translator* from the score level to the

level of staves. In this example we used the ‘tirnaveni’ template, which is configured to do just that. You may want to study this template (in the “templates” directory of the abjad distribution). Refer to the LilyPond documentation on [Polymetric notation](#) to learn all about how this works.

MOZART: *MUSIKALISCHES WÜRFELSPIEL*

Mozart’s dice game is a method for aleatorically generating sixteen-measure-long minuets. For each measure, two six-sided dice are rolled, and the sum of the dice used to look up a measure number in one of two tables (one for each half of the minuet). The measure number then locates a single measure from a collection of musical fragments. The fragments are concatenated together, and “music” results.

Implementing the dice game in a composition environment is somewhat akin to (although also somewhat more complicated than) the ubiquitous `hello world program` which every programming language uses to demonstrate its basic syntax.

Note: The musical dice game in question (*k516f*) has long been attributed to Mozart, albeit inconclusively. Its actual provenance is a musicological problem with which we are unconcerned here.

7.1 The materials

At the heart of the dice game is a large collection, *or corpus*, of musical fragments. Each fragment is a single 3/8 measure, consisting of a treble voice and a bass voice. Traditionally, these fragments are stored in a “score”, or “table of measures”, and located via two tables of measure numbers, which act as lookups, indexing into that collection.

Duplicate measures in the original corpus are common. Notably, the 8th measure - actually a pair of measures represent the first and second alternate ending of the first half of the minuet - are always identical. The last measure of the piece is similarly limited - there are only two possibilities rather than the usual eleven (for the numbers 2 to 12, being all the possible sums of two 6-sided dice).

How might we store this corpus compactly?

Some basic musical information in Abjad can be stored as strings, rather than actual collections of class instances. Abjad can parse simple LilyPond strings via `abjad.tools.iotools.parse_lilypond_input_string()`, which interprets a subset of LilyPond syntax, and understands basic concepts like notes, chords, rests and skips, as well as beams, slurs, ties, and articulations.

```
lily_string = "c'4 ( d'4 <cs' e'>8 ) -. r8 <g' b' d''>4 ^\marcato ~ <g' b' d''>1"
abjad> parsed_result = iotools.parse_lilypond_input_string(lily_string)
LilyPond file written to 'mozart-parsing-example.ly' ...
abjad> show(parsed_result)
```

WOLFGANG AMADEUS MOZART

Musikalisches Würfelspiel

Table of Measure Numbers

Part One

	I	II	III	IV	V	VI	VII	VIII
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

Part Two

	I	II	III	IV	V	VI	VII	VIII
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

Table of Measures



Figure 7.1: Part of a pen-and-paper implementation from the 20th century.



So, instead of storing our musical information as Abjad components, we'll represent each fragment in the corpus as a pair of strings: one representing the bass voice contents, and the other representing the treble. This pair of strings can be packaged together into a collection. For this implementation, we'll package them into a dictionary. Python dictionaries are cheap, and often provide more clarity than lists; the composer does not have to rely on remembering a convention for what data should appear in which position in a list - they can simply label that data semantically. In our musical dictionary, the treble voice will use the key 't' and the bass voice will use the key 'b'.

```
abjad> fragment = {'t': "g''8 ( e''8 c''8 )", 'b': '<c e>4 r8'}
```

Instead of relying on measure number tables to find our fragments - as in the original implementation, we'll package our fragment dictionaries into a list of lists of fragment dictionaries. That is to say, each of the sixteen measures in the piece will be represented by a list of fragment dictionaries. Furthermore, the 8th measure, which breaks the pattern, will simply be a list of two fragment dictionaries. Structuring our information in this way lets us avoid using measure number tables entirely; Python's list-indexing affordances will take care of that for us. The complete corpus looks like this:

```
measures = [
    [ # measure 1 choices
        {'b': 'c4 r8', 't': "e''8 c''8 g'8"},
        {'b': '<c e>4 r8', 't': "g'8 c''8 e''8"},
        {'b': '<c e>4 r8', 't': "g''8 ( e''8 c''8 )"},
        {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 g'16 e''16 c''16"},
        {'b': 'c4 r8', 't': "e''16 d'16 e''16 g'16 c''16 g'16"},
        {'b': '<c e>4 r8', 't': "g''8 f'16 e''16 d'16 c''16"},
        {'b': '<c e>4 r8', 't': "e''16 c'16 g'16 e''16 c''16 g'16"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c'8 g'8 e'8"},
        {'b': '<c e>4 r8', 't': "g''8 c''8 e''8"},
        {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
    ],
    [ # measure 2 choices
        {'b': 'c4 r8', 't': "e''8 c''8 g'8"},
        {'b': '<c e>4 r8', 't': "g'8 c''8 e''8"},
        {'b': '<c e>4 r8', 't': "g''8 e''8 c''8"},
        {'b': '<e g>4 r8', 't': "c''16 g'16 c''16 e''16 g'16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 g'16 e''16 c''16"},
        {'b': 'c4 r8', 't': "e''16 d'16 e''16 g'16 c''16 g'16"},
        {'b': '<c e>4 r8', 't': "g''8 f'16 e''16 d'16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c'16 g'16 e''16"},
        {'b': '<c e>4 r8', 't': "c'8 g'8 e'8"},
        {'b': '<c e>4 <c g>8', 't': "g''8 c''8 e''8"},
        {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"},
    ],
    [ # measure 3 choices
        {'b': '<b, g>4 g,8', 't': "d'16 e'16 f'16 d'16 c'16 b'16"},
        {'b': 'g,4 r8', 't': "b'8 d'8 g'8"},
        {'b': 'g,4 r8', 't': "b'8 d'16 b'16 a'16 g'16"},
        {'b': '<g b>4 r8', 't': "f'8 d'8 b'8"},
        {'b': '<b, d>4 r8', 't': "g'16 f'16 g'16 d'16 b'16 g'16"},
        {'b': '<g b>4 r8', 't': "f'16 e'16 f'16 d'16 c'16 b'16"},
        {'b': '<g, g>4 <b, g>8', 't': "b'16 c'16 d'16 e'16 f'16 d'16"},
        {'b': 'g8 g8 g8', 't': "<b' d''>8 <b' d''>8 <b' d''>8"},
        {'b': 'g,4 r8', 't': "b'16 c'16 d'16 b'16 a'16 g'16"},
    ]
]
```

```

    {'b': 'b,4 r8', 't': "d''8 ( b'8 g'8 )"},
    {'b': 'g4 r8', 't': "b'16 a'16 b'16 c''16 d''16 b'16"},
],
[ # measure 4 choices
    {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 e''16 g'8"},
    {'b': 'c4 r8', 't': "e''16 c''16 b'16 c''16 g'8"},
    {'b': '<e g>4 r8', 't': "c''8 ( g'8 e'8 )"},
    {'b': '<e g>4 r8', 't': "c''8 e''8 g'8"},
    {'b': '<e g>4 r8', 't': "c''16 b'16 c''16 g'16 e'16 c'16"},
    {'b': '<c e>4 r8', 't': "c''8 c''16 d''16 e''8"},
    {'b': 'c4 r8', 't': "<c' e''>8 <c' e''>16 <d' f''>16 <e' g''>8"},
    {'b': '<e g>4 r8', 't': "c''8 e''16 c''16 g'8"},
    {'b': '<e g>4 r8', 't': "c''16 g'16 e''16 c''16 g''8"},
    {'b': '<e g>4 r8', 't': "c''8 e''16 c''16 g''8"},
    {'b': '<e g>4 r8', 't': "c''16 e''16 c''16 g'16 e'8"},
],
[ # measure 5 choices
    {'b': 'c4 r8', 't': "fs''8 a''16 fs''16 d''16 fs''16"},
    {'b': 'c8 c8 c8', 't': "<fs' d''>8 <d' fs''>8 <fs' a''>8"},
    {'b': 'c4 r8', 't': "d''16 a'16 fs''16 d''16 a''16 fs''16"},
    {'b': 'c8 c8 c8', 't': "<fs' d''>8 <fs' d''>8 <fs' d''>8"},
    {'b': 'c4 r8', 't': "d''8 a'8 ^\\turn fs''8"},
    {'b': 'c4 r8', 't': "d''16 cs''16 d''16 fs''16 a''16 fs''16"},
    {'b': '<c a>4 <c a>8', 't': "fs''8 a''8 d''8"},
    {'b': '<c fs>8 <c fs>8 <c a>8', 't': "a'8 a'16 d''16 fs''8"},
    {'b': 'c8 c8 c8', 't': "<d' fs''>8 <d' fs''>8 <d' fs''>8"},
    {'b': '<c d>8 <c d>8 <c d>8', 't': "fs''8 fs''16 d''16 a''8"},
    {'b': '<c a>4 r8', 't': "fs''16 d''16 a'16 a''16 fs''16 d''16"},
],
[ # measure 6 choices
    {'b': '<b, d>8 <b, d>8 <b, d>8', 't': "g''16 fs''16 g''16 b''16 d''8"},
    {'b': '<b, d>4 r8', 't': "g''8 b''16 g''16 d''16 b'16"},
    {'b': '<b, d>4 r8', 't': "g''8 b''8 d''8"},
    {'b': '<b, g>4 r8', 't': "a'8 fs'16 g'16 b'16 g'16"},
    {'b': '<b, d>4 <b, g>8', 't': "g''16 fs''16 g''16 d''16 b'16 g'16"},
    {'b': 'b,4 r8', 't': "g''8 b''16 g''16 d''16 g'16"},
    {'b': '<b, g>4 r8', 't': "d''8 g'16 d''16 b'16 d'16"},
    {'b': '<b, g>4 r8', 't': "d''8 d'16 g'16 b'8"},
    {'b': '<b, d>8 <b, d>8 <b, g>8', 't': "a'16 g'16 fs''16 g'16 d''8"},
    {'b': '<b, d>4 r8', 't': "g''8 g'16 d'16 b'8"},
    {'b': '<b, d>4 r8', 't': "g''16 b''16 g''16 d'16 b'8"},
],
[ # measure 7 choices
    {'b': 'c8 d8 d,8', 't': "e''16 c''16 b'16 a'16 g'16 fs'16"},
    {'b': 'c8 d8 d,8', 't': "a'16 e''16 <b' d''>16 <a' c''>16 <g' b'>16 <fs' a'>16"},
    {'b': 'c8 d8 d,8', 't': "<b' d''>16 ( <a' c''>16 ) <a' c''>16 ( <g' b'>16 ) <g' b'>16 ( <fs' a'>16 )"},
    {'b': 'c8 d8 d,8', 't': "e''16 g''16 d''16 c''16 b'16 a'16"},
    {'b': 'c8 d8 d,8', 't': "a'16 e''16 d''16 g''16 fs''16 a''16"},
    {'b': 'c8 d8 d,8', 't': "e''16 a''16 g''16 b''16 fs''16 a''16"},
    {'b': 'c8 d8 d,8', 't': "c''16 e''16 g''16 d''16 a'16 fs''16"},
    {'b': 'c8 d8 d,8', 't': "e''16 g''16 d''16 g'16 a'16 fs''16"},
    {'b': 'c8 d8 d,8', 't': "e''16 c''16 b'16 g'16 a'16 fs'16"},
    {'b': 'c8 d8 d,8', 't': "e''16 c''16 b''16 g''16 a''16 fs''16"},
    {'b': 'c8 d8 d,8', 't': "a'8 d''16 c''16 b'16 a'16"},
],
[ # measure 8 choices (always using both)
    {'b': 'g,8 g16 f16 e16 d16', 't': "<g' b' d'' g''>4 r8"},
    {'b': 'g,8 b16 g16 fs16 e16', 't': "<g' b' d'' g''>4 r8"}],

```

```

],
[ # measure 9 choices
  {'b': 'd4 c8', 't': "fs''8 a''16 fs''16 d''16 fs''16"},
  {'b': '<d fs>4 r8', 't': "d''16 a'16 d''16 fs''16 a''16 fs''16"},
  {'b': '<d a>8 <d fs>8 <c d>8', 't': "fs''8 a''8 fs''8"},
  {'b': '<c a>4 <c a>8', 't': "fs''16 a''16 d''16 a''16 fs''16 a''16"},
  {'b': 'd4 c8', 't': "d'16 fs'16 a'16 d''16 fs''16 a''16"},
  {'b': 'd,16 d16 cs16 d16 c16 d16', 't': "<a' d'' fs''>8 fs''4 ^\\tr"},
  {'b': '<d fs>4 <c fs>8', 't': "a''8 ( fs''8 d''8 )"},
  {'b': '<d fs>4 <c fs>8', 't': "d''8 a''16 fs''16 d''16 a'16"},
  {'b': '<d fs>4 r8', 't': "d''16 a'16 d''8 fs''8"},
  {'b': '<c a>4 <c a>8', 't': "fs''16 d''16 a'8 fs''8"},
  {'b': '<d fs>4 <c a>8', 't': "a'8 d''8 fs''8"},
],
[ # measure 10 choices
  {'b': '<b, g>4 r8', 't': "g''8 b''16 g''16 d''8"},
  {'b': 'b,16 d16 g16 d16 b,16 g,16', 't': "g''8 g'8 g'8"},
  {'b': 'b,4 r8', 't': "g''16 b''16 g''16 b''16 d''8"},
  {'b': '<b, d>4 <b, d>8', 't': "a''16 g''16 b''16 g''16 d''16 g''16"},
  {'b': '<b, d>4 <b, d>8', 't': "g''8 d''16 b'16 g'8"},
  {'b': '<b, d>4 <b, d>8', 't': "g''16 b''16 d''16 b''16 g''8"},
  {'b': '<b, d>4 r8', 't': "g''16 b''16 g''16 d''16 b'16 g'16"},
  {'b': '<b, d>4 <b, d>8', 't': "g''16 d''16 g''16 b''16 g''16 d''16"},
  {'b': '<b, d>4 <b, g>8', 't': "g''16 b''16 g''8 d''8"},
  {'b': 'g,16 b,16 g8 b,8', 't': "g''8 d''4 ^\\tr"},
  {'b': 'b,4 r8', 't': "g''8 b''16 d''16 d''8"},
],
[ # measure 11 choices
  {'b': 'c16 e16 g16 e16 c'16 c16', 't': "<c'' e''>8 <c'' e''>8 <c'' e''>8"},
  {'b': 'e4 e16 c16', 't': "c''16 g'16 c''16 e''16 g''16 <c'' e''>16"},
  {'b': '<c g>4 <c e>8', 't': "e''8 g''16 e''16 c''8"},
  {'b': '<c g>4 r8', 't': "e''16 c''16 e''16 g''16 c''16 g''16"},
  {'b': '<c g>4 <c g>8', 't': "e''16 g''16 c''16 g''16 e''16 c''16"},
  {'b': 'c16 b,16 c16 d16 e16 fs16', 't': "<g' c'' e''>8 e''4 ^\\tr"},
  {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 c''8 g'8"},
  {'b': '<c g>4 <c e>8', 't': "e''8 c''16 e''16 g''16 c''16"},
  {'b': '<c g>4 <c e>8', 't': "e''16 c''16 e''8 g''8"},
  {'b': '<c g>4 <c g>8', 't': "e''16 c''16 g'8 e''8"},
  {'b': '<c g>4 <c e>8', 't': "e''8 ( g''8 c''8 )"},
],
[ # measure 12 choices
  {'b': 'g4 g,8', 't': "<c'' e''>8 <b' d''>8 r8"},
  {'b': '<g, g>4 g8', 't': "d''16 b'16 g'8 r8"},
  {'b': 'g8 g,8 r8', 't': "<c'' e''>8 <b' d''>16 <g' b'>16 g'8"},
  {'b': 'g4 r8', 't': "e''16 c''16 d''16 b'16 g'8"},
  {'b': 'g8 g,8 r8', 't': "g''16 e''16 d''16 b'16 g'8"},
  {'b': 'g4 g,8', 't': "b'16 d''16 g''16 d''16 b'8"},
  {'b': 'g8 g,8 r8', 't': "e''16 c''16 b'16 d''16 g''8"},
  {'b': '<g b>4 r8', 't': "d''16 b''16 g''16 d''16 b'8"},
  {'b': '<b, g>4 <b, d>8', 't': "d''16 b'16 g'8 g''8"},
  {'b': 'g16 fs16 g16 d16 b,16 g,16', 't': "d''8 g'4"},
],
[ # measure 13 choices
  {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 c''8 g'8"},
  {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g'8 c''8 e''8"},
  {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 e''8 c''8"},
  {'b': '<c e>4 <c e>8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
  {'b': '<c e>4 <c g>8', 't': "c''16 b''16 c''16 g''16 e''16 c''16"},

```

```

        {'b': '<c g>4 <c e>8', 't': "e''16 d''16 e''16 g''16 c''16 g''16"},
        {'b': '<c e>4 r8', 't': "g''8 f''16 e''16 d''16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c''16 g'16 e''16"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 c''8 e''8"},
        {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"}],
    ],
    [ # measure 14 choices
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 ( c''8 g'8 )"},
        {'b': '<c e>4 <c g>8', 't': "g'8 ( c''8 e''8 )"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 e''8 c''8"},
        {'b': '<c e>4 <c e>8', 't': "c''16 b'16 c''16 e''16 g'16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 b'16 c''16 g'16 e''16 c''16"},
        {'b': '<c g>4 <c e>8', 't': "e''16 d''16 e''16 g''16 c''16 g''16"},
        {'b': '<c e>4 <e g>8', 't': "g''8 f''16 e''16 d''16 c''16"},
        {'b': '<c e>4 r8', 't': "c''16 g'16 e''16 c''16 g'16 e''16"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "c''8 g'8 e''8"},
        {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "g''8 c''8 e''8"},
        {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"}],
    ],
    [ # measure 15 choices
        {'b': "<f a>4 <g d'>8", 't': "d''16 f''16 d''16 f''16 b'16 d''16"},
        {'b': 'f4 g8', 't': "d''16 f''16 a''16 f''16 d''16 b'16"},
        {'b': 'f4 g8', 't': "d''16 f''16 a'16 d''16 b'16 d''16"},
        {'b': 'f4 g8', 't': "d''16 ( cs''16 ) d''16 f''16 g'16 b'16"},
        {'b': 'f8 d8 g8', 't': "f''8 d''8 g''8"},
        {'b': 'f16 e16 d16 e16 f16 g16', 't': "f''16 e''16 d''16 e''16 f''16 g''16"},
        {'b': 'f16 e16 d8 g8', 't': "f''16 e''16 d''8 g''8"},
        {'b': 'f4 g8', 't': "f''16 e''16 d''16 c''16 b'16 d''16"},
        {'b': 'f4 g8', 't': "f''16 d''16 a'8 b'8"},
        {'b': 'f4 g8', 't': "f''16 a''16 a'8 b'16 d''16"},
        {'b': 'f4 g8', 't': "a'8 f''16 d''16 a'16 b'16"}],
    ],
    [ # measure 16 choices
        {'b': 'c8 g,8 c,8', 't': "c''4 r8"},
        {'b': 'c4 c,8', 't': "c''8 c'8 r8"}],
    ],
]

```

We can then use the `parse_lilypond_input_string()` function we saw earlier to “build” the treble and bass components of a measure like this:

```

def build_one_mozart_measure(measure_dict):
    # parse the contents of a measure definition dictionary
    treble = iotools.parse_lilypond_input_string(measure_dict['t'])
    bass = iotools.parse_lilypond_input_string(measure_dict['b'])
    return treble, bass

```

```

my_measure_dict = {'b': 'c4 ^\tr r8', 't': "e''8 ( c''8 g'8 )"}
abjad> treble, bass = build_one_mozart_measure(my_measure_dict)
abjad> f(treble)
{
    e''8 (
    c''8
    g'8 )
}
abjad> f(bass)
{

```



```

        c4 ^\tr
        r8
    }

```

7.2 The structure

After storing all of the musical fragments into a corpus, concatenating those elements into a musical structure is relatively trivial. We'll use the `choice()` function from Python's *random* module. `random.choice()` randomly selects one element from an input list.

```

abjad> import random
abjad> my_list = [1, 'b', 3]
abjad> my_result = [random.choice(my_list) for i in range(20)]
abjad> print my_result
[1, 3, 3, 3, 1, 1, 1, 'b', 3, 'b', 1, 'b', 'b', 1, 3, 'b', 3, 1, 'b', 1]

```

Our corpus is a list comprising sixteen sublists, one for each measure in the minuet. To build our musical structure, we can simply iterate through the corpus and call *choice* on each sublist, appending the chosen results to another list. The only catch is that the *eighth* measure of our minuet is actually the first-and-second-ending for the repeat of the first phrase. The sublist of the corpus for measure eight contains *only* the first and second ending definitions, and both of those measures should appear in the final piece, always in the same order. We'll have to intercept that sublist while we iterate through the corpus and apply some different logic.

The easiest way to intercept measure eight is to use the Python builtin *enumerate*, which allows you to iterate through a collection while also getting the index of each element in that collection:

```

def choose_mozart_measures( ):
    chosen_measures = [ ]
    for i, choices in enumerate(measures):
        if i == 7: # get both alternative endings for mm. 8
            chosen_measures.extend(choices)
        else:
            choice = random.choice(choices)
            chosen_measures.append(choice)
    return chosen_measures

```

Note: In *choose_mozart_measures* we test for index 7, rather than 8, because list indices count from 0 instead of 1.

The result will be a *seventeen*-item-long list of measure definitions:

```

abjad> choices = choose_mozart_measures( )
abjad> for i, measure in enumerate(choices): print i, measure
0 {'b': '< e>4 r8', 't': "e''16 c''16 g''16 e''16 c''16 g''16"}
1 {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"}
2 {'b': 'g,4 r8', 't': "b'8 d''8 g''8"}
3 {'b': '<e g>4 r8', 't': "c''16 g'16 e''16 c''16 g''8"}
4 {'b': 'c4 r8', 't': "d''8 a'8 ^\\turn fs''8"}
5 {'b': '<b, g>4 r8', 't': "a'8 fs'16 g'16 b'16 g''16"}
6 {'b': 'c8 d8 d,8', 't': "e''16 a''16 g''16 b''16 fs''16 a''16"}
7 {'b': 'g,8 g16 f16 e16 d16', 't': "<g' b' d'' g''>4 r8"}
8 {'b': 'g,8 b16 g16 fs16 e16', 't': "<g' b' d'' g''>4 r8"}
9 {'b': 'd4 c8', 't': "d'16 fs'16 a'16 d''16 fs''16 a''16"}
10 {'b': '<b, d>4 <b, g>8', 't': "g''16 b''16 g''8 d''8"}
11 {'b': '<c g>4 <c g>8', 't': "e''16 g''16 c''16 g''16 e''16 c''16"}
12 {'b': 'g4 g,8', 't': "b'16 d''16 g''16 d''16 b'8"}

```

```
13 {'b': 'c8 c8 c8', 't': "<e' c''>8 <e' c''>8 <e' c''>8"}
14 {'b': '<c e>16 g16 <c e>16 g16 <c e>16 g16', 't': "e''8 ( c''8 g'8 )"}
15 {'b': 'f4 g8', 't': "f''16 d''16 a'8 b'8"}
16 {'b': 'c8 g,8 c,8', 't': "c''4 r8"}
```

7.3 The score

Now that we have our raw materials, and a way to organize them, we can start building our score. The tricky part here is figuring out how to implement LilyPond’s repeat structure in Abjad. LilyPond structures its repeats something like this:

```
\repeat volta n {
    music to be repeated
}

\alternative {
    { ending 1 }
    { ending 2 }
    { ending n }
}

...music after the repeat...
```

What you see above is really just two containers, each with a little text (“repeat volta n” and “alternative”) prepended to their opening curly brace. To create that structure in Abjad, we’ll need to use the `LilyPondCommandMark` class, which allows you to place LilyPond commands like “break” relative to any score component:

```
abjad> con = Container("c'4 d'4 e'4 f'4")
abjad> marktools.LilyPondCommandMark('before-the-container', 'before')(con)
abjad> marktools.LilyPondCommandMark('after-the-container', 'after')(con)
abjad> marktools.LilyPondCommandMark('opening-of-the-container', 'opening')(con)
abjad> marktools.LilyPondCommandMark('closing-of-the-container', 'closing')(con)
abjad> marktools.LilyPondCommandMark('to-the-right-of-a-note', 'right')(con[2])
abjad> f(con)
\before-the-container
{
    \opening-of-the-container
    c'4
    d'4
    e'4 \to-the-right-of-a-note
    f'4
    \closing-of-the-container
}
\after-the-container
```

Notice the second argument to each `LilyPondCommandMark` above, like *before* and *closing*. These are format slot indications, which control where the command is placed in the LilyPond code relative to the score element it is attached to. To mimic LilyPond’s repeat syntax, we’ll have to create two `LilyPondCommandMark` instances, both using the “before” format slot, insuring that their command is placed before their container’s opening curly brace.

Now let’s take a look at the code that puts our score together:

```
def build_mozart_piano_staff( ):
    treble_staff = Staff([ ])
    bass_staff = Staff([ ])
```

```

# select the measures to use
choices = choose_mozart_measures( )

# create and populate the volta containers
treble_volta = Container([ ])
bass_volta = Container([ ])
for choice in choices[:7]:
    treble, bass = build_one_mozart_measure(choice)
    treble_volta.append(treble)
    bass_volta.append(bass)

# add marks to the volta containers
marktools.LilyPondCommandMark('repeat volta 2', 'before')(treble_volta)
marktools.LilyPondCommandMark('repeat volta 2', 'before')(bass_volta)

# add the volta containers to our staves
treble_staff.append(treble_volta)
bass_staff.append(bass_volta)

# create and populate the alternative ending containers
treble_alternative = Container([ ])
bass_alternative = Container([ ])
for choice in choices[7:9]:
    treble, bass = build_one_mozart_measure(choice)
    treble_alternative.append(treble)
    bass_alternative.append(bass)

# add marks to the alternative containers
marktools.LilyPondCommandMark('alternative', 'before')(treble_alternative)
marktools.LilyPondCommandMark('alternative', 'before')(bass_alternative)

# add the alternative containers to our staves
treble_staff.append(treble_alternative)
bass_staff.append(bass_alternative)

# create the remaining measures
for choice in choices[9:]:
    treble, bass = build_one_mozart_measure(choice)
    treble_staff.append(treble)
    bass_staff.append(bass)

# add meter
contexttools.TimeSignatureMark((3, 8))(treble_staff)

# add bass clef
contexttools.ClefMark('bass')(bass_staff)

# add the final double bar line at the end of each final measure
marktools.BarLine('|.') (treble_staff[-1])
marktools.BarLine('|.') (bass_staff[-1])

# combine into a PianoStaff
piano_staff = scoretools.PianoStaff([treble_staff, bass_staff])

# add an instrument name via contexttools.InstrumentMark
contexttools.InstrumentMark('Katzenklavier', 'kk.',
    target_context = scoretools.PianoStaff)(piano_staff)

```

```
return piano_staff
```

```
abjad> piano_staff = build_mozart_piano_staff( )
LilyPond file written to 'mozart-piano-staff.ly' ...
abjad> show(piano_staff)
```



Note: Our instrument name got cut off! Looks like we need to do a little formatting. Keep reading...

7.4 The document

As you can see above, we’ve now got our randomized minuet. However, we can still go a bit further. LilyPond provides a wide variety of settings for controlling the overall *look* of a musical document, often through its *header*, *layout* and *paper* blocks. Abjad, in turn, gives us object-oriented access to these settings through the its *lilypondfiletools* module.

We’ll use `abjad.tools.lilypondfiletools.make_basic_lilypond_file()` to wrap our `PianoStaff` inside a `LilyPondFile` instance. From there we can access the other “blocks” of our document to add a title, a composer’s name, change the global staff size, paper size, staff spacing and so forth.

```
def build_mozart_lily(piano_staff):

    # wrap the PianoStaff with a LilyPondFile
    lily = lilypondfiletools.make_basic_lilypond_file(piano_staff)

    # create some markup to use in our header block
    title = markuptools.Markup('\\bold \\sans "Ein Musikalisches Wuerfelspiel"')
    composer = schemetools.SchemeString("W. A. Mozart (maybe?)")

    # change various settings
    lily.global_staff_size = 12
    lily.header_block.title = title
    lily.header_block.composer = composer
    lily.layout_block.ragged_right = True
```

```

lily.paper_block.markup_system_spacing__basic_distance = 8
lily.paper_block.paper_width = 180

return lily

abjad> lily = build_mozart_lily(piano_staff)
abjad> print lily
LilyPondFile(PianoStaff<<2>>)

abjad> print lily.header_block
HeaderBlock(2)
abjad> f(lily.header_block)
\header {
  composer = #"W. A. Mozart (maybe?)"
  title = \markup { \bold \sans "Ein Musikalisches Wuerfelspiel" }
}

abjad> print lily.layout_block
LayoutBlock(1)
abjad> f(lily.layout_block)
\layout {
  ragged-right = ##t
}

abjad> print lily.paper_block
PaperBlock(2)
abjad> f(lily.paper_block)
\paper {
  markup-system-spacing #'basic-distance = #20
  paper-width = #180
}

```

And now the final result:

LilyPond file written to 'mozart-lily.ly' ...
 abjad> show(lily)

Ein Musikalisches Wuerfelspiel

W. A. Mozart (maybe?)

Katztenklavier

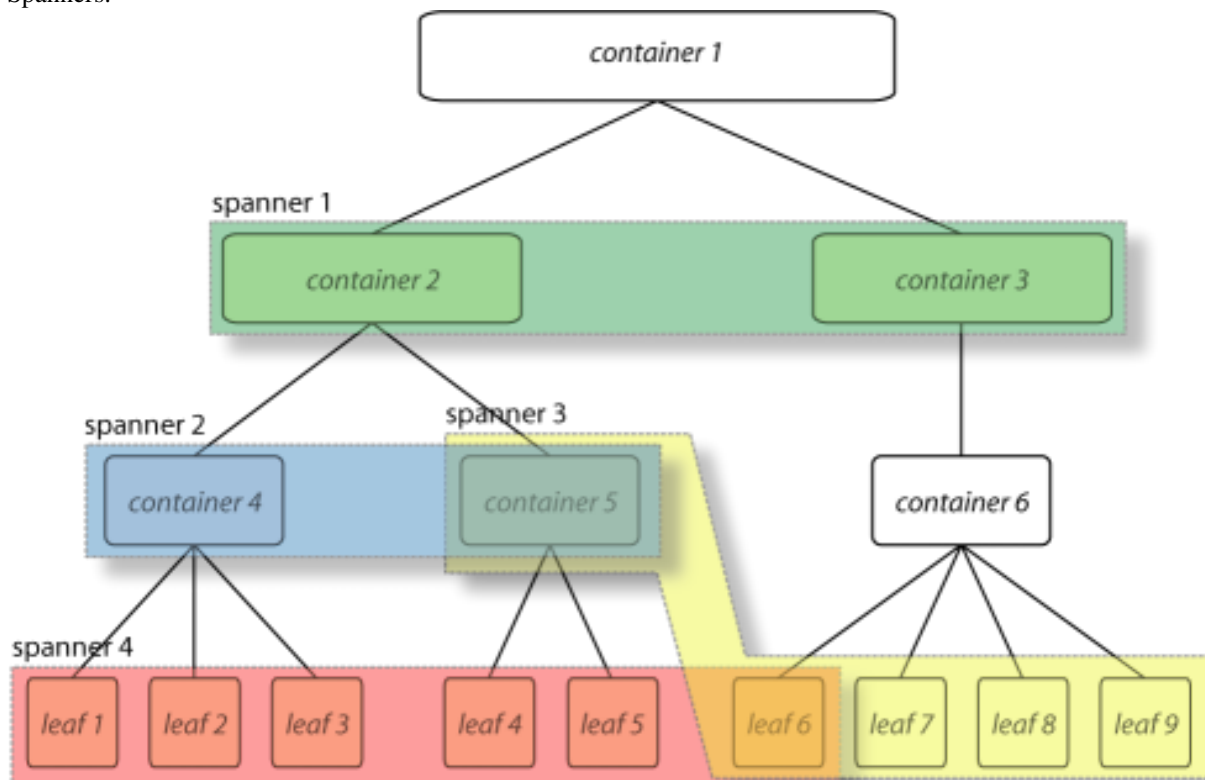
kk.

Tutorials

LEAF, CONTAINER, SPANNER, MARK

At the heart of Abjad's Symbolic Score-Control lies a powerful model that we call the Leaf Container Spanner Mark, or LCSM, model of the musical score.

The LCSM model can be schematically visualized as a superposition of two complementary and completely independent layers of structure: a *tree* that includes the Containers and the Leaves, and a layer of free floating *connectors* or Spanners.



There can be any number of Spanners, they may overlap, and they may connect to different levels of the tree hierarchy. The spanner attach to the elements of the tree, so a tree structure must exist for spanners to be made manifest.

8.1 Example 1

To understand the whys and hows of the LCSM model implemented in Abjad, it is probably easier to base the discussion on concrete musical examples. Let's begin with a simple and rather abstract musical fragment: a measure with nested tuplets.



What we see in this little fragment is a measure with 4/4 meter, 14 notes and four tuplet brackets prolating the notes. The three bottom tuplets (with ratios 5:4, 3:2, 5:4) prolates all but the last note. The topmost tuplet prolates all the notes in the measure and combines with the bottom three tuplets to doubly prolates all but the last note. The topmost tuplet as thus prolates three tuplets, each of which in turn prolates a group of notes. We can think of a tuplet as *containing* notes or other tuplets or both. Thus, in our example, the topmost tuplet contains three tuplets and a half note. Each of the tuplets contained by the topmost tuplet in turn contains five, three, and five notes respectively. If we add the measure, then we have a measure that contains a tuplet that contains tuplets that contain notes. The structure of the measure with nested tuplets as we have just described it has two important properties:

1. It is a *hierarchical* structure.
2. It follows *exclusive membership*, meaning that each element in the hierarchy (a note, a tuplet or a measure) has one and only one *parent*. In other words a single note is not contained in more than one tuplet simultaneously, and no one tuplet is contained in more than one other tuplet at the same time.

What we are describing here is a tree, and it is the structure of Abjad *containers*.

While this tree structure seem like the right way to represent the relationships between the elements of a score, it is not enough. Consider the tuplet example again with the following beaming alternatives:

Beaming alternative 1:



Beaming alternative 2:



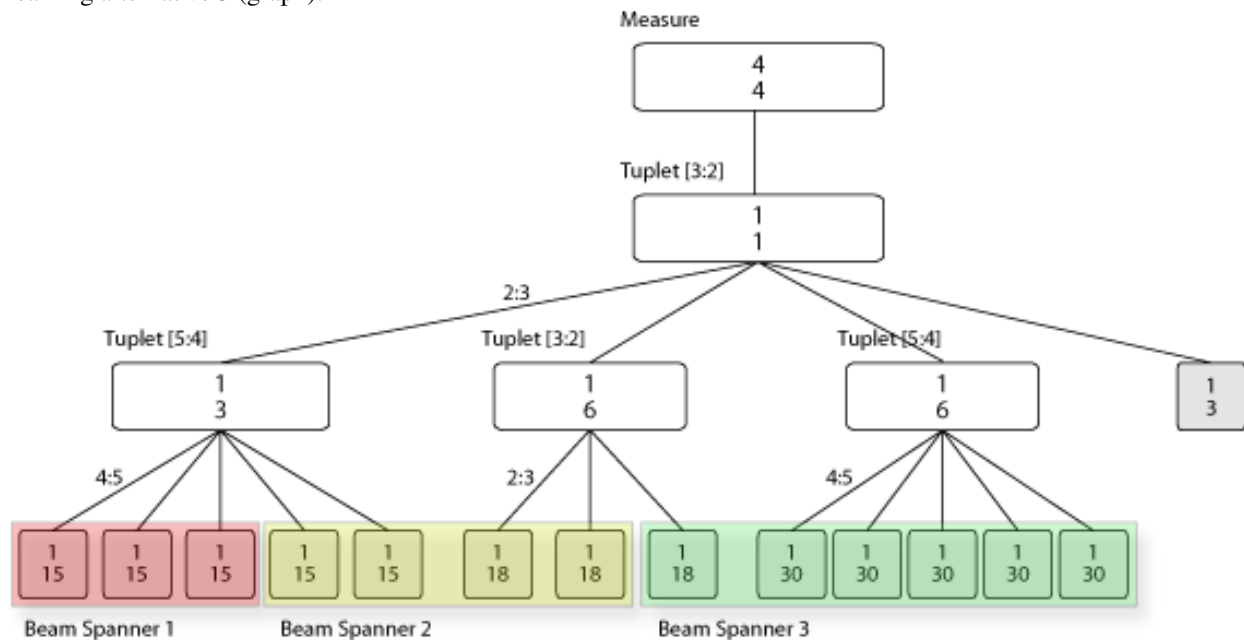
Beaming alternative 3:



Clearly the beaming of notes can be totally independent from the tuplet groupings. Beaming across tuplet groups implies beaming across nodes in the tree structure, which means that the beams do not adhere to the *exclusive (parent-hood) membership* characteristic of the tree. Beams must then be modeled independently as a separate and complementary structure. These are the Abjad *spanners*.

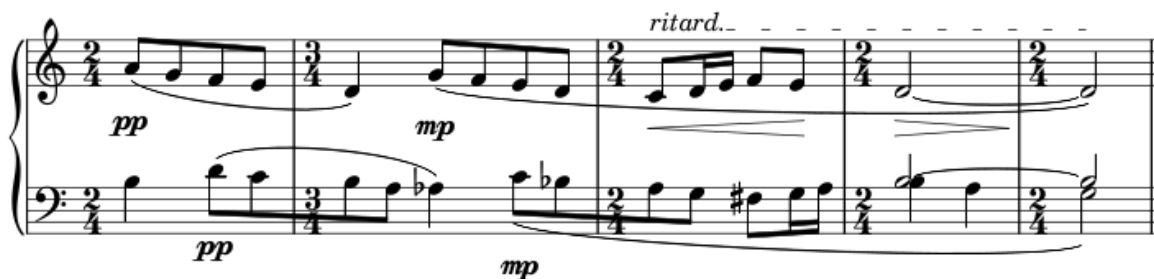
Below we have the score of our tuplet example with alternative beaming and its the Leaf-Container-Spanner graph. Notice that the colored blocks represent spanners.

Beaming alternative 3 (graph):



8.2 Example 2

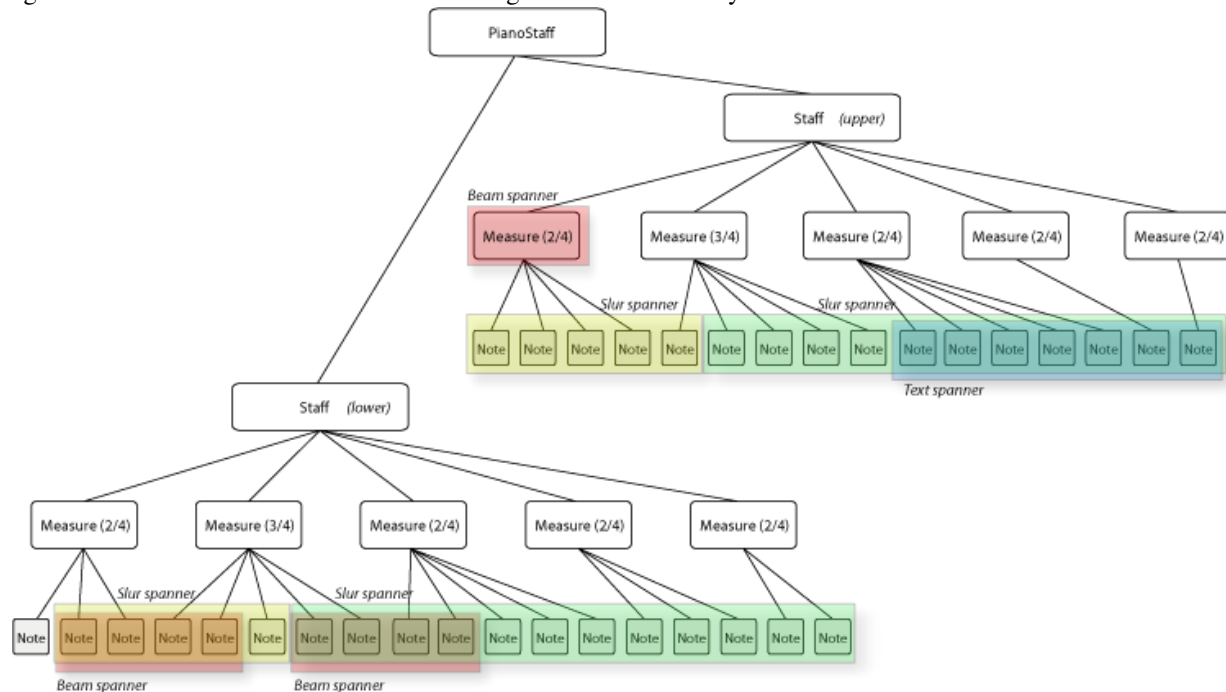
As a second example let's look at the last five measures of Bartók's *Wandering* from *Mikrokosmos* vol. III. As simple as it may seem, these five measures carry with them a lot of information pertaining to musical notation.



Note: Please refer to the [Bartok example](#) for a step by step construction of the musical fragment and its full Abjad code.

There are many musical signs of different types on the pages: notes, dynamic markings, clefs, staves, slurs, etc. These signs are structurally related to each other in different ways. Let's start by looking at the larger picture. The piano piece is written in two staves. As is customary, the staves are graphically grouped with a large curly brace attaching to them at the beginning of each system. Notice that each staff has a variety of signs associated with it. There are notes printed on the staff lines as well as meter indications and bar lines. Each note, for example, is in one and only one staff. A note is never in two staves at the same time. This is also true for measures. A measure in the top staff is not simultaneously drawn on the top staff and the bottom staff. It is better to think of each staff as having its own set of measures. Notice also that the notes in each staff fall within the region of one and only one measure, i.e. measures seem to contain notes. There is not one note that is at once in two measures (this is standard practice in musical notation, but it need not always be the case.)

As we continue describing the relationships between the musical signs in the page, we begin to discover a certain structure, or a convenient way of structuring the score for conceptualization and manipulation. All the music in a piano score seems to be written in what we might call a *staff group*. The staff group is *composed of* two staves. Each staff in turn appears to be composed of a series or measures, and each measure is composed of a series of notes. So again we find that the score structure can be organized hierarchically as a tree. This tree structure looks like this:

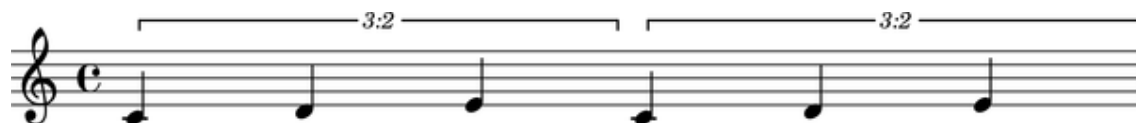


Notice again though that there are elements in the score that imply and require a different kind of grouping. The two four eighth-note runs in the lower staff are beamed together across the bar line and, based on our tree structure, across tree nodes. So do the slurs, the dynamics markings and the ritardando indication at the top of the score. As we have seen in the tuplets example, all these groups running across the tree structure can be defined with *spanners*.

WORKING WITH COMPONENT PARENTAGE

Many score objects contain other score objects.

```
abjad> tuplet = Tuplet(Fraction(2, 3), "c'4 d'4 e'4")
abjad> staff = Staff(2 * tuplet)
abjad> score = Score([staff])
abjad> show(score)
```



Abjad uses the idea of parentage to model the way objects contain each other.

9.1 Improper parentage

The improper parentage of the first note in score begins with the note itself:

```
abjad> note = score.leaves[0]
Note("c'4")

abjad> componenttools.get_improper_parentage_of_component(note)
(Note("c'4"), Tuplet(2/3, [c'4, d'4, e'4]), Staff{2}, Score<<1>>)
```

9.2 Proper parentage

The proper parentage of the note begins with only the immediate parent of the note:

```
abjad> componenttools.get_proper_parentage_of_component(note)
(Tuplet(2/3, [c'4, d'4, e'4]), Staff{2}, Score<<1>>)
```

Note: the length of the improper parentage of any component equals the length of the proper parentage of the component plus 1.

9.3 Parentage attributes

Use component tools to find score depth:

```
abjad> componenttools.component_to_score_depth(note)
3
```

Or score root:

```
abjad> componenttools.component_to_score_root(note)
Score<<1>>
```

Or to find whether a component has no (proper) parentage at all:

```
abjad> componenttools.is_orphan_component(note)
False
```

WORKING WITH THREADS

10.1 What is a thread?

A thread is a structural relationship binding a set of strictly sequential voice-level components.

Threads may be explicitly defined via voice instances:

```
abjad> v = Voice()
```

Or they may exist implicitly in certain score constructs in the absence of voice containers:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

Two contiguous voices must have the same name in order to be part of the same thread.

Here a thread does **not** exist between notes in different voices:

```
abjad> v_one = Voice("c'16 d'16 e'16 f'16")
abjad> v_two = Voice("c'8 d'8")
abjad> staff = Staff([v_one, v_two])
abjad> f(staff)
\new Staff {
    \new Voice {
        c'16
        d'16
        e'16
        f'16
    }
    \new Voice {
        c'8
        d'8
    }
}
```

Here a thread does exist:

```
abjad> v_one.name = 'flute'
abjad> v_two.name = 'flute'
abjad> f(staff)
\new Staff {
    \context Voice = "flute" {
        c'16
        d'16
        e'16
        f'16
    }
}
```

```
}
\context Voice = "flute" {
  c'8
  d'8
}
}
```

10.2 What are threads for?

Consider the following situation:



Are the two eighth notes in the second half of the measure the continuation of the ascending line in the first half, or is it the quarter note? Is the very last *C* the continuation of the top melodic line or is it the *A*? The stems might suggest an answer, but for Abjad, stem direction is not structural. What path should Abjad take to traverse this little score from the first note to the last *A*? This same problem appears when trying to apply spanners to parallel structures. Thus, threads are important in both score navigation and the application of spanners. In fact, threads are a requirement for spanner application.

In Abjad, the ambiguity is resolved through the explicit use of named voices.

The musical fragment above is constructed with the following code:

```
abjad> vA = Voice(notetools.make_notes([5, 7, 9, 11], [(1, 8)] * 4))
abjad> vB = Voice(notetools.make_notes([12, 11, 9], [(1, 8), (1, 8), (1, 4)]))
abjad> vC = Voice(Note(12, (1, 4)) * 2)
abjad> marktools.LilyPondCommandMark('voiceOne')(vA[0])
abjad> marktools.LilyPondCommandMark('voiceOne')(vB[0])
abjad> marktools.LilyPondCommandMark('voiceTwo')(vC[0])
abjad> p = Container([vB, vC])
abjad> p.is_parallel = True
abjad> staff = Staff([vA, p])
```



There's a staff that sequentially contains a voice and a parallel container. The container in turn holds two voices running simultaneously.

It is now clear from the code that the last *A* belongs with the two descending eighth notes. But there's still no indication about a relationship of continuity between the first voice in the sequence (*vA*) and any of the two following voices. Note that, while the LilyPond voice number commands setting may suggest that *vA* and *vB* belong together, this is not the case. The LilyPond voice number commands simply set the direction of stems in printed output.

To see this more clearly, suppose we want to add a slur spanner starting on the first note and ending on one of the last simultaneous notes. To attach the slur spanner to the voices we could try either:

```
abjad> spannertools.SlurSpanner([vA, vB])
```

or

```
abjad> spannertools.SlurSpanner([vA, vC])
```

But both raise a contiguity error. Abjad needs to see an explicit connection between either vA and vB or between vA and vC .

Observe the behavior of the `iterate_thread_forward_in_expr()` iterator on the *staff*:

```
::
```

```
abjad> from abjad.tools import threadtools
abjad> vA_thread_signature = threadtools.component_to_thread_signature(vA)
abjad> notes = threadtools.iterate_thread_forward_in_expr(staff, Note, vA_thread_signature)
abjad> print list(notes)
[Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8")]

abjad> vB_thread_signature = threadtools.component_to_thread_signature(vB)
abjad> notes = threadtools.iterate_thread_forward_in_expr(staff, Note, vB_thread_signature)
abjad> print list(notes)
[Note("c''8"), Note("b'8"), Note("a'4")]

abjad> vC_thread_signature = threadtools.component_to_thread_signature(vC)
abjad> notes = threadtools.iterate_thread_forward_in_expr(staff, Note, vC_thread_signature)
abjad> print list(notes)
[Note("c''4"), Note("c''4")]
```

In each case we are passing a different **thread signature** to the `iterate_thread_forward_in_expr()` iterator, so each case returns a different list of notes.

We can see that the thread signature of each voice is indeed different by printing it:

```
abjad> vA_thread_signature = threadtools.component_to_thread_signature(vA)
abjad> vA_thread_signature
<      root: Staff-8112592 (8112592) *      score: * staffgroup: *      staff: Staff-8112592 *

abjad> vB_thread_signature = threadtools.component_to_thread_signature(vB)
abjad> vB_thread_signature
<      root: Staff-8108496 (8108496) *      score: * staffgroup: *      staff: Staff-8108496 *

abjad> vC_thread_signature = threadtools.component_to_thread_signature(vC)
abjad> vC_thread_signature
<      root: Staff-8108496 (8108496) *      score: * staffgroup: *      staff: Staff-8108496 *
```

And by comparing them with the binary equality operator:

```
abjad> vA_thread_signature == vB_thread_signature
False
abjad> vA_thread_signature == vC_thread_signature
False
abjad> vB_thread_signature == vC_thread_signature
False
```

To allow Abjad to treat the content of, say, voices vA and vB as belonging together, we explicitly define a thread between them. To do this all we need to do is give both voices the same name:

```
abjad> vA.name = 'piccolo'
abjad> vB.name = 'piccolo'
```

Now vA and vB and all their content belong to the same thread:

```
abjad> vA_thread_signature == vB_thread_signature
False
```

Note how the thread signatures have changed:

```

abjad> vA_thread_signature = threadtools.component_to_thread_signature(vA)
abjad> print vA_thread_signature
      root: Staff-8108496 (8108496)
      score:
staffgroup:
  staff: Staff-8108496
  voice: Voice-piccolo
  self: Voice-piccolo

abjad> vB_thread_signature = threadtools.component_to_thread_signature(vB)
abjad> print vB_thread_signature
      root: Staff-8108496 (8108496)
      score:
staffgroup:
  staff: Staff-8108496
  voice: Voice-piccolo
  self: Voice-piccolo

abjad> vC_thread_signature = threadtools.component_to_thread_signature(vC)
abjad> print vC_thread_signature
      root: Staff-8108496 (8108496)
      score:
staffgroup:
  staff: Staff-8108496
  voice: Voice-8108384
  self: Voice-8108384

```

And how the `threadtools.iterate_thread_forward_in_expr()` function returns all the notes belonging to both `vA` and `vB` when passing it the full staff and the thread signature of `vA`:

```

abjad> notes = threadtools.iterate_thread_forward_in_expr(staff, Note, vA_thread_signature)
abjad> print list(notes)
[Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8"), Note("c'8"), Note("b'8"), Note("a'4")]

```

Now the slur spanner can be applied to voices `vA` and `vB`:

```

abjad> spannertools.SlurSpanner([vA, vB])

```

or directly to the notes returned by the `iterate_thread_forward_in_expr()` iteration tool, which are the notes belonging to both `vA` and `vB`:

```

abjad> notes = threadtools.iterate_thread_forward_in_expr(staff, Note, vA_thread_signature)
abjad> spannertools.SlurSpanner(list(notes))

```

```

abjad> show(staff)

```



10.3 Coda

We could have constructed this score in a simpler way with only two voices, one of them starting with a LilyPond skip:


```
abjad> vX = Voice(notetools.make_notes([5, 7, 9, 11, 12, 11, 9], [(1, 8)] * 6 + [(1, 4)]))
abjad> vY = Voice([skiptools.Skip((2, 4))] + Note(12, (1, 4)) * 2)
abjad> marktools.LilyPondCommandMark('voiceOne')(vX[0])
abjad> marktools.LilyPondCommandMark('voiceTwo')(vY[0])
abjad> staff = Staff([vX, vY])
abjad> staff.is_parallel = True
```



UNDERSTANDING LILYPOND GROBS

LilyPond models music notation as a collection of graphic objects or grobs.

11.1 Grobs control typography

LilyPond grobs control the typographic details of the score:

```
\new Staff {  
  c'4 (  
  d'4 )  
  e'4 (  
  f'4 )  
  g'4 (  
  a'4 )  
  g'2  
}
```



In the example above LilyPond creates a grob for every printed glyph. This includes the clef and time signature as well as the note heads, stems and slurs. If the example included beams, articulations or an explicit key signature then LilyPond would create grobs for those as well.

11.2 Grobs can be overridden

You can change the appearance of LilyPond grobs with grob overrides:

```
\new Staff \with {  
  \override NoteHead #'color = #red  
  \override StaffSymbol #'color = #blue  
  \override Stem #'color = #red  
} {  
  c'4 (  
  d'4 )  
  e'4 (  
  f'4 )  
  g'4 (  
  a'4 )  
}
```



11.3 Check the LilyPond docs

New grobs are added to LilyPond from time to time.

For a complete list of LilyPond grobs see the [LilyPond documentation](#).

UNDERSTANDING ABJAD OVERRIDES

12.1 Grob-override component plug-ins

All Abjad containers have a grob-override plug-in:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")

abjad> staff.override.staff_symbol.color = 'blue'

abjad> staff.override
LilyPondGrobOverrideComponentPlugIn(staff_symbol__color = 'blue')
```

All Abjad leaves have a grob-override plug-in, too:

```
abjad> leaf = staff[-1]

abjad> leaf.override.note_head.color = 'red'
abjad> leaf.override.stem.color = 'red'

abjad> leaf.override
LilyPondGrobOverrideComponentPlugIn(note_head__color = 'red', stem__color = 'red')
```

And so do Abjad spanners:

```
abjad> slur = spannertools.SlurSpanner(staff[:])

abjad> slur.override.slur.color = 'red'

abjad> slur.override
LilyPondGrobOverrideComponentPlugIn(slur__color = 'red')
```

12.2 Grob proxies

Grob-override plug-ins contain grob proxies:

```
abjad> leaf.override.note_head
LilyPondGrobProxy(color = 'red')

abjad> leaf.override.stem
LilyPondGrobProxy(color = 'red')
```

12.3 Dot-chained override syntax

The's dot-chained grob override syntax shown here results from the special way that the Abjad grob-override plug-in and grob proxy set and get their attributes.

TIME SIGNATURE MARKS BY EXAMPLE

In this tutorial is to take a deeper look at what happens when we attach time signature marks to staves and other score components. To work through the tutorial, enter each of the examples into the Abjad interpreter and study what comes back. At the end of the tutorial you'll understand how time signature marks are created. You'll also understand how the states of different objects change when time signature marks are attached and detached.

First we start by creating a staff full of notes:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4 g'2")
```

If we ask the Abjad interpreter about our staff reference Abjad will respond with the interpreter display of the object:

```
abjad> staff
Staff{5}
```

The 5 in `Staff{5}` shows that the staff contains 5 top-level components. The curly braces in `Staff{5}` show that the contents of the staff are to be read sequentially through time rather than in parallel.

Before we get to time signature marks let's take a moment and examine the state of the staff we've created. We can motivate this a bit by asking two questions:

1. what time signature is currently in effect for the staff we have just created?
2. **what is the time signature currently in effect for** the five notes contained within the staff we have just created?

The answer to both questions is the same: there is no time signature currently in effect for either our staff or for the five notes it contains.

We can see that this is the case with tools from the API:

```
abjad> contexttools.get_effective_time_signature(staff) is None
True
```

And:

```
abjad> for leaf in staff:
...     contexttools.get_effective_time_signature(leaf) is None
...
True
True
True
True
True
```

If we want, we can iterate both the staff and its leaves at one and the same time like this:

```
abjad> for component in componenttools.iterate_components_forward_in_expr(staff):
...     component, contexttools.get_effective_time_signature(component)
...
(Staff{5}, None)
(Note("c'4"), None)
(Note("d'4"), None)
(Note("e'4"), None)
(Note("f'4"), None)
(Note("g'2"), None)
```

This confirms the answer to our questions that there is not yet any time signature in effect for any component in our staff because we have not yet attached a time signature mark to any component in our staff.

So what happens if we format our staff and send it off to LilyPond to render as a PDF? Will LilyPond render the staff with a time signature? Without a time signature? Will LilyPond refuse to render the example at all?

We find out like this:

```
abjad> show(staff)
```



It turns out LilyPond defaults to a time signature of 4/4.

What's important to note here is that because we have not yet attached a time signature mark any component in our staff Abjad says “no effective time signature here” while LilyPond says “OK, I’ll default to 4/4 so we can get on with rendering your music.”

We can further confirm that this is the case by asking Abjad for the LilyPond format of our staff:

```
abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

The LilyPond format of our staff contains no LilyPond `\time` command. This is, again, because we have not yet attached a time signature mark to any component in our staff.

We can no practice attaching and detaching time signature marks to different components in our staff and study what happens as we do.

We’ll start with 3/4.

The easiest thing to do is to attach a time signature mark to the staff itself.

We’ll do this in two separate steps and study each step to understand exactly what’s going on.

First, we create a 3/4 time signature mark:

```
abjad> time_signature_mark = contexttools.TimeSignatureMark((3, 4))
```

If we ask the Abjad interpreter for the interpreter display of our time signature mark we get the following:

```
abjad> time_signature_mark
TimeSignatureMark((3, 4))
```


All this tells us is that we have in fact created a 3/4 time signature mark. Nothing too exciting yet. At this point our 3/4 time signature is not yet attached to anything. We could say that the “state” of our time signature mark is “unattached.” And we can see this like so:

```
abjad> time_signature_mark.start_component is None
True
```

What does it mean for a time signature mark to have ‘start_component’ equal to none? It means that the time signature isn’t yet attached to any score component anywhere.

So now we attach our time signature mark to our staff:

```
abjad> time_signature_mark.attach(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

Abjad responds immediately by returning the time signature mark we have just attached.

Notice that our time signature mark’s repr has changed. The repr of our 3/4 time signature mark now includes the repr of the staff to which we have just attached the time signature mark. That is to say that the repr of our time signature mark is `statat`.

Our time signature mark has transitioned from an “unattached” state to an “attached” state. We can see this like so:

```
abjad> time_signature_mark.start_component
Staff{5}
```

And our staff has likewise transitioned from a state of having no effective time signature to a state of having an effective time signature:

```
abjad> contexttools.get_effective_time_signature(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

And what about the leaves inside our staff? Do the leaves now “know” that they are governed by a 3/4 time signature?

Indeed they do:

```
abjad> for leaf in staff.leaves:
...     leaf, contexttools.get_effective_time_signature(leaf)
...
(Note("c'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("d'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("e'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("f'4"), TimeSignatureMark((3, 4))(Staff{5}))
(Note("g'2"), TimeSignatureMark((3, 4))(Staff{5}))
```

So to briefly resume:

What we just did was to:

1. create a time signature mark
2. attach the time signature to a score component

This 2-step pattern is always the same when dealing with context marks: create then attach.

(We will find out later that there are short-cuts for different parts of this process. Right now we’ve chosen to create in a first step and attach in a second step so that we can examine the changing states of the objects involved.)

Before moving on let’s look at the PDF corresponding to our staff:

```
abjad> show(staff)
```



And let's confirm what we see in the PDF in the staff's format:

```
abjad> f(staff)
\new Staff {
    \time 3/4
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

The staff's format now contains a LilyPond `\time` command because we have attached an Abjad time signature mark to the staff.

What we've just been through above will cover over 80% of what you'll ever wind up doing with time signature marks: creating them and attaching them directly to staves. But what if we wanna get rid of a time signature mark? Or what if the time signature will be changing all over the place? We cover those cases next.

Detaching a time signature mark is easy:

```
abjad> time_signature_mark.detach()
TimeSignatureMark((3, 4))
```

The Abjad returns the mark we have just detached. And, observing the repr of the time signature mark, we see that the time signature mark has again changed state: the time signature mark has transitioned from attached to unattached. We confirm this like so:

```
abjad> time_signature_mark.start_component is None
True
```

And also like so:

```
abjad> contexttools.get_effective_time_signature(staff) is None
True
```

Yup: our time signature mark knows nothing about our staff. And vice versa. This is good.

So now what if we want to set up a time signature of 2/4? That fits our music, too.

We have a couple of options.

We can simply create and attach a new time signature mark:

```
abjad> duple_time_signature_mark = contexttools.TimeSignatureMark((2, 4))
abjad> duple_time_signature_mark.attach(staff)
TimeSignatureMark((2, 4)) (Staff{5})

abjad> f(staff)
\new Staff {
    \time 2/4
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

```
abjad> show(staff)
```



Yup. That works.

On the other hand, we could simply reuse our previous 3/4 time signature mark.

To do this we'll first detach our 2/4 time signature mark ...

```
abjad> duple_time_signature_mark.detach()
```

```
abjad> duple_time_signature_mark.detach()
TimeSignatureMark((2, 4))
```

... confirm that our staff is now time signatureless ...

```
abjad> contexttools.get_effective_time_signature(staff) is None
True
```

```
abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

... reattach our previous 3/4 time signature ...

```
abjad> time_signature_mark.attach(staff)
```

```
abjad> time_signature_mark.attach(staff)
TimeSignatureMark((4, 4)) (Staff{5})
```

... change the numerator of our time signature mark ...

```
abjad> time_signature_mark.numerator = 2
```

... and check to make sure that everything is as it should be:

```
abjad> contexttools.get_effective_time_signature(staff)
TimeSignatureMark((2, 4)) (Staff{5})
abjad> time_signature_mark.start_component
Staff{5}
```

```
abjad> f(staff)
\new Staff {
    \time 2/4
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

```
abjad> show(staff)
```



And everything works as it should.

To change to, for example, 4/4 we change just change the time signature mark's numerator again:

```
abjad> time_signature_mark.numerator = 4
```

```
abjad> f(staff)
\new Staff {
    \time 4/4
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

But what if our time signature has a 2/4 pick-up?

The LilyPond command for pick-ups is `\partial`. Abjad time signature marks implement this as a read / write attribute:

```
abjad> time_signature_mark.partial = Duration(2, 4)
```

```
abjad> f(staff)
\new Staff {
    \partial 2
    \time 4/4
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

```
abjad> show(staff)
```



And what if time signature changes all over the place?

We'll use the trivial example of a measure in 4/4 followed by a measure in 2/4.

To do this we will need two time signature marks.

We've already got a 4/4 time signature mark attached to our staff:

```
abjad> f(staff)
\new Staff {
    \partial 2
    \time 4/4
    c' 4
    d' 4
    e' 4
}
```

```

        f'4
        g'2
    }

```

Let's get rid of the pick-up:

```
abjad> time_signature_mark.partial = None
```

```

abjad> f(staff)
\new Staff {
    \time 4/4
    c'4
    d'4
    e'4
    f'4
    g'2
}

```

Now what about the 2/4 time signature mark?

We create it in the usual way:

```

abjad> duple_time_signature_mark = contexttools.TimeSignatureMark((2, 4))
abjad> duple_time_signature_mark
TimeSignatureMark((2, 4))

```

But should we attach it? We can't attach our 2/4 time signature to our staff because we've already attached our 4/4 time signature to our staff. And it only makes sense to attach one time signature to any given score component.

Observe that we've built our score in a very straightforward way: we have a single staff that contains a (flat) sequence of notes. This means that we have only one choice for where to attach the new 2/4 time signature mark. And that is one the $g'2$ that comes on the downbeat of the second measure. We do that like this:

```
abjad> duple_time_signature_mark.attach(staff[4])
```

```

abjad> duple_time_signature_mark.attach(staff[4])
TimeSignatureMark((2, 4))(g'2)

```

```

abjad> f(staff)
\new Staff {
    \time 4/4
    c'4
    d'4
    e'4
    f'4
    \time 2/4
    g'2
}

```

```
abjad> show(staff)
```



And everything works as we would like.

Incidentally, `staff[4]` means the component sitting at index 4 inside our staff. Using the interpreter we can verify that this is $g'2$:

```
abjad> staff[4]
Note("g' 2")
```

Depending on how we had chosen to build our staff we would have had more options for where to attach our 2/4 time signature mark. If, for example, we had chosen to populate our staff with a series of measures then it's possible we could have attached our 2/4 time signature to a measure instead of a note.

That covers the vast majority of things you'll do with time signature marks.

But before we stop we should mention another useful API function and then talk about some short-cuts.

First an API function to detach ALL context marks attaching to a component:

We call the function a first time:

```
abjad> contexttools.detach_context_marks_attached_to_component(staff)
(TimeSignatureMark((4, 4)),)
```

```
abjad> f(staff)
\new Staff {
    c' 4
    d' 4
    e' 4
    f' 4
    \time 2/4
    g' 2
}
```

And then a second time:

```
:: abjad> contexttools.detach_context_marks_attached_to_component(staff[4]) (TimeSignatureMark((2, 4)),)
```

```
abjad> f(staff)
\new Staff {
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

Now there are now context marks of any sort attached to our staff or to the notes in our staff.

Be careful with this function, though: it removes *all* context marks. So even though we just used the function to remove time signature marks, it also would have removed any clef marks or tempo marks if we had had those attached to our score, too.

And now for the short-cuts:

Our staff currently has no time signature marks attached:

```
abjad> f(staff)
\new Staff {
    c' 4
    d' 4
    e' 4
    f' 4
    g' 2
}
```

So to recreate our 3/4 time signature we can do this ...

```
abjad> time_signature_mark = contexttools.TimeSignatureMark((3, 4))
```

... and then use a short-cut to avoid calling `time_signature_mark.attach()` like this:

```
abjad> time_signature_mark(staff)
TimeSignatureMark((3, 4))(Staff{5})
```

```
abjad> f(staff)
\new Staff {
    \time 3/4
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

What's going on here is that all context marks implement the special `__call__()` method as a short-cut for `attach()`. What is the special `__call__()` method? The `__call__()` method is what makes a function, class or any other Python object callable. The statement `time_signature_mark(staff)` has parentheses in it because the time signature mark is callable; and the time signature mark is callable because all context marks implement the special `__call__()` method.

Note too that all context marks understand an *empty call* as a short-cut for `detach()`. Like this:

```
abjad> time_signature_mark()
TimeSignatureMark((3, 4))
```

```
abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

The empty call made against the time signature mark causes the time signature mark to detach from its start component.

The fact that context marks implement the special `__call__()` method as a short-cut for `attach()` means that context marks can be created and attached in a single line:

```
abjad> contexttools.TimeSignatureMark((2, 4))(staff)
TimeSignatureMark((2, 4))(Staff{5})
```

```
abjad> f(staff)
\new Staff {
    \time 2/4
    c'4
    d'4
    e'4
    f'4
    g'2
}
```

What's going on here?

What's going on is that `contexttools.TimeSignatureMark((2, 4))` creates a time signature mark in the usual way and that – immediately after this – the newly created time signature mark is available for us to call it against our staff.

This last short-cut form of ...

```
abjad> contexttools.TimeSignatureMark((2, 4))(staff)
```

... is the usual way that you will see context marks of all sorts presented in the docs.

Reference manual

ANNOTATIONS

Annotate components with user-specific information for future use.

Annotations do not impact formatting.

14.1 Creating annotations

Use mark tools to create annotations:

```
abjad> annotation = marktools.Annotation('special pitch', pitchtools.NamedChromaticPitch('bs'))

abjad> annotation
Annotation('special pitch', NamedChromaticPitch('bs'))
```

14.2 Attaching annotations to a component

Attach annotations to any component with `attach()`:

```
abjad> note = Note("c'4")
abjad> annotation.attach(note)

abjad> annotation
Annotation('special pitch', NamedChromaticPitch('bs'))(c'4)

abjad> another_annotation = marktools.Annotation('special pitch', pitchtools.NamedChromaticPitch('bs'))
abjad> another_annotation.attach(note)

abjad> another_annotation
Annotation('special pitch', NamedChromaticPitch('bs'))(c'4)
```

14.3 Getting the annotations attached to a component

Use mark tools to get all the annotations attached to a component:

```
abjad> marktools.get_annotations_attached_to_component(note)
(Annotation('special pitch', NamedChromaticPitch('bs'))(c'4), Annotation('special pitch', NamedChromaticPitch('bs'))(c'4))
```

14.4 Detaching annotations from a component one at a time

Use `detach()` to detach annotations from a component one at a time:

```
abjad> annotation.detach()

abjad> annotation
Annotation('special pitch', NamedChromaticPitch('bs'))
```

14.5 Detaching all annotations attached to a component at once

Or use mark tools to detach all annotations attached to a component at once:

```
abjad> print marktools.detach_annotations_attached_to_component(note)
(Annotation('special pitch', NamedChromaticPitch('bs')),)

abjad> marktools.get_annotations_attached_to_component(note)
()
```

14.6 Inspecting the component to which an annotation is attached

Use `start_component` to inspect the component to which an annotation is attached:

```
abjad> annotation.attach(note)

abjad> annotation.start_component
Note("c'4")
```

14.7 Inspecting annotation name

Use `name` to get the name of any annotation:

```
abjad> annotation.name
'special pitch'
```

14.8 Inspecting annotation value

And use `value` to get the value of any annotation:

```
abjad> annotation.value
NamedChromaticPitch('bs')
```

ARTICULATIONS

Articulations model staccati, marcati, tenuti and other symbols. Articulations attach notes, rests or chords.

15.1 Creating articulations

Use `marktools` to create articulations:

```
articulation = marktools.Articulation('turn')

abjad> articulation
Articulation('turn')
```

15.2 Attaching articulations to a leaf

Use `attach()` to attach articulations to a leaf:

```
abjad> staff = Staff([])
abjad> key_signature = contexttools.KeySignatureMark('g', 'major')
abjad> key_signature.attach(staff)
time_signature = contexttools.TimeSignatureMark((2, 4), partial = Duration(1, 8))
abjad> time_signature.attach(staff)

abjad> staff.extend("d'8 f'8 a'8 d''8 f''8 gs'4 r8 e'8 gs'8 b'8 e''8 gs''8 a'4")

abjad> articulation.attach(staff[5])

abjad> show(staff)
```



(The example is based on Haydn's piano sonata number 42, Hob. XVI/27.)

15.3 Attaching articulations to many notes and chords at once

Use `marktools` to attach articulations to many notes and chords at one time:

```
abjad> marktools.attach_articulations_to_notes_and_chords_in_expr(staff[:6], ['.'])
```

```
abjad> show(staff)
```



15.4 Getting the articulations attached to a leaf

Use `marktools` to get the articulations attached to a leaf:

```
abjad> marktools.get_articulations_attached_to_component(staff[5])
(Articulation('turn')(gs'4), Articulation('.') (gs'4))
```

15.5 Detaching articulations from a leaf one at a time

Detach articulations by hand with `detach()`:

```
abjad> articulation.detach()
```

```
abjad> articulation
-\turn
```

```
abjad> show(staff)
```



15.6 Detaching all articulations attached to a leaf at once

Use `marktools` to detach all articulations attached to a leaf at once:

```
abjad> staff[0]
Note("d'8")
```

```
abjad> marktools.detach_articulations_attached_to_component(staff[0])
```

```
abjad> show(staff)
```



15.7 Inspecting the leaf to which an articulation is attached

Use `start_component` to inspect the component to which an articulation is attached:

```
abjad> articulation = marktools.Articulation('turn')
abjad> articulation.attach(staff[-1])
```

```
abjad> show(staff)
```



```
abjad> articulation.start_component
Note("a' 4")
```

15.8 Understanding the interpreter display of an articulation that is not attached to a leaf

The interpreter display of an articulation that is not attached to a leaf contains three parts:

```
abjad> articulation = marktools.Articulation('staccato')
```

```
abjad> articulation
abjad> print repr(articulation)
Articulation('staccato')
```

Articulation tells you the articulation's class.

'staccato' tells you the articulation's name.

If you set the direction string of the articulation then that will appear, too:

```
abjad> articulation.direction_string = '^'
```

```
abjad> articulation
abjad> print repr(articulation)
Articulation('staccato', '^')
```

15.9 Understanding the interpreter display of an articulation that is attached to a leaf

The interpreter display of an articulation that is attached to a leaf contains four parts:

```
abjad> articulation.attach(staff[-1])
```

```
abjad> articulation
abjad> print repr(articulation)
Articulation('staccato', '^') (a'4)
```

```
abjad> show(staff)
```



Articulation tells you the articulation's class.

'staccato' tells you the articulation's name.

'^' tells you the articulation's direction string.

(a"4) tells you the component to which the articulation is attached.

If you set the direction string of the articulation to none then the direction will no longer appear:

```
abjad> articulation.direction_string = None
```

```
abjad> articulation
Articulation('staccato')(a'4)
```

15.10 Understanding the string representation of an articulation

The string representation of an articulation comprises two parts:

```
abjad> str(articulation)
'\staccato'
```

– tells you the articulation's direction string.

staccato tells you the articulation's name.

15.11 Inspecting the LilyPond format of an articulation

Get the LilyPond input format of an articulation with `format`:

```
abjad> articulation.format
'\staccato'
```

Use `f()` as a short-cut to print the LilyPond format of an articulation:

```
abjad> f(articulation)
-\staccato
```

15.12 Controlling whether an articulation appears above or below the staff

Set `direction_string` to '^' to force an articulation to appear above the staff:

```
abjad> articulation.direction_string = '^'
```

```
abjad> show(staff)
```



Set `direction_string` to '_' to force an articulation to appear below the staff:

```
abjad> articulation.direction_string = '_'
```

```
abjad> show(staff)
```



Set `direction_string` to `none` to allow LilyPond to position an articulation automatically:

```
abjad> articulation.direction_string = None
```

```
abjad> show(staff)
```



15.13 Getting and setting the name of an articulation

Set the name of an articulation to change the symbol an articulation prints:

```
abjad> articulation.name = 'staccatissimo'
```

```
abjad> show(staff)
```



15.14 Copying articulations

Use `copy.copy()` to copy an articulation:

```
abjad> import copy
```

```
abjad> articulation_copy_1 = copy.copy(articulation)
```

```
abjad> articulation_copy_1
Articulation('staccatissimo')
```

```
abjad> articulation_copy_1.attach(staff[1])
```

```
abjad> show(staff)
```



Or use `copy.deepcopy()` to do the same thing.

15.15 Comparing articulations

Articulations compare equal with equal direction names and direction strings:

```
abjad> articulation.name
'staccatissimo'
abjad> articulation.direction_string
None

abjad> articulation_copy_1.name
'staccatissimo'
abjad> articulation_copy_1.direction_string
None

abjad> articulation == articulation_copy_1
True
```

Otherwise articulations do not compare equal.

15.16 Overriding attributes of the LilyPond script grob

Override attributes of the LilyPond script grob like this:

```
abjad> staff.override.script.color = 'red'

abjad> f(staff)
\new Staff \with {
  \override Script #'color = #red
} {
  \key g \major
  \partial 8
  \time 2/4
  d'8
  f'8 -\staccatissimo -\staccato
  a'8 -\staccato
  d''8 -\staccato
  f''8 -\staccato
  gs'4 -\staccato
  r8
  e'8
  gs'8
  b'8
  e''8
  gs''8
  a'4 -\staccatissimo -\turn
}
```

```
abjad> show(staff)
```



See the LilyPond documentation for a list of script grob attributes available.

CHORDS

16.1 Making chords from a LilyPond input string

You can make chords from a LilyPond input string:

```
abjad> chord = Chord("<c' d' bf'>4")
```

```
abjad> show(chord)
```



16.2 Making chords from chromatic pitch numbers and duration

You can also make chords from chromatic pitch numbers and duration:

```
abjad> chord = Chord([0, 2, 10], Duration(1, 4))
```

```
abjad> show(chord)
```



16.3 Getting all the written pitches of a chord at once

You can get all the written pitches of a chord at one time:

```
abjad> chord.written_pitches  
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"), NamedChromaticPitch("bf'"))
```

Abjad returns a read-only tuple of named chromatic pitches.

16.4 Getting the written pitches of a chord one at a time

You can get the written pitches of a chord one at a time:

```
abjad> chord.written_pitches[0]  
NamedChromaticPitch("c' ")
```

Chords index the pitch they contain starting from 0 (just like tuples and lists).

16.5 Adding one pitch to a chord at a time

Use `append()` to add one note to a chord.

You can add a pitch to a chord with a chromatic pitch number:

```
abjad> chord.append(9)
```

```
abjad> show(chord)
```



Or you can add a pitch to a chord with a chromatic pitch name:

```
abjad> chord.append("df' ")
```

```
abjad> show(chord)
```



Chords sort their pitches every time you add a new one.

This means you can add pitches to your chord in any order.

16.6 Adding many pitches to a chord at once

Use `extend()` to add many pitches to a chord.

You can use chromatic pitch numbers:

```
abjad> chord.extend([3, 4, 14])
```

```
abjad> show(chord)
```



Or you can use chromatic pitch names:

```
abjad> chord.extend(["g' ", "af' "])
```

```
abjad> show(chord)
```



16.7 Deleting pitches from a chord

Delete pitches from a chord with `del()`:

```
abjad> del(chord[0])
```

```
abjad> show(chord)
```



```
abjad> del(chord[0])
```

```
abjad> show(chord)
```



Negative indices work too:

```
abjad> del(chord[-1])
```

```
abjad> show(chord)
```



16.8 Formatting chords

Get the LilyPond input format of any Abjad object with `format`:

```
abjad> chord.format
<ef' e' a' bf' df'' d'' g''>4
```

Use `f()` as a short-cut to print the LilyPond input format of any Abjad object:

```
abjad> f(chord)
<ef' e' a' bf' df'' d'' g''>4
```

16.9 Working with note heads

Most of the time you will work with the pitches of a chord. But you can get the note heads of a chord, too:

```
abjad> chord.note_heads
(NoteHead("ef'"), NoteHead("e'"), NoteHead("a'"), NoteHead("bf'"), NoteHead("df''"), NoteHead("d''"))
```

This is useful when you want to apply LilyPond overrides to note heads in a chord one at a time:

```
abjad> chord[2].tweak.color = 'red'
abjad> chord[3].tweak.color = 'blue'
abjad> chord[4].tweak.color = 'green'
```

```
abjad> f(chord)
<
    ef'
    e'
    \tweak #'color #red
    a'
    \tweak #'color #blue
    bf'
    \tweak #'color #green
    df''
    d''
    g''
>4
```

```
abjad> show(chord)
```



16.10 Working with empty chords

Abjad allows empty chords:

```
abjad> chord = Chord([], Duration(1, 4))
Chord('<>4')
```

Abjad formats empty chords, too:

```
abjad> f(chord)
<>4
```

But if you pass empty chords to `show()` LilyPond will complain because empty chords don't constitute valid LilyPond input.

When you are done working with an empty chord you can add pitches back into it chord in any of the ways described above:

```
abjad> chord.extend(["gf'", "df''", "g''"])
```

```
abjad> show(chord)
```



CONTAINERS

17.1 Creating containers

Create a container with components:

```
abjad> container = Container([Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16")])
```

```
abjad> show(container)
```



Or with a note-entry string:

```
abjad> container = Container("ds'16 cs'16 e'16 c'16 d'2 ~ d'8")
```

```
abjad> show(container)
```



17.2 Inspecting music

Return the components in a container with `music`:

```
abjad> container.music  
(Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16"), Note("d'2"), Note("d'8"))
```

Or with a special call to `__getslice__`:

```
abjad> container[:]  
[Note("ds'16"), Note("cs'16"), Note("e'16"), Note("c'16"), Note("d'2"), Note("d'8")]
```

17.3 Inspecting length

Get the length of a container with `len()`:

```
abjad> len(container)
6
```

17.4 Inspecting duration

Contents duration equals the sum of the duration of everything inside the container:

```
abjad> container.contents_duration
Duration(7, 8)
```

17.5 Adding one component to the end of a container

Add one component to the end of a container with `append`:

```
abjad> container.append(Note("af' 32"))

abjad> show(container)
```



17.6 Adding many components to the end of a container

Add many components to the end of a container with `extend`:

```
abjad> container.extend([Note("c' ' 32"), Note("a' 32")])

abjad> show(container)
```



17.7 Finding the index of a component

Find the index of a component with `index`:

```
abjad> note = container[7]

abjad> container.index(note)
7
```

17.8 Inserting a component by index

Insert a component by index with `insert`:

```
abjad> container.insert(-3, Note("g'32"))
```

```
abjad> show(container)
```



17.9 Removing a component by index

Remove a component by index with `pop`:

```
abjad> container.pop(-1)
```

```
abjad> show(container)
```



17.10 Removing a component by reference

Remove a component by reference with `remove`:

```
abjad> container.remove(container[-1])
```

```
abjad> show(container)
```



Note: `__getslice__`, `__setslice__` and `__delslice__` remain to be documented.

17.11 Naming containers

You can name Abjad containers:

```
abjad> flute_staff = Staff("c'8 d'8 e'8 f'8")
abjad> flute_staff.name = 'Flute'
abjad> violin_staff = Staff("c'8 d'8 e'8 f'8")
abjad> violin_staff.name = 'Violin'
abjad> staff_group = scoretools.StaffGroup([flute_staff, violin_staff])
abjad> score = Score([staff_group])
```

Container names appear in LilyPond input:

```
abjad> f(score)
\new Score <<
  \new StaffGroup <<
    \context Staff = "Flute" {
      c'8
      d'8
      e'8
      f'8
    }
    \context Staff = "Violin" {
      c'8
      d'8
      e'8
      f'8
    }
  >>
>>
```

And make it easy to retrieve containers later:

```
abjad> componenttools.get_first_component_in_expr_with_name(score, 'Flute')
Staff-"Flute">{4}
```

But container names do not appear in notational output:

```
abjad> show(score)
```



17.12 Understanding { } and << >> in LilyPond

LilyPond uses curly { } braces to wrap a stream of musical events that are to be engraved one after the other:

```
\new Voice {
  e''4
  f''4
  g''4
  g''4
  f''4
  e''4
  d''4
  d''4 \fermata
}
```



LilyPond uses skeleton << >> braces to wrap two or more musical expressions that are to be played at the same time:


```

\new Staff <<
  \new Voice {
    \voiceOne
    e''4
    f''4
    g''4
    g''4
    f''4
    e''4
    d''4
    d''4 \fermata
  }
  \new Voice {
    \voiceTwo
    c''4
    c''4
    b'4
    c''4
    c''8
    b'8
    c''4
    b'4
    b'4 \fermata
  }
>>

```



The examples above are both LilyPond input.

The most common use of LilyPond { } is to group a potentially long stream of notes and rests into a single expression.

The most common use of LilyPond << >> is to group a relatively smaller number of note lists together polyphonically.

17.13 Understanding sequential and parallel containers

Abjad implements LilyPond { } and << >> in the container `is_parallel` attribute.

Some containers set `is_parallel` to false at initialization:

```

staff = Staff([])
staff.is_parallel
False

```

Other containers set `is_parallel` to true:

```

score = Score([])
score.is_parallel
True

```

17.14 Changing sequential and parallel containers

Set `is_parallel` by hand as necessary:

```
voice_1 = Voice(r"e''4 f''4 g''4 g''4 f''4 e''4 d''4 d''4  ermata")
voice_2 = Voice(r"c''4 c''4 b'4 c''4 c''8 b'8 c''4 b'4 b'4  ermata")
abjad> staff = Staff([voice_1, voice_2])
abjad> staff.is_parallel = True
abjad> marktools.LilyPondCommandMark('voiceOne')(voice_1)
abjad> marktools.LilyPondCommandMark('voiceTwo')(voice_2)
abjad> show(staff)
```



The staff in the example above is set to parallel after initialization to create a type of polyphonic staff:

```
abjad> f(staff)
\new Staff <<
  \new Voice {
    \voiceOne
    e''4
    f''4
    g''4
    g''4
    f''4
    e''4
    d''4
    d''4 -\fermata
  }
  \new Voice {
    \voiceTwo
    c''4
    c''4
    b'4
    c''4
    c''8
    b'8
    c''4
    b'4
    b'4 -\fermata
  }
>>
```

17.15 Overriding containers

The symbols below are black with fixed thickness and predetermined spacing:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
abjad> slur_1 = spannertools.SlurSpanner(staff[:2])
abjad> slur_2 = spannertools.SlurSpanner(staff[2:4])
abjad> slur_3 = spannertools.SlurSpanner(staff[4:6])
```

```
abjad> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}
```

```
abjad> show(staff)
```



But you can override LilyPond grobs to change the look of Abjad containers:

```
abjad> staff.override.staff_symbol.color = 'blue'
```

```
abjad> f(staff)
\new Staff \with {
    \override StaffSymbol #'color = #blue
} {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}
```

```
abjad> show(staff)
```



17.16 Overriding containers' contents

You can override LilyPond grobs to change the look of containers' contents, too:

```
abjad> staff.override.note_head.color = 'red'
abjad> staff.override.stem.color = 'red'
```

```
abjad> f(staff)
\new Staff \with {
    \override NoteHead #'color = #red
    \override StaffSymbol #'color = #blue
    \override Stem #'color = #red
} {
    c'4 (
    d'4 )
}
```

```

    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}

```

```
abjad> show(staff)
```



17.17 Removing container overrides

Delete grob overrides you no longer want:

```
abjad> del(staff.override.staff_symbol)
```

```

abjad> f(staff)
\new Staff \with {
    \override NoteHead #'color = #red
    \override Stem #'color = #red
} {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}

```

```
abjad> show(staff)
```



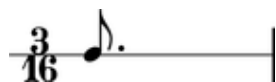
DURATIONS

18.1 Introduction

Abjad publishes information about many durated score objects.

Notes, rests, chords and skips carry some duration attributes:

```
abjad> note = Note(0, (3, 16))
abjad> measure = Measure((3, 16), [note])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> note.written_duration
Duration(3, 16)
```



Tuplets, measures, voices, staves and the other containers carry duration attributes, too:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(3, 16), Note(0, (1, 16)) * 5)
abjad> measure = Measure((3, 16), [tuplet])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> tuplet.multiplier
Duration(3, 5)
```



The next chapters document core duration concepts in Abjad.

18.2 Assignability

Western notation readily admits rational values like $1/4$. But values like $1/5$ notate only with tuplet brackets or special time signatures. Abjad formalizes the difference between rationals like $1/4$ and $1/5$ in the definition of rational assignability.

Rational values n/d are assignable when and only when numerator n is of the form $k(2^u - 1)$ and denominator d is of the form 2^v . In this definition u and v must be nonnegative integers, k must be a positive integer, and j must be either 0 or 1.

Abjad initializes notes, rests and chords with assignable durations only.

18.3 Prolation

Abjad uses **prolation** as a cover term for rhythmic augmentation and diminution. Augmentation increases the duration of notes, rests and chords. Diminution does the opposite. Western notation employs tuplet brackets and special types of time signature to effect prolation.

18.3.1 Tuplet prolation

Tuplets prolate their contents:

```
abjad> tuplet = Tuplet(Fraction(5, 4), 4 * Note("c'8"))
abjad> staff = stafftools.RhythmicStaff([Measure((5, 8), [tuplet])])
abjad> spannertools.BeamSpanner(tuplet)
abjad> show(staff)
```



```
abjad> note = tuplet[0]
abjad> note.written_duration
Duration(1, 8)

abjad> note.prolation
Fraction(5, 4)

abjad> note.prolated_duration
Duration(5, 32)
```

Notes here with written duration 1/8 carry prolation factor 5/4 and prolated duration 5/32.

18.3.2 Meter prolation

Time signatures in western notation usually carry a denominator equal to a nonnegative integer power of 2. Abjad calls these conventional meters **binary meters**. Denominators equal to integers other than integer powers of 2 are also possible. Such **nonbinary meters** rhythmically diminish the contents of the measures they govern:

```
abjad> measure = Measure((4, 10), Note(0, (1, 8)) * 4)
abjad> spannertools.BeamSpanner(measure)
abjad> staff = stafftools.RhythmicStaff([measure])
```



```
abjad> note = staff.leaves[0]
abjad> note.prolation
Fraction(4, 5)

abjad> note.prolated_duration
Duration(1, 8)

abjad> note.prolation
Fraction(4, 5)
```

```
abjad> note.prolated_duration
Duration(1, 10)
```

Notes here with written duration $1/8$ carry prolation factor $4/5$ and prolated duration $1/10$.

18.3.3 The prolation chain

Tuplets nest and combine freely with different types of meter. When two or more **prolation donors** conspire, the prolation factor they collectively bestow on leaf-level music equals the cumulative product of all prolation factors in the **prolation chain**. All durated components carry a prolation chain:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(4, 8), Note(0, (1, 16)) * 7)
abjad> spannertools.BeamSpanner(tuplet)
abjad> measure = Measure((4, 10), [tuplet])
abjad> staff = stafftools.RhythmicStaff([measure])
```



```
abjad> measure.multiplier
Fraction(4, 5)

abjad> note = measure.leaves[0]
abjad> note.prolation
Duration(32, 35)

abjad> note.prolated_duration
Duration(2, 35)
```

Notes here with written duration $1/16$ carry prolated duration $2/35$.

Note: Western notation does not recognize tuplet brackets carrying one-to-one ratios. Such **trivial tuplets** may, however, be useful during different stages of composition, and Abjad allows them for that reason. Trivial tuplets carry **zero prolation**. Zero-prolated tuplets neither augment nor diminish the music they contain.

Note: Abjad implements one of two competing nonbinary **meter-interpretation schemes**. The first, **implicit meter-interpretation** given here, follows, for example, Ferneyhough, in that nonbinary meters prolate the contents of the measures they govern implicitly, ie, without recourse to tuplet brackets. The second, **explicit meter-interpretation**, which we find in, for example, Sciarrino, insists instead on the presence of some tuplet bracket, usually engraved in some broken or incomplete way. The implicit meter-interpretation that Abjad implements differs from the explicit meter-interpretation native to LilyPond. Abjad will eventually implement both implicit and explicit meter-interpretation, settable on a container-by-container basis.

Note: Nonbinary meter n/d rhythmically diminishes the contents of the measure it governs by a factor j/k , with $k=d$, and with j equal to the greatest integer power of 2 less than d . That is, $j=2^{**int(\log_2(d))}$.

18.4 Duration types

Abjad publishes duration information about all score components.

18.4.1 Written duration

Abjad uses **written duration** to refer to the face value of notes, rests and chords prior to prolation. Abjad written duration corresponds to the informal names most frequently used when talking about note duration.

These sixteenth notes are worth a sixteenth of a whole note:

```
abjad> measure = Measure((5, 16), Note(0, (1, 16)) * 5)
abjad> spannertools.BeamSpanner(measure)
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> note = measure[0]
abjad> note.written_duration
Duration(1, 16)
```



These sixteenth notes are worth more than a sixteenth of a whole note:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(5, 16), Note(0, (1, 16)) * 4)
abjad> spannertools.BeamSpanner(tuplet)
abjad> measure = Measure((5, 16), [tuplet])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> note = tuplet[0]
abjad> note.written_duration
Duration(1, 16)
```



The notes in these examples are ‘sixteenth notes’ that carry different prolated durations. Abjad written duration captures the fact that the note heads and flag counts of the two examples match.

Written duration is a user-assignable rational number. Users can assign and reassign the written duration of notes, rests and chords at initialization and at any time during the life of the note, rest or chord. Written durations must be assignable; see the chapter on *assignability* for details. Note that Abjad containers do not carry written duration.

18.4.2 Prolated duration

Prolation refers to the duration-scaling effects of tuplets and special types of time signature. Prolation is a way of thinking about the contribution that musical structure makes to the duration of score objects. All durated Abjad objects carry a prolated duration. Prolated duration is an emergent property of notes, tuplets and other durated objects. The prolated duration of notes, rests and chords equals the product of the written duration and prolation of those objects. The prolated duration of tuplets, measures and other containers equals the the container’s duration interface multiplied by the container’s prolation.

18.4.3 Contents duration

Abjad defines the **contents duration** of tuplets, measures, voices, staves and other containers equal to the sum of the **preprolated duration** of each of the elements in the container.

The measure here contains two eighth notes and tuplet. These elements carry preprolated durations equal to 1/8, 1/8 and 2/8, respectively:


```
abjad> notes = Note(0, (1, 8)) * 2
abjad> spannertools.BeamSpanner(notes)
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), Note(0, (1, 8)) * 3)
abjad> spannertools.BeamSpanner(tuplet)
abjad> measure = Measure((4, 8), notes + [tuplet])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> measure.contents_duration
Duration(1, 2)
```



The contents duration of the measure here equals $1/8 + 1/8 + 2/8 = 4/8$.

18.4.4 Target duration

Abjad defines the target duration of fixed-duration tuplets equal to composer-settable duration to which the tuplet prolates its contents.

This fixed-duration tuplet carries a target duration equal to $4/8$:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(4, 8), Note(0, (1, 8)) * 5)
abjad> spannertools.BeamSpanner(tuplet)
abjad> measure = Measure((4, 8), [tuplet])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> print tuplet.contents_duration
5/8
abjad> tuplet.target_duration
Duration(1, 2)
```



The tuplet contents sum to $5/8$. But tuplet target duration always equals $4/8$.

18.4.5 Multiplied duration

Abjad defines the multiplied duration of notes, rests and chords equal to the product of written duration and leaf multiplier.

The first two notes below carry leaf multipliers equal to $2/1$:

```
abjad> notes = Note(0, (1, 16)) * 4
abjad> notes[0].duration_multiplier = Fraction(2, 1)
abjad> notes[1].duration_multiplier = Fraction(2, 1)
abjad> measure = Measure((3, 8), notes)
abjad> spannertools.BeamSpanner(measure)
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> note = measure[0]
abjad> note.written_duration
Duration(1, 16)
```



```
abjad> note.duration_multiplier
Fraction(2, 1)
```

```
abjad> note.written_duration * note.duration_multiplier
Duration(1, 8)
abjad> note.multiplied_duration
Duration(1, 8)
```

The written duration of these first two notes equals 1/16 and so the multiplied duration of these first two notes equals $1/16 * 2/1 = 1/8$.

18.5 Duration initialization

Durated Abjad classes initialize duration from arguments in the form (n, d) with numerator n and denominator d .

```
abjad> note = Note(0, (3, 16))
```



Durated classes include notes, rests, chords, skips, tuplets and measures.

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), Note(0, (1, 8)) * 3)
abjad> spannertools.BeamSpanner(tuplet)
```



Abjad restricts notes, rests, chords and skips to durations like 3/16 that can be written with dots, beams and flags without ties or brackets. Abjad allows arbitrary positive durations like 5/8 for tuplets and measures.

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(5, 8), Note(0, (1, 8)) * 4)
abjad> spannertools.BeamSpanner(tuplet)
```



Abjad supports breves.

```
abjad> note = Note(0, (2, 1))
```



And longas.

```
abjad> note = Note(0, (4, 1))
```



Note: The restriction that the written durations of notes, rests, chords and skips be expressible with some combination of dots, flags and beams without recourse to ties and brackets generalizes to the condition of `note_head` assignability.

Values (n, d) are `note_head`-assignable when and only when (1) d is a nonnegative integer power of 2; (2) n is either a nonnegative integer power of 2 or is a nonnegative integer power of 2, minus 1; and (3) n/d is less than or equal to 8. Condition (3) captures the fact that LilyPond provides no glyph with greater duration than the maxima (equal to eight whole notes).

Note: Integer forms like 4 as a substitute for $(4, 1)$ in `Note(0, (4, 1))` are undocumented but allowed.

Note: Abjad allows maxima `note_heads` as in `Note(0, (8, 1))`. LilyPond implements a *maxima* command but does not supply a corresponding glyph for the `note_head`.

18.6 LilyPond multipliers

LilyPond provides an asterisk `*` operator to scale the durations of notes, rests and chords by arbitrarily positive rational values. LilyPond multipliers are invisible and generate no typographic output of their own. However, while independent from the typographic output, LilyPond multipliers do factor in in calculations of duration and time.

Abjad implements LilyPond multipliers as the settable `duration.multiplier` attribute of notes, rests and chords.

```
abjad> note = Note("c'4")
abjad> note.duration_multiplier = Fraction(1, 2)
abjad> note.duration_multiplier
Fraction(1, 2)
```

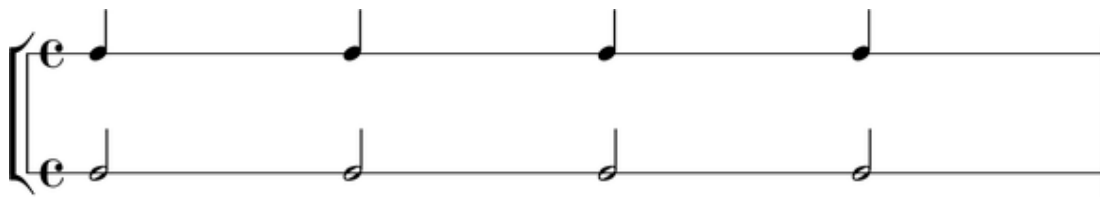
```
abjad> f(note)
c'4 * 1/2
```

Abjad also implements a `duration.multiplied` attribute to examine the duration of a note, rest or chord as affected by the multiplier.

```
abjad> note.multiplied_duration
Duration(1, 8)
```

LilyPond multipliers give the half notes here multiplied durations equal to a quarter note.

```
abjad> notes = Note("c'4") * 4
abjad> multiplied_note = Note(0, (1, 2))
abjad> multiplied_note.duration_multiplier = Fraction(1, 2)
abjad> multiplied_notes = multiplied_note * 4
abjad> top = stafftools.RhythmicStaff(notes)
abjad> bottom = stafftools.RhythmicStaff(multiplied_notes)
abjad> staves = scoretools.StaffGroup([top, bottom])
```



Note: Abjad models multiplication fundamentally differently than prolation. See the chapter on *Prolation* for more information.

Note: The LilyPond multiplication `*` operator differs from the Abjad multiplication `*` operator. LilyPond multiplication scales duration of LilyPond notes, rests and chords. Abjad multiplication copies Abjad containers and leaves.

18.7 Duration interfaces compared

type	core	leaf	container	measure	tuplet	fd tuple	fm tuple
contents	–	–	R	R	R	R	R
multiplied	–	R	–	–	–	R	R
multiplier	–	RW	–	R	R	R	RW
preprolated	R	R	R	R	R	R	R
prolated	R	R	R	R	R	R	R
prolation	R	R	R	R	R	R	R
target	–	–	–	–	–	RW	–
written	–	RW	–	–	–	–	–

The table contains a total of only four settable duration attributes, divided among only three classes. Durated Abjad classes offer up many read-only duration attributes but very few read-write duration attributes.

All classes carry all three prolation-related attributes because all classes can nest inside containers. It is possible, for example, to nest an entire voice within a fixed-duration tuple.

Note: Leaf multipliers and tuple multipliers differ.

INSTRUMENT MARKS

Instrument marks appear as markup in the left margin of your score.

19.1 Creating instrument marks

Use `contexttools` to create instrument marks:

```
abjad> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')

abjad> instrument_mark
InstrumentMark('Violin ', 'Vn. ')
```

19.2 Attaching instrument marks to a component

Use `attach()` to attach any mark to a component:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")

abjad> instrument_mark.attach(staff)

abjad> show(staff)
```



19.3 Getting the instrument mark attached to a component

Use `contexttools` to get the instrument mark attached to a component:

```
abjad> contexttools.get_instrument_mark_attached_to_component(staff)
InstrumentMark('Violin ', 'Vn. ')(Staff{4})
```

19.4 Getting the instrument in effect for a component

Or to get the instrument currently in effect for a component:

```
abjad> contexttools.get_effective_instrument(staff[1])
InstrumentMark('Violin ', 'Vn. ') (Staff{4})
```

19.5 Detaching instrument marks from a component one at a time

Use `detach()` to detach instrument marks from a component one at a time:

```
abjad> instrument_mark.detach()

abjad> instrument_mark
InstrumentMark('Violin ', 'Vn. ')

abjad> show(staff)
```



19.6 Detaching all instrument marks attached to a component at once

Or use `contexttools` to detach instrument marks all at once:

```
abjad> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')
abjad> instrument_mark.attach(staff)

abjad> instrument_mark
InstrumentMark('Violin ', 'Vn. ') (Staff{4})

abjad> show(staff)
```



```
abjad> contexttools.detach_instrument_marks_attached_to_component(staff)

abjad> instrument_mark
InstrumentMark('Violin ', 'Vn. ')

abjad> show(staff)
```



19.7 Inspecting the component to which an instrument mark is attached

Use `start_component` to inspect the component to which an instrument mark is attached:

```
abjad> instrument_mark = contexttools.InstrumentMark('Flute ', 'Fl. ')
abjad> instrument_mark.attach(staff)

abjad> show(staff)
```



```
abjad> instrument_mark.start_component
Staff{4}
```

19.8 Inspecting the instrument name of an instrument mark

Use `instrument_name_markup` to get the instrument name of any instrument mark:

```
abjad> instrument_mark.instrument_name_markup
Markup('Flute ')
```

19.9 Inspecting the short instrument name of an instrument mark

And use `short_instrument_name_markup` to get the short instrument name of any instrument mark:

```
abjad> instrument_mark.short_instrument_name_markup
Markup('Fl. ')
```


20.1 Reopening Abjad PDFs

After you build a piece of notation and open with `show()` you will usually close the resulting PDF and continue working, changing your output notation in an iterative and incremental way.

```
abjad> staff = Staff(construct.scale(8))
abjad> show(staff)
```

But what if you need to go back and open the resulting PDF again? Abjad provides `pdf()` for precisely this purpose. Type the following at the Abjad prompt to open the most recent PDF written by Abjad.

```
abjad> pdf()
```

If you want to open not the next-to-most recent PDF generated by Abjad, pass in a `-1`. And for the next-to-next-to-most recent, pass in a `-2`, and so on.

20.2 Looking at LilyPond output

Abjad generates a LilyPond `.ly` file for every Abjad expression that you build and `show()`. To look at these LilyPond `.ly` files that Abjad builds behind the scenes, use `ly()`.

```
abjad> ly()

% Abjad revision 2362
% 2009-06-25 10:30

\version "2.12.2"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/scm/abjad.scm"

\new Staff {
  c'8
  d'8
  e'8
  f'8
  g'8
  a'8
  b'8
  c''8
}
```

Abjad opens the LilyPond `.ly` file in your favorite text editor.

These LilyPond `.ly` files that Abjad generates all have the same basic structure. The current version of Abjad and the date appear first, followed by the mandatory LilyPond version string and LilyPond directives for English note names and the default Abjad `.scm` file. The remainder of the file is reserved for the LilyPond input code corresponding to the expression you just built in Abjad.

When you are done looking at the LilyPond `.ly` file quit your text editor to return to the Abjad interpreter.

20.3 Looking at the LilyPond log

If things go wrong when you call `show()` or one of the other Abjad functions that call LilyPond behind the scenes, it may be helpful to examine the output that LilyPond writes to the LilyPond log.

```
abjad> log()

GNU LilyPond 2.12.2
Processing '1420.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Layout output to '1420.ps'...
Converting to './1420.pdf'...
```

This is the normal output that LilyPond generates every time you call the program behind. When you are done looking at the LilyPond log, quit your text editor to return to the Abjad interpreter.

LILYPOND COMMAND MARKS

LilyPond command marks allow you to attach arbitrary LilyPond commands to Abjad score components.

21.1 Creating LilyPond command marks

Use `marktools` to create LilyPond command marks:

```
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('bar "||"', 'after')

abjad> lilypond_command_mark
LilyPondCommandMark('bar "||"')
```

21.2 Attaching LilyPond command marks to Abjad components

Use `attach()` to attach a LilyPond command mark to any Abjad component:

```
abjad> import copy
abjad> staff = Staff([])
abjad> key_signature = contexttools.KeySignatureMark('f', 'major')
abjad> key_signature.attach(staff)
abjad> staff.extend(iotools.parse_lilypond_input_string("d''16 ( c''16 fs''16 g''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("f''16 ( e''16 d''16 c''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("cs''16 ( d''16 f''16 d''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("a'8 b'8"))
abjad> staff.extend(iotools.parse_lilypond_input_string("d''16 ( c''16 fs''16 g''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("f''16 ( e''16 d''16 c''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("cs''16 ( d''16 f''16 d''16 )"))
abjad> staff.extend(iotools.parse_lilypond_input_string("a'8 b'8 c''2"))

abjad> lilypond_command_mark.attach(staff[-2])

abjad> show(staff)
```



21.3 Getting the LilyPond command marks attached to an Abjad component

Use `marktools` to get the `lilypond_command_marks` attached to a leaf:

```
abjad> marktools.get_lilypond_command_marks_attached_to_component(staff[-2])
(LilyPondCommandMark('bar "||"' (b'8),)
```

21.4 Detaching LilyPond command marks from components one at a time

Use `detach()` to detach LilyPond command marks one at a time:

```
abjad> lilypond_command_mark.detach()
```

```
abjad> lilypond_command_mark
LilyPondCommandMark('bar "||"')
```

```
abjad> show(staff)
```



21.5 Detaching all LilyPond command marks attached to a component at once

Use `marktools` to detach all LilyPond command marks attached to a component at once:

```
abjad> lilypond_command_mark_1 = marktools.LilyPondCommandMark('bar "||"', 'closing')
abjad> lilypond_command_mark_1.attach(staff[-2])
```

```
abjad> lilypond_command_mark_2 = marktools.LilyPondCommandMark('bar "||"', 'closing')
abjad> lilypond_command_mark_2.attach(staff[-16])
```

```
abjad> show(staff)
```



```
abjad> marktools.detach_lilypond_command_marks_attached_to_component(staff[-16])
```

```
abjad> show(staff)
```



21.6 Inspecting the component to which a LilyPond command mark is attached

Use `start_component` to inspect the component to which a LilyPond command mark is attached:

```
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('bar "|"', 'closing')
abjad> lilypond_command_mark.attach(staff[-2])
```

```
abjad> show(staff)
```



```
abjad> lilypond_command_mark.start_component
Note("b'8")
```

21.7 Getting and setting the command name of a LilyPond command mark

Set the `command_name` of a LilyPond command mark to change the LilyPond command a LilyPond command mark prints:

```
abjad> lilypond_command_mark.command_name = 'bar "|."'
```

```
abjad> show(staff)
```



21.8 Copying LilyPond commands

Use `copy.copy()` to copy a LilyPond command mark:

```
abjad> import copy
```

```
abjad> lilypond_command_mark_copy_1 = copy.copy(lilypond_command_mark)
```

```
abjad> lilypond_command_mark_copy_1
LilyPondCommandMark('bar "|."')
```

```
abjad> lilypond_command_mark_copy_1.attach(staff[-1])
```

```
abjad> show(staff)
```



Or use `copy.deepcopy()` to do the same thing.

21.9 Comparing LilyPond command marks

LilyPond command marks compare equal with equal command names:

```
abjad> lilypond_command_mark.command_name
'bar "|".'
```

```
abjad> lilypond_command_mark_copy_1.command_name
'bar "|".'
```

```
abjad> lilypond_command_mark == lilypond_command_mark_copy_1
True
```

Otherwise LilyPond command marks do not compare equal.

LILYPOND COMMENTS

LilyPond comments begin with the % sign. Abjad models LilyPond comments as marks.

22.1 Creating LilyPond comments

Use `marktools` to create LilyPond comments:

```
abjad> comment_1 = marktools.LilyPondComment('This is a LilyPond comment before a note.', 'before')

abjad> comment_1
LilyPondComment('This is a LilyPond comment before a note.')
```

22.2 Attaching LilyPond comments to leaves

Attach LilyPond comments to a note, rest or chord with `attach()`:

```
abjad> note = Note("cs''4")
abjad> show(note)
```



```
abjad> comment_1.attach(note)

abjad> f(note)
% This is a LilyPond comment before a note.
cs''4
```

You can add LilyPond comments before, after or to the right of any leaf.

22.3 Attaching LilyPond comments to containers

Use `attach()` to attach LilyPond comments to a container:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> show(staff)
```



```
abjad> staff_comment_1 = marktools.LilyPondComment('Here is a LilyPond comment before the staff.', 'before')
abjad> staff_comment_2 = marktools.LilyPondComment('Here is a LilyPond comment in the staff opening.', 'opening')
abjad> staff_comment_3 = marktools.LilyPondComment('Here is another LilyPond comment in the staff opening.', 'opening')
abjad> staff_comment_4 = marktools.LilyPondComment('LilyPond comment in the staff closing.', 'closing')
abjad> staff_comment_5 = marktools.LilyPondComment('LilyPond comment after the staff.', 'after')

abjad> staff_comment_1.attach(staff)
abjad> staff_comment_2.attach(staff)
abjad> staff_comment_3.attach(staff)
abjad> staff_comment_4.attach(staff)
abjad> staff_comment_5.attach(staff)

abjad> f(staff)
% Here is a LilyPond comment before the staff.
\new Staff {
    % Here is a LilyPond comment in the staff opening.
    % Here is another LilyPond comment in the staff opening.
    c'8
    d'8
    e'8
    f'8
    % LilyPond comment in the staff closing.
}
% LilyPond comment after the staff.
```

You can add LilyPond comments before, after, in the opening or in the closing of any container.

22.4 Getting the LilyPond comments attached to a component

Use `marktools` to get all the LilyPond comments attached to a component:

```
abjad> marktools.get_lilypond_comments_attached_to_component(note)
(LilyPondComment('This is a LilyPond comment before a note.')(cs''4),)
```

Abjad returns a tuple of zero or more LilyPond comments.

22.5 Detaching LilyPond comments from a component one at a time

Use `detach()` to detach LilyPond comments from a component one at a time:

```
abjad> comment_1 = marktools.get_lilypond_comments_attached_to_component(note)[0]

abjad> comment_1.detach()
LilyPondComment('This is a LilyPond comment before a note.')

abjad> f(note)
cs''4
```


22.6 Detaching all LilyPond comments attached to a component at once

Or use `marktools` to detach all LilyPond comments attached to a component at once:

```
abjad> for comment in marktools.get_lilypond_comments_attached_to_component(staff): print comment
LilyPondComment('Here is a LilyPond comment before the staff.')(Staff{4})
LilyPondComment('Here is a LilyPond comment in the staff opening.')(Staff{4})
LilyPondComment('Here is another LilyPond comment in the staff opening.')(Staff{4})
LilyPondComment('LilyPond comment in the staff closing.')(Staff{4})
LilyPondComment('LilyPond comment after the staff.')(Staff{4})

abjad> marktools.detach_lilypond_comments_attached_to_component(staff)

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

22.7 Inspecting the component to which a LilyPond comment is attached

Use `start_component` to inspect the component to which a LilyPond comment is attached:

```
abjad> comment_1.attach(note)

abjad> comment_1.start_component
Note("cs' '4")
```

22.8 Inspecting contents string of a LilyPond comment

Use `contents_string` to inspect the written contents of a LilyPond comment:

```
abjad> comment_1.contents_string
'This is a LilyPond comment before a note.'
```


LILYPOND FILES

23.1 Making LilyPond files

Make a basic LilyPond input file with the `lilyfiletools` package:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> lilypond_file = lilyfiletools.make_basic_lilypond_file(staff)

abjad> lilypond_file
LilyPondFile(Staff{4})
```

23.2 Inspecting file output

LilyPond input files that you create this way come equipped with many attributes that appear in file output:

```
abjad> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

23.3 Setting default paper size

Set default LilyPond paper size like this:

```
abjad> lilypond_file.default_paper_size = '11x17', 'landscape'
```

```
abjad> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

#(set-default-paper-size "11x17" 'landscape)

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

23.4 Setting global staff size

Set global staff size like this:

```
abjad> lilypond_file.global_staff_size = 16

abjad> f(lilypond_file)
% Abjad revision 4746
% 2011-09-04 17:36

\version "2.15.9"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

#(set-default-paper-size "11x17" 'landscape)
#(set-global-staff-size 16)

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

MEASURES

24.1 Understanding measures in LilyPond

In LilyPond you specify time signatures by hand and LilyPond creates measures automatically:

```
\new Staff {  
  \time 3/8  
  c'8  
  d'8  
  e'8  
  d'8  
  e'8  
  f'8  
  \time 2/4  
  g'4  
  e'4  
  f'4  
  d'4  
  c'2  
}
```



Here LilyPond creates five measures from two time signatures. This happens because behind-the-scenes LilyPond time-keeping tells the program when measures start and stop and how to draw the barlines that come between them.

24.2 Understanding measures in Abjad

Measures are optional in Abjad, too, and you may omit them in favor of time signatures:

```
abjad> staff = Staff("c'8 d'8 e'8 d'8 e'8 f'8 g'4 e'4 f'4 d'4 c'2")  
  
abjad> contexttools.TimeSignatureMark((3, 8))(staff)  
abjad> contexttools.TimeSignatureMark((2, 4))(staff[6])  
  
abjad> show(staff)
```



But you may also include explicit measures in the Abjad scores you build. The following sections explain how.

24.3 Creating measures

Create a measure with a meter and music:

```
abjad> measure = Measure((3, 8), "c'8 d'8 e'8")
```

```
abjad> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}
```

```
abjad> show(measure)
```



24.4 Working with dynamic measures

Dynamic measures adjust their time signatures on the fly as you add and remove music.

Create dynamic measures without a time signature:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")
```

```
abjad> show(measure)
```



24.5 Adding music to dynamic measures

Add music to dynamic measures the same as to all containers:

```
abjad> measure.extend([Note("fs'8"), Note("gs'8")])
```

```
abjad> show(measure)
```



24.6 Removing music from dynamic measures

Remove music from dynamic measures the same as with other containers:

```
abjad> del(measure[1:3])
```

```
abjad> show(measure)
```



24.7 Setting the denominator of dynamic measures

You can set the denominator of dynamic measures to any integer power of 2:

```
abjad> measure.denominator = 32
```

```
abjad> show(measure)
```



24.8 Suppressing the meter of dynamic measures

You can temporarily suppress the meter of dynamic measures:

```
abjad> measure.suppress_meter = True
```

```
abjad> f(measure)
```

```
{
    c' 8
    fs' 8
    gs' 8
}
```

LilyPond will engrave the last active meter.

24.9 Working with anonymous measures

Anonymous determine their time signatures on the fly and then hide them at format time.

Create anonymous measures without a time signature:

```
abjad> measure = measuretools.AnonymousMeasure("c' 8 d' 8 e' 8")
```

```
abjad> show(measure)
```



24.10 Adding music to anonymous measures

Add music to anonymous measures the same as to other containers:

```
abjad> measure.extend([Note("fs'8"), Note("gs'8")])
```

```
abjad> show(measure)
```



24.11 Removing music from anonymous measures

Remove music from anonymous measure the same as from other containers:

```
abjad> del(measure[1:3])
```

```
abjad> show(measure)
```



NOTES

25.1 Making notes from a string

You can make notes from string:

```
abjad> note = Note("c' 4")
```

```
abjad> show(note)
```



25.2 Making notes from chromatic pitch number and duration

You can also make notes from chromatic pitch number and duration:

```
abjad> note = Note(0, Duration(1, 4))
```

```
abjad> show(note)
```



(You even use `Note("c' 4")` to create notes with numbers alone.)

25.3 Getting the written pitch of notes

You can get the written pitch of notes:

```
abjad> note.written_pitch  
NamedChromaticPitch("c' ")
```

25.4 Changing the written pitch of notes

And you can change the written pitch of notes:

```
abjad> note.written_pitch = "cs' "
```



(You can use `note.written_pitch = 1` to change pitch with numbers, too.)

25.5 Getting the duration attributes of notes

Get the written duration of notes like this:

```
abjad> note.written_duration
Duration(1, 4)
```

Which is usually the same as preprolated duration:

```
abjad> note.preprolated_duration
Duration(1, 4)
```

And prolated duration:

```
abjad> note.prolated_duration
Duration(1, 4)
```

Except for notes inside a tuplet:

```
abjad> tuplet = Tuplet(Fraction(2, 3), [Note("c'4"), Note("d'4"), Note("e'4")])
```

```
abjad> show(tuplet)
```



```
abjad> note = tuplet[0]
```

Tupletted notes carry written duration:

```
abjad> note.written_duration
Duration(1, 4)
```

Prolation:

```
abjad> note.prolation
Fraction(2, 3)
```

And prolated duration that is the product of the two:

```
abjad> note.prolated_duration
Duration(1, 6)
```

25.6 Changing the written duration of notes

You can change the written duration of notes:

```
abjad> tuplet[0].written_duration = Duration(1, 8)
abjad> tuplet[1].written_duration = Duration(1, 8)
abjad> tuplet[2].written_duration = Duration(1, 8)
```

```
abjad> show(tuplet)
```



Other duration attributes are read-only.

25.7 Overriding notes

The notes below are black with fixed thickness and predetermined spacing:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
abjad> slur_1 = spannertools.SlurSpanner(staff[:2])
abjad> slur_2 = spannertools.SlurSpanner(staff[2:4])
abjad> slur_3 = spannertools.SlurSpanner(staff[4:6])
```

```
abjad> f(staff)
```

```
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}
```

```
abjad> show(staff)
```



But you can override LilyPond grobs to change the look of notes, rests and chords:

```
abjad> staff[-1].override.note_head.color = 'red'
abjad> staff[-1].override.stem.color = 'red'
```

```
abjad> f(staff)
```

```
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    \once \override NoteHead #'color = #red
    \once \override Stem #'color = #red
    g'2
}
```

```
abjad> show(staff)
```



25.8 Removing note overrides

Delete grob overrides you no longer want:

```
abjad> del(staff[-1].override.stem)
```

```
abjad> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    \once \override NoteHead #'color = #red
    g'2
}
```

```
abjad> show(staff)
```



PITCHES

Named chromatic pitches are the everyday pitches attached to notes and chords:

```
abjad> note = Note("cs''8")

abjad> note.written_pitch
NamedChromaticPitch("cs''")
```

26.1 Creating pitches

Use pitch tools to create named chromatic pitches:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs''")

abjad> named_chromatic_pitch
NamedChromaticPitch("cs''")
```

26.2 Inspecting the name of a pitch

Use `str()` to get the name of named chromatic pitches:

```
abjad> str(named_chromatic_pitch)
cs''
```

26.3 Inspecting the octave of a pitch

Get the octave number of named chromatic pitches with `octave_number`:

```
abjad> named_chromatic_pitch.octave_number
5
```

26.4 Working with pitch deviation

Use deviation to model the fact that two pitches differ by a fraction of a semitone:

```
abjad> note_1 = Note(24, (1, 2))
abjad> note_2 = Note(24, (1, 2))
abjad> staff = Staff([note_1, note_2])
```

```
abjad> show(staff)
```



```
abjad> note_2.written_pitch = pitchtools.NamedChromaticPitch(24, deviation = -31)
```

The pitch of the the first note is greater than the pitch of the second:

```
abjad> note_1.written_pitch > note_2.written_pitch
True
```

Use markup to include indications of pitch deviation in your score:

```
abjad> markuptools.Markup(note_2.written_pitch.deviation_in_cents, 'up')(note_2)
```



26.5 Sorting pitches

Named chromatic pitches sort by octave, diatonic pitch-class and accidental, in that order:

```
abjad> pitchtools.NamedChromaticPitch('es') < pitchtools.NamedChromaticPitch('ff')
True
```

26.6 Comparing pitches

Compare named chromatic pitches to each other:

```
abjad> named_chromatic_pitch_1 = pitchtools.NamedChromaticPitch("c'")
abjad> named_chromatic_pitch_2 = pitchtools.NamedChromaticPitch("d'")
```

```
abjad> named_chromatic_pitch_1 == named_chromatic_pitch_2
False
```

```
abjad> named_chromatic_pitch_1 != named_chromatic_pitch_2
True
```

```
abjad> named_chromatic_pitch_1 > named_chromatic_pitch_2
False
```

```
abjad> named_chromatic_pitch_1 < named_chromatic_pitch_2
True
```

```
abjad> named_chromatic_pitch_1 >= named_chromatic_pitch_2
False
```

```
abjad> named_chromatic_pitch_1 <= named_chromatic_pitch_2
True
```

26.7 Converting one type of pitch to another

Convert any named chromatic pitch to a named diatonic pitch:

```
abjad> named_chromatic_pitch.named_diatonic_pitch
NamedDiatonicPitch("c'")
```

To a numbered chromatic pitch:

```
abjad> named_chromatic_pitch.numbered_chromatic_pitch
NumberedChromaticPitch(13)
```

Or to a numbered diatonic pitch:

```
abjad> named_chromatic_pitch.numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

26.8 Converting pitches to pitch-classes

Convert any named chromatic pitch to a named chromatic pitch-class:

```
abjad> named_chromatic_pitch.named_chromatic_pitch_class
NamedChromaticPitchClass('cs')
```

To a named diatonic pitch-class:

```
abjad> named_chromatic_pitch.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

To a numbered chromatic pitch-class:

```
abjad> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Or to a numbered diatonic pitch-class:

```
abjad> named_chromatic_pitch.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

26.9 Copying pitches

Use `copy.copy()` to copy named chromatic pitches:

```
abjad> import copy

abjad> copy.copy(named_chromatic_pitch)
NamedChromaticPitch("cs'")
```

Or use `copy.deepcopy()` to do the same thing.

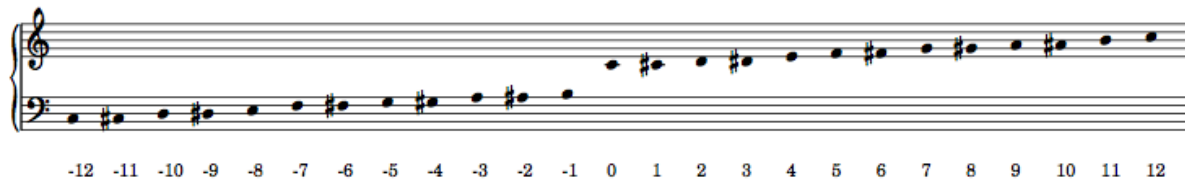
26.10 Accidental abbreviations

Abjad abbreviates accidentals according to the LilyPond `english.ly` module:

accidental name	abbreviation
quarter sharp	‘qs’
quarter flat	‘qf’
sharp	‘s’
flat	‘f’
three-quarters sharp	‘tqs’
three-quarters flat	‘tqf’
double sharp	‘ss’
double flat	‘ff’

26.11 Chromatic pitch numbers

Abjad numbers chromatic pitches by semitone with middle C set equal to 0:



The code to generate this table is as follows:

```
score, treble_staff, bass_staff = scoretools.make_empty_piano_score()
duration = Fraction(1, 32)

treble = measuretools.AnonymousMeasure([])
bass = measuretools.AnonymousMeasure([])

treble_staff.append(treble)
bass_staff.append(bass)

pitches = range(-12, 12 + 1)

configurationtools.set_default_accidental_spelling('sharps')

for i in pitches:
    note = Note(i, duration)
    rest = Rest(duration)
    clef = pitchtools.suggest_clef_for_named_chromatic_pitches([note.pitch])
    if clef == contexttools.ClefMark('treble'):
        treble.append(note)
        bass.append(rest)
    else:
        treble.append(rest)
        bass.append(note)
    diatonic_pitch_number = str(note.pitch.numbered_chromatic_pitch)
    markuptools.Markup(diatonic_pitch_number, 'down')(bass[-1])

score.override.rest.transparent = True
score.override.stem.stencil = False
```



```
show(score, 'paris.ly')
```

26.12 Diatonic pitch numbers

Abjad numbers diatonic pitches by staff space with middle C set equal to 0:



The code to generate this table is as follows:

```
score, treble_staff, bass_staff = scoretools.make_empty_piano_score()
duration = Fraction(1, 32)

treble = measuretools.AnonymousMeasure([])
bass = measuretools.AnonymousMeasure([])

treble_staff.append(treble)
bass_staff.append(bass)

pitches = []
diatonic_pitches = [0, 2, 4, 5, 7, 9, 11]

pitches.extend([-24 + x for x in diatonic_pitches])
pitches.extend([-12 + x for x in diatonic_pitches])
pitches.extend([0 + x for x in diatonic_pitches])
pitches.extend([12 + x for x in diatonic_pitches])
pitches.append(24)
configurationtools.set_default_accidental_spelling('sharps')

for i in pitches:
    note = Note(i, duration)
    rest = Rest(duration)
    clef = pitchtools.suggest_clef_for_named_chromatic_pitches([note.pitch])
    if clef == contexttools.ClefMark('treble'):
        treble.append(note)
        bass.append(rest)
    else:
        treble.append(rest)
        bass.append(note)
    diatonic_pitch_number = abs(note.pitch.numbered_diatonic_pitch)
    markuptools.Markup(diatonic_pitch_number, 'down')(bass[-1])

score.override.rest.transparent = True
score.override.stem.stencil = False

show(score, 'paris.ly')
```

26.13 Octave designation

Abjad designates octaves with both numbers and ticks:

Octave notation	Tick notation
C7	c'','',''
C6	c'','','
C5	c'',''
C4	c''
C3	c'
C2	c,
C1	c,,

26.14 Accidental spelling

Abjad chooses between enharmonic spellings at pitch-initialization according to the following table:

Chromatic pitch-class number	Chromatic pitch-class name (default)
0	C
1	C#
2	D
3	Eb
4	E
5	F
6	F#
7	G
8	Gb
9	A
10	Bb
11	B

```
abjad> staff = Staff([Note(n, (1, 8)) for n in range(12)])
abjad> show(staff)
```



Use pitch tools to respell with sharps:

```
abjad> pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps(staff)
abjad> show(staff)
```



Or flats:

```
abjad> pitchtools.respell_named_chromatic_pitches_in_expr_with_flats(staff)
abjad> show(staff)
```



WORKING WITH LISTS OF NUMBERS

Python provides a built-in `list` class that you can use to carry around almost anything. The examples here show how to create a list of numbers and then do things with the numbers in the list.

Create a list with square brackets.

```
abjad> my_list = [23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3]
abjad> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3]
```

Use `len()` to find the number of elements in any list.

```
abjad> len(my_list)
12
```

Use `append()` to add one element to a list.

```
abjad> my_list.append(5)
abjad> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3, 5]
```

Use `extend()` to extend one list with the contents of another.

```
abjad> my_other_list = [19, 11, 4, 10, 12]
abjad> my_list.extend(my_other_list)
abjad> my_list
[23, 7, 10, 18, 13, 20, 3, 2, 18, 9, 14, 3, 5, 19, 11, 4, 10, 12]
```

Use `reverse()` to reverse the elements in a list.

```
abjad> my_list.reverse()
abjad> my_list
[12, 10, 4, 11, 19, 5, 3, 14, 9, 18, 2, 3, 20, 13, 18, 10, 7, 23]
```

You can return a single value from a list with a numeric index.

```
abjad> my_list[0]
12
abjad> my_list[1]
10
abjad> my_list[2]
4
```

You can return many values from a list with slice notation.

```
abjad> my_list[:4]
[12, 10, 4, 11]
```

More information on these and all other operations defined on the built-in Python `list` is available in the [Python tutorial](#).

RESTS

28.1 Making rests from strings

You can make rests from a string:

```
abjad> rest = Rest('r8')
```

```
abjad> show(rest)
```



28.2 Making rests from durations

You can also make rests from a duration:

```
abjad> rest = Rest(Duration(1, 4))
```

```
abjad> show(rest)
```



(You can even use `Rest((1, 8))` to make rests from a duration pair.)

28.3 Getting the duration attributes of rests

Get the written duration of rests like this:

```
abjad> rest.written_duration  
Duration(1, 4)
```

Which is usually the same as preprolated duration:

```
abjad> rest.preprolated_duration  
Duration(1, 4)
```

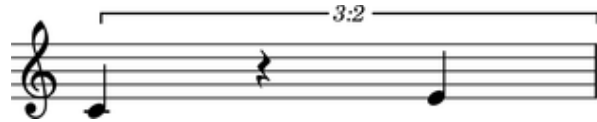
And prolated duration:

```
abjad> rest.prolated_duration
Duration(1, 4)
```

Except for rests inside a tuplet:

```
abjad> tuplet = Tuplet(Fraction(2, 3), [Note("c'4"), Rest('r4'), Note("e'4")])
```

```
abjad> show(tuplet)
```



```
abjad> rest = tuplet[1]
```

Tupletted rests carry written duration:

```
abjad> rest.written_duration
Duration(1, 4)
```

Prolation:

```
abjad> rest.prolation
Fraction(2, 3)
```

And prolated duration that is the product of the two:

```
abjad> rest.prolated_duration
Duration(1, 6)
```

28.4 Changing the written duration of rests

You can change the written duration of notes and rests:

```
abjad> tuplet[0].written_duration = Duration(1, 8)
abjad> tuplet[1].written_duration = Duration(1, 8)
abjad> tuplet[2].written_duration = Duration(1, 8)
```

```
abjad> show(tuplet)
```



Other duration attributes are read-only.

SCORES

29.1 Creating scores

Create a score like this:

```
abjad> treble_staff_1 = Staff("e'4 d'4 e'4 f'4 g'1")
abjad> treble_staff_2 = Staff("c'2. b8 a8 b1")

abjad> score = Score([treble_staff_1, treble_staff_2])

abjad> show(score)
```



29.2 Inspecting score music

Return score components with `music`:

```
abjad> score.music
(Staff{5}, Staff{4})
```

29.3 Inspecting score length

Get score length with `len()`:

```
abjad> len(score)
2
```

29.4 Inspecting score duration

Score contents duration is equal to the duration of the longest component in score:

```
abjad> score.contents_duration
Duration(2, 1)
```

29.5 Adding one component to the bottom of a score

Add one component to the bottom of a score with `append`:

```
abjad> bass_staff = Staff("g4 f4 e4 d4 d1")
abjad> contexttools.ClefMark('bass')(bass_staff)
```

```
abjad> score.append(bass_staff)
```

```
abjad> show(score)
```



29.6 Finding the index of a score component

Find the index of a score component with `index`:

```
abjad> score.index(treble_staff_1)
0
```

29.7 Removing a score component by index

Use `pop` to remove a score component by index:

```
abjad> score.pop(1)
```

```
abjad> show(score)
```



29.8 Removing a score component by reference

Remove a score component by reference with `remove`:

```
abjad> score.remove(treble_staff_1)
```

```
abjad> show(score)
```



29.9 Testing score containment

Use `in` to find out whether a score contains a given component:

```
abjad> treble_staff_1 in score
False
```

```
abjad> treble_staff_2 in score
False
```

```
abjad> bass_staff in score
True
```

29.10 Naming scores

You can name Abjad scores:

```
abjad> score.name = 'Example Score'
```

Score names appear in LilyPond input:

```
abjad> f(score)
\context Score = "Example Score" <<
  \new Staff {
    \clef "bass"
    g4
    f4
    e4
    d4
    d1
  }
>>
```

But do not appear in notational output:

```
abjad> show(score)
```



SPANNERS

30.1 Overriding spanners

The symbols below are black with fixed thickness and predetermined spacing:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4 g'4 a'4 g'2")
abjad> slur_1 = spannertools.SlurSpanner(staff[:2])
abjad> slur_2 = spannertools.SlurSpanner(staff[2:4])
abjad> slur_3 = spannertools.SlurSpanner(staff[4:6])
```

```
abjad> f(staff)
\new Staff {
    c'4 (
    d'4 )
    e'4 (
    f'4 )
    g'4 (
    a'4 )
    g'2
}
```

```
abjad> show(staff)
```



But you can override LilyPond grobs to change the look of spanners:

```
abjad> slur_1.override.slur.color = 'red'
abjad> slur_3.override.slur.color = 'red'
```

```
abjad> f(staff)
\new Staff {
    \override Slur #'color = #red
    c'4 (
    d'4 )
    \revert Slur #'color
    e'4 (
    f'4 )
    \override Slur #'color = #red
    g'4 (
    a'4 )
}
```

```
\revert Slur #'color
g'2
}
```

```
abjad> show(staff)
```



30.2 Overriding the components to which spanners attach

You can override LilyPond grobs to change spanners' contents:

```
abjad> slur_2.override.slur.color = 'blue'
abjad> slur_2.override.note_head.color = 'blue'
abjad> slur_2.override.stem.color = 'blue'
```

```
abjad> f(staff)
\new Staff {
  \override Slur #'color = #red
  c'4 (
  d'4 )
  \revert Slur #'color
  \override NoteHead #'color = #blue
  \override Slur #'color = #blue
  \override Stem #'color = #blue
  e'4 (
  f'4 )
  \revert NoteHead #'color
  \revert Slur #'color
  \revert Stem #'color
  \override Slur #'color = #red
  g'4 (
  a'4 )
  \revert Slur #'color
  g'2
}
```

```
abjad> show(staff)
```



30.3 Removing spanner overrides

Delete grob overrides you no longer want:

```
abjad> del(slur_1.override.slur)
abjad> del(slur_3.override.slur)
```

```
abjad> f(staff)
\new Staff {
  c'4 (
  d'4 )
  \override NoteHead #'color = #blue
  \override Slur #'color = #blue
  \override Stem #'color = #blue
  e'4 (
  f'4 )
  \revert NoteHead #'color
  \revert Slur #'color
  \revert Stem #'color
  g'4 (
  a'4 )
  g'2
}

abjad> show(staff)
```



STAVES

31.1 Creating staves

Create staves like this:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'4 c''1")
```

```
abjad> show(staff)
```



31.2 Inspecting staff music

Return staff components with `music`:

```
abjad> staff.music  
(Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"), Note("b'4"), Note("c''1"))
```

31.3 Inspecting staff length

Get staff length with `len()`:

```
abjad> len(staff)  
8
```

31.4 Inspecting staff duration

Staff contents durations equals the sum of staff components' duration:

```
abjad> staff.contents_duration  
Duration(2, 1)
```

31.5 Adding one component to the end of a staff

Add one component to the end of a staff with `append`:

```
abjad> staff.append(Note("d' '2"))
```

```
abjad> show(staff)
```



31.6 Adding many components to the end of a staff

Add many components to the end of a staff with `extend`:

```
abjad> notes = [Note("e' '8"), Note("d' '8"), Note("c' '4")]
```

```
abjad> staff.extend(notes)
```

```
abjad> show(staff)
```



31.7 Finding the index of a staff component

Find staff component index with `index`:

```
abjad> notes[0]  
Note("e' '8")
```

```
abjad> staff.index(notes[0])  
9
```

31.8 Removing a staff component by index

Use `pop` to remove a staff component by index:

```
abjad> staff[8]  
Note("d' '2")
```

```
abjad> staff.pop(8)
```

```
abjad> show(staff)
```




```
abjad> staff.context
'CustomUserStaff'

abjad> f(staff)
\context CustomUserStaff = "Example Staff" {
    c' 8
    d' 8
    e' 8
    f' 8
    g' 8
    a' 8
    b' 4
    c'' 1
    e'' 8
    d'' 8
}
```

Force context when you have defined a new LilyPond context.

TUPLETS

32.1 Making a tuplet from a LilyPond input string

You can make an Abjad tuplet from a multiplier and a LilyPond input string:

```
abjad> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```
abjad> show(tuplet)
```



32.2 Making a tuplet from a list of other Abjad components

You can also make a tuplet from a multiplier and a list of other Abjad components:

```
abjad> leaves = [Note("fs'8"), Note("g'8"), Rest('r8')]
```

```
abjad> tuplet = Tuplet(Fraction(2, 3), leaves)
```

```
abjad> show(tuplet)
```



32.3 Understanding the interpreter display of a tuplet

The interpreter display of an Abjad tuplet contains three parts:

```
abjad> tuplet
Tuplet(2/3, [fs'8, g'8, r8])
```

`Tuplet` tells you the tuplet's class.

`2/3` tells you the tuplet's multiplier.

The list `[fs'8, g'8, r8]` shows the top-level components the tuplet contains.

32.4 Understanding the string representation of a tuplet

The string representation of a tuplet contains four parts:

```
abjad> print tuplet
{* 3:2 fs'8, g'8, r8 *}
```

Curly braces { and } indicate that the tuplet's music is interpreted sequentially instead of in parallel.

The asterisks * denote a fixed-multiplier tuplet.

3 : 2 tells you the tuplet's ratio.

The remaining arguments show the top-level components of tuplet.

32.5 Inspecting the LilyPond format of a tuplet

Get the LilyPond input format of any Abjad object with `format`:

```
abjad> tuplet.format
"\times 2/3 {\n\tfs'8\n\tg'8\n\tr8\n}"
```

Use `f()` as a short-cut to print the LilyPond format of any Abjad object:

```
abjad> f(tuplet)
\times 2/3 {
    fs'8
    g'8
    r8
}
```

32.6 Inspecting the music in a tuplet

Get the music in any Abjad container with `music`:

```
abjad> tuplet.music
(Note("fs'8"), Note("g'8"), Rest('r8'))
```

Abjad returns a read-only tuple of components.

32.7 Inspecting a tuplet's leaves

Get the leaves in any Abjad container with `leaves`:

```
abjad> tuplet.leaves
(Note("fs'8"), Note("g'8"), Rest('r8'))
```

Abjad returns a read-only tuple of leaves.

32.8 Getting the length of a tuplet

Get the length of any Abjad container with `len()`:

```
abjad> len(tuplet)
3
```

The length of every Abjad container is defined equal to the number of top-level components present in the container.

32.9 Getting the duration attributes of a tuplet

You set the multiplier of a tuplet at initialization:

```
abjad> tuplet.multiplier
Fraction(2, 3)
```

The contents durations of a tuplet equals the sum of written durations of the components in the tuplet:

```
abjad> tuplet.contents_duration
Duration(3, 8)
```

The multiplied duration of a tuplet equals the product of the tuplet's multiplier and the tuplet's contents duration:

```
abjad> tuplet.multiplied_duration
Duration(1, 4)
```

32.10 Understanding rhythmic augmentation and diminution

A tuplet with a multiplier less than 1 constitutes a type of rhythmic diminution:

```
abjad> tuplet.multiplier
Fraction(2, 3)

abjad> tuplet.is_diminution
True
```

A tuplet with a multiplier greater than 1 is a type of rhythmic augmentation:

```
abjad> tuplet.is_augmentation
False
```

32.11 Understanding binary and nonbinary tuplets

A tuplet is considered binary if the numerator of the tuplet multiplier is an integer power of 2:

```
abjad> tuplet.multiplier
Fraction(2, 3)

abjad> tuplet.is_binary
True
```

Other tuplets are nonbinary:

```
abjad> tuplet.is_nonbinary
False
```

32.12 Adding one component to the end of a tuplet

Add one component to the end of a tuplet with `append`:

```
abjad> tuplet.append(Note("e'4."))
```

```
abjad> show(tuplet)
```



32.13 Adding many components to the end of a tuplet

Add many components to the end of a tuplet with `extend`:

```
abjad> notes = [Note("fs'8"), Note("e'8"), Note("d'8"), Note("c'4.")]
abjad> tuplet.extend(notes)
```

```
abjad> show(tuplet)
```



32.14 Finding the index of a component in a tuplet

Find the index of a component in a tuplet with `index()`:

```
abjad> notes[1]
Note("e'8")
```

```
abjad> tuplet.index(notes[1])
5
```

32.15 Removing a tuplet component by index

Use `pop()` to remove a tuplet component by index:

```
abjad> tuplet[7]
Note("c'4.")
```

```
abjad> tuplet.pop(7)
```

```
abjad> show(tuplet)
```




32.16 Removing a tuplet component by reference

Remove tuplet components by reference with `remove()`:

```
abjad> tuplet.remove(tuplet[3])
```

```
abjad> show(tuplet)
```



32.17 Overriding attributes of the LilyPond tuplet number grob

Override attributes of the LilyPond tuplet number grob like this:

```
abjad> tuplet.override.tuplet_number.text = schemetools.SchemeFunction('tuplet-number::calc-fraction-
abjad> tuplet.override.tuplet_number.color = 'red'
```

```
abjad> f(tuplet)
\override TupletNumber #'color = #red
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
  fs'8
  g'8
  r8
  fs'8 [
  e'8
  d'8 ]
}
\revert TupletNumber #'color
\revert TupletNumber #'text
```

```
abjad> show(tuplet)
```



See the LilyPond docs for lists of grob attributes available.

32.18 Overriding attributes of the LilyPond tuplet bracket grob

Override attributes of the LilyPond tuplet bracket grob like this:

```
abjad> tuplet.override.tuplet_bracket.color = 'red'
```

```

abjad> f(tuplet)
\override TupletBracket #'color = #red
\override TupletNumber #'color = #red
\override TupletNumber #'text = #tuplet-number::calc-fraction-text
\times 2/3 {
    fs'8
    g'8
    r8
    fs'8 [
    e'8
    d'8 ]
}
\revert TupletBracket #'color
\revert TupletNumber #'color
\revert TupletNumber #'text

```

```
abjad> show(tuplet)
```



See the LilyPond docs for lists of grob attributes available.

VOICES

33.1 Making a voice from a LilyPond input string

You can make an Abjad voice from a LilyPond input string:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8 g'8 a'8 b'4 c''1")
```

```
abjad> show(voice)
```



33.2 Making a voice from a list of other Abjad components

You can also make a voice from a list of other Abjad components:

```
abjad> components = [Tuplet(Fraction(2, 3), "c'4 d'4 e'4"), Note("f'2"), Note("g'1")]
```

```
abjad> voice = Voice(components)
```

```
abjad> show(voice)
```



33.3 Understanding the repr of a voice

The repr of an Abjad voice contains three parts:

```
abjad> voice
Voice{3}
```

Voice tells you the voice's class.

3 tells you the voice's length (which is the number of top-level components the voice contains).

Curly braces { and } tell you that the music inside the voice is interpreted sequentially rather than in parallel.

33.4 Inspecting the LilyPond format of a voice

Get the LilyPond input format of any Abjad object with `format`:

```
abjad> voice.format
"\new Voice {\n\t\times 2/3 {\n\t\t\tc'4\n\t\t\td'4\n\t\t\te'4\n\t\t}\n\t\tf'2\n\t\tg'1\n}"
```

Use `f()` as a short-cut to print the LilyPond format of any Abjad object:

```
abjad> f(voice)
\nnew Voice {
    \times 2/3 {
        c'4
        d'4
        e'4
    }
    f'2
    g'1
}
```

33.5 Inspecting the music in a voice

Get voice components with `music`:

```
abjad> voice.music
(Tuplet(2/3, [c'4, d'4, e'4]), Note("f'2"), Note("g'1"))
```

Abjad returns a read-only tuple of components.

33.6 Inspecting a voice's leaves

Get the leaves in a voice with `leaves`:

```
abjad> voice.leaves
(Note("c'4"), Note("d'4"), Note("e'4"), Note("f'2"), Note("g'1"))
```

Abjad returns a read-only tuple of leaves.

33.7 Getting the length of a voice

Get voice length with `len()`:

```
abjad> len(voice)
3
```

The length of a voice is defined equal to the number of top-level components the voice contains.

33.8 Getting the duration attributes of a voice

The contents durations of a voice equals the sum of durations of the components in the voice:

```
abjad> voice.contents_duration
Duration(2, 1)
```

The preprolated duration of a voice is usually equal to the voice's contents duration:

```
abjad> voice.preprolated_duration
Duration(2, 1)
```

The prolated duration of a voice is usually equal to the voice's contents duration, too:

```
abjad> voice.preprolated_duration
Duration(2, 1)
```

Only when you nest a very small voice inside a tuplet will the prolated and preprolated duration of a voice differ.

Voices that are not nested inside a tuplet carry a prolation of 1:

```
abjad> voice.prolation
Fraction(1, 1)
```

All voice duration attributes are read-only.

33.9 Adding one component to the end of a voice

Add one component to the end of a voice with `append`:

```
abjad> voice.append(Note("af' 2"))

abjad> show(voice)
```



33.10 Adding many components to the end of a voice

Add many components to the end of a voice with `extend`:

```
abjad> notes = [Note("g' 4"), Note("f' 4")]
abjad> voice.extend(notes)

abjad> show(voice)
```



33.11 Finding the index of a component in a voice

Find the index of a component in a voice with `index()`:

```
abjad> notes[0]
Note("g' 4")
```

```
abjad> voice.index(notes[0])  
4
```

33.12 Removing a voice component by index

Use `pop()` to remove a voice component by index:

```
abjad> voice[5]  
Note("f'4")  
  
abjad> voice.pop(5)  
  
abjad> show(voice)
```



33.13 Removing a voice component by reference

Remove voice components by reference with `remove()`:

```
abjad> voice.remove(voice[-1])  
  
abjad> show(voice)
```



33.14 Naming voices

You can name Abjad voices:

```
abjad> voice.name = 'Upper Voice'
```

Voice names appear in LilyPond input:

```
abjad> f(voice)  
\context Voice = "Upper Voice" {  
  \times 2/3 {  
    c'4  
    d'4  
    e'4  
  }  
  f'2  
  g'1  
  af'2  
}
```

But not in notational output:

```
abjad> show(voice)
```



33.15 Changing the context of a voice

The context of a voice is set to 'Voice' by default:

```
abjad> voice.context
'Voice'
```

But you can change the context of a voice if you want:

```
abjad> voice.context = 'SpeciallyDefinedVoice'
```

```
abjad> voice.context
'SpeciallyDefinedVoice'
```

```
abjad> f(voice)
\context SpeciallyDefinedVoice = "Upper Voice" {
    \times 2/3 {
        c' 4
        d' 4
        e' 4
    }
    f' 2
    g' 1
    af' 2
}
```

Change the context of a voice when you have defined a new LilyPond context based on a LilyPond voice.

Developer documentation

CODEBASE

34.1 How the Abjad codebase is laid out

The Abjad codebase comprises twelve top-level directories.

```
abjad$ ls
__init__.py  cfg          core          docs          interfaces  scr          tools
book         checks       demos         exceptions    opt         templates
```

Of these, it is in the `tools` directory that the bulk of the musical reasoning implemented in Abjad resides.

```
abjad$ ls tools/
__init__.py      importtools      markuptools      quantizationtools  stafftools
configurationtools  instrumenttools  mathtools        resttools          tempotools
chordtools       intervaltreertools  measuretools     schemetools        threadtools
componenttools   iotools          timesignaturetools  scoretools         tietools
containertools  layouttools      musicxmltools     sequencetools      tonalitytools
contexttools     leaftools        notetools         sievetools          tuplettools
durationtools    lilyfiletools    pitcharraytools  skiptools           verticalitytools
gracetools       marktools        pitchtools       spannertools        voicetools
```

The remaining sections of this chapter cover the topics necessary to familiarize developers coming to the project for the first time.

34.2 Removing prebuilt versions of Abjad before you check out

If you'd like to be at the cutting edge of the Abjad development then you should check out from Google Code and tell Python and your operating system about Abjad. You can do this by following the steps below.

But before you do this you should realize that there are two ways to get Abjad up and running on your computer. The first way is by downloading a compressed version of Abjad from the [Python Package Index](#). You probably did this when you first discovered Abjad and started to use the system. The second way is by following the steps below to check out a copy of the most recent version of the Abjad repository hosted on Google Code. If you already have a version of Abjad running on your computer but you haven't yet followed the steps below to check out from Google Code, then you probably downloaded a compressed version of Abjad from the Python Package Index.

Before you check out from Google Code you should remove all prebuilt versions of Abjad from your machine. The reason you need to do this is that having both a prebuilt version of Abjad and a Subversion-managed version of Abjad on your machine can confuse your operating system and lead to weird results when you try to start Abjad.

You remove prebuilt versions of Abjad resident on your computer by finding your site packages directory and removing the so-called Abjad 'egg' that Python has installed there. After you remove the Abjad egg from your site packages

directory you will also need to remove the `abj`, `abjad` and `abjad-book` scripts from `/usr/local/bin` or from the directory that is equivalent to `/usr/local/bin` under your operating system.

First note the version of Python you're currently running.

```
$ python --version
Python 2.6.1
```

This is important because you may have more than one version of Python installed on your machine. (Which tends especially to be the case if you're running a Apple's OS X.)

Then note that the site packages directory is a part of your filesystem into which Python installs third-party Python packages like Abjad. The location of the site packages directory varies from one operating system to the next and you may have to Google to find the exact location of the site packages directory on your machine. Under OS X you can check `/Library/Python/2.x/site-packages/`. Under Linux the site packages directory is usually `/usr/lib/python2.x/site-packages`.

Once you've found your site packages directory you can list its contents to see if Python has installed an Abjad egg in it.

```
site-packages$ ls
Abjad-2.0-py2.6.egg      Sphinx-1.0.7-py2.6.egg  py-1.3.4-py2.6.egg
Jinja2-2.5-py2.6.egg    docutils-0.7-py2.6.egg  py-1.4.0-py2.6.egg
Pygments-1.3.1-py2.6.egg easy-install.pth        py-1.4.4-py2.6.egg
README                  guppy                   pytest-2.0.0-py2.6.egg
Sphinx-1.0.1-py2.6.egg  guppy-0.1.9-py2.6.egg-info pytest-2.1.0-py2.6.egg
Sphinx-1.0.4-py2.6.egg  py-1.3.1-py2.6.egg
```

Remove any Abjad eggs Python has installed in your site packages directory.

After you've done this you should check `/usr/local/bin` or equivalent to see if the `abj`, `abjad` or `abjad-book` scripts are installed there.

```
bin$ ls
abj      abjad    abjad-book
```

Remove any of the three scripts you find installed there so that you can use the new versions of the scripts you will download from Google Code instead.

```
bin$ sudo rm abj*
```

Now proceed to the steps below to check out from Google Code.

34.3 Installing the development version

Follow the steps listed above to remove prebuilt versions of Abjad from your machine. Then follow the steps below to check out from Google Code.

1. Make sure Subversion is installed on your machine.

```
svn --version
```

If Subversion responds then it is already installed. Otherwise visit the [Subversion](#) website.

2. Check out a copy of the main line of the Abjad codebase.

```
svn checkout http://abjad.googlecode.com/svn/abjad/trunk abjad-trunk
```

3. Add the abjad trunk directory to your `PYTHONPATH` environment variable.

```
export PYTHONPATH="/path/to/abjad-trunk:$PYTHONPATH
```

4. Alternatively you may symlink your Python site packages directory to the abjad trunk directory.

```
ln -s /path/to/abjad-trunk /path/to/site-package/abjad
```

5. Finally, add `abjad-trunk/scr/` to your `PATH` environment variable.

```
export PATH="/path/to/abjad-trunk/scr:$PATH
```

You will then be able to run Abjad with the ```abjad``` command.

You now have a copy of the main line of the most recent version of the Abjad repository checked out to your machine.

DOCS

The reST-based sources for the Abjad documentation are included in their entirety in every installation of Abjad. You may add to and edit these reST-based sources as soon as you install Abjad. However, to build human-readable HTML or PDF versions of the docs you will first need to download and install Sphinx.

The remaining sections of this chapter describe how the Abjad docs are laid out and how to build the docs with Sphinx.

35.1 How the Abjad docs are laid out

The source files for the Abjad docs are included in the `docs` directory of every Abjad install. The `docs` directory contains everything required to build HTML, PDF and other versions of the Abjad docs.

```
abjad$ ls docs/
Makefile  _templates  chapters    index.rst  scr
_static   _themes     conf.py     make.bat
```

The bulk of the Abjad docs live in `docs/chapters`. The chapter directories mirror the main sections on Abjad documentation. What you'll find as you inspect the chapter directories are a collection of `.rst` files organized into groups. The `.rst` extension identifies files written in restructured text.

One example:

```
abjad$ ls docs/chapters/appendices/glossary
index.rst
```

35.2 Installing Sphinx

Sphinx is the automated documentation system used by Python, Abjad and [other projects](#) implemented in Python. Because Sphinx is not included in the Python standard library you will probably need to download and install it.

First check to see if Sphinx is already installed on your machine.

```
$ sphinx-build --version
```

If Sphinx responds then the program is already installed on your machine. Otherwise visit the [Sphinx](#) website.

35.3 Removing old builds of the docs

After installing Sphinx, change to the Abjad `docs` directory and use the Sphinx makefile to remove any existing `docs/_build` directory prior to making a new build of the docs.

```
abjad$ cd docs

docs$ make clean
rm -rf _build/*
```

35.4 Generating the Abjad API

The `docs/scr` directory includes a script to generate the Abjad API. Run this script before building the Abjad docs for the first time.

```
docs$ scr/make-abjad-api
Building TOC tree ...
Now making Sphinx TOC ...

... Done.

Now building the HTML docs ...

sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... done

... (many lines omitted) ...

Build finished. The HTML pages are in _build/html.
```

Rerun `make-abjad-api` any time you add or remove a public class, method or function from the codebase.

35.5 Building the HTML docs

Change to the Abjad `docs` directory and run `make html`.

```
abjad$ cd docs

docs$ make html
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... not found
building [html]: targets for 568 source files that are out of date
updating environment: 568 added, 0 changed, 0 removed
reading sources... [ 13%] chapters/api/debug/debugghandlertoregators
reading sources... [ 37%] chapters/api/tools/clonewp/by_leaf_counts_with_parenta
reading sources... [ 38%] chapters/api/tools/clonewp/by_leaf_range_with_parentag
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_crosser
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_preprol
reading sources... [ 39%] chapters/api/tools/componenttools/get_le_duration_prol

... (many more lines omitted) ...
```

```
writing output... [ 85%] chapters/api/tools/spannertools/give_attached_to_childr
writing output... [ 95%] chapters/fundamentals/duration/interfaces_compared/inde
writing output... [100%] index /indexdexexexng/indexxxdexindex
writing additional files... genindex modindex search
copying images... done
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.
```

Build finished. The HTML pages are in `_build/html`.

You will then find the complete HTML version of the docs in `docs/_build/html`.

```
docs$ ls _build/
doctrees html
```

The output from Sphinx is verbose the first time you build the docs. On sequent builds, Sphinx reports changes only.

```
docs$ make html
sphinx-build -b html -d _build/doctrees . _build/html
Running Sphinx v1.0.7
loading pickled environment... done
building [html]: targets for 1 source files that are out of date
updating environment: 0 added, 1 changed, 0 removed
reading sources... [100%] chapters/devel/documentation/index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index ation/index
writing additional files... genindex modindex search
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.
```

Build finished. The HTML pages are in `_build/html`.

35.6 Building a PDF of the docs

Building a PDF of the docs is a two-step process. First you build a LaTeX version of the docs. Then you typeset the LaTeX docs as a PDF.

First change to the Abjad docs directory.

```
abjad$ docs
```

Then make LaTeX sources of the docs.

```
docs$ make latex
sphinx-build -b latex -d _build/doctrees . _build/latex
Running Sphinx v1.0.7
loading pickled environment... done
building [latex]: all documents
updating environment: 0 added, 0 changed, 0 removed
looking for now-outdated files... none found
```

```
processing Abjad.tex... index chapters/start_here/abjad/index chapters/examples/bartok...
(... many lines omitted ...)
...ndices/pitch_conventions/images/example-3.png chapters/examples/ligeti/images/desordre.jpg
copying TeX support files... done
build succeeded.
```

Build finished; the LaTeX files are in `_build/latex`.
Run `'make all-pdf'` or `'make all-ps'` in that directory to run these through `(pdf)latex`.

Now follow the instructions provided by Sphinx and change to the LaTeX build directory.

```
docs$ cd _build/latex/
```

Then make a PDF version of the docs from the LaTeX sources.

```
latex$ make all-pdf

pdflatex 'Abjad.tex'
This is pdfTeXk, Version 3.141592-1.40.3 (Web2C 7.5.6)
  %&-line parsing enabled.
entering extended mode
(./Abjad.tex
LaTeX2e <2005/12/01>
Babel <v3.8h> and hyphenation patterns for english, usenglishmax, dumylang, noh
ypheation, arabic, basque, bulgarian, coptic, welsh, czech, slovak, german, ng
erman, danish, esperanto, spanish, catalan, galician, estonian, farsi, finnish,

(... many lines omitted ...)
```

The resulting docs will appear as `Abjad.pdf` in the LaTeX build directory you're currently in.

35.7 Building a coverage report

Change to the Abjad docs directory and call `sphinx-build` explicitly with the coverage builder, source directory and target directory.

```
docs$ sphinx-build -b coverage . _build/coverage
Making output directory...
Running Sphinx v1.0.7
loading pickled environment... not found
building [coverage]: coverage overview
updating environment: 568 added, 0 changed, 0 removed
reading sources... [ 37%] chapters/api/tools/clonewp/by_leaf_counts_with_parenta
reading sources... [ 38%] chapters/api/tools/clonewp/by_leaf_range_with_parentag
reading sources... [ 38%] chapters/api/tools/componenttools/get_duration_crosser

... (many lines omitted) ...

reading sources... [ 85%] chapters/api/tools/spannertools/withdraw_from_containe
reading sources... [ 95%] chapters/fundamentals/duration/interfaces_compared/ind
reading sources... [100%] index                                t/indexdexexexng/indexxxdexindex
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
build succeeded.
```


The coverage report is now available in the `docs/_build/coverage` directory.

```
docs$ ls _build/
coverage doctrees html
```

35.8 Building other versions of the docs

Examine the Sphinx makefile in the Abjad `docs/` directory or change to the `docs/` directory and type `make` with no arguments to see a list of the other versions of the Abjad docs that are available to build.

```
docs$ make
Please use 'make <target>' where <target> is one of
  html          to make standalone HTML files
  dirhtml       to make HTML files named index.html in directories
  pickle        to make pickle files
  json          to make JSON files
  htmlhelp      to make HTML files and a HTML help project
  qthelp        to make HTML files and a qthelp project
  latex         to make LaTeX files, you can set PAPER=a4 or PAPER=letter
  changes       to make an overview of all changed/added/deprecated items
  linkcheck     to check all external links for integrity
  doctest       to run all doctests embedded in the documentation (if enabled)
```

35.9 Inserting images with `abjad-book`

Use *abjad-book* to insert snippets of notation in the docs you write in reST.

Embed Abjad code between open and close `<abjad>` `</abjad>` tags in your `.rst.raw` sourcefile and then call `abjad-book` to create a pure `.rst` file.

```
abjad-book foo.rst.raw foo.rst
```

```
Parsing file ...
Rendering "example-1.ly" ...
Rendering "example-2.ly" ...
```

You will need to build the HTML docs again to see your work.

```
make html
```

35.10 Updating Sphinx

It is important periodically to update your version of Sphinx. If you used `easy_install` to install Sphinx then the usual command to update Sphinx is this:

```
$ sudo easy_install -U Sphinx
```

This will usually work. But if Sphinx fails to update then it may be because you have multiple versions of Python installed on your computer. (This tends especially to be the case under Apple's OS X.)

To get around this first note the version of Python you're currently running:

```
$ python --version  
Python 2.6.1
```

Then use a version-explicit form of `easy_install` to update Sphinx:

```
$ sudo easy_install-2.6 -U Sphinx
```

TESTS

Abjad includes an extensive battery of tests. Abjad is in a state of rapid development and extension. Major refactoring efforts are common every six to eight months and are likely to remain so for several years. And yet Abjad continues to allow the creation of complex pieces of fully notated score in the midst of these changes. We believe this is due to the extensive coverage provided by the automated regression battery described in the following sections.

36.1 Automated regression?

A battery is any collection of tests. Regression tests differ from other types of test in that they are designed to be run again and again during many different stages of the development process. Regression tests help ensure that the system continues to function correctly as developers make changes to it. An automated regression battery is one that can be run automatically by some sort of driver with minimal manual intervention.

Several different test drivers are now in use in the Python community. Abjad uses `py.test`. The `py.test` distribution is not included in the Python standard library, so one of the first thing new contributors to Abjad should do is download and install `py.test`, and then run the existing battery.

36.2 Running the battery

Change to the directory where you have Abjad installed. Then run `py.test`.

```
abjad$ py.test
===== test session starts =====
platform darwin -- Python 2.6.1 -- pytest-2.1.0
collected 4235 items

core/LilyPondContextProxy/test/test_LilypondContextProxy__eq__.py .
core/LilyPondContextProxy/test/test_LilypondContextProxy__repr__.py .
core/LilyPondContextProxy/test/test_LilypondContextProxy__setattr__.py ..

... (many lines omitted) ...

tools/voicetools/test/test_voicetools_iterate_semantic_voices_forward_in_expr.py .
tools/voicetools/test/test_voicetools_iterate_voices_backward_in_expr.py .
tools/voicetools/test/test_voicetools_iterate_voices_forward_in_expr.py .

===== 4235 passed in 127.06 seconds =====
```

Abjad r4629 includes 4235 tests.

36.3 Reading test output

`py.test` crawls the entire directory structure from which you call it, running tests in alphabetical order. `py.test` prints the total number of tests per file in square brackets and prints test results as a single `.` dot for success or else an `F` for failure.

36.4 Writing tests

Project check-in standards ask that tests accompany all code committed to the Abjad repository. If you add a new function, class or method to Abjad, you should add a new test file for that function, class or method. If you fix or extend an existing function, class or method, you should find the existing test file that covers that code and then either add a completely new test to the test file or else update an existing test already present in the test file.

36.5 Test files start with `test_`

When `py.test` first starts up it crawls the entire directory structure from which you call it prior to running a single test. As `py.test` executes this preflight work, it looks for any files beginning or ending with the string `test` and then collects and alphabetizes these. Only after making such a catalog of tests does `py.test` begin execution. This collect-and-cache behavior leads to the important point about naming, below.

36.6 Avoiding name conflicts

Note that the names of **test functions** must be absolutely unique across the entire directory structure on which you call `py.test`. You must never share names between test functions. For example, you must not have two tests named `test_grob_handling_01()` **even if both tests live in different test files**. That is, a test named `test_grob_handling_01()` living in the file `test_accidental_grob_handling.py` and a second test named `test_grob_handling_01()` living in the file `test_notehead_grob_handling.py` will conflict with the each other when `py.test` runs. And, unfortunately, **“py.test is silent about such conflicts when it runs**. That is, should you run `py.test` with the duplicate naming situation described here, what will happen is that `py.test` will correctly run and report results for the **first** such test it finds. However, when `py.test` encounters the second like-named test, `py.test` will incorrectly report cached results for the **first** test rather than the second. The take-away is to include some sort of namespacing indicators in every test name and not to be afraid of long test names. The `test_grob_handling_01()` example given here fixes easily when the two tests rename to `test_accidental_grob_handling_01()` and `test_notehead_grob_handling_01()`.

36.7 Updating `py.test`

It is important periodically to update `py.test`.

The usual command to do this is:

```
$ sudo easy_install -U pytest
```

Note that `pytest` is here spelled without the intervening period.

36.8 Running doctest on the tools directory

The Python standard library includes the `doctest` module as way of checking the correctness of examples included in Python docstrings. The module searches for instances of the Python interpreter prompt `'>>>'` and executes any code that follows. Abjad docs display the Abjad prompt `'abjad>'` instead of the Python prompt. This means that all instances of the Abjad prompt must be changed to Python prompts before running `doctest` on the Abjad codebase. Three scripts in `abjad/scr/devel` help do this.

First change to the subdirectory of the Abjad source tree on which you'd like to run `doctest`. Then run these scripts:

```
replace-abjad-prompts-with-python-prompts
```

```
run-doctest-on-all-modules-in-tree
```

```
replace-python-prompts-with-abjad-prompts
```

After running `run-doctest-on-all-modules-in-tree` you can inspect the results that come back from `doctest` and make any fixes as required.

SCRIPTS

The `abjad/scr/devel` directory contains scripts for Abjad developers. Add `abjad/scr/devel` to your `PATH` to use the scripts described below.

```
abjad$ ls scr/devel
abj-grep
abj-grp
abj-rmpycs
abj-src-grp
abj-test-grp
abj-update
capitalize-test-file-names
conjoin-multiline-import-statements
count-source-lines
count-tools
duplicate-test-file
find-and-fix-manual-class-package-initializers
find-duplicate-module-names
find-duplicate-tool-module-names
find-import-as-statements
find-local-import-statements
find-lower-camel-case-definitions
find-lower-camel-case-modules
find-manual-class-loads-in-initializers
find-misnamed-private-modules
find-missing-test-modules
find-module-headers
find-modules-with-chevrons
find-multifunction-modules
find-multiline-import-statements
find-nonalphabetized-module-headers
find-nontrivial-subdirectories
find-public-helpers-without-docstrings
find-undocumented-tools
fix-nonalphabetized-module-headers
fix-test-case-block-comments
fix-test-case-names
fix-test-case-numbers
format-lilypond-context-names-with-underscores
list-private-modules
rebuild-docs
reindent-3-spaces-as-4
reindent-4-spaces-as-3
reindent-spaces-variably
remove-tmp-out-directories
rename-public-helper
replace-abjad-prompts-with-python-prompts
replace-in-files
replace-python-prompts-with-abjad-prompts
run-doctest-on-all-modules-in-tree
```

37.1 Searching the Abjad codebase with `abj-grep`

Abjad provides a wrapper around UNIX `grep` in the form of `abj-grep`. Use this script to recursively search the entire Abjad codebase, leaving out non-human-readable files, files located in special `.svn` Subversion subdirectories, and all files in the `abjad/documentation` directories. You can run `abj-grep` from any directory on your system; you needn't be in the Abjad source directories when you call `abj-grep`.

```
$ abj-grep 'is_assignable('
leaf/duration.py:111:         if not durationtools.is_assignable(rational):
tempo/indication.py:67:         assert durationtools.is_assignable(arg)
tools/check/are_scalable.py:12:         if not durationtools.is_assignable(candidate_duration):
tools/durationtools/is_assignable.py:5: def is_assignable(duration):
tools/durationtools/prolated_to_written.py:2: from abjad.tools.durationtools.is_assignable import is_a
```

```
tools/durationtools/prolated_to_written.py:15:    if is_assignable(prolated_duration):
tools/tietools/duration_change.py:28:    if durationtools.is_assignable(new_written_duration):
tools/tupletttools/contents_scale.py:30:    if durationtools.is_assignable(multiplier):
```

37.2 Removing old *.pyc files with `abj-rmpycs`

See the section on `abj-update` below for the reasons that it is a good idea to periodically remove the byte-compiled *.pyc files that Python generates for its own use behind the scenes. Abjad supplies `abj-rmpycs` to delete all the *.pyc in the Abjad codebase, leaving other *.pyc on your system untouched.

37.3 Updating your development copy of Abjad with `abj-update`

The normal way of updating your working copy of a Subversion repository is with the `svn update` or `svn up` command. You can update your working copy of Abjad in the usual way with `svn up`. But Abjad supplies an `abj-update` script as a wrapper around the usual Subversion update commands. In addition to updating your working copy of Abjad, `abj-update` populates the `abjad/.version` file with the most recent revision number of the system, and then removes all *.pyc files from your Abjad install. The benefits here are twofold. First, Abjad adds the most recent revision number of the system to all .ly files that you generate when working with Abjad. If you do not update the Abjad version file on a regular basis, the headers in your Abjad-generated .ly files will list the wrong version of the system. Second, as is the case in working with any substantial Python codebase, it is a good idea to periodically remove the byte-compiled *.pyc files that Python creates for its own use. The reason for this is inadvertant name aliasing. That is, if there was previously a module named `foo.py` somewhere in the system and if Python had at some point imported the module and created `foo.pyc` as a byproduct, this .pyc file will remain on the filesystem even if you later decide to remove, or rename, the source `foo.py` module. This lead to confusion because days or weeks after `foo.py` has been removed, Python will still find `foo.pyc` and seem to make the contents of `foo.py` available from beyond the grave. Updating with `abj-update` takes care of these two situations.

37.4 Counting lines of code with `count-source-lines`

Run `count-source-lines` for a count of lines of count divided between source and test files.

```
abjad$ count-source-lines

source_modules: 1703
test_modules:   1812

source_lines:   73942
test_lines:     76636

total lines:    150578
test-to-source ratio is 1 : 1
```

The script is directory-dependent so you can run it any the entire Abjad codebase or any subdirectory of the codebase.

37.5 Global search-and-replace with `replace-in-files`

You probably won't need to use `replace-in-files` very often. But if you are making changes to Abjad that will cause some name, such as `FooBar`, to be globally changed everywhere in the Abjad codebase to, say to `foo_bar`,

then you can use `replace-in-files` to save lots of time.

```
$ replace-in-files --help
```

Usage:

```
replace-in-files DIR OLD_TEXT NEW_TEXT [CONFIRM=true/false]
```

Crawl directory DIR and read every file in it recursively.
Replace OLD_TEXT with NEW_TEXT in each file.

Set CONFIRM to 'false' to replace without prompting.

37.6 Adding new development scripts

If you write and then find yourself using a certain script over and over again when you're developing new code for Abjad, consider contributing back to the project so we can include your script in the next public release of Abjad. Scripts in the the Abjad script directories end with no file extension and try to be as OS-portable as possible, which usually means writing the script in Python, rather than your operating system's shell, and relying heavily on Python's `os` module.

USING ABJAD-BOOK

`abjad-book` is an independent application included in every installation of Abjad. `abjad-book` allows you to write Abjad code in the middle of documents written in HTML, LaTeX or ReST. We created `abjad-book` to help us document Abjad. Our work on `abjad-book` was inspired by `lilypond-book`, which does for LilyPond much what `abjad-book` does for Abjad.

38.1 HTML with embedded Abjad

To see `abjad-book` in action, open a file and write some HTML by hand. Add some Abjad code to your HTML between open and close `<abjad>` `</abjad>` tags.

```
<html>

<p>This is an <b>HTML</b> document.</p>

<p>The code is standard hypertext mark-up.</p>

<p>Here is some music notation generated automatically by Abjad:</p>

<abjad>
v = Voice(construct.scale(8))
Beam(v)
write_ly(v, 'abjad-book-1') <hide
show(v)
</abjad>

<p>And here is more ordinary <b>HTML</b>.</p>

</html>
```

Save your the file with the name `example.html.raw`. You now have an HTML file with embedded Abjad code.

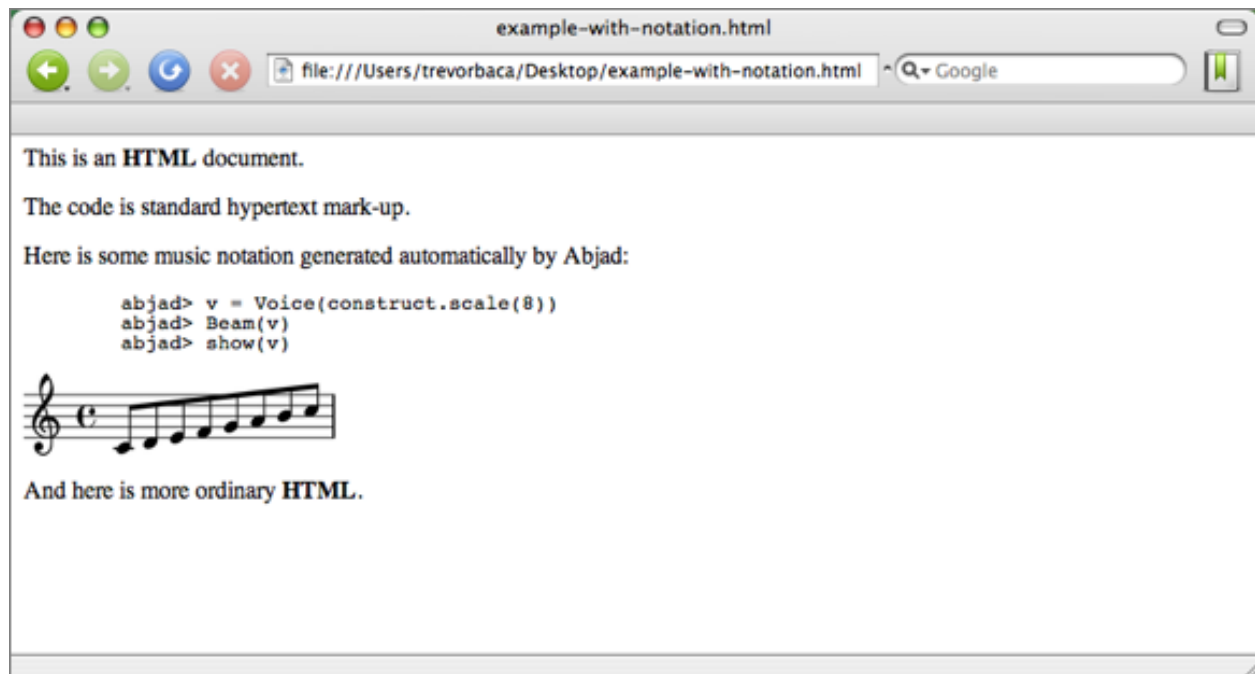
In the terminal, call `abjad-book` on `example.html.raw`.

```
$ abjad-book example.html.raw example.html
```

```
Parsing file...
Rendering "abjad-book-1.ly"...
```

The application opens `example.html.raw`, finds all Abjad code between `<abjad>` `</abjad>` tags, executes it, and then creates and inserts image files of music notation accordingly.

Open `example.html` with your browser.



That's all there is to it. `abjad-book` lets you open a file and type HTML by hand with Abjad sandwiched between the special `<abjad>` `</abjad>` tags described here. Run `abjad-book` on such a hybrid file to create pure HTML with images of music notation created by Abjad.

Note: `abjad-book` makes use of ImageMagick's `convert` application to crop and scale PNG images generated for HTML and ReST documents. For LaTeX documents, `abjad-book` uses `pdfcrop` for cropping PDFs.

38.2 LaTeX with embedded Abjad

You can use `abjad-book` to insert Abjad code and score excerpts into any LaTeX you create. Type the sample code below into a file.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{listings}
\begin{document}
```

This is a standard LaTeX document with embedded Abjad.

The code below creates an Abjad measure and then prints the measure format string.

```
<abjad>
measure = RigidMeasure((5, 8), construct.scale(5))
print measure.format
</abjad>
```

This next bit of code knows about the measure we defined earlier. This code renders the measure as a PDF using a template suitable for inclusion in LaTeX documents.

```
<abjad>
write_ly(measure, 'abjad-book-1', 'oedo') <hide
</abjad>
```

And this is the end of the our sample LaTeX document.

```
\end{document}
```

Save your file with the name `example.tex.raw`. You now have a LaTeX file with embedded Abjad code.

In the terminal, call `abjad-book` on `example.tex.raw`.

```
$ abjad-book example.tex.raw example.tex
```

```
Processing 'example.tex.raw'. Will write output to 'example.tex'...
Parsing file...
Rendering "abjad-book-1.ly"...
```

The application open `example.tex.raw`, finds all code between Abjad tags, executes it, and then creates and inserts Abjad interpreter output and PDF files of music notation. You can view the contents of the next LaTeX file `abjad-book` has created.

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{listings}
\begin{document}
```

This is a standard LaTeX document with embedded Abjad.

The code below creates an Abjad measure and then prints the measure format string.

```
\begin{lstlisting}[basicstyle=\footnotesize, tabsize=4, showtabs=false, showspace=false]
  abjad> measure = RigidMeasure((5, 8), construct.scale(5))
  abjad> print measure.format
  {
    \time 5/8
    c'8
    d'8
    e'8
    f'8
    g'8
  }
\end{lstlisting}
```

This next bit of code knows about the measure we defined earlier. This code renders the measure as a PDF using a template suitable for inclusion in LaTeX documents.

```
\includegraphics{images/abjad-book-1.pdf}
```

And this is the end of the our sample LaTeX document.

```
\end{document}
```

You can now process the file `example.tex` just like any other LaTeX file, using `pdflatex` or `TexShop` or whatever LaTeX compilation program you normally use on your computer.

```
$ pdflatex example.tex
```

```
This is pdfTeX, Version 3.141592-1.40.3 (Web2C 7.5.6)
 %&-line parsing enabled.
entering extended mode
...
```

And then open the resulting PDF.

38.3 Using abjad-book on ReST documents

You can call `abjad-book` on ReST documents, too. Follow the examples given here for HTML and LaTeX documents and modify accordingly.

38.4 Using `[hide = True]`

You can add `[hide = True]` to any `abjad-book` example to show only music notation.

```
<abjad>[hide = True]
staff = Staff(construct.scale(8))
write_ly(staff, 'staff-example', 'oedo')
</abjad>
```

TIMING CODE

You can time code with Python's built-in `timeit` module:

```
from abjad import *
import timeit

timer = timeit.Timer('Note(0, (1, 4))', 'from __main__ import Note')
print timer.timeit(1000)

0.225436925888
```

These results show that 1000 notes take 0.23 seconds to create.

Other Python timing modules are available for download on the public Internet.

PROFILING CODE

Profile code with `profile_expr()` in the `iotools` package:

```
abjad> iotools.profile_expr('Note(0, (1, 4))')
Sun Aug 14 16:50:36 2011      _tmp_abj_profile

      327 function calls (312 primitive calls) in 0.001 CPU seconds

Ordered by: cumulative time
List reduced from 96 to 12 due to restriction <12>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
      1   0.000    0.000    0.001    0.001 <string>:1(<module>)
      1   0.000    0.000    0.001    0.001 Note.py:18(__init__)
      1   0.000    0.000    0.001    0.001 Note.py:133(fset)
      1   0.000    0.000    0.001    0.001 NoteHead.py:18(__init__)
      1   0.000    0.000    0.001    0.001 NoteHead.py:121(fset)
      1   0.000    0.000    0.001    0.001 NamedChromaticPitch.py:28(__new__)
      1   0.000    0.000    0.000    0.000 Leaf.py:18(__init__)
      1   0.000    0.000    0.000    0.000 chromatic_pitch_name_to_diatonic_pitch_numbe
      1   0.000    0.000    0.000    0.000 octave_tick_string_to_octave_number.py:4(oct
      1   0.000    0.000    0.000    0.000 re.py:134(match)
      1   0.000    0.000    0.000    0.000 re.py:227(_compile)
      1   0.000    0.000    0.000    0.000 sre_compile.py:501(compile)
```

These results show 327 function calls to create a note.

The `profile_expr()` function wraps the Python `cProfile` and `pstats` modules.

MEMORY CONSUMPTION

You can examine memory consumption with tools included in the `guppy` module:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
notes = [Note(0, (1, 4)) for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 11024 objects. Total size = 586364 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	9	124000	21	124000 21 abjad.tools.notetools.Note.Note.Note
1	1004	9	116464	20	240464 41 __builtin__.set
2	2003	18	76300	13	316764 54 list
3	1000	9	52000	9	368764 63 abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch
4	1000	9	44000	8	412764 70 abjad.interfaces._OffsetInterface._OffsetInterface._OffsetInterface
5	1000	9	44000	8	456764 78 abjad.tools.notetools.NoteHead.NoteHead.NoteHead
6	1000	9	40000	7	496764 85 0x23add0
7	1000	9	32000	5	528764 90 abjad.interfaces.ParentageInterface.ParentageInterface.ParentageInterface
8	1011	9	28568	5	557332 95 str
9	1000	9	28000	5	585332 100 abjad.interfaces._NavigationInterface._NavigationInterface._NavigationInterface

<6 more rows. Type e.g. `'_.more'` to view.>

These results show 586K for 1000 notes.

You must download `guppy` from the public Internet because the module is not included in the Python standard library.

CLASS ATTRIBUTES

Consider the definition of this class:

```
class FooWithInstanceAttribute(object):

    def __init__(self):
        self.constants = (
            'red', 'orange', 'yellow', 'green',
            'blue', 'indigo', 'violet',
        )
```

1000 objects consume 176k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [FooWithInstanceAttribute() for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 2004 objects. Total size = 176536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	50	140000	79	140000 79 dict of __main__.FooWithInstanceAttribute
1	1000	50	32000	18	172000 97 __main__.FooWithInstanceAttribute
2	1	0	4132	2	176132 100 list
3	1	0	348	0	176480 100 types.FrameType
4	1	0	44	0	176524 100 __builtin__.weakref
5	1	0	12	0	176536 100 int

But consider the definition of this class:

```
class FooWithSharedClassAttribute(object):

    def __init__(self):
        pass

    self.constants = (
        'red', 'orange', 'yellow', 'green',
        'blue', 'indigo', 'violet',
    )
```

1000 objects consume only 36k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
```

```
objects = [FooWithClassAttribute() for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 1004 objects. Total size = 36536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	100	32000	88	32000 88 __main__.FooWithClassAttribute
1	1	0	4132	11	36132 99 list
2	1	0	348	1	36480 100 types.FrameType
3	1	0	44	0	36524 100 __builtin__.weakref
4	1	0	12	0	36536 100 int

Objects that share class attributes between them can consume less memory than objects that don't. But consider the usual provisions between class attributes and instance attributes when implementing custom classes. Class attributes make sense when objects will never modify the attribute in question. Class attributes also make sense when objects will modify the attribute in question and will desire to change the attribute in question for all other like objects at the same time. Probably best to use instance attributes in most other cases.

USING SLOTS

Consider the definition of this class:

```
class Foo(object)

    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
```

1000 objects consume 176k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [Foo(1, 2, 3) for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 2004 objects. Total size = 176536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	50	140000	79	140000 79 dict of __main__.FooWithInstanceAttribute
1	1000	50	32000	18	172000 97 __main__.FooWithInstanceAttribute
2	1	0	4132	2	176132 100 list
3	1	0	348	0	176480 100 types.FrameType
4	1	0	44	0	176524 100 __builtin__.weakref
5	1	0	12	0	176536 100 int

But consider the definition of this class:

```
class FooWithSlots(object):

    __slots__ = ('a', 'b', 'c')
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
```

1000 objects consume only 40k:

```
from guppy import hpy
hp = hpy()
hp.setrelheap()
objects = [FooWithSlots(1, 2, 3) for x in range(1000)]
h = hp.heap()
print h
```

Partition of a set of 1004 objects. Total size = 40536 bytes.

Index	Count	%	Size	% Cumulative	% Kind (class / dict of class)
0	1000	100	36000	89	36000 89 <code>__main__.Bar</code>
1	1	0	4132	10	40132 99 <code>list</code>
2	1	0	348	1	40480 100 <code>types.FrameType</code>
3	1	0	44	0	40524 100 <code>__builtin__.weakref</code>
4	1	0	12	0	40536 100 <code>int</code>

The example here confirms the Python Reference Manual 3.4.2.4: “By default, instances of both old and new-style classes have a dictionary for attribute storage. This wastes space for objects having very few instance variables. The space consumption can become acute when creating large numbers of instances.”

CODING STANDARDS

Indent with spaces, not with tabs. Use four spaces at a time:

```
def foo(x, y):  
    return x + y
```

Introduce comments with one pound sign and a single space:

```
# comment before foo  
def foo(x, y):  
    return x + y
```

Favor early imports at the head of each module. Only one import per line:

```
from foo import x  
from foo import y  
from foo import z
```

Include two blank lines after import statements before the rest of the module:

```
from foo import x  
from foo import y  
from foo import z
```

```
class Foo(object):  
    ...  
    ...
```

Wrap docstrings with triple apostrophes and align like this:

```
def foo(x, y):  
    '''This is the first line of the foo docstring.  
       This is the second line of the foo docstring.  
       And this is the last line of the foo docstring.  
    '''
```

Use paired apostrophes to delimit strings:

```
s = 'foo'
```

Use paired quotation marks to delimit strings within a string:

```
s = 'foo and "bar"'
```

Name classes in upper camelcase:

```
def FooBar(object):  
    ...  
    ...
```

Name bound methods in underscore-delimited lowercase:

```
def Foo(object):  
  
    def bar_blah(self):  
        ...  
  
    def bar_baz(self):  
        ...
```

Name module-level functions in underscore-delimited lowercase:

```
def foo_bar():  
    ...  
  
def foo_blah():  
    ...
```

Separate bound method definitions with a single empty line:

```
class FooBar(object):  
  
    def __init__(self, x, y):  
        ...  
  
    def bar_blah(self):  
        ...  
  
    def bar_baz(self):  
        ...
```

Organize the definitions of core classes into the five following major sections plus initialization:

```
class FooBar(object):  
  
    def __init__(self, x, y):  
        ...  
  
    ### SPECIAL METHODS ###  
  
    def __repr__(self):  
        ...  
  
    def __str__(self):  
        ...  
  
    ### PRIVATE PROPERTIES ###  
  
    @property  
    def _foo(self):  
        ...  
  
    ### PUBLIC PROPERTIES ###  
  
    @property  
    def bar(self):
```

```
...

### PRIVATE METHODS ###

def _blah(self, x, y):
    ...

### PUBLIC METHODS ###

def baz(self, z):
    ...
```

Precede private class attributes with a single underscore:

```
class FooBar(object):

    ### PRIVATE PROPERTIES ###

    @property
    def _foo(self):
        ...

    ### PRIVATE METHODS ###

    def _blah(self, x, y):
        ...
```

Alphabetize method names.

Use < less-than signs in preference to greater-than signs:

```
if x < y < z:
    ...
```

Limit lines to 110 characters and use \ to break lines where necessary.

Eliminate trivial slice indices. Use `s[:4]` instead of `s[0:4]`.

Do not abbreviate variable names.

Name variables that represent a list or other collection of objects in the plural.

Implement only one class per module.

Implement only one function per module.

Author one `py.test` test file for every module-level function.

Author one `py.test` test file for every bound method in the public interface of a class.

Appendices

FROM TREVOR AND VÍCTOR

We are composers Trevor Bača and Víctor Adán, creators of Abjad, and our earliest collaborative work dates back to shared undergraduate years in Austin. It was the mid- to late-90s and we found ourselves interested in ways of building up ever larger sets of musical materials in our scores, with ever greater amounts of musical information.

Our work then began with pitch formalization, creating materials in C and then writing the results as MIDI to hear what we'd created. Turns out that this is a fairly common gateway into materials generation for many composers, and so it was for us. Probably this was, and is, due to the ever present availability of MIDI and, to a lesser extent, CSound. But even back then it was clear to us to finding ways to embody other aspects of the musical score – from nested rhythms to the different approaches to the musical measure to the arbitrarily complex structures possible with overlapping musical voices – would require a wholly different level of consideration, and different development techniques as well.

As an example, consider flat lists of floating-point values. This basic data structure, together with the constant need some type of quantification or rounding, feeds much of most composers' work with CSound, pd and the like. It is a good thing, therefore, that essentially all modern programming languages include tools for manipulating flat lists of floats out of the box, or in the standard library. But what happens when you want to think of pitch as something much more than integers for core values with, perhaps, floats for microtones? What if you want to work with pitches as fully-fledged objects? Objects capable of carrying arbitrarily large sets of attributes and values? Objects that might group together, first into sets, and then into larger assemblages, and then into still larger complexes of pitch information loaded, or even overloaded, with cross-relationships or textural implications? Carrying this surplus of information about pitch, or the potential uses of pitch, in data structures limited to, or centered around, the list-of-floats paradigm then becomes a burden.

And what of working with rhythms not only as offset values, as implied by the list-of-floats approach, but as arbitrarily nested, stretched, compressed and stacked sets of values, as allowed by the tupleting and measure structures of conventional score? A different approach is needed.

There was, and still is, no reason to believe that general purpose programming languages and development tools should come readily supplied with the objects and methods most suitable for composerly applications. And this means that the attributes of a domain-specific language that will best meet the needs of composers interested in working formally with the full complement of capabilities in traditional score remains an open question.

We continued our work in score formalization independently until 2005, Trevor in a system that would come to be called Lascaux, and Víctor in a system dubbed Cuepatlahto. We experimented with C, Mathematica and Matlab as the core programming languages driving our systems before settling independently on Python, Víctor out of experience at MIT, where he was working on his masters at the Media Lab with Berry Vercoe, and Trevor out of the working necessities of a professional developer and engineer.

We passed through independent experiences using Finale, Sibelius, Leland Smith's SCORE, and even Adobe Illustrator as the notational rendering engines for Lascaux and Cuepatlahto. Through all of this, both systems were designed to tackle a shared set of problems. These included:

1. The difficulty involved in transcribing larger scale and highly parameterized gestures and textures into traditional Western notation.

2. The general inflexibility of closed, commercial music notation software packages.
3. The relative inability of objects on the printed page in conventional score to point to each other — or, indeed, to other objects or ideas outside the printed page — in ways rich enough to help capture, model and develop long-range, nonlocal relationships throughout our scores.

Afer collaborating on a joint paper describing the two systems, and after discussing collaborative design and implementation at length, both online and in weekends' long review of our respective codebases, we decided to combine our efforts into a single, unified project. That project is now Abjad.

In our work on Abjad we strive to develop a powerful and flexible symbolic system. We picked the phrase 'formalized score control', or FSC, as a nod to Xenakis, who was so far ahead in so many ways, and also to highlight our primary project goal: to bring the full power of modern programming languages, and tools in mathematics, text processing, pattern recognition, and modular, iterative and incremental development to bear on all parts of the compositional process.

WHY MIDI IS NOT ENOUGH

Given that Abjad models written musical score, it might seem odd for MIDI to be even mentioned in this manual. Yet, until fairly recently, MIDI has played a role (sometimes tangential, other times fundamental) in a variety of software tools related to music notation and engraving.

46.1 A very brief overview of MIDI

MIDI (Musical Instrument Digital Interface) was first introduced in 1981 by Dave Smith, the founder of Sequential Circuits. The original purpose of MIDI was to allow the communication between different electronic musical instruments; more specifically, to allow one device to send **control** data to another device. Typical messages might be “note On” (play a *note*) “note Off” (turn off a *note*). A MIDI “note” message, for example, is composed of three bytes: the first byte (the Status byte) tells the device what kind of message this is (e.g. a Note On message). The second byte encodes key number (which key was pressed) and the third byte, velocity (how hard the key was pressed). It should be clear that a *Note* in this context means something very different than *Note* in the context of a traditional printed score. While the bias towards keyboard interfaces is clear in the definition of the MIDI Note control message, one can still give the MIDI note a more general use by reinterpreting “key number” as pitch and “velocity” as loudness, the usual perceptual correlates of these control changes as well as the most meaningful musical parameters in western music.

With the subsequent proliferation of music production software, the SMF (Standard Midi File) was introduced to allow the recording and storage of the control data from a MIDI stream. The SMF required a time stamp to keep track of when control messages took place. These are called “delta-times” in the SMF specification.

“The MTrk chunk type is where actual song data is stored. It is simply a stream of MIDI events (and non-MIDI events), preceded by delta-time values.”

In combination with the MIDI Note message, the addition of duration now allowed one to have a minimal but sufficient **machine** representation—a machine score—of music requiring only these parameters: duration, pitch and loudness. Such is the case of most piano music.

46.2 Limitations of MIDI from the point of view of score modeling

But, alas, there is much more information in a printed score that can not be practically encoded in a SMF. Common musical notions such as meter, clef, key signature, articulation, to name only a few, are ignored. A desire to include some of these concepts in MIDI is evident in the inclusion of some so called *meta-events*. From the SMF specification: “specifies non-MIDI information useful to this format or to sequencers.” Examples of *meta-events* are *Time Signature* and *Key Signature*. In addition to the semantic elements just mentioned, there are also the typographical elements (such as line thickness, spacing, color, fonts, etc.) that all printed scores carry. This extra layer of information is completely absent in a SMF. However, from the point of view of encoding a printed score, the main limitation of MIDI is not the lack musical features or the absence of typographical data, but the assumption that musical durations, pitches

and loudnesses can be each fully and efficiently encoded with integers or even fractions. In a printed score, this is not the case for any of them. MIDI encodes only *magnitudes*: time interval magnitudes, pitch interval magnitudes, velocity magnitudes. While these may be sufficient attributes for an automated piano performance, they are not all the attributes of notes in a printed score.

46.3 Written note durations vs. MIDI delta-times

Assume a fixed tempo has been set. Assume that all magnitudes are represented with (and limited to) rational numbers. A time interval magnitude $d = 1/4$ has an infinity of equivalent representations in terms of magnitude: $d = 1/4 = 1/8 * 2 = 1/8 + 1/16 * 2 \dots$ etc. So, for example, while equivalent in magnitude, these are not the same notated durations:

```
abjad> m1 = measuretools.AnonymousMeasure([Note("c'4")])
abjad> m2 = measuretools.AnonymousMeasure(Note(0, (1, 8)) * 2)
abjad> tietools.TieSpanner(m2)
abjad> m3 = measuretools.AnonymousMeasure([Note(0, (1, 8))] + Note(0, (1, 16)) * 2)
abjad> tietools.TieSpanner(m3)
abjad> r = stafftools.RhythmicStaff([m1, m2, m3])
abjad> iotools.write_expr_to_ly(r, 'MIDI-1')
```



46.4 Written note pitch vs. MIDI note-on

A similar thing happens with pitches. In MIDI, key (pitch) number 61 is a half tone above middle C. But how is this pitch to be notated? As a C sharp or a B flat?

```
abjad> m1 = measuretools.AnonymousMeasure([Note(1, (1, 4))])
abjad> m2 = measuretools.AnonymousMeasure([Note(('df', 4), (1, 4))])
abjad> r = Staff([m1, m2])
abjad> iotools.write_expr_to_ly(r, 'MIDI-2')
```



46.5 Conclusion

MIDI was not designed for score representation. MIDI is a simple communication protocol intended for real-time control. As such, it naturally lacks the adequate model to represent the full range of information found in printed scores.

WHY LILYPOND IS RIGHT FOR ABJAD

Early versions of Abjad wrote MIDI files for input to Finale and Sibelius. Later versions of Abjad wrote .pbx files for input into Leland Smith's SCORE. Over time we found LilyPond superior to Finale, Sibelius and SCORE.

47.1 Nested tuplets works out of the box

LilyPond uses a single construct to nest tuplets arbitrarily:

```
\new stafftools.RhythmicStaff {  
  \time 7/8  
  \times 7/8 {  
    c8.  
    \times 7/5 { c16 c16 c16 c16 c16 }  
    \times 3/5 { c8 c8 c8 c8 c8 }  
  }  
}
```

```
abjad> staff = stafftools.RhythmicStaff([Measure((7, 8), [])])  
abjad> measure = staff[0]  
abjad> measure.append(Note('c8.'))  
abjad> measure.append(Tuplet(Fraction(7, 5), 5 * Note('c16')))  
abjad> spannertools.BeamSpanner(measure[-1])  
abjad> measure.append(Tuplet(Fraction(3, 5), 5 * Note('c8')))  
abjad> spannertools.BeamSpanner(measure[-1])  
abjad> Tuplet(Fraction(7, 8), measure.music)  
abjad> staff.override.tuplet_bracket.bracket_visibility = True  
abjad> staff.override.tuplet_bracket.padding = 1.6  
abjad> show(staff)
```



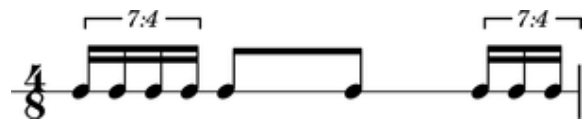
LilyPond's tuplet input syntax works the same as any other recursive construct.

47.2 Broken tuplets work out of the box

LilyPond engraves tupletted notes interrupted by nontupletted notes correctly:

```
\new Staff {
  \times 4/7 { c'16 c'16 c'16 c'16 }
  c'8 c'8
  \times 4/7 { c'16 c'16 c'16 }
}

abjad> t = Tuplet(Fraction(4, 7), Note(0, (1, 16)) * 4)
abjad> notes = Note(0, (1, 8)) * 2
abjad> u = Tuplet(Fraction(4, 7), Note(0, (1, 16)) * 3)
abjad> spannertools.BeamSpanner(t)
abjad> spannertools.BeamSpanner(notes)
abjad> spannertools.BeamSpanner(u)
abjad> measure = Measure((4, 8), [t] + notes + [u])
abjad> staff = stafftools.RhythmicStaff([measure])
abjad> show(staff)
```

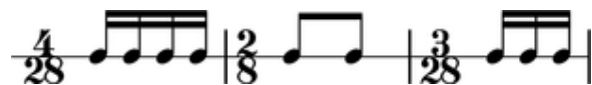


47.3 Nonbinary meters work out of the box

The rhythm above rewrites with time signatures in place of tuplets:

```
\new Staff {
  \time 4/28 c'16 c'16 c'16 c'16 |
  \time 2/8 c'8 c'8 |
  \time 3/28 c'16 c'16 c'16 |
}

abjad> t = Measure((4, 28), Note(0, (1, 16)) * 4)
abjad> u = Measure((2, 8), Note(0, (1, 8)) * 2)
abjad> v = Measure((3, 28), Note(0, (1, 16)) * 3)
abjad> spannertools.BeamSpanner(t)
abjad> spannertools.BeamSpanner(u)
abjad> spannertools.BeamSpanner(v)
abjad> staff = stafftools.RhythmicStaff([t, u, v])
abjad> show(staff)
```



The time signatures 4/28 and 3/28 here have a denominator not equal to 4, 8, 16 or any other nonnegative integer power of two. Abjad calls such time signatures **nonbinary meters** and LilyPond engraves them correctly.

47.4 Lilypond models the musical measure correctly

Most engraving packages make the concept of the measure out to be more important than it should. We see evidence of this wherever an engraving package makes it difficult for either a long note or the notes of a tuplet to cross a barline. These difficulties come from working the idea of measure-as-container deep into object model of the package.

There is a competing way to model the musical measure that we might call the measure-as-background way of thinking about things. Western notation practice started absent any concept of the barline, introduced the idea gradually, and

has since retreated from the necessity of the convention. Engraving packages that pick out an understanding of the barline from the 18th or 19th centuries subscribe to the measure-as-container view of things and oversimplify the problem. One result of this is to render certain barline-crossing rhythmic figures either an inelegant hack or an outright impossibility. LilyPond eschews the measure-as-container model in favor of the measure-as-background model better able to handle both earlier and later notation practice.

LILYPOND TEMPLATE GALLERY

Abjad provides a number of score templates in the `abjad/templates` directory:

```
abjad> from abjad.tools import configurationtools
abjad> configurationtools.list_abjad_templates()
('coventry.ly', 'lagos.ly', 'oedo.ly', 'paris.ly', 'tangiers.ly', 'thebes.ly', 'tirnaveni.ly')
```

Templates provide header, layout, paper and grob settings for different types of score.

48.1 Default LilyPond layout

```
abjad> import random
abjad> pitches = [random.randrange(0, 25) for x in range(32)]
abjad> staff_1 = Staff([])
abjad> staff_2 = Staff([])
abjad> score = Score([staff_1, staff_2])
abjad> staff_1.extend([Note(x, (1, 8)) for x in pitches[:16]])
abjad> staff_2.extend([Note(x, (1, 8)) for x in pitches[16:]])
abjad> show(score)
```



48.2 `lagos.ly`

```
abjad> pitches = [random.randrange(0, 25) for x in range(32)]
abjad> staff_1 = Staff([])
abjad> staff_2 = Staff([])
abjad> score = Score([staff_1, staff_2])
abjad> staff_1.extend([Note(x, (1, 8)) for x in pitches[:16]])
abjad> staff_2.extend([Note(x, (1, 8)) for x in pitches[16:]])
abjad> show(score, template = 'lagos')
```



48.3 oedo.ly

```
abjad> pitches = [random.randrange(0, 25) for x in range(32)]
abjad> staff_1 = Staff([])
abjad> staff_2 = Staff([])
abjad> score = Score([staff_1, staff_2])
abjad> staff_1.extend([Note(x, (1, 8)) for x in pitches[:16]])
abjad> staff_2.extend([Note(x, (1, 8)) for x in pitches[16:]])
abjad> show(score, template = 'oedo')
```



48.4 tangiers.ly

```
abjad> pitches = [random.randrange(0, 25) for x in range(32)]
abjad> staff_1 = Staff([])
abjad> staff_2 = Staff([])
abjad> score = Score([staff_1, staff_2])
abjad> staff_1.extend([Note(x, (1, 8)) for x in pitches[:16]])
abjad> staff_2.extend([Note(x, (1, 8)) for x in pitches[16:]])
abjad> show(score, template = 'tangiers')
```



48.5 tirnaveni.ly

```
abjad> pitches = [random.randrange(0, 25) for x in range(32)]
abjad> staff_1 = Staff([])
abjad> staff_2 = Staff([])
abjad> score = Score([staff_1, staff_2])
abjad> staff_1.extend([Note(x, (1, 8)) for x in pitches[:16]])
abjad> staff_2.extend([Note(x, (1, 8)) for x in pitches[16:]])
abjad> show(score, template = 'tirnaveni')
```



LILYPOND TEXT ALIGNMENT

LilyPond provides many ways to position text.

49.1 Default alignment

LilyPond left-aligns markup relative to the left edge of note head by default.

```
abjad> notes = notetools.make_repeated_notes(1, Fraction(1, 4))
abjad> staff = stafftools.RhythmicStaff(notes)
abjad> leaves = staff.leaves
abjad> markuptools.Markup('XX', 'up')(leaves[0])
abjad> show(staff, 'thebes')
```



49.2 TextScript #'self-alignment-X

Use #'self-alignment-X to left-, center- or right-align markup relative to the left edge of note head.

Note: changes to #'self-alignment-X do not change the fact that markup positioning is by default relative to the LEFT edge of note head.

```
abjad> notes = notetools.make_repeated_notes(3, Fraction(1, 4))
abjad> staff = stafftools.RhythmicStaff(notes)
abjad> leaves = staff.leaves
abjad> markuptools.Markup('XX', 'up')(leaves[0])
abjad> leaves[0].override.text_script.self_alignment_X = 'left'
abjad> markuptools.Markup('XX', 'up')(leaves[1])
abjad> leaves[1].override.text_script.self_alignment_X = 'center'
abjad> markuptools.Markup('XX', 'up')(leaves[2])
abjad> leaves[2].override.text_script.self_alignment_X = 'right'
abjad> show(staff, 'thebes')
```



49.3 TextScript #'X-offset

Use #'X-offset to offset markup by some number of magic units in the horizontal direction.

Note: Specify #'X-offset arguments as numbers like #2.5. Do not specify #'X-offset arguments as direction constants like #right.

Note: changes to #'X-offset do not change the fact that markup positioning is by default relative to the LEFT edge of note head.

```
abjad> notes = notetools.make_repeated_notes(4, Fraction(1, 4))
abjad> staff = stafftools.RhythmicStaff(notes)
abjad> leaves = staff.leaves
abjad> markuptools.Markup('XX', 'up')(leaves[0])
abjad> leaves[0].override.text_script.X_offset = 0
abjad> markuptools.Markup('XX', 'up')(leaves[1])
abjad> leaves[1].override.text_script.X_offset = 2
abjad> markuptools.Markup('XX', 'up')(leaves[2])
abjad> leaves[2].override.text_script.X_offset = 4
abjad> markuptools.Markup('XX', 'up')(leaves[3])
abjad> leaves[3].override.text_script.X_offset = 6
abjad> show(staff, 'thebes')
```



BIBLIOGRAPHY

ABJAD API

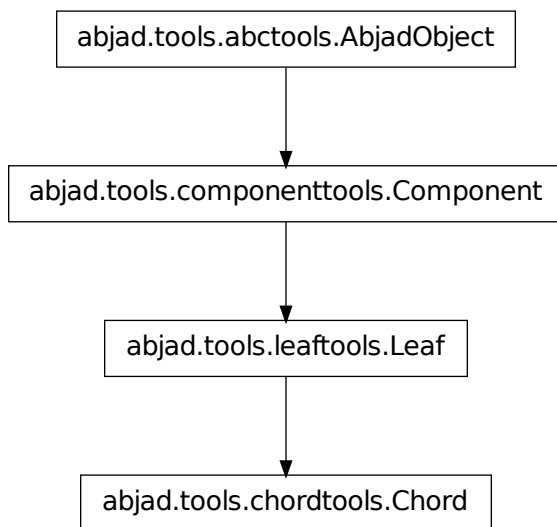
51.1 Abjad API

51.1.1 Abjad composition packages

`chordtools`

`chordtools` concrete classes

`chordtools.Chord`



```
class abjad.tools.chordtools.Chord.Chord(*args, **kwargs)
    Abjad model of a chord:
```

```
abjad> Chord([4, 13, 17], (1, 4))
Chord("<e' cs'' f''>4")
```

Return chord instance.

Read-only Properties

Chord.duration_in_seconds

Note: Inherited from `leaftools.Leaf`

Chord.fingered_pitches

Read-only fingered pitches:

```
abjad> staff = Staff("<c''' e'''>4 <d''' fs'''>4")
abjad> glockenspiel = instrumenttools.Glockenspiel()(staff)
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch(
    staff)

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Glockenspiel }
  \set Staff.shortInstrumentName = \markup { Gkspl. }
  <c' e'>4
  <d' fs'>4
}

abjad> staff[0].fingered_pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
```

Return tuple of named chromatic pitches.

Chord.format

Note: Inherited from `leaftools.Leaf`

Chord.leaf_index

Note: Inherited from `leaftools.Leaf`

Chord.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Chord.multiplied_duration

Note: Inherited from `leaftools.Leaf`

Chord.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Chord.parent

Note: Inherited from `componenttools.Component`

Chord.preprolated_duration

Note: Inherited from `leaftools.Leaf`

Chord.prolated_duration

Note: Inherited from `componenttools.Component`

Chord.prolation

Note: Inherited from `componenttools.Component`

Chord.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Chord.sounding_pitches

Read-only sounding pitches:

```
abjad> staff = Staff("<c'' e''>4 <d'' fs''>4")
abjad> glockenspiel = instrumenttools.Glockenspiel()(staff)
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch()
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Glockenspiel }
  \set Staff.shortInstrumentName = \markup { Gkspl. }
  <c' e'>4
  <d' fs'>4
}
```

```
abjad> staff[0].sounding_pitches
(NamedChromaticPitch("c''"), NamedChromaticPitch("e''"))
```

Return tuple of named chromatic pitches.

Chord.spanners

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`Chord.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`Chord.note_heads`

Get read-only tuple of note heads in chord:

```
abjad> chord = Chord([7, 12, 16], (1, 4))
abjad> chord.note_heads
(NoteHead("g'"), NoteHead("c'"), NoteHead("e'"))
```

Set chord note heads from any iterable:

```
abjad> chord = Chord([7, 12, 16], (1, 4))
abjad> chord.note_heads = [0, 2, 6]
abjad> chord
Chord("<c' d' fs'>4")
```

`Chord.written_duration`

Note: Inherited from `leaftools.Leaf`

`Chord.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`Chord.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

`Chord.written_pitches`

Get read-only tuple of pitches in chord:

```
abjad> chord = Chord([7, 12, 16], (1, 4))
abjad> chord.written_pitches
(NamedChromaticPitch("g'"), NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
```

Set chord pitches from any iterable:

```
abjad> chord = Chord([7, 12, 16], (1, 4))
abjad> chord.written_pitches = [0, 2, 6]
abjad> chord
Chord("<c' d' fs'>4")
```

Methods

`Chord.append(note_head_token)`

Append *note_head_token* to chord:

```
abjad> chord = Chord([4, 13, 17], (1, 4))
abjad> chord
Chord("<e' cs'' f''>4")

abjad> chord.append(19)
abjad> chord
Chord("<e' cs'' f'' g''>4")
```

Sort chord note heads automatically after append and return none.

`Chord.clear()`

Clear chord:

```
abjad> chord = Chord("<e' cs'' f''>4")
abjad> chord
Chord("<e' cs'' f''>4")

abjad> chord.clear()
abjad> chord
Chord('<>4')
```

Return none.

`Chord.extend(note_head_tokens)`

Extend chord with *note_head_tokens*:

```
abjad> chord = Chord([4, 13, 17], (1, 4))
abjad> chord
Chord("<e' cs'' f''>4")

abjad> chord.extend([2, 12, 18])
abjad> chord
Chord("<d' e' c'' cs'' f'' fs''>4")
```

Sort chord note heads automatically after extend and return none.

`Chord.pop(i=-1)`

Remove note head at index *i* in chord:

```
abjad> chord = Chord([4, 13, 17], (1, 4))
abjad> chord
Chord("<e' cs'' f''>4")

abjad> chord.pop(1)
NoteHead("cs'")

abjad> chord
Chord("<e' f''>4")
```

Return note head.

`Chord.remove(note_head)`

Remove *note_head* from chord:

```
abjad> chord = Chord([4, 13, 17], (1, 4))
abjad> chord
Chord("<e' cs'' f''>4")
```

```
abjad> chord.remove(chord[1])
abjad> chord
Chord("<e' f''>4")
```

Return none.

Special Methods

Chord.__and__(arg)

Note: Inherited from `leaftools.Leaf`

Chord.__contains__(arg)

Chord.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Chord.__delitem__(i)

Chord.__eq__(arg)
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Chord.__ge__(arg)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Chord.__getitem__(i)

Chord.__gt__(arg)
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Chord.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

Chord.__le__(arg)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Chord.__len__()`

`Chord.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Chord.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Chord.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Chord.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`Chord.__repr__()`

Note: Inherited from `leaftools.Leaf`

`Chord.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`Chord.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Chord.__setitem__(i, arg)`

`Chord.__str__()`

Note: Inherited from `leaftools.Leaf`

`Chord.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`Chord.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

chordtools functions

chordtools.all_are_chords

`abjad.tools.chordtools.all_are_chords.all_are_chords(*args)`

New in version 2.6. True when *expr* is a sequence of Abjad chords:

```
abjad> chords = [Chord("<c' e' g'>4"), Chord("<c' f' a'>4")]
```

```
abjad> chordtools.all_are_chords(chords)
True
```

True when *expr* is an empty sequence:

```
abjad> chordtools.all_are_chords([])
True
```

Otherwise false:

```
abjad> chordtools.all_are_chords('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

chordtools.arpeggiate_chord

`abjad.tools.chordtools.arpeggiate_chord.arpeggiate_chord(*args)`

New in version 1.1. Arpeggiate *chord*:

```
abjad> chord = Chord("<c' d' ef'>8")
```

```
abjad> chordtools.arpeggiate_chord(chord)
[Note("c'8"), Note("d'8"), Note("ef'8")]
```

Arpeggiated notes inherit *chord* written duration.

Arpeggiated notes do not inherit other *chord* attributes.

Return list of newly constructed notes. Changed in version 2.0: renamed `chordtools.arpeggiate()` to `chordtools.arpeggiate_chord()`.

chordtools.change_defective_chord_to_note_or_rest

`abjad.tools.chordtools.change_defective_chord_to_note_or_rest`. **change_defective_chord_to_note_or_rest**

New in version 1.1. Change zero-length *chord* to rest:

```
abjad> chord = Chord([], (3, 16))
```

```
abjad> chord
Chord('<>8.')
```

```
abjad> chordtools.change_defective_chord_to_note_or_rest(chord)
Rest('r8.')
```

Change length-one chord to note:

```
abjad> chord = Chord("<cs''>8.")
```

```
abjad> chord
Chord("<cs''>8.")
```

```
abjad> chordtools.change_defective_chord_to_note_or_rest(chord)
Note("cs''8.")
```

Return chords with length greater than one unchanged:

```
abjad> chord = Chord("<c' c'' cs''>8.")
```

```
abjad> chord
Chord("<c' c'' cs''>8.")
```

```
abjad> chordtools.change_defective_chord_to_note_or_rest(chord)
Chord("<c' c'' cs''>8.")
```

Return notes unchanged:

```
abjad> note = Note("c'4")
```

```
abjad> note
Note("c'4")
```

```
abjad> chordtools.change_defective_chord_to_note_or_rest(note)
Note("c'4")
```

Return rests unchanged:

```
abjad> rest = Rest('r4')
```

```
abjad> rest
Rest('r4')
```

```
abjad> chordtools.change_defective_chord_to_note_or_rest(rest)
Rest('r4')
```

Return note, rest, chord or none. Changed in version 2.0: renamed `chordtools.cast_defective()` to `chordtools.change_defective_chord_to_note_or_rest()`.

chordtools.color_chord_note_heads_by_pitch_class_color_map

abjad.tools.chordtools.color_chord_note_heads_by_pitch_class_color_map.**color_chord_note_heads_by_pitch_class_color_map**

New in version 2.0. Color *chord* note heads by pitch-class *color_map*:

```
abjad> chord = Chord([12, 14, 18, 21, 23], (1, 4))

abjad> pitches = [[-12, -10, 4], [-2, 8, 11, 17], [19, 27, 30, 33, 37]]
abjad> colors = ['red', 'blue', 'green']
abjad> color_map = pitchtools.NumberedChromaticPitchClassColorMap(pitches, colors)

abjad> chordtools.color_chord_note_heads_by_pitch_class_color_map(chord, color_map)
Chord("<c'' d'' fs'' a'' b''>4")

abjad> f(chord)
<
    \tweak #'color #red
    c''
    \tweak #'color #red
    d''
    \tweak #'color #green
    fs''
    \tweak #'color #green
    a''
    \tweak #'color #blue
    b''
>4
```

Also works on notes:

```
abjad> note = Note("c'4")

abjad> chordtools.color_chord_note_heads_by_pitch_class_color_map(note, color_map)
Note("c'4")

abjad> f(note)
\once \override NoteHead #'color = #red
c'4
```

When *chord* is neither a chord nor note return *chord* unchanged:

```
abjad> staff = Staff([])

abjad> chordtools.color_chord_note_heads_by_pitch_class_color_map(staff, color_map)
Staff{}
```

Return *chord*. Changed in version 2.0: renamed `chordtools.color_note_heads_by_pc()` to `chordtools.color_chord_note_heads_by_pitch_class_color_map()`.

chordtools.divide_chord_by_chromatic_pitch_number

abjad.tools.chordtools.divide_chord_by_chromatic_pitch_number.**divide_chord_by_chromatic_pitch_number**

New in version 1.1. Divide *chord* by chromatic *pitch* number:

```
abjad> chord = Chord(range(12), Duration(1, 4))
```

```
abjad> chord
Chord("<c' cs' d' ef' e' f' fs' g' af' a' bf' b'>4")
```

```
abjad> chordtools.divide_chord_by_chromatic_pitch_number(chord, pitchtools.NamedChromaticPitch(6)
(Chord("<fs' g' af' a' bf' b'>4"), Chord("<c' cs' d' ef' e' f'>4")))
```

Input *chord* may be a note, rest or chord but not a skip.

Zero-length parts return rests, length-one parts return notes and other parts return chords.

Return pair of newly constructed leaves. Changed in version 2.0: renamed `chordtools.split_by_pitch_number()` to `chordtools.divide_chord_by_chromatic_pitch_number()`

`chordtools.divide_chord_by_diatonic_pitch_number`

`abjad.tools.chordtools.divide_chord_by_diatonic_pitch_number.divide_chord_by_diatonic_pitch_number`
New in version 1.1. Divide *chord* by diatonic *pitch* number:

```
abjad> chord = Chord(range(12), Duration(1, 4))
```

```
abjad> chord
Chord("<c' cs' d' ef' e' f' fs' g' af' a' bf' b'>4")
```

```
abjad> chordtools.divide_chord_by_diatonic_pitch_number(chord, pitchtools.NamedChromaticPitch(6)
(Chord("<f' fs' g' af' a' bf' b'>4"), Chord("<c' cs' d' ef' e'>4")))
```

Input *chord* may be a note, rest or chord but not a skip.

Zero-length parts return as rests, length-one parts return as notes and other parts return as chords.

Return pair of newly constructed leaves. Changed in version 2.0: renamed `chordtools.split_by_altitude()` to `chordtools.divide_chord_by_diatonic_pitch_number()`.

`chordtools.get_arithmetic_mean_of_chord`

`abjad.tools.chordtools.get_arithmetic_mean_of_chord.get_arithmetic_mean_of_chord(*args)`
New in version 2.0. Get arithmetic mean of chromatic pitch number of pitches in *chord*:

```
abjad> chord = Chord("<g' c'' e''>4")
```

```
abjad> chordtools.get_arithmetic_mean_of_chord(chord)
11.666666666666666
```

Return none when *chord* is empty:

```
abjad> chord = Chord("< >4")
```

```
abjad> chordtools.get_arithmetic_mean_of_chord(chord) is None
True
```

Return number or none.

`chordtools.get_note_head_from_chord_by_pitch`

`abjad.tools.chordtools.get_note_head_from_chord_by_pitch.get_note_head_from_chord_by_pitch`
New in version 2.0. Get note head from *chord* by *pitch*:

```
abjad> chord = Chord("<c' d' b'>4")

abjad> chordtools.get_note_head_from_chord_by_pitch(chord, 14)
NoteHead("d' ")
```

Raise missing note head error when *chord* contains no note head with pitch equal to *pitch*.

Raise extra note head error when *chord* contains more than one note head with pitch equal to *pitch*. Changed in version 2.0: renamed `chordtools.get_note_head()` to `chordtools.get_note_head_from_chord_by_pitch()`.

`chordtools.iterate_chords_backward_in_expr`

`abjad.tools.chordtools.iterate_chords_backward_in_expr.iterate_chords_backward_in_expr(*args)`
 New in version 2.0. Iterate chords backward in *expr*:

```
abjad> staff = Staff("<e' g' c'>8 a'8 r8 <d' f' b'>8 r2")

abjad> f(staff)
\new Staff {
    <e' g' c'>8
    a'8
    r8
    <d' f' b'>8
    r2
}

abjad> for chord in chordtools.iterate_chords_backward_in_expr(staff):
...     chord
Chord("<d' f' b'>8")
Chord("<e' g' c'>8")
```

Ignore threads.

Return generator.

`chordtools.iterate_chords_forward_in_expr`

`abjad.tools.chordtools.iterate_chords_forward_in_expr.iterate_chords_forward_in_expr(*args)`
 New in version 2.0. Iterate chords forward in *expr*:

```
abjad> staff = Staff("<e' g' c'>8 a'8 r8 <d' f' b'>8 r2")

abjad> f(staff)
\new Staff {
    <e' g' c'>8
    a'8
    r8
    <d' f' b'>8
    r2
}

abjad> for chord in chordtools.iterate_chords_forward_in_expr(staff):
...     chord
Chord("<e' g' c'>8")
Chord("<d' f' b'>8")
```


Ignore threads.

Return generator.

`chordtools.yield_all_subchords_of_chord`

`abjad.tools.chordtools.yield_all_subchords_of_chord.yield_all_subchords_of_chord(*args)`

New in version 2.0. Yield all subchords of *chord* in binary string order:

```
abjad> chord = Chord("<c' d' af' a'>4")

abjad> for subchord in chordtools.yield_all_subchords_of_chord(chord):
...     subchord
...
Rest('r4')
Note("c'4")
Note("d'4")
Chord("<c' d'>4")
Note("af'4")
Chord("<c' af'>4")
Chord("<d' af'>4")
Chord("<c' d' af'>4")
Note("a'4")
Chord("<c' a'>4")
Chord("<d' a'>4")
Chord("<c' d' a'>4")
Chord("<af' a'>4")
Chord("<c' af' a'>4")
Chord("<d' af' a'>4")
Chord("<c' d' af' a'>4")
```

Include empty chord as rest.

Return generator of newly constructed leaves. Changed in version 2.0: renamed `chordtools.subchords()` to `chordtools.yield_all_subchords_of_chord()`.

`chordtools.yield_groups_of_chords_in_sequence`

`abjad.tools.chordtools.yield_groups_of_chords_in_sequence.yield_groups_of_chords_in_sequence`

New in version 2.0. Yield groups of chords in *sequence*:

```
abjad> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
```

```

        <c' ' e''>8
    }

```

```

abjad> for chord in chordtools.yield_groups_of_chords_in_sequence(staff):
...     chord
...
(Chord("<e' g'>8"), Chord("<f' a'>8"))
(Chord("<b' d''>8"), Chord("<c' ' e''>8"))

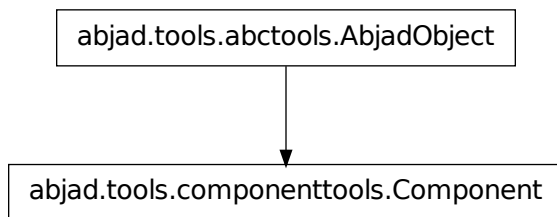
```

Return generator.

componenttools

componenttools concrete classes

componenttools.Component



class `abjad.tools.componenttools.Component.Component` **Component**

Read-only Properties

`Component.format`

Read-only LilyPond input format of component.

`Component.marks`

Read-only tuple of marks attached to component.

`Component.override`

Read-only reference to LilyPond grob override component plug-in.

`Component.parent`

`Component.prolated_duration`

`Component.prolation`

`Component.set`

Read-only reference LilyPond context setting component plug-in.

`Component.spanners`

Read-only reference to unordered set of spanners attached to component.

Special Methods

`Component.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Component.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Component.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Component.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`Component.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Component.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Component.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Component.__mul__(n)`

`Component.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Component.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`Component.__rmul__(n)`

`Component.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Component.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

componenttools functions

componenttools.all_are_components

`abjad.tools.componenttools.all_are_components.all_are_components(expr, classes=None)`

New in version 1.1. True when elements in *expr* are all components:

```
abjad> componenttools.all_are_components(3 * Note("c'4"))
True
```

Otherwise false:

```
abjad> componenttools.all_are_components(['foo', 'bar'])
False
```

True when elements in *expr* are all *classes*:

```
abjad> componenttools.all_are_components(3 * Note("c'4"), classes = Note)
True
```

Otherwise false:

```
abjad> componenttools.all_are_components(['foo', 'bar'], classes = Note)
False
```

Return boolean.

`componenttools.all_are_components_in_same_parent`

`abjad.tools.componenttools.all_are_components_in_same_parent.all_are_components_in_same_pa`

New in version 1.1. True when elements in *expr* are all components in same parent. Otherwise false:

```
abjad> staff = Staff(notetools.make_notes([12, 14, 16], [(1, 8)]))
abjad> componenttools.all_are_components_in_same_parent(staff.leaves)
True
```

True when elements in *expr* are all *klases* in same parent. Otherwise false:

```
abjad> staff = Staff(notetools.make_notes([12, 14, 16], [(1, 8)]))
abjad> componenttools.all_are_components_in_same_parent(staff.leaves, classes = (Note, ))
True
```

Return boolean.

`componenttools.all_are_components_in_same_score`

`abjad.tools.componenttools.all_are_components_in_same_score.all_are_components_in_same_sco`

New in version 1.1. True when elements in *expr* are all components in same score. Otherwise false:

```
abjad> score = Score([Staff("c'8 d'8 e'8")])
abjad> componenttools.all_are_components_in_same_score(score.leaves)
True
```

True when elements in *expr* are all *klases* in same score. Otherwise false:

```
abjad> score = Score([Staff("c'8 d'8 e'8")])
abjad> componenttools.all_are_components_in_same_score(score.leaves, classes = (Note, ))
True
```

Return boolean.

`componenttools.all_are_components_in_same_thread`

`abjad.tools.componenttools.all_are_components_in_same_thread.all_are_components_in_same_th`

New in version 1.1. True when elements in *expr* are all components in same thread. Otherwise false:

```
abjad> voice = Voice("c'8 d'8 e'8")
abjad> componenttools.all_are_components_in_same_thread(voice.leaves)
True
```

True when elements in *expr* are all *klases* in same thread. Otherwise false:

```
abjad> voice = Voice("c'8 d'8 e'8")
abjad> componenttools.all_are_components_in_same_thread(voice.leaves, classes = Note)
True
```

Return boolean.

`componenttools.all_are_components_scalable_by_multiplier`

`abjad.tools.componenttools.all_are_components_scalable_by_multiplier.all_are_components_scalable_by_multiplier`

New in version 1.1. True when *components* are all scalable by *multiplier*:

```
abjad> components = [Note(0, (1, 8))]
abjad> componenttools.all_are_components_scalable_by_multiplier(components, Duration(3, 2))
True
```

Otherwise false:

```
abjad> components = [Note(0, (1, 8))]
abjad> componenttools.all_are_components_scalable_by_multiplier(components, Duration(2, 3))
False
```

Return boolean. Changed in version 2.0: renamed `durationtools.are_scalable()` to `componenttools.all_are_components_scalable_by_multiplier()`.

`componenttools.all_are_contiguous_components`

`abjad.tools.componenttools.all_are_contiguous_components.all_are_contiguous_components` (*expr*, *klass*, *al-*, *low_*)

New in version 1.1. True when elements in *expr* are all contiguous components. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes*. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components(staff.leaves, classes = Note)
True
```

Return boolean.

`componenttools.all_are_contiguous_components_in_same_parent`

`abjad.tools.componenttools.all_are_contiguous_components_in_same_parent.all_are_contiguous_components_in_same_parent`

New in version 1.1. True when elements in *expr* are all contiguous components in same parent. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components_in_same_parent(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same parent. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components_in_same_parent(staff.leaves, classes = Note)
True
```

Return boolean.

componenttools.all_are_contiguous_components_in_same_score

`abjad.tools.componenttools.all_are_contiguous_components_in_same_score.all_are_contiguous_c`

New in version 1.1. True when elements in *expr* are all contiguous components in same score. Otherwise false:

```
abjad> score = Score([Staff("c'8 d'8 e'8")])
abjad> componenttools.all_are_contiguous_components_in_same_score(score.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same score. Otherwise false:

```
abjad> score = Score([Staff("c'8 d'8 e'8")])
abjad> componenttools.all_are_contiguous_components_in_same_score(score.leaves, classes = Note)
True
```

Return boolean.

componenttools.all_are_contiguous_components_in_same_thread

`abjad.tools.componenttools.all_are_contiguous_components_in_same_thread.all_are_contiguous_c`

New in version 1.1. True when elements in *expr* are all contiguous components in same thread. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components_in_same_thread(staff.leaves)
True
```

True when elements in *expr* are all contiguous *classes* in same thread. Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8")
abjad> componenttools.all_are_contiguous_components_in_same_thread(staff.leaves, classes = Note)
True
```

Return boolean.

componenttools.all_are_orphan_components

`abjad.tools.componenttools.all_are_orphan_components.all_are_orphan_components(expr)`

New in version 2.0. True when *expr* is an iterable of zero or more orphan components.

Otherwise false.

componenttools.all_are_thread_contiguous_components

abjad.tools.componenttools.all_are_thread_contiguous_components.**all_are_thread_contiguous_components**

New in version 1.1. True when elements in *expr* are all thread-contiguous components:

```
t = Voice(notetools.make_repeated_notes(4))
t.insert(2, Voice(notetools.make_repeated_notes(2)))
Container(t[:2])
Container(t[-2:])
pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)

\new Voice {
  {
    c'8
    d'8
  }
  \new Voice {
    e'8
    f'8
  }
  {
    g'8
    a'8
  }
}

assert _are_thread_contiguous_components(t[0:1] + t[-1:])
assert _are_thread_contiguous_components(t[0][:] + t[-1:])
assert _are_thread_contiguous_components(t[0:1] + t[-1][:])
assert _are_thread_contiguous_components(t[0][:] + t[-1][:])
```

Return boolean.

Thread-contiguous components are, by definition, spannable.

componenttools.component_to_parentage_signature

abjad.tools.componenttools.component_to_parentage_signature.**component_to_parentage_signature**

New in version 1.1. Change *component* to parentage signature:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> note = staff.leaves[0]
abjad> print componenttools.component_to_parentage_signature(note)
  root: Staff-... (...)
  score:
staffgroup:
  staff: Staff-...
  voice:
  self: Note-...
```

Return parentage signature.

componenttools.component_to_pitch_and_rhythm_skeleton

abjad.tools.componenttools.component_to_pitch_and_rhythm_skeleton.**component_to_pitch_and_rhythm_skeleton**

New in version 2.0. Change *component* to pitch and rhythm skeleton:

```
abjad> tuplet = Tuplet(Fraction(3, 4), "c'8 d'8 e'8 f'8")
abjad> measure = Measure((6, 16), [tuplet])
abjad> staff = Staff([measure])
abjad> score = Score(staff * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
```

```
abjad> skeleton = componenttools.component_to_pitch_and_rhythm_skeleton(score)
```

```
abjad> print skeleton
```

```
Score([
  Staff([
    Measure((6, 16), [
      Tuplet(Fraction(3, 4), [
        Note(('c', 4), Duration(1, 8)),
        Note(('d', 4), Duration(1, 8)),
        Note(('e', 4), Duration(1, 8)),
        Note(('f', 4), Duration(1, 8))
      ])
    ])
  ]),
  Staff([
    Measure((6, 16), [
      Tuplet(Fraction(3, 4), [
        Note(('g', 4), Duration(1, 8)),
        Note(('a', 4), Duration(1, 8)),
        Note(('b', 4), Duration(1, 8)),
        Note(('c', 5), Duration(1, 8))
      ])
    ])
  ])
])
```

```
abjad> new = eval(skeleton)
```

```
abjad> new
```

```
Score<<2>>
```

```
abjad> f(new)
```

```
\new Score <<
  \new Staff {
    {
      \time 6/16
      \fraction \times 3/4 {
        c'8
        d'8
        e'8
        f'8
      }
    }
  }
  \new Staff {
    {
      \time 6/16
      \fraction \times 3/4 {
        g'8
      }
    }
  }
}
```

```

        a'8
        b'8
        c''8
    }
}
>>

```

Return string.

`componenttools.component_to_score_depth`

`abjad.tools.componenttools.component_to_score_depth.component_to_score_depth(component)`

New in version 1.1. Change *component* to score depth:

```

abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> componenttools.component_to_score_depth(staff.leaves[0])
2

```

Return nonnegative integer.

`componenttools.component_to_score_index`

`abjad.tools.componenttools.component_to_score_index.component_to_score_index(component)`

New in version 2.0. Change *component* to score index:

```

abjad> staff_1 = Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_n
abjad> staff_2 = Staff([tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_n
abjad> score = Score([staff_1, staff_2])
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
\new Score <<
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
      a'8
    }
  }
  \new Staff {
    \times 2/3 {
      b'8
      c''8
      d''8
    }
  }
}
>>

abjad> for leaf in score.leaves:
...     leaf, componenttools.component_to_score_index(leaf)

```

```
...
(Note("c'8"), (0, 0, 0))
(Note("d'8"), (0, 0, 1))
(Note("e'8"), (0, 0, 2))
(Note("f'8"), (0, 1, 0))
(Note("g'8"), (0, 1, 1))
(Note("a'8"), (0, 1, 2))
(Note("b'8"), (1, 0, 0))
(Note("c''8"), (1, 0, 1))
(Note("d''8"), (1, 0, 2))
```

Return tuple of zero or more nonnegative integers.

componenttools.component_to_score_root

`abjad.tools.componenttools.component_to_score_root.component_to_score_root(component)`
 New in version 1.1. Change *component* to score root:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> note = staff.leaves[0]
abjad> componenttools.component_to_score_root(note)
Staff{1}
```

Return score root.

componenttools.component_to_tuplet_depth

`abjad.tools.componenttools.component_to_tuplet_depth.component_to_tuplet_depth(component)`
 New in version 1.1. Change *component* to tuplet depth:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> note = staff.leaves[0]

abjad> componenttools.component_to_tuplet_depth(note)
1

abjad> componenttools.component_to_tuplet_depth(tuplet)
0

abjad> componenttools.component_to_tuplet_depth(staff)
0
```

Return nonnegative integer.

componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts

`abjad.tools.componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts.copy_and_partition_governed_component_subtree_by_leaf_counts(container, leaf_counts)`

New in version 1.1. Copy *container* and partition copy according to *leaf_counts*:

```
abjad> voice = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_note("c'8", 3)))
abjad> spannertools.BeamSpanner(voice[0].leaves)
BeamSpanner(c'8, c'8, c'8)
```

```

abjad> spannertools.BeamSpanner(voice[1].leaves)
BeamSpanner(c'8, c'8, c'8)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
      d'8
      e'8 ]
  }
  \times 2/3 {
    f'8 [
      g'8
      a'8 ]
  }
}

abjad> first, second, third = componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts(voice)

abjad> f(first)
\new Voice {
  \times 2/3 {
    c'8 [ ]
  }
}

abjad> f(second)
\new Voice {
  \times 2/3 {
    d'8 [
      e'8 ]
  }
}

abjad> f(third)
\new Voice {
  \times 2/3 {
    f'8 [
      g'8
      a'8 ]
  }
}

```

Set *leaf_counts* to an iterable of zero or more positive integers.

Return a list of parts equal in length to that of *leaf_counts*. Changed in version 2.0: renamed `clonewp.by_leaf_counts_with_parentage()` to `componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts()`.

componenttools.copy_components_and_covered_spanners

```
abjad.tools.componenttools.copy_components_and_covered_spanners.copy_components_and_covered_spanners
```

New in version 1.1. Clone *components* and covered spanners.

The *components* must be thread-contiguous.

Covered spanners are those spanners that cover *components*.

The steps taken in this function are as follows. Withdraw *components* from crossing spanners. Preserve spanners that *components* cover. Deep copy *components*. Reapply crossing spanners to source *components*. Return copied components with covered spanners.

```
abjad> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  }
  {
    \time 2/8
    e'8
    f'8 ]
  }
  {
    \time 2/8
    g'8
    a'8
  }
}

abjad> result = componenttools.copy_components_and_covered_spanners(voice.leaves)
abjad> result
(Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"))

abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
  g'8
  a'8
}

abjad> voice.leaves[0] is new_voice.leaves[0]
False
```

Clone *components* a total of *n* times.

```
abjad> result = componenttools.copy_components_and_covered_spanners(voice.leaves[:2], n = 3)
abjad> result
(Note("c'8"), Note("d'8"), Note("c'8"), Note("d'8"), Note("c'8"), Note("d'8"))

abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
  c'8
  d'8
  c'8
  d'8
  c'8
  d'8
}
```

```
}
```

Changed in version 2.0: renamed `clone.covered()` to `componenttools.copy_components_and_covered_spanners()`.
 in version 2.0: renamed `componenttools.clone_components_and_covered_spanners()` to `componenttools.copy_components_and_covered_spanners()`.

`componenttools.copy_components_and_fracture_crossing_spanners`

`abjad.tools.componenttools.copy_components_and_fracture_crossing_spanners.copy_components_and_fracture_crossing_spanners`

New in version 1.1. Clone *components* and fracture crossing spanners.

The *components* must be thread-contiguous.

The steps this function takes are as follows. Deep copy *components*. Deep copy spanners that attach to any component in *components*. Fracture spanners that attach to components not in *components*. Return Python list of copied components.

```
abjad> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    \time 2/8
    e'8
    f'8 ]
  }
  {
    \time 2/8
    g'8
    a'8
  }
}
```

```
abjad> result = componenttools.copy_components_and_fracture_crossing_spanners(voice.leaves[2:4])
abjad> result
(Note("e'8"), Note("f'8"))
```

```
abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
  e'8 [
  f'8 ]
}
```

```
abjad> voice.leaves[2] is new_voice.leaves[0]
False
```

Clone *components* a total of *n* times.

```

abjad> result = componenttools.copy_components_and_fracture_crossing_spanners(voice.leaves[2:4],
abjad> result
(Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"))

abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
  e'8 [
  f'8 ]
  e'8 [
  f'8 ]
  e'8 [
  f'8 ]
}

```

Changed in version 2.0: renamed `clone.fracture()` to `componenttools.copy_components_and_fracture_crossing_spanners()`
 in version 2.0: renamed `componenttools.clone_components_and_fracture_crossing_spanners()`
 to `componenttools.copy_components_and_fracture_crossing_spanners()`.

`componenttools.copy_components_and_immediate_parent_of_first_component`

`abjad.tools.componenttools.copy_components_and_immediate_parent_of_first_component.copy_components_and_immediate_parent_of_first_component`
 New in version 1.1. Clone *components* and immediate parent of first component.

The *components* must be thread-contiguous.

Return in newly created container equal to type of first element in *components*.

If the parent of the first element in *components* is a tuplet then insure that the tuplet multiplier of the function output equals the tuplet multiplier of the parent of the first element in *components*.

```

abjad> voice = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_note(
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
    d'8
    e'8
  }
  \times 2/3 {
    f'8 ]
    g'8
    a'8
  }
  \times 2/3 {
    b'8
    c''8
    d''8
  }
}
abjad> new_tuplet = componenttools.copy_components_and_immediate_parent_of_first_component(voice)
abjad> new_tuplet
FixedDurationTuplet(1/6, [c'8, d'8])
abjad> f(new_tuplet)
\times 2/3 {
  c'8 [

```

```

    d'8 ]
}

```

Parent-contiguity is not required. Thread-contiguous *components* suffice.

```

abjad> new_tuplet = componenttools.copy_components_and_immediate_parent_of_first_component(voice)
abjad> new_tuplet
FixedDurationTuplet(5/12, [c'8, d'8, e'8, f'8, g'8])
abjad> f(new_tuplet)
\times 2/3 {
    c'8 [
    d'8
    e'8
    f'8 ]
    g'8
}

```

Note: this function copies only the *immediate parent* of the first element in *components*. This function ignores any further parentage of *components* above the immediate parent of *components*.

Todo

this function should (but does not) copy marks that attach to *components* and to the immediate parent of the first component; extend function to do so.

Changed in version 2.0: renamed `clonewp.with_parent()` to `componenttools.copy_components_and_immediate_parent_of_first_component()`. Changed in version 2.0: renamed `componenttools.clone_components_and_immediate_parent_of_first_component()` to `componenttools.copy_components_and_immediate_parent_of_first_component()`.

componenttools.copy_components_and_remove_all_spanners

```

abjad.tools.componenttools.copy_components_and_remove_all_spanners.copy_components_and_remove_all_spanners

```

New in version 1.1. Clone *components* and remove all spanners.

The *components* must be thread-contiguous.

The steps taken by this function are as follows. Withdraw all components at any level in *components* from spanners. Deep copy unspanned components in *components*. Reapply spanners to all components at any level in *components*.

```

abjad> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    \time 2/8
    e'8
  }
}

```



```

        f'8 ]
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> result = componenttools.copy_components_and_remove_all_spanners(voice.leaves[2:4])
abjad> result
(Note("e'8"), Note("f'8"))

abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
    e'8
    f'8
}

abjad> voice.leaves[2] is new_voice.leaves[0]
False

```

Clone *components* a total of n times.

```

abjad> result = componenttools.copy_components_and_remove_all_spanners(voice.leaves[2:4], n = 3)
abjad> result
(Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"), Note("e'8"), Note("f'8"))

abjad> new_voice = Voice(result)
abjad> f(new_voice)
\new Voice {
    e'8
    f'8
    e'8
    f'8
    e'8
    f'8
}

```

Changed in version 2.0: renamed `clone.unspan()` to `componenttools.copy_components_and_remove_all_sp`
in version 2.0: renamed `componenttools.clone_components_and_remove_all_spanners()`
to `componenttools.copy_components_and_remove_all_spanners()`.

componenttools.copy_governed_component_subtree_by_leaf_range

```
abjad.tools.componenttools.copy_governed_component_subtree_by_leaf_range.copy_governed_comp
```

New in version 1.1. Clone governed *component* subtree by leaf range.

Governed subtree means *component* together with children of *component*.

Leaf range refers to the sequential parentage of *component* from *start* leaf index to *stop* leaf index:

```

abjad> t = Staff([Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> f(t)

```

```

\new Staff {
  \new Voice {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
      a'8
    }
  }
}

abjad> u = componenttools.copy_governed_component_subtree_by_leaf_range(t, 1, 5)
abjad> f(u)
\new Staff {
  \new Voice {
    \times 2/3 {
      d'8
      e'8
    }
    \times 2/3 {
      f'8
      g'8
    }
  }
}

```

Clone sequential containers in leaves' parentage up to the first parallel container in leaves' parentage.

Trim and shrink cloned containers as necessary.

When *stop* is *none* copy all leaves from *start* forward. Changed in version 2.0: renamed `clonewp.by_leaf_range_with_parentage()` to `componenttools.copy_governed_component_subtree_by_leaf_range()`. Changed in version 2.0: renamed `componenttools.clone_governed_component_subtree_by_leaf_range()` to `componenttools.copy_governed_component_subtree_by_leaf_range()`.

`componenttools.copy_governed_component_subtree_from_prolated_offset_to`

`abjad.tools.componenttools.copy_governed_component_subtree_from_prolated_offset_to`. **copy_gov**

New in version 1.1. Clone governed *component* subtree from *start* prolated duration to *stop* prolated duration.

Governed subtree refers to *component* together with the children of *component*:

```

abjad> voice = Voice(notetools.make_repeated_notes(2))
abjad> voice.append(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_note
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voi
abjad> f(voice)
\new Voice {
  c'8
  d'8
  \times 2/3 {
    e'8
  }
}

```

```

        f'8
        g'8
    }
}

abjad> new = componenttools.copy_governed_component_subtree_from_prolated_offset_to(voice, (0, 8))
abjad> f(new)
\new Voice {
    c'8
    d'8
    \times 2/3 {
        e'8
        f'16
    }
}

```

Raise contiguity error if asked to slice a parallel container.

```

abjad> staff = Staff(Voice("c'8 d'8") * 2)
abjad> staff.is_parallel = True
abjad> f(staff)
\new Staff <<
\new Voice {
    c'8
    d'8
}
\new Voice {
    c'8
    d'8
}
>>

```

Raise contiguity error when attempting to copy fleaves from parallel container.

But note that cases with `0 = start` work correctly:

```

abjad> new = componenttools.copy_governed_component_subtree_from_prolated_offset_to(voice, (0, 8))
abjad> f(new)
\new Voice {
    c'8
}

```

Cases with `0 < start` do not work correctly:

```

abjad> new = componenttools.copy_governed_component_subtree_from_prolated_offset_to(voice, (1, 8))
abjad> f(new)
\new Voice {
    c'8
    d'8
}

```

Create ad hoc tuplets as required:

```

abjad> voice = Voice([Note("c'4")])
abjad> new = componenttools.copy_governed_component_subtree_from_prolated_offset_to(voice, 0, (1, 4))
abjad> f(new)
\new Voice {
    \times 2/3 {
        c'8
    }
}

```

```
}
}
```

Function does NOT clone parentage of *component* when *component* is a leaf:

```
abjad> voice = Voice([Note("c'4")])
abjad> new_leaf = componenttools.copy_governed_component_subtree_from_prolated_offset_to(voice[0])
abjad> f(new_leaf)
c'8
abjad> new_leaf._parentage.parent is None
True
```

Return (untrimmed_copy, first_dif, second_dif). Changed in version 2.0: renamed `componenttools.clone_governed_component_subtree_from_prolated_duration_to()` to `componenttools.copy_governed_component_subtree_from_prolated_offset_to()`.

componenttools.cut_component_at_prolated_duration

`abjad.tools.componenttools.cut_component_at_prolated_duration.cut_component_at_prolated_duration`

New in version 2.0. Cut *component* at dotted *prolated_duration*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> componenttools.cut_component_at_prolated_duration(staff, Duration(1, 32))
abjad> f(staff)
\new Staff {
    c'16. [
    d'8
    e'8
    f'8 ]
}
```

Cut *component* at tied *prolated_duration*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> componenttools.cut_component_at_prolated_duration(staff, Duration(3, 64))
abjad> f(staff)
\new Staff {
    c'16 [ ~
    c'64
    d'8
    e'8
    f'8 ]
}
```

Cut *component* at nonbinary *prolated_duration*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> componenttools.cut_component_at_prolated_duration(staff, Duration(1, 24))
abjad> f(staff)
\new Staff {
```

```

\times 2/3 {
    c'8 [
    ]
    d'8
    e'8
    f'8 ]
}

```

Return none.

componenttools.extend_in_parent_of_component_and_do_not_grow_spanners

abjad.tools.componenttools.extend_in_parent_of_component_and_do_not_grow_spanners.**extend_in**

New in version 1.1. Extend *components* in parent of *component* and do not grow spanners:

```

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> t = Voice("c'8 d'8 e'8")
abjad> spannertools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8, e'8)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> componenttools.extend_in_parent_of_component_and_do_not_grow_spanners(t[-1], notes)
[Note("e'8"), Note("c'8"), Note("d'8"), Note("e'8")]

abjad> print t.format
\new Voice {
    c'8 [
    d'8
    e'8 ]
    c'8
    d'8
    e'8
}

```

Return list of *component* and *components*. Changed in version 2.0: renamed `extend_in_parent()` to `extend_in_parent_of_component_and_do_not_grow_spanners()`.

componenttools.extend_in_parent_of_component_and_grow_spanners

abjad.tools.componenttools.extend_in_parent_of_component_and_grow_spanners.**extend_in_parent**

New in version 2.0. Extend *new_components* in parent of *component* and grow spanners:

```

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> voice = Voice(notes)
abjad> spannertools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8)

abjad> f(voice)
\new Voice {
    c'8 [
    d'8
    e'8 ]
}

```

```

abjad> new_components = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> componenttools.extend_in_parent_of_component_and_grow_spanners(voice.leaves[-1], new_comp
[Note("e'8"), Note("c'8"), Note("d'8"), Note("e'8")]

abjad> f(voice)
\new Voice {
    c'8 [
    d'8
    e'8
    c'8
    d'8
    e'8 ]
}

```

Return *component* and *new_components* together in list.

componenttools.extend_left_in_parent_of_component_and_do_not_grow_spanners

abjad.tools.componenttools.extend_left_in_parent_of_component_and_do_not_grow_spanners.**ext**

New in version 1.1. Extend *components* left in parent of *component* and do not grow spanners:

```

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> t = Voice(notes)
abjad> spannertools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8, e'8)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8")]
abjad> componenttools.extend_left_in_parent_of_component_and_do_not_grow_spanners(t[0], notes)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("c'8")]

abjad> print t.format
\new Voice {
    c'8
    d'8
    e'8
    c'8 [
    d'8
    e'8 ]
}

```

Return *components* and *component* together in newly created list. Changed in version 2.0: renamed `extend_left_in_parent()` to `extend_left_in_parent_of_component_and_do_not_grow_spanners()`.

componenttools.extend_left_in_parent_of_component_and_grow_spanners

abjad.tools.componenttools.extend_left_in_parent_of_component_and_grow_spanners.**extend_left**

New in version 2.0. Extend *new_components* left in parent of *component* and grow spanners:

```

abjad> voice = Voice("c'8 d'8 e'8")
abjad> spannertools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8)

```

```

abjad> f(voice)
\new Voice {
    c'8 [
    d'8
    e'8 ]
}

abjad> new_components = 3 * Note(0, (1, 16))
abjad> componenttools.extend_left_in_parent_of_component_and_grow_spanners(voice[0], new_componen
[Note("c'16"), Note("c'16"), Note("c'16"), Note("c'8")])

abjad> f(voice)
\new Voice {
    c'16 [
    c'16
    c'16
    c'8
    d'8
    e'8 ]
}

```

Return *new_components* and *component* together in newly created list. Changed in version 2.0: renamed `splice_left()` to `componenttools.extend_left_in_parent_of_component_and_grow_spanners()`.

componenttools.get_component_start_offset

`abjad.tools.componenttools.get_component_start_offset.get_component_start_offset(component)`

New in version 1.1. Get *component* start offset:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> componenttools.get_component_start_offset(staff[1])
Offset(1, 8)

```

Return nonnegative fraction.

componenttools.get_component_start_offset_in_seconds

`abjad.tools.componenttools.get_component_start_offset_in_seconds.get_component_start_offset_in_seconds(component)`

New in version 1.1. Get *component* start offset in seconds:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score = Score([staff])
abjad> contexttools.TempoMark(Duration(1, 4), 52)(score)
TempoMark(Duration(1, 4), 52)(Score<<1>>)
abjad> f(score) # doctest: +SKIP
\new Score <<
    \new Staff {
        \tempo 4=52
    }
>>

```

```

        c'8
        d'8
        e'8
        f'8
    }
>>

abjad> componenttools.get_component_start_offset_in_seconds(score.leaves[1])
Offset(15, 26)

```

Return nonnegative fraction.

componenttools.get_component_stop_offset

`abjad.tools.componenttools.get_component_stop_offset.get_component_stop_offset(component)`
 New in version 1.1. Get *component* stop offset:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> componenttools.get_component_stop_offset(staff[1])
Offset(1, 4)

```

Return positive fraction.

componenttools.get_component_stop_offset_in_seconds

`abjad.tools.componenttools.get_component_stop_offset_in_seconds.get_component_stop_offset_in_seconds(component)`
 New in version 1.1. Get *component* stop offset in seconds:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score = Score([staff])
abjad> contexttools.TempoMark(Duration(1, 4), 52)(score)
TempoMark(Duration(1, 4), 52)(Score<<1>>)
abjad> f(score) # doctest: +SKIP
\new Score <<
    \new Staff {
        \tempo 4=52
        c'8
        d'8
        e'8
        f'8
    }
>>

abjad> componenttools.get_component_stop_offset_in_seconds(score.leaves[1])
Offset(15, 13)

```

Return positive fraction.

componenttools.get_first_component_in_expr_with_name

`abjad.tools.componenttools.get_first_component_in_expr_with_name.get_first_component_in_expr`

New in version 1.1. Get first component in *expr* with *name*:

```
abjad> flute_staff = Staff("c'8 d'8 e'8 f'8")
abjad> flute_staff.name = 'Flute'
abjad> violin_staff = Staff("c'8 d'8 e'8 f'8")
abjad> violin_staff.name = 'Violin'
abjad> staff_group = scoretools.StaffGroup([flute_staff, violin_staff])
abjad> score = Score([staff_group])

abjad> componenttools.get_first_component_in_expr_with_name(score, 'Violin')
Staff-"Violin"{4}
```

Changed in version 2.0: Function returns first component found. Function previously returned tuple of all components found. Changed in version 2.0: renamed `scoretools.find()` to `componenttools.get_first_component_in_expr_with_name()`. Changed in version 2.0: Removed *klass* and *context* keywords. Function operates only on component name.

componenttools.get_first_component_with_name_in_improper_parentage_of_component

`abjad.tools.componenttools.get_first_component_with_name_in_improper_parentage_of_component`

New in version 2.0. Get first component with *name* in improper parentage of *component*:

```
abjad> score = Score([Staff("c'4 d'4 e'4 f'4")])
abjad> score.name = 'The Score'

abjad> f(score)
\context Score = "The Score" <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>

abjad> leaf = score.leaves[0]

abjad> componenttools.get_first_component_with_name_in_improper_parentage_of_component(leaf, 'The
Score-"The Score"<<1>>

abjad> componenttools.get_first_component_with_name_in_improper_parentage_of_component(leaf, 'fo
True
```

Return component or none.

componenttools.get_first_component_with_name_in_proper_parentage_of_component

`abjad.tools.componenttools.get_first_component_with_name_in_proper_parentage_of_component`

New in version 2.0. Get first component with *name* in proper parentage of *component*:

```

abjad> score = Score([Staff("c'4 d'4 e'4 f'4")])
abjad> score.name = 'The Score'

abjad> f(score)
\context Score = "The Score" <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>

abjad> leaf = score.leaves[0]

abjad> componenttools.get_first_component_with_name_in_proper_parentage_of_component(leaf, 'The
Score-"The Score"<<1>>

abjad> componenttools.get_first_component_with_name_in_proper_parentage_of_component(leaf, 'foo'
True

```

Return component or none.

componenttools.get_first_instance_of_klass_in_improper_parentage_of_component

```
abjad.tools.componenttools.get_first_instance_of_klass_in_improper_parentage_of_component.
```

New in version 2.0. Get first instance of *klass* in improper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> componenttools.get_first_instance_of_klass_in_improper_parentage_of_component(staff[0], N
Note("c'8")

```

Return component or none.

componenttools.get_first_instance_of_klass_in_proper_parentage_of_component

```
abjad.tools.componenttools.get_first_instance_of_klass_in_proper_parentage_of_component.get
```

New in version 1.1. Get first instance of *klass* in proper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> componenttools.get_first_instance_of_klass_in_proper_parentage_of_component(staff[0], Sta
Staff{4}

```

Return component or none. Changed in version 2.0: renamed `componenttools.get_first()` to `componenttools.get_first_instance_of_klass_in_proper_parentage_of_component()`.

componenttools.get_improper_parentage_of_component

```
abjad.tools.componenttools.get_improper_parentage_of_component.get_improper_parentage_of_c
```

New in version 1.1. Get improper parentage of *component*:

```

abjad> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> note = staff.leaves[0]

```

```
abjad> componenttools.get_improper_parentage_of_component(note)
(Note("c'8"), Tuplet(2/3, [c'8, d'8, e'8]), Staff{1})
```

Return tuple of zero or more components.

componenttools.get_likely_multiplier_of_components

abjad.tools.componenttools.get_likely_multiplier_of_components.**get_likely_multiplier_of_components**
 New in version 2.0. Get likely multiplier of *components*:

```
abjad> staff = Staff("c'8.. d'8.. e'8.. f'8..")
abjad> f(staff)
\new Staff {
    c'8..
    d'8..
    e'8..
    f'8..
}
abjad> componenttools.get_likely_multiplier_of_components(staff[:])
Duration(7, 4)
```

Return 1 when no multiplier is likely:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
abjad> componenttools.get_likely_multiplier_of_components(staff[:])
Duration(1, 1)
```

Return none when more than one multiplier is likely:

```
abjad> staff = Staff(notetools.make_notes([0, 2, 4, 5], [(3, 16), (7, 32)]))
abjad> f(staff)
\new Staff {
    c'8.
    d'8..
    e'8.
    f'8..
}
abjad> componenttools.get_likely_multiplier_of_components(staff[:]) is None
True
```

Return fraction or none.

componenttools.get_nth_component_in_expr

abjad.tools.componenttools.get_nth_component_in_expr.**get_nth_component_in_expr**(*expr*,
klass,
n=0)

New in version 1.1. Get component *n* in the *klass*s of *expr*:

```

abjad> staff = Staff([])
abjad> durations = [Duration(n, 16) for n in range(1, 5)]
abjad> notes = notetools.make_notes([0, 2, 4, 5], durations)
abjad> rests = resttools.make_rests(durations)
abjad> from abjad.tools import sequencetools
abjad> leaves = sequencetools.interlace_sequences(notes, rests)
abjad> staff.extend(leaves)

abjad> print staff.format
\new Staff {
    c'16
    r16
    d'8
    r8
    e'8.
    r8.
    f'4
    r4
}

abjad> for n in range(4):
...     componenttools.get_nth_component_in_expr(staff, Note, n)
...
Note("c'16")
Note("d'8")
Note("e'8.")
Note("f'4")

abjad> for n in range(4):
...     componenttools.get_nth_component_in_expr(staff, Rest, n)
...
Rest('r16')
Rest('r8')
Rest('r8.')
Rest('r4')

abjad> componenttools.get_nth_component_in_expr(staff, Staff)
Staff{8}

```

Read right-to-left for negative values of *n*:

```

abjad> for n in range(3, -1, -1):
...     componenttools.get_nth_component_in_expr(staff, Rest, n)
...
Rest('r4')
Rest('r8.')
Rest('r8')
Rest('r16')

```

Return component or none. Changed in version 2.0: renamed `iterate.get_nth()` to `componenttools.get_nth_component_in_expr()`.

componenttools.get_nth_namesake_from_component

`abjad.tools.componenttools.get_nth_namesake_from_component.get_nth_namesake_from_component`

New in version 2.0. For positive *n*, return namesake to the right of *component*:

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> componenttools.get_nth_namesake_from_component(t[1], 1)
Note("e'8")
```

For negative *n*, return namesake to the left of *component*:

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> componenttools.get_nth_namesake_from_component(t[1], -1)
Note("c'8")
```

Return *component* when *n* is zero:

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> componenttools.get_nth_namesake_from_component(t[1], 0)
Note("d'8")
```

Return component or none.

componenttools.get_parent_and_start_stop_indices_of_components

`abjad.tools.componenttools.get_parent_and_start_stop_indices_of_components.get_parent_and_start_stop_indices_of_components`

New in version 1.1. Get parent and start / stop indices of *components*:

```
abjad> t = Staff("c'8 d'8 e'8 f'8 g'8 a'8")
abjad> print t.format
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}

abjad> leaves = t[-2:]
abjad> leaves
[Note("g'8"), Note("a'8")]
abjad> componenttools.get_parent_and_start_stop_indices_of_components(leaves)
(Staff{6}, 4, 5)
```

Return parent / start index / stop index triple. Return parent as component or none. Return nonnegative integer start index and nonnegative index stop index. Changed in version 2.0: renamed `componenttools.get_with_indices()` to `componenttools.get_parent_and_start_stop_indices_of_components()`.

componenttools.get_proper_parentage_of_component

`abjad.tools.componenttools.get_proper_parentage_of_component.get_proper_parentage_of_component`

New in version 1.1. Get proper parentage of *component*:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> staff = Staff([tuplet])
abjad> note = staff.leaves[0]
abjad> componenttools.get_proper_parentage_of_component(note)
(FixedDurationTuplet(1/4, [c'8, d'8, e'8]), Staff{1})
```

Return tuple of zero or more components.

`componenttools.is_beamable_component`

`abjad.tools.componenttools.is_beamable_component.is_beamable_component(expr)`

New in version 1.1. True when *expr* is a beamable component. Otherwise false:

```
abjad> componenttools.is_beamable_component(Note(13, (1, 16)))
True
```

Return boolean.

`componenttools.is_orphan_component`

`abjad.tools.componenttools.is_orphan_component.is_orphan_component(component)`

New in version 1.1. True when *component* has no parent. Otherwise false:

```
abjad> note = Note("c'4")
abjad> componenttools.is_orphan_component(note)
True
```

Return boolean. Changed in version 2.0: renamed `componenttools.component_is_orphan()` to `componenttools.is_orphan_component()`.

`componenttools.is_well_formed_component`

`abjad.tools.componenttools.is_well_formed_component.is_well_formed_component(expr,`

al-
low_empty_contain

New in version 1.1. True when *component* is well formed:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> componenttools.is_well_formed_component(staff)
True
```

Otherwise false:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> staff[1].written_duration = Duration(1, 4)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
abjad> componenttools.is_well_formed_component(staff)
False
```

Beamed quarter notes are not well formed.

Return boolean.

componenttools.iterate_components_backward_in_expr

`abjad.tools.componenttools.iterate_components_backward_in_expr.iterate_components_backward_in_expr`

New in version 1.1. Iterate components backward in *expr*:

```
abjad> staff = Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_note(
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    \times 2/3 {
        f'8
        g'8
        a'8
    }
}
abjad> for x in componenttools.iterate_components_backward_in_expr(staff, Note):
...     x
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")
Note("d'8")
Note("c'8")
```

New in version 2.0: optional *start* and *stop* keyword parameters.

```
abjad> for x in componenttools.iterate_components_backward_in_expr(staff, Note, start = 0, stop = 4):
...     x
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")

abjad> for x in componenttools.iterate_components_backward_in_expr(staff, Note, start = 4):
...     x
...
Note("d'8")
Note("c'8")

abjad> for x in componenttools.iterate_components_backward_in_expr(staff, Note, start = 4, stop = 6):
...     x
...
Note("d'8")
Note("c'8")
```

This function is thread-agnostic. Changed in version 2.0: renamed `iterate.backwards()` to `componenttools.iterate_components_backward_in_expr()`.

`componenttools.iterate_components_depth_first`

`abjad.tools.componenttools.iterate_components_depth_first.iterate_components_depth_first` (ca

New in version 1.1. Iterate components depth-first from *component*.

Todo

Add usage examples.

Changed in version 2.0: renamed `iterate.depth_first()` to `componenttools.iterate_components_depth_first()`.

`componenttools.iterate_components_forward_in_expr`

`abjad.tools.componenttools.iterate_components_forward_in_expr.iterate_components_forward_in`

New in version 1.1. Iterate components forward in *expr*:

```
abjad> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'voice 1'
abjad> container[1].name = 'vocie 2'
abjad> staff = Staff(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(sta
abjad> f(staff)
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "vocie 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "vocie 2" {
      b'8
      c''8
    }
  >>
}
```



```

    }
    >>
}
abjad> for x in componenttools.iterate_components_forward_in_expr(staff, Note):
...     x
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")
Note("b'8")
Note("c''8")

```

New in version 2.0: optional *start* and *stop* keyword parameters.

```

abjad> for x in componenttools.iterate_components_forward_in_expr(staff, Note, start = 0, stop = 6):
...     x
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")

abjad> for x in componenttools.iterate_components_forward_in_expr(staff, Note, start = 4):
...     x
...
Note("g'8")
Note("a'8")
Note("b'8")
Note("c''8")

abjad> for x in componenttools.iterate_components_forward_in_expr(staff, Note, start = 4, stop = 6):
...     x
...
Note("g'8")
Note("a'8")

```

This function is thread-agnostic. Changed in version 2.0: renamed `iterate.naive()` to `componenttools.iterate_components_forward_in_expr()`. Changed in version 2.0: *klass* now defaults to `Component`.

`componenttools.iterate_namesakes_backward_from_component`

`abjad.tools.componenttools.iterate_namesakes_backward_from_component.iterate_namesakes_backward_from_component`

New in version 2.0. Iterate namesakes backward from *component*:

```

abjad> container = Container(Staff(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'staff 1'
abjad> container[1].name = 'staff 2'
abjad> score = Score([])
abjad> score.is_parallel = False
abjad> score.extend(container * 2)

```

```

abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> print score.format
\new Score {
  <<
    \context Staff = "staff 1" {
      c'8
      d'8
    }
    \context Staff = "staff 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Staff = "staff 1" {
      g'8
      a'8
    }
    \context Staff = "staff 2" {
      b'8
      c''8
    }
  >>
}

abjad> for staff in componenttools.iterate_namesakes_backward_from_component(score[-1][0]):
...     print staff.format
...
\context Staff = "staff 1" {
  g'8
  a'8
}
\context Staff = "staff 1" {
  c'8
  d'8
}

```

Return generator.

componenttools.iterate_namesakes_forward_from_component

`abjad.tools.componenttools.iterate_namesakes_forward_from_component.iterate_namesakes_forwards`

New in version 1.1. Iterate namesakes forward from *component*:

```

abjad> container = Container(Staff(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'staff 1'
abjad> container[1].name = 'staff 2'
abjad> score = Score([])
abjad> score.is_parallel = False
abjad> score.extend(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> print score.format
\new Score {
  <<

```

```

        \context Staff = "staff 1" {
            c'8
            d'8
        }
        \context Staff = "staff 2" {
            e'8
            f'8
        }
    >>
    <<
        \context Staff = "staff 1" {
            g'8
            a'8
        }
        \context Staff = "staff 2" {
            b'8
            c''8
        }
    >>
}

abjad> for staff in componenttools.iterate_namesakes_forward_from_component(score[0][0]):
...     print staff.format
...
\context Staff = "staff 1" {
    c'8
    d'8
}
\context Staff = "staff 1" {
    g'8
    a'8
}

```

Return generator.

componenttools.iterate_timeline_backward_from_component

`abjad.tools.componenttools.iterate_timeline_backward_from_component.iterate_timeline_backward`

New in version 2.0. Iterate timeline backward from *component*:

```

abjad> score = Score([])
abjad> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
abjad> score.append(Staff(notetools.make_repeated_notes(4)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
\new Score <<
    \new Staff {
        c'4
        d'4
        e'4
        f'4
    }
    \new Staff {
        g'8
        a'8
        b'8
    }

```

```

        c''8
    }
>>
abjad> for leaf in componenttools.iterate_timeline_backward_from_component(score[1][2]):
...     leaf
...
Note("b'8")
Note("c'4")
Note("a'8")
Note("g'8")

```

Yield components sorted backward by score offset stop time.

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

componenttools.iterate_timeline_backward_in_expr

abjad.tools.componenttools.iterate_timeline_backward_in_expr.**iterate_timeline_backward_in_expr**

New in version 2.0. Iterate timeline backward in *expr*:

```

abjad> score = Score([])
abjad> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
abjad> score.append(Staff(notetools.make_repeated_notes(4)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
  \new Staff {
    g'8
    a'8
    b'8
    c''8
  }
>>
abjad> for leaf in componenttools.iterate_timeline_backward_in_expr(score):
...     leaf
...
Note("f'4")
Note("e'4")
Note("d'4")
Note("c''8")
Note("b'8")
Note("c'4")
Note("a'8")
Note("g'8")

```

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

componenttools.iterate_timeline_forward_from_component

`abjad.tools.componenttools.iterate_timeline_forward_from_component.iterate_timeline_forward`

New in version 2.0. Iterate timeline forward from *component*:

```
abjad> score = Score([])
abjad> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
abjad> score.append(Staff(notetools.make_repeated_notes(4)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
  \new Staff {
    g'8
    a'8
    b'8
    c''8
  }
>>
abjad> for leaf in componenttools.iterate_timeline_forward_from_component(score[1][2]):
...     leaf
...
Note("b'8")
Note("c''8")
Note("e'4")
Note("f'4")
```

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

componenttools.iterate_timeline_forward_in_expr

`abjad.tools.componenttools.iterate_timeline_forward_in_expr.iterate_timeline_forward_in_expr`

New in version 2.0. Iterate timeline forward in *expr*:

```
abjad> score = Score([])
abjad> score.append(Staff(notetools.make_repeated_notes(4, Duration(1, 4))))
abjad> score.append(Staff(notetools.make_repeated_notes(4)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
```

```
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
  \new Staff {
    g'8
    a'8
    b'8
    c''8
  }
>>
abjad> for leaf in componenttools.iterate_timeline_forward_in_expr(score):
...     leaf
...
Note("c'4")
Note("g'8")
Note("a'8")
Note("d'4")
Note("b'8")
Note("c''8")
Note("e'4")
Note("f'4")
```

Iterate leaves when *klass* is none.

Todo

optimize to avoid behind-the-scenes full-score traversal.

`componenttools.list_badly_formed_components_in_expr`

`abjad.tools.componenttools.list_badly_formed_components_in_expr.list_badly_formed_components_in_expr`

New in version 1.1. List badly formed components in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> staff[1].written_duration = Duration(1, 4)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
abjad> f(staff)
\new Staff {
  c'8 [
  d'4
  e'8
  f'8 ]
}
abjad> componenttools.list_badly_formed_components_in_expr(staff)
[Note("d'4")]
```

Beamed quarter notes are not well formed.

Return newly created list of zero or more components.

componenttools.list_improper_contents_of_component_that_cross_prolated_offset

```
abjad.tools.componenttools.list_improper_contents_of_component_that_cross_prolated_offset.
```

New in version 2.0. List improper contents of *component* that cross *prolated_offset*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
}
```

Examples refer to the score above.

No components cross prolated offset 0:

```
abjad> componenttools.list_improper_contents_of_component_that_cross_prolated_offset(staff, 0)
[]
```

Staff, measure and leaf cross prolated offset 1/16:

```
abjad> componenttools.list_improper_contents_of_component_that_cross_prolated_offset(staff, Duration[Staff{2}, Measure(2/8, [c'8, d'8]), Note("c'8")])
```

Staff and measure cross prolated offset 1/8:

```
abjad> componenttools.list_improper_contents_of_component_that_cross_prolated_offset(staff, Duration[Staff{2}, Measure(2/8, [c'8, d'8])])
```

Staff crosses prolated offset 1/4:

```
abjad> componenttools.list_improper_contents_of_component_that_cross_prolated_offset(staff, Duration[Staff{2}])
```

No components cross prolated offset 99:

```
abjad> componenttools.list_improper_contents_of_component_that_cross_prolated_offset(staff, 99)
[]
```

Return list.

componenttools.list_leftmost_components_with_prolated_duration_at_most

```
abjad.tools.componenttools.list_leftmost_components_with_prolated_duration_at_most.list_leftmost_components_with_prolated_duration_at_most.
```

New in version 2.0. List leftmost components in *component* with prolated duration at most *prolated_duration*.

Return tuple of components[:i] together with the prolated duration of components[:i]:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> componenttools.list_leftmost_components_with_prolated_duration_at_most(voice[:], Duration(
([Note("c'8"), Note("d'8")], Duration(1, 4))
```

Maximize i such that the prolated duration of `components[:i]` is no greater than *prolated_duration*.

Input *components* must be thread-contiguous.

Todo

```
implement componenttools.list_leftmost_components_with_prolated_duration_at_least().
```

Todo

```
implement componenttools.list_rightmost_components_with_prolated_duration_at_most().
```

Todo

```
implement componenttools.list_rightmost_components_with_prolated_duration_at_least().
```

Changed in version 2.0: renamed `componenttools.get_le_duration_prolated()` to `componenttools.list_leftmost_components_with_prolated_duration_at_most()`.

`componenttools.move_component_subtree_to_right_in_immediate_parent_of_component`

`abjad.tools.componenttools.move_component_subtree_to_right_in_immediate_parent_of_component`

New in version 2.0. Move *component* subtree to right in immediate parent of *component*:

```
abjad> t = Voice("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(t[:2])
BeamSpanner(c'8, d'8)
abjad> spannertools.BeamSpanner(t[2:])
BeamSpanner(e'8, f'8)
abjad> f(t)
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> componenttools.move_component_subtree_to_right_in_immediate_parent_of_component(t[1])
abjad> f(t)
\new Voice {
  c'8 [
  e'8 ]
  d'8 [
  f'8 ]
}
```

Return `none`.

Todo

add `n = 1` keyword to generalize flipped distance.

Todo

`make` `componenttools.move_component_subtree_to_right_in_immediate_parent_of_component()`
 work when spanners attach to children of component:

```
abjad> voice = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_not
abjad> spannertools.BeamSpanner(voice.leaves[:4])
BeamSpanner(c'8, c'8, c'8, c'8)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voi
abjad> componenttools.move_component_subtree_to_right_in_immediate_parent_of_component(voice[0])
abjad> f(voice)
\new Voice {
  \times 2/3 {
    f'8 ]
    g'8
    a'8
  }
  \times 2/3 {
    c'8 [
    d'8
    e'8
  }
}
abjad> componenttools.is_well_formed_component(voice)
False
```

Preserve spanners. Changed in version 2.0: renamed `componenttools.flip()` to
`componenttools.move_component_subtree_to_right_in_immediate_parent_of_component()`.

componenttools.move_parentage_and_spanners_from_components_to_components

```
abjad.tools.componenttools.move_parentage_and_spanners_from_components_to_components.move_p
```

New in version 1.1. Move parentage and spanners from *donors* to *recipients*.

Give everything from donors to recipients. Almost exactly the same as container `setitem` logic.
 This helper works with orphan donors. Container `setitem` logic can not work with orphan
 donors. Return donors. Changed in version 2.0: renamed `scoretools.bequeath()` to
`componenttools.move_parentage_and_spanners_from_components_to_components()`.

componenttools.number_is_between_prolated_start_and_stop_offsets_of_component

```
abjad.tools.componenttools.number_is_between_prolated_start_and_stop_offsets_of_component.i
```

New in version 2.0. True when *timepoint* is within the prolated duration of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> leaf = staff.leaves[0]
```

```
abjad> componenttools.number_is_between_prolated_start_and_stop_offsets_of_component(Duration(1, 2), 0.1, 0.333, leaf)
True
abjad> componenttools.number_is_between_prolated_start_and_stop_offsets_of_component(Duration(1, 2), 0.1, 0.333, leaf)
True
```

Otherwise false:

```
abjad> componenttools.number_is_between_prolated_start_and_stop_offsets_of_component(Duration(1, 2), 0.1, 0.333, leaf)
False
```

Return boolean.

componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds

```
abjad.tools.componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds
```

New in version 2.0. True when *timepoint* is within the duration of *component* in seconds:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TempoMark(Duration(1, 2), 60, target_context = Staff)(staff)
TempoMark(Duration(1, 2), 60) (Staff{4})

abjad> leaf = staff.leaves[0]
abjad> componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds(0.1, leaf)
True
abjad> componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds(0.333, leaf)
True
```

Otherwise false:

```
abjad> componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds(0.5, leaf)
False
```

Return boolean.

componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang

```
abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_exactly
```

New in version 1.1. Partition *components* cyclically by *durations_in_seconds* exactly with overhang.

componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_without_overhang

```
abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_exactly
```

New in version 1.1. Partition *components* cyclically by *durations_in_seconds* exactly without overhang.

componenttools.partition_components_cyclically_by_durations_in_seconds_ge_with_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_ge_with`

New in version 1.1. Partition *components* cyclically by durations in seconds greater than or equal to *durations_in_seconds*, with overhang.

componenttools.partition_components_cyclically_by_durations_in_seconds_ge_without_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_ge_with`

New in version 1.1. Partition *components* cyclically by durations in seconds that are equal to or just greater than *durations_in_seconds*, without overhang.

componenttools.partition_components_cyclically_by_durations_in_seconds_le_with_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_le_with`

New in version 1.1. Partition *components* cyclically by durations in seconds equal to or just less than *durations_in_seconds*, with overhang.

componenttools.partition_components_cyclically_by_durations_in_seconds_le_without_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_le_with`

New in version 1.1. Partition *components* cyclically by durations in seconds that equal or are just less than *durations_in_seconds*, without overhang.

componenttools.partition_components_cyclically_by_prolated_durations_exactly_with_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_exactly_w`

New in version 1.1. Partition *components* cyclically by *prolated_durations* exactly, with overhang.

componenttools.partition_components_cyclically_by_prolated_durations_exactly_without_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_exactly_w`

New in version 1.1. Partition *components* cyclically by *prolated_durations* exactly, without overhang.

componenttools.partition_components_cyclically_by_prolated_durations_ge_with_overhang

abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_ge_with_ove

New in version 1.1. Partition *components* cyclically by *prolated_durations* greater than or equal, with overhang:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
    {
        \time 2/8
        b'8
        c''8
    }
}

abjad> groups = componenttools.partition_components_cyclically_by_prolated_durations_ge_with_ove

abjad> for group in groups:
...     group
...
[Note("c'8"), Note("d'8")]
[Note("e'8")]
[Note("f'8"), Note("g'8")]
[Note("a'8")]
[Note("b'8"), Note("c''8")]
```

Return list of lists.

Note: function works not just on components but on any durated objects including spanners.

componenttools.partition_components_cyclically_by_prolated_durations_ge_without_overhang

abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_ge_without

New in version 1.1. Partition *components* cyclically by *prolated_durations* that equal or are just greater than *prolated_durations*, without overhang.

componenttools.partition_components_cyclically_by_prolated_durations_le_with_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_le_with_ov`

New in version 1.1. Partition *components* cyclically by prolated duration that equal or are just less than *prolated_durations*, with overhang.

componenttools.partition_components_cyclically_by_prolated_durations_le_without_overhang

`abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_le_without`

New in version 1.1. Partition *components* cyclically by prolated durations that equal or are just less than *prolated_durations*, without overhang.

componenttools.partition_components_once_by_durations_in_seconds_exactly_with_overhang

`abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_exactly_with_c`

New in version 1.1. Partition *components* once by *durations_in_seconds* exactly, with overhang.

componenttools.partition_components_once_by_durations_in_seconds_exactly_without_overhang

`abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_exactly_withou`

New in version 1.1. Partition *components* cyclically by *durations_in_seconds* exactly, without overhang.

componenttools.partition_components_once_by_durations_in_seconds_ge_with_overhang

`abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_ge_with_overha`

New in version 1.1. Partition *components* once by durations in seconds that equal or are just greater than *durations_in_seconds*, with overhang.

componenttools.partition_components_once_by_durations_in_seconds_ge_without_overhang

`abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_ge_without_ov`

New in version 1.1. Partition *components* once by durations in seconds that equal or are just greater than *durations_in_seconds*, without overhang.

componenttools.partition_components_once_by_durations_in_seconds_le_with_overhang

```
abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_le_with_overhang
```

New in version 1.1. Partition *components* once by durations in seconds that equal or are just less than *durations_in_seconds*, with overhang.

componenttools.partition_components_once_by_durations_in_seconds_le_without_overhang

```
abjad.tools.componenttools.partition_components_once_by_durations_in_seconds_le_without_overhang
```

New in version 1.1. Partition *components* once by durations in seconds that equal or are just less than *durations_in_seconds*, without overhang.

componenttools.partition_components_once_by_prolated_durations_exactly_with_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_exactly_with_overhang
```

New in version 1.1. Partition *components* once by *prolated_durations* exactly, with overhang.

componenttools.partition_components_once_by_prolated_durations_exactly_without_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_exactly_without_overhang
```

New in version 1.1. Partition *components* once by *prolated_durations* exactly, without overhang.

componenttools.partition_components_once_by_prolated_durations_ge_with_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_ge_with_overhang
```

New in version 1.1. Partition *components* cyclically by prolated durations that equal or are just greater than *prolated_durations*, with overhang.

componenttools.partition_components_once_by_prolated_durations_ge_without_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_ge_without_overhang
```

New in version 1.1. Partition *components* cyclically by prolated durations that equal or are just greater than *prolated_durations*, without overhang.

componenttools.partition_components_once_by_prolated_durations_le_with_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_le_with_overhang
```

New in version 1.1. Partition *components* once by prolated durations that equal or are just less than *prolated_durations*, with overhang.

componenttools.partition_components_once_by_prolated_durations_le_without_overhang

```
abjad.tools.componenttools.partition_components_once_by_prolated_durations_le_without_overhang
```

New in version 1.1. Partition *components* once by prolated durations that equal or are just less than *prolated_durations*, without overhang.

componenttools.remove_component_subtree_from_score_and_spanners

```
abjad.tools.componenttools.remove_component_subtree_from_score_and_spanners.remove_component_subtree_from_score_and_spanners
```

New in version 1.1. Remove arbitrary *components* and children of *components* from score and spanners:

```
abjad> score = Voice(notetools.make_repeated_notes(2))
abjad> score.insert(1, Container(notetools.make_repeated_notes(2)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> spannertools.BeamSpanner(score.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> spannertools.GlissandoSpanner(score.leaves)
GlissandoSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> f(score)
\new Voice {
  c'8 [ \glissando
  {
    d'8 \glissando
    e'8 \glissando
  }
  f'8 ]
}
```

Examples refer to the score above.

Remove one leaf from score:

```
abjad> componenttools.remove_component_subtree_from_score_and_spanners(score.leaves[1:2]) # doctest: +SKIP
(Note(d', 8),)
```

```
abjad> f(score) # doctest: +SKIP
\new Voice {
  c'8 [ \glissando
  {
    e'8 \glissando
  }
  f'8 ]
}
```

Remove contiguous leaves from score:

```
abjad> result = componenttools.remove_component_subtree_from_score_and_spanners(score.leaves[:2]
(Note(c', 8), Note(d', 8))

abjad> f(score) # doctest: +SKIP
\new Voice {
  {
    e'8 [ \glissando
  }
  f'8 ]
}
```

Remove noncontiguous leaves from score:

```
abjad> componenttools.remove_component_subtree_from_score_and_spanners([score.leaves[0], score.l
[Note(c', 8), Note(e', 8)]

abjad> f(score) # doctest: +SKIP
\new Voice {
  {
    d'8 [ \glissando
  }
  f'8 ]
}
```

Remove container from score:

```
abjad> result = componenttools.remove_component_subtree_from_score_and_spanners(score[1:2])
abjad> result # doctest: +SKIP
[{d'8, e'8}]

abjad> f(score) # doctest: +SKIP
\new Voice {
  c'8 [ \glissando
  f'8 ]
}
```

Withdraw *components* and children of *components* from spanners.

Return either tuple or list of *components* and children of *components*.

Todo

regularize return value of function.

Note: rename to `componenttools.remove_components_from_score_deep()`.

Changed in version 2.0: renamed `componenttools.detach()` to `componenttools.remove_component_subtree_from_score_and_spanners()`.

`componenttools.replace_components_with_children_of_components`

`abjad.tools.componenttools.replace_components_with_children_of_components.replace_components`
 New in version 1.1. Remove arbitrary *components* from score but retain children of *components* in score:


```
abjad> staff = Staff(Container(notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> spannertools.SlurSpanner(staff[:])
SlurSpanner({c'8, d'8}, {e'8, f'8})
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  {
    c'8 [ (
    d'8
  }
  {
    e'8
    f'8 ] )
  }
}

abjad> componenttools.replace_components_with_children_of_components(staff[0:1])
[{}]
```

```
abjad> f(staff)
\new Staff {
  c'8 [ (
  d'8
  {
    e'8
    f'8 ] )
  }
}
```

Return *components*.

Note: should be renamed to `componenttools.remove_components_from_score_shallow()`

Changed in version 2.0: renamed `componenttools.slip()` to `componenttools.replace_components_with_children_of_components()`.

componenttools.report_component_format_contributions_as_string

```
abjad.tools.componenttools.report_component_format_contributions_as_string.report_component
```

New in version 1.1. Report *component* format contributions as string.

Set *verbose* to True or False.

componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing_spanners

```
abjad.tools.componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing
```

New in version 1.1. Split *component* at *prolated_duration* and do not fracture crossing spanners.

Leave spanners untouched.

Return split parts:

```
abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> spannertools.BeamSpanner(t[0])
BeamSpanner(|2/8(2)|)
abjad> spannertools.BeamSpanner(t[1])
BeamSpanner(|2/8(2)|)
abjad> spannertools.SlurSpanner(t.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
    {
        \time 2/8
        c'8 [ (
        d'8 ]
    }
    {
        \time 2/8
        e'8 [
        f'8 ] )
    }
}

abjad> halves = componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing
abjad> f(t)
\new Staff {
    {
        \time 2/8
        c'32 [ (
        c'16.
        d'8 ]
    }
    {
        \time 2/8
        e'8 [
        f'8 ] )
    }
}
```

Works on both leaves and containers. Changed in version 2.0: renamed `split.unfractured_at_duration()` to `componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing_spanners`

`componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners`

```
abjad.tools.componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners
```

New in version 1.1. Split *component* at *prolated_duration* and fracture crossing spanners.

Return split parts:

```
abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> spannertools.BeamSpanner(t[0])
```

```

BeamSpanner(|2/8(2)|)
abjad> spannertools.BeamSpanner(t[1])
BeamSpanner(|2/8(2)|)
abjad> spannertools.SlurSpanner(t.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
  {
    \time 2/8
    c'8 [ (
    d'8 ]
  }
  {
    \time 2/8
    e'8 [
    f'8 ] )
  }
}

halves = componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners(t.leaves)
\new Staff {
  {
    \time 2/8
    c'32 () [
    c'16. (
    d'8 ]
  }
  {
    \time 2/8
    e'8 [
    f'8 ] )
  }
}

```

Function works on both leaves and containers. Changed in version 2.0: renamed `split.fractured_at_duration()` to `componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners()`.

componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossing_spanners

`abjad.tools.componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossing_spanners()`

New in version 1.1. Partition *components* cyclically by *prolated durations* and do not fracture spanners:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> spannertools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
abjad> spannertools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
abjad> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)
\new Staff {
  {

```

```

        \time 2/8
        c'8 [ (
        d'8 ]
    }
    {
        \time 2/8
        e'8 [
        f'8 ] )
    }
}

abjad> durations = [Duration(3, 32)]
abjad> componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossings(
[[Note("c'16."), [Note("c'32"), Note("d'16")],
[Note("d'16"), Note("e'32")], [Note("e'16."), [Note("f'16."), [Note("f'32")]]]

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'16. [ (
        c'32
        d'16
        d'16 ]
    }
    {
        \time 2/8
        e'32 [
        e'16.
        f'16.
        f'32 ] )
    }
}

```

Return list of partitioned components. Changed in version 2.0:
 renamed `partition.cyclic_unfractured_by_durations()` to
`componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossings()`

componenttools.split_components_cyclically_by_prolated_durations_and_fracture_crossing_spanners

`abjad.tools.componenttools.split_components_cyclically_by_prolated_durations_and_fracture_crossing_spanners`

New in version 1.1. Partition *components* cyclically by prolated *durations* and fracture spanners:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> spannertools.BeamSpanner(staff[0])
BeamSpanner(|2/8(2)|)
abjad> spannertools.BeamSpanner(staff[1])
BeamSpanner(|2/8(2)|)
abjad> spannertools.SlurSpanner(staff.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)
\new Staff {

```

```

{
    \time 2/8
    c'8 [ (
    d'8 ]
}
{
    \time 2/8
    e'8 [
    f'8 ] )
}
}

abjad> durations = [Duration(3, 32)]
abjad> componenttools.split_components_cyclically_by_prolated_durations_and_fracture_crossing_sp
[[Note("c'16."), [Note("c'32"), Note("d'16")], [Note("d'16"), Note("e'32")],
[Note("e'16."), [Note("f'16."), [Note("f'32")]]]

abjad> f(staff)
\new Staff {
{
    \time 2/8
    c'16. ( ) [
    c'32 (
    d'16 )
    d'16 ] (
}
{
    \time 2/8
    e'32 ) [
    e'16. (
    f'16. )
    f'32 ] ( )
}
}

```

Return list of partitioned components. Changed in version 2.0: renamed `partition.cyclic_fractured_by_durations()` to `componenttools.split_components_cyclically_`

componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_crossing_spanners

`abjad.tools.componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_`

New in version 1.1. Split *components* once by prolated *durations* and do not fracture crossing spanners:

```

abjad> t = Staff(Container(notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> spannertools.BeamSpanner(t[0])
BeamSpanner({c'8, d'8})
abjad> spannertools.BeamSpanner(t[1])
BeamSpanner({e'8, f'8})
abjad> spannertools.SlurSpanner(t.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {

```

```

    {
        c'8 [ (
        d'8 ]
    }
    {
        e'8 [
        f'8 ] )
    }
}

```

```

abjad> durations = [Duration(1, 32), Duration(3, 32), Duration(5, 32)]
abjad> parts = componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_crossin

```

```

abjad> f(t)
\new Staff {
    {
        c'32 [ (
    }
    {
        c'16.
    }
    {
        d'8 ]
    }
    {
        e'8 [
        f'8 ] )
    }
}

```

Changed in version 2.0: renamed `partition.unfractured_by_durations()` to `componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_crossin`

`componenttools.split_components_once_by_prolated_durations_and_fracture_crossing_spanners`

```

abjad.tools.componenttools.split_components_once_by_prolated_durations_and_fracture_crossin

```

New in version 1.1. Split *components* once by prolated *durations* and fracture crossing spanners:

```

abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> spannertools.BeamSpanner(t[0])
BeamSpanner(|2/8(2)|)
abjad> spannertools.BeamSpanner(t[1])
BeamSpanner(|2/8(2)|)
abjad> spannertools.SlurSpanner(t.leaves)
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
    {
        \time 2/8
        c'8 [ (
        d'8 ]
    }
}

```

```

    {
        \time 2/8
        e'8 [
        f'8 ] )
    }
}

abjad> durations = [Duration(1, 32), Duration(3, 32), Duration(5, 32)]
abjad> parts = componenttools.split_components_once_by_prolated_durations_and_fracture_crossing_
abjad> f(t)
\new Staff {
    {
        \time 1/32
        c'32 [ ] ( )
    }
    {
        \time 3/32
        c'16. [ ] ( )
    }
    {
        \time 4/32
        d'8 [ ] (
    }
    {
        \time 2/8
        e'8 [
        f'8 ] )
    }
}

```

Changed in version 2.0: renamed `partition.fractured_by_durations()` to `componenttools.split_components_once_by_prolated_durations_and_fracture_crossing_spann`

componenttools.sum_duration_of_components_in_seconds

`abjad.tools.componenttools.sum_duration_of_components_in_seconds.sum_duration_of_components`

New in version 1.1. Sum duration of *components* in seconds:

```

abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> score = Score([Staff([tuplet])])
abjad> contexttools.TempoMark(Duration(1, 4), 48)(score)
TempoMark(Duration(1, 4), 48)(Score<<1>>)
abjad> f(score) # doctest: +SKIP
\new Score <<
    \new Staff {
        \times 2/3 {
            \tempo 4=48
            c'8
            d'8
            e'8
        }
    }
>>

abjad> componenttools.sum_duration_of_components_in_seconds(tuplet[:])
Duration(5, 4)

```

Changed in version 2.0: renamed `durationtools.sum_seconds()` to `componenttools.sum_duration_of_components_in_seconds()`.

`componenttools.sum_preprolated_duration_of_components`

`abjad.tools.componenttools.sum_preprolated_duration_of_components.sum_preprolated_duration_of_components`

New in version 1.1. Sum preprolated duration of *components*:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> componenttools.sum_preprolated_duration_of_components(tuplet[:])
Duration(3, 8)
```

Return zero on empty iterable:

```
abjad> componenttools.sum_preprolated_duration_of_components([])
0
```

Raise contiguity error on nonparent-contiguous *components*:

```
abjad> t = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_notes(3,
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> f(t)
\new Voice {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  \times 2/3 {
    f'8
    g'8
    a'8
  }
}
abjad> componenttools.sum_preprolated_duration_of_components(t.leaves)
Duration(3, 4)
```

Changed in version 2.0: renamed `componenttools.get_duration_preprolated()` to `componenttools.sum_preprolated_duration_of_components()`.

`componenttools.sum_prolated_duration_of_components`

`abjad.tools.componenttools.sum_prolated_duration_of_components.sum_prolated_duration_of_components`

New in version 1.1. Sum prolated duration of *components*:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}
abjad> componenttools.sum_prolated_duration_of_components(tuplet[:])
Duration(1, 4)
```

Changed in version 2.0: renamed `durationtools.sum_prolated()` to `componenttools.sum_prolated_duration_of_components()`.

componenttools.tabulate_well_formedness_violations_in_expr

`abjad.tools.componenttools.tabulate_well_formedness_violations_in_expr.tabulate_well_formedness_violations_in_expr`

New in version 1.1. Tabulate well-formedness violations in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> staff[1].written_duration = Duration(1, 4)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'4, e'8, f'8)
abjad> f(staff)
\new Staff {
    c'8 [
    d'4
    e'8
    f'8 ]
}

abjad> componenttools.tabulate_well_formedness_violations_in_expr(staff)
1 /    4 beamed quarter note
0 /    1 discontinuous spanner
0 /    5 duplicate id
0 /    1 empty container
0 /    0 intermarked hairpin
0 /    0 misdurated measure
0 /    0 misfilled measure
0 /    4 mispitched tie
0 /    4 misrepresented flag
0 /    5 missing parent
0 /    0 nested measure
0 /    0 overlapping beam
0 /    0 overlapping glissando
0 /    0 overlapping octavation
0 /    0 short hairpin
```

Beamed quarter notes are not well formed.

componenttools.yield_components_grouped_by_preprolated_duration

`abjad.tools.componenttools.yield_components_grouped_by_preprolated_duration.yield_components_grouped_by_preprolated_duration`

New in version 2.0. Yield components grouped by preprolated duration:

```
abjad> notes = notetools.make_notes([0], [(1, 4), (1, 4), (1, 8), (1, 16), (1, 16), (1, 16)])
abjad> for x in componenttools.yield_components_grouped_by_preprolated_duration(notes):
...     x
...
(Note("c'4"), Note("c'4"))
(Note("c'8"),)
(Note("c'16"), Note("c'16"), Note("c'16"))
```

Return generator.

componenttools.yield_components_grouped_by_prolated_duration

`abjad.tools.componenttools.yield_components_grouped_by_prolated_duration.yield_components_grouped_by_prolated_duration`

New in version 2.0. Yield *component* grouped by prolated duration:

```
abjad> notes = notetools.make_notes([0], [(1, 4), (1, 4), (1, 8), (1, 16), (1, 16), (1, 16)])
abjad> for x in componenttools.yield_components_grouped_by_prolated_duration(notes):
...     x
...
(Note("c'4"), Note("c'4"))
(Note("c'8"),)
(Note("c'16"), Note("c'16"), Note("c'16"))
```

Return generator.

componenttools.yield_groups_of_mixed_klasses_in_sequence

`abjad.tools.componenttools.yield_groups_of_mixed_klasses_in_sequence.yield_groups_of_mixed_klasses_in_sequence`

New in version 2.0. Yield groups of mixed *klasses* in *sequence*:

```
abjad> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}
```

```
abjad> for group in componenttools.yield_groups_of_mixed_klasses_in_sequence(staff, (Note, Chord)):
...     group
(Note("c'8"), Note("d'8"))
(Chord("<e' g'>8"), Chord("<f' a'>8"), Note("g'8"), Note("a'8"))
(Chord("<b' d''>8"), Chord("<c'' e''>8"))
```

Return generator.

componenttools.yield_topmost_components_grouped_by_type

`abjad.tools.componenttools.yield_topmost_components_grouped_by_type.yield_topmost_components_grouped_by_type`

New in version 2.0. Yield topmost components in *expr* grouped by type:

```
abjad> staff = Staff(leaftools.make_leaves([0, 2, 4, None, None, 5, 7], [(1, 8)]))
abjad> for x in componenttools.yield_topmost_components_grouped_by_type(staff):
...     x
...
(Note("c'8"), Note("d'8"), Note("e'8"))
(Rest('r8'), Rest('r8'))
(Note("f'8"), Note("g'8"))
```

Return generator.

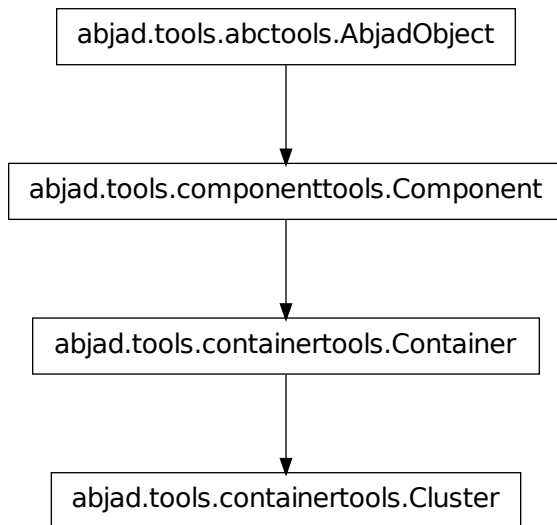
componenttools.yield_topmost_components_of_klass_grouped_by_type

`abjad.tools.componenttools.yield_topmost_components_of_klass_grouped_by_type.yield_topmost`

New in version 2.0. Yield topmost components of *klass* in *expr* grouped by type:

```
abjad> staff = Staff(leaftools.make_leaves([0, 2, 4, None, None, 5, 7], [(1, 8)]))
abjad> for x in componenttools.yield_topmost_components_of_klass_grouped_by_type(staff, Note):
...     x
...
(Note("c'8"), Note("d'8"), Note("e'8"))
(Note("f'8"), Note("g'8"))
```

Return generator.

containertools**containertools concrete classes****containertools.Cluster**

class `abjad.tools.containertools.Cluster.Cluster.Cluster` (*music=None, **kwargs*)

New in version 1.1. Abjad model of a tone cluster container:

```
abjad> cluster = containertools.Cluster("c'8 d'8 b'8")

abjad> cluster
Cluster(c'8, d'8, b'8)
```

```
abjad> f(cluster)
\makeClusters {
  c'8
  d'8
  b'8
}
```

Return cluster object.

Read-only Properties

`Cluster.contents_duration`

Note: Inherited from `containertools.Container`

`Cluster.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Cluster.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`Cluster.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

`Cluster.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`Cluster.music`

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

`Cluster.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`Cluster.parent`

Note: Inherited from `componenttools.Component`

`Cluster.preprolated_duration`

Note: Inherited from `containertools.Container`

`Cluster.prolated_duration`

Note: Inherited from `componenttools.Component`

`Cluster.prolation`

Note: Inherited from `componenttools.Component`

`Cluster.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`Cluster.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`Cluster.is_parallel`

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
```

```
    }
    \new Voice {
      g4.
    }
  }

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
      c'8
      d'8
      e'8
    }
    \new Voice {
      g4.
    }
  >>
```

Return none.

Note: Inherited from `containertools.Container`

Methods

`Cluster.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
  c'8 [
  d'8
  e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
  c'8 [
  d'8
  e'8 ]
  f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

Cluster.**extend** (*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Cluster.**index** (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Cluster.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
```

```

        e'8 ]
    }

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Cluster.**pop** (*i=-1*)

Pop component at index *i* from container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

Cluster.**remove** (*component*)

Remove *component* from container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```



```
abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`Cluster.__add__(expr)`

Concatenate containers self and *expr*. The operation $c = a + b$ returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`Cluster.__contains__(expr)`

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Cluster.__delattr__()`

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`Cluster.__delitem__(i)`

Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Cluster.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Cluster.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Cluster.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Cluster.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Cluster.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Cluster.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Cluster.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Cluster.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Cluster.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Cluster.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Cluster.__mul__(n)

Note: Inherited from `componenttools.Component`

Cluster.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Cluster.__radd__(expr)

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

Cluster.__repr__()

Cluster.__rmul__(n)

Note: Inherited from `componenttools.Component`

Cluster.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Cluster.__setitem__(i, expr)

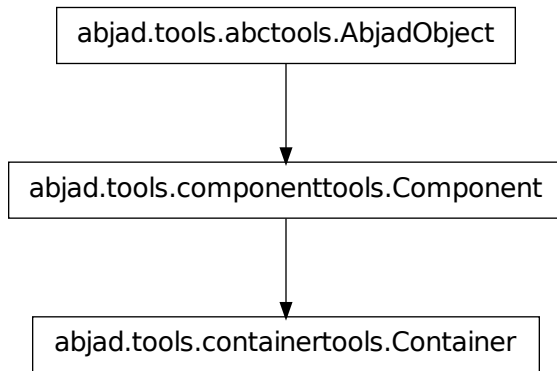
Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

Cluster.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

containertools.Container



class `abjad.tools.containertools.Container.Container`, **Container** (*music=None*, ***kwargs*)

Abjad model of a music container:

```

abjad> container = Container("c'8 d'8 e'8 f'8")
abjad> f(container)
{
    c'8
    d'8
    e'8
    f'8
}
  
```

Return container object.

Read-only Properties

`Container.contents_duration`

`Container.duration_in_seconds`

`Container.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`Container.leaves`

Read-only tuple of leaves in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
  
```

Return tuple of zero or more leaves.

Container.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Container.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Container.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Container.parent

Note: Inherited from `componenttools.Component`

Container.preprolated_duration**Container.prolated_duration**

Note: Inherited from `componenttools.Component`

Container.prolation

Note: Inherited from `componenttools.Component`

Container.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Container.spanners

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties**Container.is_parallel**

Get parallel container:

```

abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')]

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False

```

Return boolean.

Set parallel container:

```

abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>

```

Return none.

Methods

`Container.append(component)`

Append *component* to container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

```

```

    f'8
}

```

Return none.

`Container.extend(expr)`

Extend *expr* against container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

`Container.index(component)`

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2

```

Return nonnegative integer.

`Container.insert(i, component)`

Insert *component* in container at index *i*:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

```

```
abjad> container.insert(1, Note("cs'8"))
```

```
abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Container.pop (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.pop(-1)
Note("e'8")
```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Container.remove (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.remove(note)
```

```
abjad> f(container)
{
    c'8 [
```



```
        d' 8 ]
    }
```

Return none.

Special Methods

`Container.__add__(expr)`

Concatenate containers self and *expr*. The operation $c = a + b$ returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

`Container.__contains__(expr)`

True if *expr* is in container, otherwise False.

`Container.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Container.__delitem__(i)`

Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

`Container.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Container.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Container.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

`Container.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Container.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Container.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Container.__**imul**__(total)

Multiply contents of container ‘total’ times. Return multiplied container.

Container.__**le**__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Container.__**len**__()

Return nonnegative integer number of components in container.

Container.__**lt**__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Container.__**mul**__(n)

Note: Inherited from `componenttools.Component`

Container.__**ne**__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Container.__**radd**__(expr)

Extend container by contents of `expr` to the right.

Container.__**repr**__()

String format of container for interpreter display.

Container.__**rmul**__(n)

Note: Inherited from `componenttools.Component`

Container.__**setattr**__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Container.__**setitem**__(i, expr)

Set ‘`expr`’ in self at nonnegative integer index `i`. Or, set ‘`expr`’ in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with ‘`expr`’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

`Container.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

containertools functions

containertools.all_are_containers

`abjad.tools.containertools.all_are_containers.expr`

New in version 2.6. True when *expr* is a sequence of Abjad containers:

```
abjad> containers = 3 * Container("c'8 d'8 e'8")
```

```
abjad> containertools.all_are_containers(containers)
True
```

True when *expr* is an empty sequence:

```
abjad> containertools.all_are_containers([])
True
```

Otherwise false:

```
abjad> containertools.all_are_containers('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

containertools.color_contents_of_container

`abjad.tools.containertools.color_contents_of_container.container, color`

New in version 2.0. Color contents of *container*:

```
abjad> measure = Measure((2, 8), "c'8 d'8")
```

```
abjad> containertools.color_contents_of_container(measure, 'red')
Measure(2/8, [c'8, d'8])
```

```
abjad> f(measure)
{
  \override Accidental #'color = #red
  \override Beam #'color = #red
  \override Dots #'color = #red
  \override NoteHead #'color = #red
  \override Rest #'color = #red
  \override Stem #'color = #red
  \override TupletBracket #'color = #red
  \override TupletNumber #'color = #red
  \time 2/8
  c'8
  d'8
  \revert Accidental #'color
```

```

\revert Beam #'color
\revert Dots #'color
\revert NoteHead #'color
\revert Rest #'color
\revert Stem #'color
\revert TupletBracket #'color
\revert TupletNumber #'color
}

```

Return *none*. Changed in version 2.0: renamed `containertools.contents_color()` to `containertools.color_contents_of_container()`.

`containertools.delete_contents_of_container`

`abjad.tools.containertools.delete_contents_of_container.delete_contents_of_container(container)`

Delete contents of *container*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

abjad> containertools.delete_contents_of_container(staff)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]

abjad> f(staff)
\new Staff {
}

```

Return *container* contents. Changed in version 2.0: renamed `containertools.contents_delete()` to `containertools.delete_contents_of_container()`.

`containertools.delete_contents_of_container_starting_at_or_after_prolated_offset`

`abjad.tools.containertools.delete_contents_of_container_starting_at_or_after_prolated_offset(container, prolated_offset)`

New in version 2.0. Delete contents of *container* starting at or after *prolated_offset*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8

```

```

        f'8 ]
    }

```

```

abjad> containertools.delete_contents_of_container_starting_at_or_after_prolated_offset(staff, D
Staff{1}

```

```

abjad> f(staff)
\new Staff {
    c'8 [ ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_not_before_prolated_offset` to `containertools.delete_contents_of_container_starting_at_or_after_prolated_offset()`.

`containertools.delete_contents_of_container_starting_before_or_at_prolated_offset`

```

abjad.tools.containertools.delete_contents_of_container_starting_before_or_at_prolated_offset

```

New in version 2.0. Delete contents of *container* starting before or at *prolated_offset*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

```

```

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

```

abjad> containertools.delete_contents_of_container_starting_before_or_at_prolated_offset(staff,
Staff{2})

```

```

abjad> f(staff)
\new Staff {
    e'8 [
    f'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_not_after_prolated_offset` to `containertools.delete_contents_of_container_starting_before_or_at_prolated_offset()`.

`containertools.delete_contents_of_container_starting_strictly_after_prolated_offset`

```

abjad.tools.containertools.delete_contents_of_container_starting_strictly_after_prolated_offset

```

New in version 2.0. Delete contents of *container* starting strictly after *prolated_offset*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

```

```

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

abjad> containertools.delete_contents_of_container_starting_strictly_after_prolated_offset(staff,
Staff{2})

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_after_prolated_offset` to `containertools.delete_contents_of_container_starting_strictly_after_prolated_offset()`

`containertools.delete_contents_of_container_starting_strictly_before_prolated_offset`

`abjad.tools.containertools.delete_contents_of_container_starting_strictly_before_prolated_offset`

New in version 2.0. Delete contents of *container* contents starting strictly before *prolated_offset*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

abjad> containertools.delete_contents_of_container_starting_strictly_before_prolated_offset(staff,
Staff{3})

abjad> f(staff)
\new Staff {
    d'8 [
    e'8
    f'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_delete_starting_before_prolated_offset` to `containertools.delete_contents_of_container_starting_strictly_before_prolated_offset`

`containertools.eject_contents_of_container`

`abjad.tools.containertools.eject_contents_of_container.eject_contents_of_container` (*container*)
 New in version 2.0. Eject contents of *container*:

```

abjad> container = Container("c'8 d'8 e'8 f'8")

abjad> f(container)
{
    c'8
    d'8
    e'8
    f'8
}

abjad> containertools.eject_contents_of_container(container)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]

abjad> container
{}

abjad> f(container)
{
}

```

Return list of *container* contents.

containertools.fuse_like_named_contiguous_containers_in_expr

abjad.tools.containertools.fuse_like_named_contiguous_containers_in_expr.fuse_like_named_c

Fuse like-named contiguous containers in *expr*:

```

abjad> staff = Staff(Voice("c'8 c'8") * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> staff[0].name = 'soprano'
abjad> staff[1].name = 'soprano'

abjad> f(staff)
\new Staff {
    \context Voice = "soprano" {
        c'8
        d'8
    }
    \context Voice = "soprano" {
        e'8
        f'8
    }
}

abjad> containertools.fuse_like_named_contiguous_containers_in_expr(staff)
Staff{1}

abjad> f(staff)
\new Staff {
    \context Voice = "soprano" {
        c'8
        d'8
        e'8
        f'8
    }
}

```

Return *expr*. Changed in version 2.0: renamed `fuse.containers_by_reference()` to `containertools.fuse_like_named_contiguous_containers_in_expr()`.

`containertools.get_element_starting_at_exactly_prolated_offset`

`abjad.tools.containertools.get_element_starting_at_exactly_prolated_offset.get_element_starting_at_exactly_prolated_offset`

New in version 2.0. Get *container* element starting at exactly *prolated_offset*:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
```

```
abjad> containertools.get_element_starting_at_exactly_prolated_offset(voice, Duration(6, 8))
Note("b'8")
```

Raise missing component error when no *container* element starts at exactly *prolated_offset*. Changed in version 2.0: renamed `containertools.get_element_starting_at_prolated_offset()` to `containertools.get_element_starting_at_exactly_prolated_offset()`.

`containertools.get_first_container_in_improper_parentage_of_component`

`abjad.tools.containertools.get_first_container_in_improper_parentage_of_component.get_first_container_in_improper_parentage_of_component`

New in version 2.0. Get first container in improper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

```
abjad> containertools.get_first_container_in_improper_parentage_of_component(staff[1])
Staff{4}
```

Return container or none.

`containertools.get_first_container_in_proper_parentage_of_component`

`abjad.tools.containertools.get_first_container_in_proper_parentage_of_component.get_first_container_in_proper_parentage_of_component`

New in version 2.0. Get first container in proper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```



```
abjad> containertools.get_first_container_in_proper_parentage_of_component(staff[1])
Staff{4}
```

Return container or none.

containertools.get_first_element_starting_at_or_after_prolated_offset

```
abjad.tools.containertools.get_first_element_starting_at_or_after_prolated_offset.get_first
```

New in version 2.0. Get first *container* element starting at or after *prolated_offset*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> containertools.get_first_element_starting_at_or_after_prolated_offset(staff, Duration(1,
Note("d'8"))
```

Return component.

Return none when no *container* element starts at or after *prolated_offset*. Changed in version 2.0: renamed `containertools.get_leftmost_element_starting_not_before_prolated_offset()` to `containertools.get_first_element_starting_at_or_after_prolated_offset()`.

containertools.get_first_element_starting_before_or_at_prolated_offset

```
abjad.tools.containertools.get_first_element_starting_before_or_at_prolated_offset.get_firs
```

New in version 2.0. Get first *container* element starting before or at *prolated_offset*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> containertools.get_first_element_starting_before_or_at_prolated_offset(staff, Duration(1,
Note("d'8"))
```

Return component.

Return none when no *container* element starts before or at *prolated_offset*. Changed in version 2.0: renamed `containertools.get_rightmost_element_starting_not_after_prolated_offset()` to `containertools.get_first_element_starting_before_or_at_prolated_offset()`.

containertools.get_first_element_starting_strictly_after_prolated_offset

```
abjad.tools.containertools.get_first_element_starting_strictly_after_prolated_offset.get_f
```

New in version 2.0. Get first *container* element starting strictly after *prolated_offset*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> containertools.get_first_element_starting_strictly_after_prolated_offset(staff, Duration(
Note("e'8"))
```

Return component.

Return none when no *container* element starts strictly after *prolated_offset*. Changed in version 2.0: renamed `containertools.get_leftmost_element_starting_after_prolated_offset()` to `containertools.get_first_element_starting_strictly_after_prolated_offset()`.

`containertools.get_first_element_starting_strictly_before_prolated_offset`

`abjad.tools.containertools.get_first_element_starting_strictly_before_prolated_offset.get_`

New in version 2.0. Get first *container* element starting strictly before *prolated_offset*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> containertools.get_first_element_starting_strictly_before_prolated_offset(staff, Duration(
Note("c'8")
```

Return component.

Return none when *container* element starts strictly before *prolated_offset*. Changed in version 2.0: renamed `containertools.get_rightmost_element_starting_before_prolated_offset()` to `containertools.get_first_element_starting_strictly_before_prolated_offset()`.

`containertools.insert_component_and_do_not_fracture_crossing_spanners`

`abjad.tools.containertools.insert_component_and_do_not_fracture_crossing_spanners.insert_c`

New in version 2.0. Insert *component* into *container* at index *i* and do not fracture crossing spanners:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

abjad> containertools.insert_component_and_do_not_fracture_crossing_spanners(staff, 1, Note("cs'8"))
Staff{5}

abjad> f(staff)
\new Staff {
  c'8 [
  cs'8
  d'8
  e'8
  f'8 ]
}
```

Return *container*. Changed in version 2.0: renamed `containertools.insert_and_do_not_fracture()` to `containertools.insert_component_and_do_not_fracture_crossing_spanners()`.

containertools.insert_component_and_fracture_crossing_spanners

`abjad.tools.containertools.insert_component_and_fracture_crossing_spanners.insert_component`

Insert *component* into *container* at index *i* and fracture spanners:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

abjad> containertools.insert_component_and_fracture_crossing_spanners(staff, 1, Rest((1, 8)))
[(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8)), (BeamSpanner(d'8, e'8, f'8), BeamSpanner(e'8, f'8, g'8))]
```

```
abjad> f(staff)
\new Staff {
    c'8 [ ]
    r8
    d'8 [
    e'8
    f'8 ]
}
```

Return list of fractured spanners. Changed in version 2.0: renamed `containertools.insert_and_fracture()` to `containertools.insert_component_and_fracture_crossing_spanners()`

containertools.iterate_containers_backward_in_expr

`abjad.tools.containertools.iterate_containers_backward_in_expr.iterate_containers_backward_in_expr`

New in version 2.0. Iterate containers backward in *expr*:

```
abjad> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
abjad> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
abjad> staff.is_parallel = True

abjad> f(staff)
\new Staff <<
    \new Voice {
        c'8
        d'8
    }
    \new Voice {
        \times 2/3 {
            e'8
            f'8
            g'8
        }
    }
}
```

```

        f'8
        g'8
    }
}
>>

```

```

abjad> for x in containertools.iterate_containers_backward_in_expr(staff):
...     x
Staff<<2>>
Voice{1}
Tuplet(2/3, [e'8, f'8, g'8])
Voice{2}

```

Ignore threads.

Return generator.

`containertools.iterate_containers_forward_in_expr`

`abjad.tools.containertools.iterate_containers_forward_in_expr.iterate_containers_forward_in_expr`

New in version 2.0. Iterate containers forward in *expr*:

```

abjad> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
abjad> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
abjad> staff.is_parallel = True

```

```

abjad> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
  }
  \new Voice {
    \times 2/3 {
      e'8
      f'8
      g'8
    }
  }
>>

```

```

abjad> for x in containertools.iterate_containers_forward_in_expr(staff):
...     x
Staff<<2>>
Voice{2}
Voice{1}
Tuplet(2/3, [e'8, f'8, g'8])

```

Ignore threads.

Return generator.

containertools.move_parentage_children_and_spanners_from_components_to_empty_container

```
abjad.tools.containertools.move_parentage_children_and_spanners_from_components_to_empty_container
```

Move parentage, children and spanners from *components* to empty *container*:

```
abjad> voice = Voice(Container("c'8 c'8") * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> spannertools.BeamSpanner(voice.leaves)
BeamSpanner(c'8, d'8, e'8, f'8, g'8, a'8)

abjad> f(voice)
\new Voice {
  {
    c'8 [
    d'8
  ]
  {
    e'8
    f'8
  ]
  {
    g'8
    a'8 ]
  }
}

abjad> tuplet = Tuplet(Fraction(3, 4), [])
abjad> containertools.move_parentage_children_and_spanners_from_components_to_empty_container(voice)

abjad> f(voice)
\new Voice {
  \fraction \times 3/4 {
    c'8 [
    d'8
    e'8
    f'8
  ]
  {
    g'8
    a'8 ]
  }
}
```

Return `none`. Changed in version 2.0: renamed `scoretools.donate()` to `containertools.move_parentage_children_and_spanners_from_components_to_empty_container`

containertools.remove_empty_containers_in_expr

```
abjad.tools.containertools.remove_empty_containers_in_expr.remove_empty_containers_in_expr
```

Remove empty containers in *expr*:

```
abjad> staff = Staff(Container(notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
```

```

abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner({c'8, d'8}, {e'8, f'8}, {g'8, a'8}, {b'8, c''8})
abjad> containertools.delete_contents_of_container(staff[1])
[Note("e'8"), Note("f'8")]
abjad> containertools.delete_contents_of_container(staff[-1])
[Note("b'8"), Note("c''8")]

abjad> f(staff)
\new Staff {
    {
        c'8 [
        d'8
    ]
    {
    }
    {
        g'8
        a'8 ]
    }
    {
    }
}

abjad> containertools.remove_empty_containers_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        c'8 [
        d'8
    ]
    {
        g'8
        a'8 ]
    }
}

```

Return none. Changed in version 2.0: renamed `containertools.remove_empty()` to `containertools.remove_empty_containers_in_expr()`.

containertools.repeat_contents_of_container

`abjad.tools.containertools.repeat_contents_of_container.repeat_contents_of_container(container, to-tal=2)`

New in version 1.1. Repeat contents of *container*:

```

abjad> staff = Staff("c'8 d'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

```

```
abjad> containertools.repeat_contents_of_container(staff, 3)
Staff{6}
```

```
abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
    c'8 [
    d'8 ]
    c'8 [
    d'8 ]
}
```

Leave *container* unchanged when *total* is 1.

Empty *container* when *total* is 0.

Return *container*. Changed in version 2.0: renamed `containertools.contents_multiply()` to `containertools.repeat_contents_of_container()`.

`containertools.repeat_last_n_elements_of_container`

```
abjad.tools.containertools.repeat_last_n_elements_of_container.repeat_last_n_elements_of_c
```

New in version 1.1. Repeat last *n* elements of *container*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
abjad> containertools.repeat_last_n_elements_of_container(staff, n = 2, total = 3)
Staff{8}
```

```
abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
    e'8 [
    f'8 ]
    e'8 [
    f'8 ]
}
```

Return *container*. Changed in version 2.0: renamed `containertools.extend_cyclic()` to `containertools.repeat_last_n_elements_of_container()`.

containertools.replace_contents_of_target_container_with_contents_of_source_container

abjad.tools.containertools.replace_contents_of_target_container_with_contents_of_source_container

New in version 2.0. Replace contents of *target_container* with contents of *source_container*:

```
abjad> staff = Staff(Tuplet(Fraction(2, 3), "c'8 d'8 e'8") * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, ... [5] ..., c''8, d''8)
```

```
abjad> f(staff)
```

```
\new Staff {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  \times 2/3 {
    f'8
    g'8
    a'8
  }
  \times 2/3 {
    b'8
    c''8
    d''8 ]
  }
}
```

```
abjad> container = Container("c'8 d'8 e'8")
abjad> spannertools.SlurSpanner(container.leaves)
SlurSpanner(c'8, d'8, e'8)
```

```
abjad> f(container)
```

```
{
  c'8 (
    d'8
    e'8 )
}
```

```
abjad> containertools.replace_contents_of_target_container_with_contents_of_source_container(staff, container)
Tuplet(2/3, [c'8, d'8, e'8])
```

```
abjad> f(staff)
```

```
\new Staff {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  \times 2/3 {
    c'8 (
      d'8
      e'8 )
  }
  \times 2/3 {
    b'8
  }
}
```



```

        c''8
        d''8 ]
    }
}

```

Leave *source_container* empty:

```

abjad> container
{}

```

Return *target_container*.

containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests

`abjad.tools.containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests`
 New in version 2.0. Replace larger left half of elements in *container* with big-endian rests:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")

```

```

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
    d''8
    e''8
}

```

```

abjad> containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests(staff)
Staff{7}

```

```

abjad> f(staff)
\new Staff {
    r2
    r8
    a'8
    b'8
    c''8
    d''8
    e''8
}

```

Return *container*.

containertools.replace_larger_left_half_of_elements_in_container_with_little_endian_rests

`abjad.tools.containertools.replace_larger_left_half_of_elements_in_container_with_little_endian_rests`
 New in version 2.0. Replace larger left half of elements in *container* with little-endian rests:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")

```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
    d''8
    e''8
}
```

```
abjad> containertools.replace_larger_left_half_of_elements_in_container_with_little_endian_rests(
Staff{7})
```

```
abjad> f(staff)
\new Staff {
    r8
    r2
    a'8
    b'8
    c''8
    d''8
    e''8
}
```

Return *container*.

containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests

`abjad.tools.containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests(container)`

New in version 2.0. Replace larger right half of elements in *container* with big-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
    d''8
    e''8
}
```

```
abjad> containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests(
Staff{7})
```

```
abjad> f(staff)
\new Staff {
    c'8
```

```

        d'8
        e'8
        f'8
        g'8
        r2
        r8
    }

```

Return *container*.

containertools.replace_larger_right_half_of_elements_in_container_with_little_endian_rests

`abjad.tools.containertools.replace_larger_right_half_of_elements_in_container_with_little_endian_rests`
 New in version 2.0. Replace larger right half of elements in *container* with little-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
    d''8
    e''8
}

```

```

abjad> containertools.replace_larger_right_half_of_elements_in_container_with_little_endian_rests
Staff{7}

```

```

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    r8
    r2
}

```

Return *container*.

containertools.replace_n_edge_elements_in_container_with_big_endian_rests

`abjad.tools.containertools.replace_n_edge_elements_in_container_with_big_endian_rests`. **repl**

New in version 2.0. Replace *n* edge elements in *container* with big-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8")
```

```

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}

abjad> containertools.replace_n_edge_elements_in_container_with_big_endian_rests(staff, -5)
Staff{3}

abjad> f(staff)
\new Staff {
    c'8
    r2
    r8
}

```

Return *container*. Changed in version 2.0: renamed `containertools.replace_first_n_elements_in_container` to `containertools.replace_n_edge_elements_in_container_with_big_endian_rests()`.

`containertools.replace_n_edge_elements_in_container_with_little_endian_rests`

`abjad.tools.containertools.replace_n_edge_elements_in_container_with_little_endian_rests`. **new**

New in version 2.0. Replace *n* edge elements in *container* with little-endian rests:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}

abjad> containertools.replace_n_edge_elements_in_container_with_little_endian_rests(staff, -5)
Staff{3}

abjad> f(staff)
\new Staff {
    c'8
    r8
    r2
}

```

Return *container*. Changed in version 2.0: renamed `containertools.replace_first_n_elements_in_container` to `containertools.replace_n_edge_elements_in_container_with_little_endian_rests()`.

containertools.replace_n_edge_elements_in_container_with_rests

`abjad.tools.containertools.replace_n_edge_elements_in_container_with_rests.replace_n_edge_e`

New in version 2.0. Replace first n elements in *container* with big-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}
```

```
abjad> containertools.replace_n_edge_elements_in_container_with_rests(staff, 5)
Staff{3}
```

```
abjad> f(staff)
\new Staff {
    r2
    r8
    a'8
}
```

Replace last n elements in *container* with little-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
}
```

```
abjad> containertools.replace_n_edge_elements_in_container_with_rests(staff, -5)
Staff{3}
```

```
abjad> f(staff)
\new Staff {
    c'8
    r8
    r2
}
```

Return *container*. Changed in version 2.0: renamed `containertools.replace_first_n_elements_in_container` to `containertools.replace_n_edge_elements_in_container_with_rests()`.

containertools.replace_smaller_left_half_of_elements_in_container_with_big_endian_rests

`abjad.tools.containertools.replace_smaller_left_half_of_elements_in_container_with_big_endian_rests`

New in version 2.0. Replace smaller left half of elements in *container* with big-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```
abjad> f(staff)
```

```
\new Staff {  
    c'8  
    d'8  
    e'8  
    f'8  
    g'8  
    a'8  
    b'8  
    c''8  
    d''8  
    e''8  
}
```

```
abjad> containertools.replace_smaller_left_half_of_elements_in_container_with_big_endian_rests(staff)  
Staff{7}
```

```
abjad> f(staff)
```

```
\new Staff {  
    r2  
    r8  
    a'8  
    b'8  
    c''8  
    d''8  
    e''8  
}
```

Return *container*.

containertools.replace_smaller_left_half_of_elements_in_container_with_little_endian_rests

`abjad.tools.containertools.replace_smaller_left_half_of_elements_in_container_with_little_endian_rests`

New in version 2.0. Replace smaller left half of elements in *container* with little-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```
abjad> f(staff)
```

```
\new Staff {  
    c'8  
    d'8  
    e'8  
    f'8  
    g'8  
    a'8  
    b'8  
    c''8  
    d''8  
    e''8  
}
```

```
abjad> containertools.replace_smaller_left_half_of_elements_in_container_with_little_endian_rests(
Staff{7})
```

```
abjad> f(staff)
\new Staff {
    r8
    r2
    a'8
    b'8
    c''8
    d''8
    e''8
}
```

Return *container*.

containertools.replace_smaller_right_half_of_elements_in_container_with_big_endian_rests

`abjad.tools.containertools.replace_smaller_right_half_of_elements_in_container_with_big_endian_rests(container)`
New in version 2.0. Relace smaller right half of elements in *container* with big-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
    d''8
    e''8
}
```

```
abjad> containertools.replace_smaller_right_half_of_elements_in_container_with_big_endian_rests(
Staff{7})
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    r2
    r8
}
```

Return *container*.

containertools.replace_smaller_right_half_of_elements_in_container_with_little_endian_rests

abjad.tools.containertools.replace_smaller_right_half_of_elements_in_container_with_little_endian_rests

New in version 2.0. Replace smaller right half of elements in *container* with little-endian rests:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8 d''8 e''8")
```

```
abjad> f(staff)
```

```
\new Staff {
  c'8
  d'8
  e'8
  f'8
  g'8
  a'8
  b'8
  c''8
  d''8
  e''8
}
```

```
abjad> containertools.replace_smaller_right_half_of_elements_in_container_with_little_endian_rests(
Staff{7})
```

```
abjad> f(staff)
```

```
\new Staff {
  c'8
  d'8
  e'8
  f'8
  g'8
  r8
  r2
}
```

Return *container*.

containertools.report_container_modifications_as_string

abjad.tools.containertools.report_container_modifications_as_string.report_container_modifications_as_string

Report *container* modifications as string:

```
abjad> container = Container("c'8 d'8 e'8 f'8")
```

```
abjad> container.override.note_head.color = 'red'
```

```
abjad> container.override.note_head.style = 'harmonic'
```

```
abjad> f(container)
```

```
{
  \override NoteHead #'color = #red
  \override NoteHead #'style = #'harmonic
  c'8
  d'8
  e'8
  f'8
  \revert NoteHead #'color
  \revert NoteHead #'style
}
```



```

abjad> string = containertools.report_container_modifications_as_string(container)

abjad> print string # doctest: +SKIP
{
    \override NoteHead #'color = #red
    \override NoteHead #'style = #'harmonic

    %%% 4 components omitted %%%

    \revert NoteHead #'color
    \revert NoteHead #'style
}

```

Return string.

containertools.reverse_contents_of_container

abjad.tools.containertools.reverse_contents_of_container.**reverse_contents_of_container**(*container*)
 New in version 1.1. Reverse contents of *container*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves[:2])
BeamSpanner(c'8, d'8)
abjad> spannertools.SlurSpanner(staff.leaves[2:])
SlurSpanner(e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
    e'8 (
    f'8 )
}

abjad> containertools.reverse_contents_of_container(staff)
Staff{4}

abjad> f(staff) # doctest: +SKIP
\new Staff {
    f'8 (
    e'8 )
    d'8 [
    c'8 ]
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_reverse()` to `containertools.reverse_contents_of_container()`.

containertools.scale_contents_of_container

abjad.tools.containertools.scale_contents_of_container.**scale_contents_of_container**(*container*, *multiplier*)
 New in version 1.1. Scale contents of *container* by dot *multiplier*:

```

abjad> staff = Staff("c'8 d'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

abjad> containertools.scale_contents_of_container(staff, Duration(3, 2))
Staff{2}

abjad> f(staff)
\new Staff {
    c'8. [
    d'8. ]
}

```

Scale contents of *container* by tie *multiplier*:

```

abjad> staff = Staff("c'8 d'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

abjad> containertools.scale_contents_of_container(staff, Duration(5, 4))
Staff{4}

abjad> f(staff)
\new Staff {
    c'8 [ ~
    c'32
    d'8 ~
    d'32 ]
}

```

Scale contents of *container* by nonbinary *multiplier*:

```

abjad> staff = Staff("c'8 d'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8 ]
}

abjad> containertools.scale_contents_of_container(staff, Duration(4, 3))
Staff{2}

```

```

abjad> f(staff)
\new Staff {
  \times 2/3 {
    c'4 [
  }
  \times 2/3 {
    d'4 ]
  }
}

```

Return *container*. Changed in version 2.0: renamed `containertools.contents_scale()` to `containertools.scale_contents_of_container()`.

`containertools.set_container_multiplier`

`abjad.tools.containertools.set_container_multiplier.set_container_multiplier` (*container*, *multiplier*)

Set *container multiplier*:

```

abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")

abjad> f(tuplet)
\times 2/3 {
  c'8
  d'8
  e'8
}

abjad> containertools.set_container_multiplier(tuplet, Duration(3, 4))

abjad> f(tuplet)
\fraction \times 3/4 {
  c'8
  d'8
  e'8
}

```

Return `none`. Changed in version 2.0: renamed `containertools.multiplier_set()` to `containertools.set_container_multiplier()`.

`containertools.split_container_at_index_and_do_not_fracture_crossing_spanners`

`abjad.tools.containertools.split_container_at_index_and_do_not_fracture_crossing_spanners.`

Split *container* at *index* and do not fracture crossing spanners:

```

abjad> voice = Voice(Measure((3, 8), "c'8 c'8 c'8") * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice[:])

abjad> f(voice)
\new Voice {
  {

```

```
\time 3/8
c'8 [
d'8
e'8
}
{
\time 3/8
f'8
g'8
a'8 ]
}

abjad> containertools.split_container_at_index_and_do_not_fracture_crossing_spanners(voice[1], 1
(Measure(1/8, [f'8]), Measure(2/8, [g'8, a'8]))

abjad> f(voice)
\new Voice {
{
\time 3/8
c'8 [
d'8
e'8
}
{
\time 1/8
f'8
}
{
\time 2/8
g'8
a'8 ]
}
}
```

Leave spanners and leaves untouched.

Resize resizable containers.

Preserve container multiplier.

Preserve meter denominator.

Return split parts. Changed in version 2.0: renamed `split.unfractured_at_index()` to `containertools.split_container_at_index_and_do_not_fracture_crossing_spanners()`.

containertools.split_container_at_index_and_fracture_crossing_spanners

`abjad.tools.containertools.split_container_at_index_and_fracture_crossing_spanners.split_co`

Split *container* at *index* and fracture crossing spanners:

```
abjad> voice = Voice(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 c'8 c'8") * 2)
abjad> tuplet = voice[1]
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voi
abjad> beam = spannertools.BeamSpanner(voice[:])
```

```

abjad> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  \times 2/3 {
    f'8
    g'8
    a'8 ]
  }
}

abjad> left, right = containertools.split_container_at_index_and_fracture_crossing_spanners(tupl

abjad> f(voice)
\new Voice {
  \times 2/3 {
    c'8 [
      d'8
      e'8
    ]
  }
  \times 2/3 {
    f'8 ]
  }
  \times 2/3 {
    g'8 [
      a'8 ]
  }
}

```

Leave leaves untouched.

Create two new copies of *container*.

Empty *container* of original contents.

Return *split* parts. Changed in version 2.0: renamed `split.fractured_at_index()` to `containertools.split_container_at_index_and_fracture_crossing_spanners()`.

containertools.split_container_cyclically_by_counts_and_do_not_fracture_crossing_spanners

```
abjad.tools.containertools.split_container_cyclically_by_counts_and_do_not_fracture_crossing
```

Split *container* cyclically by *counts* and do not fracture crossing spanners:

```

abjad> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> voice = Voice([container])
abjad> beam = spannertools.BeamSpanner(voice)
abjad> slur = spannertools.SlurSpanner(container)

abjad> f(voice)
\new Voice {
  {
    c'8 [ (
      d'8
      e'8

```

```

        f'8
        g'8
        a'8
        b'8
        c''8 ] )
    }
}

abjad> containertools.split_container_cyclically_by_counts_and_do_not_fracture_crossing_spanners
[[{c'8}], [{d'8, e'8, f'8}], [{g'8}], [{a'8, b'8, c''8}]]

abjad> f(voice)
\new Voice {
    {
        c'8 [ (
    }
    {
        d'8
        e'8
        f'8
    }
    {
        g'8
    }
    {
        a'8
        b'8
        c''8 ] )
    }
}

```

Return list of list-wrapped container pieces. Changed in version 2.0: renamed `partition.cyclic_unfractured_by_counts()` to `containertools.split_container_cyclically_by_counts_and_do_not_fracture_crossing_spanners`

`containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners`

`abjad.tools.containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners`

Split *container* cyclically by *counts* and fracture crossing spanners:

```

abjad> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> voice = Voice([container])
abjad> beam = spannertools.BeamSpanner(voice)
abjad> slur = spannertools.SlurSpanner(container)

abjad> f(voice)
\new Voice {
    {
        c'8 [ (
        d'8
        e'8
        f'8
        g'8
        a'8
        b'8
        c''8 ] )
    }
}

```

```

    }
}

abjad> containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners(contain
[[{c'8}], [{d'8, e'8, f'8}], [{g'8}], [{a'8, b'8, c''8}]]

abjad> f(voice)
\new Voice {
  {
    c'8 ( ) [
  ]
  {
    d'8 (
    e'8
    f'8 )
  ]
  {
    g'8 ( )
  ]
  {
    a'8 (
    b'8
    c''8 ] )
  ]
}

```

Return list of list-wrapped container pieces. Changed in version 2.0: renamed `partition.cyclic_fractured_by_counts()` to `containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners()`.

containertools.split_container_once_by_counts_and_do_not_fracture_crossing_spanners

`abjad.tools.containertools.split_container_once_by_counts_and_do_not_fracture_crossing_spans`

Split *container* once by *counts* and do no fracture crossing spanners:

```

abjad> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> voice = Voice([container])
abjad> beam = spannertools.BeamSpanner(voice)
abjad> slur = spannertools.SlurSpanner(container)

abjad> f(voice)
\new Voice {
  {
    c'8 [ (
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8 ] )
  ]
}

abjad> containertools.split_container_once_by_counts_and_do_not_fracture_crossing_spanners(conta
[[{c'8}], [{d'8, e'8, f'8}], [{g'8, a'8, b'8, c''8}]]

```

```

abjad> f(voice)
\new Voice {
  {
    c'8 [ (
  }
  {
    d'8
    e'8
    f'8
  }
  {
    g'8
    a'8
    b'8
    c''8 ] )
  }
}

```

Return list of list-wrapped container pieces. Changed in version 2.0: renamed `partition.unfractured_by_counts()` to `containertools.split_container_once_by_counts_and_fracture_crossing_spanners()`.

`containertools.split_container_once_by_counts_and_fracture_crossing_spanners`

`abjad.tools.containertools.split_container_once_by_counts_and_fracture_crossing_spanners`. **SP**

Split *container* once by *counts* and fracture crossing spanners:

```

abjad> container = Container("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> voice = Voice([container])
abjad> beam = spannertools.BeamSpanner(voice)
abjad> slur = spannertools.SlurSpanner(container)

```

```

abjad> f(voice)
\new Voice {
  {
    c'8 [ (
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8 ] )
  }
}

```

```

abjad> containertools.split_container_once_by_counts_and_fracture_crossing_spanners(container, [
[[{c'8}], [{d'8, e'8, f'8}], [{g'8, a'8, b'8, c''8}]]

```

```

abjad> f(voice)
\new Voice {
  {
    c'8 ( ) [
  }
  {
    d'8 (
    e'8

```



```

        f'8 )
    }
    {
        g'8 (
        a'8
        b'8
        c''8 ] )
    }
}

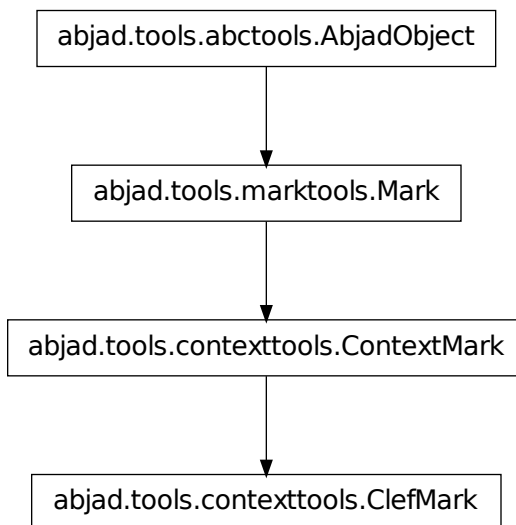
```

Return list of list-wrapped container pieces. Changed in version 2.0: renamed `partition.fractured_by_counts()` to `containertools.split_container_once_by_counts_and_fra`

contexttools

contexttools concrete classes

contexttools.ClefMark



class `abjad.tools.contexttools.ClefMark.ClefMark`(*arg*, *target_context=None*)

New in version 2.0. Abjad model of a clef:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

abjad> f(staff)
\new Staff {
    \clef "treble"

```

```

        c' 8
        d' 8
        e' 8
        f' 8
    }

```

Clef marks target the staff context by default.

Read-only Properties

`ClefMark.effective_context`

Read-only reference to effective context of context mark:

```

abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ClefMark.format`

Read-only LilyPond format of clef:

```

abjad> clef = contexttools.ClefMark('treble')
abjad> clef.format
'\clef "treble"'

```

Return string.

`ClefMark.middle_c_position`

Read-only middle-C position of clef:

```

abjad> clef = contexttools.ClefMark('treble')
abjad> clef.middle_c_position
-6

```

Return integer number of stafflines.

`ClefMark.start_component`

Read-only reference to mark start component:

```

abjad> note = Note("c' 4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c' 4")

```

Return component or none.

Note: Inherited from `marktools.Mark`

`ClefMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

`ClefMark.clef_name`

Get clef name:

```
abjad> clef = contexttools.ClefMark('treble')
abjad> clef.clef_name
'treble'
```

Set clef name:

```
abjad> clef.clef_name = 'alto'
abjad> clef.clef_name
'alto'
```

Return string.

Methods

`ClefMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`ClefMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

ClefMark.__call__(*args)

Note: Inherited from `marktools.Mark`

ClefMark.__delattr__(*args)

Note: Inherited from `marktools.Mark`

ClefMark.__eq__(arg)

ClefMark.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ClefMark.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

ClefMark.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

ClefMark.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ClefMark.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ClefMark.__ne__(arg)

Note: Inherited from `marktools.Mark`

ClefMark.__repr__()

Note: Inherited from `marktools.Mark`

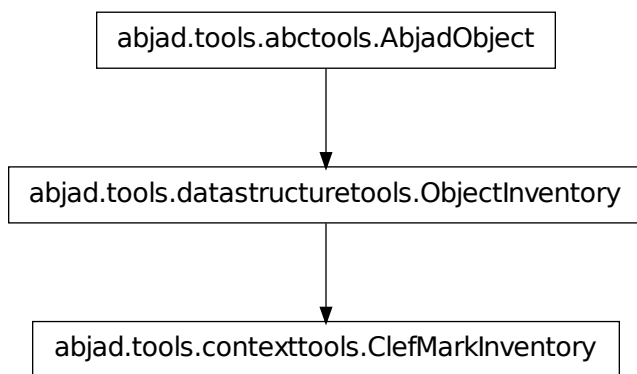
```
ClefMark.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
ClefMark.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

`contexttools.ClefMarkInventory`



```
class abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory (tokens=None, inventory_name=None)
```

New in version 2.8. Abjad model of an ordered list of clefs:

```
abjad> inventory = contexttools.ClefMarkInventory(['treble', 'bass'])
```

```
abjad> inventory
ClefMarkInventory([ClefMark('treble'), ClefMark('bass')])
```

```
abjad> 'treble' in inventory
True
```

```
abjad> contexttools.ClefMark('treble') in inventory
True
```

```
abjad> 'alto' in inventory
False
```

Clef mark inventories implement list interface and are mutable.

Read/write Properties

`ClefMarkInventory.inventory_name`

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`ClefMarkInventory.append(token)`

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.extend(tokens)`

Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.reverse()`
L.reverse() – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.sort()`
L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

Note: Inherited from `__builtin__.list`

Special Methods

`ClefMarkInventory.__add__()`
x.__add__(y) <==> x+y

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.__delattr__()`
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

`ClefMarkInventory.__delitem__()`
x.__delitem__(y) <==> del x[y]

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__delslice__()`
x.__delslice__(i, j) <==> del x[i:j]

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__eq__()`
x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__ge__()`
x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__getitem__()
x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__getslice__()
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__gt__()
x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__iadd__()
x.__iadd__(y) <==> x+=y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__imul__()
x.__imul__(y) <==> x*=y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__le__()
x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__len__() <==> len(x)

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__lt__()
x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.list`

ClefMarkInventory.__mul__()
x.__mul__(n) <==> x*n

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__repr__()`

Note: Inherited from `datastructuretools.ObjectInventory`

`ClefMarkInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__rmul__()`
`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ClefMarkInventory.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

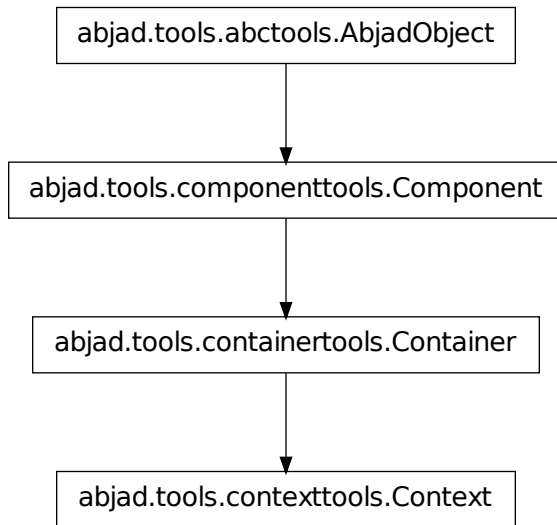
`ClefMarkInventory.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`
 Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ClefMarkInventory.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

contexttools.Context



class `abjad.tools.contexttools.Context.Context`.**Context** (*music=None*, *text_name='Context'*, *name=None*)

Abjad model of a horizontal layer of music.

Read-only Properties

`Context.contents_duration`

Note: Inherited from `containertools.Container`

`Context.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Context.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```

abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
  
```

Context.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Context.format

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

Context.is_semantic**Context.leaves**

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Context.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Context.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Context.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Context.**parent**

Note: Inherited from `componenttools.Component`

Context.**preprolated_duration**

Note: Inherited from `containertools.Container`

Context.**prolated_duration**

Note: Inherited from `componenttools.Component`

Context.**prolation**

Note: Inherited from `componenttools.Component`

Context.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Context.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Context.**context_name**

Read / write name of context as a string.

Context.**is_nonsemantic**

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'
```

```
abjad> voice.is_nonsemantic = True
```

```
abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
```

```

        s1 * 5/16
    }
    {
        \time 5/16
        s1 * 5/16
    }
}

```

```

abjad> voice.is_nonsemantic
True

```

Get indicator of nonsemantic voice:

```

abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False

```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Context **.is_parallel**

Get parallel container:

```

abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False

```

Return boolean.

Set parallel container:

```

abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

```

```
}
>>
```

Return none.

Note: Inherited from `containertools.Container`

`Context.name`

Read-write name of context. Must be string or none.

Methods

`Context.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.append(Note("f'8"))
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`Context.extend(expr)`

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])
```

```
abjad> f(container)
{
    c'8 [
```

```
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Context.**index** (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Context.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Context.**pop** (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

Context. **remove** (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

Context. **__add__** (*expr*)

Concatenate containers self and *expr*. The operation `c = a + b` returns a new Container *c* with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`Context.__contains__(expr)`
True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Context.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Context.__delitem__(i)`
Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Context.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Context.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Context.__getitem__(i)`
Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Context.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`Context.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

Context.**__iadd__**(*expr*)
__iadd__ avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

Context.**__imul__**(*total*)
Multiply contents of container 'total' times. Return multiplied container.

Note: Inherited from `containertools.Container`

Context.**__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Context.**__len__**()
Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

Context.**__lt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Context.**__mul__**(*n*)

Note: Inherited from `componenttools.Component`

Context.**__ne__**(*arg*)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Context.**__radd__**(*expr*)
Extend container by contents of *expr* to the right.

Note: Inherited from `containertools.Container`

Context.**__repr__**()
Changed in version 2.0. Named contexts now print name at the interpreter.

Context.__rmul__(n)

Note: Inherited from `componenttools.Component`

Context.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

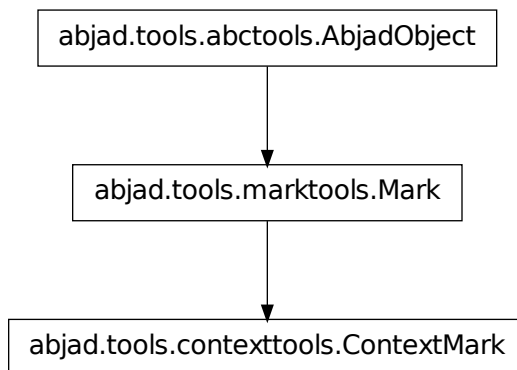
Context.__setitem__(i, expr)
 Set 'expr' in self at nonnegative integer index i. Or, set 'expr' in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

Context.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

contexttools.ContextMark



class `abjad.tools.contexttools.ContextMark.ContextMark`(target_context=None)
 New in version 2.0. Abstract class from which concrete context marks inherit:

```
abjad> note = Note("c'4")
```

```
abjad> contexttools.ContextMark()(note)
ContextMark()(c'4)
```

Context marks override `__call__` to attach to Abjad components.

Context marks implement `__slots__`.

Read-only Properties

`ContextMark.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True
```

Return context mark or none.

`ContextMark.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ContextMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Methods

`ContextMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

`ContextMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Special Methods

ContextMark.__call__(*args)

Note: Inherited from `marktools.Mark`

ContextMark.__delattr__(*args)

Note: Inherited from `marktools.Mark`

ContextMark.__eq__(arg)

Note: Inherited from `marktools.Mark`

ContextMark.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ContextMark.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

ContextMark.__hash__() $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

ContextMark.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ContextMark.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ContextMark.__ne__(arg)

Note: Inherited from `marktools.Mark`

```
ContextMark.__repr__()
```

Note: Inherited from `marktools.Mark`

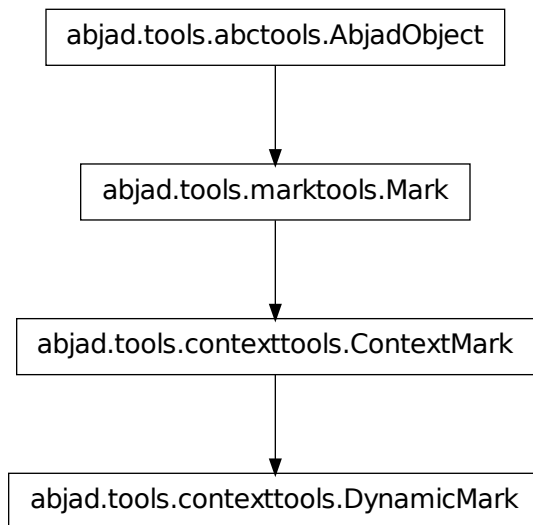
```
ContextMark.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

$$\text{ContextMark}._\textbf{str}_\text{()}\leq\Rightarrow str(x)$$

Note: Inherited from `__builtin__.object`

contexttools.DynamicMark

[illegible]

New in version 2.0. Abjad model of a dynamic mark:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

abjad> f(staff)
\new Staff {
    c'8 \f
    d'8
    e'8
    f'8
}
```

Dynamic marks target the staff context by default.

Read-only Properties

`DynamicMark.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`DynamicMark.format`

Read-only LilyPond input format of dynamic mark:

```
abjad> dynamic_mark = contexttools.DynamicMark('f')
abjad> dynamic_mark.format
'\f'
```

Return string.

`DynamicMark.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`DynamicMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

`DynamicMark.dynamic_name`

Get dynamic name:

```
abjad> dynamic = contexttools.DynamicMark('f')
abjad> dynamic.dynamic_name
'f'
```

Set dynamic name:

```
abjad> dynamic.dynamic_name = 'p'
abjad> dynamic.dynamic_name
'p'
```

Return string.

Methods

`DynamicMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`DynamicMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

`DynamicMark.__call__(*args)`

`DynamicMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`DynamicMark.__eq__(arg)`

`DynamicMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DynamicMark.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DynamicMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMark.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`DynamicMark.__repr__()`

Note: Inherited from `marktools.Mark`

`DynamicMark.__setattr__()`

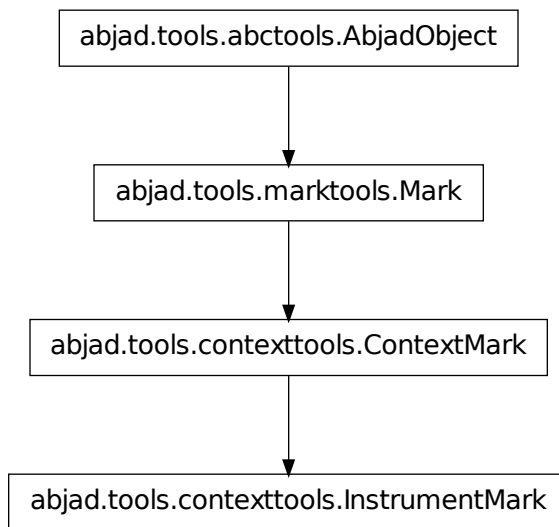
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DynamicMark.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

contexttools.InstrumentMark



```

class abjad.tools.contexttools.InstrumentMark.InstrumentMark (instrument_name,
                                                                short_instrument_name,
                                                                in-
                                                                stru-
                                                                ment_name_markup=None,
                                                                short_instrument_name_
                                                                tar-
                                                                get_context=None)
  
```

New in version 2.0. Abjad model of an instrument change:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  
```

```
e' 8
f' 8
}
```

Instrument marks target staff context by default.

Read-only Properties

`InstrumentMark.default_instrument_name`

Read-only default instrument name.

Return string.

`InstrumentMark.default_short_instrument_name`

Read-only default short instrument name.

Return string.

`InstrumentMark.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`InstrumentMark.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

`InstrumentMark.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`InstrumentMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

`InstrumentMark.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

`InstrumentMark.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

`InstrumentMark.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

`InstrumentMark.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Methods

`InstrumentMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`InstrumentMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

`InstrumentMark.__call__(*args)`

Note: Inherited from `marktools.Mark`

`InstrumentMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`InstrumentMark.__eq__(arg)`

`InstrumentMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`InstrumentMark.__hash__()`

`InstrumentMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentMark.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`InstrumentMark.__repr__()`

Note: Inherited from `marktools.Mark`

`InstrumentMark.__setattr__()`

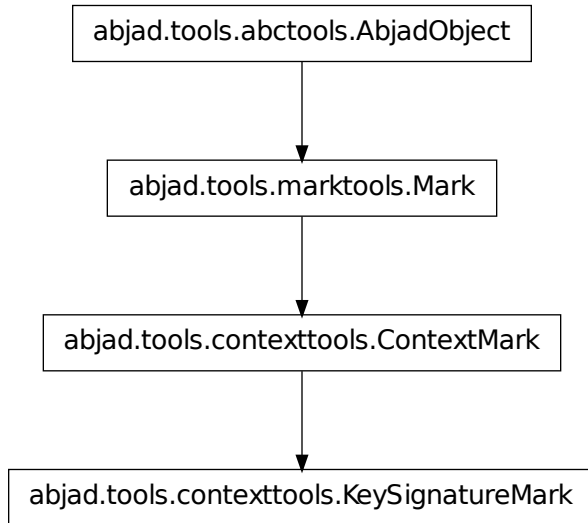
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InstrumentMark.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

contexttools.KeySignatureMark



class `abjad.tools.contexttools.KeySignatureMark.KeySignatureMark` (*tonic*,
mode,
tar-
get_context=No

New in version 2.0. Abjad model of a key signature setting or key signature change:

```

abjad> staff = Staff("e'8 fs'8 gs'8 a'8")

abjad> contexttools.KeySignatureMark('e', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('e'), Mode('major'))(Staff{4})

abjad> f(staff)
\new Staff {
  \key e \major
  e'8
  fs'8
  gs'8
  a'8
}
  
```

Key signature marks target staff context by default.

Read-only Properties

`KeySignatureMark.effective_context`

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
  
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

KeySignatureMark.format

Read-only LilyPond format of key signature mark:

```
abjad> key_signature = contexttools.KeySignatureMark('e', 'major')
abjad> key_signature.format
'\\key e \\major'
```

Return string.

KeySignatureMark.name

Read-only name of key signature:

```
abjad> key_signature = contexttools.KeySignatureMark('e', 'major')
abjad> key_signature.name
'E major'
```

Return string.

KeySignatureMark.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

KeySignatureMark.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

KeySignatureMark.mode

Get mode of key signature:


```
abjad> key_signature = contexttools.KeySignatureMark('e', 'major')
abjad> key_signature.mode
Mode('major')
```

Set mode of key signature:

```
abjad> key_signature.mode = 'minor'
abjad> key_signature.mode
Mode('minor')
```

Return mode.

`KeySignatureMark.tonic`

Get tonic of key signature:

```
abjad> key_signature = contexttools.KeySignatureMark('e', 'major')
abjad> key_signature.tonic
NamedChromaticPitchClass('e')
```

Set tonic of key signature:

```
abjad> key_signature.tonic = 'd'
abjad> key_signature.tonic
NamedChromaticPitchClass('d')
```

Return named chromatic pitch.

Methods

`KeySignatureMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`KeySignatureMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.start_component
Note("c'4")
```

```
abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

`KeySignatureMark.__call__(*args)`

Note: Inherited from `marktools.Mark`

`KeySignatureMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`KeySignatureMark.__eq__(arg)`

`KeySignatureMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`KeySignatureMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`KeySignatureMark.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`KeySignatureMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`KeySignatureMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`KeySignatureMark.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`KeySignatureMark.__repr__()`

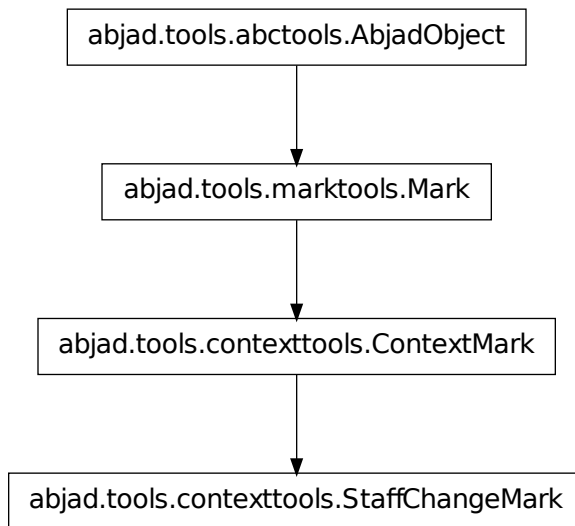
Note: Inherited from `marktools.Mark`

```
KeySignatureMark.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
KeySignatureMark.__str__()
```

contexttools.StaffChangeMark



```
class abjad.tools.contexttools.StaffChangeMark.StaffChangeMark (staff,
                                                                tar-
                                                                get_context=None)
```

New in version 2.0. Abjad model of a staff change:

```
abjad> piano_staff = scoretools.PianoStaff([])
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'
abjad> piano_staff.extend([rh_staff, lh_staff])
```

```
abjad> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
```

```

        s2
    }
>>

abjad> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

abjad> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

```

Staff change marks target staff context by default.

Read-only Properties

StaffChangeMark.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

StaffChangeMark.format

Read-only LilyPond format of staff change mark:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> staff.name = 'RHStaff'
abjad> staff_change = contexttools.StaffChangeMark(staff)
abjad> staff_change.format
'\change Staff = RHStaff'

```

Return string.

StaffChangeMark.start_component

Read-only reference to mark start component:

```

abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

```

Return component or none.

Note: Inherited from `marktools.Mark`

`StaffChangeMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

`StaffChangeMark.staff`

Get staff of staff change mark:

```
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> staff_change = contexttools.StaffChangeMark(rh_staff)
abjad> staff_change.staff
Staff-"RHStaff"{4}
```

Set staff of staff change mark:

```
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'
abjad> staff_change.staff = lh_staff
abjad> staff_change.staff
Staff-"LHStaff"{1}
```

Return staff.

Methods

`StaffChangeMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`StaffChangeMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

`StaffChangeMark.__call__(*args)`

Note: Inherited from `marktools.Mark`

`StaffChangeMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`StaffChangeMark.__eq__(arg)`

`StaffChangeMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffChangeMark.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`StaffChangeMark.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`StaffChangeMark.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffChangeMark.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffChangeMark.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`StaffChangeMark.__repr__()`

Note: Inherited from `marktools.Mark`

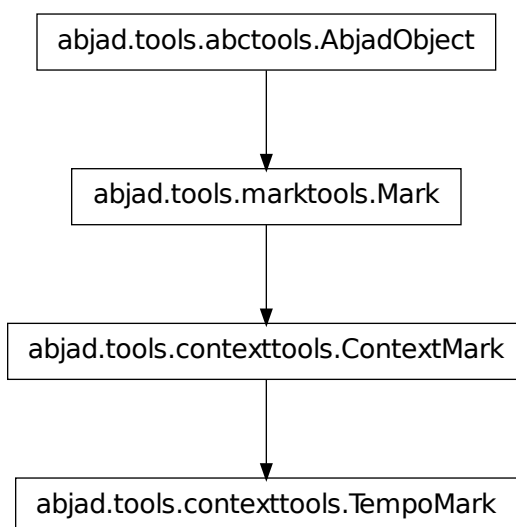
`StaffChangeMark.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`StaffChangeMark.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`contexttools.TempoMark`



class `abjad.tools.contexttools.TempoMark.TempoMark(*args, **kwargs)`
 New in version 2.0. Abjad model of a tempo indication:

```

abjad> score = Score([])
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score.append(staff)

abjad> contexttools.TempoMark(Duration(1, 8), 52)(staff[0])
TempoMark(Duration(1, 8), 52)(c'8)

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

```

Tempo marks target **score** context by default.

Initialization allows many different types of input argument structure.

Read-only Properties

TempoMark.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

TempoMark.format

Read-only LilyPond format of tempo mark:

```

abjad> tempo = contexttools.TempoMark(Duration(1, 8), 52)
abjad> tempo.format
'\tempo 8=52'
abjad> tempo.textual_indication = 'Gingerly'
abjad> tempo.format
'\tempo "Gingerly" 8=52'
abjad> tempo.units_per_minute = (52, 56)
abjad> tempo.format
'\tempo "Gingerly" 8=52~56'

```

Return string.

TempoMark.is_imprecise

True if tempo mark is entirely textual, or if tempo mark's `units_per_minute` is a range:

```

abjad> contexttools.TempoMark(Duration(1, 4), 60).is_imprecise
False
abjad> contexttools.TempoMark('Langsam', 4, 60).is_imprecise
False

```



```
abjad> contexttools.TempoMark('Langsam').is_imprecise
True
abjad> contexttools.TempoMark('Langsam', 4, (35, 50)).is_imprecise
True
abjad> contexttools.TempoMark(Duration(1, 4), (35, 50)).is_imprecise
True
```

Return boolean.

TempoMark.quarters_per_minute

Read-only quarters per minute of tempo mark:

```
abjad> tempo = contexttools.TempoMark(Duration(1, 8), 52)
abjad> tempo.quarters_per_minute
Duration(104, 1)
```

Return fraction, or tuple if units_per_minute is a range, or None if tempo mark is imprecise.

TempoMark.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

TempoMark.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

TempoMark.duration

Get duration of tempo mark:

```
abjad> tempo = contexttools.TempoMark(Duration(1, 8), 52)
abjad> tempo.duration
Duration(1, 8)
```

Set duration of tempo mark:

```
abjad> tempo.duration = Duration(1, 4)
abjad> tempo.duration
Duration(1, 4)
```

Return duration, or None if tempo mark is imprecise.

`TempoMark.textual_indication`

Get textual indication of tempo mark:

```
abjad> tempo = contexttools.TempoMark('Langsam', Duration(1, 8), 52)
abjad> tempo.textual_indication
'Langsam'
```

Return string or None.

`TempoMark.units_per_minute`

Get units per minute of tempo mark:

```
abjad> tempo = contexttools.TempoMark(Duration(1, 8), 52)
abjad> tempo.units_per_minute
52
```

Set units per minute of tempo mark:

```
abjad> tempo.units_per_minute = 56
abjad> tempo.units_per_minute
56
```

Return number.

Methods

`TempoMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`TempoMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`TempoMark.is_tempo_mark_token(expr)`

True when *expr* has the form of a tempo mark initializer:

```
abjad> tempo_mark = contexttools.TempoMark(Duration(1, 4), 72)
abjad> tempo_mark.is_tempo_mark_token((Duration(1, 4), 84))
True
```

Otherwise false:

```
abjad> tempo_mark.is_tempo_mark_token(84)
False
```

Return boolean.

Special Methods

TempoMark.__add__(*expr*)

TempoMark.__call__(**args*)

Note: Inherited from `marktools.Mark`

TempoMark.__delattr__(**args*)

Note: Inherited from `marktools.Mark`

TempoMark.__div__(*expr*)

TempoMark.__eq__(*expr*)

TempoMark.__ge__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TempoMark.__gt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

TempoMark.__hash__() $\leq \Rightarrow$ *hash*(*x*)

Note: Inherited from `__builtin__.object`

TempoMark.__le__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TempoMark.__lt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TempoMark.__mul__` (*multiplier*)

`TempoMark.__ne__` (*arg*)

Note: Inherited from `marktools.Mark`

`TempoMark.__repr__` ()

Note: Inherited from `marktools.Mark`

`TempoMark.__setattr__` ()
`x.__setattr__('name', value) <==> x.name = value`

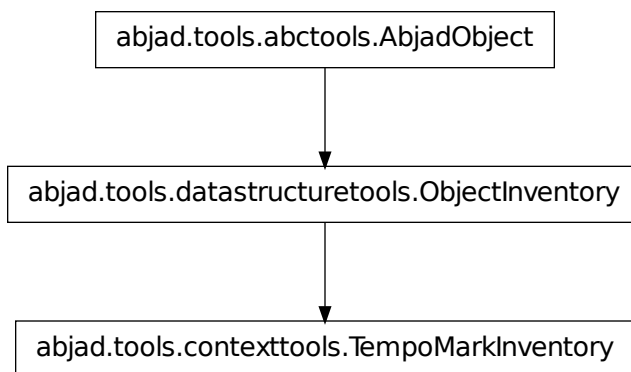
Note: Inherited from `__builtin__.object`

`TempoMark.__str__` () <==> *str(x)*

Note: Inherited from `__builtin__.object`

`TempoMark.__sub__` (*expr*)

`contexttools.TempoMarkInventory`



class `abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory` (*tokens:*
in-
ven-
tory_no)

New in version 2.7. Abjad model of an ordered list of tempo marks:

```
abjad> contexttools.TempoMarkInventory([(‘Andante’, Duration(1, 8), 72), (‘Allegro’, Duration(1, 8), 72)])
TempoMarkInventory([TempoMark(‘Andante’, Duration(1, 8), 72), TempoMark(‘Allegro’, Duration(1, 8), 72)])
```

Tempo mark inventories implement list interface and are mutable.

Read/write Properties

`TempoMarkInventory.inventory_name`

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`TempoMarkInventory.append(token)`

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.extend(tokens)`

Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`TempoMarkInventory.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TempoMarkInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__getitem__()
 $x.\text{__getitem__}(y) \iff x[y]$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__getslice__()
 $x.\text{__getslice__}(i, j) \iff x[i:j]$

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__gt__()
 $x.\text{__gt__}(y) \iff x > y$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__iadd__()
 $x.\text{__iadd__}(y) \iff x += y$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__imul__()
 $x.\text{__imul__}(y) \iff x *= y$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__iter__() $\iff \text{iter}(x)$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__le__()
 $x.\text{__le__}(y) \iff x \leq y$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__len__() $\iff \text{len}(x)$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__lt__()
 $x.\text{__lt__}(y) \iff x < y$

Note: Inherited from `__builtin__.list`

TempoMarkInventory.__mul__()
 $x.\text{__mul__}(n) \iff x * n$

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__repr__()`

Note: Inherited from `datastructuretools.ObjectInventory`

`TempoMarkInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__rmul__()`
`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TempoMarkInventory.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

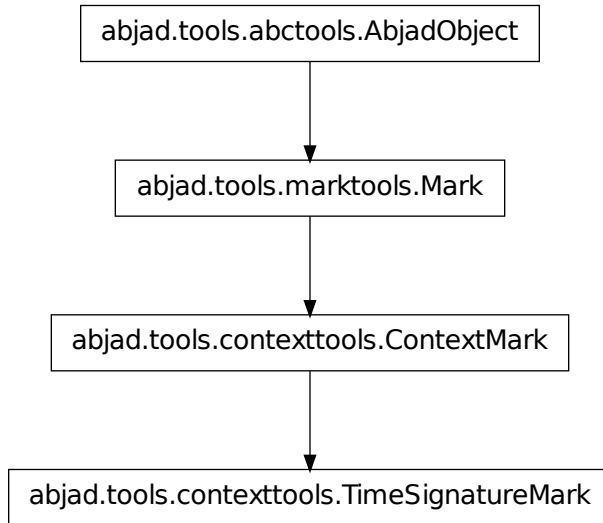
`TempoMarkInventory.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`TempoMarkInventory.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

contexttools.TimeSignatureMark

class `abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark(*args, **kwargs)`

New in version 2.0. Abjad model of a time signature:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)

abjad> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
  
```

Abjad time signature marks target **staff context** by default.

Initialize time signature marks to **score context** like this:

```

abjad> contexttools.TimeSignatureMark((4, 8), target_context = Score)
TimeSignatureMark((4, 8), target_context = Score)
  
```

Time signatures are immutable.

Read-only Properties

`TimeSignatureMark.duration`

Read-only duration of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.duration
Duration(3, 8)
```

Return fraction.

TimeSignatureMark.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

TimeSignatureMark.format

Read-only LilyPond format of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.format
'\\time 3/8'
```

Return string.

TimeSignatureMark.is_nonbinary

Read-only indicator true when time signature mark is nonbinary:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.is_nonbinary
False
```

Return boolean.

TimeSignatureMark.multiplier

Read-only multiplier of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.multiplier
Fraction(1, 1)
```

Return fraction.

TimeSignatureMark.pair

New in version 2.8. Read-only numerator / denominator pair of time signature:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.pair
(3, 8)
```

Return length-2 tuple.

TimeSignatureMark.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`TimeSignatureMark.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Read/write Properties

`TimeSignatureMark.denominator`

Get denominator of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature
TimeSignatureMark((3, 8))
abjad> time_signature.denominator
8
```

Set denominator of time signature mark:

```
abjad> time_signature.denominator = 16
abjad> time_signature.denominator
16
```

Return integer.

`TimeSignatureMark.numerator`

Get numerator of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8))
abjad> time_signature.numerator
3
```

Set numerator of time signature mark:

```
abjad> time_signature.numerator = 4
abjad> time_signature.numerator
4
```

Set integer.

`TimeSignatureMark.partial`

Get partial measure pick-up of time signature mark:

```
abjad> time_signature = contexttools.TimeSignatureMark((3, 8), partial = Duration(1, 8))
abjad> time_signature.partial
Duration(1, 8)
```

Set partial measure pick-up of time signature mark:

```
abjad> time_signature.partial = Duration(1, 4)
abjad> time_signature.partial
Duration(1, 4)
```

Set fraction or none.

Methods

`TimeSignatureMark.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`TimeSignatureMark.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.start_component
Note("c'4")
```

```
abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Special Methods

`TimeSignatureMark.__call__(*args)`

Note: Inherited from `marktools.Mark`

`TimeSignatureMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`TimeSignatureMark.__eq__(arg)`

`TimeSignatureMark.__ge__(arg)`

`TimeSignatureMark.__gt__(arg)`

TimeSignatureMark.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

TimeSignatureMark.__le__(arg)

TimeSignatureMark.__lt__(arg)

TimeSignatureMark.__ne__(arg)

TimeSignatureMark.__nonzero__()

TimeSignatureMark.__repr__()

TimeSignatureMark.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

TimeSignatureMark.__str__()

contexttools functions

contexttools.detach_clef_marks_attached_to_component

abjad.tools.contexttools.detach_clef_marks_attached_to_component.**detach_clef_marks_attached_to_component**

New in version 2.3. Detach clef marks attached to *component*:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> clef_mark = contexttools.ClefMark('treble')
abjad> clef_mark.attach(staff)
ClefMark('treble') (Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \clef "treble"
  c'4
  d'4
  e'4
  f'4
}
```

```
abjad> contexttools.detach_clef_marks_attached_to_component(staff)
(ClefMark('treble'),)
```

```
abjad> f(staff)
\new Staff {
  c'4
  d'4
  e'4
  f'4
}
```

Return tuple of zero or more clef marks.

contexttools.detach_context_marks_attached_to_component

`abjad.tools.contexttools.detach_context_marks_attached_to_component.detach_context_marks_at`

New in version 2.0. Detach context marks attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> clef_mark = contexttools.ClefMark('treble')(staff)
abjad> dynamic_mark = contexttools.DynamicMark('p')(staff[0])
abjad> f(staff)
\new Staff {
    \clef "treble"
    c'8 \p
    d'8
    e'8
    f'8
}

abjad> contexttools.detach_context_marks_attached_to_component(staff[0])
(DynamicMark('p'),)

abjad> f(staff)
\new Staff {
    \clef "treble"
    c'8
    d'8
    e'8
    f'8
}
```

Return tuple of zero or context marks.

contexttools.detach_dynamic_marks_attached_to_component

`abjad.tools.contexttools.detach_dynamic_marks_attached_to_component.detach_dynamic_marks_at`

New in version 2.3. Detach dynamic marks attached to *component*:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> dynamic_mark = contexttools.DynamicMark('p')
abjad> dynamic_mark.attach(staff[0])
DynamicMark('p')(c'4)

abjad> f(staff)
\new Staff {
    c'4 \p
    d'4
    e'4
    f'4
}

abjad> contexttools.detach_dynamic_marks_attached_to_component(staff[0])
(DynamicMark('p'),)
```

```

abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}

```

Return tuple of zero or more dynamic marks.

contexttools.detach_instrument_marks_attached_to_component

abjad.tools.contexttools.detach_instrument_marks_attached_to_component.**detach_instrument_marks_attached_to_component**

New in version 2.1. Detach instrument marks attached to *component*:

```

abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> instrument_mark = contexttools.InstrumentMark('Violin ', 'Vn. ')
abjad> instrument_mark.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ') (Staff{4})

abjad> f(staff)
\new Staff {
    \set Staff.instrumentName = \markup { Violin }
    \set Staff.shortInstrumentName = \markup { Vn. }
    c'4
    d'4
    e'4
    f'4
}

abjad> contexttools.detach_instrument_marks_attached_to_component(staff)
(InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. '),)

abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
}

```

Return tuple of zero or more instrument marks.

contexttools.detach_key_signature_marks_attached_to_component

abjad.tools.contexttools.detach_key_signature_marks_attached_to_component.**detach_key_signature_marks_attached_to_component**

New in version 2.3. Detach key signature marks attached to *component*:

```

abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> key_signature_mark = contexttools.KeySignatureMark('c', 'major')
abjad> key_signature_mark.attach(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')) (Staff{4})

abjad> f(staff)
\new Staff {
    \key c \major
}

```

```

        c' 4
        d' 4
        e' 4
        f' 4
    }

abjad> contexttools.detach_key_signature_marks_attached_to_component(staff)
(KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major')),)

abjad> f(staff)
\new Staff {
    c' 4
    d' 4
    e' 4
    f' 4
}

```

Return tuple of zero or more key signature marks.

contexttools.detach_staff_change_marks_attached_to_component

abjad.tools.contexttools.detach_staff_change_marks_attached_to_component.**detach_staff_change_marks_attached_to_component**
 New in version 2.3. Detach staff change marks attached to *component*:

```

abjad> piano_staff = scoretools.PianoStaff([])
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'
abjad> piano_staff.extend([rh_staff, lh_staff])
abjad> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

abjad> f(piano_staff)
\new PianoStaff <<
    \context Staff = "RHStaff" {
        c' 8
        d' 8
        \change Staff = LHStaff
        e' 8
        f' 8
    }
    \context Staff = "LHStaff" {
        s2
    }
>>

abjad> contexttools.detach_staff_change_marks_attached_to_component(rh_staff[2])
(StaffChangeMark(Staff-"LHStaff"{1}),)

```

Return tuple of zero or more staff change marks.

contexttools.detach_tempo_marks_attached_to_component

abjad.tools.contexttools.detach_tempo_marks_attached_to_component.**detach_tempo_marks_attached_to_component**
 New in version 2.3. Detach tempo marks attached to *component*:


```

abjad> score = Score([])
abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> score.append(staff)

abjad> tempo_mark = contexttools.TempoMark(Duration(1, 8), 52)
abjad> tempo_mark.attach(staff)
TempoMark(Duration(1, 8), 52) (Staff{4})

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>

abjad> contexttools.detach_tempo_marks_attached_to_component(staff)
(TempoMark(Duration(1, 8), 52),)

abjad> f(score)
\new Score <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
  }
>>

```

Return tuple of zero or more tempo marks.

contexttools.detach_time_signature_marks_attached_to_component

abjad.tools.contexttools.detach_time_signature_marks_attached_to_component.**detach_time_sig**

New in version 2.0. Detach time signature marks attached to *component*:

```

abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> contexttools.TimeSignatureMark((4, 4))(staff[0])
TimeSignatureMark((4, 4))(c'4)

abjad> f(staff)
\new Staff {
  \time 4/4
  c'4
  d'4
  e'4
  f'4
}

abjad> contexttools.detach_time_signature_marks_attached_to_component(staff[0])
(TimeSignatureMark((4, 4)),)

abjad> f(staff)
\new Staff {

```

```

    c' 4
    d' 4
    e' 4
    f' 4
}

```

Return tuple of zero or more time signature marks.

contexttools.get_clef_mark_attached_to_component

abjad.tools.contexttools.get_clef_mark_attached_to_component.get_clef_mark_attached_to_comp

New in version 2.3. Get clef mark attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

```

```

abjad> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

```

```

abjad> contexttools.get_clef_mark_attached_to_component(staff)
ClefMark('treble')(Staff{4})

```

Return clef mark.

Raise missing mark error when no clef mark attached to *component*.

contexttools.get_clef_marks_attached_to_component

abjad.tools.contexttools.get_clef_marks_attached_to_component.get_clef_marks_attached_to_c

New in version 2.3. Get clef marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

```

```

abjad> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

```

```

abjad> contexttools.get_clef_marks_attached_to_component(staff)
(ClefMark('treble')(Staff{4}),)

```

Return tuple of zero or more clef marks.

contexttools.get_context_mark_attached_to_component

`abjad.tools.contexttools.get_context_mark_attached_to_component.get_context_mark_attached_to_component`

New in version 2.3. Get context mark attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

abjad> f(staff)
\new Staff {
    \clef "treble"
    c'8
    d'8
    e'8
    f'8
}

abjad> contexttools.get_context_mark_attached_to_component(staff)
ClefMark('treble')(Staff{4})
```

Return context mark.

Raise missing mark error when no context mark attaches to *component*.

contexttools.get_context_marks_attached_to_any_improper_parent_of_component

`abjad.tools.contexttools.get_context_marks_attached_to_any_improper_parent_of_component.get_context_marks_attached_to_any_improper_parent_of_component`

New in version 2.0. Get all context marks attached to any improper parent of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
abjad> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

abjad> f(staff)
\new Staff {
    \clef "treble"
    c'8 \f
    d'8
    e'8
    f'8
}

abjad> contexttools.get_context_marks_attached_to_any_improper_parent_of_component(staff[0]) # d
set([DynamicMark('f')(c'8), ClefMark('treble')(Staff{4})])
```

Return unordered set of zero or more context marks. Changed in version 2.0: renamed `contexttools.get_all_context_marks_attached_to_any_improper_parent_of_component()` to `contexttools.get_context_marks_attached_to_any_improper_parent_of_component()`.

contexttools.get_context_marks_attached_to_component

abjad.tools.contexttools.get_context_marks_attached_to_component.get_context_marks_attached_to_component

New in version 2.0. Get context marks attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
abjad> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

abjad> f(staff)
\new Staff {
  \clef "treble"
  c'8 \p
  d'8
  e'8
  f'8
}

abjad> contexttools.get_context_marks_attached_to_component(staff[0])
(DynamicMark('p')(c'8),)
```

Return tuple of zero or more context marks. Changed in version 2.0: re-named contexttools.get_context_marks_attached_to_component() to contexttools.get_context_marks_attached_to_component().

contexttools.get_dynamic_mark_attached_to_component

abjad.tools.contexttools.get_dynamic_mark_attached_to_component.get_dynamic_mark_attached_to_component

New in version 2.3. Get dynamic mark attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

abjad> f(staff)
\new Staff {
  c'8 \p
  d'8
  e'8
  f'8
}

abjad> contexttools.get_dynamic_mark_attached_to_component(staff[0])
DynamicMark('p')(c'8)
```

Return dynamic mark.

Raise missing mark error when no dynamic mark attaches to *component*.

Raise extra mark error when more than one dynamic mark attaches to *component*.

contexttools.get_dynamic_marks_attached_to_component

`abjad.tools.contexttools.get_dynamic_marks_attached_to_component.get_dynamic_marks_attached_to_component`

New in version 2.0. Get dynamic marks attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

abjad> f(staff)
\new Staff {
    c'8 \p
    d'8
    e'8
    f'8
}

abjad> contexttools.get_dynamic_marks_attached_to_component(staff[0])
(DynamicMark('p')(c'8),)
```

Return tuple of zero or more dynamic marks.

contexttools.get_effective_clef

`abjad.tools.contexttools.get_effective_clef.get_effective_clef(component)`

New in version 2.0. Get effective clef of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

abjad> f(staff)
\new Staff {
    \clef "treble"
    c'8
    d'8
    e'8
    f'8
}

abjad> for note in staff:
...     print note, contexttools.get_effective_clef(note)
...
c'8 ClefMark('treble')(Staff{4})
d'8 ClefMark('treble')(Staff{4})
e'8 ClefMark('treble')(Staff{4})
f'8 ClefMark('treble')(Staff{4})
```

Return clef mark or none.

contexttools.get_effective_context_mark

`abjad.tools.contexttools.get_effective_context_mark.get_effective_context_mark(component, klass)`

New in version 2.0. Get effective context mark of *klass* from *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((4, 8))(staff)
TimeSignatureMark((4, 8))(Staff{4})

abjad> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_effective_context_mark(staff[0], contexttools.TimeSignatureMark)
TimeSignatureMark((4, 8))(Staff{4})

```

Return context mark or none.

contexttools.get_effective_dynamic

`abjad.tools.contexttools.get_effective_dynamic.get_effective_dynamic(component)`

New in version 2.0. Get effective dynamic of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.DynamicMark('f')(staff[0])
DynamicMark('f')(c'8)

abjad> f(staff)
\new Staff {
  c'8 \f
  d'8
  e'8
  f'8
}

abjad> for note in staff:
...     print note, contexttools.get_effective_dynamic(note)
...
c'8 DynamicMark('f')(c'8)
d'8 DynamicMark('f')(c'8)
e'8 DynamicMark('f')(c'8)
f'8 DynamicMark('f')(c'8)

```

Return dynamic mark or none.

contexttools.get_effective_instrument

`abjad.tools.contexttools.get_effective_instrument.get_effective_instrument(component)`

New in version 2.0. Get effective instrument of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

```

```

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  e'8
  f'8
}

abjad> for note in staff:
...     print note, contexttools.get_effective_instrument(note)
...
c'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})
d'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})
e'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})
f'8 InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

```

Return instrument mark or none.

contexttools.get_effective_key_signature

abjad.tools.contexttools.get_effective_key_signature.get_effective_key_signature(*component*)

New in version 2.0. Get effective key signature of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

abjad> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}

abjad> for note in staff:
...     note, contexttools.get_effective_key_signature(note)
...
(Note("c'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}))
(Note("d'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}))
(Note("e'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}))
(Note("f'8"), KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}))

```

Return key signature mark or none.

contexttools.get_effective_staff

abjad.tools.contexttools.get_effective_staff.get_effective_staff(*component*)

New in version 2.0. Get effective staff of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> staff.name = 'First Staff'

```

```

abjad> f(staff)
\context Staff = "First Staff" {
    c'8
    d'8
    e'8
    f'8
}

abjad> for note in staff:
...     print note, contexttools.get_effective_staff(note)
...
c'8 Staff-"First Staff"{4}
d'8 Staff-"First Staff"{4}
e'8 Staff-"First Staff"{4}
f'8 Staff-"First Staff"{4}

```

Return staff or none.

contexttools.get_effective_tempo

`abjad.tools.contexttools.get_effective_tempo.get_effective_tempo(component)`
 New in version 2.0. Get effective tempo of *component*:

```

abjad> score = Score([])
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score.append(staff)
abjad> contexttools.TempoMark(Duration(1, 8), 52)(staff[0])
TempoMark(Duration(1, 8), 52)(c'8)

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

abjad> for note in staff:
...     print note, contexttools.get_effective_tempo(note)
...
c'8 TempoMark(Duration(1, 8), 52)(c'8)
d'8 TempoMark(Duration(1, 8), 52)(c'8)
e'8 TempoMark(Duration(1, 8), 52)(c'8)
f'8 TempoMark(Duration(1, 8), 52)(c'8)

```

Return tempo mark or none.

contexttools.get_effective_time_signature

`abjad.tools.contexttools.get_effective_time_signature.get_effective_time_signature(component)`
 New in version 2.0. Get effective time signature of *component*:


```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((4, 8))(staff)
TimeSignatureMark((4, 8))(Staff{4})

abjad> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}

abjad> for note in staff:
...     note, contexttools.get_effective_time_signature(note)
...
(Note("c'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("d'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("e'8"), TimeSignatureMark((4, 8))(Staff{4}))
(Note("f'8"), TimeSignatureMark((4, 8))(Staff{4}))

```

Return time signature mark or none.

contexttools.get_instrument_mark_attached_to_component

abjad.tools.contexttools.get_instrument_mark_attached_to_component.get_instrument_mark_attached_to_component

New in version 2.1. Get instrument mark attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> violin = contexttools.InstrumentMark('Violin ', 'Vn. ')
abjad> violin.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ')(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_instrument_mark_attached_to_component(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ')(Staff{4})

```

Return instrument mark.

Raise missing mark error when no instrument mark attaches to *component*.

contexttools.get_instrument_marks_attached_to_component

abjad.tools.contexttools.get_instrument_marks_attached_to_component.get_instrument_marks_attached_to_component

New in version 2.3. Get instrument marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.InstrumentMark('Flute', 'Fl.')(staff)
InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_instrument_marks_attached_to_component(staff)
(InstrumentMark(instrument_name='Flute', short_instrument_name='Fl.')(Staff{4}),)

```

Return tuple of zero or more instrument marks.

contexttools.get_key_signature_mark_attached_to_component

abjad.tools.contexttools.get_key_signature_mark_attached_to_component.**get_key_signature_ma**
 New in version 2.3. Get key signature mark attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

abjad> f(staff)
\new Staff {
  \key c \major
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_key_signature_mark_attached_to_component(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

```

Return key signature mark.

Raise missing mark error when no key signature mark attaches to component.

contexttools.get_key_signature_marks_attached_to_component

abjad.tools.contexttools.get_key_signature_marks_attached_to_component.**get_key_signature_ma**
 New in version 2.3. Get key signature marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

abjad> f(staff)
\new Staff {
  \key c \major

```

```

    c'8
    d'8
    e'8
    f'8
}

```

```

abjad> contexttools.get_key_signature_marks_attached_to_component(staff)
(KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4}),)

```

Return tuple of zero or more key signature marks.

contexttools.get_staff_change_mark_attached_to_component

abjad.tools.contexttools.get_staff_change_mark_attached_to_component.get_staff_change_mark

New in version 2.3. Get staff change mark attached to *component*:

```

abjad> piano_staff = scoretools.PianoStaff([])
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'
abjad> piano_staff.extend([rh_staff, lh_staff])
abjad> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

```

```

abjad> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

```

```

abjad> contexttools.get_staff_change_mark_attached_to_component(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

```

Return staff change mark.

Raise missing mark error when no staff change mark attaches to *component*.

contexttools.get_staff_change_marks_attached_to_component

abjad.tools.contexttools.get_staff_change_marks_attached_to_component.get_staff_change_mar

New in version 2.3. Get staff change marks attached to *component*:

```

abjad> piano_staff = scoretools.PianoStaff([])
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'

```

```

abjad> piano_staff.extend([rh_staff, lh_staff])
abjad> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

abjad> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

abjad> contexttools.get_staff_change_marks_attached_to_component(rh_staff[2])
(StaffChangeMark(Staff-"LHStaff"{1})(e'8),)

```

Return tuple of zero or more staff change marks.

contexttools.get_tempo_mark_attached_to_component

abjad.tools.contexttools.get_tempo_mark_attached_to_component.get_tempo_mark_attached_to_component

New in version 2.3. Get tempo mark attached to *component*:

```

abjad> score = Score([])
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score.append(staff)

abjad> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

abjad> contexttools.get_tempo_mark_attached_to_component(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

```

Return tempo mark.

Raise missing mark error when no tempo mark attaches to *component*.

contexttools.get_tempo_marks_attached_to_component

abjad.tools.contexttools.get_tempo_marks_attached_to_component.get_tempo_marks_attached_to_component

New in version 2.3. Get tempo marks attached to *component*:

```

abjad> score = Score([])
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score.append(staff)

abjad> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

abjad> contexttools.get_tempo_marks_attached_to_component(staff)
(TempoMark(Duration(1, 8), 52)(Staff{4}),)

```

Return tuple of zero or more tempo marks.

contexttools.get_time_signature_mark_attached_to_component

`abjad.tools.contexttools.get_time_signature_mark_attached_to_component.get_time_signature_r`
 New in version 2.0. Get time signature mark attached to *component*:

```

abjad> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

abjad> f(measure)
{
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_time_signature_mark_attached_to_component(measure)
TimeSignatureMark((4, 8))(|4/8, c'8, d'8, e'8, f'8|)

```

Return time signature mark.

Raise missing mark error when no time signature mark attaches to *component*.

contexttools.get_time_signature_marks_attached_to_component

`abjad.tools.contexttools.get_time_signature_marks_attached_to_component.get_time_signature_r`
 New in version 2.3. Get time signature marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((2, 4))(staff)
TimeSignatureMark((2, 4))(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \time 2/4
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.get_time_signature_marks_attached_to_component(staff)
(TimeSignatureMark((2, 4))(Staff{4}),)
```

Return tuple of zero or more `time_signature` marks.

`contexttools.is_component_with_clef_mark_attached`

`abjad.tools.contexttools.is_component_with_clef_mark_attached.is_component_with_clef_mark_attached`

New in version 2.3. True when *expr* is a component with clef mark attached:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})

abjad> f(staff)
\new Staff {
  \clef "treble"
  c'8
  d'8
  e'8
  f'8
}

abjad> contexttools.is_component_with_clef_mark_attached(staff)
True
```

False otherwise:

```
abjad> contexttools.is_component_with_clef_mark_attached(staff[0]) False
```

Return boolean.

`contexttools.is_component_with_context_mark_attached`

`abjad.tools.contexttools.is_component_with_context_mark_attached.is_component_with_context_mark_attached`

New in version 2.0. True when *expr* is a component with context mark of *klases* attached:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)
abjad> f(staff)
\new Staff {
  \time 4/8
```

```

c'8
d'8
e'8
f'8
}
abjad> contexttools.is_component_with_context_mark_attached(staff[0])
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_context_mark_attached(staff)
False

```

Return boolean.

contexttools.is_component_with_dynamic_mark_attached

`abjad.tools.contexttools.is_component_with_dynamic_mark_attached.is_component_with_dynamic_mark_attached`
 New in version 2.3. True when *expr* is a component and has a dynamic mark attached:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.DynamicMark('p')(staff[0])
DynamicMark('p')(c'8)

abjad> f(staff)
\new Staff {
  c'8 \p
  d'8
  e'8
  f'8
}

abjad> contexttools.is_component_with_dynamic_mark_attached(staff[0])
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_dynamic_mark_attached(staff)
False

```

Return boolean.

contexttools.is_component_with_instrument_mark_attached

`abjad.tools.contexttools.is_component_with_instrument_mark_attached.is_component_with_instrument_mark_attached`
 New in version 2.3. True when *expr* is a component with instrument mark attached:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> violin = contexttools.InstrumentMark('Violin ', 'Vn. ')
abjad> violin.attach(staff)
InstrumentMark(instrument_name='Violin ', short_instrument_name='Vn. ')(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
}

```

```

    d'8
    e'8
    f'8
}

```

```

abjad> contexttools.is_component_with_instrument_mark_attached(staff)
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_instrument_mark_attached(staff[0])
False

```

Return boolean.

contexttools.is_component_with_key_signature_mark_attached

abjad.tools.contexttools.is_component_with_key_signature_mark_attached.**is_component_with_key_signature_mark_attached**
 New in version 2.3. True when *expr* is a component with key signature mark attached:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.KeySignatureMark('c', 'major')(staff)
KeySignatureMark(NamedChromaticPitchClass('c'), Mode('major'))(Staff{4})

```

```

abjad> f(staff)
\new Staff {
    \key c \major
    c'8
    d'8
    e'8
    f'8
}

```

```

abjad> contexttools.is_component_with_key_signature_mark_attached(staff)
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_key_signature_mark_attached(staff[0])
False

```

Return boolean.

contexttools.is_component_with_staff_change_mark_attached

abjad.tools.contexttools.is_component_with_staff_change_mark_attached.**is_component_with_staff_change_mark_attached**
 New in version 2.3. True when *expr* is a component with staff change mark attached:

```

abjad> piano_staff = scoretools.PianoStaff([])
abjad> rh_staff = Staff("c'8 d'8 e'8 f'8")
abjad> rh_staff.name = 'RHStaff'
abjad> lh_staff = Staff("s2")
abjad> lh_staff.name = 'LHStaff'
abjad> piano_staff.extend([rh_staff, lh_staff])
abjad> contexttools.StaffChangeMark(lh_staff)(rh_staff[2])
StaffChangeMark(Staff-"LHStaff"{1})(e'8)

```



```

abjad> f(piano_staff)
\new PianoStaff <<
  \context Staff = "RHStaff" {
    c'8
    d'8
    \change Staff = LHStaff
    e'8
    f'8
  }
  \context Staff = "LHStaff" {
    s2
  }
>>

```

```

abjad> contexttools.is_component_with_staff_change_mark_attached(rh_staff[2])
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_staff_change_mark_attached(rh_staff)
False

```

Return boolean.

contexttools.is_component_with_tempo_mark_attached

abjad.tools.contexttools.is_component_with_tempo_mark_attached.is_component_with_tempo_mar

New in version 2.3. True when *expr* is a component with tempo mark attached:

```

abjad> score = Score([])
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score.append(staff)

abjad> contexttools.TempoMark(Duration(1, 8), 52)(staff)
TempoMark(Duration(1, 8), 52)(Staff{4})

abjad> f(score)
\new Score <<
  \tempo 8=52
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

abjad> contexttools.is_component_with_tempo_mark_attached(staff)
True

```

Otherwise false:

```

abjad> contexttools.is_component_with_tempo_mark_attached(staff[0])
False

```

Return boolean.

contexttools.is_component_with_time_signature_mark_attached

abjad.tools.contexttools.is_component_with_time_signature_mark_attached.**is_component_with_t**

New in version 2.0. True when *expr* is a component with time signature mark attached:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((4, 8))(staff[0])
TimeSignatureMark((4, 8))(c'8)
```

```
abjad> f(staff)
\new Staff {
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
```

```
abjad> contexttools.is_component_with_time_signature_mark_attached(staff[0])
True
```

Otherwise false:

```
abjad> contexttools.is_component_with_time_signature_mark_attached(staff)
False
```

Return boolean.

contexttools.iterate_contexts_backward_in_expr

abjad.tools.contexttools.iterate_contexts_backward_in_expr.**iterate_contexts_backward_in_expr**

New in version 2.0. Iterate contexts backward in *expr*:

```
abjad> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
abjad> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
abjad> staff.is_parallel = True
```

```
abjad> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
  }
  \new Voice {
    \times 2/3 {
      e'8
      f'8
      g'8
    }
  }
>>
```

```
abjad> for x in contexttools.iterate_contexts_backward_in_expr(staff):
...     x
Staff<<2>>
```

```
Voice{1}
Voice{2}
```

Ignore threads.

Return generator.

contexttools.iterate_contexts_forward_in_expr

abjad.tools.contexttools.iterate_contexts_forward_in_expr.**iterate_contexts_forward_in_expr**

New in version 2.0. Iterate contexts forward in *expr*:

```
abjad> staff = Staff([Voice("c'8 d'8"), Voice("e'8 f'8 g'8")])
abjad> Tuplet(Fraction(2, 3), staff[1][:])
Tuplet(2/3, [e'8, f'8, g'8])
abjad> staff.is_parallel = True

abjad> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
  }
  \new Voice {
    \times 2/3 {
      e'8
      f'8
      g'8
    }
  }
>>

abjad> for x in contexttools.iterate_contexts_forward_in_expr(staff):
...     x
Staff<<2>>
Voice{2}
Voice{1}
```

Ignore threads.

Return generator.

contexttools.list_clef_names

abjad.tools.contexttools.list_clef_names.**list_clef_names()**

New in version 2.8. List clef names:

```
abjad> contexttools.list_clef_names()
['alto', 'baritone', 'bass', 'mezzosoprano', 'percussion', 'soprano', 'treble']
```

Return list of strings.

contexttools.set_accidental_style_on_sequential_contexts_in_expr

`abjad.tools.contexttools.set_accidental_style_on_sequential_contexts_in_expr.set_accidental`

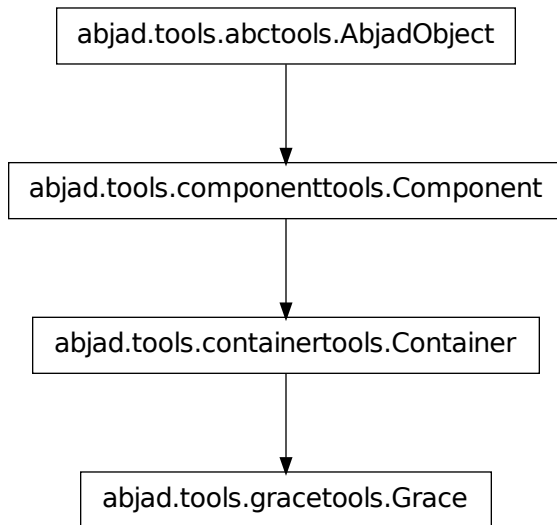
New in version 2.0. Set *accidental_style* for sequential semantic contexts in *expr*:

```
abjad> score = Score(Staff("c'8 d'8") * 2)
abjad> contexttools.set_accidental_style_on_sequential_contexts_in_expr(score, 'forget')

abjad> f(score)
\new Score <<
  \new Staff {
    #(set-accidental-style 'forget)
    c'8
    d'8
  }
  \new Staff {
    #(set-accidental-style 'forget)
    c'8
    d'8
  }
>>
```

Skip nonsemantic contexts.

Function looks like a hack but isn't. LilyPond uses the dedicated command shown here to set accidental style. This means that it is not possible to set accidental style on a top-level context like `score` with a single override.

gracetools**gracetools concrete classes****gracetools.Grace**

class `abjad.tools.gracetools.Grace.Grace`.**Grace** (*music=None, kind='grace', **kwargs*)

Abjad model of grace music:

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(voice)
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}

abjad> grace_notes = [Note("c'16"), Note("d'16")]
abjad> gracetools.Grace(grace_notes, kind = 'grace')(voice[1])
Note("d'8")

abjad> f(voice)
\new Voice {
  c'8 [
  \grace {
    c'16
    d'16
  }
  ]
}

```

```

    }
    d'8
    e'8
    f'8 ]
}

abjad> after_grace_notes = [Note("e'16"), Note("f'16")]
abjad> gracetools.Grace(after_grace_notes, kind = 'after')(voice[1])
Note("d'8")

abjad> f(voice)
\new Voice {
  c'8 [
    \grace {
      c'16
      d'16
    }
    \afterGrace
    d'8
    {
      e'16
      f'16
    }
  }
  e'8
  f'8 ]
}

```

Grace objects are containers you can fill with notes, rests and chords.

Grace containers override the special `__call__` method.

Use `Grace()` to attach grace containers to nongrace notes, rests and chords.

Read-only Properties

`Grace.contents_duration`

Note: Inherited from `containertools.Container`

`Grace.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Grace.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`Grace.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Grace.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Grace.**music**

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Grace.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Grace.**parent**

Note: Inherited from `componenttools.Component`

Grace.**preprolated_duration**

Note: Inherited from `containertools.Container`

Grace.**prolated_duration**

Note: Inherited from `componenttools.Component`

Grace.**prolation**

Note: Inherited from `componenttools.Component`

Grace.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Grace.spanners

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Grace.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
abjad> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
abjad> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True
```

```
abjad> f(container)
```

```
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Grace.kind

Get *kind* of grace container:


```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> gracetools.Grace([Note("cs'16")], kind = 'grace')(staff[1])
Note("d'8")
abjad> grace_container = staff[1].grace
abjad> grace_container.kind
'grace'
```

Return string.

Set *kind* of grace container:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> gracetools.Grace([Note("cs'16")], kind = 'grace')(staff[1])
Note("d'8")
abjad> grace_container = staff[1].grace
abjad> grace_container.kind = 'acciaccatura'
abjad> grace_container.kind
'acciaccatura'
```

Set string.

Valid options include 'after', 'grace', 'acciaccatura', 'appoggiatura'.

Methods

Grace.**append**(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

Grace.**detach**()

Detach grace container from leaf:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> grace_container = gracetools.Grace([Note("cs'16")], kind = 'grace')
abjad> grace_container(staff[1])
Note("d'8")
```

```

abjad> f(staff)
\new Staff {
    c'8
    \grace {
        cs'16
    }
    d'8
    e'8
    f'8
}

abjad> grace_container.detach()
Grace()
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

```

Return grace container.

Grace.**extend**(*expr*)

Extend *expr* against container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Grace.**index**(*component*)

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

```

```
abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Grace.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.insert(1, Note("cs'8"))
```

```
abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Grace.**pop** (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.pop(-1)
Note("e'8")
```

```
abjad> f(container)
{
```

```

        c'8 [
        d'8 ]
    }

```

Return component.

Note: Inherited from `containertools.Container`

Grace.**remove**(*component*)

Remove *component* from container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}

```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

Grace.**__add__**(*expr*)

Concatenate containers self and *expr*. The operation `c = a + b` returns a new Container `c` with the content of both `a` and `b`. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

Grace.**__call__**(*arg*)

Grace.**__contains__**(*expr*)

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

Grace.**__delattr__**()

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Grace.**__delitem__**(*i*)

Find component(s) at index or slice ‘i’ in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

Grace.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Grace.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Grace.**__getitem__**(*i*)

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

Grace.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Grace.**__hash__**() \Leftrightarrow *hash(x)*

Note: Inherited from `__builtin__.object`

Grace.**__iadd__**(*expr*)

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

Grace.**__imul__**(*total*)

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

Grace.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Grace.__len__()

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

Grace.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Grace.__mul__(n)

Note: Inherited from `componenttools.Component`

Grace.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Grace.__radd__(expr)

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

Grace.__repr__()

Grace.__rmul__(n)

Note: Inherited from `componenttools.Component`

Grace.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Grace.__setitem__(i, expr)

Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate

`self[i]` and children of `self[i]`. Replace contents at `self[i]` with `'expr'`. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`Grace.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

gracetools functions

gracetools.all_are_grace_containers

`abjad.tools.gracetools.all_are_grace_containers.expr(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad grace containers:

```
abjad> graces = [gracetools.Grace("<c' e' g'>4"), gracetools.Grace("<c' f' a'>4")]
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> grace_notes = [Note("c'16"), Note("d'16")]
abjad> grace_container = gracetools.Grace(grace_notes, kind = 'grace')
abjad> grace_container(voice[1])
Note("d'8")
```

```
abjad> f(voice)
\new Voice {
  c'8
  \grace {
    c'16
    d'16
  }
  d'8
  e'8
  f'8
}
```

```
abjad> gracetools.all_are_grace_containers([grace_container])
True
```

True when *expr* is an empty sequence:

```
abjad> gracetools.all_are_grace_containers([])
True
```

Otherwise false:

```
abjad> gracetools.all_are_grace_containers('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

gracetools.detach_grace_containers_attached_to_leaf

abjad.tools.gracetools.detach_grace_containers_attached_to_leaf.**detach_grace_containers_attached_to_leaf**

New in version 2.0. Detach grace containers attached to *leaf*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> grace_container = gracetools.Grace([Note("cs'16")], kind = 'grace')
abjad> grace_container(staff[1])
Note("d'8")
```

```
abjad> f(staff)
\new Staff {
  c'8
  \grace {
    cs'16
  }
  d'8
  e'8
  f'8
}
```

```
abjad> gracetools.get_grace_containers_attached_to_leaf(staff[1])
(Grace(cs'16),)
```

```
abjad> gracetools.detach_grace_containers_attached_to_leaf(staff[1])
(Grace(),)
```

```
abjad> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}
```

```
abjad> gracetools.get_grace_containers_attached_to_leaf(staff[1])
()
```

Return tuple.

gracetools.get_grace_containers_attached_to_leaf

abjad.tools.gracetools.get_grace_containers_attached_to_leaf.**get_grace_containers_attached_to_leaf**

New in version 2.0. Get grace containers attached to leaf:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> gracetools.Grace([Note("cs'16")], kind = 'grace')(staff[1])
Note("d'8")
abjad> gracetools.Grace([Note("ds'16")], kind = 'after')(staff[1])
Note("d'8")
```

```
abjad> f(staff)
\new Staff {
  c'8
  \grace {
    cs'16
  }
}
```



```

    \afterGrace
    d'8
    {
        ds'16
    }
    e'8
    f'8
}

```

```

abjad> gracetools.get_grace_containers_attached_to_leaf(staff[1])
(Grace(cs'16), Grace(ds'16))

```

Return tuple.

gracetools.iterate_components_and_grace_containers_forward_in_expr

abjad.tools.gracetools.iterate_components_and_grace_containers_forward_in_expr.**iterate_comp**

Iterate components of *klass* forward in *expr*:

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(voice[:])
BeamSpanner(c'8, d'8, e'8, f'8)

```

```

abjad> grace_notes = [Note("c'16"), Note("d'16")]
abjad> gracetools.Grace(grace_notes, kind = 'grace')(voice[1])
Note("d'8")

```

```

abjad> after_grace_notes = [Note("e'16"), Note("f'16")]
abjad> gracetools.Grace(after_grace_notes, kind = 'after')(voice[1])
Note("d'8")

```

```

abjad> f(voice)
\new Voice {
  c'8 [
    \grace {
      c'16
      d'16
    }
    \afterGrace
    d'8
    {
      e'16
      f'16
    }
  ]
  e'8
  f'8 ]
}

```

```

abjad> for note in gracetools.iterate_components_and_grace_containers_forward_in_expr(voice, Not
...     note
...
Note("c'8")
Note("c'16")
Note("d'16")
Note("d'8")
Note("e'16")

```

```
Note("f'16")
Note("e'8")
Note("f'8")
```

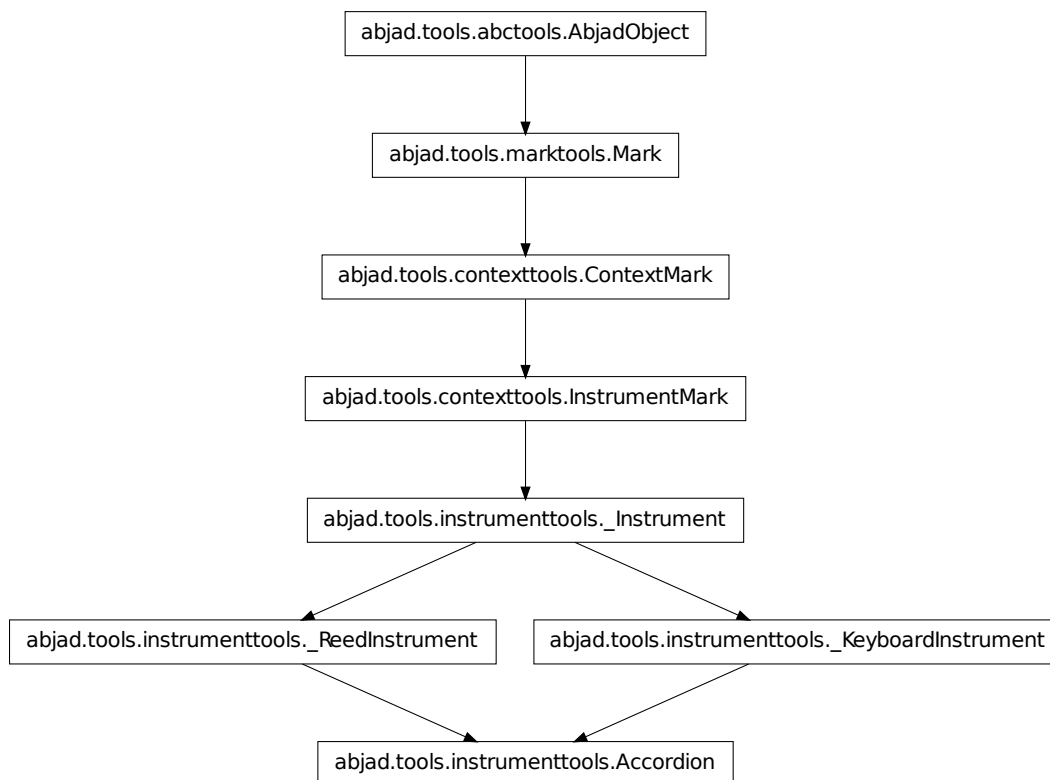
Include grace leaves before main leaves.

Include grace leaves after main leaves. Changed in version 2.0: renamed `iterate.grace()` to `componenttools.iterate_components_and_grace_containers_forward_in_expr()`.

instrumenttools

instrumenttools concrete classes

instrumenttools.Accordion



```
class abjad.tools.instrumenttools.Accordion.Accordion(Accordion(target_context=None,
                                                             **kwargs))
```

Abjad model of the accordion:

```
abjad> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])

abjad> instrumenttools.Accordion()(piano_staff)
Accordion()(PianoStaff<<2>>)
```

```
abjad> f(piano_staff)
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Accordion }
  \set PianoStaff.shortInstrumentName = \markup { Acc. }
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'4
    b4
  }
>>
```

The accordion targets piano staff context by default.

Read-only Properties

Accordion.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Accordion.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Accordion.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Accordion.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contextttools.InstrumentMark`

Accordion.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Accordion.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Accordion.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Accordion.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Accordion.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Accordion.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contextttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contextttools.ContextMark`

Accordion.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**Accordion.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

Accordion.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Accordion.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Accordion.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Accordion.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

`Accordion.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Accordion.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Accordion.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`Accordion.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`Accordion.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Accordion.get_default_performer_name (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

Accordion.get_performer_names ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

Accordion.__call__ (*args)

Note: Inherited from `marktools.Mark`

Accordion.__delattr__ (*args)

Note: Inherited from `marktools.Mark`

Accordion.__eq__ (arg)

Note: Inherited from `contexttools.InstrumentMark`

Accordion.__ge__ (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Accordion.__gt__ (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Accordion.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`Accordion.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Accordion.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Accordion.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`Accordion.__repr__()`

Note: Inherited from `marktools.Mark`

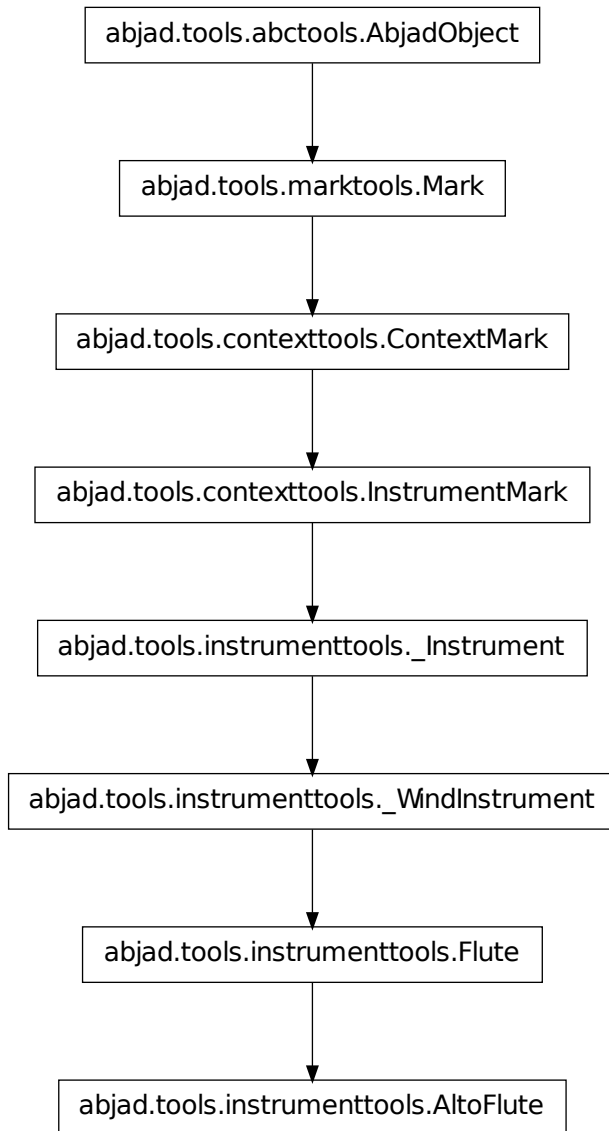
`Accordion.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Accordion.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Alt flute

class `abjad.tools.instrumenttools.Alt flute.Alt flute.Alt flute(**kwargs)`

Abjad model of the alto flute:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Alt flute()(staff)
Alt flute()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Alto flute }
  \set Staff.shortInstrumentName = \markup { Alt. fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The alto flute targets staff context by default.

Read-only Properties

AltoFlute.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

AltoFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

AltoFlute.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

AltoFlute.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

AltoFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`AltoFlute.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`AltoFlute.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`AltoFlute.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoFlute.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`AltoFlute.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoFlute.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`AltoFlute.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`AltoFlute.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`AltoFlute.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`AltoFlute.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`AltoFlute.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`AltoFlute.__call__` (*args)

Note: Inherited from `marktools.Mark`

`AltoFlute.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`AltoFlute.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`AltoFlute.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AltoFlute.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

AltoFlute.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

AltoFlute.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AltoFlute.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AltoFlute.__ne__(arg)

Note: Inherited from `marktools.Mark`

AltoFlute.__repr__()

Note: Inherited from `marktools.Mark`

AltoFlute.__setattr__()

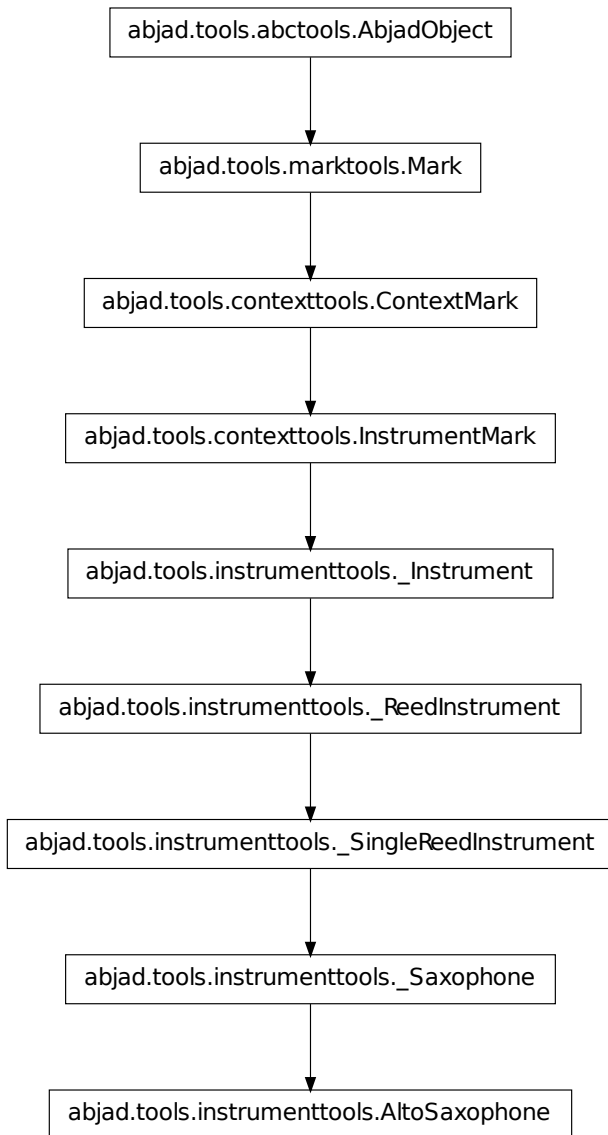
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

AltoFlute.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.AltoSaxophone



class `abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone`.**AltoSaxophone** (***kwargs*)

New in version 2.6. Abjad model of the alto saxophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.AltoSaxophone()(staff)
AltoSaxophone()(Staff{4})
  
```



```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Alto saxophone }
  \set Staff.shortInstrumentName = \markup { Alto sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The alto saxophone is pitched in E-flat.

The alto saxophone targets staff context by default.

Read-only Properties

`AltoSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`AltoSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`AltoSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`AltoSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`AltoSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**AltoSaxophone.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

AltoSaxophone.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

AltoSaxophone.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

AltoSaxophone.pitch_range

Note: Inherited from `instrumenttools._Instrument`

AltoSaxophone.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

AltoSaxophone.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`AltoSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`AltoSaxophone.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`AltoSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`AltoSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`AltoSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`AltoSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`AltoSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AltoSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`AltoSaxophone.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`AltoSaxophone.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AltoSaxophone.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AltoSaxophone.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`AltoSaxophone.__repr__()`

Note: Inherited from `marktools.Mark`

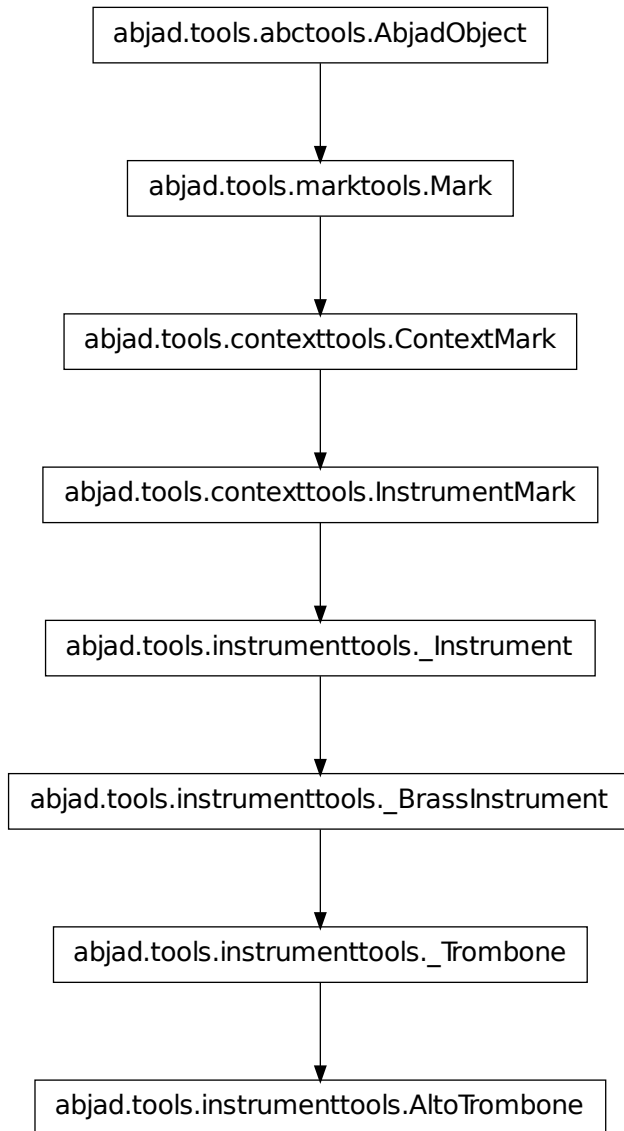
`AltoSaxophone.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`AltoSaxophone.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.AltiTrombone

class `abjad.tools.instrumenttools.AltiTrombone.AltiTrombone`. **AltiTrombone** (**kwargs)

New in version 2.0. Abjad model of the alto trombone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.AltiTrombone()(staff)
AltiTrombone()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Alto trombone }
  \set Staff.shortInstrumentName = \markup { Alt. trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

`AltoTrombone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`AltoTrombone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`AltoTrombone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoTrombone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`AltoTrombone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`AltoTrombone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`AltoTrombone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`AltoTrombone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

AltoTrombone.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

AltoTrombone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

AltoTrombone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

AltoTrombone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

AltoTrombone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

AltoTrombone.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`AltoTrombone.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`AltoTrombone.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`AltoTrombone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`AltoTrombone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`AltoTrombone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`AltoTrombone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`AltoTrombone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`AltoTrombone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AltoTrombone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

AltoTrombone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

AltoTrombone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AltoTrombone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AltoTrombone.__ne__(arg)

Note: Inherited from `marktools.Mark`

AltoTrombone.__repr__()

Note: Inherited from `marktools.Mark`

AltoTrombone.__setattr__()

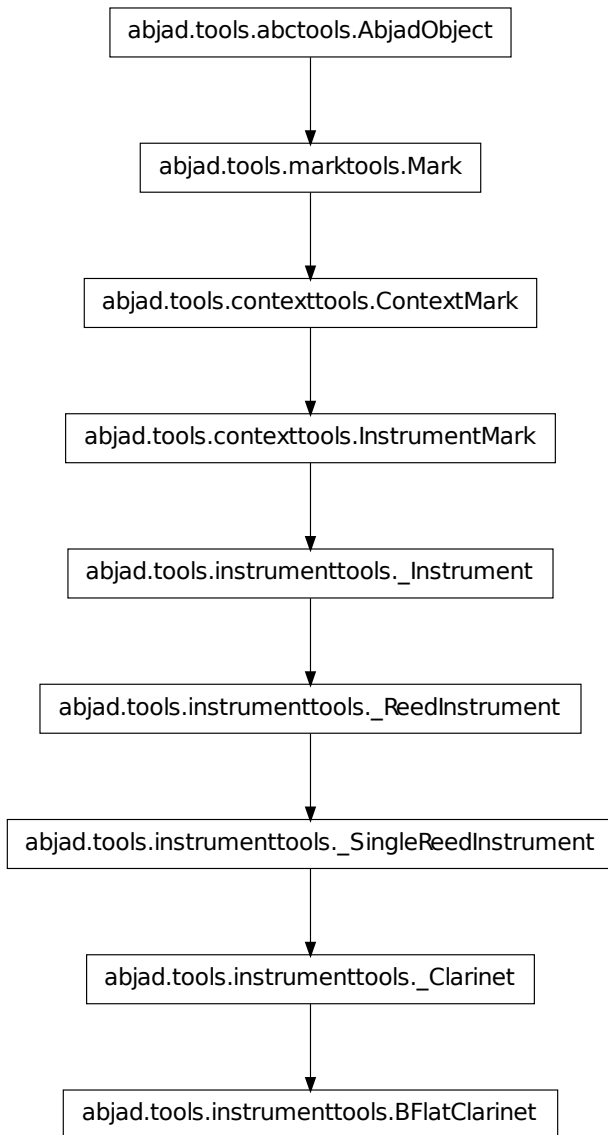
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

AltoTrombone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BFlatClarinet



class `abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet(**kwargs)`

New in version 2.0. Abjad model of the B-flat clarinet:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.BFlatClarinet()(staff)
BFlatClarinet()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  c'8
  d'8
  e'8
  f'8
}
```

The B-flat clarinet targets staff context by default.

Read-only Properties

BFlatClarinet.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BFlatClarinet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BFlatClarinet.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BFlatClarinet.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

BFlatClarinet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BFlatClarinet.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BFlatClarinet.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BFlatClarinet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BFlatClarinet.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BFlatClarinet.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BFlatClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**BFlatClarinet.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

BFlatClarinet.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BFlatClarinet.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BFlatClarinet.pitch_range

Note: Inherited from `instrumenttools._Instrument`

BFlatClarinet.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

BFlatClarinet.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BFlatClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BFlatClarinet.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BFlatClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BFlatClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BFlatClarinet.__call__(*args)`

Note: Inherited from `marktools.Mark`

`BFlatClarinet.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`BFlatClarinet.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BFlatClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BFlatClarinet.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`BFlatClarinet.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BFlatClarinet.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BFlatClarinet.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BFlatClarinet.__repr__()`

Note: Inherited from `marktools.Mark`

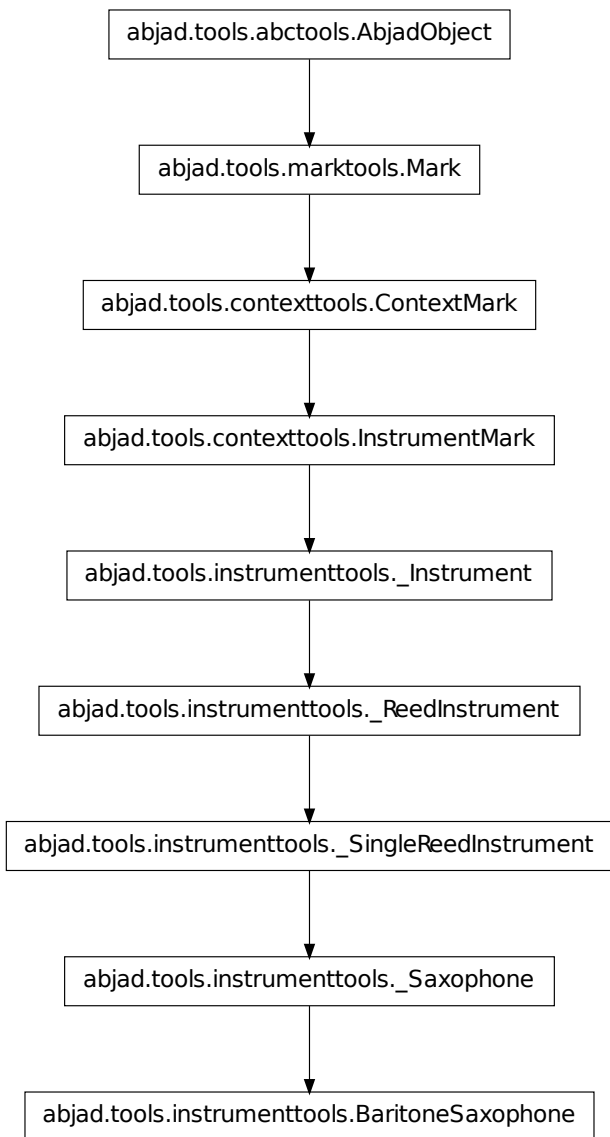
`BFlatClarinet.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BFlatClarinet.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BaritoneSaxophone

class `abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone`.**BaritoneSaxophone** ***kwargs*

New in version 2.6. Abjad model of the baritone saxophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.BaritoneSaxophone()(staff)
BaritoneSaxophone()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Baritone saxophone }
  \set Staff.shortInstrumentName = \markup { Bar. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The baritone saxophone is pitched in E-flat.

The baritone saxophone targets staff context by default.

Read-only Properties

`BaritoneSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BaritoneSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

BaritoneSaxophone.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

BaritoneSaxophone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

BaritoneSaxophone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

BaritoneSaxophone.**short_instrument_name**

Get short instrument name:


```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BaritoneSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BaritoneSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BaritoneSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`BaritoneSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`BaritoneSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BaritoneSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

BaritoneSaxophone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

BaritoneSaxophone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BaritoneSaxophone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BaritoneSaxophone.__ne__(arg)

Note: Inherited from `marktools.Mark`

BaritoneSaxophone.__repr__()

Note: Inherited from `marktools.Mark`

BaritoneSaxophone.__setattr__()

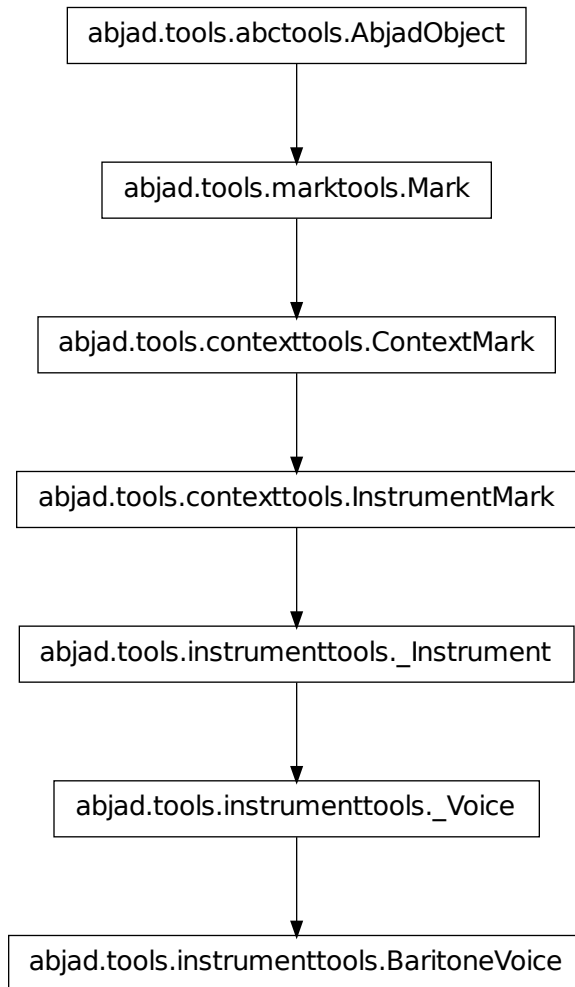
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

BaritoneSaxophone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BaritoneVoice



class `abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice(**kwargs)`
 New in version 2.8. Abjad model of the baritone voice:

```

abjad> staff = Staff("c8 d8 e8 f8")

abjad> instrumenttools.BaritoneVoice()(staff)
BaritoneVoice()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Baritone voice }
  \set Staff.shortInstrumentName = \markup { Baritone }
  c8
  d8

```

```

        e8
        f8
    }

```

The baritone voice targets staff context by default.

Read-only Properties

`BaritoneVoice.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contextttools.InstrumentMark`

`BaritoneVoice.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contextttools.InstrumentMark`

`BaritoneVoice.effective_context`

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contextttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contextttools.ContextMark`

`BaritoneVoice.format`

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contextttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Note: Inherited from `contextttools.InstrumentMark`

`BaritoneVoice.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BaritoneVoice.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BaritoneVoice.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`BaritoneVoice.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BaritoneVoice.attach` (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BaritoneVoice.detach` ()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BaritoneVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BaritoneVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BaritoneVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`BaritoneVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`BaritoneVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BaritoneVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BaritoneVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`BaritoneVoice.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BaritoneVoice.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BaritoneVoice.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BaritoneVoice.__repr__()`

Note: Inherited from `marktools.Mark`

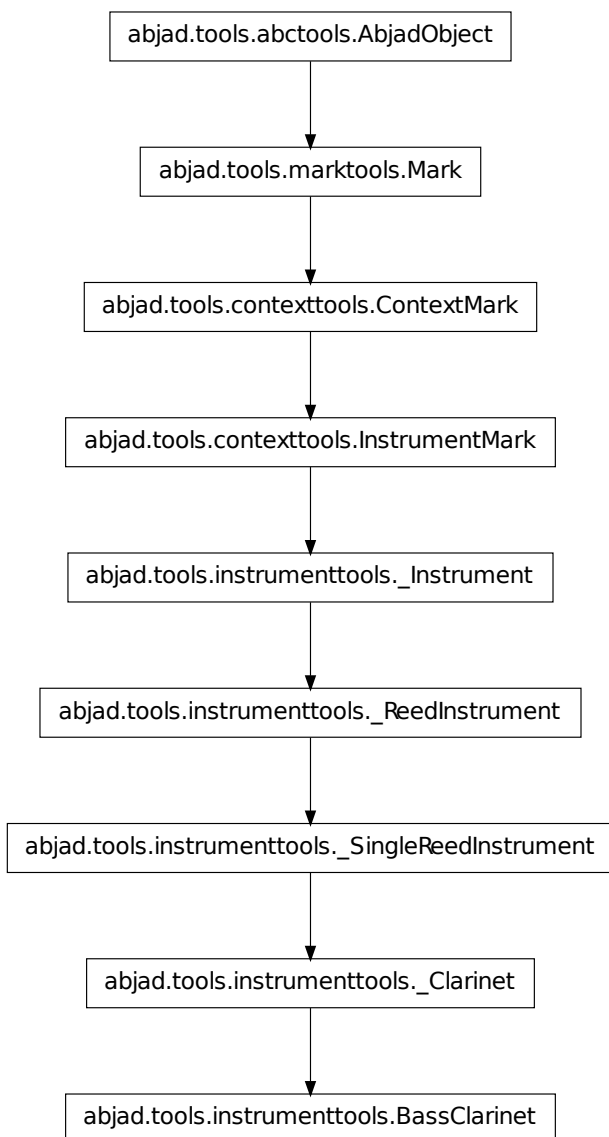
`BaritoneVoice.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BaritoneVoice.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BassClarinet

class `abjad.tools.instrumenttools.BassClarinet.BassClarinet` **BassClarinet** (***kwargs*)
 New in version 2.0. Abjad model of the bass clarinet:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.BassClarinet()(staff)
BassClarinet()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass clarinet }
  \set Staff.shortInstrumentName = \markup { Bass cl. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass clarinet targets staff context by default.

Read-only Properties

BassClarinet.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassClarinet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassClarinet.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BassClarinet.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

BassClarinet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BassClarinet.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BassClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`BassClarinet.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassClarinet.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BassClarinet.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassClarinet.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BassClarinet.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BassClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BassClarinet.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BassClarinet.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BassClarinet.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BassClarinet.__call__` (*args)

Note: Inherited from `marktools.Mark`

`BassClarinet.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`BassClarinet.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`BassClarinet.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassClarinet.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

BassClarinet.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

BassClarinet.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassClarinet.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassClarinet.__ne__(arg)

Note: Inherited from `marktools.Mark`

BassClarinet.__repr__()

Note: Inherited from `marktools.Mark`

BassClarinet.__setattr__()

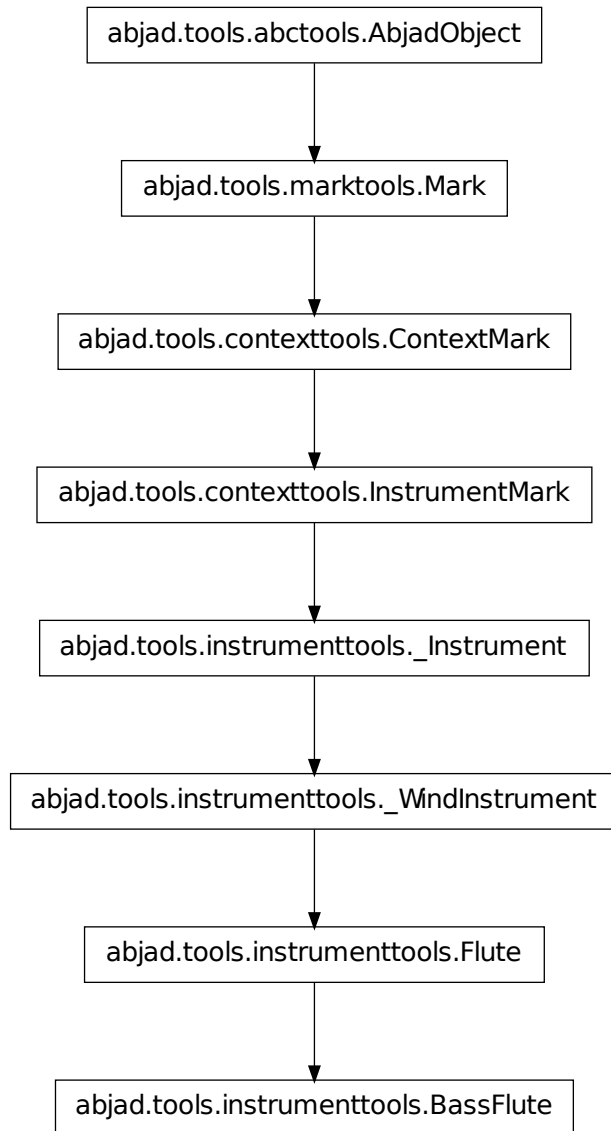
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

BassClarinet.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BassFlute



class `abjad.tools.instrumenttools.BassFlute.BassFlute`.**BassFlute** *(**kwargs)*

New in version 2.0. Abjad model of the bass flute:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.BassFlute()(staff)
BassFlute()(Staff{4})
  
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass flute }
  \set Staff.shortInstrumentName = \markup { Bass fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass flute targets staff context by default.

Read-only Properties

BassFlute.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BassFlute.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BassFlute.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassFlute.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassFlute.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BassFlute.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BassFlute.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BassFlute.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**BassFlute.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

BassFlute.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.pitch_range

Note: Inherited from `instrumenttools._Instrument`

BassFlute.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

BassFlute.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BassFlute.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods

BassFlute.attach(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

BassFlute.detach()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BassFlute.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BassFlute.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BassFlute.__call__` (*args)

Note: Inherited from `marktools.Mark`

`BassFlute.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`BassFlute.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`BassFlute.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassFlute.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BassFlute.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`BassFlute.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassFlute.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassFlute.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BassFlute.__repr__()`

Note: Inherited from `marktools.Mark`

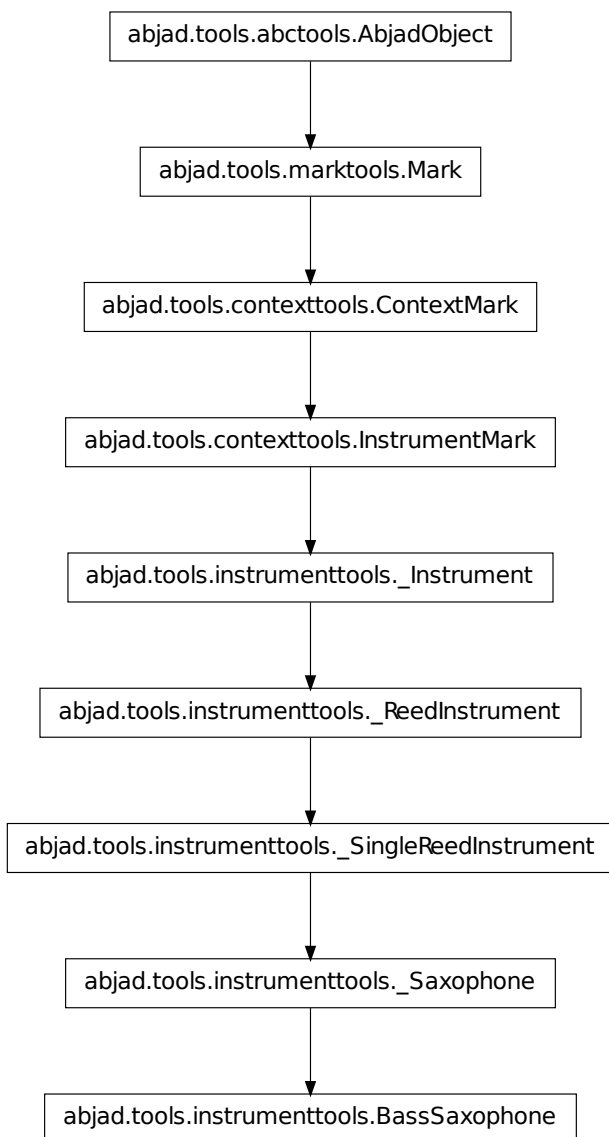
`BassFlute.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BassFlute.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BassSaxophone

class `abjad.tools.instrumenttools.BassSaxophone.BassSaxophone` **BassSaxophone** (***kwargs*)
 New in version 2.6. Abjad model of the bass saxophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.BassSaxophone()(staff)
BassSaxophone()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass saxophone }
  \set Staff.shortInstrumentName = \markup { Bass sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The bass saxophone is pitched in B-flat.

The bass saxophone targets staff context by default.

Read-only Properties

`BassSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BassSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BassSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BassSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BassSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BassSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

BassSaxophone.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

BassSaxophone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassSaxophone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BassSaxophone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

BassSaxophone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

BassSaxophone.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BassSaxophone.attach` (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BassSaxophone.detach` ()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BassSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BassSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BassSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`BassSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`BassSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`BassSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

BassSaxophone. **__hash__** ()

Note: Inherited from `contexttools.InstrumentMark`

BassSaxophone. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassSaxophone. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassSaxophone. **__ne__** (*arg*)

Note: Inherited from `marktools.Mark`

BassSaxophone. **__repr__** ()

Note: Inherited from `marktools.Mark`

BassSaxophone. **__setattr__** ()

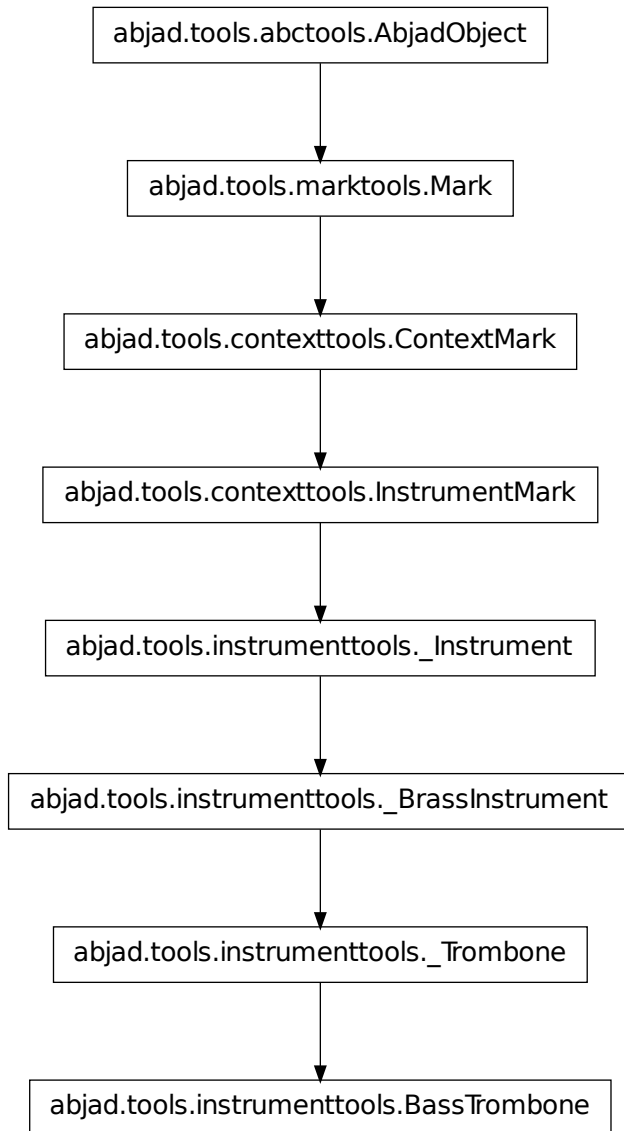
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

BassSaxophone. **__str__** () <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BassTrombone



class `abjad.tools.instrumenttools.BassTrombone.BassTrombone` **BassTrombone** *(**kwargs)*

New in version 2.0. Abjad model of the tenor trombone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

```

```

abjad> instrumenttools.BassTrombone()(staff)
BassTrombone()(Staff{4})

```



```
abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Bass trombone }
  \set Staff.shortInstrumentName = \markup { Bass trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

BassTrombone.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BassTrombone.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`BassTrombone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassTrombone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`BassTrombone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`BassTrombone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`BassTrombone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`BassTrombone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**BassTrombone.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

BassTrombone.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.pitch_range

Note: Inherited from `instrumenttools._Instrument`

BassTrombone.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

BassTrombone.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods

BassTrombone.attach(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

BassTrombone.detach()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BassTrombone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BassTrombone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BassTrombone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`BassTrombone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`BassTrombone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`BassTrombone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassTrombone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

BassTrombone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

BassTrombone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassTrombone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

BassTrombone.__ne__(arg)

Note: Inherited from `marktools.Mark`

BassTrombone.__repr__()

Note: Inherited from `marktools.Mark`

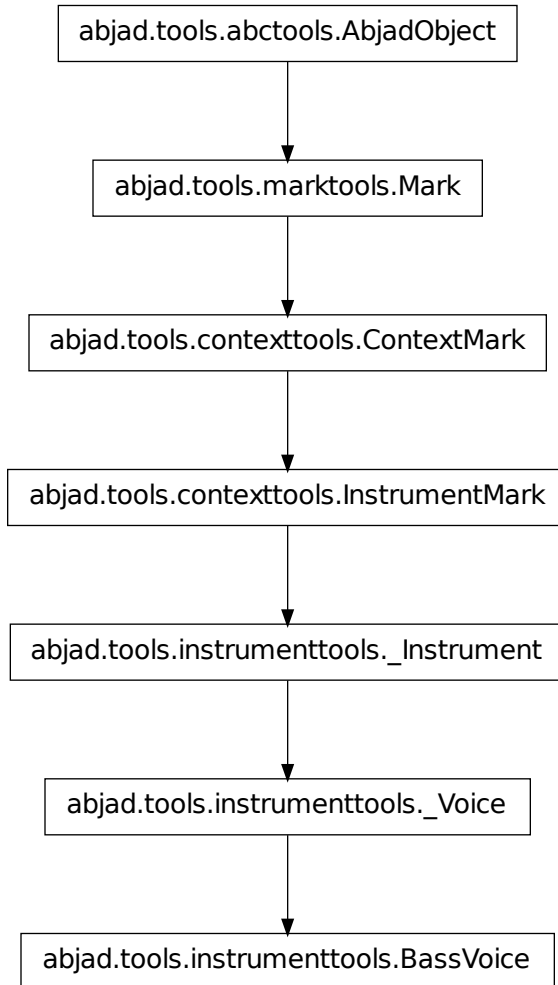
BassTrombone.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

BassTrombone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.BassVoice

class `abjad.tools.instrumenttools.BassVoice.BassVoice`.**BassVoice** *(**kwargs)*

New in version 2.8. Abjad model of the bass voice:

```
abjad> staff = Staff("c8 d8 e8 f8")
```

```
abjad> instrumenttools.BassVoice()(staff)
BassVoice()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Bass voice }
  \set Staff.shortInstrumentName = \markup { Bass }
  c8
  d8
```

```

        e8
        f8
    }

```

The bass voice targets staff context by default.

Read-only Properties

BassVoice.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

BassVoice.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BassVoice.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

BassVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

BassVoice.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

BassVoice.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

BassVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

BassVoice.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

BassVoice.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

BassVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

BassVoice.all_clefs

Note: Inherited from `instrumenttools._Instrument`

`BassVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`BassVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`BassVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`BassVoice.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`BassVoice.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`BassVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`BassVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`BassVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`BassVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`BassVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BassVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`BassVoice.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassVoice.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BassVoice.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BassVoice.__repr__()`

Note: Inherited from `marktools.Mark`

`BassVoice.__setattr__()`

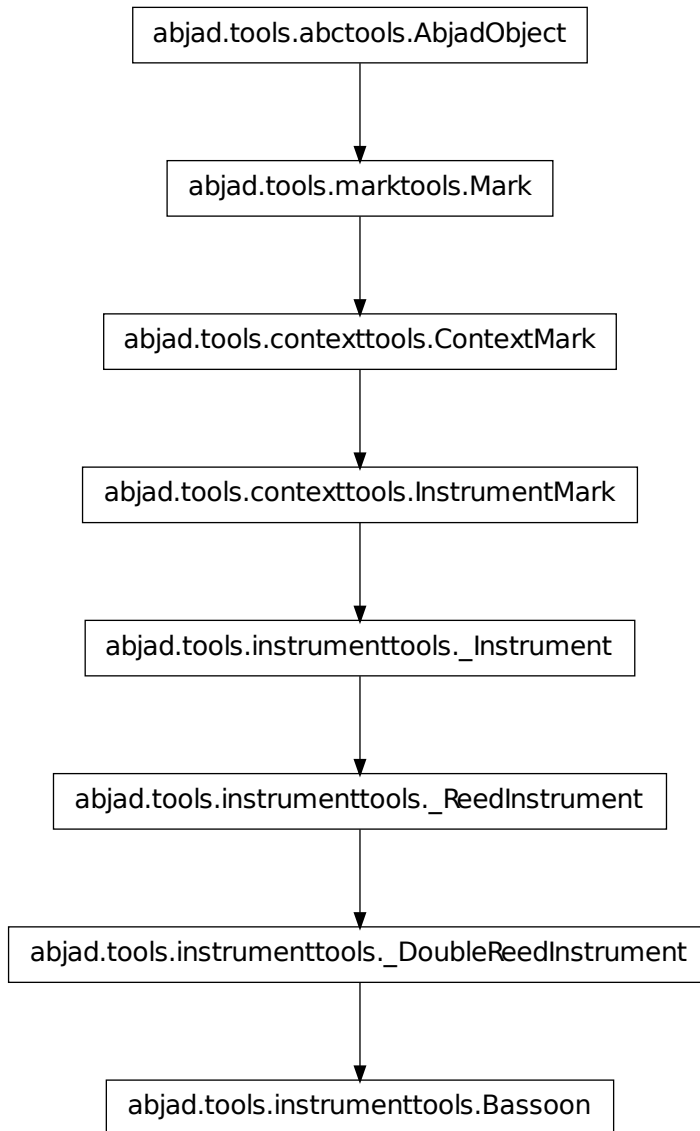
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BassVoice.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Bassoon



class `abjad.tools.instrumenttools.Bassoon.Bassoon`.**Bassoon** (**kwargs)

New in version 2.0. Abjad model of the bassoon:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

```

```

abjad> instrumenttools.Bassoon()(staff)
Bassoon()(Staff{4})

```

```

abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Bassoon }
  \set Staff.shortInstrumentName = \markup { Bsn. }
  c'8
  d'8
  e'8
  f'8
}

```

The bassoon targets staff context by default.

Read-only Properties

Bassoon.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Bassoon.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.'

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Bassoon.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Bassoon.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Bassoon.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Bassoon.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Bassoon.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Bassoon.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**Bassoon.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

Bassoon.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Bassoon.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Bassoon.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods

Bassoon.attach (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Bassoon.detach ()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Bassoon.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Bassoon.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Bassoon.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Bassoon.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Bassoon.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Bassoon.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Bassoon.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Bassoon.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

Bassoon.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Bassoon.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Bassoon.__ne__(arg)

Note: Inherited from `marktools.Mark`

Bassoon.__repr__()

Note: Inherited from `marktools.Mark`

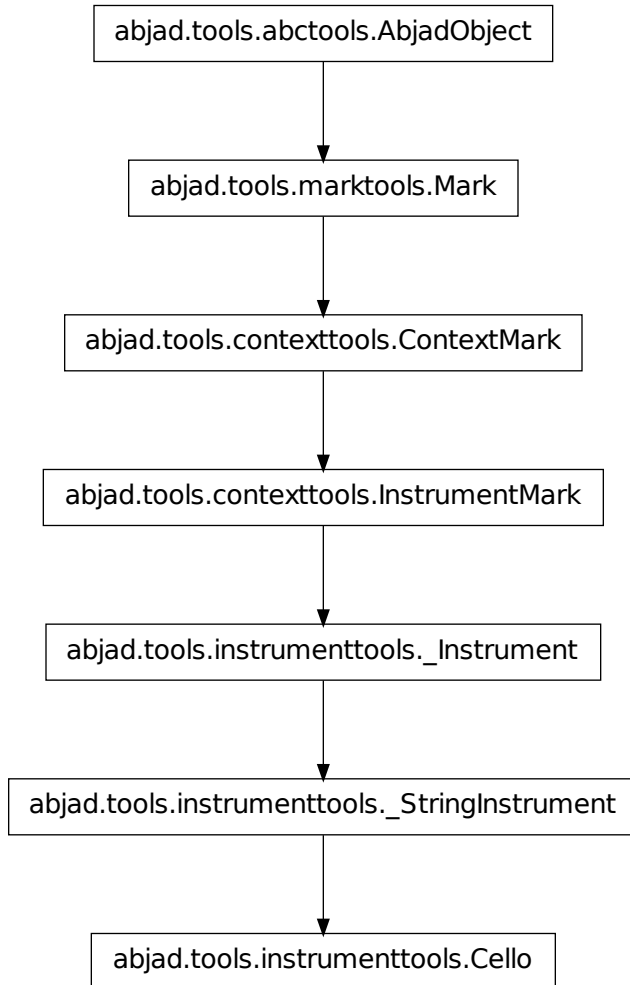
Bassoon.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Bassoon.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Cello

class `abjad.tools.instrumenttools.Cello.Cello.Cello` **Cello** *(**kwargs)*
 New in version 2.0. Abjad model of the cello:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.Cello()(staff)
Cello()(Staff{4})

abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Cello }

```

```

        \set Staff.shortInstrumentName = \markup { Vc. }
        c' 8
        d' 8
        e' 8
        f' 8
    }

```

The cello targets staff context by default.

Read-only Properties

Cello.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Cello.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Cello.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Cello.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Cello.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Cello.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Cello.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Cello.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Cello.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Cello.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Cello.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Cello.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Cello.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Cello.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Cello.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Cello.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Cello.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```


Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contextttools.InstrumentMark`

Cello.`short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contextttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contextttools.InstrumentMark`

Cello.`sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumentttools._Instrument`

Methods

Cello.`attach`(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contextttools.ContextMark`

Cello.`detach`()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contextttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Cello.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Cello.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Cello.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Cello.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Cello.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Cello.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Cello.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Cello.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Cello.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Cello.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Cello.__ne__(arg)

Note: Inherited from `marktools.Mark`

Cello.__repr__()

Note: Inherited from `marktools.Mark`

Cello.__setattr__()

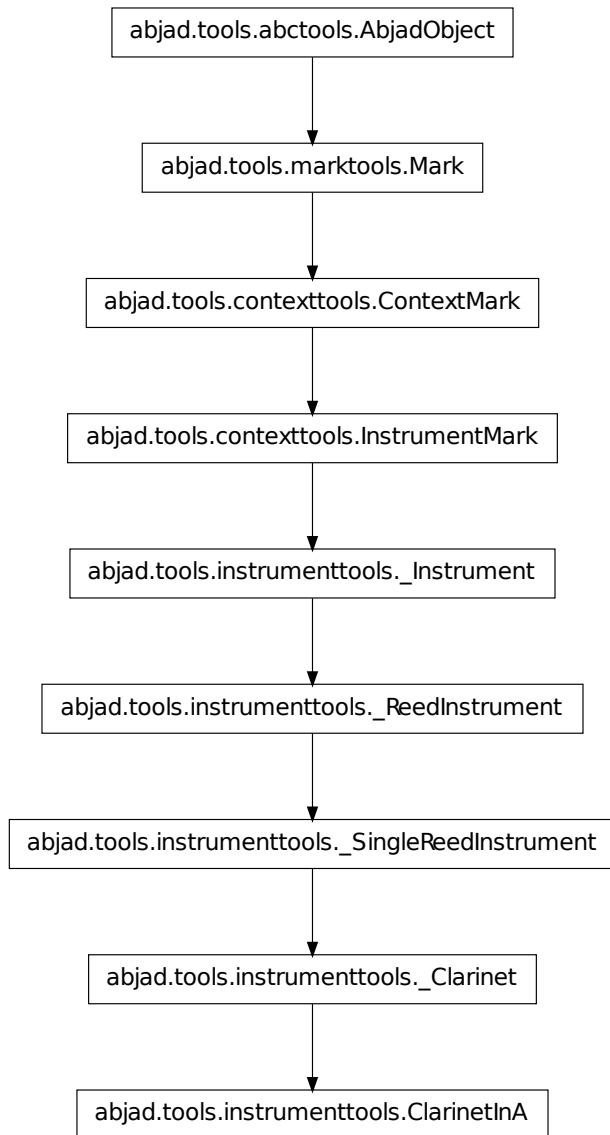
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Cello.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.ClarinetInA



class `abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA(**kwargs)`

New in version 2.6. Abjad model of the clarinet in A:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.ClarinetInA()(staff)
ClarinetInA()(Staff{4})
  
```

```

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in A }
  \set Staff.shortInstrumentName = \markup { Cl. A \natural }
  c'8
  d'8
  e'8
  f'8
}

```

The clarinet in A targets staff context by default.

Read-only Properties

ClarinetInA.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

ClarinetInA.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`ClarinetInA.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ClarinetInA.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ClarinetInA.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`ClarinetInA.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ClarinetInA.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ClarinetInA.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**ClarinetInA.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

ClarinetInA.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.pitch_range

Note: Inherited from `instrumenttools._Instrument`

ClarinetInA.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

ClarinetInA.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

ClarinetInA.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

ClarinetInA.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```



```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`ClarinetInA.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`ClarinetInA.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`ClarinetInA.__call__(*args)`

Note: Inherited from `marktools.Mark`

`ClarinetInA.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`ClarinetInA.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`ClarinetInA.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClarinetInA.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

ClarinetInA.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

ClarinetInA.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ClarinetInA.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ClarinetInA.__ne__(arg)

Note: Inherited from `marktools.Mark`

ClarinetInA.__repr__()

Note: Inherited from `marktools.Mark`

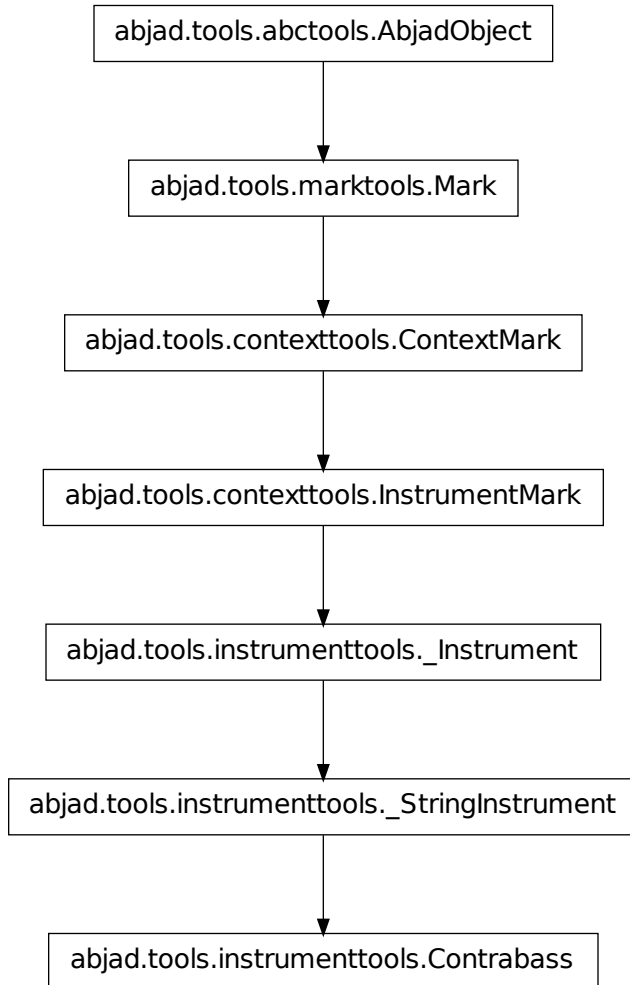
ClarinetInA.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

ClarinetInA.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Contrabass

class `abjad.tools.instrumenttools.Contrabass.Contrabass` **Contrabass** *(**kwargs)*
 New in version 2.0. Abjad model of the contrabass:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.Contrabass()(staff)
Contrabass()(Staff{4})

abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Contrabass }

```

```

    \set Staff.shortInstrumentName = \markup { Vb. }
    c' 8
    d' 8
    e' 8
    f' 8
}

```

The contrabass targets staff context by default.

Read-only Properties

Contrabass.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabass.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabass.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Contrabass.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Contrabass.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`Contrabass.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Contrabass.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Contrabass.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Contrabass.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Contrabass.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Contrabass.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Contrabass.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Contrabass.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabass.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Contrabass.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Contrabass.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Contrabass.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Contrabass.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Contrabass.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`Contrabass.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`Contrabass.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Contrabass.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Contrabass.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Contrabass.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Contrabass.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Contrabass.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Contrabass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Contrabass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Contrabass.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`Contrabass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Contrabass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Contrabass.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`Contrabass.__repr__()`

Note: Inherited from `marktools.Mark`

`Contrabass.__setattr__()`

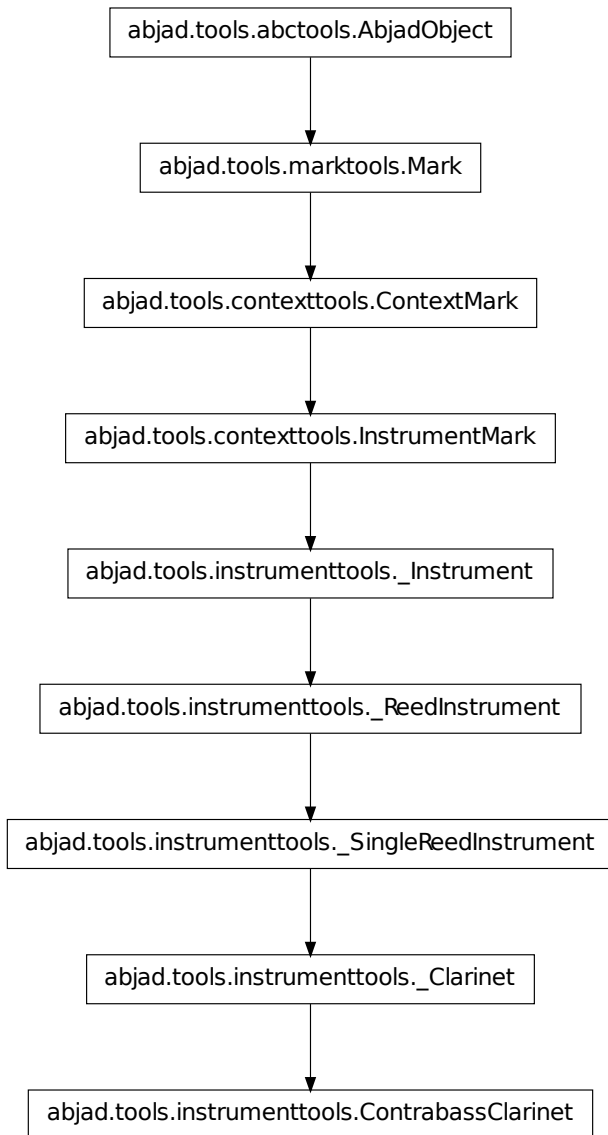
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Contrabass.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.ContrabassClarinet



```

class abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet (**)
    New in version 2.6. Abjad model of the contrabass clarinet:

    abjad> staff = Staff("c'8 d'8 e'8 f'8")

    abjad> instrumenttools.ContrabassClarinet()(staff)
    ContrabassClarinet()(Staff{4})
  
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass clarinet }
  \set Staff.shortInstrumentName = \markup { Cbass cl. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass clarinet targets staff context by default.

Read-only Properties

`ContrabassClarinet.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContrabassClarinet.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ContrabassClarinet.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContrabassClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**ContrabassClarinet.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

ContrabassClarinet.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassClarinet.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassClarinet.pitch_range

Note: Inherited from `instrumenttools._Instrument`

ContrabassClarinet.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

ContrabassClarinet.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`ContrabassClarinet.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`ContrabassClarinet.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`ContrabassClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassClarinet.__call__(*args)`

Note: Inherited from `marktools.Mark`

`ContrabassClarinet.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`ContrabassClarinet.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassClarinet.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassClarinet.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`ContrabassClarinet.__repr__()`

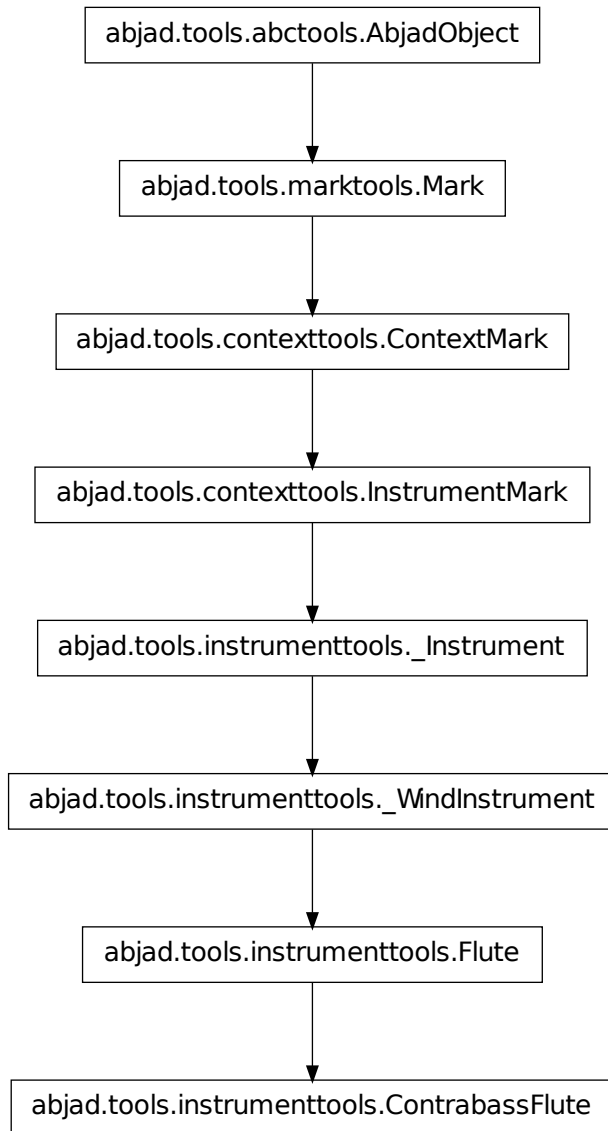
Note: Inherited from `marktools.Mark`

`ContrabassClarinet.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ContrabassClarinet.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.ContrabassFlute

class `abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute(**kwargs)`
 New in version 2.0. Abjad model of the contrabass flute:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.ContrabassFlute()(staff)
ContrabassFlute()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass flute }
  \set Staff.shortInstrumentName = \markup { Cbass. fl. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass flute targets staff context by default.

Read-only Properties

ContrabassFlute.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassFlute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassFlute.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

ContrabassFlute.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassFlute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ContrabassFlute.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContrabassFlute.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`ContrabassFlute.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`ContrabassFlute.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`ContrabassFlute.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`ContrabassFlute.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassFlute.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassFlute.__call__` (*args)

Note: Inherited from `marktools.Mark`

`ContrabassFlute.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`ContrabassFlute.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassFlute.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassFlute.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

ContrabassFlute.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

ContrabassFlute.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ContrabassFlute.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ContrabassFlute.__ne__(arg)

Note: Inherited from `marktools.Mark`

ContrabassFlute.__repr__()

Note: Inherited from `marktools.Mark`

ContrabassFlute.__setattr__()

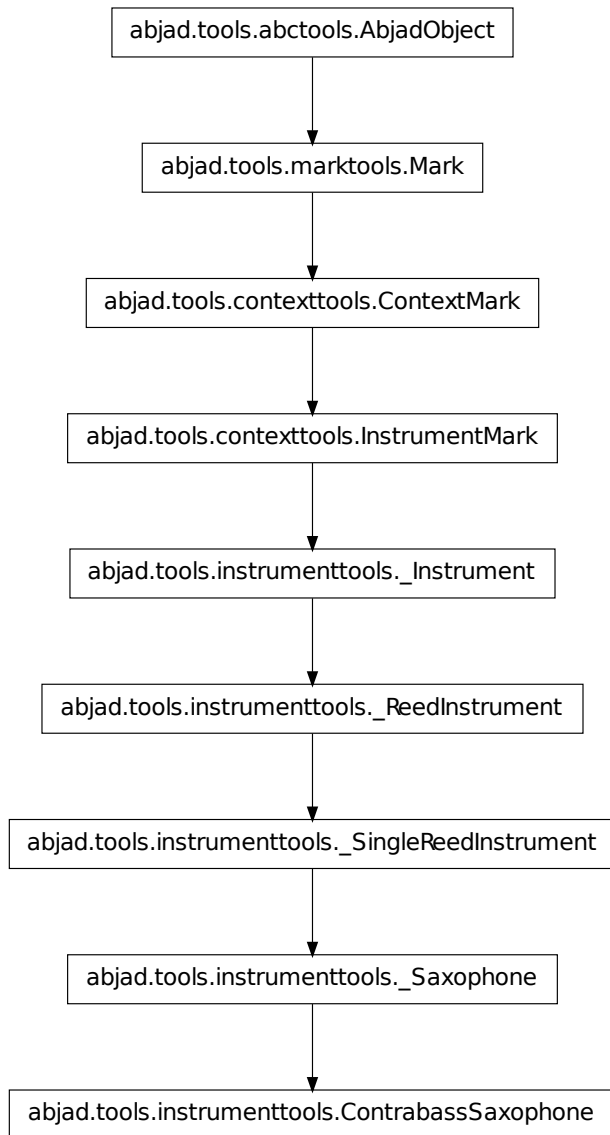
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

ContrabassFlute.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.ContrabassSaxophone



class `abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone`. **ContrabassSaxophone**
 New in version 2.6. Abjad model of the bass saxophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.ContrabassSaxophone()(staff)
ContrabassSaxophone()(Staff{4})
  
```



```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contrabass saxophone }
  \set Staff.shortInstrumentName = \markup { Cbass. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabass saxophone is pitched in E-flat.

The contrabass saxophone targets staff context by default.

Read-only Properties

`ContrabassSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ContrabassSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**ContrabassSaxophone.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

ContrabassSaxophone.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassSaxophone.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

ContrabassSaxophone.pitch_range

Note: Inherited from `instrumenttools._Instrument`

ContrabassSaxophone.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

ContrabassSaxophone.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone`.**`short_instrument_name_markup`**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone`.**`sounding_pitch_of_fingered_middle_c`**

Note: Inherited from `instrumenttools._Instrument`

Methods

`ContrabassSaxophone`.**`attach`**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`ContrabassSaxophone`.**`detach`**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`ContrabassSaxophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`ContrabassSaxophone.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`ContrabassSaxophone.__call__(*args)`

Note: Inherited from `marktools.Mark`

`ContrabassSaxophone.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`ContrabassSaxophone.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`ContrabassSaxophone.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContrabassSaxophone.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`ContrabassSaxophone.__repr__()`

Note: Inherited from `marktools.Mark`

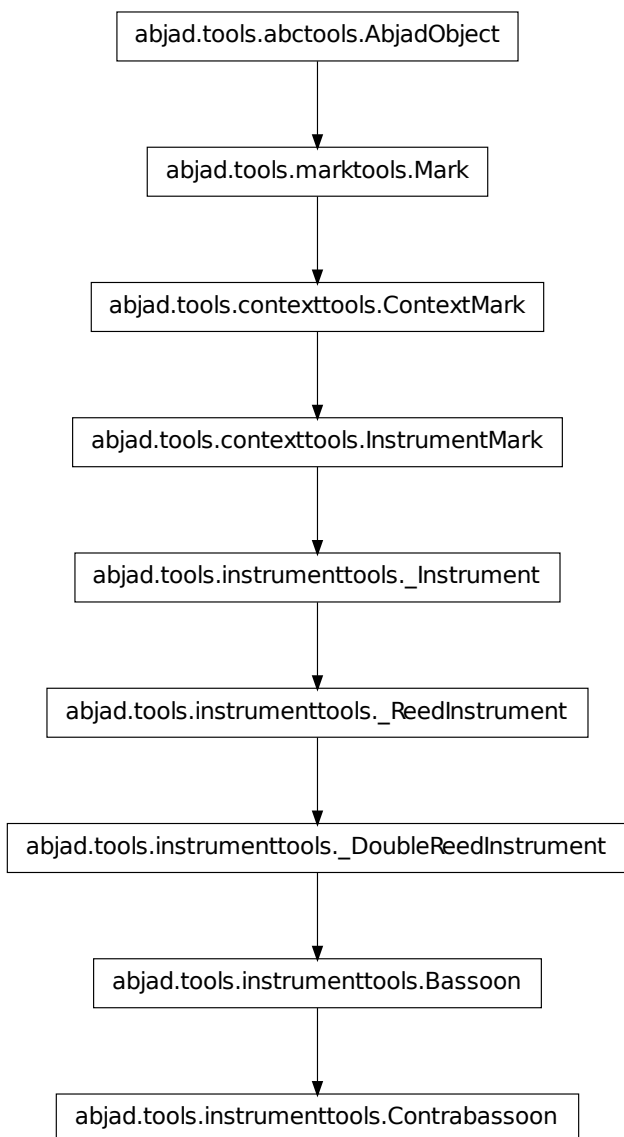
`ContrabassSaxophone.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ContrabassSaxophone.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Contrabassoon

class `abjad.tools.instrumenttools.Contrabassoon.Contrabassoon(**kwargs)`
 New in version 2.0. Abjad model of the contrabassoon:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.Contrabassoon()(staff)
Contrabassoon()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Contrabassoon }
  \set Staff.shortInstrumentName = \markup { Contrabsn. }
  c'8
  d'8
  e'8
  f'8
}
```

The contrabassoon targets staff context by default.

Read-only Properties

Contrabassoon.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Contrabassoon.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`Contrabassoon.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Contrabassoon.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Contrabassoon.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Contrabassoon.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Contrabassoon.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Contrabassoon.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Contrabassoon.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

Contrabassoon.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Contrabassoon.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Contrabassoon.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Contrabassoon.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Contrabassoon.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`Contrabassoon.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`Contrabassoon.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Contrabassoon.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Contrabassoon.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Contrabassoon.__call__` (*args)

Note: Inherited from `marktools.Mark`

`Contrabassoon.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`Contrabassoon.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`Contrabassoon.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Contrabassoon.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Contrabassoon.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

Contrabassoon.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Contrabassoon.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Contrabassoon.__ne__(arg)

Note: Inherited from `marktools.Mark`

Contrabassoon.__repr__()

Note: Inherited from `marktools.Mark`

Contrabassoon.__setattr__()

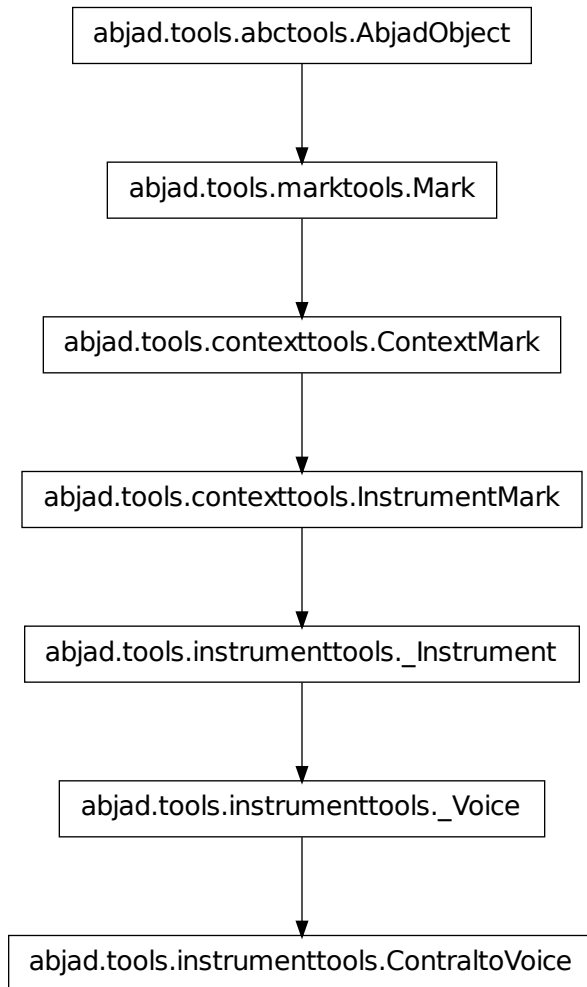
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Contrabassoon.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.ContraltoVoice



class `abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice(**kwargs)`
 New in version 2.8. Abjad model of the contralto voice:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.ContraltoVoice()(staff)
ContraltoVoice()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Contralto voice }
  \set Staff.shortInstrumentName = \markup { Contralto }
  c'8
  d'8

```

```

        e' 8
        f' 8
    }

```

The contralto voice targets staff context by default.

Read-only Properties

`ContraltoVoice.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.effective_context`

Read-only reference to effective context of context mark:

```

abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContraltoVoice.format`

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`ContraltoVoice.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`ContraltoVoice.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`ContraltoVoice.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`ContraltoVoice.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`ContraltoVoice.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`ContraltoVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`ContraltoVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`ContraltoVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`ContraltoVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`ContraltoVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContraltoVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ContraltoVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`ContraltoVoice.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContraltoVoice.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ContraltoVoice.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`ContraltoVoice.__repr__()`

Note: Inherited from `marktools.Mark`

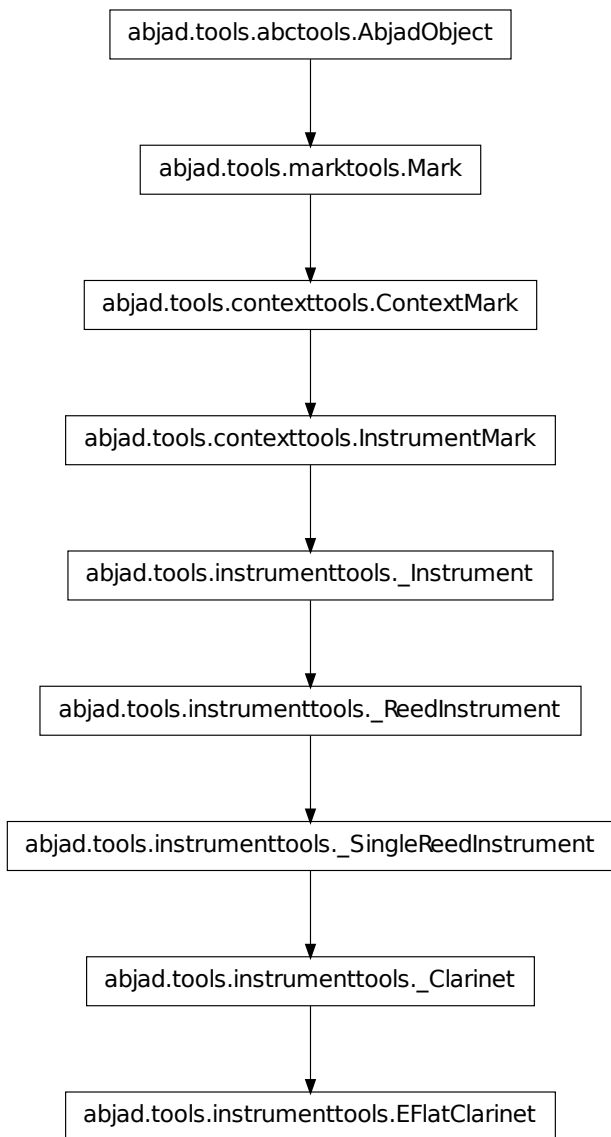
`ContraltoVoice.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ContraltoVoice.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.EFlatClarinet

class `abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet(**kwargs)`

New in version 2.0. Abjad model of the E-flat clarinet:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.EFlatClarinet()(staff)
EFlatClarinet()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in E-flat }
  \set Staff.shortInstrumentName = \markup { Cl. E-flat }
  c'8
  d'8
  e'8
  f'8
}
```

The E-flat clarinet targets staff context by default.

Read-only Properties

`EFlatClarinet.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`EFlatClarinet.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`EFlatClarinet.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`EFlatClarinet.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`EFlatClarinet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`EFlatClarinet.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`EFlatClarinet.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`EFlatClarinet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

EFlatClarinet.all_clefs

Note: Inherited from `instrumenttools._Instrument`

EFlatClarinet.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EFlatClarinet.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

EFlatClarinet.pitch_range

Note: Inherited from `instrumenttools._Instrument`

EFlatClarinet.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

EFlatClarinet.short_instrument_name

Get short instrument name:


```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EFlatClarinet.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

EFlatClarinet.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

EFlatClarinet.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

EFlatClarinet.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`EFlatClarinet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`EFlatClarinet.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`EFlatClarinet.__call__(*args)`

Note: Inherited from `marktools.Mark`

`EFlatClarinet.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`EFlatClarinet.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`EFlatClarinet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`EFlatClarinet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

EFlatClarinet.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

EFlatClarinet.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

EFlatClarinet.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

EFlatClarinet.__ne__(arg)

Note: Inherited from `marktools.Mark`

EFlatClarinet.__repr__()

Note: Inherited from `marktools.Mark`

EFlatClarinet.__setattr__()

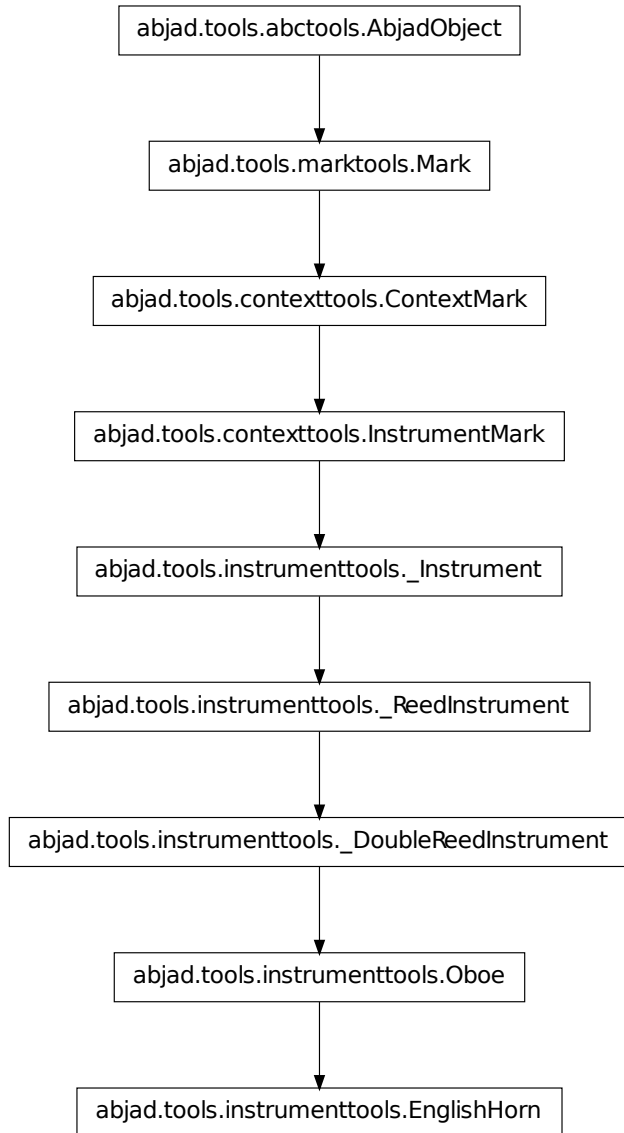
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

EFlatClarinet.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.EnglishHorn



class `abjad.tools.instrumenttools.EnglishHorn.EnglishHorn`.**EnglishHorn** *(**kwargs)*

New in version 2.0. Abjad model of the English horn:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.EnglishHorn()(staff)
EnglishHorn()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { English horn }
  \set Staff.shortInstrumentName = \markup { Eng. hn. }
  c'8
  d'8
  e'8
  f'8
}
```

The English horn targets staff context by default.

Read-only Properties

EnglishHorn.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

EnglishHorn.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`EnglishHorn.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`EnglishHorn.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`EnglishHorn.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`EnglishHorn.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`EnglishHorn.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`EnglishHorn.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**EnglishHorn.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

EnglishHorn.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.pitch_range

Note: Inherited from `instrumenttools._Instrument`

EnglishHorn.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

EnglishHorn.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

EnglishHorn.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

EnglishHorn.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```



```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`EnglishHorn.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`EnglishHorn.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`EnglishHorn.__call__(*args)`

Note: Inherited from `marktools.Mark`

`EnglishHorn.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`EnglishHorn.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`EnglishHorn.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`EnglishHorn.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

EnglishHorn.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

EnglishHorn.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

EnglishHorn.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

EnglishHorn.__ne__(arg)

Note: Inherited from `marktools.Mark`

EnglishHorn.__repr__()

Note: Inherited from `marktools.Mark`

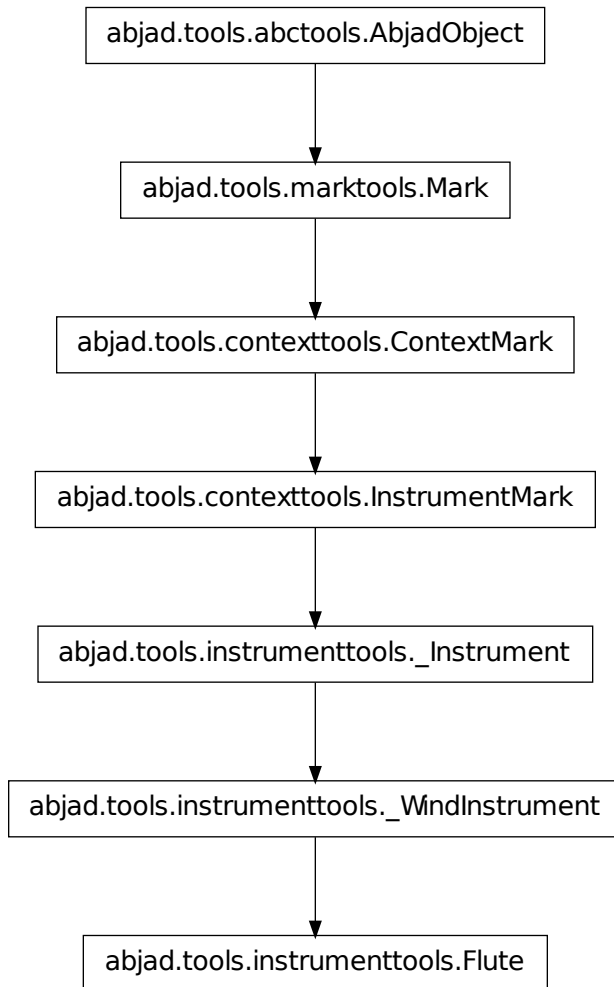
EnglishHorn.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

EnglishHorn.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Flute

class `abjad.tools.instrumenttools.Flute.Flute`**.Flute** *(**kwargs)*

New in version 2.0. Abjad model of the flute:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Flute()(staff)
Flute()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Flute }
  \set Staff.shortInstrumentName = \markup { Fl. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The flute targets staff context by default.

Read-only Properties

Flute.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Flute.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Flute.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Flute.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Flute.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Flute.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Flute.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Flute.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Flute.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Flute.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Flute.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Flute.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Flute.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Flute.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Flute.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Flute.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Flute.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Flute.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Flute.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Flute.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Flute.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Flute.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Flute.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Flute.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Flute.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Flute.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Flute.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Flute.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Flute.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Flute.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Flute.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Flute.__ne__(arg)

Note: Inherited from `marktools.Mark`

Flute.__repr__()

Note: Inherited from `marktools.Mark`

Flute.__setattr__()

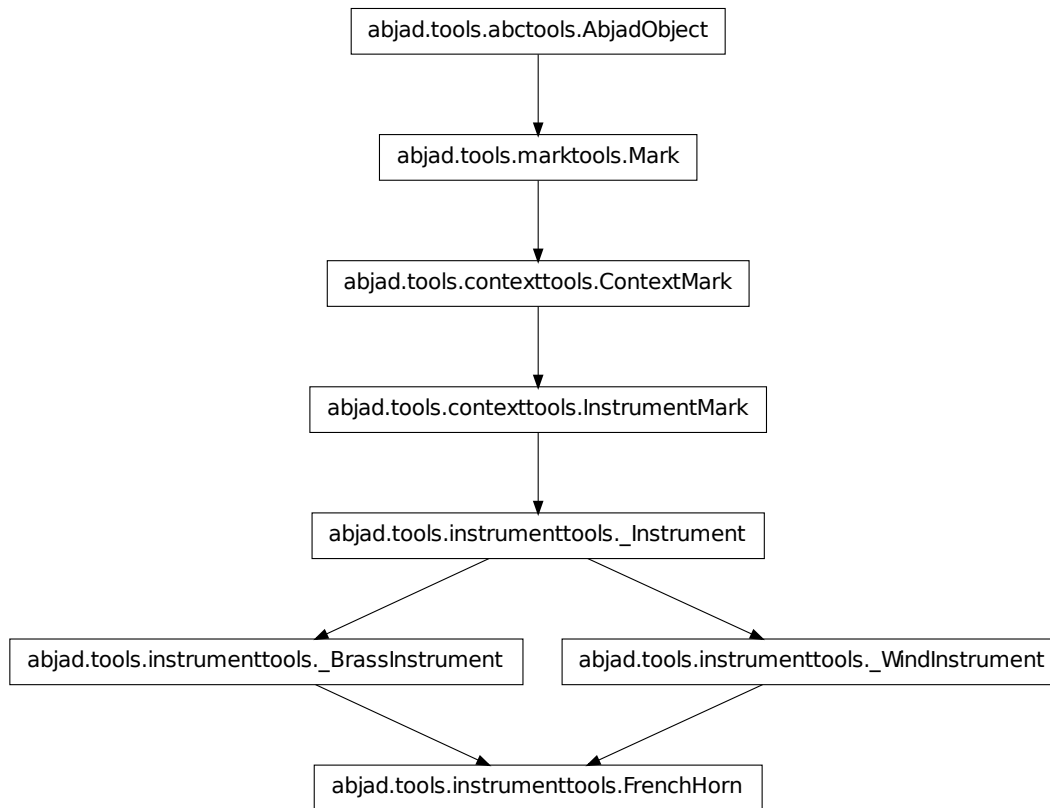
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Flute.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.FrenchHorn



class `abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn` (**kwargs)
 New in version 2.0. Abjad model of the French horn:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.FrenchHorn()(staff)
FrenchHorn()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Horn }
  \set Staff.shortInstrumentName = \markup { Hn. }
  c'8
  d'8
  e'8
  f'8
}
```

The French horn targets staff context by default.

Read-only Properties

FrenchHorn.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

FrenchHorn.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

FrenchHorn.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

FrenchHorn.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**FrenchHorn.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

FrenchHorn.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

FrenchHorn.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods**FrenchHorn.attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

FrenchHorn.detach()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

FrenchHorn.get_default_performer_name(*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`FrenchHorn.get_performer_names()`
New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`FrenchHorn.__call__(*args)`

Note: Inherited from `marktools.Mark`

`FrenchHorn.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`FrenchHorn.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`FrenchHorn.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FrenchHorn.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`FrenchHorn.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`FrenchHorn.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FrenchHorn.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FrenchHorn.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`FrenchHorn.__repr__()`

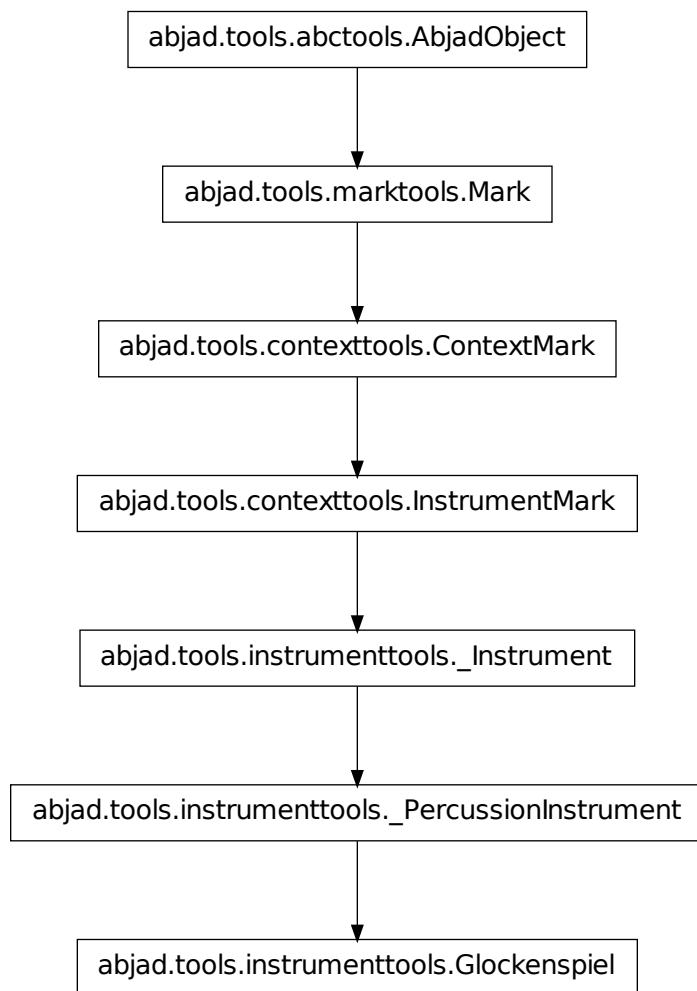
Note: Inherited from `marktools.Mark`

`FrenchHorn.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`FrenchHorn.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Glockenspiel

class `abjad.tools.instrumenttools.Glockenspiel.Glockenspiel(**kwargs)`
 New in version 2.0. Abjad model of the glockenspiel:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Glockenspiel()(staff)
Glockenspiel()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Glockenspiel }
  \set Staff.shortInstrumentName = \markup { Gkspl. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The `glockenspiel` targets staff context by default.

Read-only Properties

`Glockenspiel.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Glockenspiel.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Glockenspiel.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Glockenspiel.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`Glockenspiel.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`Glockenspiel.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`Glockenspiel.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Glockenspiel.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Glockenspiel.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Glockenspiel.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Glockenspiel.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Glockenspiel.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Glockenspiel.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Glockenspiel.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Glockenspiel.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Glockenspiel.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Glockenspiel.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Glockenspiel.__ne__(arg)

Note: Inherited from `marktools.Mark`

Glockenspiel.__repr__()

Note: Inherited from `marktools.Mark`

Glockenspiel.__setattr__()

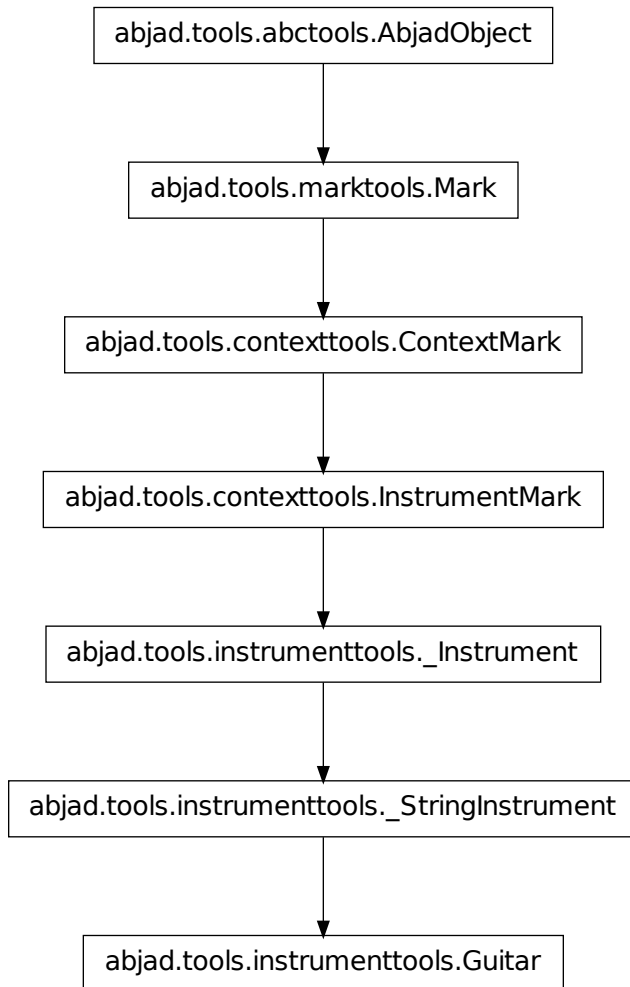
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Glockenspiel.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Guitar



class `abjad.tools.instrumenttools.Guitar.Guitar.Guitar` **Guitar** *(**kwargs)*

New in version 2.0. Abjad model of the guitar:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Guitar()(staff)
Guitar()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Guitar }
  \set Staff.shortInstrumentName = \markup { Gt. }
  c'8
  d'8

```



```

        e' 8
        f' 8
    }

```

The guitar targets staff context by default.

Read-only Properties

Guitar.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Guitar.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Guitar.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Guitar.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Guitar.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Guitar.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Guitar.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Guitar.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Guitar.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Guitar.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Guitar.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Guitar.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Guitar.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Guitar.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Guitar.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Guitar.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Guitar.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Guitar.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Guitar.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Guitar.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Guitar.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Guitar.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Guitar.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Guitar.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Guitar.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Guitar.__ne__(arg)

Note: Inherited from `marktools.Mark`

Guitar.__repr__()

Note: Inherited from `marktools.Mark`

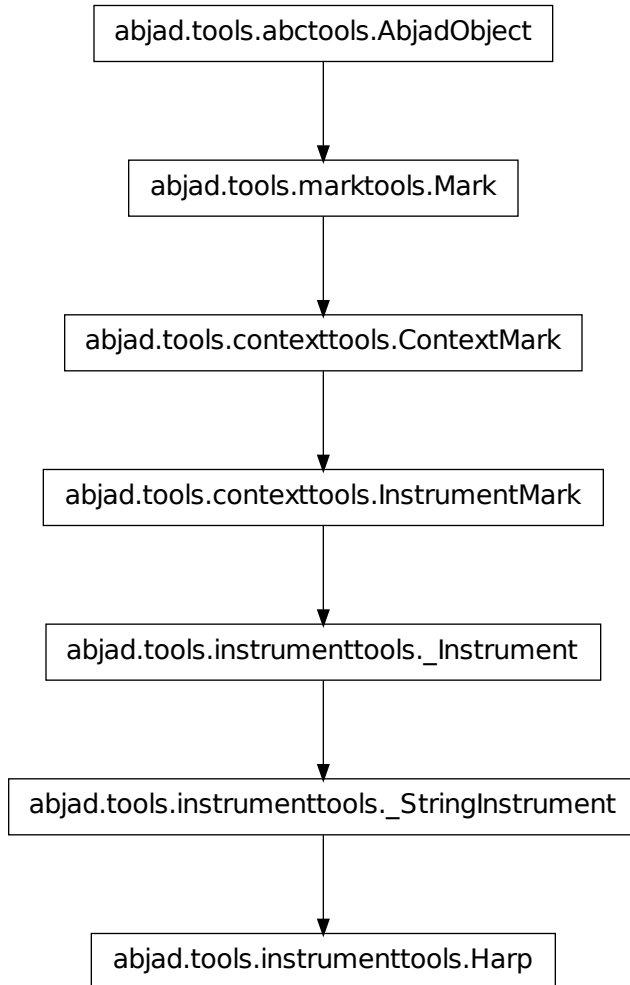
Guitar.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Guitar.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Harp

class `abjad.tools.instrumenttools.Harp.Harp.Harp` (*target_context=None, **kwargs*)

New in version 2.0. Abjad model of the harp:

```
abjad> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
abjad> instrumenttools.Harp()(piano_staff)
Harp()(PianoStaff<<2>>)
```

```
abjad> f(piano_staff)
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Harp }
  \set PianoStaff.shortInstrumentName = \markup { Hp. }
  \new Staff {
    c'8
```

```

        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>

```

The harp targets piano staff context by default.

Read-only Properties

Harp.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harp.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harp.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Harp.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Harp.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Harp.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Harp.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Harp.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Harp.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Harp.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Harp.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Harp.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Harp.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harp.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Harp.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Harp.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Harp.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harp.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Harp.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Harp.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Harp.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Harp.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

Harp.**get_performer_names** ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

Harp.**__call__** (*args)

Note: Inherited from `marktools.Mark`

Harp.**__delattr__** (*args)

Note: Inherited from `marktools.Mark`

Harp.**__eq__** (arg)

Note: Inherited from `contexttools.InstrumentMark`

Harp.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Harp.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Harp.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

Harp.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Harp.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Harp.__ne__(arg)

Note: Inherited from `marktools.Mark`

Harp.__repr__()

Note: Inherited from `marktools.Mark`

Harp.__setattr__()

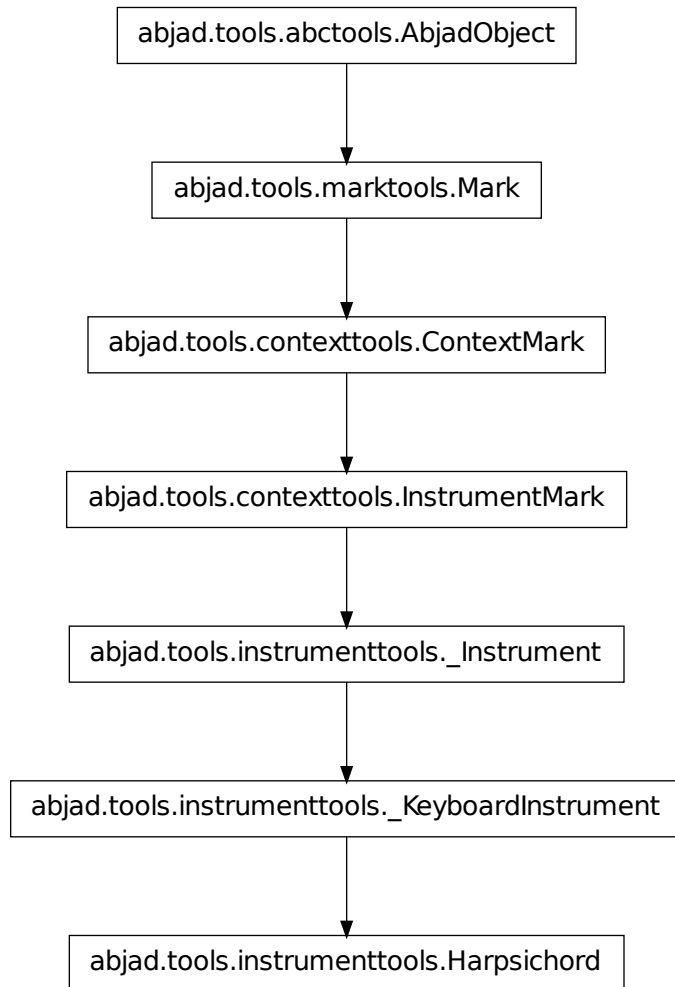
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Harp.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Harpsichord



class `abjad.tools.instrumenttools.Harpsichord.Harpsichord`(*target_context=None*,
***kwargs*)

New in version 2.5. Abjad model of the harpsichord:

```
abjad> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
abjad> instrumenttools.Harpsichord()(piano_staff)
Harpsichord() (PianoStaff<<2>>)
```

```
abjad> f(piano_staff)
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Harpsichord }
  \set PianoStaff.shortInstrumentName = \markup { Hpschd. }
  \new Staff {
```

```
        c'8
        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>
```

The harpsichord targets piano staff context by default.

Return instrument.

Read-only Properties

Harpsichord.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harpsichord.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harpsichord.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Harpsichord.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`Harpsichord.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Harpsichord.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Harpsichord.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`Harpsichord.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Harpsichord.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Harpsichord.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

Harpsichord.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Harpsichord.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Harpsichord.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods**Harpsichord.attach** (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Harpsichord.detach ()

Detach mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c' 4")
```

```
abjad> context_mark.detach()  
ContextMark()
```

```
abjad> context_mark.start_component is None  
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Harpsichord.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Harpsichord.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Harpsichord.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Harpsichord.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Harpsichord.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Harpsichord.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Harpsichord.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Harpsichord.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`Harpsichord.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Harpsichord.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Harpsichord.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`Harpsichord.__repr__()`

Note: Inherited from `marktools.Mark`

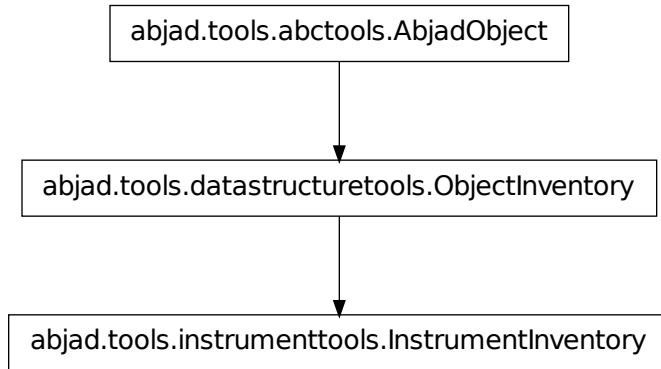
`Harpsichord.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Harpsichord.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.InstrumentInventory

class `abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory`. **InstrumentInventory**

New in version 2.8. Abjad model of an ordered list of instruments:

```

abjad> inventory = instrumenttools.InstrumentInventory([instrumenttools.Flute(), instrumenttools.
abjad> inventory
InstrumentInventory([Flute(), Guitar()])
  
```

Instrument inventories implement list interface and are mutable.

Read/write Properties

`InstrumentInventory.inventory_name`

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`InstrumentInventory.append(token)`

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.count(value) → integer` – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`InstrumentInventory.extend(tokens)`
Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.index(value[, start[, stop]])` → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`InstrumentInventory.insert()`
`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`InstrumentInventory.pop([index])` → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`InstrumentInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`InstrumentInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`InstrumentInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`InstrumentInventory.__add__()`
`x.__add__(y)` <==> `x+y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.__delattr__()`
`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`InstrumentInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__iter__()` $\iff iter(x)$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__le__()`
`x.__le__(y)` $\iff x \leq y$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__len__()` $\iff len(x)$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__mul__()`
`x.__mul__(n)` $\iff x * n$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__repr__()`

Note: Inherited from `datastructuretools.ObjectInventory`

`InstrumentInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`InstrumentInventory.__rmul__()`
`x.__rmul__(n)` $\iff n * x$

Note: Inherited from `__builtin__.list`

```
InstrumentInventory.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
InstrumentInventory.__setitem__()  
x.__setitem__(i, y) <==> x[i]=y
```

Note: Inherited from `__builtin__.list`

```
InstrumentInventory.__setslice__()  
x.__setslice__(i, j, y) <==> x[i:j]=y
```

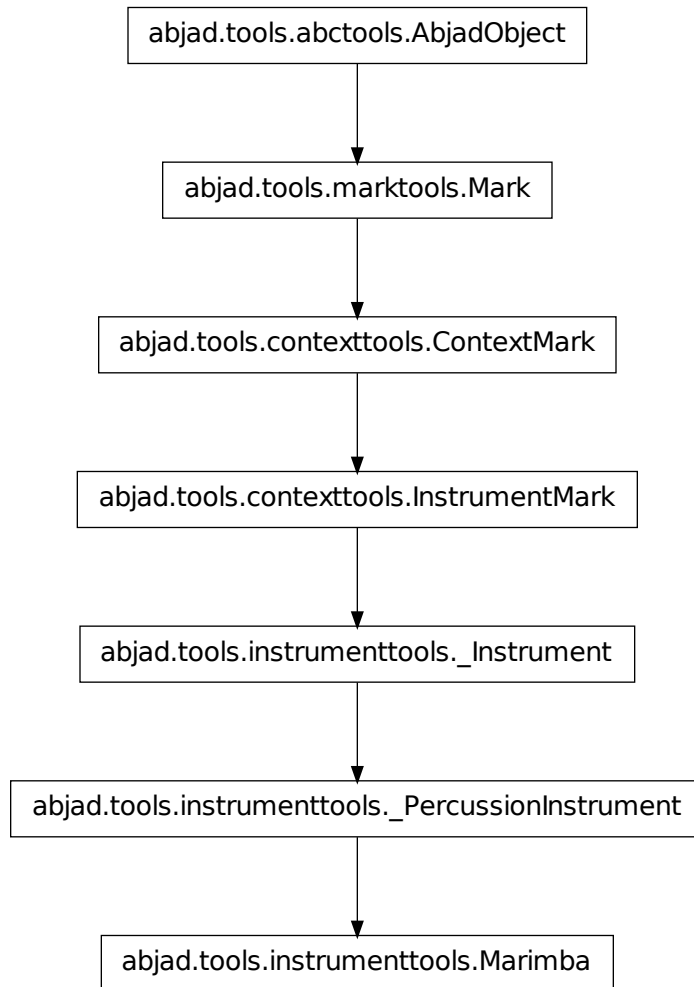
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
InstrumentInventory.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

instrumenttools.Marimba



class `abjad.tools.instrumenttools.Marimba.Marimba`.**Marimba** (**kwargs)

New in version 2.0. Abjad model of the marimba:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Marimba()(staff)
Marimba()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Marimba }
  \set Staff.shortInstrumentName = \markup { Mb. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The marimba targets staff context by default.

Read-only Properties

Marimba.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Marimba.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Marimba.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Marimba.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Marimba.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Marimba.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Marimba.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Marimba.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Marimba.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Marimba.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Marimba.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Marimba.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Marimba.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Marimba.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Marimba.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Marimba.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Marimba.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Marimba.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Marimba.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`Marimba.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`Marimba.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Marimba.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Marimba.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Marimba.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Marimba.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Marimba.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Marimba.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Marimba.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Marimba.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Marimba.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Marimba.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Marimba.__ne__(arg)

Note: Inherited from `marktools.Mark`

Marimba.__repr__()

Note: Inherited from `marktools.Mark`

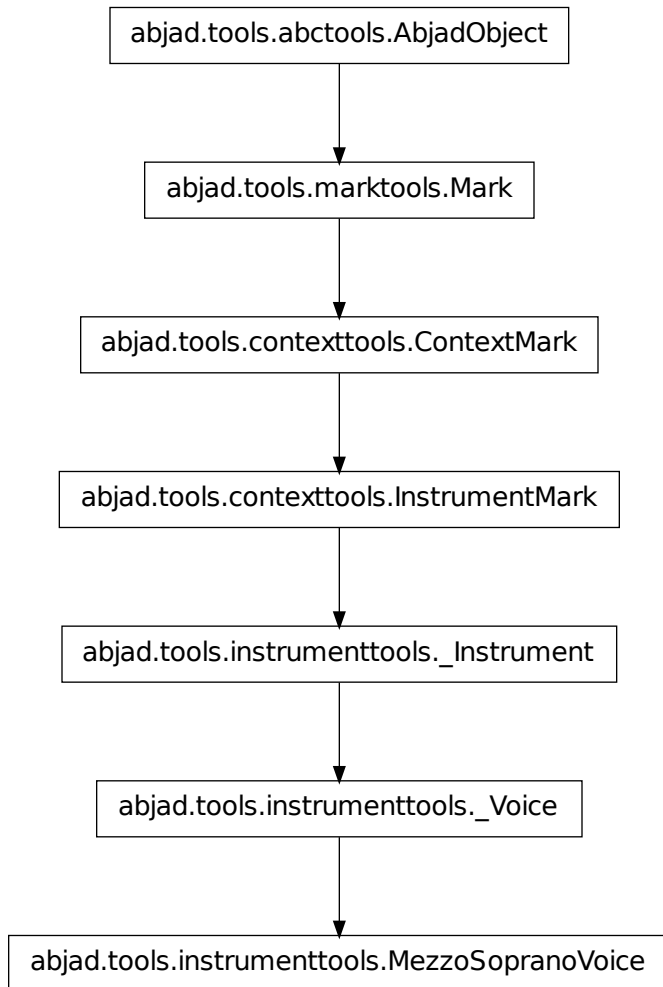
Marimba.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Marimba.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.MezzoSopranoVoice

class `abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice` *(**kwargs)*

New in version 2.8. Abjad model of the mezzo-soprano voice:

```
abjad> staff = Staff("c''8 d''8 e''8 f''8")
```

```
abjad> instrumenttools.MezzoSopranoVoice()(staff)
MezzoSopranoVoice()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Mezzo-soprano voice }
  \set Staff.shortInstrumentName = \markup { Mezzo-soprano }
  c''8
  d''8
```

```
e''8
f''8
}
```

The mezzo-soprano voice targets staff context by default.

Read-only Properties

`MezzoSopranoVoice.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`MezzoSopranoVoice.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`MezzoSopranoVoice.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`MezzoSopranoVoice.attach` (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.detach` ()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.start_component
Note("c'4")
```

```
abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`MezzoSopranoVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`MezzoSopranoVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`MezzoSopranoVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`MezzoSopranoVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`MezzoSopranoVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`MezzoSopranoVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MezzoSopranoVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MezzoSopranoVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

MezzoSopranoVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MezzoSopranoVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MezzoSopranoVoice.__ne__(arg)

Note: Inherited from `marktools.Mark`

MezzoSopranoVoice.__repr__()

Note: Inherited from `marktools.Mark`

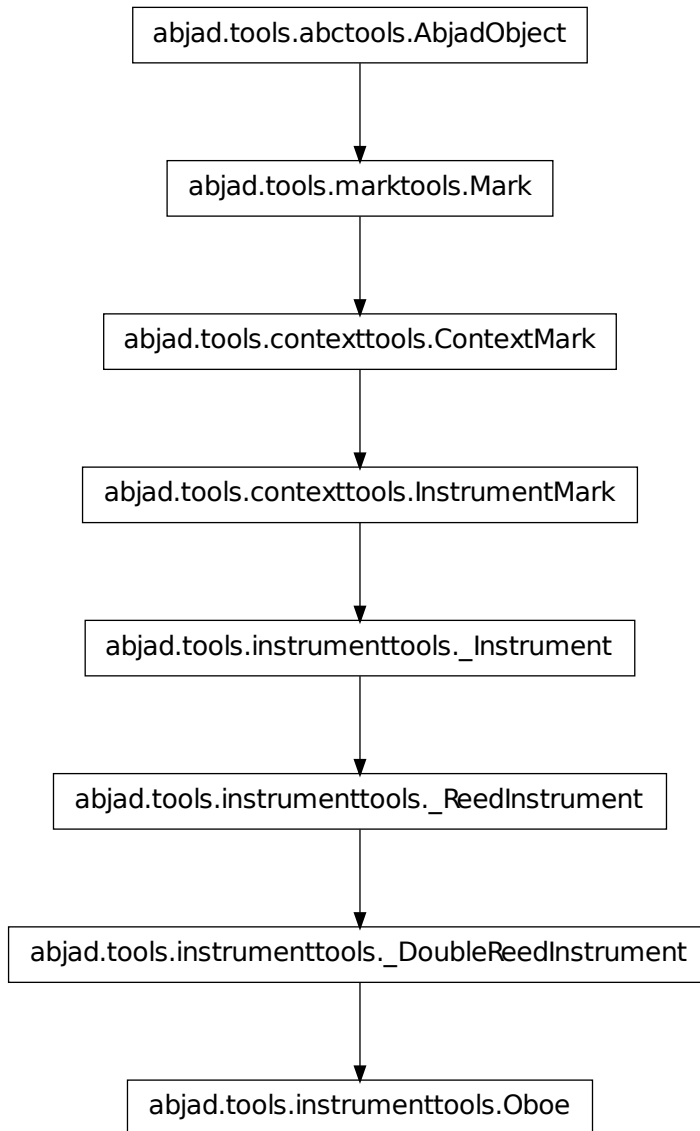
MezzoSopranoVoice.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

MezzoSopranoVoice.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Oboe

class `abjad.tools.instrumenttools.Oboe.Oboe`.**Oboe** *(**kwargs)*

New in version 2.0. Abjad model of the oboe:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.Oboe()(staff)
```

```
Oboe()(Staff{4})
```



```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Oboe }
  \set Staff.shortInstrumentName = \markup { Ob. }
  c'8
  d'8
  e'8
  f'8
}
```

The oboe targets staff context by default.

Read-only Properties

Oboe.**default_instrument_name**

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**default_short_instrument_name**

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**effective_context**

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Oboe.**format**

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**interval_of_transposition**

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Oboe.**`is_primary_instrument`**

Note: Inherited from `instrumenttools._Instrument`

Oboe.**`is_secondary_instrument`**

Note: Inherited from `instrumenttools._Instrument`

Oboe.**`is_transposing`**

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Oboe.**`start_component`**

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Oboe.**`target_context`**

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Oboe.**`traditional_pitch_range`**

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**Oboe.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

Oboe.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Oboe.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Oboe.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Oboe.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Oboe.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

Oboe.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

Oboe.**get_performer_names** ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

Oboe.**__call__** (*args)

Note: Inherited from `marktools.Mark`

Oboe.**__delattr__** (*args)

Note: Inherited from `marktools.Mark`

Oboe.**__eq__** (arg)

Note: Inherited from `contexttools.InstrumentMark`

Oboe.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Oboe.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Oboe. **__hash__** ()

Note: Inherited from `contexttools.InstrumentMark`

Oboe. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Oboe. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Oboe. **__ne__** (*arg*)

Note: Inherited from `marktools.Mark`

Oboe. **__repr__** ()

Note: Inherited from `marktools.Mark`

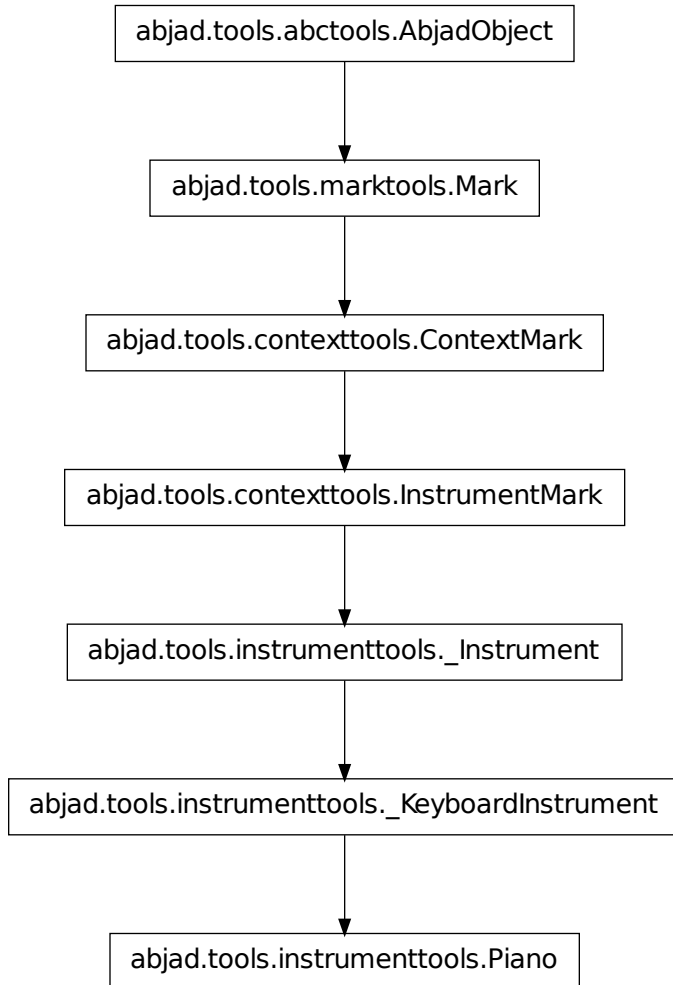
Oboe. **__setattr__** ()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Oboe. **__str__** () <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Piano

class `abjad.tools.instrumenttools.Piano.Piano`.**Piano** (*target_context=None*, ***kwargs*)

New in version 2.0. Abjad model of the piano:

```
abjad> piano_staff = scoretools.PianoStaff([Staff("c'8 d'8 e'8 f'8"), Staff("c'4 b4")])
```

```
abjad> instrumenttools.Piano()(piano_staff)
Piano() (PianoStaff<<2>>)
```

```
abjad> f(piano_staff)
\new PianoStaff <<
  \set PianoStaff.instrumentName = \markup { Piano }
  \set PianoStaff.shortInstrumentName = \markup { Pf. }
  \new Staff {
    c'8
```

```

        d'8
        e'8
        f'8
    }
    \new Staff {
        c'4
        b4
    }
>>

```

The piano targets piano staff context by default.

Read-only Properties

Piano.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piano.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piano.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Piano.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Piano.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Piano.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Piano.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Piano.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Piano.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Piano.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Piano.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Piano.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Piano.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piano.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Piano.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Piano.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Piano.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piano.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Piano.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Piano.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Piano.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Piano.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Piano.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Piano.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Piano.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Piano.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Piano.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Piano.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Piano.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

Piano.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Piano.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Piano.__ne__(arg)

Note: Inherited from `marktools.Mark`

Piano.__repr__()

Note: Inherited from `marktools.Mark`

Piano.__setattr__()

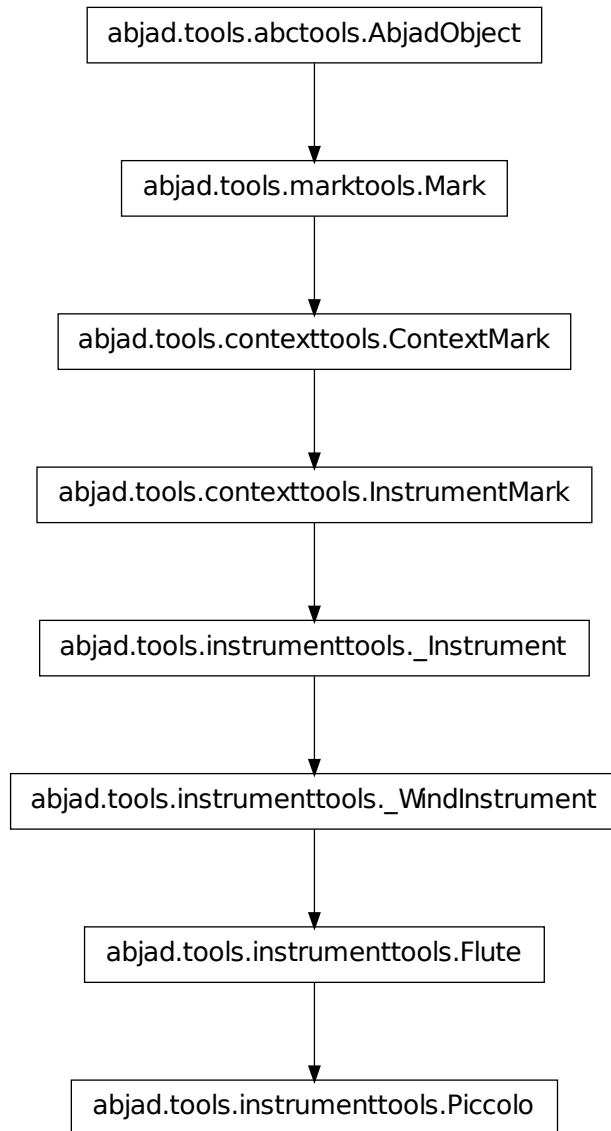
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Piano.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Piccolo



class `abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo(**kwargs)`

New in version 2.0. Abjad model of the piccolo:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.Piccolo()(staff)
Piccolo()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  c'8
  d'8
  e'8
  f'8
}
```

The piccolo targets staff context by default.

Read-only Properties

Piccolo.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Piccolo.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Piccolo.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Piccolo.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Piccolo.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Piccolo.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Piccolo.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Piccolo.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**Piccolo.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

Piccolo.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Piccolo.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Piccolo.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.short_instrument_name_markup

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.sounding_pitch_of_fingered_middle_c

Note: Inherited from `instrumenttools._Instrument`

Methods

Piccolo.attach(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Piccolo.detach()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Piccolo.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Piccolo.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Piccolo.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Piccolo.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Piccolo.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Piccolo.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Piccolo.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Piccolo.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

Piccolo.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Piccolo.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Piccolo.__ne__(arg)

Note: Inherited from `marktools.Mark`

Piccolo.__repr__()

Note: Inherited from `marktools.Mark`

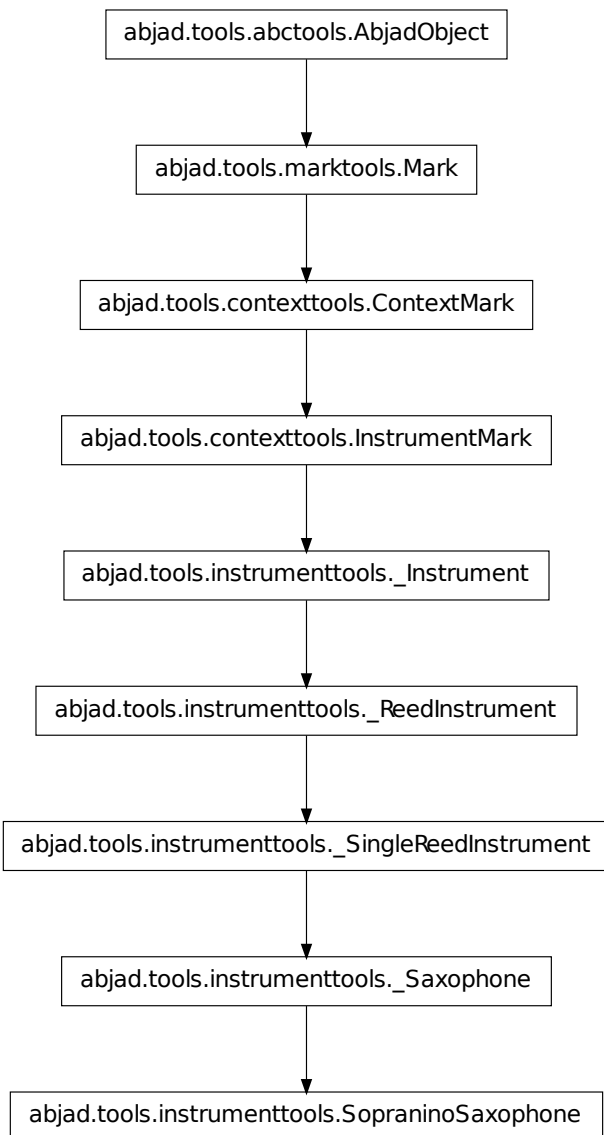
Piccolo.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Piccolo.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.SopraninoSaxophone

class `abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone` **SopraninoSaxophone** (**)

New in version 2.6. Abjad model of the sopranino saxophone:

`abjad> staff = Staff("c'8 d'8 e'8 f'8")`

`abjad> instrumenttools.SopraninoSaxophone()(staff)`

`SopraninoSaxophone()(Staff{4})`

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Sopranino saxophone }
  \set Staff.shortInstrumentName = \markup { Sopranino sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The soprano saxophone is pitched in E-flat.

The soprano saxophone targets staff context by default.

Read-only Properties

`SopraninoSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`SopraninoSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`SopraninoSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`SopraninoSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

SopraninoSaxophone.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

SopraninoSaxophone.**short_instrument_name**

Get short instrument name:


```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

SopraninoSaxophone.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

SopraninoSaxophone.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`SopraninoSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`SopraninoSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`SopraninoSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`SopraninoSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`SopraninoSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`SopraninoSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SopraninoSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

SopraninoSaxophone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

SopraninoSaxophone.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

SopraninoSaxophone.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

SopraninoSaxophone.__ne__(arg)

Note: Inherited from `marktools.Mark`

SopraninoSaxophone.__repr__()

Note: Inherited from `marktools.Mark`

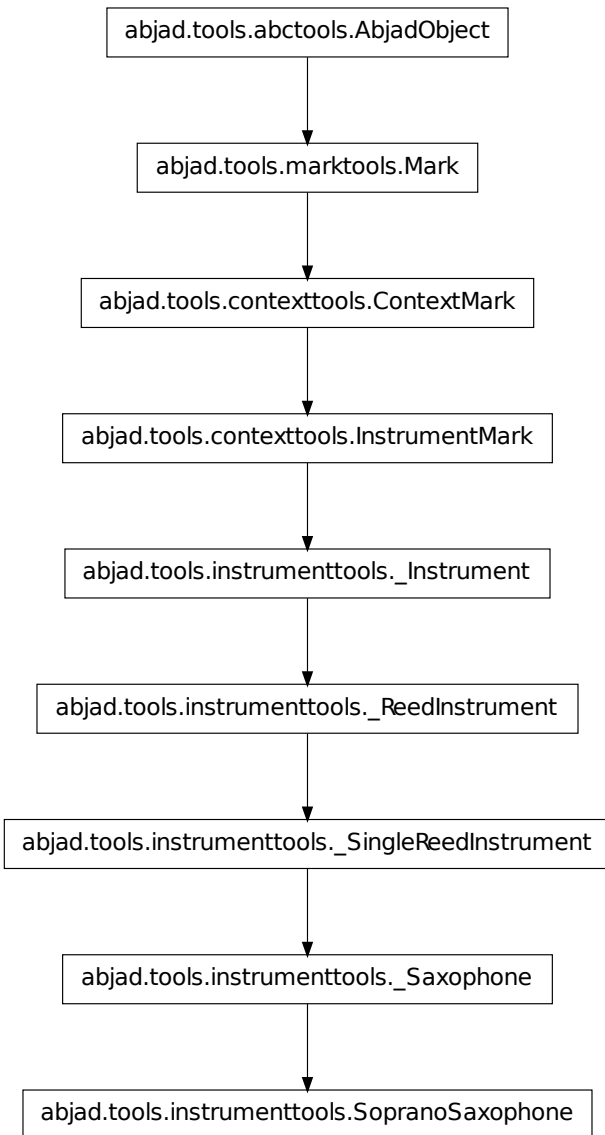
SopraninoSaxophone.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

SopraninoSaxophone.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

instrumenttools.SopranoSaxophone



class `abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone` **SopranoSaxophone** (***kwargs*)

New in version 2.6. Abjad model of the soprano saxophone:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> instrumenttools.SopranoSaxophone()(staff)
SopranoSaxophone()(Staff{4})
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Soprano saxophone }
  \set Staff.shortInstrumentName = \markup { Sop. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The soprano saxophone is pitched in B-flat.

The soprano saxophone targets staff context by default.

Read-only Properties

`SopranoSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`SopranoSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`SopranoSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`SopranoSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**SopranoSaxophone.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

SopranoSaxophone.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.pitch_range

Note: Inherited from `instrumenttools._Instrument`

SopranoSaxophone.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

SopranoSaxophone.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

SopranoSaxophone.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

SopranoSaxophone.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

SopranoSaxophone.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```



```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`SopranoSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`SopranoSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`SopranoSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`SopranoSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`SopranoSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`SopranoSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SopranoSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

SopranoSaxophone. **__hash__** ()

Note: Inherited from `contexttools.InstrumentMark`

SopranoSaxophone. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

SopranoSaxophone. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

SopranoSaxophone. **__ne__** (*arg*)

Note: Inherited from `marktools.Mark`

SopranoSaxophone. **__repr__** ()

Note: Inherited from `marktools.Mark`

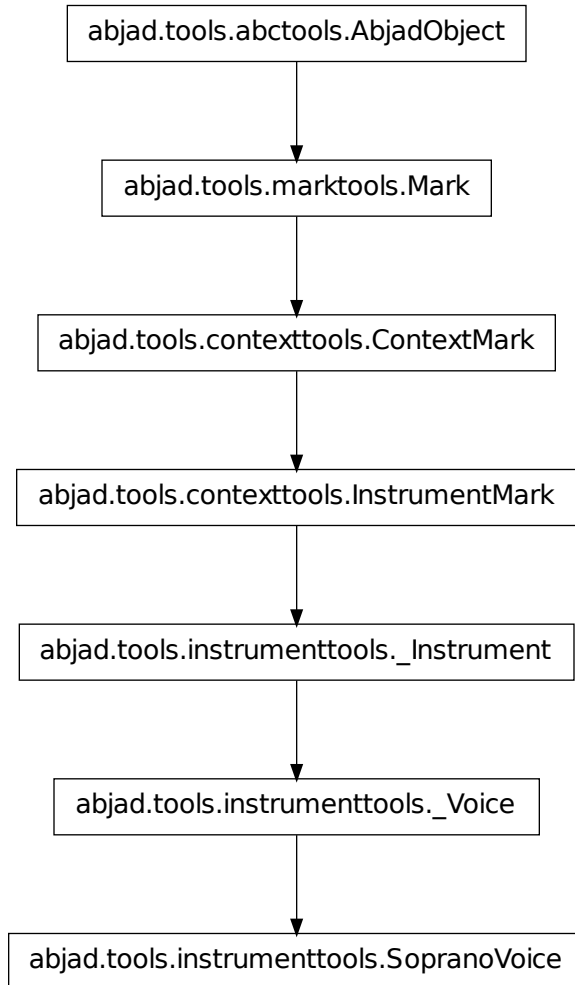
SopranoSaxophone. **__setattr__** ()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

SopranoSaxophone. **__str__** () <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.SopranoVoice

class `abjad.tools.instrumenttools.SopranoVoice.SopranoVoice(**kwargs)`
 New in version 2.8. Abjad model of the soprano voice:

```

abjad> staff = Staff("c''8 d''8 e''8 f''8")

abjad> instrumenttools.SopranoVoice()(staff)
SopranoVoice()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Soprano voice }
  \set Staff.shortInstrumentName = \markup { Soprano }
  c''8
  d''8

```

```
e''8
f''8
}
```

The soprano voice targets staff context by default.

Read-only Properties

SopranoVoice.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopranoVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopranoVoice.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

SopranoVoice.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

SopranoVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

SopranoVoice.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

SopranoVoice.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

SopranoVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

SopranoVoice.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

SopranoVoice.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

SopranoVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

SopranoVoice.all_clefs

Note: Inherited from `instrumenttools._Instrument`

`SopranoVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`SopranoVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`SopranoVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`SopranoVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`SopranoVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

SopranoVoice.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

SopranoVoice.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

SopranoVoice.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

SopranoVoice.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`SopranoVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`SopranoVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`SopranoVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`SopranoVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`SopranoVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`SopranoVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SopranoVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SopranoVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

`SopranoVoice.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SopranoVoice.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SopranoVoice.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`SopranoVoice.__repr__()`

Note: Inherited from `marktools.Mark`

`SopranoVoice.__setattr__()`

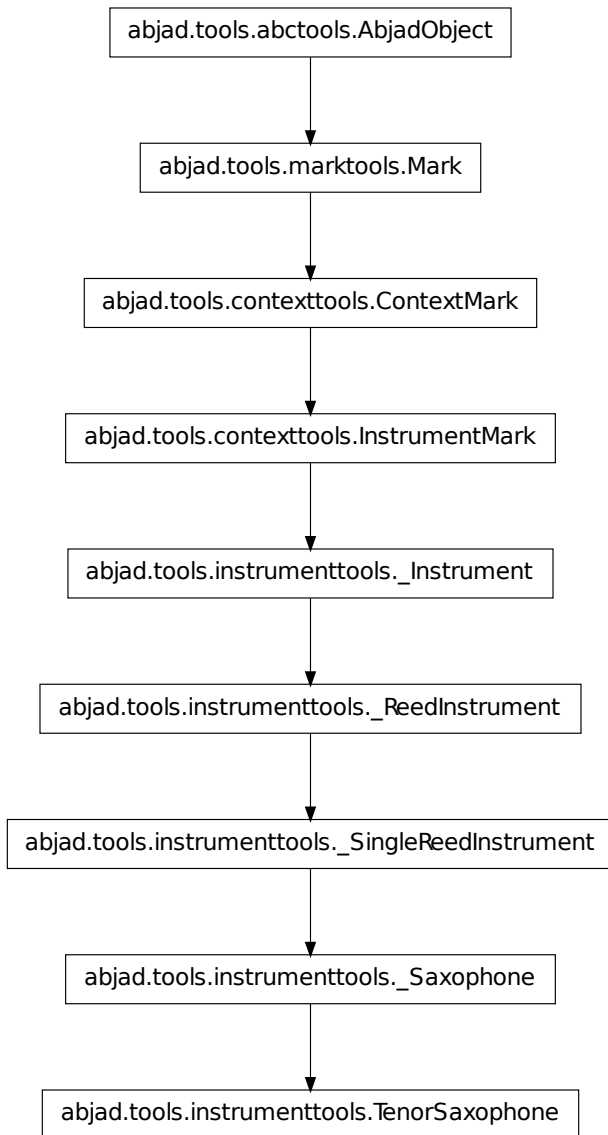
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SopranoVoice.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.TenorSaxophone



class `abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone`. **TenorSaxophone** (**kwargs)

New in version 2.6. Abjad model of the tenor saxophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.TenorSaxophone()(staff)
TenorSaxophone()(Staff{4})
  
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Tenor saxophone }
  \set Staff.shortInstrumentName = \markup { Ten. sax. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor saxophone is pitched in B-flat.

The tenor saxophone targets staff context by default.

Read-only Properties

`TenorSaxophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`TenorSaxophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`TenorSaxophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`TenorSaxophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`TenorSaxophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`TenorSaxophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties**TenorSaxophone.all_clefs**

Note: Inherited from `instrumenttools._Instrument`

TenorSaxophone.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorSaxophone.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

TenorSaxophone.pitch_range

Note: Inherited from `instrumenttools._Instrument`

TenorSaxophone.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

TenorSaxophone.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`TenorSaxophone.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`TenorSaxophone.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`TenorSaxophone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`TenorSaxophone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`TenorSaxophone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`TenorSaxophone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`TenorSaxophone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`TenorSaxophone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TenorSaxophone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

TenorSaxophone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

TenorSaxophone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorSaxophone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorSaxophone.__ne__(arg)

Note: Inherited from `marktools.Mark`

TenorSaxophone.__repr__()

Note: Inherited from `marktools.Mark`

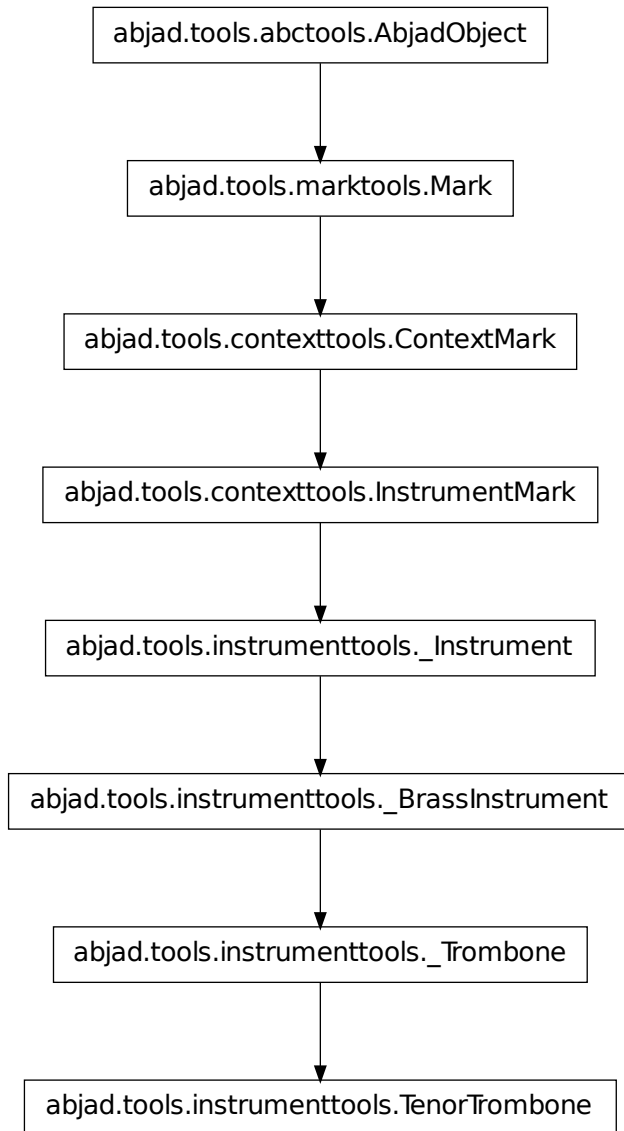
TenorSaxophone.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

TenorSaxophone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.TenorTrombone

class `abjad.tools.instrumenttools.TenorTrombone.TenorTrombone`. **TenorTrombone** (**kwargs)

New in version 2.0. Abjad model of the tenor trombone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.TenorTrombone()(staff)
TenorTrombone()(Staff{4})

```

```
abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Tenor trombone }
  \set Staff.shortInstrumentName = \markup { Ten. trb. }
  c'8
  d'8
  e'8
  f'8
}
```

The tenor trombone targets staff context by default.

Read-only Properties

TenorTrombone.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

TenorTrombone.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`TenorTrombone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`TenorTrombone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`TenorTrombone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`TenorTrombone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`TenorTrombone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`TenorTrombone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

TenorTrombone.**all_clefs**

Note: Inherited from `instrumenttools._Instrument`

TenorTrombone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

TenorTrombone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

TenorTrombone.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorTrombone.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`TenorTrombone.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`TenorTrombone.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`TenorTrombone.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()
```

```
abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`TenorTrombone.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`TenorTrombone.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`TenorTrombone.__call__` (*args)

Note: Inherited from `marktools.Mark`

`TenorTrombone.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`TenorTrombone.__eq__` (arg)

Note: Inherited from `contexttools.InstrumentMark`

`TenorTrombone.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TenorTrombone.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

TenorTrombone.__hash__()

Note: Inherited from `contexttools.InstrumentMark`

TenorTrombone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorTrombone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorTrombone.__ne__(arg)

Note: Inherited from `marktools.Mark`

TenorTrombone.__repr__()

Note: Inherited from `marktools.Mark`

TenorTrombone.__setattr__()

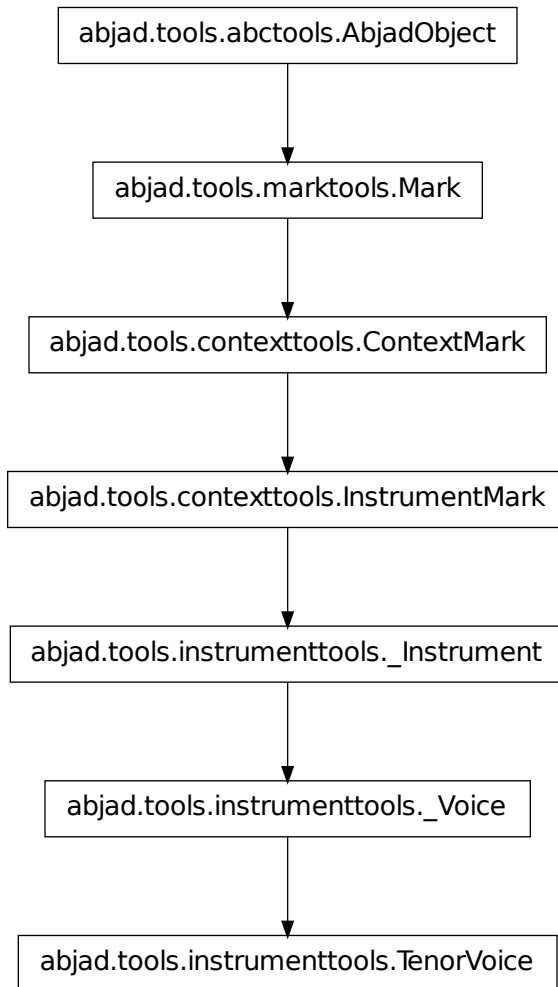
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

TenorTrombone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.TenorVoice



class `abjad.tools.instrumenttools.TenorVoice.TenorVoice`. **TenorVoice** (**kwargs)
 New in version 2.8. Abjad model of the tenor voice:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.TenorVoice()(staff)
TenorVoice()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Tenor voice }
  \set Staff.shortInstrumentName = \markup { Tenor }
  c'8
  d'8

```



```

        e' 8
        f' 8
    }

```

The tenor voice targets staff context by default.

Read-only Properties

TenorVoice.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorVoice.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

TenorVoice.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)

```

```

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

TenorVoice.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

TenorVoice.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

TenorVoice.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

TenorVoice.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

TenorVoice.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

TenorVoice.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

TenorVoice.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

TenorVoice.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

TenorVoice.all_clefs

Note: Inherited from `instrumenttools._Instrument`

`TenorVoice.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorVoice.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`TenorVoice.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`TenorVoice.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`TenorVoice.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`TenorVoice.short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`TenorVoice.sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

`TenorVoice.attach(start_component)`

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

`TenorVoice.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`TenorVoice.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`TenorVoice.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`TenorVoice.__call__(*args)`

Note: Inherited from `marktools.Mark`

`TenorVoice.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`TenorVoice.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`TenorVoice.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TenorVoice.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TenorVoice.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

TenorVoice.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorVoice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TenorVoice.__ne__(arg)

Note: Inherited from `marktools.Mark`

TenorVoice.__repr__()

Note: Inherited from `marktools.Mark`

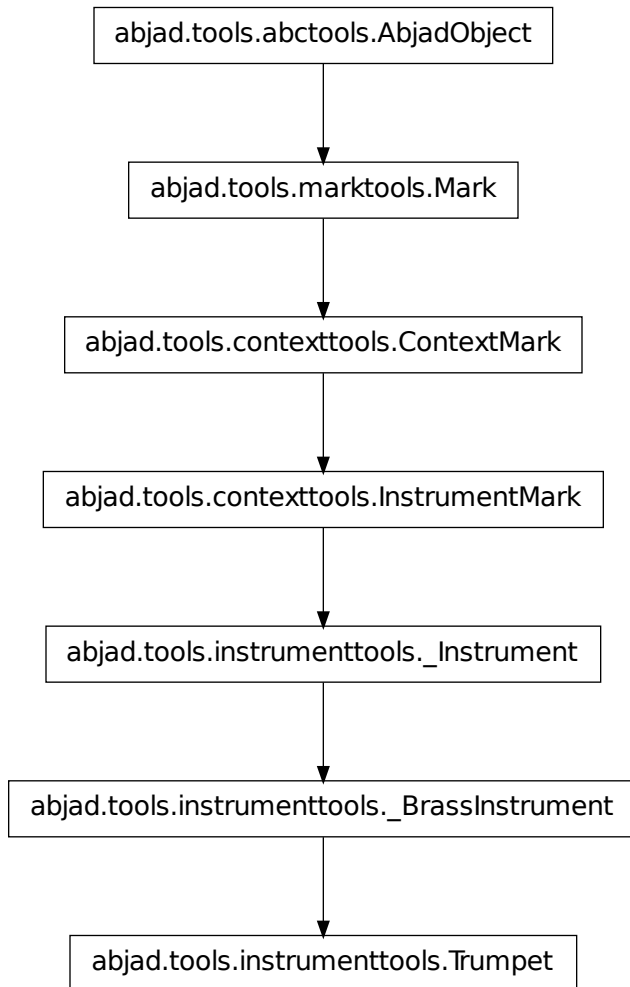
TenorVoice.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

TenorVoice.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Trumpet

class `abjad.tools.instrumenttools.Trumpet.Trumpet` **Trumpet** (**kwargs)
 New in version 2.0. Abjad model of the trumpet:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Trumpet()(staff)
Trumpet()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Trumpet }
  \set Staff.shortInstrumentName = \markup { Tp. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The trumpet targets staff context by default.

Read-only Properties

Trumpet.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Trumpet.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl'

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Trumpet.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

`Trumpet.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Trumpet.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Trumpet.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Trumpet.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Trumpet.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`Trumpet.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

Trumpet.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Trumpet.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Trumpet.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Trumpet.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Trumpet.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Trumpet.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Trumpet.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Trumpet.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Trumpet.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Trumpet.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Trumpet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Trumpet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Trumpet.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Trumpet.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Trumpet.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Trumpet.**__ne__**(arg)

Note: Inherited from `marktools.Mark`

Trumpet.**__repr__**()

Note: Inherited from `marktools.Mark`

Trumpet.**__setattr__**()

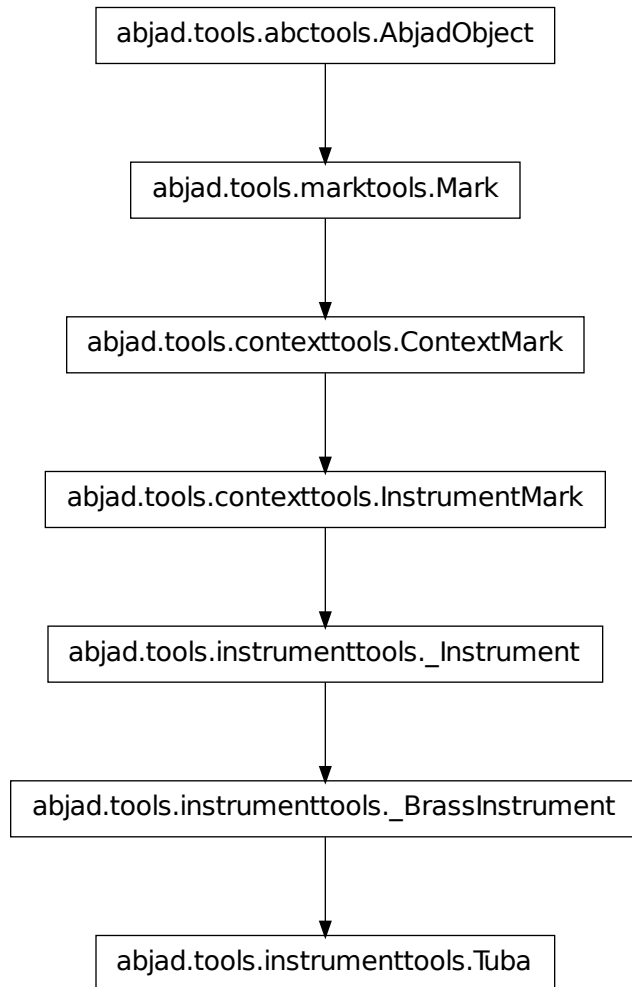
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Trumpet.**__str__**() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Tuba



class `abjad.tools.instrumenttools.Tuba.Tuba.Tuba` **Tuba** *(**kwargs)*
 New in version 2.0. Abjad model of the tuba:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('bass')(staff)
ClefMark('bass')(Staff{4})

abjad> instrumenttools.Tuba()(staff)
Tuba()(Staff{4})

abjad> f(staff)
\new Staff {
  \clef "bass"
  \set Staff.instrumentName = \markup { Tuba }

```

```
\set Staff.shortInstrumentName = \markup { Tb. }
c' 8
d' 8
e' 8
f' 8
}
```

The tuba targets staff context by default.

Read-only Properties

Tuba.**default_instrument_name**

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Tuba.**default_short_instrument_name**

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Tuba.**effective_context**

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Tuba.**format**

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl.
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Tuba.**interval_of_transposition**

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Tuba.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Tuba.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Tuba.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Tuba.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Tuba.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Tuba.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Tuba.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Tuba.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Tuba.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Tuba.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Tuba.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Tuba.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contextttools.InstrumentMark`

Tuba.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contextttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contextttools.InstrumentMark`

Tuba.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Tuba.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contextttools.ContextMark`

Tuba.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contextttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contextttools.ContextMark`

`Tuba.get_default_performer_name` (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Tuba.get_performer_names` ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Tuba.__call__` (*args)

Note: Inherited from `marktools.Mark`

`Tuba.__delattr__` (*args)

Note: Inherited from `marktools.Mark`

`Tuba.__eq__` (arg)

Note: Inherited from `contextttools.InstrumentMark`

`Tuba.__ge__` (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tuba.__gt__` (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Tuba.__hash__` ()

Note: Inherited from `contextttools.InstrumentMark`

Tuba. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Tuba. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Tuba. **__ne__** (*arg*)

Note: Inherited from `marktools.Mark`

Tuba. **__repr__** ()

Note: Inherited from `marktools.Mark`

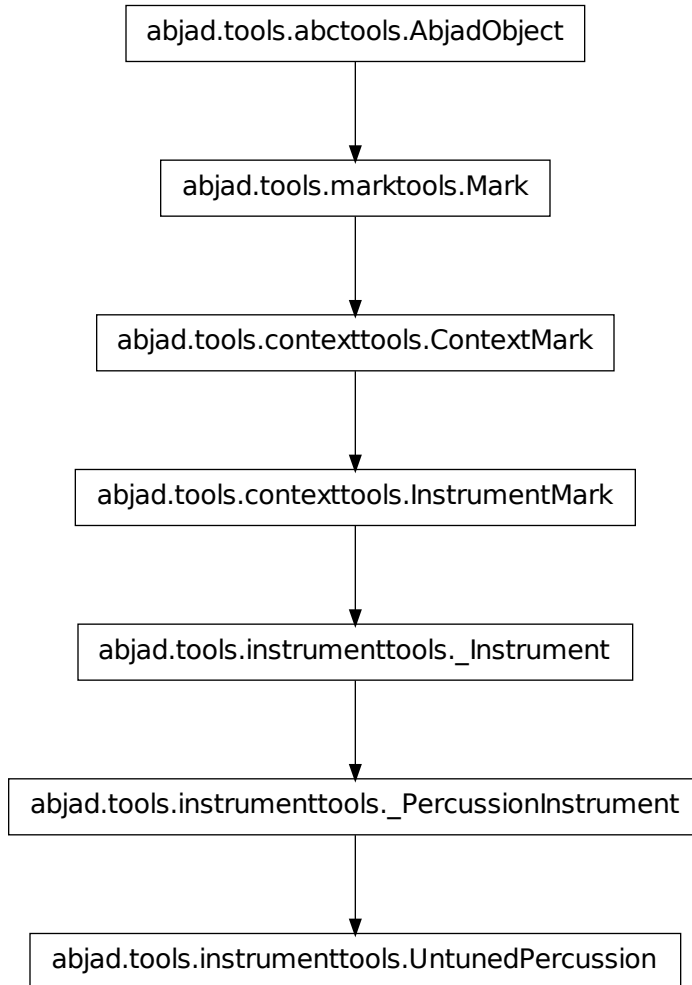
Tuba. **__setattr__** ()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Tuba. **__str__** () <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.UntunedPercussion

class `abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion` *(**kwargs)*
 New in version 2.0. Abjad model of untuned percussion:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.UntunedPercussion()(staff)
UntunedPercussion()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Untuned percussion }
  \set Staff.shortInstrumentName = \markup { Perc. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

Untuned percussion targets the staff context by default.

Read-only Properties

`UntunedPercussion.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`UntunedPercussion.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`UntunedPercussion.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`UntunedPercussion.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl'

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`UntunedPercussion.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`UntunedPercussion.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`UntunedPercussion.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`UntunedPercussion.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`UntunedPercussion.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`UntunedPercussion.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

UntunedPercussion.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

UntunedPercussion.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

UntunedPercussion.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:


```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

UntunedPercussion.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

UntunedPercussion.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

UntunedPercussion.**get_default_performer_name** (*locale=None*)

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

UntunedPercussion.**get_performer_names** ()

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

UntunedPercussion.**__call__** (*args)

Note: Inherited from `marktools.Mark`

UntunedPercussion.**__delattr__** (*args)

Note: Inherited from `marktools.Mark`

UntunedPercussion.**__eq__** (arg)

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.**__ge__** (arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

UntunedPercussion.**__gt__** (arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

UntunedPercussion.**__hash__** ()

Note: Inherited from `contexttools.InstrumentMark`

UntunedPercussion.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

UntunedPercussion.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

UntunedPercussion.__ne__(arg)

Note: Inherited from `marktools.Mark`

UntunedPercussion.__repr__()

Note: Inherited from `marktools.Mark`

UntunedPercussion.__setattr__()

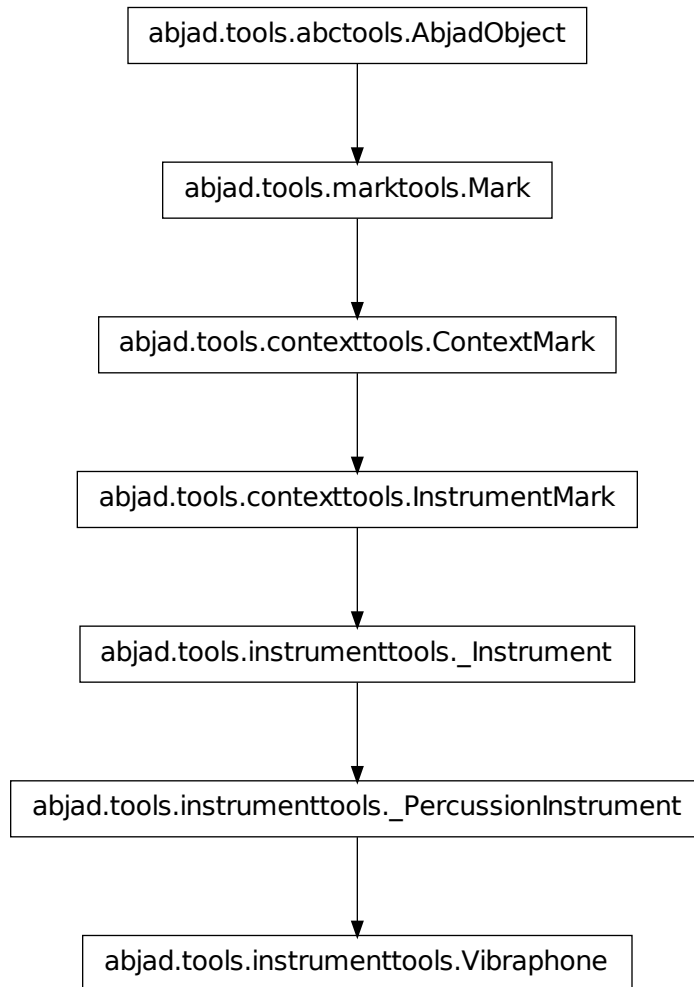
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

UntunedPercussion.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Vibraphone



class `abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone` (***kwargs*)
 New in version 2.0. Abjad model of the vibraphone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Vibraphone()(staff)
Vibraphone()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Vibraphone }
  \set Staff.shortInstrumentName = \markup { Vibr. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The vibraphone targets staff context by default.

Read-only Properties

Vibraphone.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Vibraphone.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Vibraphone.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Vibraphone.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Vibraphone.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Vibraphone.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Vibraphone.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Vibraphone.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Vibraphone.all_clefs

Note: Inherited from `instrumenttools._Instrument`

`Vibraphone.instrument_name`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Vibraphone.instrument_name_markup`

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

`Vibraphone.pitch_range`

Note: Inherited from `instrumenttools._Instrument`

`Vibraphone.primary_clefs`

Note: Inherited from `instrumenttools._Instrument`

`Vibraphone.short_instrument_name`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumenttools._Instrument`

Methods

Vibraphone.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Vibraphone.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Vibraphone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Vibraphone.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Vibraphone.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Vibraphone.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Vibraphone.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Vibraphone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Vibraphone.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Vibraphone.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Vibraphone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Vibraphone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Vibraphone.__ne__(arg)

Note: Inherited from `marktools.Mark`

Vibraphone.__repr__()

Note: Inherited from `marktools.Mark`

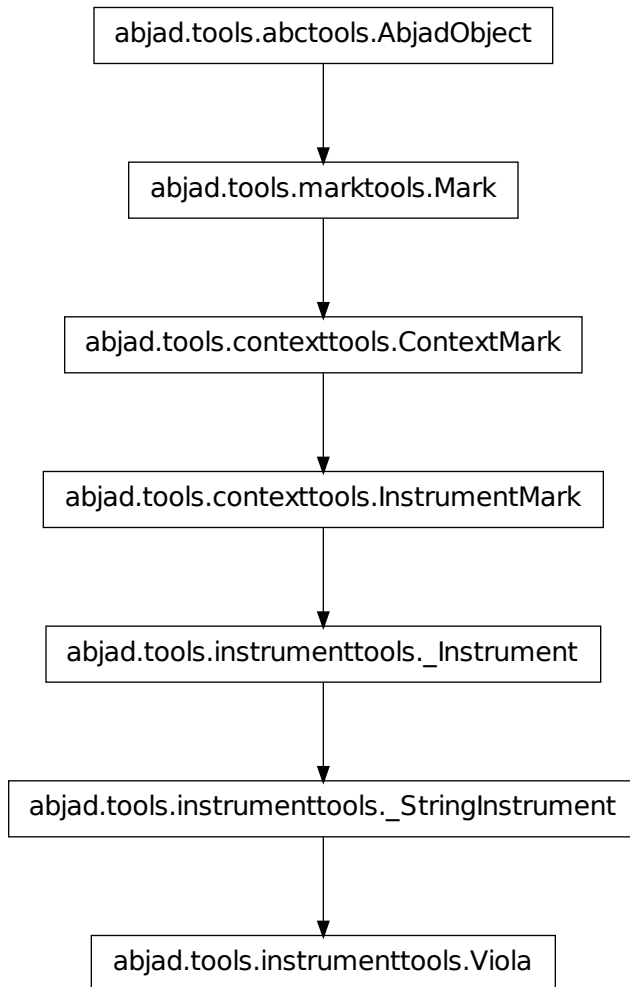
Vibraphone.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Vibraphone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Viola

class `abjad.tools.instrumenttools.Viola.Viola` **Viola** *(**kwargs)*
 New in version 2.0. Abjad model of the viola:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('alto')(staff)
ClefMark('alto')(Staff{4})

abjad> instrumenttools.Viola()(staff)
Viola()(Staff{4})

abjad> f(staff)
\new Staff {
    \clef "alto"
    \set Staff.instrumentName = \markup { Viola }
  
```

```

        \set Staff.shortInstrumentName = \markup { Va. }
        c' 8
        d' 8
        e' 8
        f' 8
    }

```

The viola targets staff context by default.

Read-only Properties

Viola.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Viola.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Viola.effective_context

Read-only reference to effective context of context mark:

```

abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.effective_context is None
True

```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Viola.format

Read-only LilyPond input format of instrument mark:

```

abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\set Staff.instrumentName = \markup { Flute }', '\set Staff.shortInstrumentName = \markup { Fl. }']

```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Viola.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Viola.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Viola.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Viola.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Viola.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Viola.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Viola.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Viola.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Viola.instrument_name

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Viola.instrument_name_markup

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Viola.pitch_range

Note: Inherited from `instrumenttools._Instrument`

Viola.primary_clefs

Note: Inherited from `instrumenttools._Instrument`

Viola.short_instrument_name

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contextttools.InstrumentMark`

Viola.**short_instrument_name_markup**

Get short instrument name:

```
abjad> instrument = contextttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contextttools.InstrumentMark`

Viola.**sounding_pitch_of_fingered_middle_c**

Note: Inherited from `instrumentttools._Instrument`

Methods

Viola.**attach**(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contextttools.ContextMark`

Viola.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contextttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Viola.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Viola.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Viola.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Viola.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Viola.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Viola.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Viola.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Viola.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Viola.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Viola.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Viola.**__ne__**(arg)

Note: Inherited from `marktools.Mark`

Viola.**__repr__**()

Note: Inherited from `marktools.Mark`

Viola.**__setattr__**()

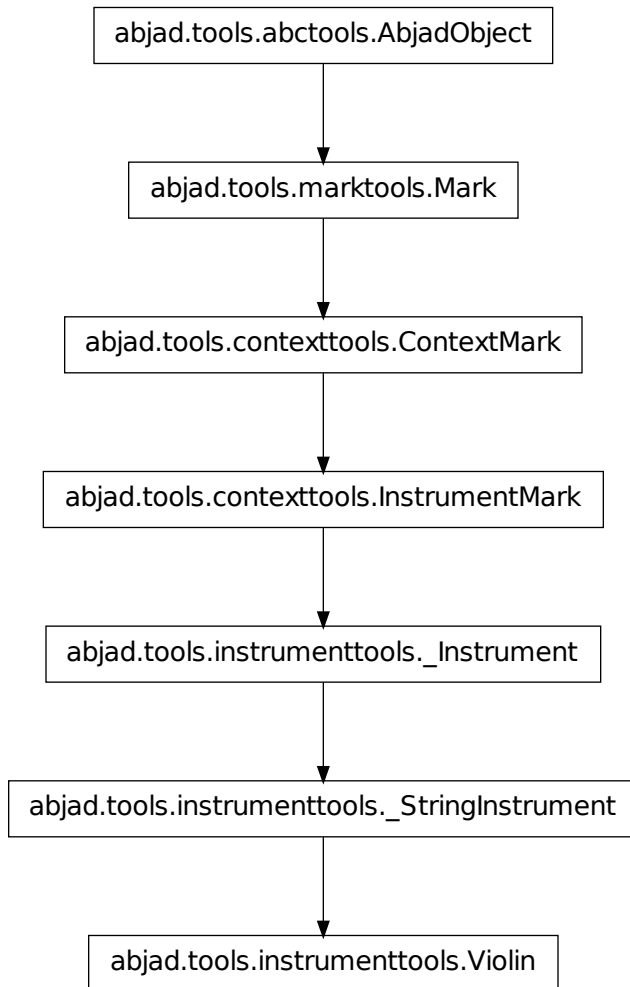
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Viola.**__str__**() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Violin



class `abjad.tools.instrumenttools.Violin.Violin`(**kwargs)

New in version 2.0. Abjad model of the violin:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The violin targets staff context by default.

Read-only Properties

Violin.default_instrument_name

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Violin.default_short_instrument_name

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Violin.effective_context

Read-only reference to effective context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Violin.format

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

Violin.interval_of_transposition

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

Violin.is_primary_instrument

Note: Inherited from `instrumenttools._Instrument`

Violin.is_secondary_instrument

Note: Inherited from `instrumenttools._Instrument`

Violin.is_transposing

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

Violin.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Violin.target_context

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

Violin.traditional_pitch_range

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

Violin.all_clefs

Note: Inherited from `instrumenttools._Instrument`

Violin.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Violin.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Violin.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Violin.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Violin.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Violin.`short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Violin.`sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

Violin.`attach` (*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Violin.`detach` ()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Violin.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are `'en-us'` and `'en-uk'`.

Note: Inherited from `instrumenttools._Instrument`

`Violin.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Violin.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Violin.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Violin.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Violin.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Violin.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Violin.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Violin.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Violin.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Violin.__ne__(arg)

Note: Inherited from `marktools.Mark`

Violin.__repr__()

Note: Inherited from `marktools.Mark`

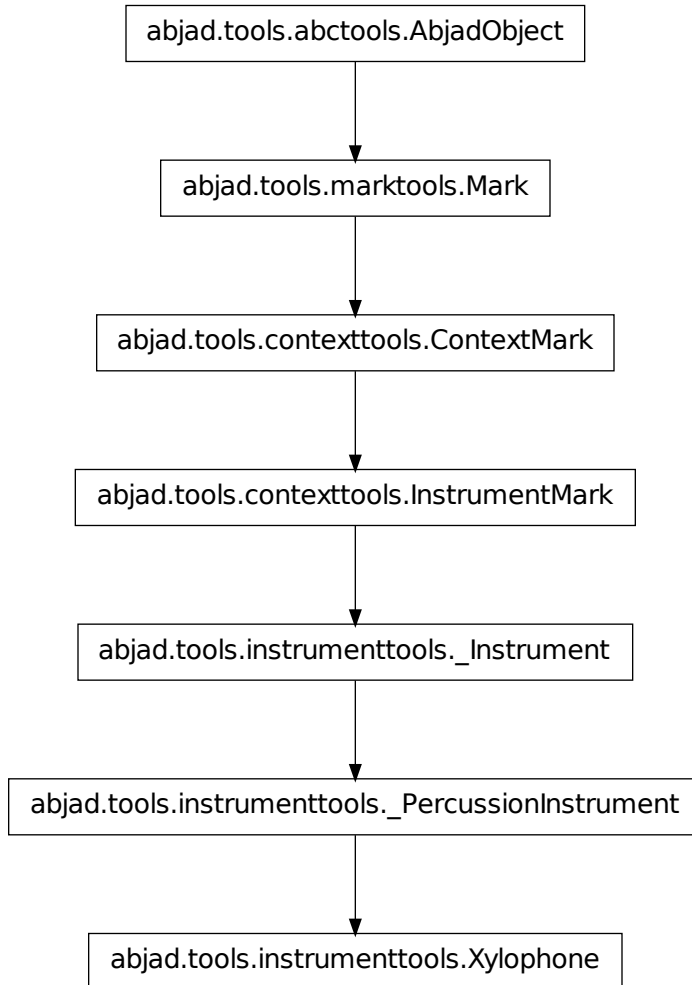
Violin.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Violin.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools.Xylophone

class `abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone` *(**kwargs)*
 New in version 2.0. Abjad model of the xylophone:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> instrumenttools.Xylophone()(staff)
Xylophone()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Xylophone }
  \set Staff.shortInstrumentName = \markup { Xyl. }
  c'8
  d'8

```

```
e' 8
f' 8
}
```

The xylophone targets staff context by default.

Read-only Properties

`Xylophone.default_instrument_name`

Read-only default instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Xylophone.default_short_instrument_name`

Read-only default short instrument name.

Return string.

Note: Inherited from `contexttools.InstrumentMark`

`Xylophone.effective_context`

Read-only reference to effective context of context mark:

```
abjad> note = Note("c' 4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.effective_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Xylophone.format`

Read-only LilyPond input format of instrument mark:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.format
['\\set Staff.instrumentName = \\markup { Flute }', '\\set Staff.shortInstrumentName = \\markup { Fl']
```

Return list.

Note: Inherited from `contexttools.InstrumentMark`

`Xylophone.interval_of_transposition`

Read-only interval of transposition.

Return melodic diatonic interval.

Note: Inherited from `instrumenttools._Instrument`

`Xylophone.is_primary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Xylophone.is_secondary_instrument`

Note: Inherited from `instrumenttools._Instrument`

`Xylophone.is_transposing`

True when instrument is transposing. False otherwise.

Return boolean.

Note: Inherited from `instrumenttools._Instrument`

`Xylophone.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

`Xylophone.target_context`

Read-only reference to target context of context mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)
```

```
abjad> context_mark.target_context is None
True
```

Return context mark or none.

Note: Inherited from `contexttools.ContextMark`

`Xylophone.traditional_pitch_range`

Read-only traditional pitch range.

Return pitch range.

Note: Inherited from `instrumenttools._Instrument`

Read/write Properties

`Xylophone.all_clefs`

Note: Inherited from `instrumenttools._Instrument`

Xylophone.**instrument_name**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name
'Flute'
```

Set instrument name:

```
abjad> instrument.instrument_name = 'Alto Flute'
abjad> instrument.instrument_name
'Alto Flute'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Xylophone.**instrument_name_markup**

Get instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.instrument_name_markup
Markup('Flute')
```

Set instrument name:

```
abjad> instrument.instrument_name_markup = 'Alto Flute'
abjad> instrument.instrument_name_markup
Markup('Alto Flute')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Xylophone.**pitch_range**

Note: Inherited from `instrumenttools._Instrument`

Xylophone.**primary_clefs**

Note: Inherited from `instrumenttools._Instrument`

Xylophone.**short_instrument_name**

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name
'Fl.'
```

Set short instrument name:

```
abjad> instrument.short_instrument_name = 'Alto Fl.'
abjad> instrument.short_instrument_name
'Alto Fl.'
```

Return string.

Note: Inherited from `contexttools.InstrumentMark`

Xylophone.`short_instrument_name_markup`

Get short instrument name:

```
abjad> instrument = contexttools.InstrumentMark('Flute', 'Fl.')
abjad> instrument.short_instrument_name_markup
Markup('Fl.')
```

Set short instrument name:

```
abjad> instrument.short_instrument_name_markup = 'Alto Fl.'
abjad> instrument.short_instrument_name_markup
Markup('Alto Fl.')
```

Return markup.

Note: Inherited from `contexttools.InstrumentMark`

Xylophone.`sounding_pitch_of_fingered_middle_c`

Note: Inherited from `instrumenttools._Instrument`

Methods

Xylophone.`attach`(*start_component*)

Make sure no context mark of same type is already attached to start component.

Note: Inherited from `contexttools.ContextMark`

Xylophone.`detach`()

Detach mark:

```
abjad> note = Note("c'4")
abjad> context_mark = contexttools.ContextMark()(note)

abjad> context_mark.start_component
Note("c'4")

abjad> context_mark.detach()
ContextMark()

abjad> context_mark.start_component is None
True
```

Return context mark.

Note: Inherited from `contexttools.ContextMark`

`Xylophone.get_default_performer_name(locale=None)`

New in version 2.5. Get default player name.

Available values for *locale* are 'en-us' and 'en-uk'.

Note: Inherited from `instrumenttools._Instrument`

`Xylophone.get_performer_names()`

New in version 2.5. Get performer names.

Note: Inherited from `instrumenttools._Instrument`

Special Methods

`Xylophone.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Xylophone.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Xylophone.__eq__(arg)`

Note: Inherited from `contexttools.InstrumentMark`

`Xylophone.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Xylophone.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Xylophone.__hash__()`

Note: Inherited from `contexttools.InstrumentMark`

Xylophone.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Xylophone.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Xylophone.__ne__(arg)

Note: Inherited from `marktools.Mark`

Xylophone.__repr__()

Note: Inherited from `marktools.Mark`

Xylophone.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Xylophone.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

instrumenttools functions

instrumenttools.default_instrument_name_to_instrument_class

`abjad.tools.instrumenttools.default_instrument_name_to_instrument_class.default_instrument_name_to_instrument_class`

New in version 2.5. Change *default_instrument_name* to class name:

```
abjad> instrumenttools.default_instrument_name_to_instrument_class('clarinet in E-flat')
<class 'abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet'>
```

Return class.

When *default_instrument_name* matches no instrument class:

```
abjad> instrumenttools.default_instrument_name_to_instrument_class('foo') is None
True
```

Return none.

instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_ranges

`abjad.tools.instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument`

New in version 2.0. Iterate notes and chords in *expr* outside traditional instrument ranges:

```
abjad> staff = Staff("c'8 r8 <d fs>8 r8")
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})

abjad> for note_or_chord in instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional
...     note_or_chord
Chord('<d fs>8')
```

Return generator.

instrumenttools.list_instrument_names

`abjad.tools.instrumenttools.list_instrument_names.list_instrument_names()`

New in version 2.5. List instrument names:

```
abjad> for instrument_name in instrumenttools.list_instrument_names():
...     instrument_name
...
'accordion'
'alto flute'
'alto saxophone'
'alto trombone'
'clarinet in B-flat'
'baritone saxophone'
'baritone voice'
'bass clarinet'
'bass flute'
'bass saxophone'
'bass trombone'
'bass voice'
'bassoon'
'cello'
'clarinet in A'
'contrabass'
'contrabass clarinet'
'contrabass flute'
'contrabass saxophone'
'contrabassoon'
'contralto voice'
'clarinet in E-flat'
'English horn'
'flute'
'horn'
'glockenspiel'
'guitar'
'harp'
'harpsichord'
'marimba'
'mezzo-soprano voice'
'oboe'
'piano'
'piccolo'
```



```

'sopranino saxophone'
'soprano saxophone'
'soprano voice'
'tenor saxophone'
'tenor trombone'
'tenor voice'
'trumpet'
'tuba'
'untuned percussion'
'vibraphone'
'viola'
'violin'
'xylophone'

```

Return list.

instrumenttools.list_instruments

`abjad.tools.instrumenttools.list_instruments.list_instruments` (*klases=None*)

New in version 2.5. List instruments in instrumenttools module:

```

abjad> for instrument in instrumenttools.list_instruments():
...     instrument
...
<class 'abjad.tools.instrumenttools.Accordion.Accordion.Accordion'>
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone'>
<class 'abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone'>
<class 'abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet'>
<class 'abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone'>
<class 'abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice'>
<class 'abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet'>
<class 'abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute'>
<class 'abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone'>
<class 'abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone'>
<class 'abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice'>
<class 'abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon'>
<class 'abjad.tools.instrumenttools.Cello.Cello.Cello'>
<class 'abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA'>
<class 'abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass'>
<class 'abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet'>
<class 'abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute'>
<class 'abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone'>
<class 'abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon'>
<class 'abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice'>
<class 'abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet'>
<class 'abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn'>
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
<class 'abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn'>
<class 'abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel'>
<class 'abjad.tools.instrumenttools.Guitar.Guitar.Guitar'>
<class 'abjad.tools.instrumenttools.Harp.Harp.Harp'>
<class 'abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord'>
<class 'abjad.tools.instrumenttools.Marimba.Marimba.Marimba'>
<class 'abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice'>
<class 'abjad.tools.instrumenttools.Oboe.Oboe.Oboe'>
<class 'abjad.tools.instrumenttools.Piano.Piano.Piano'>

```

```
<class 'abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo'>
<class 'abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone'>
<class 'abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone'>
<class 'abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice'>
<class 'abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone'>
<class 'abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone'>
<class 'abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice'>
<class 'abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet'>
<class 'abjad.tools.instrumenttools.Tuba.Tuba.Tuba'>
<class 'abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion'>
<class 'abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone'>
<class 'abjad.tools.instrumenttools.Viola.Viola.Viola'>
<class 'abjad.tools.instrumenttools.Violin.Violin.Violin'>
<class 'abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone'>
```

Return list.

instrumenttools.list_primary_instruments

`abjad.tools.instrumenttools.list_primary_instruments.list_primary_instruments()`

New in version 2.5. List primary instruments:

```
abjad> for primary_instrument in instrumenttools.list_primary_instruments():
...     primary_instrument
...
<class 'abjad.tools.instrumenttools.Accordion.Accordion.Accordion'>
<class 'abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone'>
<class 'abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet'>
<class 'abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice'>
<class 'abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice'>
<class 'abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon'>
<class 'abjad.tools.instrumenttools.Cello.Cello.Cello'>
<class 'abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass'>
<class 'abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice'>
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
<class 'abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn'>
<class 'abjad.tools.instrumenttools.Guitar.Guitar.Guitar'>
<class 'abjad.tools.instrumenttools.Harp.Harp.Harp'>
<class 'abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord'>
<class 'abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice'>
<class 'abjad.tools.instrumenttools.Oboe.Oboe.Oboe'>
<class 'abjad.tools.instrumenttools.Piano.Piano.Piano'>
<class 'abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice'>
<class 'abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone'>
<class 'abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice'>
<class 'abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet'>
<class 'abjad.tools.instrumenttools.Tuba.Tuba.Tuba'>
<class 'abjad.tools.instrumenttools.Viola.Viola.Viola'>
<class 'abjad.tools.instrumenttools.Violin.Violin.Violin'>
```

Return list

instrumenttools.list_secondary_instruments

`abjad.tools.instrumenttools.list_secondary_instruments.list_secondary_instruments()`

New in version 2.5. List secondary instruments:

```

abjad> for secondary_instrument in instrumenttools.list_secondary_instruments():
...     secondary_instrument
...
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone'>
<class 'abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone'>
<class 'abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet'>
<class 'abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute'>
<class 'abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone'>
<class 'abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone'>
<class 'abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA'>
<class 'abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet'>
<class 'abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute'>
<class 'abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone'>
<class 'abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon'>
<class 'abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet'>
<class 'abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn'>
<class 'abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel'>
<class 'abjad.tools.instrumenttools.Marimba.Marimba.Marimba'>
<class 'abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo'>
<class 'abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone'>
<class 'abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone'>
<class 'abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone'>
<class 'abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion'>
<class 'abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone'>
<class 'abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone'>

```

Return list

instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs

```
abjad.tools.instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs.notes_and_chords
```

New in version 2.0. True when notes and chords in *expr* are on expected clefs:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('treble')(staff)
ClefMark('treble')(Staff{4})
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})

abjad> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff)
True

```

False otherwise:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('alto')(staff)
ClefMark('alto')(Staff{4})
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})

abjad> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff)
False

```

Allow percussion clef when *percussion_clef_is_allowed* is true:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.ClefMark('percussion')(staff)
ClefMark('percussion')(Staff{4})
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})

abjad> f(staff)
\new Staff {
  \clef "percussion"
  \set Staff.instrumentName = \markup { Violin }
  \set Staff.shortInstrumentName = \markup { Vn. }
  c'8
  d'8
  e'8
  f'8
}
```

```
abjad> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff, percussion_clef_is_
True
```

Disallow percussion clef when *percussion_clef_is_allowed* is false:

```
abjad> instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs(staff, percussion_clef_is_
False
```

Return boolean.

instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges

```
abjad.tools.instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges
```

New in version 2.0. True when notes and chords in *expr* are within traditional instrument ranges:

```
abjad> staff = Staff("c'8 r8 <d' fs'>8 r8")
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})
```

```
abjad> instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges(staff)
True
```

False otherwise:

```
abjad> staff = Staff("c'8 r8 <d fs>8 r8")
abjad> instrumenttools.Violin()(staff)
Violin()(Staff{4})
```

```
abjad> instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges(staff)
False
```

Return boolean.

instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch

```
abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch
```

New in version 2.0. Transpose notes and chords in *expr* from sounding pitch to fingered pitch:

```

abjad> staff = Staff("<c' e' g'>4 d'4 r4 e'4")
abjad> instrumenttools.BFlatClarinet()(staff)
BFlatClarinet()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <c' e' g'>4
  d'4
  r4
  e'4
}

abjad> for leaf in staff.leaves:
...   leaf.written_pitch_indication_is_at_sounding_pitch = False

abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch(

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <bf d' f'>4
  c'4
  r4
  d'4
}

```

Return none.

instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch

abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch
 New in version 2.0. Transpose notes and chords in *expr* from sounding pitch to fingered pitch:

```

abjad> staff = Staff("<c' e' g'>4 d'4 r4 e'4")
abjad> instrumenttools.BFlatClarinet()(staff)
BFlatClarinet()(Staff{4})

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <c' e' g'>4
  d'4
  r4
  e'4
}

abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch(

abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Clarinet in B-flat }
  \set Staff.shortInstrumentName = \markup { Cl. in B-flat }
  <d' fs' a'>4
}

```

```

        e' 4
        r4
        fs' 4
    }

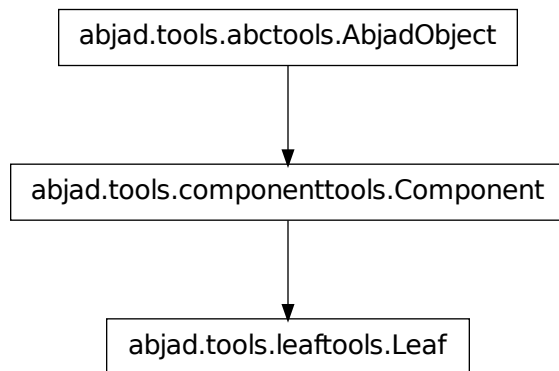
```

Return none.

leaftools

leaftools concrete classes

leaftools.Leaf



class `abjad.tools.leaftools.Leaf.Leaf`(*written_duration*, *duration_multiplier=None*)

Read-only Properties

`Leaf.duration_in_seconds`

`Leaf.format`

`Leaf.leaf_index`

`Leaf.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`Leaf.multiplied_duration`

`Leaf.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Leaf.**parent**

Note: Inherited from `componenttools.Component`

Leaf.**preprolated_duration**

Leaf.**prolated_duration**

Note: Inherited from `componenttools.Component`

Leaf.**prolation**

Note: Inherited from `componenttools.Component`

Leaf.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Leaf.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Leaf.**duration_multiplier**

Leaf.**written_duration**

Leaf.**written_pitch_indication_is_at_sounding_pitch**

Leaf.**written_pitch_indication_is_nonsemantic**

Special Methods

Leaf.**__and__**(arg)

Leaf.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Leaf.**__eq__**(arg)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Leaf.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Leaf.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Leaf.**__hash__**() $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

Leaf.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Leaf.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Leaf.**__mul__**(*n*)

Note: Inherited from `componenttools.Component`

Leaf.**__ne__**(*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Leaf.**__or__**(*arg*)

Leaf.**__repr__**()

Leaf.**__rmul__**(*n*)

Note: Inherited from `componenttools.Component`

```
Leaf.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
Leaf.__str__()  
Leaf.__sub__(arg)  
Leaf.__xor__(arg)
```

leaftools functions

leaftools.all_are_leaves

`abjad.tools.leaftools.all_are_leaves.all_are_leaves(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad leaves:

```
abjad> leaves = [Note("c'4"), Rest('r4'), Note("d'4")]
```

```
abjad> leaftools.all_are_leaves(leaves)  
True
```

True when *expr* is an empty sequence:

```
abjad> leaftools.all_are_leaves([])  
True
```

Otherwise false:

```
abjad> leaftools.all_are_leaves('foo')  
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration

`abjad.tools.leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration.`

New in version 1.1. Change *leaf* written duration to *written_duration* and preserve preprolated *leaf* duration:

```
abjad> note = Note("c'4")  
abjad> note.written_duration  
Duration(1, 4)  
abjad> note.preprolated_duration  
Duration(1, 4)
```

```
abjad> leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration(note, Durat  
Note("c'8. * 4/3")
```

```
abjad> note.written_duration
Duration(3, 16)
abjad> note.preprolated_duration
Duration(1, 4)
```

Add LilyPond multiplier where necessary.

Return *leaf*. Changed in version 2.0: Renamed from `leaftools.duration_rewrite()`.
`leaftools.change_written_leaf_duration_and_preserve_preprolated_leaf_duration()`.

leaftools.color_leaf

`abjad.tools.leaftools.color_leaf.color_leaf(leaf, color)`

New in version 2.0. Color note:

```
abjad> note = Note("c'4")

abjad> leaftools.color_leaf(note, 'red')
Note("c'4")

abjad> f(note)
\once \override Accidental #'color = #red
\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
c'4
```

Color rest:

```
abjad> rest = Rest('r4')

abjad> leaftools.color_leaf(rest, 'red')
Rest('r4')

abjad> f(rest)
\once \override Dots #'color = #red
\once \override Rest #'color = #red
r4
```

Color chord:

```
abjad> chord = Chord("<c' e' bf'>4")

abjad> leaftools.color_leaf(chord, 'red')
Chord("<c' e' bf'>4")

abjad> f(chord)
\once \override Accidental #'color = #red
\once \override Dots #'color = #red
\once \override NoteHead #'color = #red
<c' e' bf'>4
```

Return *leaf*.

leaftools.color_leaves_in_expr

`abjad.tools.leaftools.color_leaves_in_expr.color_leaves_in_expr(expr, color)`

New in version 2.0. Color leaves in *expr*:

```

abjad> staff = Staff([Note(1, (3, 16)), Rest((3, 16)), skiptools.Skip((3, 16)), Chord([0, 1, 9],
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(cs'8., r8., s8., <c' cs' a'>8.)
abjad> f(staff)
\new Staff {
    cs'8. [
    r8.
    s8.
    <c' cs' a'>8. ]
}

abjad> leaftools.color_leaves_in_expr(staff, 'red')

abjad> f(staff)
\new Staff {
    \once \override Accidental #'color = #red
    \once \override Dots #'color = #red
    \once \override NoteHead #'color = #red
    cs'8. [
    \once \override Dots #'color = #red
    \once \override Rest #'color = #red
    r8.
    s8.
    \once \override Accidental #'color = #red
    \once \override Dots #'color = #red
    \once \override NoteHead #'color = #red
    <c' cs' a'>8. ]
}

```

Return none.

leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf

```
abjad.tools.leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf.copy_written_c
```

New in version 2.0. Copy written duration and multiplier from *source_leaf* to *target_leaf*:

```

abjad> note = Note("c'4")
abjad> note.duration_multiplier = Duration(1, 2)
abjad> rest = Rest((1, 64))
abjad> leaftools.copy_written_duration_and_multiplier_from_leaf_to_leaf(note, rest)
Rest('r4 * 1/2')

```

Return *target_leaf*.

leaftools.divide_leaf_meiotically

```
abjad.tools.leaftools.divide_leaf_meiotically.divide_leaf_meiotically(leaf,
                                                                    n=2)
```

New in version 1.1. Divide *leaf* meiotically *n* times:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)

```

```
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

abjad> leaftools.divide_leaf_meiotically(staff[0], n = 4)

abjad> f(staff)
\new Staff {
    c'32 [
    c'32
    c'32
    c'32
    d'8
    e'8
    f'8 ]
}
```

Replace *leaf* with *n* new leaves.

Preserve parentage and spanners.

Allow divisions into only 1, 2, 4, 8, 16, ... and other nonnegative integer powers of 2.

Produce only leaves and never tuplets or other containers.

Return none.

leaftools.divide_leaves_in_expr_meiotically

```
abjad.tools.leaftools.divide_leaves_in_expr_meiotically.divide_leaves_in_expr_meiotically(  
    /
```

New in version 1.1. Divide leaves meiotically in *expr* *n* times:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

abjad> leaftools.divide_leaves_in_expr_meiotically(staff[2:], n = 4)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'32
    e'32
    e'32
    e'32
    f'32
    f'32
}
```

```
f' 32
f' 32 ]
}
```

Replace every leaf in *expr* with *n* new leaves.

Preserve parentage and spanners.

Allow divisions into only 1, 2, 4, 8, 16, ... and other nonnegative integer powers of 2.

Produce only leaves and never tuplets or other containers.

Return `none`. Changed in version 2.0: renamed `leaftools.meiose()` to `leaftools.divide_leaves_in_expr_meiotically()`.

leaftools.expr_has_leaf_with_dotted_written_duration

`abjad.tools.leaftools.expr_has_leaf_with_dotted_written_duration.expr_has_leaf_with_dotted_written_duration`
New in version 2.0. True when *expr* has at least one leaf with dotted written duration:

```
abjad> notes = notetools.make_notes([0], [(1, 16), (2, 16), (3, 16)])
abjad> leaftools.expr_has_leaf_with_dotted_written_duration(notes)
True
```

False otherwise:

```
abjad> notes = notetools.make_notes([0], [(1, 16), (2, 16), (4, 16)])
abjad> leaftools.expr_has_leaf_with_dotted_written_duration(notes)
False
```

Return boolean.

leaftools.fuse_leaves_big_endian

`abjad.tools.leaftools.fuse_leaves_big_endian.fuse_leaves_big_endian`(*leaves*)
New in version 1.1. Fuse thread-contiguous *leaves*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.fuse_leaves_big_endian(staff[1:])
[Note("d'4.") ]
abjad> f(staff)
\new Staff {
    c'8
    d'4.
}
```

Rewrite duration of first leaf in *leaves*.

Detach all leaves in *leaves* other than first leaf from score.

Return list of first leaf in *leaves*. Changed in version 2.0: renamed `fuse.leaves_by_reference()` to `leaftools.fuse_leaves_big_endian()`.

leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_notes

`abjad.tools.leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_notes.fuse_leaves_in_container_once_by_counts_into_big_endian_notes`

New in version 1.1. Fuse leaves in *container* once by *counts* into big-endian notes.

leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_rests

`abjad.tools.leaftools.fuse_leaves_in_container_once_by_counts_into_big_endian_rests.fuse_1`

New in version 1.1. Fuse leaves in *container* once by *counts* into big-endian rests.

leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_notes

`abjad.tools.leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_notes.fuse`

New in version 1.1. Fuse leaves in *container* once by *counts* into little-endian notes.

leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_rests

`abjad.tools.leaftools.fuse_leaves_in_container_once_by_counts_into_little_endian_rests.fuse`

New in version 1.1. Fuse leaves in *container* once by *counts* into little-endian rests.

leaftools.fuse_leaves_in_tie_chain_by_immediate_parent_big_endian

`abjad.tools.leaftools.fuse_leaves_in_tie_chain_by_immediate_parent_big_endian.fuse_leaves_`

New in version 1.1. Fuse leaves in *tie_chain* by immediate parent:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> tietools.TieSpanner(staff.leaves)
TieSpanner(c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8 ~
        c'8 ~
    }
    {
        \time 2/8
        c'8 ~
        c'8
    }
}

abjad> tie_chain = tietools.get_tie_chain(staff.leaves[0])
abjad> leaftools.fuse_leaves_in_tie_chain_by_immediate_parent_big_endian(tie_chain)
[[Note("c'4")], [Note("c'4")]]

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'4 ~
    }
    {
        \time 2/8
        c'4
    }
}
```

Return list of fused notes by parent. Changed in version 2.0: renamed `fuse.leaves_in_tie_chain()` to `leaftools.fuse_leaves_in_tie_chain_by_immediate_parent_big_endian()`.

`leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang`

`abjad.tools.leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang`

New in version 1.1. Fuse tied leaves in *components* once by *prolated_durations* without overhang:

```
abjad> staff = Staff(notetools.make_repeated_notes(8))
abjad> tietools.TieSpanner(staff.leaves)
TieSpanner(c'8, c'8, c'8, c'8, c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8 ~
    c'8
}

abjad> leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang(staff)

abjad> f(staff)
\new Staff {
    c'4. ~
    c'4. ~
    c'8 ~
    c'8
}
```

Return none. Changed in version 2.0: renamed `fuse.tied_leaves_by_prolated_durations()` to `leaftools.fuse_tied_leaves_in_components_once_by_prolated_durations_without_overhang()`.

`leaftools.get_composite_offset_difference_series_from_leaves_in_expr`

`abjad.tools.leaftools.get_composite_offset_difference_series_from_leaves_in_expr.get_composite_offset_difference_series`

New in version 2.0. Get composite offset difference series from leaves in *expr*:

```
abjad> staff_1 = Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeated_notes(4))])
abjad> staff_2 = Staff(notetools.make_repeated_notes(4))
abjad> score = Score([staff_1, staff_2])
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
\new Score <<
    \new Staff {
        \fraction \times 4/3 {
            c'8
            d'8
            e'8
        }
    }
}
```

```

        \new Staff {
            f'8
            g'8
            a'8
            b'8
        }
    >>
abjad> leaftools.get_composite_offset_difference_series_from_leaves_in_expr(score)
[Offset(1, 8), Offset(1, 24), Offset(1, 12), Offset(1, 12), Offset(1, 24), Offset(1, 8)]

```

Composite offset difference series defined equal to time intervals between unique start and stop offsets of leaves in *expr*.

Return list of fractions.

leaftools.get_composite_offset_series_from_leaves_in_expr

abjad.tools.leaftools.get_composite_offset_series_from_leaves_in_expr.get_composite_offset_series_from_leaves_in_expr
 New in version 2.0. Get composite offset series from leaves in *expr*:

```

abjad> staff_1 = Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeated_note(4))
abjad> staff_2 = Staff(notetools.make_repeated_note(4))
abjad> score = Score([staff_1, staff_2])
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(score)
abjad> f(score)
    \new Score <<
        \new Staff {
            \fraction \times 4/3 {
                c'8
                d'8
                e'8
            }
        }
        \new Staff {
            f'8
            g'8
            a'8
            b'8
        }
    >>
abjad> leaftools.get_composite_offset_series_from_leaves_in_expr(score)
[Offset(0, 1), Offset(1, 8), Offset(1, 6), Offset(1, 4), Offset(1, 3), Offset(3, 8), Offset(1, 2)]

```

Equal to list of unique start and stop offsets of leaves in *expr*.

Return list of fractions.

leaftools.get_leaf_at_index_in_measure_number_in_expr

abjad.tools.leaftools.get_leaf_at_index_in_measure_number_in_expr.get_leaf_at_index_in_measure_number_in_expr

New in version 2.0. Get leaf at *leaf_index* in *measure_number* in *expr*:


```

abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> f(t)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> leaftools.get_leaf_at_index_in_measure_number_in_expr(t, 2, 0)
Note("e'8")

```

Return leaf or none.

leaftools.get_leaf_in_expr_with_maximum_prolated_duration

`abjad.tools.leaftools.get_leaf_in_expr_with_maximum_prolated_duration.get_leaf_in_expr_with...`
 New in version 2.5. Get leaf in *expr* with maximum prolated duration:

```

abjad> staff = Staff("c'4.. d'16 e'4.. f'16")

abjad> leaftools.get_leaf_in_expr_with_maximum_prolated_duration(staff)
Note("c'4..")

```

When two leaves in *expr* are both of equally maximal prolated duration, return the first leaf encountered in forward iteration.

Return none when *expr* contains no leaves:

```

abjad> leaftools.get_leaf_in_expr_with_maximum_prolated_duration([]) is None
True

```

Return leaf.

leaftools.get_leaf_in_expr_with_minimum_prolated_duration

`abjad.tools.leaftools.get_leaf_in_expr_with_minimum_prolated_duration.get_leaf_in_expr_with...`
 New in version 2.5. Get leaf in *expr* with minimum prolated duration:

```

abjad> staff = Staff("c'4.. d'16 e'4.. f'16")

abjad> leaftools.get_leaf_in_expr_with_minimum_prolated_duration(staff)
Note("d'16")

```

When two leaves in *expr* are both of equally minimal prolated duration, return the first leaf encountered in forward iteration.

Return none when *expr* contains no leaves:

```
abjad> leaftools.get_leaf_in_expr_with_minimum_prolated_duration([]) is None
True
```

Return leaf.

leaftools.get_nth_leaf_in_expr

`abjad.tools.leaftools.get_nth_leaf_in_expr.get_nth_leaf_in_expr(expr, n=0)`

New in version 2.0. Get *n* th leaf in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> for n in range(6):
...     leaftools.get_nth_leaf_in_expr(staff, n)
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")
```

Read backwards for negative values of *n*.

```
abjad> leaftools.get_nth_leaf_in_expr(staff, -1)
Note("a'8")
```

Note: Because this function returns as soon as it finds instance *n* of *klases*, it is more efficient to call `leaftools.get_nth_leaf_in_expr(expr, 0)` than `expr.leaves[0]`. It is likewise more efficient to call `leaftools.get_nth_leaf_in_expr(expr, -1)` than `expr.leaves[-1]`.

Return leaf of none.

leaftools.get_nth_leaf_in_thread_from_leaf

`abjad.tools.leaftools.get_nth_leaf_in_thread_from_leaf.get_nth_leaf_in_thread_from_leaf(leaf, n=0)`

New in version 2.0. Get n th leaf in thread from *leaf*:

```
abjad> staff = Staff(2 * Voice("c'8 d'8 e'8 f'8"))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
  \new Voice {
    g'8
    a'8
    b'8
    c''8
  }
}

abjad> for n in range(8):
...     print n, leaftools.get_nth_leaf_in_thread_from_leaf(staff[0][0], n)
...
0 c'8
1 d'8
2 e'8
3 f'8
4 None
5 None
6 None
7 None
```

Return leaf or none.

leaftools.is_bar_line_crossing_leaf

`abjad.tools.leaftools.is_bar_line_crossing_leaf.is_bar_line_crossing_leaf(leaf)`

New in version 2.0. True when *leaf* crosses bar line:

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> t[2].written_duration *= 2
abjad> contexttools.TimeSignatureMark((2, 8), partial = Duration(1, 8))(t[2])
TimeSignatureMark((2, 8), partial=Duration(1, 8))(e'4)
abjad> f(t)
\new Staff {
  c'8
  d'8
  \partial 8
  \time 2/8
  e'4
  f'8
}
abjad> leaftools.is_bar_line_crossing_leaf(t.leaves[2])
```

True

Otherwise false:

```
abjad> leaftools.is_bar_line_crossing_leaf(t.leaves[3])
False
```

Return boolean.

leaftools.iterate_leaf_pairs_forward_in_expr

abjad.tools.leaftools.iterate_leaf_pairs_forward_in_expr.iterate_leaf_pairs_forward_in_expr

New in version 2.0. Iterate leaf pairs forward in *expr*:

```
abjad> score = Score([])
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'4")]
abjad> score.append(Staff(notes))
abjad> notes = [Note(x, (1, 4)) for x in [-12, -15, -17]]
abjad> score.append(Staff(notes))
abjad> contexttools.ClefMark('bass')(score[1])
ClefMark('bass')(Staff{3})
```

```
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
    g'4
  }
  \new Staff {
    \clef "bass"
    c4
    a,4
    g,4
  }
>>
```

```
abjad> for pair in leaftools.iterate_leaf_pairs_forward_in_expr(score):
...     pair
(Note("c'8"), Note('c4'))
(Note("c'8"), Note("d'8"))
(Note('c4'), Note("d'8"))
(Note("d'8"), Note("e'8"))
(Note("d'8"), Note('a,4'))
(Note('c4'), Note("e'8"))
(Note('c4'), Note('a,4'))
(Note("e'8"), Note('a,4'))
(Note("e'8"), Note("f'8"))
(Note('a,4'), Note("f'8"))
(Note("f'8"), Note("g'4"))
(Note("f'8"), Note('g,4'))
(Note('a,4'), Note("g'4"))
(Note('a,4'), Note('g,4'))
(Note("g'4"), Note('g,4'))
```

Iterate leaf pairs left-to-right and top-to-bottom.

Return generator.

leaftools.iterate_leaves_backward_in_expr

abjad.tools.leaftools.iterate_leaves_backward_in_expr(*expr*, *start*=*len(expr)*, *stop*=*0*)

New in version 2.0. Iterate leaves backward in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> for leaf in leaftools.iterate_leaves_backward_in_expr(staff):
...     leaf
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")
Note("d'8")
Note("c'8")
```

Use the optional *start* and *stop* keyword parameters to control the indices of iteration.

```
abjad> for leaf in leaftools.iterate_leaves_backward_in_expr(staff, start = 3):
...     leaf
...
Note("e'8")
Note("d'8")
Note("c'8")

abjad> for leaf in leaftools.iterate_leaves_backward_in_expr(staff, start = 0, stop = 3):
...     leaf
...
Note("a'8")
Note("g'8")
Note("f'8")
```

```
abjad> for leaf in leaftools.iterate_leaves_backward_in_expr(staff, start = 2, stop = 4):
...     leaf
...
Note("f'8")
Note("e'8")
```

Ignore threads.

Return generator.

leaftools.iterate_leaves_forward_in_expr

abjad.tools.leaftools.iterate_leaves_forward_in_expr.**iterate_leaves_forward_in_expr**(*expr*,
start=0,
stop=None

New in version 2.0. Iterate leaves forward in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
  }
  {
    \time 2/8
    g'8
    a'8
  }
}

abjad> for leaf in leaftools.iterate_leaves_forward_in_expr(staff):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")
```

Use the optional *start* and *stop* keyword parameters to control the start and stop indices of iteration.

```
abjad> for leaf in leaftools.iterate_leaves_forward_in_expr(staff, start = 3):
...     leaf
...
Note("f'8")
Note("g'8")
Note("a'8")
```

```

abjad> for leaf in leaftools.iterate_leaves_forward_in_expr(staff, start = 0, stop = 3):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")

abjad> for leaf in leaftools.iterate_leaves_forward_in_expr(staff, start = 2, stop = 4):
...     leaf
...
Note("e'8")
Note("f'8")

```

Ignore threads.

Return generator.

leaftools.iterate_notes_and_chords_backward_in_expr

abjad.tools.leaftools.iterate_notes_and_chords_backward_in_expr.**iterate_notes_and_chords_ba**

New in version 2.0. Iterate notes and chords backward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")
```

```

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

```

```

abjad> for leaf in leaftools.iterate_notes_and_chords_backward_in_expr(staff):
...     leaf
Chord("<d' f' b'>8")
Note("a'8")
Chord("<e' g' c''>8")

```

Ignore threads.

Return generator. Changed in version 2.0: renamed `pitchtools.iterate_notes_and_chords_backward_in_expr` to `leaftools.iterate_notes_and_chords_backward_in_expr()`.

leaftools.iterate_notes_and_chords_forward_in_expr

abjad.tools.leaftools.iterate_notes_and_chords_forward_in_expr.**iterate_notes_and_chords_fo**

New in version 2.0. Iterate notes and chords forward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")
```

```

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

abjad> for leaf in leaftools.iterate_notes_and_chords_forward_in_expr(staff):
...     leaf
Chord("<e' g' c''>8")
Note("a'8")
Chord("<d' f' b'>8")

```

Ignore threads.

Return generator. Changed in version 2.0: renamed `pitchtools.iterate_notes_and_chords_forward_in_expr()` to `leaftools.iterate_notes_and_chords_forward_in_expr()`.

leaftools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes

`abjad.tools.leaftools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes`

New in version 2.0. Label leaves in *expr* with inversion-equivalent chromatic interval classes:

```

abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)
abjad> leaftools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { 1 }
    cs''8 ^ \markup { 2 }
    b'8 ^ \markup { 2 }
    af8 ^ \markup { 2 }
    bf,8 ^ \markup { 1 }
    b,8 ^ \markup { 2 }
    a'8 ^ \markup { 1 }
    bf'8 ^ \markup { 4 }
    fs'8 ^ \markup { 1 }
    f'8
}

```

Return none.

leaftools.label_leaves_in_expr_with_leaf_depth

`abjad.tools.leaftools.label_leaves_in_expr_with_leaf_depth.label_leaves_in_expr_with_leaf_depth`

New in version 1.1. Label leaves in *expr* with leaf depth:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8")
abjad> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[-3:])
FixedDurationTuplet(1/4, [e'8, f'8, g'8])
abjad> leaftools.label_leaves_in_expr_with_leaf_depth(staff)
abjad> f(staff)
\new Staff {

```



```

c'8 _ \markup { \small 1 }
d'8 _ \markup { \small 1 }
\times 2/3 {
  e'8 _ \markup { \small 2 }
  f'8 _ \markup { \small 2 }
  g'8 _ \markup { \small 2 }
}

```

Changed in version 2.0: renamed `label.leaf_depth()` to `leaftools.label_leaves_in_expr_with_leaf_dep`.
Return none.

leaftools.label_leaves_in_expr_with_leaf_durations

`abjad.tools.leaftools.label_leaves_in_expr_with_leaf_durations.label_leaves_in_expr_with_leaf_durations`

New in version 1.1. Label leaves in *expr* with leaf durations:

```

abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
abjad> leaftools.label_leaves_in_expr_with_leaf_durations(tuplet)
abjad> f(tuplet)
\times 2/3 {
  c'8 _ \markup { \column { \small 1/8 \small 1/12 } }
  d'8 _ \markup { \column { \small 1/8 \small 1/12 } }
  e'8 _ \markup { \column { \small 1/8 \small 1/12 } }
}

```

Label both written duration and prolated duration.

Return none.

leaftools.label_leaves_in_expr_with_leaf_indices

`abjad.tools.leaftools.label_leaves_in_expr_with_leaf_indices.label_leaves_in_expr_with_leaf_indices`

New in version 2.0. Label leaves in *expr* with leaf indices:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.label_leaves_in_expr_with_leaf_indices(staff)
abjad> f(staff)
\new Staff {
  c'8 _ \markup { \small 0 }
  d'8 _ \markup { \small 1 }
  e'8 _ \markup { \small 2 }
  f'8 _ \markup { \small 3 }
}

```

Return none.

leaftools.label_leaves_in_expr_with_leaf_numbers

`abjad.tools.leaftools.label_leaves_in_expr_with_leaf_numbers.label_leaves_in_expr_with_leaf_numbers`

New in version 1.1. Label leaves in *expr* with leaf numbers:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.label_leaves_in_expr_with_leaf_numbers(staff)
abjad> f(staff)
\new Staff {
    c'8 _ \markup { \small 1 }
    d'8 _ \markup { \small 2 }
    e'8 _ \markup { \small 3 }
    f'8 _ \markup { \small 4 }
}
```

Number leaves starting from 1. Changed in version 2.0: renamed `label.leaf_numbers()` to `leaftools.label_leaves_in_expr_with_leaf_numbers()`. Return `none`.

`leaftools.label_leaves_in_expr_with_melodic_chromatic_interval_classes`

`abjad.tools.leaftools.label_leaves_in_expr_with_melodic_chromatic_interval_classes`. **label_leaves_in**

New in version 2.0. Label leaves in *expr* with melodic chromatic interval classes:

```
abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)]))
abjad> leaftools.label_leaves_in_expr_with_melodic_chromatic_interval_classes(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +1 }
    cs'''8 ^ \markup { -2 }
    b'8 ^ \markup { -2 }
    af8 ^ \markup { -10 }
    bf,8 ^ \markup { +1 }
    b,8 ^ \markup { +10 }
    a'8 ^ \markup { +1 }
    bf'8 ^ \markup { -4 }
    fs'8 ^ \markup { -1 }
    f'8
}
```

Return `none`.

`leaftools.label_leaves_in_expr_with_melodic_chromatic_intervals`

`abjad.tools.leaftools.label_leaves_in_expr_with_melodic_chromatic_intervals`. **label_leaves_in**

New in version 2.0. Label leaves in *expr* with melodic chromatic intervals:

```
abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)]))
abjad> leaftools.label_leaves_in_expr_with_melodic_chromatic_intervals(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +25 }
    cs'''8 ^ \markup { -14 }
    b'8 ^ \markup { -15 }
    af8 ^ \markup { -10 }
    bf,8 ^ \markup { +1 }
    b,8 ^ \markup { +22 }
    a'8 ^ \markup { +1 }
    bf'8 ^ \markup { -4 }
    fs'8 ^ \markup { -1 }
}
```

```

    f'8
}

```

Return none.

leaftools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes

abjad.tools.leaftools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes.**label_leaves_in_expr_with_melodic_counterpoint_interval_classes**

New in version 2.0. Label leaves in *expr* with melodic counterpoint interval classes:

```

abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
abjad> leaftools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +8 }
    cs'''8 ^ \markup { -2 }
    b'8 ^ \markup { -2 }
    af8 ^ \markup { -7 }
    bf,8 ^ \markup { +1 }
    b,8 ^ \markup { +7 }
    a'8 ^ \markup { +2 }
    bf'8 ^ \markup { -4 }
    fs'8 ^ \markup { +1 }
    f'8
}

```

Return none.

leaftools.label_leaves_in_expr_with_melodic_counterpoint_intervals

abjad.tools.leaftools.label_leaves_in_expr_with_melodic_counterpoint_intervals.**label_leaves_in_expr_with_melodic_counterpoint_intervals**

New in version 2.0. Label leaves in *expr* with melodic counterpoint intervals:

```

abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
abjad> leaftools.label_leaves_in_expr_with_melodic_counterpoint_intervals(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +15 }
    cs'''8 ^ \markup { -9 }
    b'8 ^ \markup { -9 }
    af8 ^ \markup { -7 }
    bf,8 ^ \markup { 1 }
    b,8 ^ \markup { +14 }
    a'8 ^ \markup { +2 }
    bf'8 ^ \markup { -4 }
    fs'8 ^ \markup { 1 }
    f'8
}

```

Return none.

leaftools.label_leaves_in_expr_with_melodic_diatonic_interval_classes

abjad.tools.leaftools.label_leaves_in_expr_with_melodic_diatonic_interval_classes.**label_leaves_in_expr_with_melodic_diatonic_interval_classes**

New in version 2.0. Label leaves in *expr* with melodic diatonic interval classes:

```
abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
abjad> leaftools.label_leaves_in_expr_with_melodic_diatonic_interval_classes(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +aug8 }
    cs'''8 ^ \markup { -M2 }
    b'8 ^ \markup { -aug2 }
    af8 ^ \markup { -m7 }
    bf,8 ^ \markup { aug1 }
    b,8 ^ \markup { +m7 }
    a'8 ^ \markup { +m2 }
    bf'8 ^ \markup { -dim4 }
    fs'8 ^ \markup { aug1 }
    f'8
}
```

Return none.

leaftools.label_leaves_in_expr_with_melodic_diatonic_intervals

abjad.tools.leaftools.label_leaves_in_expr_with_melodic_diatonic_intervals.**label_leaves_in_expr_with_melodic_diatonic_intervals**

New in version 2.0. Label leaves in *expr* with melodic diatonic intervals:

```
abjad> staff = Staff(notetools.make_notes([0, 25, 11, -4, -14, -13, 9, 10, 6, 5], [Duration(1, 8)])
abjad> leaftools.label_leaves_in_expr_with_melodic_diatonic_intervals(staff)
abjad> f(staff)
\new Staff {
    c'8 ^ \markup { +aug15 }
    cs'''8 ^ \markup { -M9 }
    b'8 ^ \markup { -aug9 }
    af8 ^ \markup { -m7 }
    bf,8 ^ \markup { +aug1 }
    b,8 ^ \markup { +m14 }
    a'8 ^ \markup { +m2 }
    bf'8 ^ \markup { -dim4 }
    fs'8 ^ \markup { -aug1 }
    f'8
}
```

Return none.

leaftools.label_leaves_in_expr_with_pitch_class_numbers

abjad.tools.leaftools.label_leaves_in_expr_with_pitch_class_numbers.**label_leaves_in_expr_with_pitch_class_numbers**

New in version 1.1. Label leaves in *expr* with pitch-class numbers:

```

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.label_leaves_in_expr_with_pitch_class_numbers(t)
abjad> print t.format
\new Staff {
    c'8 _ \markup { \small 0 }
    d'8 _ \markup { \small 2 }
    e'8 _ \markup { \small 4 }
    f'8 _ \markup { \small 5 }
}

```

When `color = True` call `color_note_head_by_numbered_chromatic_pitch_class_color_map()`.

```

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.label_leaves_in_expr_with_pitch_class_numbers(t, color = True, number = False)
abjad> print t.format
\new Staff {
    \once \override NoteHead #'color = #(x11-color 'red)
    c'8
    \once \override NoteHead #'color = #(x11-color 'orange)
    d'8
    \once \override NoteHead #'color = #(x11-color 'ForestGreen)
    e'8
    \once \override NoteHead #'color = #(x11-color 'MediumOrchid)
    f'8
}

```

You can set *number* and *color* at the same time. Changed in version 2.0: renamed `label.leaf_pcs()` to `leaftools.label_leaves_in_expr_with_pitch_class_numbers()`. Return `none`.

leaftools.label_leaves_in_expr_with_pitch_numbers

`abjad.tools.leaftools.label_leaves_in_expr_with_pitch_numbers.label_leaves_in_expr_with_pi`

New in version 1.1. Label leaves in *expr* with pitch numbers:

```

abjad> staff = Staff(leaftools.make_leaves([None, 12, [13, 14, 15], None], [(1, 4)]))
abjad> leaftools.label_leaves_in_expr_with_pitch_numbers(staff)
abjad> f(staff)
\new Staff {
    r4
    c''4 _ \markup { \small 12 }
    <cs'' d'' ef''>4 _ \markup { \column { \small 15 \small 14 \small 13 } }
    r4
}

```

Return `none`. Changed in version 2.0: renamed `label.leaf_pitch_numbers()` to `leaftools.label_leaves_in_expr_with_pitch_numbers()`.

leaftools.label_leaves_in_expr_with_prolated_leaf_duration

`abjad.tools.leaftools.label_leaves_in_expr_with_prolated_leaf_duration.label_leaves_in_expr`

New in version 1.1. Label leaves in *expr* with prolated leaf duration:

```

abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
abjad> leaftools.label_leaves_in_expr_with_prolated_leaf_duration(tuplet)

```

```
abjad> f(tuplet)
\times 2/3 {
    c'8 _ \markup { \small 1/12 }
    d'8 _ \markup { \small 1/12 }
    e'8 _ \markup { \small 1/12 }
}
```

Return none.

leaftools.label_leaves_in_expr_with_tuplet_depth

abjad.tools.leaftools.label_leaves_in_expr_with_tuplet_depth.label_leaves_in_expr_with_tuplet_depth

New in version 1.1. Label leaves in *expr* with tuplet depth:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8")
abjad> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[-3:])
FixedDurationTuplet(1/4, [e'8, f'8, g'8])
abjad> leaftools.label_leaves_in_expr_with_tuplet_depth(staff)
abjad> f(staff)
\new Staff {
    c'8 _ \markup { \small 0 }
    d'8 _ \markup { \small 0 }
    \times 2/3 {
        e'8 _ \markup { \small 1 }
        f'8 _ \markup { \small 1 }
        g'8 _ \markup { \small 1 }
    }
}
```

Return none. Changed in version 2.0: renamed `label.leaf_depth_tuplet()` to `leaftools.label_leaves_in_expr_with_tuplet_depth()`.

leaftools.label_leaves_in_expr_with_written_leaf_duration

abjad.tools.leaftools.label_leaves_in_expr_with_written_leaf_duration.label_leaves_in_expr_with_written_leaf_duration

New in version 1.1. Label leaves in *expr* with written leaf duration:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
abjad> leaftools.label_leaves_in_expr_with_leaf_durations(tuplet)
abjad> f(tuplet)
\times 2/3 {
    c'8 _ \markup { \column { \small 1/8 \small 1/12 } }
    d'8 _ \markup { \column { \small 1/8 \small 1/12 } }
    e'8 _ \markup { \column { \small 1/8 \small 1/12 } }
}
```

Return none.

leaftools.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration

abjad.tools.leaftools.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration

New in version 2.0. Change *leaf* to augmented tuplet with *n* notes of equal written duration:

```

abjad> for n in range(1, 11):
...     note = Note(0, (3, 16))
...     tuplet = leaftools.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration(note,
...     print tuplet
...
{@ 1:1 c'8. @}
{@ 1:1 c'16., c'16. @}
{@ 1:1 c'16, c'16, c'16 @}
{@ 1:1 c'32., c'32., c'32., c'32. @}
{@ 5:8 c'64., c'64., c'64., c'64., c'64. @}
{@ 1:1 c'32, c'32, c'32, c'32, c'32, c'32 @}
{@ 7:8 c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 1:1 c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 3:4 c'64, c'64, c'64, c'64, c'64, c'64, c'64, c'64 @}
{@ 5:8 c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128., c'128. @}

```

Return augmented fixed-duration tuplet.

leaftools.leaf_to_augmented_tuplet_with_proportions

```
abjad.tools.leaftools.leaf_to_augmented_tuplet_with_proportions.leaf_to_augmented_tuplet_w
```

New in version 2.0. Change *leaf* to augmented tuplet with *proportions*:

```

abjad> note = Note(0, (3, 16))
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1])
{@ 1:1 c'8. @}
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1, 2])
{@ 1:1 c'16, c'8 @}
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1, 2, 2])
{@ 5:8 c'64., c'32., c'32. @}
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1, 2, 2, 3])
{@ 2:3 c'64, c'32, c'32, c'32. @}
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1, 2, 2, 3, 3])
{@ 11:12 c'64, c'32, c'32, c'32., c'32. @}
abjad> print leaftools.leaf_to_augmented_tuplet_with_proportions(note, [1, 2, 2, 3, 3, 4])
{@ 5:8 c'128, c'64, c'64, c'64., c'64., c'32 @}

```

Return augmented fixed-duration tuplet.

leaftools.leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration

```
abjad.tools.leaftools.leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration.leaf
```

New in version 2.0. Change *leaf* to diminished tuplet with *n* notes of equal written duration:

```

abjad> for n in range(1, 11):
...     note = Note(0, (3, 16))
...     tuplet = leaftools.leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration(note)
...     print tuplet
...
{@ 1:1 c'8. @}
{@ 1:1 c'16., c'16. @}
{@ 1:1 c'16, c'16, c'16 @}

```

```
{@ 1:1 c'32., c'32., c'32., c'32. @}
{@ 5:4 c'32., c'32., c'32., c'32., c'32. @}
{@ 1:1 c'32, c'32, c'32, c'32, c'32, c'32 @}
{@ 7:4 c'32., c'32., c'32., c'32., c'32., c'32., c'32. @}
{@ 1:1 c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
{@ 3:2 c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32 @}
{@ 5:4 c'64., c'64., c'64., c'64., c'64., c'64., c'64., c'64. @}
```

Return diminished fixed-duration tuplet.

leaftools.leaf_to_diminished_tuplet_with_proportions

abjad.tools.leaftools.leaf_to_diminished_tuplet_with_proportions.leaf_to_diminished_tuplet.

New in version 2.0. Change *leaf* to diminished tuplet with *proportions*:

```
abjad> note = Note(0, (3, 16))
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1])
{@ 1:1 c'8. @}
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1, 2])
{@ 1:1 c'16, c'8 @}
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1, 2, 2])
{@ 5:4 c'32., c'16., c'16. @}
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1, 2, 2, 3])
{@ 4:3 c'32, c'16, c'16, c'16. @}
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1, 2, 2, 3, 3])
{@ 11:6 c'32, c'16, c'16, c'16., c'16. @}
abjad> print leaftools.leaf_to_diminished_tuplet_with_proportions(note, [1, 2, 2, 3, 3, 4])
{@ 5:4 c'64, c'32, c'32, c'32., c'32., c'16 @}
```

Return diminished fixed-duration tuplet.

leaftools.list_prolated_durations_of_leaves_in_expr

abjad.tools.leaftools.list_prolated_durations_of_leaves_in_expr.list_prolated_durations_of_leaves_in_expr.

New in version 2.0. List prolated durations of leaves in *expr*:

```
abjad> staff = Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2)
abjad> leaftools.list_prolated_durations_of_leaves_in_expr(staff)
[Duration(1, 12), Duration(1, 12), Duration(1, 12), Duration(1, 12), Duration(1, 12), Duration(1, 12)]
```

Return list of fractions.

leaftools.list_written_durations_of_leaves_in_expr

abjad.tools.leaftools.list_written_durations_of_leaves_in_expr.list_written_durations_of_leaves_in_expr.

New in version 2.0. List the written durations of leaves in *expr*:

```
abjad> staff = Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2)
abjad> leaftools.list_written_durations_of_leaves_in_expr(staff)
[Duration(1, 8), Duration(1, 8), Duration(1, 8), Duration(1, 8), Duration(1, 8), Duration(1, 8)]
```

Return list of fractions.

leaftools.make_leaves

`abjad.tools.leaftools.make_leaves`.**make_leaves** (*pitches*, *durations*, *direction*=*'big-endian'*, *tied_rests*=*False*)

New in version 1.1. Construct a list of notes, rests or chords.

Set *pitches* is a single pitch, or a list of pitches, or a tuple of pitches.

Integer pitches create notes.

```
abjad> leaftools.make_leaves([2, 4, 19], [(1, 4)])
[Note("d'4"), Note("e'4"), Note("g'4")]
```

Tuple pitches create chords.

```
abjad> leaftools.make_leaves([(0, 1, 2), (3, 4, 5), (6, 7, 8)], [(1, 4)])
[Chord("<c' cs' d'>4"), Chord("<ef' e' f'>4"), Chord("<fs' g' af'>4")]
```

Set *pitches* to a list of none to create rests.

```
abjad> leaftools.make_leaves([None, None, None, None], [(1, 8)])
[Rest('r8'), Rest('r8'), Rest('r8'), Rest('r8')]
```

You can mix and match pitch values.

```
abjad> leaftools.make_leaves([12, (1, 2, 3), None, 12], [(1, 4)])
[Note("c''4"), Chord("<cs' d' ef'>4"), Rest('r4'), Note("c''4")]
```

If the length of *pitches* is less than the length of *durations*, the function reads *durations* cyclically.

```
abjad> leaftools.make_leaves([13], [(1, 8), (1, 8), (1, 4), (1, 4)])
[Note("cs''8"), Note("cs''8"), Note("cs''4"), Note("cs''4")]
```

Set *durations* to a single duration, a list of duration, or a tuple of durations.

If the length of *durations* is less than the length of *pitches*, the function reads *pitches* cyclically.

```
abjad> leaftools.make_leaves([13, 14, 15, 16], [(1, 8)])
[Note("cs''8"), Note("d''8"), Note("ef''8"), Note("e''8")]
```

Duration values not of the form $m / 2 ** n$ return leaves nested inside a fixed-multiplier tuplet.

```
abjad> leaftools.make_leaves([14], [(1, 12), (1, 12), (1, 12)])
[Tuplet(2/3, [d''8, d''8, d''8])]
```

Set *direction* to *'little-endian'* to return tied leaf durations from least to greatest.

```
abjad> staff = Staff(leaftools.make_leaves([15], [(13, 16)], direction = 'little-endian'))
abjad> f(staff)
\new Staff {
    ef''16 ~
    ef''2.
}
```

Set *tied_rests* to true to return tied rests for durations like 5/16 and 9/16.

```
abjad> staff = Staff(leaftools.make_leaves([None], [(5, 16)], tied_rests = True))
abjad> f(staff)
\new Staff {
    r4 ~
    r16
}
```

Return list of leaves. Changed in version 2.0: renamed `construct.leaves()` to `leaftools.make_leaves()`.

`leaftools.make_leaves_from_note_value_signal`

`abjad.tools.leaftools.make_leaves_from_note_value_signal.make_leaves_from_note_value_signal`

New in version 2.0. Make leaves from *note_value_signal* and *denominator_of_signal*:

```
abjad> leaves = leaftools.make_leaves_from_note_value_signal([3, -3, 5, -5], 8)
abjad> staff = Staff(leaves)
```

```
abjad> f(staff)
\new Staff {
    c'4.
    r4.
    c'2 ~
    c'8
    r2
    r8
}
```

Interpret positive elements in *note_value_signal* as notes.

Interpret negative elements in *note_value_signal* as rests.

Set the pitch of all notes to middle C.

Return list of notes and / or rests.

`leaftools.remove_initial_rests_from_sequence`

`abjad.tools.leaftools.remove_initial_rests_from_sequence.remove_initial_rests_from_sequence`

New in version 2.0. Remove initial rests from *sequence*:

```
abjad> staff = Staff("r8 r8 c'8 d'8 r4 r4")
```

```
abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}
```

```
abjad> leaftools.remove_initial_rests_from_sequence(staff)
[Note("c'8"), Note("d'8"), Rest('r4'), Rest('r4')]
```

```

abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}

```

Return list.

leaftools.remove_leaf_and_shrink_durated_parent_containers

abjad.tools.leaftools.remove_leaf_and_shrink_durated_parent_containers.**remove_leaf_and_shrink_durated_parent_containers**

New in version 1.1. Remove *leaf* and shrink durated parent containers:

```

abjad> measure = Measure((4, 8), tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(measure)
abjad> spannertools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8, f'8, g'8, a'8)
abjad> f(measure)
{
    \time 4/8
    \times 2/3 {
        c'8 [
        d'8
        e'8
    ]
    \times 2/3 {
        f'8
        g'8
        a'8 ]
    }
}

abjad> leaftools.remove_leaf_and_shrink_durated_parent_containers(measure.leaves[0])

abjad> f(measure)
{
    \time 5/12
    \scaleDurations #'(2 . 3) {
        {
            d'8 [
            e'8
        ]
        {
            f'8
            g'8
            a'8 ]
        }
    }
}

```

Return none.

leaftools.remove_outer_rests_from_sequence

`abjad.tools.leaftools.remove_outer_rests_from_sequence.remove_outer_rests_from_sequence(sequence)`

New in version 2.0. Remove outer rests from *sequence*:

```
abjad> staff = Staff("r8 r8 c'8 d'8 r4 r4")

abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}

abjad> leaftools.remove_outer_rests_from_sequence(staff)
[Note("c'8"), Note("d'8")]

abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}
```

Return list.

leaftools.remove_terminal_rests_from_sequence

`abjad.tools.leaftools.remove_terminal_rests_from_sequence.remove_terminal_rests_from_sequence(sequence)`

New in version 2.0. Remove terminal rests from *sequence*:

```
abjad> staff = Staff("r8 r8 c'8 d'8 r4 r4")

abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
    d'8
    r4
    r4
}

abjad> leaftools.remove_terminal_rests_from_sequence(staff)
[Rest('r8'), Rest('r8'), Note("c'8"), Note("d'8")]

abjad> f(staff)
\new Staff {
    r8
    r8
    c'8
}
```

```

        d'8
        r4
        r4
    }

```

Return list.

leaftools.repeat_leaf_and_extend_spanners

abjad.tools.leaftools.repeat_leaf_and_extend_spanners.**repeat_leaf_and_extend_spanners**(*leaf*, *total*)

New in version 1.1. Repeat *leaf* and extend spanners:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

```

abjad> leaftools.repeat_leaf_and_extend_spanners(staff[0], total = 3)

```

```

abjad> f(staff)
\new Staff {
    c'8 [
    c'8
    c'8
    d'8
    e'8
    f'8 ]
}

```

Preserve *leaf* written duration.

Preserve parentage and spanners.

Return none. Changed in version 2.0: renamed `leaftools.clone_and_splice_leaf()` to `leaftools.repeat_leaf_and_extend_spanners()`.

leaftools.repeat_leaves_in_expr_and_extend_spanners

abjad.tools.leaftools.repeat_leaves_in_expr_and_extend_spanners.**repeat_leaves_in_expr_and**

New in version 1.1. Repeat leaves in *expr* and extend spanners:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> f(staff)
\new Staff {
    c'8 [

```

```

        d'8
        e'8
        f'8 ]
    }

abjad> result = leaftools.repeat_leaves_in_expr_and_extend_spanners(staff[2:], total = 3)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    e'8
    e'8
    f'8
    f'8
    f'8 ]
}

```

Preserve leaf written durations.

Preserve parentage and spanners.

Return `none`. Changed in version 2.0: renamed `leaftools.multiply()` to `leaftools.repeat_leaves_in_expr_and_extend_spanners()`.

leaftools.replace_leaves_in_expr_with_named_parallel_voices

`abjad.tools.leaftools.replace_leaves_in_expr_with_named_parallel_voices.replace_leaves_in_expr`

Replace leaves in *expr* with two parallel voices containing copies of leaves in *expr*, with the upper voice named *upper_name* and the lower voice named *lower_name*:

```

abjad> c = p('{ c c c c }')
abjad> f(c)
{
    c4
    c4
    c4
    c4
}

abjad> leaftools.replace_leaves_in_expr_with_named_parallel_voices(c.leaves[1:3], 'upper', 'lower',
([Note('c4'), Note('c4')], [Note('c4'), Note('c4')]))

abjad> f(c)
{
    c4
    <<
        \context Voice = "upper" {
            c4
            c4
        }
        \context Voice = "lower" {
            c4

```

```

        c4
    }
    >>
    c4
}

```

If leaves in *expr* have different immediate parents, parallel voices will be created in each parent:

```

abjad> c = p(r'{ c8 \times 2/3 { c8 c c } \times 4/5 { c16 c c c c } c8 }')
abjad> f(c)

```

```

{
  c8
  \times 2/3 {
    c8
    c8
    c8
  }
  \times 4/5 {
    c16
    c16
    c16
    c16
    c16
  }
  c8
}

```

```

abjad> leaftools.replace_leaves_in_expr_with_named_parallel_voices(c.leaves[2:7], 'upper', 'lower',
([Note('c8'), Note('c8'), Note('c16'), Note('c16'), Note('c16')], [Note('c8'), Note('c8'), Note('c8')]))

```

```

abjad> f(c)

```

```

{
  c8
  \times 2/3 {
    c8
    <<
      \context Voice = "upper" {
        c8
        c8
      }
      \context Voice = "lower" {
        c8
        c8
      }
    >>
  }
  \times 4/5 {
    <<
      \context Voice = "upper" {
        c16
        c16
        c16
      }
      \context Voice = "lower" {
        c16
        c16
        c16
      }
    >>
  }
}

```

```

        c16
        c16
    }
    c8
}

```

Returns a list leaves in upper voice, and a list of leaves in lower voice.

leaftools.replace_leaves_in_expr_with_parallel_voices

`abjad.tools.leaftools.replace_leaves_in_expr_with_parallel_voices.replace_leaves_in_expr_w`

Replace leaves in *expr* with two parallel voices containing copies of leaves in *expr*:

```

abjad> c = p('{ c c c c }')
abjad> f(c)
{
    c4
    c4
    c4
    c4
}

abjad> leaftools.replace_leaves_in_expr_with_parallel_voices(c.leaves[1:3])
([Note('c4'), Note('c4')], [Note('c4'), Note('c4')])

abjad> f(c)
{
    c4
    <<
        \new Voice {
            c4
            c4
        }
        \new Voice {
            c4
            c4
        }
    >>
    c4
}

```

If leaves in *expr* have different immediate parents, parallel voices will be created in each parent:

```

abjad> c = p(r'{ c8 \times 2/3 { c8 c c } \times 4/5 { c16 c c c c } c8 }')
abjad> f(c)
{
    c8
    \times 2/3 {
        c8
        c8
        c8
    }
    \times 4/5 {
        c16
        c16
        c16
        c16
    }
}

```



```

        c16
    }
    c8
}

abjad> leaftools.replace_leaves_in_expr_with_parallel_voices(c.leaves[2:7])
([Note('c8'), Note('c8'), Note('c16'), Note('c16'), Note('c16')], [Note('c8'), Note('c8'), Note('c8')])

abjad> f(c)
{
    c8
    \times 2/3 {
        c8
        <<
            \new Voice {
                c8
                c8
            }
            \new Voice {
                c8
                c8
            }
        >>
    }
    \times 4/5 {
        <<
            \new Voice {
                c16
                c16
                c16
            }
            \new Voice {
                c16
                c16
                c16
            }
        >>
        c16
        c16
    }
    c8
}

```

Returns a list leaves in upper voice, and a list of leaves in lower voice.

leaftools.scale_preprolated_leaf_duration

abjad.tools.leaftools.scale_preprolated_leaf_duration.**scale_preprolated_leaf_duration**(*leaf*, *multiplier*)

New in version 1.1. Scale preprolated *leaf* leaf duration by dotted *multiplier*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(3, 2))

```

```
[Note("d'8.")]  
abjad> f(staff)  
\new Staff {  
    c'8 [  
        d'8.  
        e'8  
        f'8 ]  
}
```

Scale preprolated *leaf* duration by tied *multiplier*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")  
abjad> spannertools.BeamSpanner(staff.leaves)  
BeamSpanner(c'8, d'8, e'8, f'8)  
abjad> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(5, 4))  
[Note("d'8"), Note("d'32")]  
abjad> f(staff)  
\new Staff {  
    c'8 [  
        d'8 ~  
        d'32  
        e'8  
        f'8 ]  
}
```

Scale preprolated *leaf* duration by nonbinary *multiplier*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")  
abjad> spannertools.BeamSpanner(staff.leaves)  
BeamSpanner(c'8, d'8, e'8, f'8)  
abjad> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(2, 3))  
[Note("d'8")]  
abjad> f(staff)  
\new Staff {  
    c'8 [  
        \times 2/3 {  
            d'8  
        }  
        e'8  
        f'8 ]  
}
```

Scale preprolated *leaf* duration by tied nonbinary *multiplier*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")  
abjad> spannertools.BeamSpanner(staff.leaves)  
BeamSpanner(c'8, d'8, e'8, f'8)  
abjad> leaftools.scale_preprolated_leaf_duration(staff[1], Duration(5, 6))  
[Note("d'8"), Note("d'32")]  
abjad> f(staff)  
\new Staff {  
    c'8 [  
        \times 2/3 {  
            d'8 ~  
            d'32  
        }  
        e'8  
        f'8 ]  
}
```

Return *leaf*. Changed in version 2.0: renamed from `leaftools.duration_scale()`.
`leaftools.scale_preprolated_leaf_duration()`.

`leaftools.set_preprolated_leaf_duration`

`abjad.tools.leaftools.set_preprolated_leaf_duration.set_preprolated_leaf_duration` (*leaf*,
new_preprolated)

New in version 1.1. Set preprolated *leaf* duration:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> leaftools.set_preprolated_leaf_duration(staff[1], Duration(3, 16))
[Note("d'8.") ]
abjad> f(staff)
\new Staff {
  c'8 [
  d'8.
  e'8
  f'8 ]
}
```

Set tied preprolated *leaf* duration:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> leaftools.set_preprolated_leaf_duration(staff[1], Duration(5, 32))
[Note("d'8"), Note("d'32")]
abjad> f(staff)
\new Staff {
  c'8 [
  d'8 ~
  d'32
  e'8
  f'8 ]
}
```

Set nonbinary preprolated *leaf* duration:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> leaftools.set_preprolated_leaf_duration(staff[1], Duration(1, 12))
[Note("d'8")]
abjad> f(staff)
\new Staff {
  c'8 [
  \times 2/3 {
    d'8
  }
  e'8
  f'8 ]
}
```

Set tied nonbinary preprolated *leaf* duration:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'8, d'8, e'8, f'8)
abjad> leaftools.set_preprolated_leaf_duration(staff[1], Duration(5, 48))
[Note("d'8"), Note("d'32")]
abjad> f(staff)
\new Staff {
  c'8 [
    \times 2/3 {
      d'8 ~
      d'32
    }
  e'8
  f'8 ]
}

```

Set preprolated *leaf* duration with LilyPond multiplier:

```

abjad> note = Note(0, (1, 8))
abjad> note.duration_multiplier = Duration(1, 2)
abjad> leaftools.set_preprolated_leaf_duration(note, Duration(5, 48))
[Note("c'8 * 5/6")]
abjad> f(note)
c'8 * 5/6

```

Return list of *leaf* and leaves newly tied to *leaf*. Changed in version 2.0: renamed `leaftools.change_leaf_preprolated_duration()` to `leaftools.set_preprolated_leaf_duration()`.

leaftools.show_leaves

`abjad.tools.leaftools.show_leaves.show_leaves` (*leaves*, *template=None*, *sup-press_pdf=False*)

New in version 2.0. Show *leaves* in temporary piano staff score:

```

abjad> leaves = leaftools.make_leaves([None, 1, (-24, -22, 7, 21), None], (1, 4))
abjad> score = leaftools.show_leaves(leaves) # doctest: +SKIP
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      r4
      cs'4
      <g' a''>4
      r4
    }
    \context Staff = "bass" {
      \clef "bass"
      r4
      r4
      <c, d,>4
      r4
    }
  >>
>>

```

Useful when working with notes, rests, chords not yet added to score.

Return temporary piano staff score.

leaftools.split_leaf_at_prolated_duration_and_rest_right_half

abjad.tools.leaftools.split_leaf_at_prolated_duration_and_rest_right_half.**split_leaf_at_pro**

New in version 1.1. Split *leaf* at *prolated_duration* and rest right half:

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.SlurSpanner(t[:])
SlurSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}

abjad> leaftools.split_leaf_at_prolated_duration_and_rest_right_half(t.leaves[1], (1, 32))
([Note("d'32")], [Note("d'16.")])

abjad> f(t)
\new Staff {
    c'8 (
    d'32
    r16.
    e'8
    f'8 )
}
```

Return list of leaves to left of *prolated_duration* together with list of leaves to right of *prolated_duration*. Changed in version 2.0: renamed `leaftools.shorten()` to `leaftools.split_leaf_at_prolated_duration_and_rest_right_half()`.

leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence

abjad.tools.leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence.**yield_groups_of_m**

New in version 2.0. Yield groups of mixed notes and chords in *sequence*:

```
abjad> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}
```

```

    <c' ' e' '>8
}

abjad> for group in leaftools.yield_groups_of_mixed_notes_and_chords_in_sequence(staff):
...     group
...
(Note("c'8"), Note("d'8"))
(Chord("<e' g'>8"), Chord("<f' a'>8"), Note("g'8"), Note("a'8"))
(Chord("<b' d'>8"), Chord("<c' ' e' '>8"))

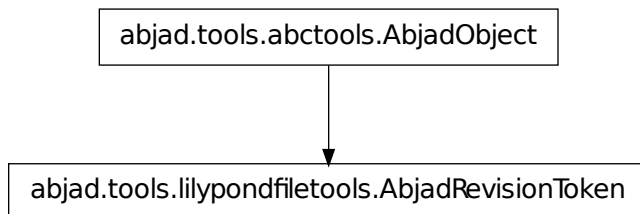
```

Return generator.

lilypondfiletools

lilypondfiletools concrete classes

lilypondfiletools.AbjadRevisionToken



class `abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken`. **AbjadRevisionToken**
 New in version 2.0. Abjad version token:

```

abjad> lilypondfiletools.AbjadRevisionToken()
AbjadRevisionToken(Abjad revision ...)

```

Return Abjad version token.

Read-only Properties

`AbjadRevisionToken.format`

Format contribution of Abjad version token:

```

abjad> lilypondfiletools.AbjadRevisionToken().format
'Abjad revision ...'

```

Return string.

Special Methods

`AbjadRevisionToken.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`AbjadRevisionToken.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`AbjadRevisionToken.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbjadRevisionToken.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

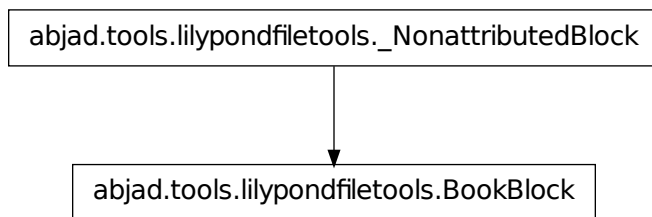
`AbjadRevisionToken.__repr__()`
`AbjadRevisionToken.__setattr__()`
`x.__setattr__('name', value)` $\leq \Rightarrow$ `x.name = value`

Note: Inherited from `__builtin__.object`

`AbjadRevisionToken.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

`lilypondfiletools.BookBlock`



class `abjad.tools.lilypondfiletools.BookBlock.BookBlock`. **BookBlock**

New in version 2.0. Abjad model of LilyPond input file book block:

```
abjad> book_block = lilypondfiletools.BookBlock()
```

```
abjad> book_block
BookBlock()
```

```
abjad> f(book_block)
\book {}
```

Return book block.

Read-only Properties

`BookBlock.format`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Read/write Properties

`BookBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Methods

`BookBlock.append()`

`L.append(object)` – append object to end

Note: Inherited from `__builtin__.list`

`BookBlock.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`BookBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Note: Inherited from `__builtin__.list`

`BookBlock.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`BookBlock.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`BookBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`BookBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`BookBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`BookBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`BookBlock.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`BookBlock.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.list`

`BookBlock.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`BookBlock.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`BookBlock.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`BookBlock.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`BookBlock.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`BookBlock.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`BookBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

BookBlock.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.list`

BookBlock.__iadd__()
 x.__iadd__(y) <==> x+=y

Note: Inherited from `__builtin__.list`

BookBlock.__imul__()
 x.__imul__(y) <==> x*=y

Note: Inherited from `__builtin__.list`

BookBlock.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.list`

BookBlock.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.list`

BookBlock.__len__() <==> len(x)

Note: Inherited from `__builtin__.list`

BookBlock.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.list`

BookBlock.__mul__()
 x.__mul__(n) <==> x*n

Note: Inherited from `__builtin__.list`

BookBlock.__ne__()
 x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.list`

BookBlock.__repr__()

Note: Inherited from `lilypondfiletools._NonattributedBlock`

`BookBlock.__reversed__()`

`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`BookBlock.__rmul__()`

`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`BookBlock.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BookBlock.__setitem__()`

`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

`BookBlock.__setslice__()`

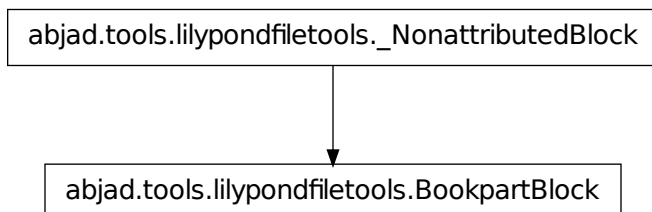
`x.__setslice__(i, j, y) <==> x[i:j]=y`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`BookBlock.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

lilypondfiletools.BookpartBlock

class `abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock` **BookpartBlock**
 New in version 2.0. Abjad model of LilyPond input file bookpart block:

```

abjad> bookpart_block = lilypondfiletools.BookpartBlock()

abjad> bookpart_block
BookpartBlock()

abjad> f(bookpart_block)
\bookpart {}
  
```

Return bookpart block.

Read-only Properties

`BookpartBlock.format`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Read/write Properties

`BookpartBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Methods

`BookpartBlock.append()`
`L.append(object)` – append object to end

Note: Inherited from `__builtin__.list`

`BookpartBlock.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`BookpartBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Note: Inherited from `__builtin__.list`

`BookpartBlock.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`BookpartBlock.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`BookpartBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`BookpartBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`BookpartBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`BookpartBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`BookpartBlock.__add__()`

`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__contains__()`

`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`BookpartBlock.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`BookpartBlock.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`BookpartBlock.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`BookpartBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

BookpartBlock.__**imul**__()
x.__imul__(y) <==> x*=y

Note: Inherited from `__builtin__.list`

BookpartBlock.__**iter**__() <==> iter(x)

Note: Inherited from `__builtin__.list`

BookpartBlock.__**le**__()
x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.list`

BookpartBlock.__**len**__() <==> len(x)

Note: Inherited from `__builtin__.list`

BookpartBlock.__**lt**__()
x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.list`

BookpartBlock.__**mul**__()
x.__mul__(n) <==> x*n

Note: Inherited from `__builtin__.list`

BookpartBlock.__**ne**__()
x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.list`

BookpartBlock.__**repr**__()

Note: Inherited from `lilypondfiletools._NonattributedBlock`

BookpartBlock.__**reversed**__()
L.__reversed__() – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

BookpartBlock.__**rmul**__()
x.__rmul__(n) <==> n*x

Note: Inherited from `__builtin__.list`

```
BookpartBlock.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
BookpartBlock.__setitem__()
x.__setitem__(i, y) <==> x[i]=y
```

Note: Inherited from `__builtin__.list`

```
BookpartBlock.__setslice__()
x.__setslice__(i, j, y) <==> x[i:j]=y
```

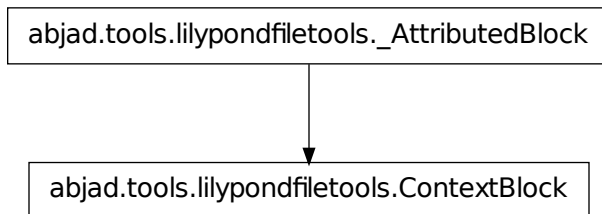
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
BookpartBlock.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

lilypondfiletools.ContextBlock



class `abjad.tools.lilypondfiletools.ContextBlock.ContextBlock(context_name=None)`
 New in version 2.5. Abjad model of LilyPond input file context block:

```
abjad> context_block = lilypondfiletools.ContextBlock()

abjad> context_block
ContextBlock()

abjad> context_block.context_name = 'Score'
abjad> context_block.override.bar_number.transparent = True
```

```

abjad> context_block.override.time_signature.break_visibility = schemetools.Scheme('end-of-line-
abjad> context_block.set.proportionalNotationDuration = schemetools.SchemeMoment((1, 45))

abjad> f(context_block)
\context {
  \Score
  \override BarNumber #'transparent = ##t
  \override TimeSignature #'break-visibility = #end-of-line-invisible
  proportionalNotationDuration = #(ly:make-moment 1 45)
}

```

Return context block.

Read-only Properties

`ContextBlock.format`

Note: Inherited from `lilypondfiletools._AttributedBlock`

`ContextBlock.override`

Read-only reference to LilyPond grob override component plug-in.

`ContextBlock.set`

Read-only reference LilyPond context setting component plug-in.

Read/write Properties

`ContextBlock.context_name`

Read / write context name.

`ContextBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Special Methods

`ContextBlock.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ContextBlock.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

`ContextBlock.__repr__()`

Note: Inherited from `lilypondfiletools._AttributedBlock`

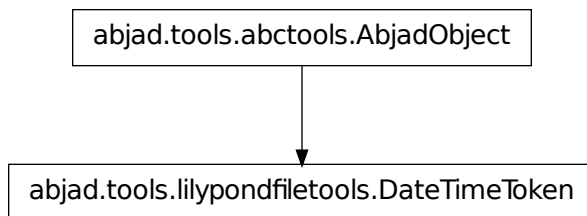
```
ContextBlock.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
ContextBlock.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

`lilypondfiletools.DateTimeToken`



class `abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken`**.DateTimeToken**

New in version 2.0. Date time token:

```
abjad> lilypondfiletools.DateTimeToken()  
DateTimeToken(...)
```

Return date / time token.

Read-only Properties

`DateTimeToken.format`

Format contribution of date time token:

```
abjad> lilypondfiletools.DateTimeToken().format  
'...'
```

Return string.

Special Methods

```
DateTimeToken.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
DateTimeToken.__eq__(arg)  
True when id(self) equals id(arg).
```

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DateTimeToken.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DateTimeToken.__repr__()`

`DateTimeToken.__setattr__()`

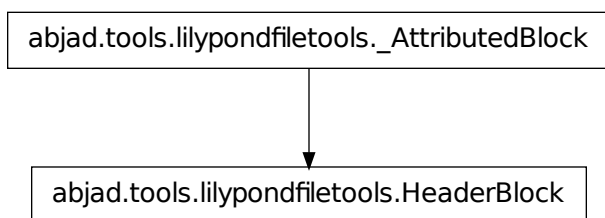
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DateTimeToken.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`lilypondfiletools.HeaderBlock`



class `abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock`. **HeaderBlock**
 New in version 2.0. Abjad model of LilyPond input file header block:

```

abjad> header_block = lilypondfiletools.HeaderBlock()

abjad> header_block
HeaderBlock()

abjad> header_block.composer = markuptools.Markup('Josquin')
abjad> header_block.title = markuptools.Markup('Missa sexti tonus')

abjad> f(header_block)
\header {
  composer = \markup { Josquin }
  title = \markup { Missa sexti tonus }
}
  
```

Return header block.

Read-only Properties

`HeaderBlock.format`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Read/write Properties

`HeaderBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Special Methods

`HeaderBlock.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HeaderBlock.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

`HeaderBlock.__repr__()`

Note: Inherited from `lilypondfiletools._AttributedBlock`

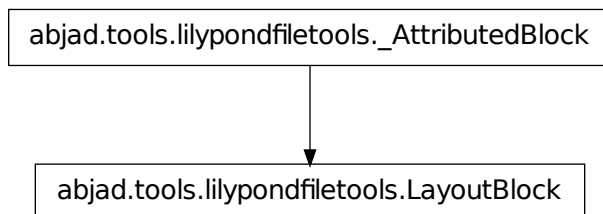
`HeaderBlock.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`HeaderBlock.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

`lilypondfiletools.LayoutBlock`



class `abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock` **LayoutBlock**
New in version 2.0. Abjad model of LilyPond input file layout block:

```
abjad> layout_block = lilypondfiletools.LayoutBlock()

abjad> layout_block
LayoutBlock()

abjad> layout_block.indent = 0
abjad> layout_block.ragged_right = True

abjad> f(layout_block)
\layout {
  indent = #0
  ragged-right = ##t
}
```

Return layout block.

Read-only Properties

`LayoutBlock.context_blocks`

Read-only list of context blocks:

```
abjad> layout_block = lilypondfiletools.LayoutBlock()

abjad> context_block = lilypondfiletools.ContextBlock('Score')
abjad> context_block.override.bar_number.transparent = True

abjad> context_block.override.time_signature.break_visibility = schemetools.Scheme('end-of-line-
abjad> layout_block.context_blocks.append(context_block)

abjad> f(layout_block)
\layout {
  \context {
    \Score
    \override BarNumber #'transparent = ##t
    \override TimeSignature #'break-visibility = #end-of-line-invisible
  }
}
```

Return list.

`LayoutBlock.contexts`

DEPRECATED. USE CONTEXT_BLOCKS INSTEAD.

`LayoutBlock.format`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Read/write Properties

`LayoutBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Special Methods

LayoutBlock.**__delattr__**()
 x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

LayoutBlock.**__hash__**() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

LayoutBlock.**__repr__**()

Note: Inherited from `lilypondfiletools._AttributedBlock`

LayoutBlock.**__setattr__**()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

LayoutBlock.**__str__**() <==> *str(x)*

Note: Inherited from `__builtin__.object`

`lilypondfiletools.LilyPondFile`

`abjad.tools.lilypondfiletools.LilyPondFile`

class `abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile` **LilyPondFile**

New in version 2.0. Abjad model of LilyPond input file:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> lilypond_file = lilypondfiletools.make_basic_lilypond_file(staff)
abjad> lilypond_file.file_initial_user_comments.append('File construct as an example.')
abjad> lilypond_file.file_initial_user_comments.append('Parts shown here for positioning.')
abjad> lilypond_file.file_initial_user_includes.append('external-settings-file-1.ly')
abjad> lilypond_file.file_initial_user_includes.append('external-settings-file-2.ly')
abjad> lilypond_file.default_paper_size = 'letter', 'portrait'
abjad> lilypond_file.global_staff_size = 16
abjad> lilypond_file.header_block.composer = markuptools.Markup('Josquin')
abjad> lilypond_file.header_block.title = markuptools.Markup('Missa sexti tonus')
abjad> lilypond_file.layout_block.indent = 0
```



```

abjad> lilypond_file.layout_block.left_margin = 15
abjad> lilypond_file.paper_block.oddFooterMarkup = markuptools.Markup('The odd-page footer')
abjad> lilypond_file.paper_block.evenFooterMarkup = markuptools.Markup('The even-page footer')

abjad> f(lilypond_file) # doctest: +SKIP
% Abjad revision 3719
% 2010-09-24 09:01

% File construct as an example.
% Parts shown here for positioning.

\version "2.13.32"
\include "english.ly"
\include "/Users/trevorbaca/Documents/abjad/trunk/abjad/cfg/abjad.scm"

\include "external-settings-file-1.ly"
\include "external-settings-file-2.ly"

#(set-default-paper-size "letter" 'portrait)
#(set-global-staff-size 16)

\header {
  composer = \markup { Josquin }
  title = \markup { Missa sexti tonus }
}

\layout {
  indent = #0
  left-margin = #15
}

\paper {
  evenFooterMarkup = \markup { The even-page footer }
  oddFooterMarkup = \markup { The odd-page footer }
}

\new Staff {
  c'8
  d'8
  e'8
  f'8
}

```

Read-only Properties**LilyPondFile.format**

Format-time contribution of LilyPond file.

Read/write Properties**LilyPondFile.default_paper_size**

LilyPond default paper size.

LilyPondFile.file_initial_system_comments

Read-only list of file-initial system comments.

LilyPondFile.file_initial_system_includes

List of file-initial system include commands.

`LilyPondFile.file_initial_user_comments`

Read-only list of file-initial user comments.

`LilyPondFile.file_initial_user_includes`

List of file-initial user include commands.

`LilyPondFile.global_staff_size`

LilyPond global staff size.

Methods

`LilyPondFile.append()`

`L.append(object)` – append object to end

Note: Inherited from `__builtin__.list`

`LilyPondFile.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`LilyPondFile.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Note: Inherited from `__builtin__.list`

`LilyPondFile.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`LilyPondFile.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`LilyPondFile.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`LilyPondFile.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`LilyPondFile.reverse()`

`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

LilyPondFile.**sort**()

L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

Note: Inherited from `__builtin__.list`

Special Methods

LilyPondFile.**__add__**()

x.__add__(y) <==> x+y

Note: Inherited from `__builtin__.list`

LilyPondFile.**__contains__**()

x.__contains__(y) <==> y in x

Note: Inherited from `__builtin__.list`

LilyPondFile.**__delattr__**()

x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

LilyPondFile.**__delitem__**()

x.__delitem__(y) <==> del x[y]

Note: Inherited from `__builtin__.list`

LilyPondFile.**__delslice__**()

x.__delslice__(i, j) <==> del x[i:j]

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

LilyPondFile.**__eq__**()

x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.list`

LilyPondFile.**__ge__**()

x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.list`

LilyPondFile.**__getitem__**()

x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__getslice__()  
x.__getslice__(i, j) <==> x[i:j]
```

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__gt__()  
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__iadd__()  
x.__iadd__(y) <==> x+=y
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__imul__()  
x.__imul__(y) <==> x*=y
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__iter__() <==> iter(x)
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__le__()  
x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__len__() <==> len(x)
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__lt__()  
x.__lt__(y) <==> x<y
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__mul__()  
x.__mul__(n) <==> x*n
```

Note: Inherited from `__builtin__.list`

```
LilyPondFile.__ne__()  
x.__ne__(y) <==> x!=y
```

Note: Inherited from `__builtin__.list`

`LilyPondFile.__repr__()`

`LilyPondFile.__reversed__()`

`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`LilyPondFile.__rmul__()`

`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`LilyPondFile.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`LilyPondFile.__setitem__()`

`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

`LilyPondFile.__setslice__()`

`x.__setslice__(i, j, y) <==> x[i:j]=y`

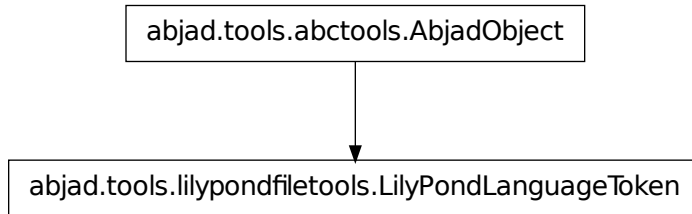
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`LilyPondFile.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

lilypondfiletools.LilyPondLanguageToken



class `abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken`. **LilyPondLanguageToken**
 New in version 2.0. LilyPond language token:

```
abjad> lilypondfiletools.LilyPondLanguageToken()
LilyPondLanguageToken(include "english.ly")
```

Return LilyPond language token.

Read-only Properties

`LilyPondLanguageToken.format`

Format contribution of LilyPond language token:

```
abjad> lilypondfiletools.LilyPondLanguageToken().format
'\\include "english.ly"'
```

Return string.

Special Methods

`LilyPondLanguageToken.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LilyPondLanguageToken.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LilyPondLanguageToken.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondLanguageToken.**__gt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

LilyPondLanguageToken.**__hash__**() $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

LilyPondLanguageToken.**__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondLanguageToken.**__lt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondLanguageToken.**__ne__**(*arg*)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

LilyPondLanguageToken.**__repr__**()

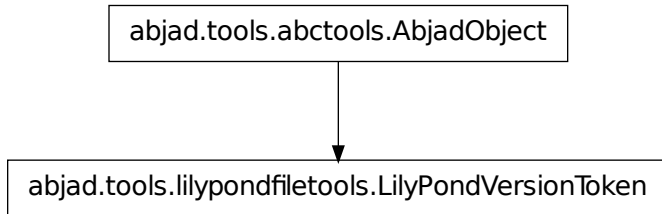
LilyPondLanguageToken.**__setattr__**()
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

LilyPondLanguageToken.**__str__**() $\leq \Rightarrow$ *str(x)*

Note: Inherited from `__builtin__.object`

lilypondfiletools.LilyPondVersionToken



class `abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken`. **LilyPondVersionToken**
 New in version 2.0. LilyPond version token:

```
abjad> lilypondfiletools.LilyPondVersionToken()
LilyPondVersionToken(\version "...")
```

Return LilyPond version token.

Read-only Properties

`LilyPondVersionToken.format`

Format contribution of LilyPond version token:

```
abjad> lilypondfiletools.LilyPondVersionToken().format
'\version "..."
```

Return string.

Special Methods

`LilyPondVersionToken.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LilyPondVersionToken.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LilyPondVersionToken.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondVersionToken.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

LilyPondVersionToken.__hash__() \Leftrightarrow hash(x)

Note: Inherited from `__builtin__.object`

LilyPondVersionToken.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondVersionToken.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondVersionToken.__ne__(arg)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

LilyPondVersionToken.__repr__()

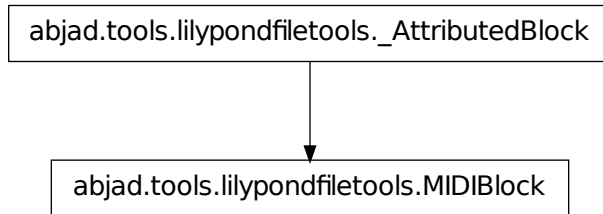
LilyPondVersionToken.__setattr__()
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

LilyPondVersionToken.__str__() \Leftrightarrow str(x)

Note: Inherited from `__builtin__.object`

`lilypondfiletools.MIDIBlock`



class `abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock`

New in version 2.0. Abjad model of LilyPond input file MIDI block:

```

abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> score = Score([staff])
abjad> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)

abjad> lilypond_file.score_block.append(lilypondfiletools.MIDIBlock())

abjad> layout_block = lilypondfiletools.LayoutBlock()
abjad> layout_block.is_formatted_when_empty = True
abjad> lilypond_file.score_block.append(layout_block)

abjad> f(lilypond_file.score_block)
\score {
  \new Score <<
    \new Staff {
      c'4
      d'4
      e'4
      f'4
    }
  >>
  \midi {}
  \layout {}
}
  
```

MIDI blocks are formatted even when they are empty.

The example here appends MIDI and layout blocks to a score block. Doing this allows LilyPond to create both MIDI and PDF output from a single input file.

Read the LilyPond docs on LilyPond file structure for the details as to why this is the case.

Read-only Properties

`MIDIBlock.format`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Read/write Properties`MIDIBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Special Methods`MIDIBlock.__delattr__()``x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MIDIBlock.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`MIDIBlock.__repr__()`

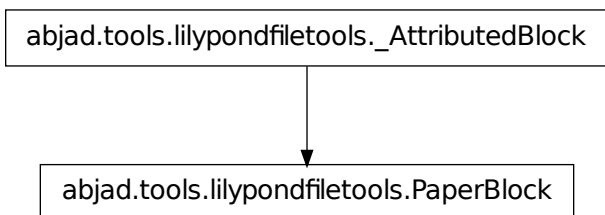
Note: Inherited from `lilypondfiletools._AttributedBlock`

`MIDIBlock.__setattr__()``x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MIDIBlock.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`lilypondfiletools.PaperBlock`

class `abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock`

New in version 2.0. Abjad model of LilyPond input file paper block:

```
abjad> paper_block = lilypondfiletools.PaperBlock()

abjad> paper_block
PaperBlock()

abjad> paper_block.print_page_number = True
abjad> paper_block.print_first_page_number = False

abjad> f(paper_block)
\paper {
  print-first-page-number = ##f
  print-page-number = ##t
}
```

Return paper block.

Read-only Properties

`PaperBlock.format`

Note: Inherited from `lilypondfiletools._AttributedBlock`

Read/write Properties

`PaperBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._AttributedBlock`

`PaperBlock.minimal_page_breaking`

Special Methods

`PaperBlock.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`PaperBlock.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

`PaperBlock.__repr__()`

Note: Inherited from `lilypondfiletools._AttributedBlock`

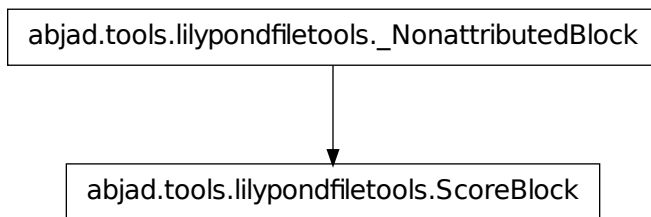
`PaperBlock.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`PaperBlock.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`lilypondfiletools.ScoreBlock`



class `abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock`, **ScoreBlock**

New in version 2.0. Abjad model of LilyPond input file score block:

```
abjad> score_block = lilypondfiletools.ScoreBlock()
```

```
abjad> score_block
ScoreBlock()
```

```
abjad> score_block.append(Staff([]))
abjad> f(score_block)
\score {
  \new Staff {
  }
}
```

`ScoreBlocks` does not format when empty, as this generates a parser error in LilyPond:

```
abjad> score_block = lilypondfiletools.ScoreBlock()
abjad> score_block.format == ''
True
```

Return score block.

Read-only Properties

`ScoreBlock.format`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Read/write Properties

`ScoreBlock.is_formatted_when_empty`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

Methods

`ScoreBlock.append()`

`L.append(object)` – append object to end

Note: Inherited from `__builtin__.list`

`ScoreBlock.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`ScoreBlock.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Note: Inherited from `__builtin__.list`

`ScoreBlock.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ScoreBlock.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`ScoreBlock.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`ScoreBlock.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ScoreBlock.reverse()`

`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`ScoreBlock.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`ScoreBlock.__add__()`

`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__contains__()`

`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ScoreBlock.__delitem__()`

`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__delslice__()`

`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ScoreBlock.__eq__()`

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__ge__()`

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ScoreBlock.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__mul__()`
`x.__mul__(n) <==> x*n`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__repr__()`

Note: Inherited from `lilypondfiletools._NonattributedBlock`

`ScoreBlock.__reversed__()`

`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`ScoreBlock.__rmul__()`

`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ScoreBlock.__setitem__()`

`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

`ScoreBlock.__setslice__()`

`x.__setslice__(i, j, y) <==> x[i:j]=y`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ScoreBlock.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

lilypondfiletools functions

lilypondfiletools.make_basic_lilypond_file

`abjad.tools.lilypondfiletools.make_basic_lilypond_file.make_basic_lilypond_file(music=None)`

New in version 2.0. Make basic LilyPond file with *music*:

```

abjad> score = Score([Staff("c'8 d'8 e'8 f'8")])
abjad> lilypond_file = lilypondfiletools.make_basic_lilypond_file(score)
abjad> lilypond_file.header_block.composer = markuptools.Markup('Josquin')
abjad> lilypond_file.layout_block.indent = 0
abjad> lilypond_file.paper_block.top_margin = 15
abjad> lilypond_file.paper_block.left_margin = 15

abjad> f(lilypond_file) # doctest: +SKIP
\header {
  composer = \markup { Josquin }
}

\layout {
  indent = #0
}

\paper {
  left-margin = #15
  top-margin = #15
}

\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

```

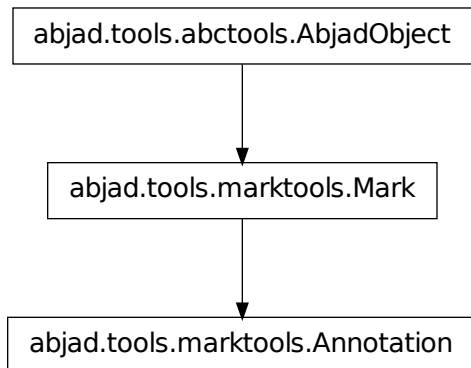
Equip LilyPond file with header, layout and paper blocks.

Return LilyPond file.

marktools

marktools concrete classes

marktools.Annotation



class `abjad.tools.marktools.Annotation.Annotation(*args)`

New in version 2.0. User-defined annotation:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

```
abjad> marktools.Annotation('special pitch', pitchtools.NamedChromaticPitch('ds'))(staff[0])
Annotation('special pitch', NamedChromaticPitch('ds'))(c'8)
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

Annotations contribute no formatting.

Annotations implement `__slots__`.

Read-only Properties

Annotation.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties**Annotation.name**

Get name of annotation:

```
abjad> annotation = marktools.Annotation('special_pitch', pitchtools.NamedChromaticPitch('ds'))
abjad> annotation.name
'special_pitch'
```

Set name of annotation:

```
abjad> annotation.name = 'revised special pitch'
abjad> annotation.name
'revised special pitch'
```

Set string.

Annotation.value

Get value of annotation:

```
abjad> annotation = marktools.Annotation('special_pitch', pitchtools.NamedChromaticPitch('ds'))
abjad> annotation.value
NamedChromaticPitch('ds')
```

Set value of annotation:

```
abjad> annotation.value = pitchtools.NamedChromaticPitch('e')
abjad> annotation.value
NamedChromaticPitch('e')
```

Set arbitrary object.

Methods**Annotation.attach(start_component)**

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()
```

```
abjad> mark.attach(note)
Mark()(c'4)
```

```
abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

`Annotation.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

```
abjad> mark.detach()
Mark()
```

```
abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

`Annotation.__call__(*args)`

Note: Inherited from `marktools.Mark`

`Annotation.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`Annotation.__eq__(arg)`

`Annotation.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Annotation.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Annotation.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Annotation.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Annotation.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Annotation.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`Annotation.__repr__()`

Note: Inherited from `marktools.Mark`

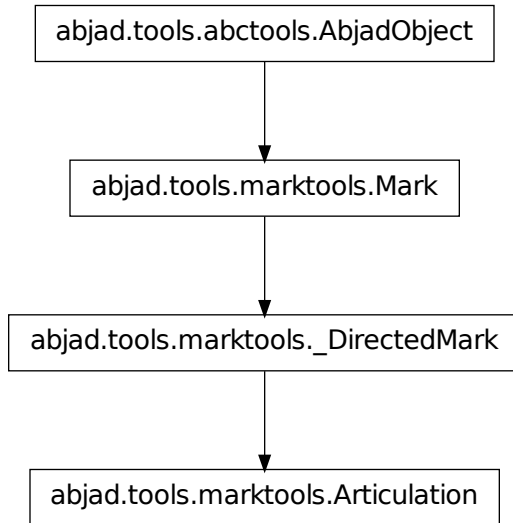
`Annotation.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Annotation.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

marktools.Articulation

class `abjad.tools.marktools.Articulation.Articulation(*args)`

Abjad model of musical articulation:

```
abjad> note = Note("c'4")
```

```
abjad> marktools.Articulation('staccato')(note)
Articulation('staccato')(c'4)
```

```
abjad> f(note)
c'4 -\staccato
```

Articulations implement `__slots__`.

Read-only Properties

`Articulation.format`

Read-only LilyPond format string of articulation:

```
abjad> articulation = marktools.Articulation('marcato', 'up')
abjad> articulation.format
'^\marcato'
```

Return string.

`Articulation.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties

Articulation.**direction**

Note: Inherited from `marktools._DirectedMark`

Articulation.**name**

Get name of articulation:

```
abjad> articulation = marktools.Articulation('staccato', 'up')
abjad> articulation.name
'staccato'
```

Set name of articulation:

```
abjad> articulation.name = 'marcato'
abjad> articulation.name
'marcato'
```

Set string.

Methods

Articulation.**attach**(*start_component*)

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark() (c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

Articulation.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```



```
abjad> mark.detach()  
Mark()
```

```
abjad> mark.start_component is None  
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

Articulation.__call__(*args)

Note: Inherited from `marktools.Mark`

Articulation.__delattr__(*args)

Note: Inherited from `marktools.Mark`

Articulation.__eq__(expr)

Articulation.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Articulation.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Articulation.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

Articulation.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Articulation.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Articulation.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`Articulation.__repr__()`

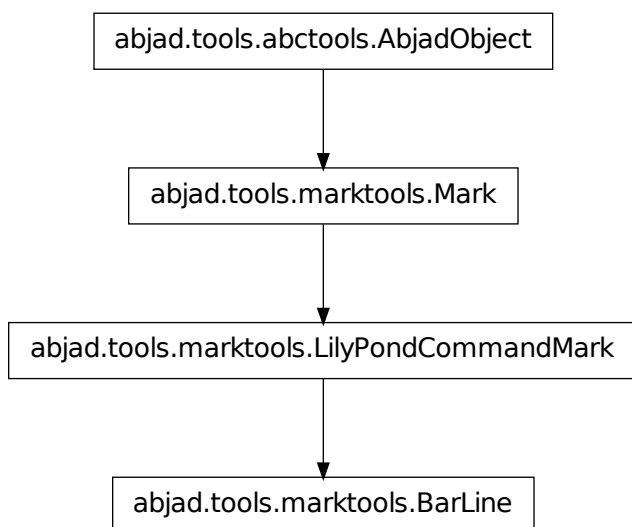
Note: Inherited from `marktools.Mark`

`Articulation.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Articulation.__str__()`

marktools.BarLine



class `abjad.tools.marktools.BarLine.BarLine`.**BarLine** (*bar_line_string='|'*, *for-*
mat_slot='after')

New in version 2.4. Abjad model of bar line:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")

abjad> bar_line = marktools.BarLine('|.')(staff[-1])

abjad> bar_line
BarLine('|.')(f'4)

abjad> f(staff)
\new Staff {
    c'4
    d'4
    e'4
    f'4
    \bar "|."
}
```

Return bar line.

Read-only Properties

BarLine.format

Read-only LilyPond input format of LilyPond command mark:

```
abjad> note = Note("c'4")
abjad> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(note)
abjad> lilypond_command.format
'\slurDotted'
```

Return string.

Note: Inherited from `marktools.LilyPondCommandMark`

BarLine.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties

BarLine.bar_line_string

Get bar line string of bar line:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")
abjad> bar_line = marktools.BarLine()(staff[-1])
abjad> bar_line.bar_line_string
'|'
```

Set bar line string of bar line:

```
abjad> bar_line.bar_line_string = '|.'
```

```
abjad> bar_line.bar_line_string
```

```
'|.'
```

```
abjad> f(staff)
```

```
\new Staff {
```

```
  c'4
```

```
  d'4
```

```
  e'4
```

```
  f'4
```

```
  \bar "|."
```

```
}
```

Set string.

BarLine.command_name

Get command name of LilyPond command mark:

```
abjad> lilypond_command = marktools.LilyPondCommandMark('slurDotted')
```

```
abjad> lilypond_command.command_name
```

```
'slurDotted'
```

Set command name of LilyPond command mark:

```
abjad> lilypond_command.command_name = 'slurDashed'
```

```
abjad> lilypond_command.command_name
```

```
'slurDashed'
```

Set string.

Note: Inherited from `marktools.LilyPondCommandMark`

BarLine.format_slot

New in version 2.3. Get format slot of LilyPond command mark:

```
abjad> note = Note("c'4")
```

```
abjad> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
```

```
abjad> lilypond_command.format_slot
```

```
'after'
```

Set format slot of LilyPond command mark:

```
abjad> note = Note("c'4")
```

```
abjad> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
```

```
abjad> lilypond_command.format_slot = 'before'
```

```
abjad> lilypond_command.format_slot
```

```
'before'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Note: Inherited from `marktools.LilyPondCommandMark`

Methods

BarLine.attach (*start_component*)

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()
```

```
abjad> mark.attach(note)
Mark() (c'4)
```

```
abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

`BarLine.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

```
abjad> mark.detach()
Mark()
```

```
abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

`BarLine.__call__(*args)`

Note: Inherited from `marktools.Mark`

`BarLine.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`BarLine.__eq__(arg)`

Note: Inherited from `marktools.LilyPondCommandMark`

`BarLine.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BarLine.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`BarLine.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`BarLine.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BarLine.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BarLine.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BarLine.__repr__()`

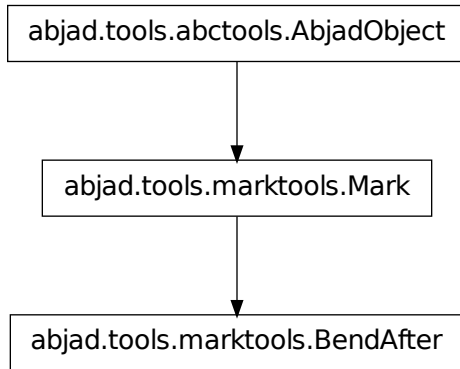
Note: Inherited from `marktools.Mark`

`BarLine.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BarLine.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

marktools.BendAfter

class `abjad.tools.marktools.BendAfter`, `BendAfter`, **BendAfter** (*args)

Abjad model of a fall or doit:

```
abjad> note = Note("c'4")

abjad> marktools.BendAfter(-4)(note)
BendAfter(-4.0)(c'4)

abjad> f(note)
c'4 - \bendAfter #'-4.0
```

BendAfter implements `__slots__`.

Read-only Properties

`BendAfter.format`

Read-only LilyPond format string:

```
abjad> bend = marktools.BendAfter(-4)
abjad> bend.format
"- \\bendAfter #'-4.0"
```

Return string.

`BendAfter.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties**BendAfter.bend_amount**

Get bend amount:

```
abjad> bend = marktools.BendAfter(8)
abjad> bend.bend_amount
8.0
```

Set bend amount:

```
abjad> bend.bend_amount = -4
abjad> bend.bend_amount
-4.0
```

Set float.

Methods**BendAfter.attach**(*start_component*)Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark() (c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

BendAfter.detach()

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

abjad> mark.detach()
Mark()

abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods**BendAfter.__call__**(*args)

Note: Inherited from `marktools.Mark`

`BendAfter.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`BendAfter.__eq__(expr)`

`BendAfter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BendAfter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BendAfter.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`BendAfter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BendAfter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BendAfter.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`BendAfter.__repr__()`

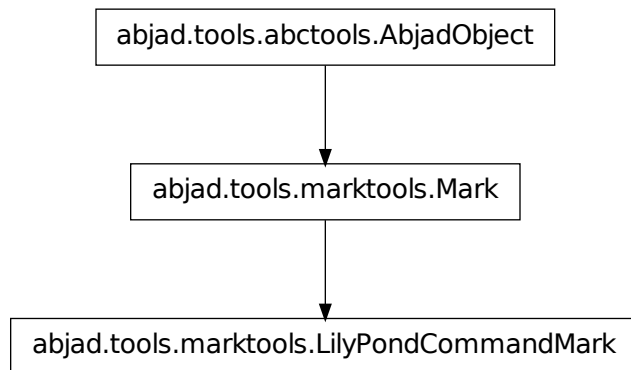
Note: Inherited from `marktools.Mark`

```
BendAfter.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
BendAfter.__str__()
```

marktools.LilyPondCommandMark



class `abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark(*args)`
New in version 2.0. LilyPond command mark:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")  
abjad> slur = spannertools.SlurSpanner(staff.leaves)  
  
abjad> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(staff[0])  
  
abjad> f(staff)  
\new Staff {  
  \slurDotted  
  c'8 (  
  d'8  
  e'8  
  f'8 )  
}
```

Initialize LilyPond command marks from command name; or from command name with format slot; or from another LilyPond command mark; or from another LilyPond command mark with format slot.

LilyPond command marks implement `__slots__`.

Read-only Properties

`LilyPondCommandMark.format`

Read-only LilyPond input format of LilyPond command mark:

```
abjad> note = Note("c'4")
abjad> lilypond_command = marktools.LilyPondCommandMark('slurDotted')(note)
abjad> lilypond_command.format
'\slurDotted'
```

Return string.

`LilyPondCommandMark.start_component`

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties

`LilyPondCommandMark.command_name`

Get command name of LilyPond command mark:

```
abjad> lilypond_command = marktools.LilyPondCommandMark('slurDotted')
abjad> lilypond_command.command_name
'slurDotted'
```

Set command name of LilyPond command mark:

```
abjad> lilypond_command.command_name = 'slurDashed'
abjad> lilypond_command.command_name
'slurDashed'
```

Set string.

`LilyPondCommandMark.format_slot`

New in version 2.3. Get format slot of LilyPond command mark:

```
abjad> note = Note("c'4")
abjad> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
abjad> lilypond_command.format_slot
'after'
```

Set format slot of LilyPond command mark:

```
abjad> note = Note("c'4")
abjad> lilypond_command = marktools.LilyPondCommandMark('break', 'after')
abjad> lilypond_command.format_slot = 'before'
abjad> lilypond_command.format_slot
'before'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Methods

`LilyPondCommandMark.attach(start_component)`

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark() (c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

```
LilyPondCommandMark.detach()
Detach mark:

abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

abjad> mark.detach()
Mark()

abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

`LilyPondCommandMark.__call__(*args)`

Note: Inherited from `marktools.Mark`

`LilyPondCommandMark.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`LilyPondCommandMark.__eq__(arg)`

`LilyPondCommandMark.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondCommandMark.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

LilyPondCommandMark.__hash__() \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

LilyPondCommandMark.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondCommandMark.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

LilyPondCommandMark.__ne__(arg)

Note: Inherited from `marktools.Mark`

LilyPondCommandMark.__repr__()

Note: Inherited from `marktools.Mark`

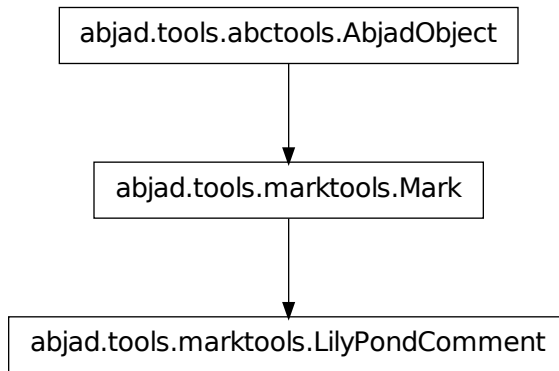
LilyPondCommandMark.__setattr__()
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

LilyPondCommandMark.__str__() \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

marktools.LilyPondComment



class `abjad.tools.marktools.LilyPondComment.LilyPondComment(*args)`
 New in version 2.0.Changed in version 2.3: Changed Comment to LilyPondComment. User-defined comment:

```

abjad> note = Note("c'4")

abjad> marktools.LilyPondComment('this is a comment')(note)
LilyPondComment('this is a comment')(c'4)

abjad> f(note)
% this is a comment
c'4
  
```

Initialize LilyPond comment from contents string; or contents string and format slot; or from other LilyPond comment; or from other LilyPond comment and format slot.

LilyPond comments implement `__slots__`.

Read-only Properties

`LilyPondComment.format`

Read-only LilyPond input format of comment:

```

abjad> comment = marktools.LilyPondComment('this is a comment.')
abjad> comment.format
'% this is a comment.'
  
```

Return string.

`LilyPondComment.start_component`

Read-only reference to mark start component:

```

abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
  
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties

`LilyPondComment.contents_string`

Get contents string of comment:

```
abjad> comment = marktools.LilyPondComment('comment contents string')
abjad> comment.contents_string
'comment contents string'
```

Set contents string of comment:

```
abjad> comment.contents_string = 'new comment contents string'
abjad> comment.contents_string
'new comment contents string'
```

Set string.

`LilyPondComment.format_slot`

New in version 2.3. Get format slot of LilyPond comment:

```
abjad> note = Note("c'4")
abjad> lilypond_comment = marktools.LilyPondComment('comment')
abjad> lilypond_comment.format_slot
'before'
```

Set format slot of LilyPond comment:

```
abjad> note = Note("c'4")
abjad> lilypond_comment = marktools.LilyPondComment('comment')
abjad> lilypond_comment.format_slot = 'after'
abjad> lilypond_comment.format_slot
'after'
```

Set to 'before', 'after', 'opening', 'closing', 'right' or none.

Methods

`LilyPondComment.attach(start_component)`

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()
```

```
abjad> mark.attach(note)
Mark() (c'4)
```

```
abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

`LilyPondComment.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

abjad> mark.detach()
Mark()

abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

`LilyPondComment.__call__(*args)`

Note: Inherited from `marktools.Mark`

`LilyPondComment.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`LilyPondComment.__eq__(arg)`

`LilyPondComment.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondComment.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`LilyPondComment.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`LilyPondComment.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondComment.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondComment.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`LilyPondComment.__repr__()`

Note: Inherited from `marktools.Mark`

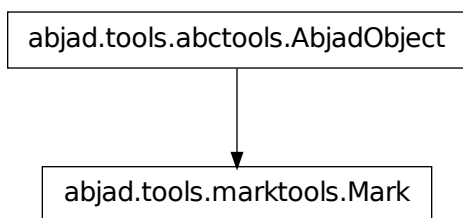
`LilyPondComment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`LilyPondComment.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

marktools.Mark



class `abjad.tools.marktools.Mark.Mark`.**Mark**(*args)
 New in version 2.0. Abstract class from which concrete marks inherit:

```
abjad> note = Note("c'4")

abjad> marktools.Mark()(note)
Mark()(c'4)
```

Marks override `__call__` to attach to a note, rest or chord.

Marks implement `__slots__`.

Read-only Properties

Mark.**start_component**

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Methods

Mark.**attach**(*start_component*)

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark()(c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Mark.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

abjad> mark.detach()
Mark()

abjad> mark.start_component is None
True
```

Return mark.

Special Methods

Mark.**__call__**(*args)

Mark.**__delattr__**(*args)

Mark.**__eq__**(arg)

Mark.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mark.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Mark.**__hash__**() $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

Mark.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mark.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mark.**__ne__**(*arg*)

Mark.**__repr__**()

Mark.**__setattr__**()

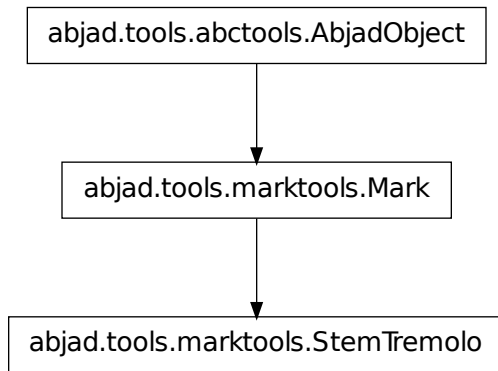
x.**__setattr__**('name', value) $\leq \Rightarrow$ *x*.name = value

Note: Inherited from `__builtin__.object`

Mark.**__str__**() $\leq \Rightarrow$ *str(x)*

Note: Inherited from `__builtin__.object`

marktools.StemTremolo



class `abjad.tools.marktools.StemTremolo.StemTremolo(*args)`

New in version 2.0. Abjad model of stem tremolo:

```

abjad> note = Note("c'4")

abjad> marktools.StemTremolo(16)(note)
StemTremolo(16)(c'4)

abjad> f(note)
c'4 :16
  
```

Stem tremolos implement `__slots__`.

Read-only Properties

`StemTremolo.format`

Read-only LilyPond format string:

```

abjad> stem_tremolo = marktools.StemTremolo(16)
abjad> stem_tremolo.format
':16'
  
```

Return string.

`StemTremolo.start_component`

Read-only reference to mark start component:

```

abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
  
```

Return component or none.

Note: Inherited from `marktools.Mark`

Read/write Properties

`StemTremolo.tremolo_flags`

Get tremolo flags:

```
abjad> stem_tremolo = marktools.StemTremolo(16)
abjad> stem_tremolo.tremolo_flags
16
```

Set tremolo flags:

```
abjad> stem_tremolo.tremolo_flags = 32
abjad> stem_tremolo.tremolo_flags
32
```

Set integer.

Methods

`StemTremolo.attach(start_component)`

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark() (c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

`StemTremolo.detach()`

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")

abjad> mark.detach()
Mark()

abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

`StemTremolo.__call__(*args)`

Note: Inherited from `marktools.Mark`

`StemTremolo.__delattr__(*args)`

Note: Inherited from `marktools.Mark`

`StemTremolo.__eq__(expr)`

`StemTremolo.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StemTremolo.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`StemTremolo.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`StemTremolo.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StemTremolo.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StemTremolo.__ne__(arg)`

Note: Inherited from `marktools.Mark`

`StemTremolo.__repr__()`

Note: Inherited from `marktools.Mark`

```
StemTremolo.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
StemTremolo.__str__()
```

marktools functions

marktools.attach_annotations_to_components_in_expr

```
abjad.tools.marktools.attach_annotations_to_components_in_expr.attach_annotations_to_compon
```

New in version 2.3. Attach *annotations* to components in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> annotation = marktools.Annotation('foo', 'bar')
abjad> marktools.attach_annotations_to_components_in_expr(staff.leaves, [annotation])

abjad> for x in staff:
...     print x, marktools.get_annotations_attached_to_component(x)
...
c'8 (Annotation('foo', 'bar')(c'8),)
d'8 (Annotation('foo', 'bar')(d'8),)
e'8 (Annotation('foo', 'bar')(e'8),)
f'8 (Annotation('foo', 'bar')(f'8),)
```

Return none.

marktools.attach_articulations_to_notes_and_chords_in_expr

```
abjad.tools.marktools.attach_articulations_to_notes_and_chords_in_expr.attach_articulations
```

New in version 2.0. Attach *articulations* to notes and chords in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.attach_articulations_to_notes_and_chords_in_expr(staff, list('^.'))

abjad> f(staff)
\new Staff {
    c'8 -\marcato -\staccato
    d'8 -\marcato -\staccato
    e'8 -\marcato -\staccato
    f'8 -\marcato -\staccato
}
```

Return none.

marktools.attach_lilypond_command_marks_to_components_in_expr

abjad.tools.marktools.attach_lilypond_command_marks_to_components_in_expr.**attach_lilypond_command_marks_to_components_in_expr**

New in version 2.3. Attach *lilypond_command_marks* to components in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> lilypond_command_mark = marktools.LilyPondCommandMark('stemUp')
abjad> marktools.attach_lilypond_command_marks_to_components_in_expr(staff.leaves, [lilypond_command_mark])

abjad> f(staff)
\new Staff {
  \stemUp
  c'8
  \stemUp
  d'8
  \stemUp
  e'8
  \stemUp
  f'8
}
```

Return none.

marktools.attach_lilypond_comments_to_components_in_expr

abjad.tools.marktools.attach_lilypond_comments_to_components_in_expr.**attach_lilypond_comments_to_components_in_expr**

New in version 2.3. Attach *lilypond_comments* to components in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> lilypond_comment = marktools.LilyPondComment('foo', 'right')
abjad> marktools.attach_lilypond_comments_to_components_in_expr(staff.leaves, [lilypond_comment])

abjad> f(staff)
\new Staff {
  c'8 % foo
  d'8 % foo
  e'8 % foo
  f'8 % foo
}
```

Return none.

marktools.attach_stem_tremolos_to_notes_and_chords_in_expr

abjad.tools.marktools.attach_stem_tremolos_to_notes_and_chords_in_expr.**attach_stem_tremolos_to_notes_and_chords_in_expr**

New in version 2.3. Attach *stem_tremolos* to notes and chords in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> stem_tremolo = marktools.StemTremolo(16)
abjad> marktools.attach_stem_tremolos_to_notes_and_chords_in_expr(staff, [stem_tremolo])
```



```

abjad> f(staff)
\new Staff {
    c'8 :16
    d'8 :16
    e'8 :16
    f'8 :16
}

```

Return none.

marktools.detach_annotations_attached_to_component

abjad.tools.marktools.detach_annotations_attached_to_component.**detach_annotations_attached_to_component**

New in version 2.0. Detach annotations attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.Annotation('annotation 1')(staff[0])
Annotation('annotation 1')(c'8)
abjad> marktools.Annotation('annotation 2')(staff[0])
Annotation('annotation 2')(c'8)

abjad> f(staff)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}

abjad> marktools.get_annotations_attached_to_component(staff[0])
(Annotation('annotation 1')(c'8), Annotation('annotation 2')(c'8))

abjad> marktools.detach_annotations_attached_to_component(staff[0])
(Annotation('annotation 1'), Annotation('annotation 2'))

abjad> marktools.get_annotations_attached_to_component(staff[0])
()

```

Return tuple or zero or more annotations detached.

marktools.detach_articulations_attached_to_component

abjad.tools.marktools.detach_articulations_attached_to_component.**detach_articulations_attached_to_component**

New in version 2.0. Detach articulations attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.Articulation('^')(staff[0])
Articulation('^')(c'8)
abjad> marktools.Articulation('.') (staff[0])
Articulation('.') (c'8)

abjad> f(staff)
\new Staff {
    c'8 -\marcato -\staccato (

```

```

        d'8
        e'8
        f'8 )
    }

abjad> marktools.get_articulations_attached_to_component(staff[0])
(Articulation('^') (c'8), Articulation('.') (c'8))

abjad> marktools.detach_articulations_attached_to_component(staff[0])
(Articulation('^'), Articulation('.'))

abjad> marktools.get_articulations_attached_to_component(staff[0])
()
```

Return tuple or zero or more articulations detached.

marktools.detach_lilypond_command_marks_attached_to_component

`abjad.tools.marktools.detach_lilypond_command_marks_attached_to_component.detach_lilypond_c`

New in version 2.0. Detach LilyPond command marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted') (c'8)
abjad> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp') (c'8)

abjad> f(staff)
\new Staff {
  \slurDotted
  \slurUp
  c'8 (
  d'8
  e'8
  f'8 )
}

abjad> marktools.detach_lilypond_command_marks_attached_to_component(staff[0])
(LilyPondCommandMark('slurDotted'), LilyPondCommandMark('slurUp'))

abjad> f(staff)
\new Staff {
  c'8 (
  d'8
  e'8
  f'8 )
}
```

Return tuple of zero or more marks detached.

marktools.detach_lilypond_comments_attached_to_component

abjad.tools.marktools.detach_lilypond_comments_attached_to_component.**detach_lilypond_comme**

New in version 2.0. Detach LilyPond comments attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
abjad> marktools.LilyPondComment('comment 2')(staff[0])
LilyPondComment('comment 2')(c'8)
```

```
abjad> f(staff)
\new Staff {
    % comment 1
    % comment 2
    c'8 (
    d'8
    e'8
    f'8 )
}
```

```
abjad> marktools.detach_lilypond_comments_attached_to_component(staff[0])
(LilyPondComment('comment 1'), LilyPondComment('comment 2'))
```

```
abjad> f(staff)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}
```

```
abjad> marktools.get_lilypond_comments_attached_to_component(staff[0])
()
```

Return tuple or zero or more LilyPond comments.

marktools.detach_marks_attached_to_component

abjad.tools.marktools.detach_marks_attached_to_component.**detach_marks_attached_to_component**

New in version 2.0. Detach marks attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.Articulation('^')(staff[0])
Articulation('^')(c'8)
abjad> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
abjad> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp')(c'8)
```

```
abjad> f(staff)
\new Staff {
    % comment 1
    \slurUp
    c'8 -\marcato (
```

```

        d'8
        e'8
        f'8 )
    }

abjad> marktools.get_marks_attached_to_component(staff[0])
(Articulation('^')(c'8), LilyPondComment('comment 1')(c'8), LilyPondCommandMark('slurUp')(c'8))

abjad> marktools.detach_marks_attached_to_component(staff[0])
(Articulation('^'), LilyPondComment('comment 1'), LilyPondCommandMark('slurUp'))

abjad> marktools.get_marks_attached_to_component(staff[0])
()
```

Return tuple or zero or more marks detached.

marktools.detach_noncontext_marks_attached_to_component

`abjad.tools.marktools.detach_noncontext_marks_attached_to_component.detach_noncontext_marks`

New in version 2.3. Detach noncontext marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((2, 4))(staff[0])
TimeSignatureMark((2, 4))(c'8)
abjad> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

abjad> f(staff)
\new Staff {
    \time 2/4
    c'8 -\staccato
    d'8
    e'8
    f'8
}

abjad> marktools.detach_noncontext_marks_attached_to_component(staff[0])
(Articulation('staccato'),)

abjad> f(staff)
\new Staff {
    \time 2/4
    c'8
    d'8
    e'8
    f'8
}
```

Return tuple of noncontext marks.

marktools.detach_stem_tremolos_attached_to_component

`abjad.tools.marktools.detach_stem_tremolos_attached_to_component.detach_stem_tremolos_atta`

New in version 2.0. Detach stem tremolos attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

abjad> f(staff)
\new Staff {
    c'8 :16
    d'8
    e'8
    f'8
}

abjad> marktools.get_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16)(c'8),)

abjad> marktools.detach_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16),)

abjad> marktools.get_stem_tremolos_attached_to_component(staff[0])
()
```

Return tuple or zero or more stem tremolos detached.

marktools.get_annotation_attached_to_component

abjad.tools.marktools.get_annotation_attached_to_component.get_annotation_attached_to_component

New in version 2.0. Get exactly one annotation attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Annotation('special information')(staff[0])
Annotation('special information')(c'8)

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> marktools.get_annotation_attached_to_component(staff[0])
Annotation('special information')(c'8)
```

Return one annotation.

Raise missing mark error when no annotation is attached.

Raise extra mark error when more than one annotation is attached.

marktools.get_annotations_attached_to_component

abjad.tools.marktools.get_annotations_attached_to_component.get_annotations_attached_to_component

New in version 2.0. Get annotations attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Annotation('annotation 1')(staff[0])
```

```

Annotation('annotation 1')(c'8)
abjad> marktools.Annotation('annotation 2')(staff[0])
Annotation('annotation 2')(c'8)

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> marktools.get_annotations_attached_to_component(staff[0])
(Annotation('annotation 1')(c'8), Annotation('annotation 2')(c'8))

```

Return tuple of zero or more annotations.

marktools.get_articulation_attached_to_component

`abjad.tools.marktools.get_articulation_attached_to_component.get_articulation_attached_to_component`
 New in version 2.0. Get exactly one articulation attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

abjad> f(staff)
\new Staff {
    c'8 -\staccato
    d'8
    e'8
    f'8
}

abjad> marktools.get_articulation_attached_to_component(staff[0])
Articulation('staccato')(c'8)

```

Return one articulation.

Raise missing mark error when no articulation is attached.

Raise extra mark error when more than one articulation is attached.

marktools.get_articulations_attached_to_component

`abjad.tools.marktools.get_articulations_attached_to_component.get_articulations_attached_to_component`
 New in version 2.0. Get articulations attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)
abjad> marktools.Articulation('marcato')(staff[0])
Articulation('marcato')(c'8)

abjad> f(staff)
\new Staff {
    c'8 -\marcato -\staccato
}

```

```

    d'8
    e'8
    f'8
}

```

```

abjad> marktools.get_articulations_attached_to_component(staff[0])
(Articulation('staccato')(c'8), Articulation('marcato')(c'8))

```

Return tuple of zero or more articulations.

marktools.get_lilypond_command_mark_attached_to_component

abjad.tools.marktools.get_lilypond_command_mark_attached_to_component.get_lilypond_command

New in version 2.0. Get exactly one LilyPond command mark attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.LilyPondCommandMark('stemUp')(staff[0])
LilyPondCommandMark('stemUp')(c'8)

```

```

abjad> f(staff)
\new Staff {
  \stemUp
  c'8
  d'8
  e'8
  f'8
}

```

```

abjad> marktools.get_lilypond_command_mark_attached_to_component(staff[0])
LilyPondCommandMark('stemUp')(c'8)

```

Return one LilyPond command mark.

Raise missing mark error when no LilyPond command mark is attached.

Raise extra mark error when more than one LilyPond command mark is attached.

marktools.get_lilypond_command_marks_attached_to_component

abjad.tools.marktools.get_lilypond_command_marks_attached_to_component.get_lilypond_command

New in version 2.0. Get LilyPond command marks attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted')(c'8)
abjad> marktools.LilyPondCommandMark('slurUp')(staff[0])
LilyPondCommandMark('slurUp')(c'8)

```

```

abjad> f(staff)
\new Staff {
  \slurDotted
  \slurUp
  c'8 (
  d'8

```

```
e'8
f'8 )
}
```

```
abjad> marktools.get_lilypond_command_marks_attached_to_component(staff[0])
(LilyPondCommandMark('slurDotted')(c'8), LilyPondCommandMark('slurUp')(c'8))
```

Return tuple of zero or more marks.

marktools.get_lilypond_comment_attached_to_component

abjad.tools.marktools.get_lilypond_comment_attached_to_component.get_lilypond_comment_attached_to_component

New in version 2.0. Get exactly one LilyPond comment attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.LilyPondComment('comment')(staff[0])
LilyPondComment('comment')(c'8)
```

```
abjad> f(staff)
\new Staff {
  % comment
  c'8
  d'8
  e'8
  f'8
}
```

```
abjad> marktools.get_lilypond_comment_attached_to_component(staff[0])
LilyPondComment('comment')(c'8)
```

Return one LilyPond comment.

Raise missing mark error when no LilyPond comment is attached.

Raise extra mark error when more than one LilyPond comment is attached.

marktools.get_lilypond_comments_attached_to_component

abjad.tools.marktools.get_lilypond_comments_attached_to_component.get_lilypond_comments_attached_to_component

New in version 2.0. Get LilyPond comments attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> marktools.LilyPondComment('comment 1')(staff[0])
LilyPondComment('comment 1')(c'8)
abjad> marktools.LilyPondComment('comment 2')(staff[0])
LilyPondComment('comment 2')(c'8)
```

```
abjad> f(staff)
\new Staff {
  % comment 1
  % comment 2
  c'8 (
  d'8
  e'8
  f'8 )
}
```



```
abjad> marktools.get_lilypond_comments_attached_to_component(staff[0])
(LilyPondComment('comment 1')(c'8), LilyPondComment('comment 2')(c'8))
```

Return tuple of zero or more LilyPond comments.

marktools.get_mark_attached_to_component

`abjad.tools.marktools.get_mark_attached_to_component.get_mark_attached_to_component` (*component*)
 New in version 2.0. Get exactly one mark attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Mark()(staff[0])
Mark()(c'8)

abjad> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}

abjad> marktools.get_mark_attached_to_component(staff[0])
Mark()(c'8)
```

Return one mark.

Raise missing mark error when no mark is attached.

Raise extra mark error when more than one mark is attached.

marktools.get_marks_attached_to_component

`abjad.tools.marktools.get_marks_attached_to_component.get_marks_attached_to_component` (*component*)
 New in version 2.0. Get all marks attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> comment_mark = marktools.LilyPondComment('beginning of note content')(staff[0])
abjad> marktools.LilyPondCommandMark('slurDotted')(staff[0])
LilyPondCommandMark('slurDotted')(c'8)

abjad> f(staff)
\new Staff {
  % beginning of note content
  \slurDotted
  c'8 (
  d'8
  e'8
  f'8 )
}

abjad> marktools.get_marks_attached_to_component(staff[0])
(LilyPondComment('beginning of note content')(c'8), LilyPondCommandMark('slurDotted')(c'8))
```

Return tuple of zero or more marks. Changed in version 2.0: re-named `marktools.get_all_marks_attached_to_component()` to `marktools.get_marks_attached_to_component()`.

marktools.get_noncontext_mark_attached_to_component

`abjad.tools.marktools.get_noncontext_mark_attached_to_component.get_noncontext_mark_attached_to_component`
 New in version 2.0. Get exactly one `noncontext_mark` attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

abjad> f(staff)
\new Staff {
  c'8 -\staccato
  d'8
  e'8
  f'8
}

abjad> marktools.get_noncontext_mark_attached_to_component(staff[0])
Articulation('staccato')(c'8)
```

Return one `noncontext_mark`.

Raise missing mark error when no `noncontext_mark` is attached.

Raise extra mark error when more than one `noncontext_mark` is attached.

marktools.get_noncontext_marks_attached_to_component

`abjad.tools.marktools.get_noncontext_marks_attached_to_component.get_noncontext_marks_attached_to_component`
 New in version 2.0. Get noncontext marks attached to component:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> contexttools.TimeSignatureMark((2, 4))(staff[0])
TimeSignatureMark((2, 4))(c'8)
abjad> marktools.Articulation('staccato')(staff[0])
Articulation('staccato')(c'8)

abjad> f(staff)
\new Staff {
  \time 2/4
  c'8 -\staccato
  d'8
  e'8
  f'8
}

abjad> marktools.get_noncontext_marks_attached_to_component(staff[0])
(Articulation('staccato')(c'8),)
```

Return tuple of zero or more marks.

marktools.get_stem_tremolo_attached_to_component

`abjad.tools.marktools.get_stem_tremolo_attached_to_component.get_stem_tremolo_attached_to_component`

New in version 2.0. Get stem tremolo attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

abjad> f(staff)
\new Staff {
    c'8 :16
    d'8
    e'8
    f'8
}

abjad> marktools.get_stem_tremolo_attached_to_component(staff[0])
StemTremolo(16)(c'8)
```

Raise missing mark error when no stem tremolo attaches to *component*.

Raise extra mark error when more than one stem tremolo attaches to *component*.

Return stem tremolo.

marktools.get_stem_tremolos_attached_to_component

`abjad.tools.marktools.get_stem_tremolos_attached_to_component.get_stem_tremolos_attached_to_component`

New in version 2.3. Get stem tremolos attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.StemTremolo(16)(staff[0])
StemTremolo(16)(c'8)

abjad> f(staff)
\new Staff {
    c'8 :16
    d'8
    e'8
    f'8
}

abjad> marktools.get_stem_tremolos_attached_to_component(staff[0])
(StemTremolo(16)(c'8),)
```

Return tuple of zero or more stem tremolos.

marktools.get_value_of_annotation_attached_to_component

`abjad.tools.marktools.get_value_of_annotation_attached_to_component.get_value_of_annotation_attached_to_component`

New in version 2.0. Get value of annotation with *name* attached to *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> marktools.Annotation('special dictionary', {})(staff[0])
Annotation('special dictionary', {})(c'8)

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> marktools.get_value_of_annotation_attached_to_component(staff[0], 'special dictionary')
{}

```

Return arbitrary value of annotation.

Return *default_value* when no annotation with *name* is attached.

Raise extra mark error when more than one annotation with *name* is attached.

marktools.is_component_with_annotation_attached

`abjad.tools.marktools.is_component_with_annotation_attached.is_component_with_annotation_attached`

New in version 2.3. True when *expr* is component with annotation attached:

```

abjad> note = Note("c'4")
abjad> marktools.Annotation('foo', 'bar')(note)
Annotation('foo', 'bar')(c'4)

abjad> marktools.is_component_with_annotation_attached(note)
True

```

False otherwise:

```

abjad> note = Note("c'4")

abjad> marktools.is_component_with_annotation_attached(note)
False

```

Return boolean.

marktools.is_component_with_articulation_attached

abjad.tools.marktools.is_component_with_articulation_attached.is_component_with_articulation_attached

New in version 2.3. True when *expr* is component with articulation attached:

```
abjad> note = Note("c'4")
abjad> marktools.Articulation('staccato')(note)
Articulation('staccato')(c'4)

abjad> marktools.is_component_with_articulation_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

abjad> marktools.is_component_with_articulation_attached(note)
False
```

Return boolean.

marktools.is_component_with_lilypond_command_mark_attached

abjad.tools.marktools.is_component_with_lilypond_command_mark_attached.is_component_with_lilypond_command_mark_attached

New in version 2.0. True when *expr* is component with LilyPond command mark attached:

```
abjad> note = Note("c'4")
abjad> marktools.LilyPondCommandMark('stemUp')(note)
LilyPondCommandMark('stemUp')(c'4)

abjad> marktools.is_component_with_lilypond_command_mark_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

abjad> marktools.is_component_with_lilypond_command_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_lilypond_comment_attached

abjad.tools.marktools.is_component_with_lilypond_comment_attached.is_component_with_lilypond_comment_attached

New in version 2.3. True when *expr* is component with LilyPond comment mark attached:

```
abjad> note = Note("c'4")
abjad> marktools.LilyPondComment('comment here')(note)
LilyPondComment('comment here')(c'4)
```

```
abjad> marktools.is_component_with_lilypond_comment_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

abjad> marktools.is_component_with_lilypond_comment_attached(note)
False
```

Return boolean.

marktools.is_component_with_mark_attached

`abjad.tools.marktools.is_component_with_mark_attached.is_component_with_mark_attached(expr)`
New in version 2.3. True when *expr* is component with mark attached:

```
abjad> note = Note("c'4")
abjad> marktools.Mark()(note)
Mark()(c'4)

abjad> marktools.is_component_with_mark_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

abjad> marktools.is_component_with_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_noncontext_mark_attached

`abjad.tools.marktools.is_component_with_noncontext_mark_attached.is_component_with_noncontext_mark_attached(expr)`
New in version 2.3. True when *expr* is component with noncontext mark attached:

```
abjad> note = Note("c'4")
abjad> marktools.Articulation('staccato')(note)
Articulation('staccato')(c'4)

abjad> marktools.is_component_with_noncontext_mark_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

abjad> marktools.is_component_with_noncontext_mark_attached(note)
False
```

Return boolean.

marktools.is_component_with_stem_tremolo_attached

`abjad.tools.marktools.is_component_with_stem_tremolo_attached.is_component_with_stem_tremolo_attached`
 New in version 2.3. True when *expr* is component with LilyPond command mark attached:

```
abjad> note = Note("c'4")
abjad> marktools.StemTremolo(16)(note)
StemTremolo(16)(c'4)
```

```
abjad> marktools.is_component_with_stem_tremolo_attached(note)
True
```

False otherwise:

```
abjad> note = Note("c'4")

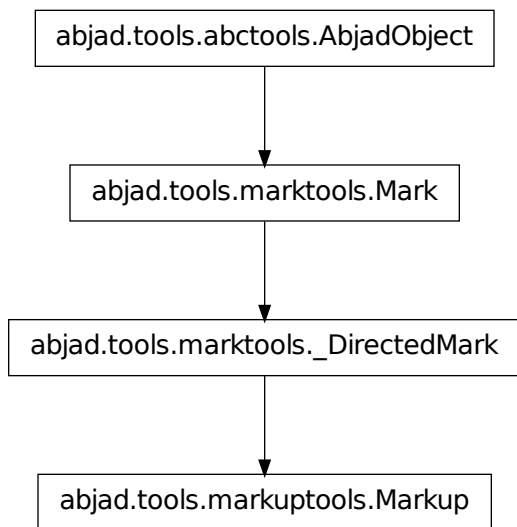
abjad> marktools.is_component_with_stem_tremolo_attached(note)
False
```

Return boolean.

markuptools

markuptools concrete classes

markuptools.Markup



```
class abjad.tools.markuptools.Markup.Markup(argument, direction=None,
                                             markup_name=None,
                                             style_string=None)
    Abjad model of backslash-style LilyPond markup or Scheme-style LilyPond markup.
```

Initialize backslash-style markup from string:

```
abjad> markup = markuptools.Markup(r'\bold { "This is markup text." }')

abjad> markup
Markup('\bold { "This is markup text." }')

abjad> f(markup)
\markup { \bold { "This is markup text." } }
```

Initialize Scheme-style markup from string:

```
abjad> markup = markuptools.Markup("(markup #:draw-line '(0 . -1))", style_string='scheme')

abjad> markup
Markup("(markup #:draw-line '(0 . -1))", style_string='scheme')

abjad> f(markup)
#(markup #:draw-line '(0 . -1))
```

Initialize any markup from existing markup:

```
abjad> markup_1 = markuptools.Markup('foo', direction='up')
abjad> markup_2 = markuptools.Markup(markup_1, direction='down')

abjad> f(markup_1)
^ \markup { foo }

abjad> f(markup_2) # doctest: +SKIP
_ \markup { foo }
```

Attach markup to score components like this:

```
abjad> note = Note("c'4")

abjad> markup = markuptools.Markup(r'\bold { "This is markup text." }')

abjad> markup(note)
Markup('\bold { "This is markup text." }')(c'4)

abjad> f(note)
c'4 - \markup { \bold { "This is markup text." } }
```

Set *direction* to 'up', 'down', 'neutral', '^', '_', '-' or None.

Set *style_string* to 'backslash', 'scheme' or none. Default to 'backslash'.

Markup objects are immutable.

Read-only Properties

Markup.contents_string

Read-only contents string of markup:

```
abjad> markup = markuptools.Markup(r'\bold { "This is markup text." }')
abjad> markup.contents_string
'\bold { "This is markup text." }'
```

Return string

Markup.format

Read-only LilyPond format of markup:

```
abjad> markup = markuptools.Markup(r'\bold { "This is markup text." }')
abjad> markup.format
'\\markup { \\bold { "This is markup text." } }'
```

Return string.

Markup.markup_name

Read-only name of markup:

```
abjad> markup = markuptools.Markup(r'\bold { allegro ma non troppo }', markup_name='non troppo')

abjad> markup.markup_name
'non troppo'
```

Return string or none.

Markup.start_component

Read-only reference to mark start component:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)

abjad> mark.start_component
Note("c'4")
```

Return component or none.

Note: Inherited from `marktools.Mark`

Markup.style_string

Read-only style string of markup.

Read/write Properties**Markup.direction**

Note: Inherited from `marktools._DirectedMark`

Methods**Markup.attach**(*start_component*)

Attach mark to *start_component*:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()

abjad> mark.attach(note)
Mark()(c'4)

abjad> mark.start_component
Note("c'4")
```

Return mark.

Note: Inherited from `marktools.Mark`

Markup.**detach**()

Detach mark:

```
abjad> note = Note("c'4")
abjad> mark = marktools.Mark()(note)
```

```
abjad> mark.start_component
Note("c'4")
```

```
abjad> mark.detach()
Mark()
```

```
abjad> mark.start_component is None
True
```

Return mark.

Note: Inherited from `marktools.Mark`

Special Methods

Markup.**__call__**(*args)

Note: Inherited from `marktools.Mark`

Markup.**__delattr__**(*args)

Note: Inherited from `marktools.Mark`

Markup.**__eq__**(expr)

Markup.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Markup.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Markup.**__hash__**()

Markup.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Markup.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Markup.__ne__(expr)

Markup.__repr__()

Note: Inherited from `marktools.Mark`

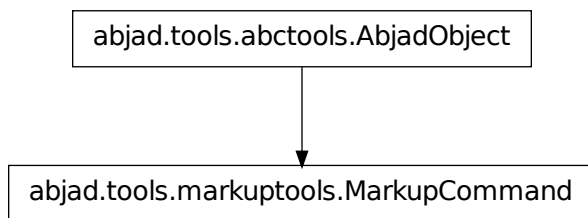
Markup.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Markup.__str__()

markuptools.MarkupCommand



```
class abjad.tools.markuptools.MarkupCommand.MarkupCommand(command,
                                                           *args)
```

Abjad model of a LilyPond markup command:

```

abjad> circle = markuptools.MarkupCommand('draw-circle', 2.5, 0.1, False)
abjad> square = markuptools.MarkupCommand('rounded-box', 'hello?')
abjad> line = markuptools.MarkupCommand('line', [square, 'wow!'])
abjad> rotate = markuptools.MarkupCommand('rotate', 60, line)
abjad> combine = markuptools.MarkupCommand('combine', rotate, circle)
  
```

```
abjad> print combine
\combine \rotate #60 \line { \rounded-box hello? wow! } \draw-circle #2.5 #0.1 ##f
```

Insert markup command in markup to attach to score components:

```
abjad> note = Note("c'4")

abjad> markup = markuptools.Markup(combine)

abjad> markup(note)
Markup('\combine \rotate #60 \line { \rounded-box hello? wow! } \draw-circle #2.5 #0.1 ##f'

abjad> f(note)
c'4 - \markup { \combine \rotate #60 \line { \rounded-box hello? wow! } \draw-circle #2.5 #0.1 #
```

Markup commands are immutable.

Read-only Properties

`MarkupCommand.args`

Read-only tuple of markup command arguments.

`MarkupCommand.command`

Read-only string of markup command command-name.

`MarkupCommand.format`

Read-only format of markup command:

```
abjad> markup_command = markuptools.MarkupCommand('draw-circle', 2.5, 0.1, False)
abjad> markup_command.format
'\draw-circle #2.5 #0.1 ##f'
```

Returns string.

Special Methods

`MarkupCommand.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MarkupCommand.__eq__(other)`

`MarkupCommand.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MarkupCommand.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

MarkupCommand.__hash__() \Leftrightarrow hash(*x*)

Note: Inherited from `__builtin__.object`

MarkupCommand.__le__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MarkupCommand.__lt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MarkupCommand.__ne__(*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

MarkupCommand.__repr__()

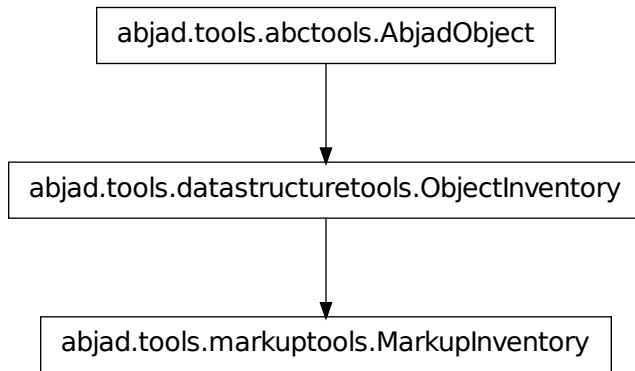
MarkupCommand.__setattr__()

`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

MarkupCommand.__str__()

markuptools.MarkupInventory



class `abjad.tools.markuptools.MarkupInventory.MarkupInventory` (*tokens=None, inventory_name=None*)

New in version 2.8. Abjad model of an ordered list of markup:

```

abjad> inventory = markuptools.MarkupInventory(['foo', 'bar'])

abjad> inventory
MarkupInventory([Markup('foo'), Markup('bar')])
  
```

Markup inventories implement the list interface and are mutable.

Read/write Properties

`MarkupInventory.inventory_name`
Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`MarkupInventory.append(token)`
Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.count(value) → integer` – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`MarkupInventory.extend(tokens)`
Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.index(value[, start[, stop]])` → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`MarkupInventory.insert()`
`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`MarkupInventory.pop([index])` → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`MarkupInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`MarkupInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`MarkupInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`MarkupInventory.__add__()`
`x.__add__(y)` <==> `x+y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.__delattr__()`
`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`MarkupInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`MarkupInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`MarkupInventory.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`MarkupInventory.__iter__()` $\iff iter(x)$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__le__()`
`x.__le__(y)` $\iff x \leq y$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__len__()` $\iff len(x)$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__mul__()`
`x.__mul__(n)` $\iff x * n$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.list`

`MarkupInventory.__repr__()`

Note: Inherited from `datastructuretools.ObjectInventory`

`MarkupInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`MarkupInventory.__rmul__()`
`x.__rmul__(n)` $\iff n * x$

Note: Inherited from `__builtin__.list`

```
MarkupInventory.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
MarkupInventory.__setitem__()  
x.__setitem__(i, y) <==> x[i]=y
```

Note: Inherited from `__builtin__.list`

```
MarkupInventory.__setslice__()  
x.__setslice__(i, j, y) <==> x[i:j]=y
```

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
MarkupInventory.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

markuptools functions

markuptools.all_are_markup

`abjad.tools.markuptools.all_are_markup.all_are_markup(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad markup:

```
abjad> markup = markuptools.Markup('some text')
```

```
abjad> markuptools.all_are_markup([markup])  
True
```

True when *expr* is an empty sequence:

```
abjad> markuptools.all_are_markup([])  
True
```

Otherwise false:

```
abjad> markuptools.all_are_markup('foo')  
False
```

Return boolean.

markuptools.combine_markup_commands

`abjad.tools.markuptools.combine_markup_commands.combine_markup_commands(*commands)`

Combine MarkupCommand and/or string objects.

LilyPond's 'combine' markup command can only take two arguments, so in order to combine more than two stencils, a cascade of 'combine' commands must be employed. `combine_markup_commands` simplifies this process.

```
abjad> from abjad.tools.markuptools import combine_markup_commands
abjad> from abjad.tools.schemetools import SchemePair
abjad> from abjad.tools.markuptools import MarkupCommand

abjad> markup_a = MarkupCommand('draw-circle', 4, 0.4, False)
abjad> markup_b = MarkupCommand('filled-box', SchemePair(-4, 4), SchemePair(-0.5, 0.5), 1)
abjad> markup_c = "some text"
abjad> print combine_markup_commands(markup_a, markup_b, markup_c).format
\combine \combine \draw-circle #4 #0.4 ##f \filled-box #'(-4 . 4) #'(-0.5 . 0.5) #1 #"some text"
```

Returns a MarkupCommand instance, or a string if that was the only argument.

markuptools.get_down_markup_attached_to_component

`abjad.tools.markuptools.get_down_markup_attached_to_component.get_down_markup_attached_to_component`

New in version 2.0. Get down-markup attached to component:

```
abjad> chord = Chord([-11, 2, 5], (1, 4))
abjad> markuptools.Markup('UP', 'up')(chord)
Markup('UP', direction='^')(<cs d' f'>4)
abjad> markuptools.Markup('DOWN', 'down')(chord)
Markup('DOWN', direction='_')(<cs d' f'>4)

abjad> markuptools.get_down_markup_attached_to_component(chord)
(Markup('DOWN', direction='_')(<cs d' f'>4),)
```

Return tuple of zero or more markup objects.

markuptools.get_markup_attached_to_component

`abjad.tools.markuptools.get_markup_attached_to_component.get_markup_attached_to_component`

New in version 2.0. Get markup attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff[:])
abjad> markuptools.Markup('foo')(staff[0])
Markup('foo')(c'8)
abjad> markuptools.Markup('bar')(staff[0])
Markup('bar')(c'8)

abjad> f(staff)
\new Staff {
  c'8 - \markup { \column { foo bar } } (
  d'8
  e'8
  f'8 )
}

abjad> markuptools.get_markup_attached_to_component(staff[0])
(Markup('foo')(c'8), Markup('bar')(c'8))
```

Return tuple of zero or more markup objects.

markuptools.get_up_markup_attached_to_component

abjad.tools.markuptools.get_up_markup_attached_to_component.get_up_markup_attached_to_component

New in version 2.0. Get up-markup attached to component:

```
abjad> chord = Chord([-11, 2, 5], (1, 4))
abjad> markuptools.Markup('UP', 'up')(chord)
Markup('UP', direction='^')(<cs d' f'>4)
abjad> markuptools.Markup('DOWN', 'down')(chord)
Markup('DOWN', direction='_')(<cs d' f'>4)

abjad> markuptools.get_up_markup_attached_to_component(chord)
(Markup('UP', direction='^')(<cs d' f'>4),)
```

Return tuple of zero or more markup objects.

markuptools.make_big_centered_page_number_markup

abjad.tools.markuptools.make_big_centered_page_number_markup.make_big_centered_page_number_markup

New in version 1.1. Make big centered page number markup:

```
abjad> markup = markuptools.make_big_centered_page_number_markup()

abjad> f(markup)
\markup {
  \fill-line {
    \bold \fontsize #3 \concat {
      \on-the-fly #print-page-number-check-first
      \fromproperty #'page:page-number-string } } }
```

Return markup. Changed in version 2.0: renamed `markuptools.big_centered_page_number()` to `markuptools.make_big_centered_page_number_markup()`.

markuptools.remove_markup_attached_to_component

abjad.tools.markuptools.remove_markup_attached_to_component.remove_markup_attached_to_component

New in version 2.0. Remove markup attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> slur = spannertools.SlurSpanner(staff[:])
abjad> markuptools.Markup('foo')(staff[0])
Markup('foo')(c'8)
abjad> markuptools.Markup('bar')(staff[0])
Markup('bar')(c'8)

abjad> f(staff)
\new Staff {
  c'8 - \markup { \column { foo bar } } (
  d'8
  e'8
  f'8 )
}

abjad> markuptools.remove_markup_attached_to_component(staff[0])
(Markup('foo'), Markup('bar'))
```

```
abjad> f(staff)
\new Staff {
  c'8 (
  d'8
  e'8
  f'8 )
}
```

Return tuple of zero or more markup objects.

markuptools.remove_markup_from_leaves_in_expr

`abjad.tools.markuptools.remove_markup_from_leaves_in_expr.remove_markup_from_leaves_in_expr`

New in version 1.1. Remove markup from leaves in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> leaftools.label_leaves_in_expr_with_pitch_class_numbers(staff)
abjad> f(staff)
\new Staff {
  c'8 _ \markup { \small 0 }
  d'8 _ \markup { \small 2 }
  e'8 _ \markup { \small 4 }
  f'8 _ \markup { \small 5 }
}

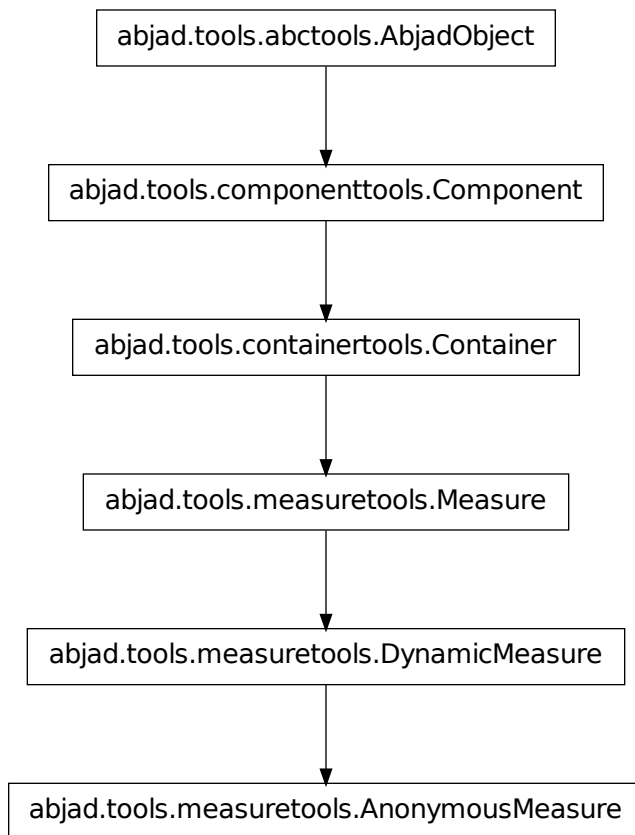
abjad> markuptools.remove_markup_from_leaves_in_expr(staff)
abjad> f(staff)
\new Staff {
  c'8
  d'8
  e'8
  f'8
}
```

Return `none`. Changed in version 2.0: renamed `label.clear_leaves()` to `markuptools.remove_markup_from_leaves_in_expr()`.

measuretools

measuretools concrete classes

measuretools.**AnonymousMeasure**



class `abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure`.**AnonymousMeasure** (*music=None*,
**kwargs)

New in version 1.1. Dynamic measure with no time signature:

```
abjad> measure = measuretools.AnonymousMeasure("c'8 d'8 e'8 f'8")
```

```
abjad> f(measure)
{
    \override Staff.TimeSignature #'stencil = ##f
    \time 1/2
    c'8
    d'8
    e'8
    f'8
    \revert Staff.TimeSignature #'stencil
}
```

```
}

abjad> notes = [Note("c'8"), Note("d'8")]
abjad> measure.extend(notes)

abjad> f(measure)
{
    \override Staff.TimeSignature #'stencil = ##f
    \time 3/4
    c'8
    d'8
    e'8
    f'8
    c'8
    d'8
    \revert Staff.TimeSignature #'stencil
}
```

Return anonymous measure.

Read-only Properties

`AnonymousMeasure.contents_duration`

Note: Inherited from `containertools.Container`

`AnonymousMeasure.duration_in_seconds`

Note: Inherited from `containertools.Container`

`AnonymousMeasure.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`AnonymousMeasure.is_binary`

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.is_full`

True when meter matches duration of measure:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")
```

```
abjad> measure.is_full
```

```
True
```

False otherwise:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8")
```

```
abjad> measure.is_full
False
```

Return boolean.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.is_nonbinary`

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.is_overfull`

New in version 1.1. True when prolated duration is greater than effective meter duration.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.is_underfull`

New in version 1.1. True when prolated duration is less than effective meter duration.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`AnonymousMeasure.measure_number`

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.multiplier`

Note: Inherited from `measuretools.Measure`

AnonymousMeasure.**music**

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

AnonymousMeasure.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

AnonymousMeasure.**parent**

Note: Inherited from `componenttools.Component`

AnonymousMeasure.**preprolated_duration**

Note: Inherited from `measuretools.DynamicMeasure`

AnonymousMeasure.**prolated_duration**

Note: Inherited from `componenttools.Component`

AnonymousMeasure.**prolation**

Note: Inherited from `componenttools.Component`

AnonymousMeasure.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

AnonymousMeasure.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties**AnonymousMeasure.denominator**

Get explicit denominator of dynamic measure:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")
```

```
abjad> measure.denominator is None
True
```

Set explicit denominator of dynamic measure:

```
abjad> measure.denominator = 8
```

```
abjad> f(measure)
{
    \time 4/8
    c'8
    d'8
    e'8
    f'8
}
```

Set positive integer or none.

Note: Inherited from `measuretools.DynamicMeasure`

AnonymousMeasure.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}
```

```
abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True
```

```
abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
```

```
\new Voice {  
  g4.  
}  
>>
```

Return none.

Note: Inherited from `containertools.Container`

AnonymousMeasure.**suppress_meter**

Get meter suppression indicator:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")
```

```
abjad> f(measure)  
{  
  \time 1/2  
  c'8  
  d'8  
  e'8  
  f'8  
}
```

```
abjad> measure.suppress_meter  
False
```

Set meter suppression indicator:

```
abjad> measure.suppress_meter = True
```

```
abjad> measure.suppress_meter  
True
```

```
abjad> f(measure)  
{  
  c'8  
  d'8  
  e'8  
  f'8  
}
```

Set boolean.

Note: Inherited from `measuretools.DynamicMeasure`

Methods

AnonymousMeasure.**append**(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")  
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)  
{  
  c'8 [  
  d'8
```

```
        e'8 ]
    }

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.extend(expr)`

Extend dynamic measure:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")

abjad> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}

abjad> measure.extend([Note("f'8"), Note("g'8")])

abjad> f(measure)
{
    \time 5/8
    c'8
    d'8
    e'8
    f'8
    g'8
}
```

Return none.

Note: Inherited from `measuretools.DynamicMeasure`

`AnonymousMeasure.index(component)`

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

AnonymousMeasure.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

AnonymousMeasure.**pop** (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`AnonymousMeasure.__add__(arg)`

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.__contains__(expr)`

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`AnonymousMeasure.__delitem__(i)`

Container deletion with meter adjustment.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__getitem__(i)`
Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`AnonymousMeasure.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__mul__(n)`

Note: Inherited from `componenttools.Component`

`AnonymousMeasure.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AnonymousMeasure.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__repr__()`

String form of measure with parentheses for interpreter display.

Note: Inherited from `measuretools.Measure`

`AnonymousMeasure.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`AnonymousMeasure.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`AnonymousMeasure.__setitem__(i, expr)`

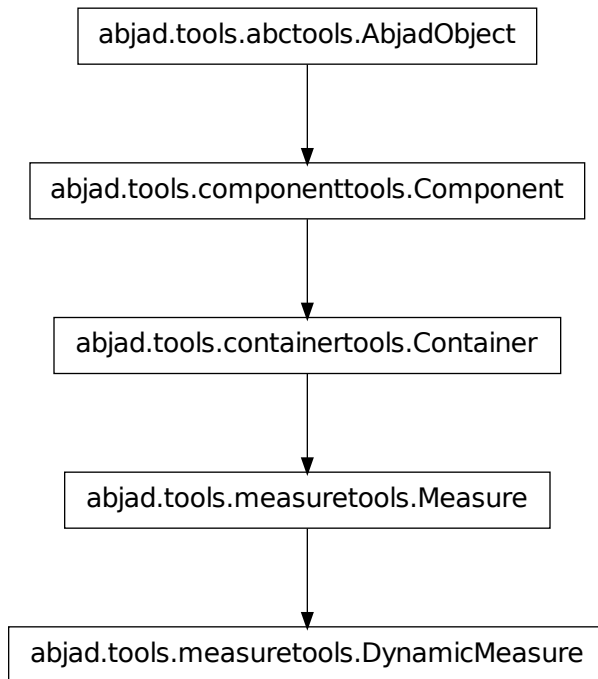
Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`AnonymousMeasure.__str__()`
String form of measure with pipes for single string display.

Note: Inherited from `measuretools.Measure`

`measuretools.DynamicMeasure`



class `abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` (*music=None*, ***kwargs*)

New in version 1.1. Measure sets meter dynamically to exactly equal contents duration:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")
```

```
abjad> measure
DynamicMeasure(3/8, [c'8, d'8, e'8])
```

```
abjad> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}
```

Return dynamic measure.

Read-only Properties

`DynamicMeasure.contents_duration`

Note: Inherited from `containertools.Container`

`DynamicMeasure.duration_in_seconds`

Note: Inherited from `containertools.Container`

`DynamicMeasure.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`DynamicMeasure.is_binary`

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.is_full`

True when meter matches duration of measure:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")
```

```
abjad> measure.is_full
True
```

False otherwise:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8")
```

```
abjad> measure.is_full
False
```

Return boolean.

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.is_nonbinary`

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.is_overfull`

New in version 1.1. True when prolated duration is greater than effective meter duration.

Note: Inherited from `measuretools.Measure`

DynamicMeasure.is_underfull

New in version 1.1. True when prolated duration is less than effective meter duration.

Note: Inherited from `measuretools.Measure`

DynamicMeasure.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

DynamicMeasure.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

DynamicMeasure.measure_number

Note: Inherited from `measuretools.Measure`

DynamicMeasure.multiplier

Note: Inherited from `measuretools.Measure`

DynamicMeasure.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

DynamicMeasure.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

DynamicMeasure.parent

Note: Inherited from `componenttools.Component`

`DynamicMeasure.preprolated_duration`

`DynamicMeasure.prolated_duration`

Note: Inherited from `componenttools.Component`

`DynamicMeasure.prolation`

Note: Inherited from `componenttools.Component`

`DynamicMeasure.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`DynamicMeasure.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`DynamicMeasure.denominator`

Get explicit denominator of dynamic measure:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")
```

```
abjad> measure.denominator is None
True
```

Set explicit denominator of dynamic measure:

```
abjad> measure.denominator = 8
```

```
abjad> f(measure)
{
  \time 4/8
  c'8
  d'8
  e'8
  f'8
}
```

Set positive integer or none.

`DynamicMeasure.is_parallel`

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

`DynamicMeasure.suppress_meter`

Get meter suppression indicator:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8 f'8")

abjad> f(measure)
{
    \time 1/2
    c'8
    d'8
    e'8
    f'8
}

abjad> measure.suppress_meter
False
```

Set meter suppression indicator:

```
abjad> measure.suppress_meter = True

abjad> measure.suppress_meter
True

abjad> f(measure)
{
    c'8
    d'8
    e'8
    f'8
}
```

Set boolean.

Methods

`DynamicMeasure.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`DynamicMeasure.extend(expr)`

Extend dynamic measure:

```
abjad> measure = measuretools.DynamicMeasure("c'8 d'8 e'8")

abjad> f(measure)
{
    \time 3/8
    c'8
    d'8
    e'8
}
```

```
abjad> measure.extend([Note("f'8"), Note("g'8")])

abjad> f(measure)
{
    \time 5/8
    c'8
    d'8
    e'8
    f'8
    g'8
}
```

Return none.

DynamicMeasure.**index** (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

DynamicMeasure.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

DynamicMeasure.**pop** (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

DynamicMeasure.**remove** (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`DynamicMeasure.__add__(arg)`

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.__contains__(expr)`

True if `expr` is in container, otherwise False.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DynamicMeasure.__delitem__(i)`

Container deletion with meter adjustment.

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DynamicMeasure.__iadd__(expr)`
__iadd__ avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__len__()`
Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__mul__(n)`

Note: Inherited from `componenttools.Component`

`DynamicMeasure.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DynamicMeasure.__radd__(expr)`
Extend container by contents of *expr* to the right.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__repr__()`
String form of measure with parentheses for interpreter display.

Note: Inherited from `measuretools.Measure`

`DynamicMeasure.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`DynamicMeasure.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

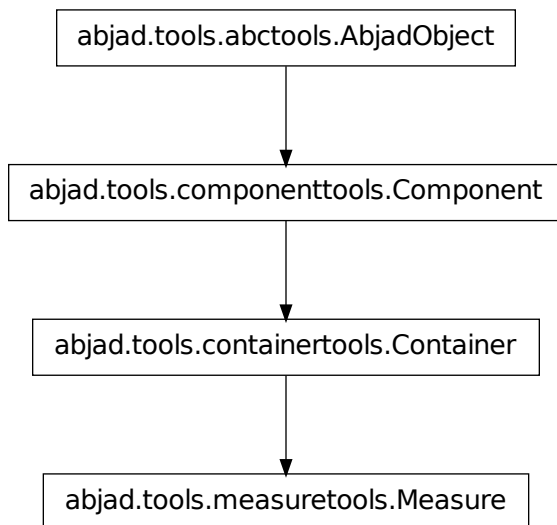
`DynamicMeasure.__setitem__(i, expr)`
 Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`DynamicMeasure.__str__()`
 String form of measure with pipes for single string display.

Note: Inherited from `measuretools.Measure`

measuretools.Measure



class `abjad.tools.measuretools.Measure.Measure.Measure` (*meter*, *music=None*, ***kwargs*)
New in version 1.1. Abjad model of a measure:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

abjad> measure
Measure(4/8, [c'8, d'8, e'8, f'8])

abjad> f(measure)
{
    \time 4/8
    c'8
    d'8
    e'8
    f'8
}
```

Return measure object.

Read-only Properties

`Measure.contents_duration`

Note: Inherited from `containertools.Container`

`Measure.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Measure.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`Measure.is_binary`

`Measure.is_full`

True when meter matches duration of measure:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8 f'8")

abjad> measure.is_full
True
```

False otherwise:

```
abjad> measure = Measure((4, 8), "c'8 d'8 e'8")

abjad> measure.is_full
False
```

Return boolean.

`Measure.is_nonbinary`

Measure.is_overfull

New in version 1.1. True when prolated duration is greater than effective meter duration.

Measure.is_underfull

New in version 1.1. True when prolated duration is less than effective meter duration.

Measure.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Measure.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Measure.measure_number**Measure.multiplier****Measure.music**

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Measure.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Measure.parent

Note: Inherited from `componenttools.Component`

Measure.preprolated_duration

Measure contents duration times effective meter multiplier.

Measure.prolated_duration

Note: Inherited from `componenttools.Component`

`Measure.prolation`

Note: Inherited from `componenttools.Component`

`Measure.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`Measure.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`Measure.is_parallel`

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')] )
```

```
abjad> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
abjad> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True
```

```
abjad> f(container)
```

```
<<
```

```
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
```

```
}  
>>
```

Return none.

Note: Inherited from `containertools.Container`

Methods

Measure.**append** (*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")  
abjad> beam = spannertools.BeamSpanner(container.music)  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]  
}  
  
abjad> container.append(Note("f'8"))  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]  
    f'8  
}
```

Return none.

Note: Inherited from `containertools.Container`

Measure.**extend** (*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")  
abjad> beam = spannertools.BeamSpanner(container.music)  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]  
}  
  
abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]  
    cs'8
```

```

        ds'8
        es'8
    }

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Measure.**index** (*component*)

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2

```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Measure.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Note: Inherited from `containertools.Container`

Measure.**pop** (*i=-1*)

Pop component at index *i* from container:


```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`Measure.remove` (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`Measure.__add__` (*arg*)

Add two measures together in-score or outside-of-score. Wrapper around `measuretools.fuse_measures`.

Measure.__contains__(*expr*)
True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

Measure.__delattr__()
x.__delattr__('name') <==> del *x*.name

Note: Inherited from `__builtin__.object`

Measure.__delitem__(*i*)
Container deletion with meter adjustment.

Measure.__eq__(*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Measure.__ge__(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Measure.__getitem__(*i*)
Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

Measure.__gt__(*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

Measure.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

Measure.__iadd__(*expr*)
__iadd__ avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

Measure.__imul__(*total*)
Multiply contents of container 'total' times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Measure.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Measure.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Measure.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Measure.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Measure.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Measure.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`Measure.__repr__()`

String form of measure with parentheses for interpreter display.

`Measure.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`Measure.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Measure.__setitem__(i, expr)`

Set ‘expr’ in self at nonnegative integer index *i*. Or, set ‘expr’ in self at slice *i*. Find spanners that dominate self[*i*] and children of self[*i*]. Replace contents at self[*i*] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`Measure.__str__()`

String form of measure with pipes for single string display.

measuretools functions

measuretools.all_are_measures

`abjad.tools.measuretools.all_are_measures.all_are_measures(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad measures:

```
abjad> measures = 3 * Measure((3, 4), "c'4 d'4 e'4")

abjad> measures
[Measure(3/4, [c'4, d'4, e'4]), Measure(3/4, [c'4, d'4, e'4]), Measure(3/4, [c'4, d'4, e'4])]

abjad> measuretools.all_are_measures(measures)
True
```

True when *expr* is an empty sequence:

```
abjad> measuretools.all_are_measures([])
True
```

Otherwise false:

```
abjad> measuretools.all_are_measures('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

measuretools.append_spacer_skip_to_underfull_measure

`abjad.tools.measuretools.append_spacer_skip_to_underfull_measure.append_spacer_skip_to_underfull_measure`

New in version 1.1. Append spacer skip to underfull *measure*:

```
abjad> measure = Measure((4, 12), "c'8 d'8 e'8 f'8")
abjad> contexttools.detach_time_signature_marks_attached_to_component(measure)
(TimeSignatureMark((4, 12)),)
abjad> contexttools.TimeSignatureMark((5, 12))(measure)
TimeSignatureMark((5, 12))(|5/12, c'8, d'8, e'8, f'8|)
abjad> measure.is_underfull
True

abjad> measuretools.append_spacer_skip_to_underfull_measure(measure)
Measure(5/12, [c'8, d'8, e'8, f'8, s1 * 1/8])
```

```

abjad> f(measure)
{
    \time 5/12
    \scaleDurations #'(2 . 3) {
        c'8
        d'8
        e'8
        f'8
        s1 * 1/8
    }
}

```

Append nothing to nonunderfull *measure*.

Return *measure*. Changed in version 2.0: renamed `measuretools.make_measures_with_full_measure_spacer_s` to `measuretools.append_spacer_skip_to_underfull_measure()`.

`measuretools.append_spacer_skips_to_underfull_measures_in_expr`

`abjad.tools.measuretools.append_spacer_skips_to_underfull_measures_in_expr.append_spacer_s`

New in version 1.1. Append spacer skips to underfull measures in *expr*:

```

abjad> staff = Staff(Measure((3, 8), "c'8 d'8 e'8") * 3)
abjad> contexttools.detach_time_signature_marks_attached_to_component(staff[1])
(TimeSignatureMark((3, 8)),)
abjad> contexttools.TimeSignatureMark((4, 8))(staff[1])
TimeSignatureMark((4, 8))(|4/8, c'8, d'8, e'8|)
abjad> contexttools.detach_time_signature_marks_attached_to_component(staff[2])
(TimeSignatureMark((3, 8)),)
abjad> contexttools.TimeSignatureMark((5, 8))(staff[2])
TimeSignatureMark((5, 8))(|5/8, c'8, d'8, e'8|)
abjad> staff[1].is_underfull
True
abjad> staff[2].is_underfull
True

abjad> measuretools.append_spacer_skips_to_underfull_measures_in_expr(staff)
[Measure(4/8, [c'8, d'8, e'8, s1 * 1/8]), Measure(5/8, [c'8, d'8, e'8, s1 * 1/4])]

abjad> f(staff)
\new Staff {
    {
        \time 3/8
        c'8
        d'8
        e'8
    }
    {
        \time 4/8
        c'8
        d'8
        e'8
        s1 * 1/8
    }
    {
        \time 5/8
        c'8

```

```

        d'8
        e'8
        s1 * 1/4
    }
}

```

Return measures treated. Changed in version 2.0: renamed `measuretools.remedy_underfull_measures()` to `measuretools.append_spacer_skips_to_underfull_measures_in_expr()`.

`measuretools.apply_beam_spanner_to_measure`

`abjad.tools.measuretools.apply_beam_spanner_to_measure.apply_beam_spanner_to_measure(measure)`

New in version 2.0. Apply beam spanner to *measure*:

```

abjad> measure = Measure((2, 8), "c'8 d'8")

abjad> f(measure)
{
    \time 2/8
    c'8
    d'8
}

abjad> measuretools.apply_beam_spanner_to_measure(measure)
BeamSpanner(|2/8(2)|)

abjad> f(measure)
{
    \time 2/8
    c'8 [
    d'8 ]
}

```

Return beam spanner.

`measuretools.apply_beam_spanners_to_measures_in_expr`

`abjad.tools.measuretools.apply_beam_spanners_to_measures_in_expr.apply_beam_spanners_to_measures_in_expr(expr)`

New in version 1.1. Apply beam spanners to measures in *expr*:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
}

```

```

abjad> measuretools.apply_beam_spanners_to_measures_in_expr(staff)
[BeamSpanner(|2/8(2)|), BeamSpanner(|2/8(2)|)]

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 [
    d'8 ]
  }
  {
    \time 2/8
    e'8 [
    f'8 ]
  }
}

```

Return list of beams created. Changed in version 2.0: renamed `measuretools.beam()` to `measuretools.apply_beam_spanners_to_measures_in_expr()`.

`measuretools.apply_complex_beam_spanner_to_measure`

`abjad.tools.measuretools.apply_complex_beam_spanner_to_measure.apply_complex_beam_spanner_to_measure`
 New in version 2.0. Apply complex beam spanner to *measure*:

```

abjad> measure = Measure((2, 8), "c'8 d'8")

abjad> f(measure)
{
  \time 2/8
  c'8
  d'8
}

abjad> measuretools.apply_complex_beam_spanner_to_measure(measure)
DulatedComplexBeamSpanner(|2/8(2)|)

abjad> f(measure)
{
  \time 2/8
  \set stemLeftBeamCount = #0
  \set stemRightBeamCount = #1
  c'8 [
  \set stemLeftBeamCount = #1
  \set stemRightBeamCount = #0
  d'8 ]
}

```

Return complex beam spanner.

`measuretools.apply_complex_beam_spanners_to_measures_in_expr`

`abjad.tools.measuretools.apply_complex_beam_spanners_to_measures_in_expr.apply_complex_beam_spanners_to_measures_in_expr`
 New in version 2.0. Apply complex beam spanners to measures in *expr*:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
}

abjad> measuretools.apply_complex_beam_spanners_to_measures_in_expr(staff)
[DuratedComplexBeamSpanner(|2/8(2)|), DuratedComplexBeamSpanner(|2/8(2)|)]

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        \set stemLeftBeamCount = #0
        \set stemRightBeamCount = #1
        c'8 [
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #0
        d'8 ]
    }
    {
        \time 2/8
        \set stemLeftBeamCount = #0
        \set stemRightBeamCount = #1
        e'8 [
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #0
        f'8 ]
    }
}

```

Return list of beams created.

measuretools.apply_durated_complex_beam_spanner_to_measures

abjad.tools.measuretools.apply_durated_complex_beam_spanner_to_measures.**apply_durated_complex_beam_spanner_to_measures**

New in version 1.1. Apply durated complex beam spanner to *measures*:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
    }
}

```



```

        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
}

abjad> measures = staff[:]
abjad> measuretools.apply_durated_complex_beam_spanner_to_measures(measures)
DuratedComplexBeamSpanner(|2/8(2)|, |2/8(2)|)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        \set stemLeftBeamCount = #0
        \set stemRightBeamCount = #1
        c'8 [
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #1
        d'8
    }
    {
        \time 2/8
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #1
        e'8
        \set stemLeftBeamCount = #1
        \set stemRightBeamCount = #0
        f'8 ]
    }
}

```

Set beam spanner durations to preprolated measure durations.

Return beam spanner created. Changed in version 2.0: renamed `measuretools.beam_together()`.

`measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr`

`abjad.tools.measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr.apply_`

New in version 2.0. Apply full-measure tuplets to contents of measures in *expr*:

```

abjad> staff = Staff([Measure((2, 8), "c'8 d'8"), Measure((3, 8), "e'8 f'8 g'8")])

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 3/8
        e'8
        f'8
    }
}

```

```

        g'8
    }
}

abjad> measuretools.apply_full_measure_tuplets_to_contents_of_measures_in_expr(staff)

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    {
      c'8
      d'8
    }
  }
  {
    \time 3/8
    {
      e'8
      f'8
      g'8
    }
  }
}

```

Return none.

measuretools.color_measure

`abjad.tools.measuretools.color_measure.color_measure(measure, color='red')`

New in version 2.0. Color *measure* with *color*:

```

abjad> measure = Measure((2, 8), "c'8 d'8")

abjad> f(measure)
{
  \time 2/8
  c'8
  d'8
}

abjad> measuretools.color_measure(measure, 'red')
Measure(2/8, [c'8, d'8])

abjad> f(measure)
{
  \override Beam #'color = #red
  \override Dots #'color = #red
  \override NoteHead #'color = #red
  \override Staff.TimeSignature #'color = #red
  \override Stem #'color = #red
  \time 2/8
  c'8
  d'8
  \revert Beam #'color
  \revert Dots #'color
  \revert NoteHead #'color
}

```

```

\revert Staff.TimeSignature #'color
\revert Stem #'color
}

```

Return colored *measure*.

Color names appear in LilyPond Learning Manual appendix B.5.

measuretools.color_nonbinary_measures_in_expr

abjad.tools.measuretools.color_nonbinary_measures_in_expr.**color_nonbinary_measures_in_expr**

New in version 2.0. Color nonbinary measures in *expr* with *color*:

```

abjad> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
abjad> measuretools.scale_measure_denominator_and_adjust_measure_contents(staff[1], 3)
Measure(3/12, [c'8., d'8.])

```

```

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 3/12
    \scaleDurations #'(2 . 3) {
      c'8.
      d'8.
    }
  }
}

```

```

abjad> measuretools.color_nonbinary_measures_in_expr(staff, 'red')
[Measure(3/12, [c'8., d'8.])]

```

```

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \override Beam #'color = #red
    \override Dots #'color = #red
    \override NoteHead #'color = #red
    \override Staff.TimeSignature #'color = #red
    \override Stem #'color = #red
    \time 3/12
    \scaleDurations #'(2 . 3) {
      c'8.
      d'8.
    }
  }
  \revert Beam #'color
  \revert Dots #'color
  \revert NoteHead #'color
}

```

```

        \revert Staff.TimeSignature #'color
        \revert Stem #'color
    }
}

```

Return list of measures colored.

Color names appear in LilyPond Learning Manual appendix B.5.

measuretools.comment_measures_in_container_with_measure_numbers

`abjad.tools.measuretools.comment_measures_in_container_with_measure_numbers.comment_measures_in_container_with_measure_numbers`

New in version 1.1. Comment measures in *container* with measure numbers:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> measuretools.comment_measures_in_container_with_measure_numbers(staff)

abjad> f(staff)
\new Staff {
    % start measure 1
    {
        \time 2/8
        c'8
        d'8
    }
    % stop measure 1
    % start measure 2
    {
        \time 2/8
        e'8
        f'8
    }
    % stop measure 2
    % start measure 3
    {
        \time 2/8
        g'8
        a'8
    }
    % stop measure 3
}

```

Changed in version 2.0: renamed `label.measure_numbers()` to `measuretools.comment_measures_in_container_with_measure_numbers()`.

measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets_to_measure_contents

`abjad.tools.measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets_to_measure_contents`

New in version 2.0. Extend measures in *expr* with *supplement* and apply full-measure tuplets to contents of measures:

```

abjad> staff = Staff([Measure((2, 8), "c'8 d'8"), Measure((3, 8), "e'8 f'8 g'8")])

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 3/8
    e'8
    f'8
    g'8
  }
}

abjad> supplement = [Rest((1, 16))]
abjad> measuretools.extend_measures_in_expr_and_apply_full_measure_tuplets_to_measure_contents(s

abjad> f(staff)
\new Staff {
  {
    \time 2/8
    \times 4/5 {
      c'8
      d'8
      r16
    }
  }
  {
    \time 3/8
    \fraction \times 6/7 {
      e'8
      f'8
      g'8
      r16
    }
  }
}

```

Return none.

measuretools.fill_measures_in_expr_with_big_endian_notes

```
abjad.tools.measuretools.fill_measures_in_expr_with_big_endian_notes.fill_measures_in_expr
```

New in version 1.1. Fill measures in *expr* with big-endian notes.

measuretools.fill_measures_in_expr_with_full_measure_spacer_skips

```
abjad.tools.measuretools.fill_measures_in_expr_with_full_measure_spacer_skips.fill_measures
```

New in version 1.1. Fill measures in *expr* with full-measure spacer skips.

measuretools.fill_measures_in_expr_with_little_endian_notes

```
abjad.tools.measuretools.fill_measures_in_expr_with_little_endian_notes.fill_measures_in_ex
```

New in version 1.1. Fill measures in *expr* with little-endian notes.

measuretools.fill_measures_in_expr_with_meter_denominator_notes

```
abjad.tools.measuretools.fill_measures_in_expr_with_meter_denominator_notes.fill_measures_
```

New in version 1.1. Fill measures in *expr* with meter denominator notes:

```
abjad> staff = Staff([Measure((3, 4), []), Measure((3, 16), []), Measure((3, 8), [])])
abjad> measuretools.fill_measures_in_expr_with_meter_denominator_notes(staff)
```

```
abjad> f(staff)
\new Staff {
  {
    \time 3/4
    c'4
    c'4
    c'4
  }
  {
    \time 3/16
    c'16
    c'16
    c'16
  }
  {
    \time 3/8
    c'8
    c'8
    c'8
  }
}
```

Delete existing contents of measures in *expr*.

Return none.

measuretools.fill_measures_in_expr_with_repeated_notes

```
abjad.tools.measuretools.fill_measures_in_expr_with_repeated_notes.fill_measures_in_expr_w
```

New in version 1.1. Fill measures in *expr* with repeated notes.

measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts

```
abjad.tools.measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts.fuse_c
```

New in version 1.1. Fuse contiguous measures in *container* cyclically by *counts*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 5)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
    {
        \time 2/8
        b'8
        c''8
    }
    {
        \time 2/8
        d''8
        e''8
    }
}

abjad> counts = (2, 1)
abjad> measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts(staff, counts)

abjad> f(staff)
\new Staff {
    {
        \time 4/8
        c'8
```

```

        d'8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
    {
        \time 4/8
        b'8
        c''8
        d''8
        e''8
    }
}

```

Return none.

Set *mark* to true to mark fused measures for later reference. Changed in version 2.0: renamed `fuse.measures_by_counts_cyclic()` to `measuretools.fuse_contiguous_measures_in_container_cyclically_by_counts()`.

measuretools.fuse_measures

`abjad.tools.measuretools.fuse_measures.fuse_measures(measures)`

New in version 1.1. Fuse *measures*:

```

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (2, 16)]))
abjad> measuretools.fill_measures_in_expr_with_repeated_notes(staff, Duration(1, 16))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> spannertools.BeamSpanner(staff.leaves)
BeamSpanner(c'16, d'16, e'16, f'16)

```

```

abjad> f(staff)
\new Staff {
    {
        \time 1/8
        c'16 [
        d'16
    ]
    {
        \time 2/16
        e'16
        f'16 ]
    }
}

```

```

abjad> measuretools.fuse_measures(staff[:])
Measure(2/8, [c'16, d'16, e'16, f'16])

```

```

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'16 [

```



```

        d'16
        e'16
        f'16 ]
    }
}

```

Return new measure.

Allow parent-contiguous *measures*.

Allow outside-of-score *measures*.

Do not define measure fusion across intervening container boundaries.

Calculate best new time signature.

Instantiate new measure.

Give *measures* contents to new measure.

Give *measures* dominant spanners to new measure.

Give *measures* parentage to new measure.

Leave *measures* empty, unspanned and outside-of-score. Changed in version 2.0: renamed `fuse.measures_by_reference()` to `measuretools.fuse_measures()`.

measuretools.get_first_measure_in_improper_parentage_of_component

`abjad.tools.measuretools.get_first_measure_in_improper_parentage_of_component.get_first_me`

New in version 2.0. Get first measure in improper parentage of *component*:

```

abjad> measure = Measure((2, 4), "c'8 d'8 e'8 f'8")
abjad> staff = Staff([measure])

abjad> f(staff)
\new Staff {
    {
        \time 2/4
        c'8
        d'8
        e'8
        f'8
    }
}

abjad> measuretools.get_first_measure_in_improper_parentage_of_component(staff.leaves[0])
Measure(2/4, [c'8, d'8, e'8, f'8])

```

Return measure or none.

measuretools.get_first_measure_in_proper_parentage_of_component

`abjad.tools.measuretools.get_first_measure_in_proper_parentage_of_component.get_first_meas`

New in version 2.0. Get first measure in proper parentage of *component*:

```

abjad> measure = Measure((2, 4), "c'8 d'8 e'8 f'8")
abjad> staff = Staff([measure])

```

```

abjad> f(staff)
\new Staff {
    {
        \time 2/4
        c'8
        d'8
        e'8
        f'8
    }
}

abjad> measuretools.get_first_measure_in_proper_parentage_of_component(staff.leaves[0])
Measure(2/4, [c'8, d'8, e'8, f'8])

```

Return measure or none.

measuretools.get_next_measure_from_component

`abjad.tools.measuretools.get_next_measure_from_component.get_next_measure_from_component(component)`
 New in version 1.1. Get next measure from *component*.

When *component* is voice, staff or other sequential context, and when *component* contains a measure, return first measure in *component*. This starts the process of forwards measure iteration.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_next_measure_from_component(staff)
Measure(2/8, [c'8, d'8])

```

When *component* is voice, staff or other sequential context, and when *component* contains no measure, raise missing measure error.

When *component* is a measure and there is a measure immediately following *component*, return measure immediately following component.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff[0]) is None
True

```

When *component* is a measure and there is no measure immediately following *component*, return None.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff[-1])
Measure(2/8, [c'8, d'8])

```

When *component* is a leaf and there is a measure in the parentage of *component*, return the measure in the parentage of *component*.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff.leaves[0])
Measure(2/8, [c'8, d'8])

```

When *component* is a leaf and there is no measure in the parentage of *component*, raise missing measure error. Changed in version 2.0: renamed `iterate.measure_next()` to `measuretools.get_next_measure_from_component()`.

measuretools.get_nth_measure_in_expr

`abjad.tools.measuretools.get_nth_measure_in_expr.get_nth_measure_in_expr(expr, n=0)`

New in version 2.0. Get *n*th measure in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}
```

Read forward for positive values of *n*.

```
abjad> for n in range(3):
...     measuretools.get_nth_measure_in_expr(staff, n)
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])
```

Read backward for negative values of *n*.

```
abjad> for n in range(3, -1, -1):
...     measuretools.get_nth_measure_in_expr(staff, n)
...
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])
```

Changed in version 2.0: renamed `iterate.get_nth_measure()` to `measuretools.get_nth_measure_in_expr()`.

measuretools.get_one_indexed_measure_number_in_expr

`abjad.tools.measuretools.get_one_indexed_measure_number_in_expr.get_one_indexed_measure_number_in_expr(expr)`

New in version 2.0. Get one-indexed *measure_number* in *expr*:

```
abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
```

```

abjad> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
  }
  {
    \time 2/8
    g'8
    a'8
  }
}
abjad> measuretools.get_one_indexed_measure_number_in_expr(t, 3)
Measure(2/8, [g'8, a'8])

```

Note that measures number from 1.

measuretools.get_prev_measure_from_component

`abjad.tools.measuretools.get_prev_measure_from_component.get_prev_measure_from_component(component)`
 New in version 1.1. Get previous measure from *component*.

When *component* is voice, staff or other sequential context, and when *component* contains a measure, return last measure in *component*. This starts the process of backwards measure iteration.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff)
Measure(2/8, [e'8, f'8])

```

When *component* is voice, staff or other sequential context, and when *component* contains no measure, raise missing measure error.

When *component* is a measure and there is a measure immediately preceeding *component*, return measure immediately preceeding component.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff[-1])
Measure(2/8, [c'8, d'8])

```

When *component* is a measure and there is no measure immediately preceeding *component*, return `None`.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff[0]) is None
True

```

When *component* is a leaf and there is a measure in the parentage of *component*, return the measure in the parentage of *component*.

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> pitchtools.set_ascending_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> measuretools.get_prev_measure_from_component(staff.leaves[0])
Measure(2/8, [c'8, d'8])

```

When *component* is a leaf and there is no measure in the parentage of *component*, raise missing measure error. Changed in version 2.0: renamed `iterate.measure_prev()` to `measuretools.get_prev_measure_from_component()`.

measuretools.iterate_measures_backward_in_expr

```
abjad.tools.measuretools.iterate_measures_backward_in_expr.iterate_measures_backward_in_expr
```

New in version 2.0. Iterate measures backward in *expr*:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> for measure in measuretools.iterate_measures_backward_in_expr(staff):
...     measure
...
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])

```

Use the optional *start* and *stop* keyword parameters to control indices of iteration.

```

abjad> for measure in measuretools.iterate_measures_backward_in_expr(staff, start = 1):
...     measure
...
Measure(2/8, [e'8, f'8])
Measure(2/8, [c'8, d'8])

abjad> for measure in measuretools.iterate_measures_backward_in_expr(staff, start = 0, stop = 2):
...     measure
...

```

```
Measure(2/8, [g'8, a'8])
Measure(2/8, [e'8, f'8])
```

Changed in version 2.0: renamed `iterate.measures_backward_in()` to `measuretools.iterate_measures_backward_in_expr()`.

`measuretools.iterate_measures_forward_in_expr`

`abjad.tools.measuretools.iterate_measures_forward_in_expr.iterate_measures_forward_in_expr`

New in version 2.0. Iterate measures forward in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

abjad> for measure in measuretools.iterate_measures_forward_in_expr(staff):
...     measure
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])
```

Use the optional *start* and *stop* keyword parameters to control the start and stop indices of iteration.

```
abjad> for measure in measuretools.iterate_measures_forward_in_expr(staff, start = 1):
...     measure
...
Measure(2/8, [e'8, f'8])
Measure(2/8, [g'8, a'8])

abjad> for measure in measuretools.iterate_measures_forward_in_expr(staff, start = 0, stop = 2):
...     measure
...
Measure(2/8, [c'8, d'8])
Measure(2/8, [e'8, f'8])
```

Changed in version 2.0: renamed `iterate.measures_forward_in()` to `measuretools.iterate_measures_forward_in_expr()`.

measuretools.list_time_signatures_of_measures_in_expr

`abjad.tools.measuretools.list_time_signatures_of_measures_in_expr.list_time_signatures_of_r`

New in version 2.0. List time signatures of measures in *expr*:

```
abjad> from abjad.tools import timesignaturetools

abjad> staff = Staff([Measure((2, 8), "c8 d8"), Measure((3, 8), "c8 d8 e8"), Measure((4, 8), "c8

abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c8
        d8
    }
    {
        \time 3/8
        c8
        d8
        e8
    }
    {
        \time 4/8
        c8
        d8
        e8
        f8
    }
}

abjad> measuretools.list_time_signatures_of_measures_in_expr(staff)
[TimeSignatureMark((2, 8))(|2/8, c8, d8|), TimeSignatureMark((3, 8))(|3/8, c8, d8, e8|), TimeSig
```

Return list of zero or more time signatures. Changed in version 2.0: re-named `measuretools.list_time_signatures_of_mesures_in_expr()` to `measuretools.list_time_signatures_of_measures_in_expr()`.

measuretools.make_measures_with_full_measure_spacer_skips

`abjad.tools.measuretools.make_measures_with_full_measure_spacer_skips.make_measures_with_fr`

New in version 1.1. Make measures with full-measure spacer skips from *meters*:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5

abjad> staff = Staff(measures)

abjad> f(staff)
\new Staff {
    {
        \time 1/8
        s1 * 1/8
    }
    {
        \time 5/16
        s1 * 5/16
    }
}
```

```
{
    \time 5/16
    s1 * 5/16
}
```

Return list of rigid measures. Changed in version 2.0: renamed `measuretools.make()` to `measuretools.make_measures_with_full_measure_spacer_skips()`.

measuretools.measure_to_one_line_input_string

`abjad.tools.measuretools.measure_to_one_line_input_string.measure_to_one_line_input_string`

New in version 2.6. Change *measure* to one-line input string:

```
abjad> measure = Measure((4, 4), "c'4 d'4 e'4 f'4")

abjad> input_string = measuretools.measure_to_one_line_input_string(measure)

abjad> print input_string
Measure((4, 4), "c'4 d'4 e'4 f'4")

abjad> new_measure = eval(input_string)

abjad> new_measure
Measure(4/4, [c'4, d'4, e'4, f'4])

abjad> f(new_measure)
{
    \time 4/4
    c'4
    d'4
    e'4
    f'4
}
```

The purpose of this function is to create an evaluable string from simple measures.

Spanners, articulations and overrides not supported.

Return string.

measuretools.move_measure_prolation_to_full_measure_tuplet

`abjad.tools.measuretools.move_measure_prolation_to_full_measure_tuplet.move_measure_prolat`

New in version 2.0. Move measure prolotion to full-measure tuplet.

Turn nonbinary measures into binary measures containing a single fixed-duration tuplet.

This is the inverse of `measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure()`.

Note that not all nonbinary measures can be made binary.

Returns `None` because processes potentially many measures. Changed in version 2.0: renamed `measuretools.project()` to `measuretools.move_measure_prolation_to_full_measure_tuplet()`.

measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure

abjad.tools.measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure.**move_pro**

New in version 1.1. Move prolotion of full-measure tuplet to meter of measure.

Measures usually become nonbinary as as result:

```
abjad> t = Measure((2, 8), [tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")])
abjad> measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure(t)
```

```
abjad> f(t)
{
    \time 3/12
    \scaleDurations #'(2 . 3) {
        c'8
        d'8
        e'8
    }
}
```

Return `none`. Changed in version 2.0: renamed `measuretools.subsume()` to `measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure()`.

measuretools.multiply_contents_of_measures_in_expr

abjad.tools.measuretools.multiply_contents_of_measures_in_expr.**multiply_contents_of_measures**

New in version 1.1. Multiply contents $n - 1$ times and adjust meter of every measure in *expr*:

```
abjad> measure = Measure((3, 8), "c'8 d'8 e'8")
abjad> spannertools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8)
```

```
abjad> f(measure)
{
    \time 3/8
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> measuretools.multiply_contents_of_measures_in_expr(measure, 3)
```

```
abjad> f(measure)
{
    \time 9/8
    c'8 [
    d'8
    e'8 ]
    c'8 [
    d'8
    e'8 ]
    c'8 [
    d'8
    e'8 ]
}
```

Changed in version 2.0: renamed `measuretools.spin()` to `measuretools.multiply_contents_of_measures_in_expr()`.

measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators

abjad.tools.measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators

New in version 1.1. Multiply contents of measures in *expr* and scale meter denominators.

Expr may be any Abjad expression. Concentration_pairs a Python list of pairs, each of the form (spin_count, scalar_denominator). Both spin_count and scalar_denominator must be positive integers.

Iterate expr. For every measure in expr, spin measure by the spin_count element in concentration_pair and scale measure by 1/scalar_denominator element in concentration_pair.

Return Python list of transformed measures:

```
abjad> t = Measure((3, 16), notetools.make_repeated_notes(3, Duration(1, 16)))
abjad> print(measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators(t,
|9/48, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32|
```

```
abjad> t = Measure((3, 16), notetools.make_repeated_notes(3, Duration(1, 16)))
abjad> print(measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators(t,
|9/32, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32, c'32|
```

```
abjad> t = Measure((3, 16), notetools.make_repeated_notes(3, Duration(1, 16)))
abjad> print(measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators(t,
|9/16, c'16, c'16, c'16, c'16, c'16, c'16, c'16, c'16, c'16|
```

Changed in version 2.0: renamed measuretools.concentrate() to measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators().

measuretools.pad_measures_in_expr_with_rests

abjad.tools.measuretools.pad_measures_in_expr_with_rests.**pad_measures_in_expr_with_rests** (ex
fr
ba
sp

New in version 1.1. Pad measures in *expr* with rests.

Iterate all measures in *expr*. Insert rest with duration equal to *front* at beginning of each measure. Insert rest with duration equal to *back* at end of each measure.

Set *front* to a positive rational or none. Set *back* to a positive rational or none.

Note that this function is designed to help create regularly spaced charts and tables of musical materials. This function makes most sense when used on anonymous measures or dynamic measures.

```
abjad> t = Staff(measuretools.AnonymousMeasure("c'8 d'8") * 2)
abjad> front, back = Duration(1, 32), Duration(1, 64)
abjad> measuretools.pad_measures_in_expr_with_rests(t, front, back)
```

```
abjad> f(t)
\new Staff {
    {
        \override Staff.TimeSignature #'stencil = ##f
        \time 19/64
        r32
```

```

        c'8
        d'8
        r64
        \revert Staff.TimeSignature #'stencil
    }
    {
        \override Staff.TimeSignature #'stencil = ##f
        \time 19/64
        r32
        c'8
        d'8
        r64
        \revert Staff.TimeSignature #'stencil
    }
}

```

Works when measures contain stacked voices:

```

abjad> measure = measuretools.DynamicMeasure(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> measure.is_parallel = True
abjad> t = Staff(measure * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> measuretools.pad_measures_in_expr_with_rests(t, Duration(1, 32), Duration(1, 64))

abjad> f(t)
\new Staff {
  <<
    \time 19/64
    \new Voice {
      r32
      c'8
      d'8
      r64
    }
    \new Voice {
      r32
      e'8
      f'8
      r64
    }
  >>
  <<
    \time 19/64
    \new Voice {
      r32
      g'8
      a'8
      r64
    }
    \new Voice {
      r32
      b'8
      c''8
      r64
    }
  >>
}

```

Set the optional *splice* keyword to `True` to extend edge spanners over newly inserted rests:

```

abjad> t = measuretools.DynamicMeasure("c'8 d'8")
abjad> spannertools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8)
abjad> measuretools.pad_measures_in_expr_with_rests(t, Duration(1, 32), Duration(1, 64), splice

abjad> f(t)
{
    \time 19/64
    r32 [
    c'8
    d'8
    r64 ]
}

```

Return none.

Raise value when *front* is neither a positive rational nor none.

Raise value when *back* is neither a positive rational nor none. Changed in version 2.0: renamed `layout.insert_measure_padding_rest()` to `measuretools.pad_measures_in_expr_with_rests()`.

measuretools.pad_measures_in_expr_with_skips

abjad.tools.measuretools.pad_measures_in_expr_with_skips.**pad_measures_in_expr_with_skips** (*expr*, *front*, *back*, *splice*)

New in version 2.0. Pad measures in *expr* with skips.

Iterate all measures in *expr*. Insert skip with duration equal to *front* at beginning of each measure. Insert skip with duration equal to *back* at end of each measure.

Set *front* to a positive rational or none. Set *back* to a positive rational or none.

Note that this function is designed to help create regularly spaced charts and tables of musical materials. This function makes most sense when used on anonymous measures and dynamic measures.

```

abjad> t = Staff(measuretools.AnonymousMeasure("c'8 d'8") * 2)
abjad> front, back = Duration(1, 32), Duration(1, 64)
abjad> measuretools.pad_measures_in_expr_with_skips(t, front, back)

```

```

abjad> f(t)
\new Staff {
    {
        \override Staff.TimeSignature #'stencil = ##f
        \time 19/64
        s32
        c'8
        d'8
        s64
        \revert Staff.TimeSignature #'stencil
    }
    {
        \override Staff.TimeSignature #'stencil = ##f
        \time 19/64
        s32
        c'8
    }
}

```

```

        d'8
        s64
        \revert Staff.TimeSignature #'stencil
    }
}

```

Works when measures contain stacked voices.

```

abjad> measure = measuretools.DynamicMeasure(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> measure.is_parallel = True
abjad> t = Staff(measure * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> measuretools.pad_measures_in_expr_with_skips(t, Duration(1, 32), Duration(1, 64))

abjad> f(t)
\new Staff {
  <<
    \time 19/64
    \new Voice {
      s32
      c'8
      d'8
      s64
    }
    \new Voice {
      s32
      e'8
      f'8
      s64
    }
  >>
  <<
    \time 19/64
    \new Voice {
      s32
      g'8
      a'8
      s64
    }
    \new Voice {
      s32
      b'8
      c''8
      s64
    }
  >>
}

```

Set the optional *splice* keyword to True to extend edge spanners over newly inserted skips:

```

abjad> t = measuretools.DynamicMeasure("c'8 d'8")
abjad> spannertools.BeamSpanner(t[:])
BeamSpanner(c'8, d'8)
abjad> measuretools.pad_measures_in_expr_with_skips(t, Duration(1, 32), Duration(1, 64), splice

abjad> f(t)
{
  \time 19/64
  s32 [

```

```

        c' 8
        d' 8
        s64 ]
    }

```

Return none.

Raise value error when *front* is neither a positive rational nor none.

Raise value error when *back* is neither a positive rational nor none. Changed in version 2.0: renamed `layout.insert_measure_padding_skip()` to `measuretools.pad_measures_in_expr_with_skips()`.

measuretools.pitch_array_row_to_measure

`abjad.tools.measuretools.pitch_array_row_to_measure.pitch_array_row_to_measure` (*pitch_array_row*, *cell_duration_denominator*)

New in version 2.0. Change *pitch_array_row* to measure with meter *pitch_array_row.width* over *cell_duration_denominator*:

```

abjad> from abjad.tools import pitcharraytools
abjad> array = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

abjad> print array
[ ] [d'] [bf bqf  ]
[g'      ] [fs'    ] [ ]

abjad> measure = measuretools.pitch_array_row_to_measure(array.rows[0])

abjad> f(measure)
{
    \time 4/8
    r8
    d' 8
    <bf bqf>4
}

```

Return measure.

measuretools.pitch_array_to_measures

`abjad.tools.measuretools.pitch_array_to_measures.pitch_array_to_measures` (*pitch_array*, *cell_duration_denominator*)

New in version 2.0. Change *pitch_array* to measures with meters *row.width* over *cell_duration_denominator* for each row in *pitch_array*:

```

abjad> from abjad.tools import pitcharraytools
abjad> array = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

abjad> print array
[ ] [d'] [bf bqf  ]
[g'      ] [fs'    ] [ ]

```

```

abjad> measuretools.pitch_array_to_measures(array)
[Measure(4/8, [r8, d'8, <bf bqf>4]), Measure(4/8, [g'4, fs'8, r8])]
abjad> for measure in _:
...     f(measure)
...
{
    \time 4/8
    r8
    d'8
    <bf bqf>4
}
{
    \time 4/8
    g'4
    fs'8
    r8
}

```

Return list of measures.

measuretools.replace_contents_of_measures_in_expr

abjad.tools.measuretools.replace_contents_of_measures_in_expr.**replace_contents_of_measures**

New in version 1.1. Replace contents of measures in *expr* with *new_contents*:

```

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (3, 16)]))

abjad> f(staff)
\new Staff {
    {
        \time 1/8
        s1 * 1/8
    }
    {
        \time 3/16
        s1 * 3/16
    }
}

abjad> notes = [Note("c'16"), Note("d'16"), Note("e'16"), Note("f'16")]
abjad> measuretools.replace_contents_of_measures_in_expr(staff, notes)
[Measure(1/8, [c'16, d'16]), Measure(3/16, [e'16, f'16, s1 * 1/16])]

abjad> f(staff)
\new Staff {
    {
        \time 1/8
        c'16
        d'16
    }
    {
        \time 3/16
        e'16
        f'16
        s1 * 1/16
    }
}

```

```
}
}
```

Preserve duration of all measures.

Skip measures that are too small.

Pad extra space at end of measures with spacer skip.

If not enough measures raise stop iteration.

Return measures iterated. Changed in version 2.0: renamed `measuretools.override_contents()` to `measuretools.replace_contents_of_measures_in_expr()`.

`measuretools.report_meter_distribution_as_string`

`abjad.tools.measuretools.report_meter_distribution_as_string`.**`report_meter_distribution_as_string`**

New in version 2.0. Report meter distribution of *expr* as string:

```
abjad> measuretools.report_meter_distribution_as_string(t) # doctest: +SKIP
'\t3/80\t2\n\t2/16\t73\n\t7/40\t1\n\t3/16\t20\n\t16/80\t1\n\t17/80\t1\n\t19/80\t1\n\t4/16\t73\n\t5/16\t62\n\t13/40\t1\n\t27/80\t1\n\t6/16\t12\n\t7/16\t16\n\t8/16\t13\n\t9/16\t15\n\t10/16\t4\n'
```

Return string.

`measuretools.scale_contents_of_measures_in_expr`

`abjad.tools.measuretools.scale_contents_of_measures_in_expr`.**`scale_contents_of_measures_in_expr`**

New in version 2.0. Scale contents of measures in *expr* by *multiplier*.

Iterate *expr*. For every measure in *expr* first multiply the measure meter by *multiplier* and then scale measure contents to fit the new meter.

Extend `containertools.scale_contents_of_container()`.

Return none.

`measuretools.scale_measure_by_multiplier_and_adjust_meter`

`abjad.tools.measuretools.scale_measure_by_multiplier_and_adjust_meter`.**`scale_measure_by_multiplier_and_adjust_meter`**

New in version 2.0. Scale *measure* by *multiplier* and adjust meter:

```
abjad> t = Measure((3, 8), "c'8 d'8 e'8")
abjad> measuretools.scale_measure_by_multiplier_and_adjust_meter(t, Duration(2, 3))
Measure(3/12, [c'8, d'8, e'8])
```

```
abjad> f(t)
{
\time 3/12
\scaleDurations #'(2 . 3) {
```



```

        c'8
        d'8
        e'8
    }
}

```

Return *measure*.

measuretools.scale_measure_denominator_and_adjust_measure_contents

abjad.tools.measuretools.scale_measure_denominator_and_adjust_measure_contents.**scale_measure**

New in version 1.1. Change binary *measure* to nonbinary measure with *new_denominator_factor*:

```

abjad> measure = Measure((2, 8), "c'8 d'8")
abjad> spannertools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8)

abjad> f(measure)
{
    \time 2/8
    c'8 [
    d'8 ]
}

abjad> measuretools.scale_measure_denominator_and_adjust_measure_contents(measure, 3)
Measure(3/12, [c'8., d'8.])

abjad> f(measure)
{
    \time 3/12
    \scaleDurations #'(2 . 3) {
        c'8. [
        d'8. ]
    }
}

```

Treat *new_denominator_factor* like clever form of 1: 3/3 or 5/5 or 7/7, etc.

Preserve *measure* prolated duration.

Derive new *measure* multiplier.

Scale *measure* contents.

Pick best new meter. Changed in version 2.0: renamed `measuretools.change_binary_measure_to_nonbinary()` to `measuretools.scale_measure_denominator_and_adjust_measure_contents()`.

measuretools.set_measure_denominator_and_adjust_numerator

abjad.tools.measuretools.set_measure_denominator_and_adjust_numerator.**set_measure_denominator**

New in version 1.1. Set *measure* meter *denominator* and multiply meter numerator accordingly:

```
abjad> measure = Measure((3, 8), "c'8 d'8 e'8")
abjad> spannertools.BeamSpanner(measure.leaves)
BeamSpanner(c'8, d'8, e'8)

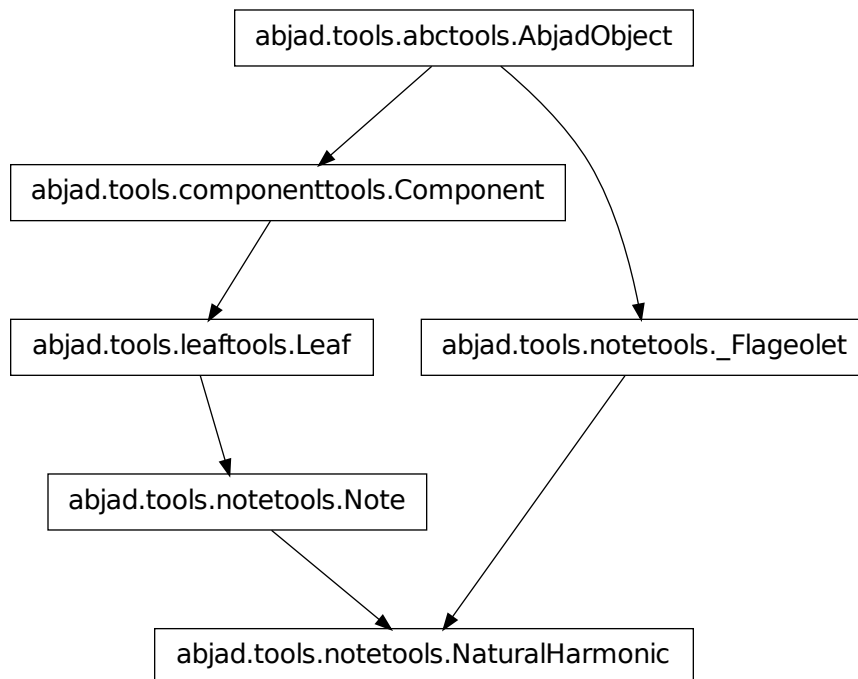
abjad> f(measure)
{
    \time 3/8
    c'8 [
    d'8
    e'8 ]
}

abjad> measuretools.set_measure_denominator_and_adjust_numerator(measure, 16)
Measure(6/16, [c'8, d'8, e'8])

abjad> f(measure)
{
    \time 6/16
    c'8 [
    d'8
    e'8 ]
}
```

Leave *measure* contents unchanged.

Return *measure*. Changed in version 2.0: renamed `measuretools.set_measure_denominator_and_multiply_numerator` to `measuretools.set_measure_denominator_and_adjust_numerator()`.

notetools**notetools concrete classes****notetools.NaturalHarmonic**

class `abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic`.**NaturalHarmonic**(*args)
 Abjad model of natural harmonic.

Initialize natural harmonic by hand:

```
abjad> notetools.NaturalHarmonic("cs'8.")
NaturalHarmonic(cs', 8.)
```

Initialize natural harmonic from note:

```
abjad> note = Note("cs'8.")

abjad> notetools.NaturalHarmonic(note)
NaturalHarmonic(cs', 8.)
```

Natural harmonics are immutable.

Read-only Properties

`NaturalHarmonic.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

NaturalHarmonic.fingered_pitch

Read-only fingered pitch of note:

```
abjad> staff = Staff("d''8 e''8 f''8 g''8")
abjad> piccolo = instrumenttools.Piccolo()(staff)
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch(
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
```

```
abjad> staff[0].fingered_pitch
NamedChromaticPitch("d' ")
```

Return named chromatic pitch.

Note: Inherited from `notetools.Note`

NaturalHarmonic.format

Note: Inherited from `leaftools.Leaf`

NaturalHarmonic.leaf_index

Note: Inherited from `leaftools.Leaf`

NaturalHarmonic.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

NaturalHarmonic.multiplied_duration

Note: Inherited from `leaftools.Leaf`

NaturalHarmonic.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

NaturalHarmonic.**parent**

Note: Inherited from `componenttools.Component`

NaturalHarmonic.**preprolated_duration**

Note: Inherited from `leaftools.Leaf`

NaturalHarmonic.**prolated_duration**

Note: Inherited from `componenttools.Component`

NaturalHarmonic.**prolation**

Note: Inherited from `componenttools.Component`

NaturalHarmonic.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

NaturalHarmonic.**sounding_pitch**

Read-only sounding pitch of note:

```
abjad> staff = Staff("d''8 e''8 f''8 g''8")
abjad> piccolo = instrumenttools.Piccolo()(staff)
```

```
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch()
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
abjad> staff[0].sounding_pitch
NamedChromaticPitch("d'")
```

Return named chromatic pitch.

Note: Inherited from `notetools.Note`

NaturalHarmonic.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

`NaturalHarmonic.suono_reale`

Actual sound of the harmonic when played.

Note: Inherited from `notetools._Flageolet`

Read/write Properties

`NaturalHarmonic.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.note_head`

Get note head of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.note_head
NoteHead("cs'")
```

Set note head of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.note_head = 14
abjad> note
Note("d'8.")
```

Note: Inherited from `notetools.Note`

`NaturalHarmonic.written_duration`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.written_pitch`

Get named pitch of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.written_pitch
NamedChromaticPitch("cs'")
```

Set named pitch of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.written_pitch = 14
abjad> note
Note("d'8.")
```

Note: Inherited from `notetools.Note`

`NaturalHarmonic.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

`NaturalHarmonic.__and__(arg)`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NaturalHarmonic.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`NaturalHarmonic.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__mul__(n)`

Note: Inherited from `componenttools.Component`

`NaturalHarmonic.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NaturalHarmonic.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.__repr__()`

`NaturalHarmonic.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`NaturalHarmonic.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NaturalHarmonic.__str__()`

Note: Inherited from `leaftools.Leaf`

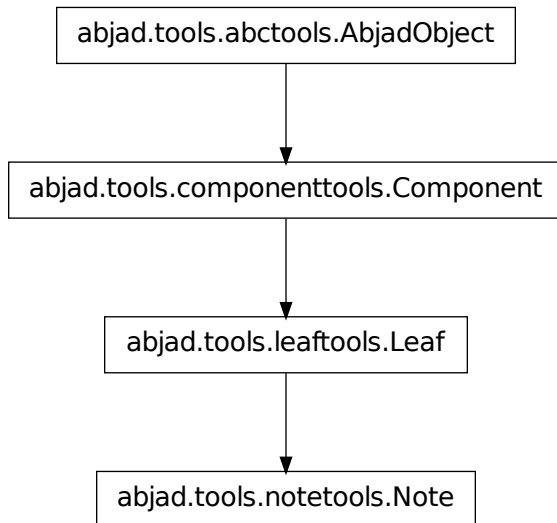
`NaturalHarmonic.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`NaturalHarmonic.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

notetools.Note



class `abjad.tools.notetools.Note.Note.Note(*args, **kwargs)`

New in version 1.0. Abjad model of a note:

```
abjad> Note(13, (3, 16))
Note("cs''8.")
```

Notes are immutable.

Read-only Properties

`Note.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`Note.fingered_pitch`

Read-only fingered pitch of note:

```
abjad> staff = Staff("d''8 e''8 f''8 g''8")
abjad> piccolo = instrumenttools.Piccolo()(staff)
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch(
    abjad> f(staff)
    \new Staff {
        \set Staff.instrumentName = \markup { Piccolo }
```

```
\set Staff.shortInstrumentName = \markup { Picc. }  
d'8  
e'8  
f'8  
g'8  
}
```

```
abjad> staff[0].fingered_pitch  
NamedChromaticPitch("d'")
```

Return named chromatic pitch.

Note.**format**

Note: Inherited from `leaftools.Leaf`

Note.**leaf_index**

Note: Inherited from `leaftools.Leaf`

Note.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Note.**multiplied_duration**

Note: Inherited from `leaftools.Leaf`

Note.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Note.**parent**

Note: Inherited from `componenttools.Component`

Note.**preprolated_duration**

Note: Inherited from `leaftools.Leaf`

Note.**prolated_duration**

Note: Inherited from `componenttools.Component`

Note.`prolation`

Note: Inherited from `componenttools.Component`

Note.`set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Note.`sounding_pitch`

Read-only sounding pitch of note:

```
abjad> staff = Staff("d''8 e''8 f''8 g''8")
abjad> piccolo = instrumenttools.Piccolo()(staff)
```

```
abjad> instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch(
```

```
abjad> f(staff)
\new Staff {
  \set Staff.instrumentName = \markup { Piccolo }
  \set Staff.shortInstrumentName = \markup { Picc. }
  d'8
  e'8
  f'8
  g'8
}
abjad> staff[0].sounding_pitch
NamedChromaticPitch("d'")
```

Return named chromatic pitch.

Note.`spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Note.`duration_multiplier`

Note: Inherited from `leaftools.Leaf`

Note.`note_head`

Get note head of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.note_head
NoteHead("cs'")
```

Set note head of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.note_head = 14
abjad> note
Note("d' '8.")
```

Note.**written_duration**

Note: Inherited from `leaftools.Leaf`

Note.**written_pitch**

Get named pitch of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.written_pitch
NamedChromaticPitch("cs' ")
```

Set named pitch of note:

```
abjad> note = Note(13, (3, 16))
abjad> note.written_pitch = 14
abjad> note
Note("d' '8.")
```

Note.**written_pitch_indication_is_at_sounding_pitch**

Note: Inherited from `leaftools.Leaf`

Note.**written_pitch_indication_is_nonsemantic**

Note: Inherited from `leaftools.Leaf`

Special Methods

Note.**__and__**(*arg*)

Note: Inherited from `leaftools.Leaf`

Note.**__delattr__**()

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Note.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Note. `__ge__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Note. `__gt__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

Note. `__hash__` () $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

Note. `__le__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Note. `__lt__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Note. `__mul__` (*n*)

Note: Inherited from `componenttools.Component`

Note. `__ne__` (*arg*)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Note. `__or__` (*arg*)

Note: Inherited from `leaftools.Leaf`

Note.**__repr__**()

Note: Inherited from `leaftools.Leaf`

Note.**__rmul__**(*n*)

Note: Inherited from `componenttools.Component`

Note.**__setattr__**()
x.**__setattr__**('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

Note.**__str__**()

Note: Inherited from `leaftools.Leaf`

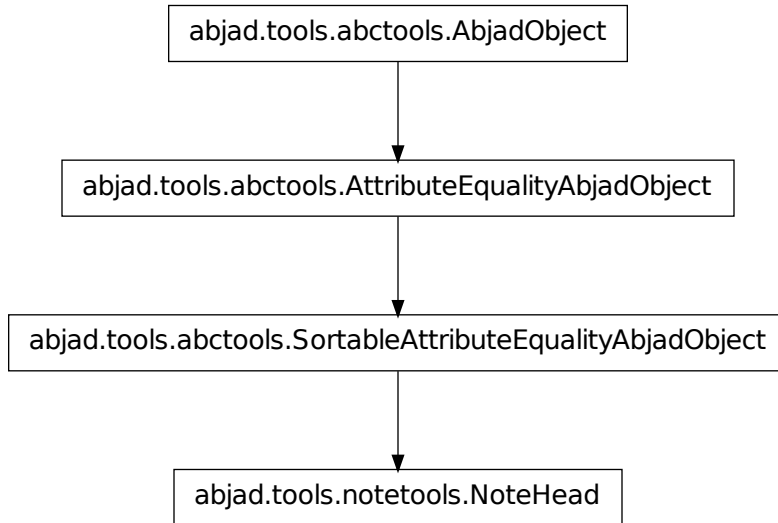
Note.**__sub__**(*arg*)

Note: Inherited from `leaftools.Leaf`

Note.**__xor__**(*arg*)

Note: Inherited from `leaftools.Leaf`

notetools.NoteHead



class `abjad.tools.notetools.NoteHead.NoteHead(*args)`

Abjad model of a note head:

```
abjad> notetools.NoteHead(13)
NoteHead("cs' ' ")
```

Note heads are immutable.

Read-only Properties

`NoteHead.format`

Read-only LilyPond input format of note head:

```
abjad> note_head = notetools.NoteHead("cs' ' ")
abjad> note_head.format
"cs' ' "
```

Return string.

`NoteHead.named_chromatic_pitch`

Read-only named chromatic pitch equal to note head:

```
abjad> note_head = notetools.NoteHead("cs' ' ")
abjad> note_head.named_chromatic_pitch
NamedChromaticPitch("cs' ' ")
```

Return named chromatic pitch.

`NoteHead.tweak`

Read-only LilyPond tweak reservoir:

```
abjad> note_head = notetools.NoteHead("cs' ' ")
abjad> note_head.tweak
LilyPondTweakReservoir()
```

Return LilyPond tweak reservoir.

Read/write Properties

NoteHead.**written_pitch**

Get named pitch of note head:

```
abjad> note_head = notetools.NoteHead("cs' ' ")
abjad> note_head.written_pitch
NamedChromaticPitch("cs' ' ")
```

Set named pitch of note head:

```
abjad> note_head = notetools.NoteHead("cs' ' ")
abjad> note_head.written_pitch = "d' ' "
abjad> note_head.written_pitch
NamedChromaticPitch("d' ' ")
```

Set pitch token.

Special Methods

NoteHead.**__delattr__**()

x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

NoteHead.**__eq__**(arg)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

NoteHead.**__ge__**(arg)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

NoteHead.**__gt__**(arg)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

NoteHead.**__hash__**() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

`NoteHead.__le__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NoteHead.__lt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NoteHead.__ne__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NoteHead.__repr__()`
`NoteHead.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NoteHead.__str__()`

notetools functions

notetools.add_artificial_harmonic_to_note

`abjad.tools.notetools.add_artificial_harmonic_to_note.add_artificial_harmonic_to_note(note, melodic_diatonic_interval)`

Add artifical harmonic to *note* at *melodic_diatonic_interval*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

```
abjad> notetools.add_artificial_harmonic_to_note(staff[0])
Chord("<c' f'>8")

abjad> f(staff)
\new Staff {
  <
    c'
    \tweak #'style #'harmonic
    f'
  >8 [
    d' 8
    e' 8
    f' 8 ]
}
```

Create new artificial harmonic chord from *note*.

Move parentage and spanners from *note* to artificial harmonic chord.

Return artificial harmonic chord. Changed in version 2.0: renamed `harmonictools.add_artificial()` to `notetools.add_artificial_harmonic_to_note()`.

notetools.all_are_notes

`abjad.tools.notetools.all_are_notes.all_are_notes(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad notes:

```
abjad> notes = [Note("c' 4"), Note("d' 4"), Note("e' 4")]
```

```
abjad> notetools.all_are_notes(notes)
True
```

True when *expr* is an empty sequence:

```
abjad> notetools.all_are_notes([])
True
```

Otherwise false:

```
abjad> notetools.all_are_notes('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map

`abjad.tools.notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map.color_note_head_by_numbered_chromatic_pitch_class_color_map(note)`

Color *pitch_carrier* note head:

```
abjad> note = Note("c' 4")
```

```
abjad> notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map(note)
Note("c' 4")
```

```
abjad> f(note)
\once \override NoteHead #'color = #(x11-color 'red)
c'4
```

Numbered chromatic pitch-class color map:

```
0: red
1: MediumBlue
2: orange
3: LightSlateBlue
4: ForestGreen
5: MediumOrchid
6: firebrick
7: DeepPink
8: DarkOrange
9: IndianRed
10: CadetBlue
11: SeaGreen
12: LimeGreen
```

Numbered chromatic pitch-class color map can not be changed.

Raise type error when *pitch_carrier* is not a pitch carrier.

Raise extra pitch error when *pitch_carrier* carries more than 1 note head.

Raise missing pitch error when *pitch_carrier* carries no note head.

Return *pitch_carrier*. Changed in version 2.0: renamed `pitchtools.color_by_pc()` to `notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map()`. Changed in version 2.0: renamed `notetools.color_note_head_by_numeric_chromatic_pitch_class_color_map()` to `notetools.color_note_head_by_numbered_chromatic_pitch_class_color_map()`.

notetools.iterate_notes_backward_in_expr

```
abjad.tools.notetools.iterate_notes_backward_in_expr.iterate_notes_backward_in_expr(expr,
                                                                                   start=0,
                                                                                   stop=None)
```

New in version 2.0. Yield right-to-left notes in *expr*:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
  }
  {
    \time 2/8
    g'8
    a'8
  }
}
```

```

    }
}

abjad> for leaf in notetools.iterate_notes_backward_in_expr(staff):
...     leaf
...
Note("a'8")
Note("g'8")
Note("f'8")
Note("e'8")
Note("d'8")
Note("c'8")

```

Use optional *start* and *stop* keyword parameters to control indices of iteration:

```

abjad> for leaf in notetools.iterate_notes_backward_in_expr(staff, start = 3):
...     leaf
...
Note("e'8")
Note("d'8")
Note("c'8")

abjad> for leaf in notetools.iterate_notes_backward_in_expr(staff, start = 0, stop = 3):
...     leaf
...
Note("a'8")
Note("g'8")
Note("f'8")

abjad> for leaf in notetools.iterate_notes_backward_in_expr(staff, start = 2, stop = 4):
...     leaf
...
Note("f'8")
Note("e'8")

```

Return note generator. Changed in version 2.0: renamed `iterate.notes_backward_in()` to `notetools.iterate_notes_backward_in_expr()`.

notetools.iterate_notes_forward_in_expr

```

abjad.tools.notetools.iterate_notes_forward_in_expr.iterate_notes_forward_in_expr(expr,
                                                                                   start=0,
                                                                                   stop=None)

```

New in version 2.0. Yield left-to-right notes in *expr*:

```

abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 3)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
    {
        \time 2/8
        c'8
        d'8
    }
    {
        \time 2/8
        e'8
    }
}

```

```

        f'8
    }
    {
        \time 2/8
        g'8
        a'8
    }
}

```

```

abjad> for leaf in notetools.iterate_notes_forward_in_expr(staff):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")
Note("f'8")
Note("g'8")
Note("a'8")

```

Use optional *start* and *stop* keyword parameters to control start and stop indices of iteration:

```

abjad> for leaf in notetools.iterate_notes_forward_in_expr(staff, start = 3):
...     leaf
...
Note("f'8")
Note("g'8")
Note("a'8")

```

```

abjad> for leaf in notetools.iterate_notes_forward_in_expr(staff, start = 0, stop = 3):
...     leaf
...
Note("c'8")
Note("d'8")
Note("e'8")

```

```

abjad> for leaf in notetools.iterate_notes_forward_in_expr(staff, start = 2, stop = 4):
...     leaf
...
Note("e'8")
Note("f'8")

```

Return generator. Changed in version 2.0: renamed `iterate.notes_forward_in()` to `notetools.iterate_notes_forward_in_expr()`.

notetools.label_notes_in_expr_with_note_indices

`abjad.tools.notetools.label_notes_in_expr_with_note_indices.label_notes_in_expr_with_note_`

New in version 2.0. Label notes in *expr* with note indices:

```

abjad> staff = Staff("c'8 d'8 r8 r8 g'8 a'8 r8 c''8")

abjad> notetools.label_notes_in_expr_with_note_indices(staff)

abjad> f(staff)
\new Staff {
    c'8 _ \markup { \small 0 }

```

```

d'8 _ \markup { \small 1 }
r8
r8
g'8 _ \markup { \small 2 }
a'8 _ \markup { \small 3 }
r8
c''8 _ \markup { \small 4 }
}

```

Return none.

notetools.make_accelerating_notes_with_lilypond_multipliers

abjad.tools.notetools.make_accelerating_notes_with_lilypond_multipliers.make_accelerating_

Make accelerating notes with LilyPond multipliers:

```

abjad> notetools.make_accelerating_notes_with_lilypond_multipliers([1,2], (1, 2), (1, 4), (1, 8))
[Note("cs'8 * 113/64"), Note("d'8 * 169/128"), Note("cs'8 * 117/128")]

abjad> voice = Voice(_)
abjad> voice.prolated_duration
Duration(1, 2)

```

Set note pitches cyclically from *pitches*.

Return as many interpolation values as necessary to fill the *total* duration requested.

Interpolate durations from *start* to *stop*.

Set note durations to *written* duration times computed interpolated multipliers.

Return list of notes. Changed in version 2.0: renamed `construct.notes_curve()` to `notetools.make_accelerating_notes_with_lilypond_multipliers()`.

notetools.make_notes

abjad.tools.notetools.make_notes.make_notes(*pitches*, *durations*, *direction*='big-endian')

Make notes according to *pitches* and *durations*.

Cycle through *pitches* when the length of *pitches* is less than the length of *durations*:

```

abjad> notetools.make_notes([0], [(1, 16), (1, 8), (1, 8)])
[Note("c'16"), Note("c'8"), Note("c'8")]

```

Cycle through *durations* when the length of *durations* is less than the length of *pitches*:

```

abjad> notetools.make_notes([0, 2, 4, 5, 7], [(1, 16), (1, 8), (1, 8)])
[Note("c'16"), Note("d'8"), Note("e'8"), Note("f'16"), Note("g'8")]

```

Create ad hoc tuplets for nonassignable durations:

```
abjad> notetools.make_notes([0], [(1, 16), (1, 12), (1, 8)])
[Note("c'16"), Tuplet(2/3, [c'8]), Note("c'8")]
```

Set *direction* to 'big-endian' to express tied values in decreasing duration:

```
abjad> notetools.make_notes([0], [(13, 16)], direction = 'big-endian')
[Note("c'2."), Note("c'16")]
```

Set *direction* to 'little-endian' to express tied values in increasing duration:

```
abjad> notetools.make_notes([0], [(13, 16)], direction = 'little-endian')
[Note("c'16"), Note("c'2.")]
```

Set *pitch*s to a single pitch or a sequence of pitches.

Set *durations* to a single duration or a list of durations.

Return list of newly constructed notes. Changed in version 2.0: renamed `construct.notes()` to `notetools.make_notes()`.

notetools.make_notes_with_multiplied_durations

```
abjad.tools.notetools.make_notes_with_multiplied_durations.make_notes_with_multiplied_durations
```

New in version 2.0. Make *written_duration* notes with *pitch* and *multiplied_durations*:

```
abjad> notetools.make_notes_with_multiplied_durations(0, Duration(1, 4), [(1, 2), (1, 3), (1, 4)])
[Note("c'4 * 2"), Note("c'4 * 4/3"), Note("c'4 * 1"), Note("c'4 * 4/5")]
```

Useful for making spatially positioned notes.

Return list of notes.

notetools.make_percussion_note

```
abjad.tools.notetools.make_percussion_note.make_percussion_note(pitch, total_duration,
max_note_duration=(1, 8))
```

Make percussion note:

```
abjad> notetools.make_percussion_note(2, (1, 4), (1, 8))
[Note("d'8"), Rest('r8')]
```

```
abjad> notetools.make_percussion_note(2, (1, 64), (1, 8))
[Note("d'64")]
```

```
abjad> notetools.make_percussion_note(2, (5, 64), (1, 8))
[Note("d'16"), Rest('r64')]
```

```
abjad> notetools.make_percussion_note(2, (5, 4), (1, 8))
[Note("d'8"), Rest('r1'), Rest('r8')]
```

Return list of newly constructed note followed by zero or more newly constructed rests.

Durations of note and rests returned will sum to *total_duration*.

Duration of note returned will be no greater than *max_note_duration*.

Duration of rests returned will sum to note duration taken from *total_duration*.

Useful for percussion music where attack duration is negligible and tied notes undesirable. Changed in version 2.0: renamed `construct.percussion_note()` to `notetools.make_percussion_note()`.

notetools.make_quarter_notes_with_lilypond_multipliers

`abjad.tools.notetools.make_quarter_notes_with_lilypond_multipliers.make_quarter_notes_with`

New in version 2.0. Make quarter notes with *pitches* and *multiplied_durations*:

```
abjad> notetools.make_quarter_notes_with_lilypond_multipliers([0, 2, 4, 5], [(1, 4), (1, 5), (1, 6)],
[Note("c'4 * 1"), Note("d'4 * 4/5"), Note("e'4 * 2/3"), Note("f'4 * 4/7")])
```

Read *pitches* cyclically where the length of *pitches* is less than the length of *multiplied_durations*:

```
abjad> notetools.make_quarter_notes_with_lilypond_multipliers([0], [(1, 4), (1, 5), (1, 6), (1, 7)],
[Note("c'4 * 1"), Note("c'4 * 4/5"), Note("c'4 * 2/3"), Note("c'4 * 4/7")])
```

Read *multiplied_durations* cyclically where the length of *multiplied_durations* is less than the length of *pitches*:

```
abjad> notetools.make_quarter_notes_with_lilypond_multipliers([0, 2, 4, 5], [(1, 5)])
[Note("c'4 * 4/5"), Note("d'4 * 4/5"), Note("e'4 * 4/5"), Note("f'4 * 4/5")]
```

Return list of zero or more newly constructed notes. Changed in version 2.0: renamed `construct.quarter_notes_with_multipliers()` to `notetools.make_quarter_notes_with_lilypond_multipliers()`.

notetools.make_repeated_notes

`abjad.tools.notetools.make_repeated_notes.make_repeated_notes(count, duration=Duration(1, 8))`

Make *count* repeated notes with note head-assignable *duration*:

```
abjad> notetools.make_repeated_notes(4)
[Note("c'8"), Note("c'8"), Note("c'8"), Note("c'8")]
```

Make *count* repeated tie chains with tied *duration*:

```
abjad> notes = notetools.make_repeated_notes(2, (5, 16))
abjad> voice = Voice(notes)
```

```
abjad> f(voice)
\new Voice {
  c'4 ~
  c'16
  c'4 ~
  c'16
}
```


Make ad hoc tuplet holding *count* repeated notes with nonbinary *duration*:

```
abjad> notetools.make_repeated_notes(3, (1, 12))
[Tuplet(2/3, [c'8, c'8, c'8])]
```

Set pitch of all notes created to middle C.

Return list of zero or more newly constructed notes or list of one newly constructed tuplet. Changed in version 2.0: renamed `construct.run()` to `notetools.make_repeated_notes()`.

notetools.make_repeated_notes_from_time_signature

`abjad.tools.notetools.make_repeated_notes_from_time_signature`.**make_repeated_notes_from_time_signature**

New in version 2.0. Make repeated notes from *time_signature*:

```
abjad> notetools.make_repeated_notes_from_time_signature((5, 32))
[Note("c'32"), Note("c'32"), Note("c'32"), Note("c'32"), Note("c'32")]
```

Make repeated notes with *pitch* from *time_signature*:

```
abjad> notetools.make_repeated_notes_from_time_signature((5, 32), pitch = "d'")
[Note("d'32"), Note("d'32"), Note("d'32"), Note("d'32"), Note("d'32")]
```

Return list of notes.

notetools.make_repeated_notes_from_time_signatures

`abjad.tools.notetools.make_repeated_notes_from_time_signatures`.**make_repeated_notes_from_time_signatures**

Make repeated notes from *time_signatures*:

```
notetools.make_repeated_notes_from_time_signatures([(2, 8), (3, 32)])
[[Note("c'8"), Note("c'8")], [Note("c'32"), Note("c'32"), Note("c'32")]]
```

Make repeated notes with *pitch* from *time_signatures*:

```
abjad> notetools.make_repeated_notes_from_time_signatures([(2, 8), (3, 32)], pitch = "d'")
[[Note("d'8"), Note("d'8")], [Note("d'32"), Note("d'32"), Note("d'32")]]
```

Return two-dimensional list of note lists.

Use `sequencetools.flatten_sequence()` to flatten output if required.

notetools.make_repeated_notes_with_shorter_notes_at_end

`abjad.tools.notetools.make_repeated_notes_with_shorter_notes_at_end`.**make_repeated_notes_with_shorter_notes_at_end**

Make repeated notes with *pitch* and *written_duration* summing to *total_duration* under *prolation*:

```
abjad> voice = Voice(notetools.make_repeated_notes_with_shorter_notes_at_end(0, Duration(1, 16),

abjad> f(voice)
\new Voice {
    c'16
    c'16
    c'16
    c'16
}
```

Fill binary remaining duration with binary notes of lesser written duration:

```
abjad> voice = Voice(notetools.make_repeated_notes_with_shorter_notes_at_end(0, Duration(1, 16),

abjad> f(voice)
\new Voice {
    c'16
    c'16
    c'16
    c'16
    c'32
}
```

Fill nonbinary remaining duration with ad hoc tuplet:

```
abjad> voice = Voice(notetools.make_repeated_notes_with_shorter_notes_at_end(0, Duration(1, 16),

abjad> f(voice)
\new Voice {
    c'16
    c'16
    c'16
    c'16
    c'16
    c'16
    \times 4/5 {
        c'32
    }
}
```

Set *prolation* when constructing notes in a nonbinary measure.

Return list of newly constructed components. Changed in version 2.0: renamed `construct.note_train()` to `notetools.make_repeated_notes_with_shorter_notes_at_end()`.

notetools.yield_groups_of_notes_in_sequence

`abjad.tools.notetools.yield_groups_of_notes_in_sequence.yield_groups_of_notes_in_sequence(s)`
 New in version 2.0. Yield groups of notes in *sequence*:

```
abjad> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
```

```

    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}

abjad> for note in notetools.yield_groups_of_notes_in_sequence(staff):
...     note
...
(Note("c'8"), Note("d'8"))
(Note("g'8"), Note("a'8"))

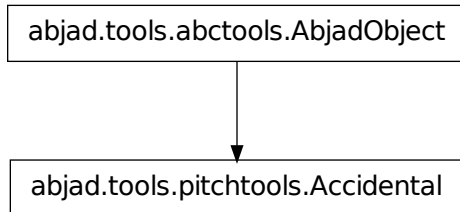
```

Return generator.

pitchtools

pitchtools concrete classes

pitchtools.Accidental



class `abjad.tools.pitchtools.Accidental.Accidental` (*arg*='')

New in version 2.0. Abjad model of the accidental:

```

abjad> pitchtools.Accidental('s')
Accidental('s')

```

Accidentals are immutable.

Read-only Properties

`Accidental.alphabetic_accidental_abbreviation`

Read-only alphabetic string:

```

abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.alphabetic_accidental_abbreviation
's'

```

Return string.

Accidental.format

Read-only LilyPond input format of accidental:

```
abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.format
's'
```

Return string.

Accidental.is_adjusted

True for all accidentals equal to a nonzero number of semitones. False otherwise:

```
abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.is_adjusted
True
```

Return boolean.

Accidental.name

Read-only name of accidental:

```
abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.name
'sharp'
```

Return string.

Accidental.semitones

Read-only semitones of accidental:

```
abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.semitones
1
```

Return number.

Accidental.symbolic_accidental_string

Read-only symbolic string of accidental:

```
abjad> accidental = pitchtools.Accidental('s')
abjad> accidental.symbolic_accidental_string
'#'
```

Return string.

Special Methods

Accidental.__add__(arg)

Accidental.__delattr__()

x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Accidental.__eq__(arg)

Accidental.__ge__(arg)

Accidental.__gt__(arg)

Accidental.__hash__() <==> hash(x)

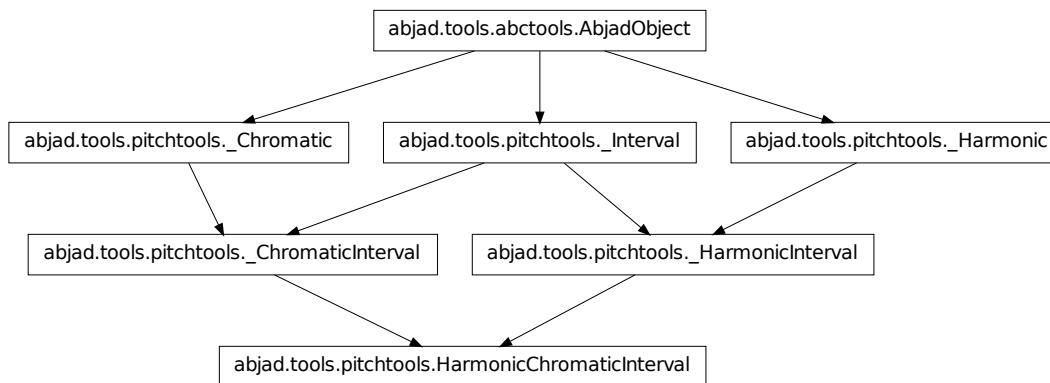
Note: Inherited from `__builtin__.object`

```
Accidental.__le__(arg)
Accidental.__lt__(arg)
Accidental.__ne__(arg)
Accidental.__neg__()
Accidental.__nonzero__()
Accidental.__repr__()
Accidental.__setattr__(arg)
    x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
Accidental.__str__()
Accidental.__sub__(arg)
```

`pitchtools.HarmonicChromaticInterval`



class `abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval`

New in version 2.0. Abjad model of harmonic chromatic interval:

```
abjad> pitchtools.HarmonicChromaticInterval(-14)
HarmonicChromaticInterval(14)
```

Harmonic chromatic intervals are immutable.

Read-only Properties

HarmonicChromaticInterval.**cents**

Note: Inherited from `pitchtools._Interval`

HarmonicChromaticInterval.**harmonic_chromatic_interval_class**

Read-only harmonic chromatic interval-class:

```
abjad> harmonic_chromatic_interval = pitchtools.HarmonicChromaticInterval(14)
abjad> harmonic_chromatic_interval.harmonic_chromatic_interval_class
HarmonicChromaticIntervalClass(2)
```

Return harmonic chromatic interval-class.

HarmonicChromaticInterval.**interval_class**

Note: Inherited from `pitchtools._Interval`

HarmonicChromaticInterval.**number**

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.**semitones**

Note: Inherited from `pitchtools._ChromaticInterval`

Special Methods

HarmonicChromaticInterval.**__abs__**()

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.**__add__**(arg)

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

HarmonicChromaticInterval.**__eq__**(arg)

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__float__()

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__ge__(arg)

HarmonicChromaticInterval.__gt__(arg)

HarmonicChromaticInterval.__hash__()

Note: Inherited from `pitchtools._Interval`

HarmonicChromaticInterval.__int__()

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__le__(arg)

HarmonicChromaticInterval.__lt__(arg)

HarmonicChromaticInterval.__ne__(arg)

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__repr__()

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__setattr__()
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

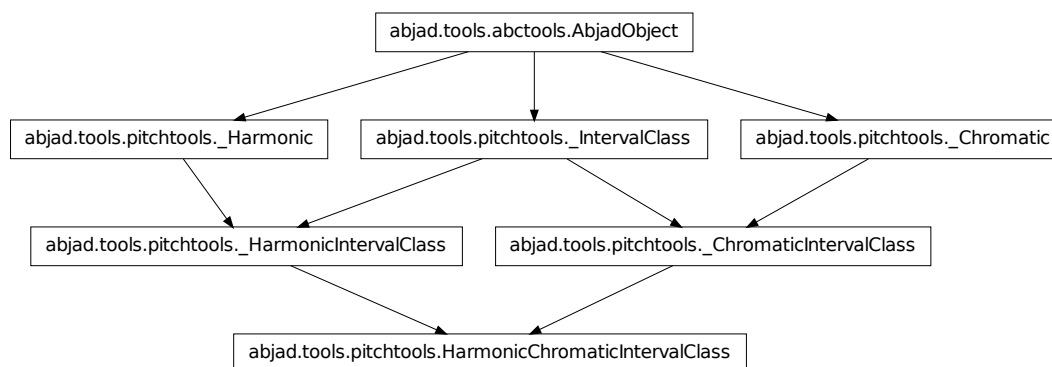
HarmonicChromaticInterval.__str__()

Note: Inherited from `pitchtools._ChromaticInterval`

HarmonicChromaticInterval.__sub__(arg)

Note: Inherited from `pitchtools._ChromaticInterval`

pitchtools.HarmonicChromaticIntervalClass



class `abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass`. **HarmonicChromaticIntervalClass**
 New in version 2.0. Abjad model of harmonic chromatic interval-class:

```
abjad> pitchtools.HarmonicChromaticIntervalClass(-14)
HarmonicChromaticIntervalClass(2)
```

Harmonic chromatic interval-classes are immutable.

Read-only Properties

`HarmonicChromaticIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

Special Methods

`HarmonicChromaticIntervalClass.__abs__()`

Note: Inherited from `pitchtools._ChromaticIntervalClass`

`HarmonicChromaticIntervalClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicChromaticIntervalClass.__eq__(arg)`

`HarmonicChromaticIntervalClass.__float__()`

Note: Inherited from `pitchtools._ChromaticIntervalClass`

HarmonicChromaticIntervalClass.__ge__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicChromaticIntervalClass.__gt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

HarmonicChromaticIntervalClass.__hash__()

Note: Inherited from `pitchtools._IntervalClass`

HarmonicChromaticIntervalClass.__int__()

Note: Inherited from `pitchtools._ChromaticIntervalClass`

HarmonicChromaticIntervalClass.__le__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicChromaticIntervalClass.__lt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicChromaticIntervalClass.__ne__(arg)

HarmonicChromaticIntervalClass.__repr__()

Note: Inherited from `pitchtools._IntervalClass`

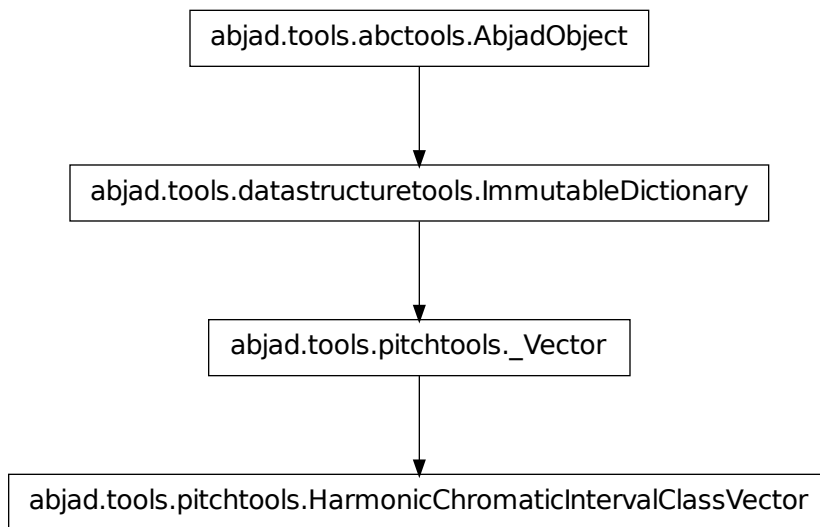
HarmonicChromaticIntervalClass.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

HarmonicChromaticIntervalClass.__str__()

Note: Inherited from `pitchtools._IntervalClass`

`pitchtools.HarmonicChromaticIntervalClassVector`



class `abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector`
 New in version 2.0. Abjad model of harmonic chromatic interval-class vector:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8")
abjad> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(staff)
abjad> print hcicv
0 1 3 2 1 2 0 1 0 0 0 0
  
```

Harmonic chromatic interval-class vector is quartertone-aware:

```

abjad> staff.append(Note(1.5, (1, 4)))
abjad> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(staff)
abjad> print hcicv
0 1 3 2 1 2 0 1 0 0 0 0
1 1 1 1 0 1 0 0 0 0 0 0
  
```

Harmonic chromatic interval-class vectors are immutable.

Methods

`HarmonicChromaticIntervalClassVector.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.has_none_of(chromatic_interval_numbers)`
 True when harmonic chromatic interval-class vector contains none of *chromatic_interval_numbers*. Otherwise false:

```
abjad> hcicv = pitchtools.HarmonicChromaticIntervalClassVector(Staff("c'8 d'8 e'8 f'8 g'8"))
abjad> hcicv.has_none_of([9, 10, 11])
True
```

Return boolean.

`HarmonicChromaticIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.pop(k[, d])` → *v*, remove specified key and return the corresponding value.
If key is not found, *d* is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.popitem()` → (*k*, *v*), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if *D* is empty.

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.setdefault(k[, d])` → *D.get(k,d)*, also set *D[k]=d* if *k* not in *D*

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.update(E, **F)` → `None`. Update *D* from dict/iterable *E* and *F*.
If *E* has a `.keys()` method, does: for *k* in *E*: *D[k] = E[k]* If *E* lacks `.keys()` method, does: for (*k*, *v*) in *E*: *D[k] = v*
In either case, this is followed by: for *k* in *F*: *D[k] = F[k]*

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.values()` → list of *D*'s values

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on *D*'s items

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on *D*'s keys

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.viewvalues()` → an object providing a view on *D*'s values

Note: Inherited from `__builtin__.dict`

Special Methods

HarmonicChromaticIntervalClassVector.__cmp__(y) <==> cmp(x, y)

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__contains__(k) → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

HarmonicChromaticIntervalClassVector.__delitem__(*args)

Note: Inherited from `datastructuretools.ImmutableDictionary`

HarmonicChromaticIntervalClassVector.__eq__()
x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__ge__()
x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__getitem__()
x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__gt__()
x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.dict`

HarmonicChromaticIntervalClassVector.__le__()
x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__len__()` $\iff \text{len}(x)$

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.dict`

`HarmonicChromaticIntervalClassVector.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.dict`

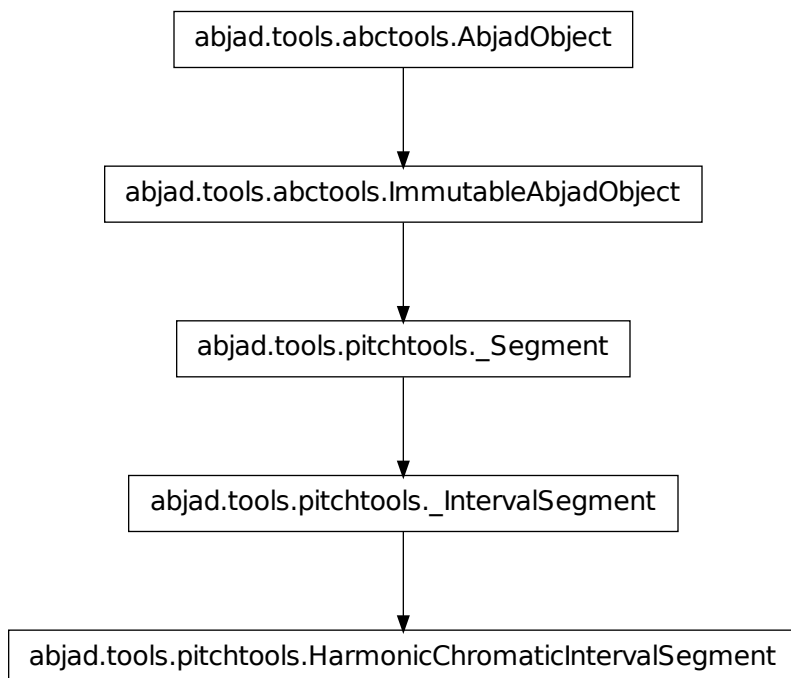
`HarmonicChromaticIntervalClassVector.__repr__()`

`HarmonicChromaticIntervalClassVector.__setattr__()`
`x.__setattr__('name', value)` $\iff x.name = value$

Note: Inherited from `__builtin__.object`

`HarmonicChromaticIntervalClassVector.__setitem__(*args)`

`HarmonicChromaticIntervalClassVector.__str__()`

pitchtools.HarmonicChromaticIntervalSegment

class `abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment`

New in version 2.0. Abjad model of harmonic chromatic interval segment:

```
abjad> pitchtools.HarmonicChromaticIntervalSegment([10, -12, -13, -13.5])
HarmonicChromaticIntervalSegment(10, 12, 13, 13.5)
```

Harmonic chromatic interval segments are immutable.

Read-only Properties

`HarmonicChromaticIntervalSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalSegment`

`HarmonicChromaticIntervalSegment.intervals`

Note: Inherited from `pitchtools._IntervalSegment`

Methods

`HarmonicChromaticIntervalSegment.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.rotate(n)`

Note: Inherited from `pitchtools._IntervalSegment`

Special Methods

`HarmonicChromaticIntervalSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`HarmonicChromaticIntervalSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicChromaticIntervalSegment.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`HarmonicChromaticIntervalSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**getslice**__(start, stop)

Note: Inherited from `pitchtools._Segment`

HarmonicChromaticIntervalSegment.__**gt**__()
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**hash**__() <==> *hash(x)*

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**iter**__() <==> *iter(x)*

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**le**__()
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**len**__() <==> *len(x)*

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**lt**__()
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**mul**__(n)

Note: Inherited from `pitchtools._Segment`

HarmonicChromaticIntervalSegment.__**ne**__()
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

HarmonicChromaticIntervalSegment.__**repr**__()

Note: Inherited from `pitchtools._IntervalSegment`

`HarmonicChromaticIntervalSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

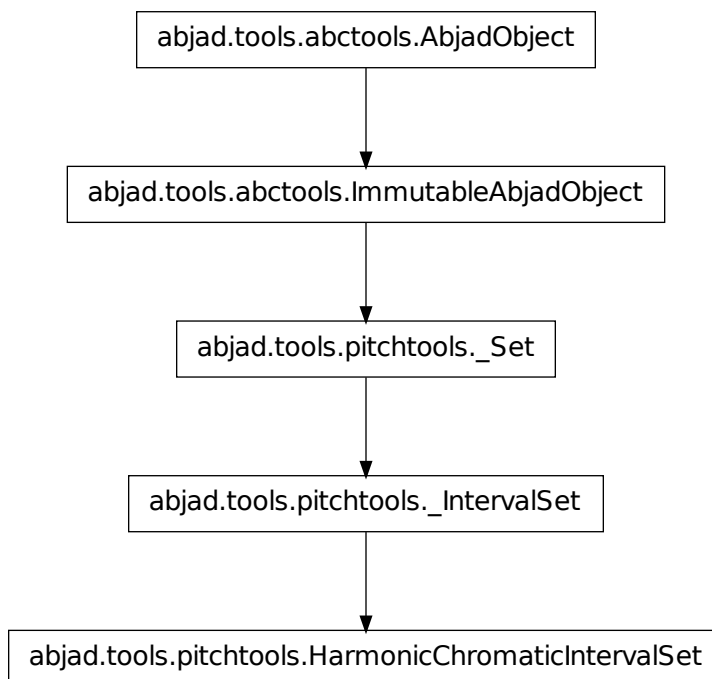
`HarmonicChromaticIntervalSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`HarmonicChromaticIntervalSegment.__str__()`

Note: Inherited from `pitchtools._IntervalSegment`

`pitchtools.HarmonicChromaticIntervalSet`



class `abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet`. **Harmon**

New in version 2.0. Abjad model of harmonic chromatic interval set:

```
abjad> pitchtools.HarmonicChromaticIntervalSet([10, -12, -13, -13, -13.5])
HarmonicChromaticIntervalSet(10, 12, 13, 13.5)
```

Harmonic chromatic interval sets are immutable.

Read-only Properties

HarmonicChromaticIntervalSet.**harmonic_chromatic_interval_numbers**

HarmonicChromaticIntervalSet.**harmonic_chromatic_intervals**

Methods

HarmonicChromaticIntervalSet.**copy**()

Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**difference**()

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**intersection**()

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**isdisjoint**()

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**issubset**()

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**issuperset**()

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**symmetric_difference**()

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**union**()

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

HarmonicChromaticIntervalSet.**__and__**()

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__cmp__**(y) <==> *cmp*(x, y)

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__contains__**()

`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

HarmonicChromaticIntervalSet.**__eq__**()

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__ge__**()

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__gt__**()

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.**__hash__**() <==> *hash*(x)

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__iter__() $\iff iter(x)$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__le()
 $x.__le__(y) \iff x \leq y$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__len__() $\iff len(x)$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__lt()
 $x.__lt__(y) \iff x < y$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__ne()
 $x.__ne__(y) \iff x \neq y$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__or()
 $x.__or__(y) \iff x \cup y$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__rand()
 $x.__rand__(y) \iff y \& x$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__repr__()

HarmonicChromaticIntervalSet.__ror()
 $x.__ror__(y) \iff y \cup x$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__rsub()
 $x.__rsub__(y) \iff y - x$

Note: Inherited from `__builtin__.frozenset`

HarmonicChromaticIntervalSet.__rxor()
 $x.__rxor__(y) \iff y \oplus x$

Note: Inherited from `__builtin__.frozenset`

`HarmonicChromaticIntervalSet.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

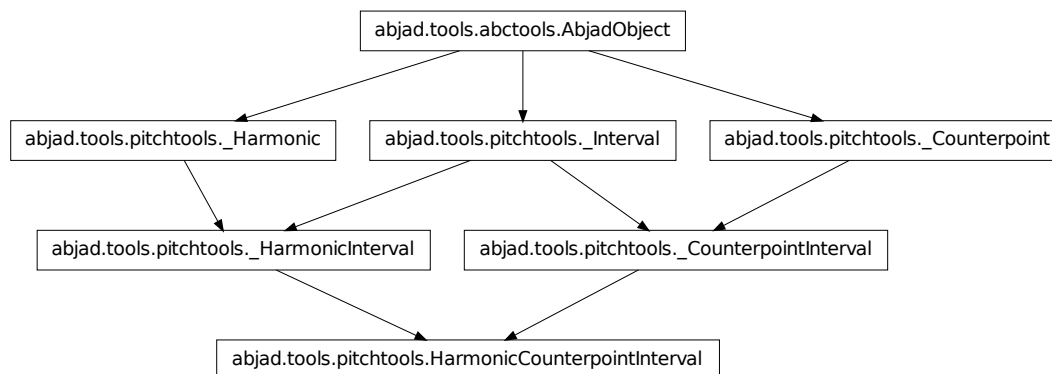
`HarmonicChromaticIntervalSet.__str__()`
`HarmonicChromaticIntervalSet.__sub__()`
`x.__sub__(y) <==> x-y`

Note: Inherited from `__builtin__.frozenset`

`HarmonicChromaticIntervalSet.__xor__()`
`x.__xor__(y) <==> x^y`

Note: Inherited from `__builtin__.frozenset`

`pitchtools.HarmonicCounterpointInterval`



class `abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval`
 New in version 2.0. Abjad model of harmonic counterpoint interval:

```

abjad> pitchtools.HarmonicCounterpointInterval(-9)
HarmonicCounterpointInterval(9)
  
```

Harmonic counterpoint intervals are immutable.

Read-only Properties

HarmonicCounterpointInterval.**cents**

Note: Inherited from `pitchtools._Interval`

HarmonicCounterpointInterval.**harmonic_counterpoint_interval_class**

HarmonicCounterpointInterval.**interval_class**

Note: Inherited from `pitchtools._Interval`

HarmonicCounterpointInterval.**number**

Note: Inherited from `pitchtools._CounterpointInterval`

HarmonicCounterpointInterval.**semitones**

Note: Inherited from `pitchtools._CounterpointInterval`

Special Methods

HarmonicCounterpointInterval.**__abs__**()

Note: Inherited from `pitchtools._CounterpointInterval`

HarmonicCounterpointInterval.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

HarmonicCounterpointInterval.**__eq__**(arg)

HarmonicCounterpointInterval.**__float__**()

Note: Inherited from `pitchtools._CounterpointInterval`

HarmonicCounterpointInterval.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicCounterpointInterval.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`HarmonicCounterpointInterval.__hash__()`

Note: Inherited from `pitchtools._Interval`

`HarmonicCounterpointInterval.__int__()`

Note: Inherited from `pitchtools._CounterpointInterval`

`HarmonicCounterpointInterval.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HarmonicCounterpointInterval.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HarmonicCounterpointInterval.__ne__(arg)`

`HarmonicCounterpointInterval.__repr__()`

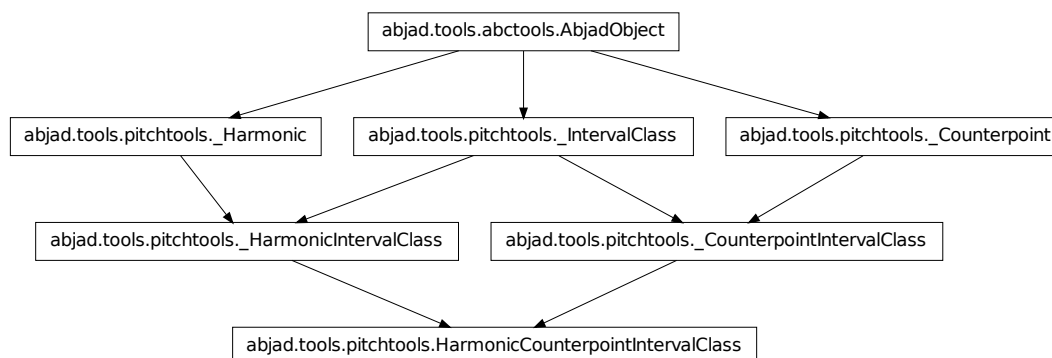
Note: Inherited from `pitchtools._Interval`

`HarmonicCounterpointInterval.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`HarmonicCounterpointInterval.__str__()`

Note: Inherited from `pitchtools._Interval`

pitchtools.HarmonicCounterpointIntervalClass

class `abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass`
 New in version 2.0. Abjad model of harmonic counterpoint interval-class:

```
abjad> pitchtools.HarmonicCounterpointIntervalClass(-9)
HarmonicCounterpointIntervalClass(2)
```

Harmonic counterpoint interval-classes are immutable.

Read-only Properties

`HarmonicCounterpointIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

Special Methods

`HarmonicCounterpointIntervalClass.__abs__()`

Note: Inherited from `pitchtools._CounterpointIntervalClass`

`HarmonicCounterpointIntervalClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicCounterpointIntervalClass.__eq__(arg)`

`HarmonicCounterpointIntervalClass.__float__()`

Note: Inherited from `pitchtools._CounterpointIntervalClass`

HarmonicCounterpointIntervalClass.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicCounterpointIntervalClass.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

HarmonicCounterpointIntervalClass.__hash__()

Note: Inherited from `pitchtools._IntervalClass`

HarmonicCounterpointIntervalClass.__int__()

Note: Inherited from `pitchtools._CounterpointIntervalClass`

HarmonicCounterpointIntervalClass.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicCounterpointIntervalClass.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicCounterpointIntervalClass.__ne__(arg)

HarmonicCounterpointIntervalClass.__repr__()

Note: Inherited from `pitchtools._IntervalClass`

HarmonicCounterpointIntervalClass.__setattr__()

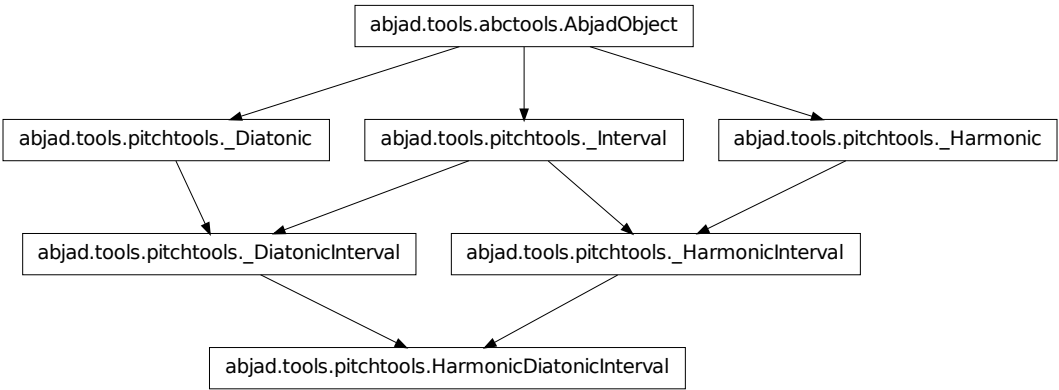
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

HarmonicCounterpointIntervalClass.__str__()

Note: Inherited from `pitchtools._IntervalClass`

pitchtools.HarmonicDiatonicInterval



class `abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval`
New in version 2.0. Abjad model harmonic diatonic interval:

```
abjad> pitchtools.HarmonicDiatonicInterval('M9')
HarmonicDiatonicInterval('M9')
```

Harmonic diatonic intervals are immutable.

Read-only Properties

`HarmonicDiatonicInterval.cents`

Note: Inherited from `pitchtools._Interval`

`HarmonicDiatonicInterval.diatonic_interval_class`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.harmonic_counterpoint_interval`

`HarmonicDiatonicInterval.harmonic_diatonic_interval_class`

`HarmonicDiatonicInterval.interval_class`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.interval_string`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.melodic_diatonic_interval_ascending`

`HarmonicDiatonicInterval.melodic_diatonic_interval_descending`

`HarmonicDiatonicInterval.number`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.quality_string`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.semitones`

`HarmonicDiatonicInterval.staff_spaces`

Special Methods

`HarmonicDiatonicInterval.__abs__()`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicDiatonicInterval.__eq__(arg)`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.__float__()`

Note: Inherited from `pitchtools._DiatonicInterval`

`HarmonicDiatonicInterval.__ge__(arg)`

`HarmonicDiatonicInterval.__gt__(arg)`

HarmonicDiatonicInterval.__hash__()

Note: Inherited from `pitchtools._Interval`

HarmonicDiatonicInterval.__int__()

Note: Inherited from `pitchtools._DiatonicInterval`

HarmonicDiatonicInterval.__le__(arg)

HarmonicDiatonicInterval.__lt__(arg)

HarmonicDiatonicInterval.__ne__(arg)

Note: Inherited from `pitchtools._DiatonicInterval`

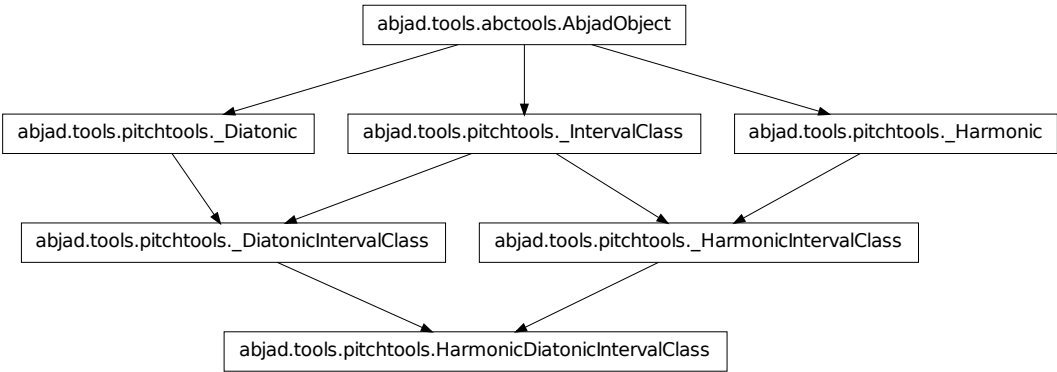
HarmonicDiatonicInterval.__repr__()

HarmonicDiatonicInterval.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

HarmonicDiatonicInterval.__str__()

pitchtools.HarmonicDiatonicIntervalClass



class `abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass`. **Harmon**
New in version 2.0. Abjad model harmonic diatonic interval-class:

```
abjad> pitchtools.HarmonicDiatonicIntervalClass('-M9')
HarmonicDiatonicIntervalClass('M2')
```

Harmonic diatonic interval-classes are immutable.

Read-only Properties

`HarmonicDiatonicIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

`HarmonicDiatonicIntervalClass.quality_string`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

Methods

`HarmonicDiatonicIntervalClass.invert()`

Read-only inversion of harmonic diatonic interval-class:

```
abjad> hdic = pitchtools.HarmonicDiatonicIntervalClass('major', -9)
abjad> hdic.invert()
HarmonicDiatonicIntervalClass('m7')
```

Return harmonic diatonic interval-class.

Special Methods

`HarmonicDiatonicIntervalClass.__abs__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`HarmonicDiatonicIntervalClass.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicDiatonicIntervalClass.__eq__(arg)`

`HarmonicDiatonicIntervalClass.__float__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`HarmonicDiatonicIntervalClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicDiatonicIntervalClass.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

HarmonicDiatonicIntervalClass.__hash__()

Note: Inherited from `pitchtools._IntervalClass`

HarmonicDiatonicIntervalClass.__int__()

Note: Inherited from `pitchtools._DiatonicIntervalClass`

HarmonicDiatonicIntervalClass.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicDiatonicIntervalClass.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

HarmonicDiatonicIntervalClass.__ne__(arg)
HarmonicDiatonicIntervalClass.__repr__()

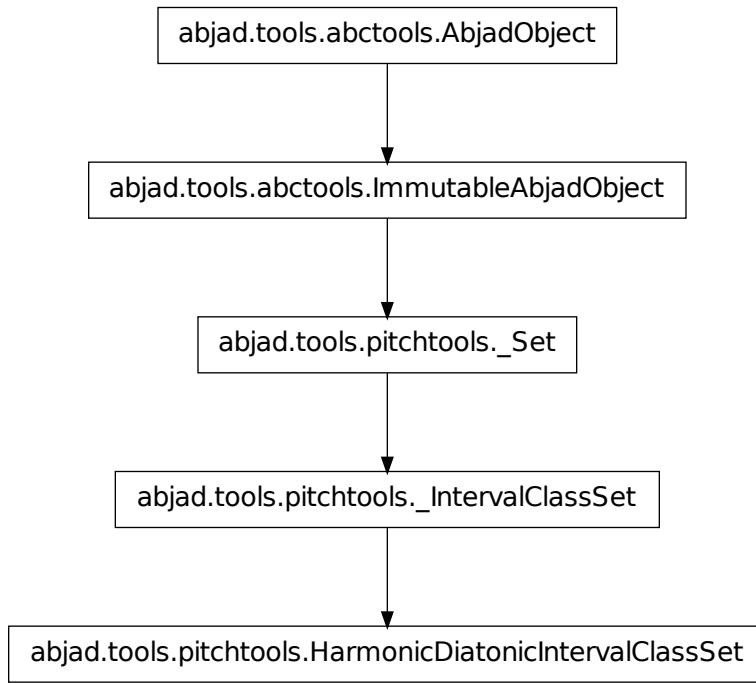
Note: Inherited from `pitchtools._DiatonicIntervalClass`

HarmonicDiatonicIntervalClass.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

HarmonicDiatonicIntervalClass.__str__()

pitchtools.HarmonicDiatonicIntervalClassSet



class `abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet`

New in version 2.0. Abjad model of harmonic diatonic interval-class set:

```
abjad> pitchtools.HarmonicDiatonicIntervalClassSet('m2 M2 m3 M3') # doctest: +SKIP
HarmonicDiatonicIntervalClassSet('m2 M2 m3 M3')
```

Harmonic diatonic interval-class sets are immutable.

Read-only Properties

`HarmonicDiatonicIntervalClassSet.harmonic_diatonic_interval_classes`

Methods

`HarmonicDiatonicIntervalClassSet.copy()`
Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`HarmonicDiatonicIntervalClassSet.difference()`
Return the difference of two or more sets as a new set.
(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**intersection**()

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**isdisjoint**()

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**issubset**()

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**issuperset**()

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**symmetric_difference**()

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**union**()

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

HarmonicDiatonicIntervalClassSet.**__and__**()

$x._\text{and}_(y) \iff x \& y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.**__cmp__**(y) $\iff \text{cmp}(x, y)$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__contains__()
 x.__contains__(y) <==> y in x.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__delattr__()
 x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

HarmonicDiatonicIntervalClassSet.__eq__()
 x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__ge__()
 x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__len__() <==> len(x)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__ne__`()
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__or__`()
`x.__or__(y) <==> x|y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__rand__`()
`x.__rand__(y) <==> y&x`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__repr__`()
 HarmonicDiatonicIntervalClassSet.`__ror__`()
`x.__ror__(y) <==> y|x`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__rsub__`()
`x.__rsub__(y) <==> y-x`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__rxor__`()
`x.__rxor__(y) <==> y^x`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__setattr__`()
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

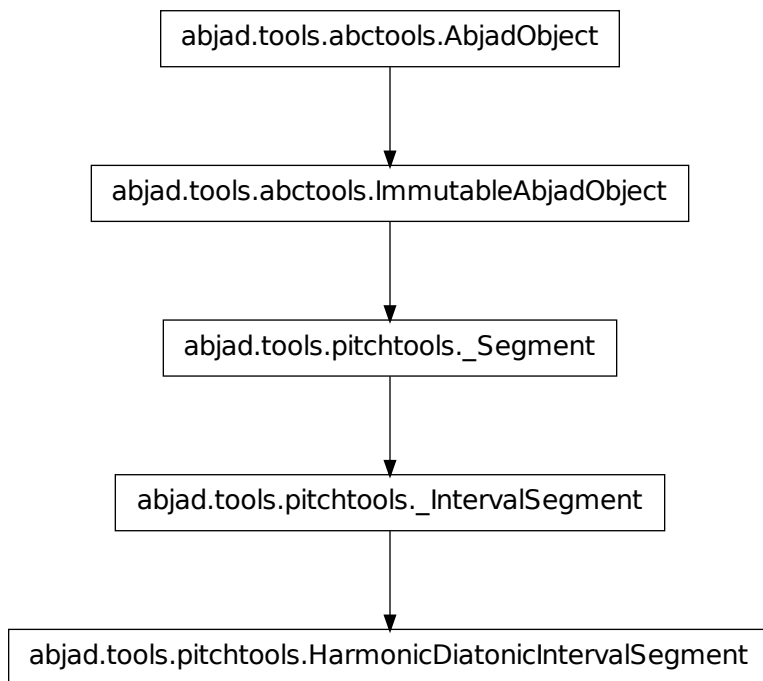
HarmonicDiatonicIntervalClassSet.`__str__`()
 HarmonicDiatonicIntervalClassSet.`__sub__`()
`x.__sub__(y) <==> x-y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalClassSet.`__xor__`()
`x.__xor__(y) <==> x^y`

Note: Inherited from `__builtin__.frozenset`

`pitchtools.HarmonicDiatonicIntervalSegment`



class `abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment`

New in version 2.0. Abjad model of harmonic diatonic interval segment:

```

abjad> pitchtools.HarmonicDiatonicIntervalSegment('m2 M9 m3 M3')
HarmonicDiatonicIntervalSegment('m2 M9 m3 M3')
```

Harmonic diatonic interval segments are immutable.

Read-only Properties

`HarmonicDiatonicIntervalSegment.harmonic_chromatic_interval_segment`
`HarmonicDiatonicIntervalSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalSegment`

`HarmonicDiatonicIntervalSegment.intervals`

Note: Inherited from `pitchtools._IntervalSegment`

`HarmonicDiatonicIntervalSegment.melodic_chromatic_interval_segment`

`HarmonicDiatonicIntervalSegment.melodic_diatonic_interval_segment`

Methods

`HarmonicDiatonicIntervalSegment.count` (*value*) → integer – return number of occurrences of *value*

Note: Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.index` (*value* [, *start* [, *stop*]]) → integer – return first index of *value*.

Raises `ValueError` if the *value* is not present.

Note: Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.rotate` (*n*)

Note: Inherited from `pitchtools._IntervalSegment`

Special Methods

`HarmonicDiatonicIntervalSegment.__add__` (*arg*)

Note: Inherited from `pitchtools._Segment`

`HarmonicDiatonicIntervalSegment.__contains__` ()
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__delattr__` ()
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HarmonicDiatonicIntervalSegment.__eq__` ()
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__ge__` ()
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__getitem__()
 x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__getslice__(start, stop)

Note: Inherited from `pitchtools._Segment`

HarmonicDiatonicIntervalSegment.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__len__() <==> len(x)

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.tuple`

HarmonicDiatonicIntervalSegment.__mul__(n)

Note: Inherited from `pitchtools._Segment`

HarmonicDiatonicIntervalSegment.__ne__()
 x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.tuple`

`HarmonicDiatonicIntervalSegment.__repr__()`

`HarmonicDiatonicIntervalSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

`HarmonicDiatonicIntervalSegment.__setattr__()`

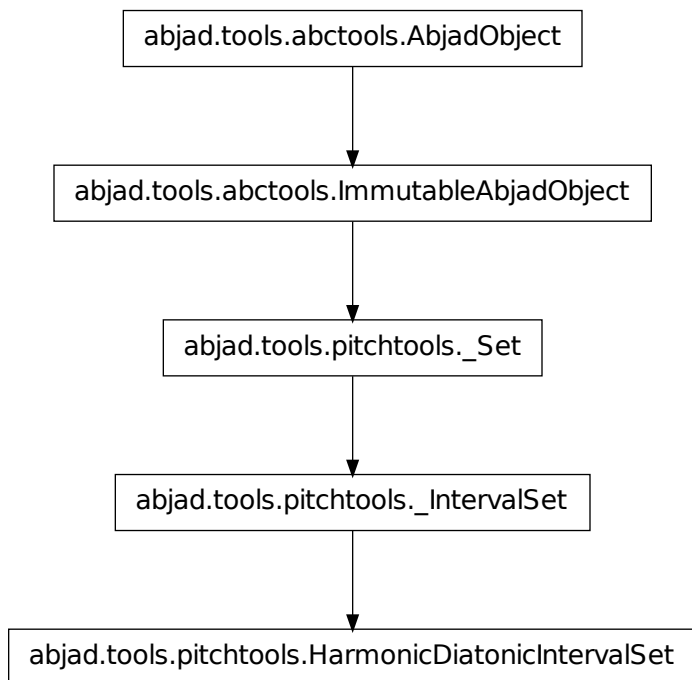
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`HarmonicDiatonicIntervalSegment.__str__()`

Note: Inherited from `pitchtools._IntervalSegment`

`pitchtools.HarmonicDiatonicIntervalSet`



class abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.**Harmonic**

New in version 2.0. Abjad model of harmonic diatonic interval set:

```
abjad> pitchtools.HarmonicDiatonicIntervalSet('m2 m2 M2 M9')
HarmonicDiatonicIntervalSet('m2 M2 M9')
```

Harmonic diatonic interval sets are immutable.

Read-only Properties

HarmonicDiatonicIntervalSet.**harmonic_chromatic_interval_set**

HarmonicDiatonicIntervalSet.**harmonic_diatonic_interval_numbers**

HarmonicDiatonicIntervalSet.**harmonic_diatonic_intervals**

Methods

HarmonicDiatonicIntervalSet.**copy()**

Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**difference()**

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**intersection()**

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**isdisjoint()**

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**issubset()**

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**issuperset()**

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**symmetric_difference**()

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**union**()

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

HarmonicDiatonicIntervalSet.**__and__**()

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**__cmp__**(y) <==> *cmp*(x, y)

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**__contains__**()

`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

HarmonicDiatonicIntervalSet.**__eq__**()

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**__ge__**()

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.**__gt__**()

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__hash__() $\iff hash(x)$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__iter__() $\iff iter(x)$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__le__()
x.__le__(y) $\iff x \leq y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__len__() $\iff len(x)$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__lt__()
x.__lt__(y) $\iff x < y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__ne__()
x.__ne__(y) $\iff x \neq y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__or__()
x.__or__(y) $\iff x \cup y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__rand__()
x.__rand__(y) $\iff y \& x$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__repr__()

HarmonicDiatonicIntervalSet.__ror__()
x.__ror__(y) $\iff y \cup x$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__rsub__()
 $x._\text{rsub}_(y) \iff y-x$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__rxor__()
 $x._\text{rxor}_(y) \iff y^x$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__setattr__()
 $x._\text{setattr}_('name', value) \iff x.name = value$

Note: Inherited from `__builtin__.object`

HarmonicDiatonicIntervalSet.__str__()

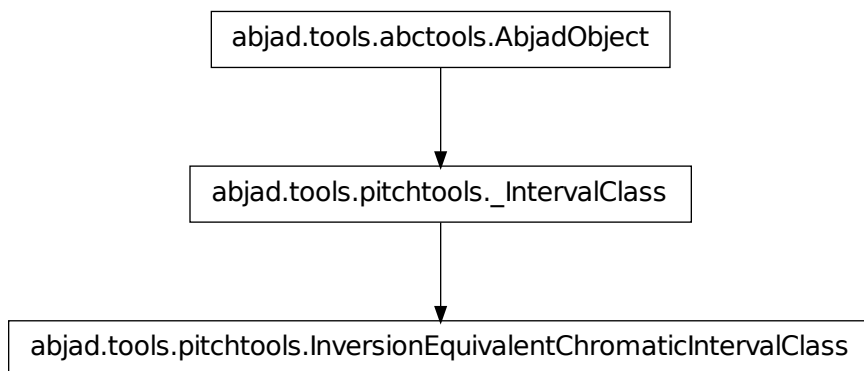
HarmonicDiatonicIntervalSet.__sub__()
 $x._\text{sub}_(y) \iff x-y$

Note: Inherited from `__builtin__.frozenset`

HarmonicDiatonicIntervalSet.__xor__()
 $x._\text{xor}_(y) \iff x^y$

Note: Inherited from `__builtin__.frozenset`

pitchtools.InversionEquivalentChromaticIntervalClass



class `abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChr`
 New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class:

```
abjad> pitchtools.InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentChromaticIntervalClass(1)
```

Inversion-equivalent chromatic interval-classes are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClass.inversion_equivalent_chromatic_interval_number`
`InversionEquivalentChromaticIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

Special Methods

`InversionEquivalentChromaticIntervalClass.__abs__()`
`InversionEquivalentChromaticIntervalClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClass.__eq__(arg)`
`InversionEquivalentChromaticIntervalClass.__float__()`

Note: Inherited from `pitchtools._IntervalClass`

`InversionEquivalentChromaticIntervalClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__hash__()`
`InversionEquivalentChromaticIntervalClass.__int__()`

Note: Inherited from `pitchtools._IntervalClass`

`InversionEquivalentChromaticIntervalClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentChromaticIntervalClass.__ne__(arg)`

`InversionEquivalentChromaticIntervalClass.__neg__()`

`InversionEquivalentChromaticIntervalClass.__repr__()`

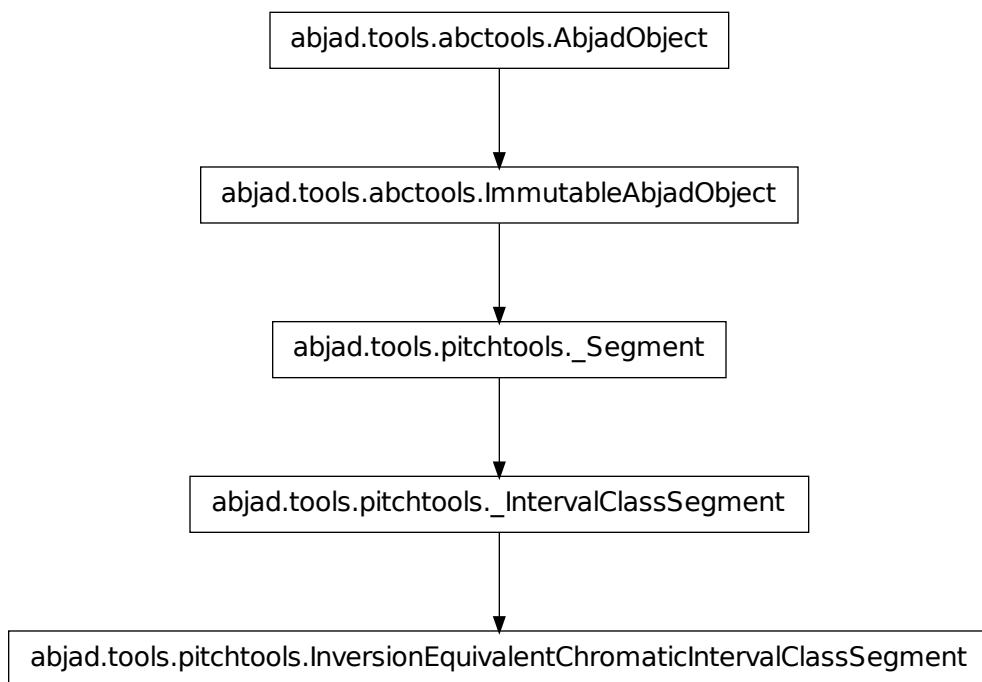
`InversionEquivalentChromaticIntervalClass.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClass.__str__()`

pitchtools.InversionEquivalentChromaticIntervalClassSegment



class `abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment`

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class segment:

```
abjad> pitchtools.InversionEquivalentChromaticIntervalClassSegment([2, 1, 0, 5.5, 6])
InversionEquivalentChromaticIntervalClassSegment(2, 1, 0, 5.5, 6)
```

Inversion-equivalent chromatic interval-class segments are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClassSegment.interval_class_numbers`

Note: Inherited from `pitchtools._IntervalClassSegment`

`InversionEquivalentChromaticIntervalClassSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalClassSegment`

Methods

`InversionEquivalentChromaticIntervalClassSegment.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

Special Methods

`InversionEquivalentChromaticIntervalClassSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`InversionEquivalentChromaticIntervalClassSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClassSegment.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`InversionEquivalentChromaticIntervalClassSegment.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__len__() <==> len(x)`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`InversionEquivalentChromaticIntervalClassSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentChromaticIntervalClassSegment.__repr__()`

`InversionEquivalentChromaticIntervalClassSegment.__rmul__(n)`

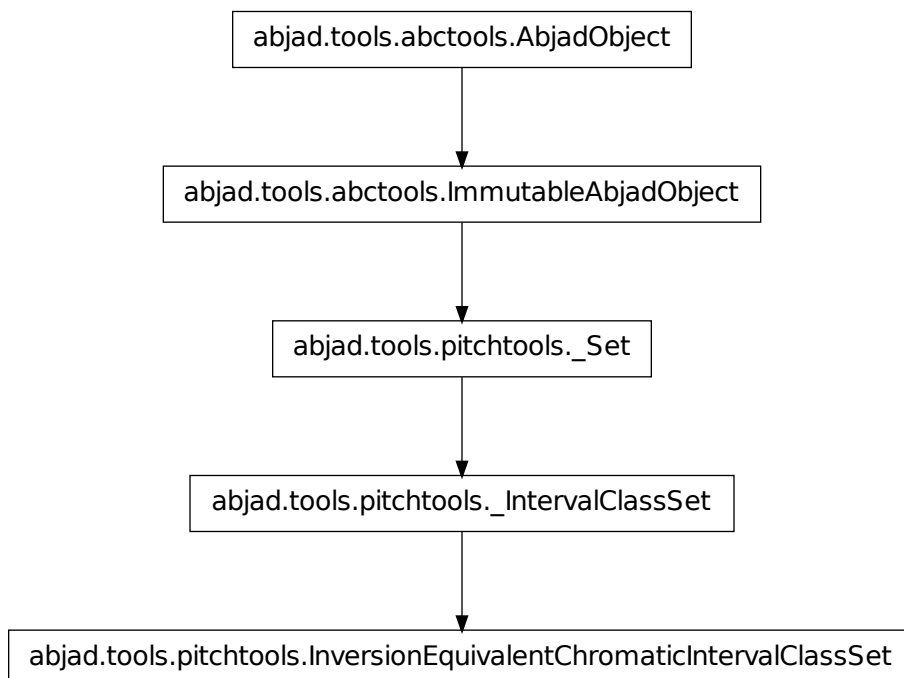
Note: Inherited from `pitchtools._Segment`

`InversionEquivalentChromaticIntervalClassSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClassSegment.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.InversionEquivalentChromaticIntervalClassSet

class `abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalent`

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class set:

```

abjad> pitchtools.InversionEquivalentChromaticIntervalClassSet([1, 1, 6, 2, 2])
InversionEquivalentChromaticIntervalClassSet(1, 2, 6)

```

Inversion-equivalent chromatic interval-class sets are immutable.

Read-only Properties

`InversionEquivalentChromaticIntervalClassSet.inversion_equivalent_chromatic_interval_class`
`InversionEquivalentChromaticIntervalClassSet.inversion_equivalent_chromatic_interval_class`

Methods

`InversionEquivalentChromaticIntervalClassSet.copy()`
 Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

`InversionEquivalentChromaticIntervalClassSet.__and__()`

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__contains__()
x.__contains__(y) <==> y in x.
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__delattr__()
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
InversionEquivalentChromaticIntervalClassSet.__eq__()
x.__eq__(y) <==> x==y
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__ge__()
x.__ge__(y) <==> x>=y
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__gt__()
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__hash__() <==> hash(x)
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__iter__() <==> iter(x)
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__le__()
x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__len__() <==> len(x)
```

Note: Inherited from `__builtin__.frozenset`

```
InversionEquivalentChromaticIntervalClassSet.__lt__()
x.__lt__(y) <==> x<y
```

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__or__()`
`x.__or__(y) <==> x|y`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__rand__()`
`x.__rand__(y) <==> y&x`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__repr__()`
`InversionEquivalentChromaticIntervalClassSet.__ror__()`
`x.__ror__(y) <==> y|x`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__rsub__()`
`x.__rsub__(y) <==> y-x`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__rxor__()`
`x.__rxor__(y) <==> y^x`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClassSet.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

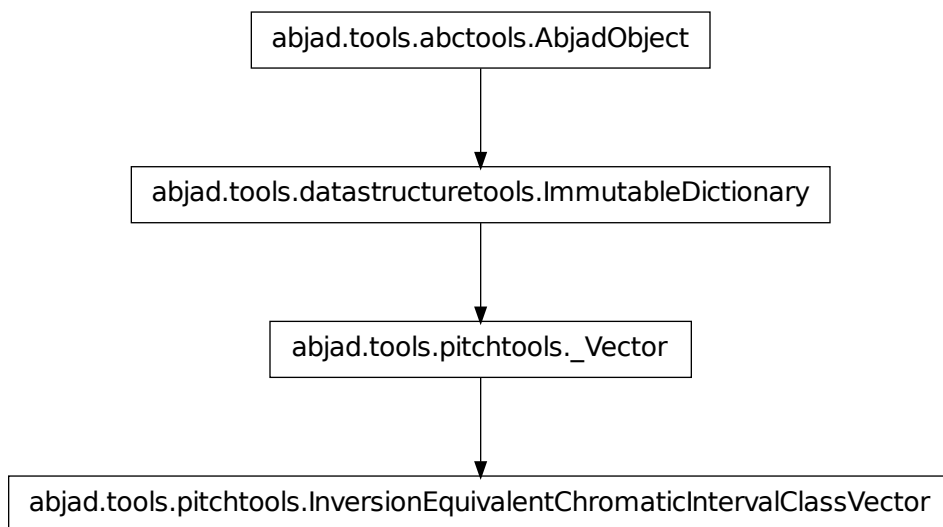
`InversionEquivalentChromaticIntervalClassSet.__sub__()`
`x.__sub__(y) <==> x-y`

Note: Inherited from `__builtin__.frozenset`

`InversionEquivalentChromaticIntervalClassSet.__xor__()`
`x.__xor__(y) <==> x^y`

Note: Inherited from `__builtin__.frozenset`

`pitchtools.InversionEquivalentChromaticIntervalClassVector`



class `abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector`.`InversionEquivalent`

New in version 2.0. Abjad model of inversion-equivalent chromatic interval-class vector:

```
abjad> pitchtools.InversionEquivalentChromaticIntervalClassVector([1, 1, 6, 2, 2, 2])
InversionEquivalentChromaticIntervalClassVector(0 | 2 3 0 0 0 1)
```

Initialize by inversion-equivalent chromatic interval-class counts:

```
abjad> pitchtools.InversionEquivalentChromaticIntervalClassVector(counts = [2, 3, 0, 0, 0, 1])
InversionEquivalentChromaticIntervalClassVector(0 | 2 3 0 0 0 1)
```

Inversion-equivalent chromatic interval-class vectors are immutable.

Methods

`InversionEquivalentChromaticIntervalClassVector.clear()` → `None`. Remove all items from `D`.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.get(k[, d])` → D[k] if k in D, else d.
d defaults to None.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.has_key(k)` → True if D has a key
k, else False

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.items()` → list of D's (key, value)
pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.iteritems()` → an iterator over the
(key, value) items of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.iterkeys()` → an iterator over the
keys of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.itervalues()` → an iterator over
the values of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.pop(k[, d])` → v, remove specified
key and return the corresponding
value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.setdefault(k[, d])` → `D.get(k, d)`, also set `D[k]=d` if k not in D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.update(E, **F)` → None. Update D from dict/iterable E and F. If E has a `.keys()` method, does: for k in E: `D[k] = E[k]` If E lacks `.keys()` method, does: for (k, v) in E: `D[k] = v` In either case, this is followed by: for k in F: `D[k] = F[k]`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`InversionEquivalentChromaticIntervalClassVector.__cmp__(y)` <==> `cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__contains__(k) → True if D has a key k, else False`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentChromaticIntervalClassVector.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentChromaticIntervalClassVector.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__len__() <==> len(x)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentChromaticIntervalClassVector.__repr__()`
`InversionEquivalentChromaticIntervalClassVector.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

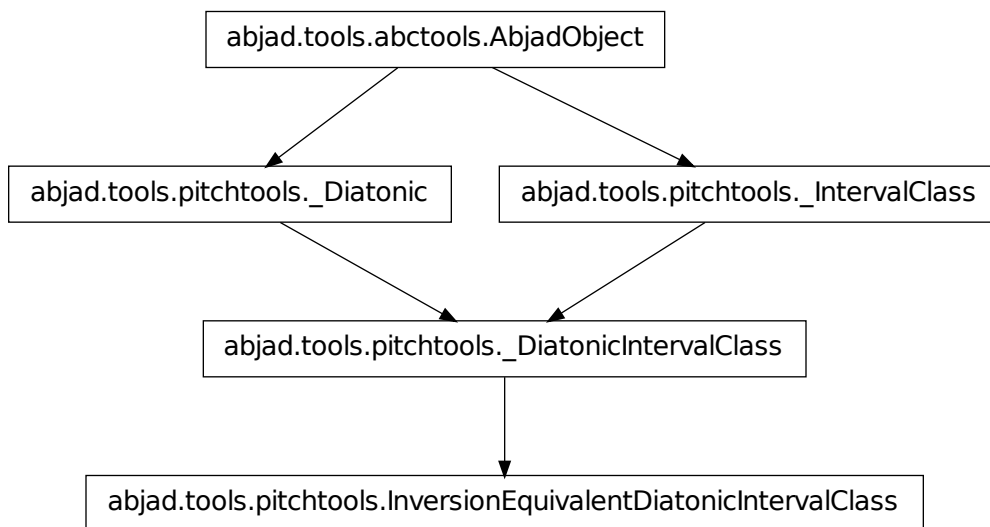
`InversionEquivalentChromaticIntervalClassVector.__setitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentChromaticIntervalClassVector.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.InversionEquivalentDiatonicIntervalClass



class `abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass`
 New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class:

```
abjad> pitchtools.InversionEquivalentDiatonicIntervalClass('-m14')
InversionEquivalentDiatonicIntervalClass('M2')
```

Inversion-equivalent diatonic interval-classes are immutable.

Read-only Properties

`InversionEquivalentDiatonicIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

`InversionEquivalentDiatonicIntervalClass.quality_string`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

Special Methods

`InversionEquivalentDiatonicIntervalClass.__abs__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`InversionEquivalentDiatonicIntervalClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClass.__eq__(arg)`

`InversionEquivalentDiatonicIntervalClass.__float__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`InversionEquivalentDiatonicIntervalClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__hash__()`

Note: Inherited from `pitchtools._IntervalClass`

`InversionEquivalentDiatonicIntervalClass.__int__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`InversionEquivalentDiatonicIntervalClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionEquivalentDiatonicIntervalClass.__ne__(arg)`

`InversionEquivalentDiatonicIntervalClass.__repr__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

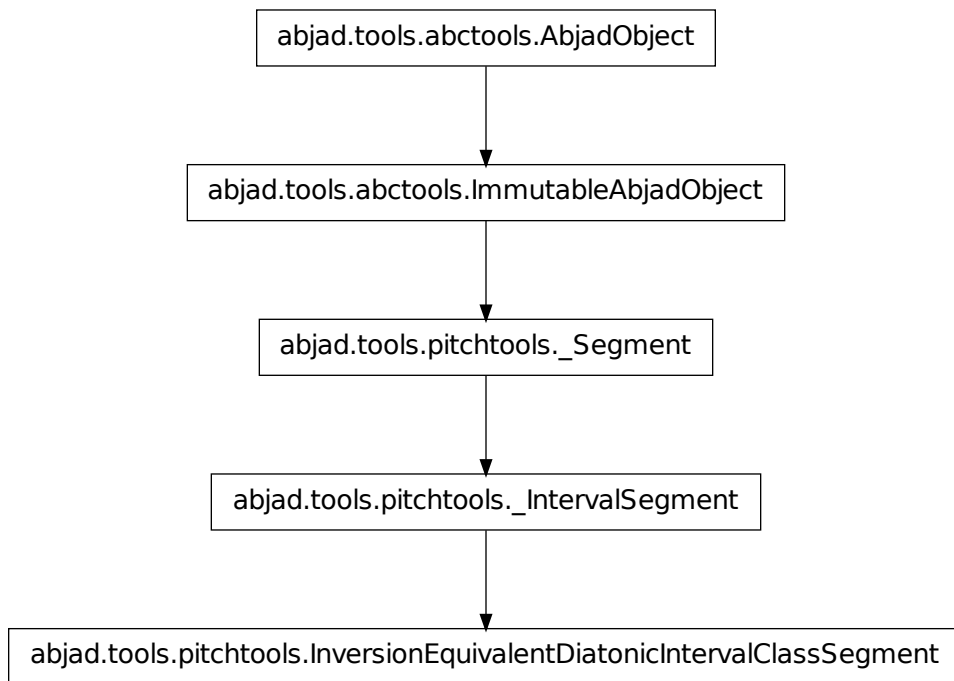
`InversionEquivalentDiatonicIntervalClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClass.__str__()`

Note: Inherited from `pitchtools._DiatonicIntervalClass`

`pitchtools.InversionEquivalentDiatonicIntervalClassSegment`



class `abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalent`

New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class segment:

`InversionEquivalentDiatonicIntervalClassSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClassSegment.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`InversionEquivalentDiatonicIntervalClassSegment.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__len__()` $\iff len(x)$

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`InversionEquivalentDiatonicIntervalClassSegment.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.tuple`

`InversionEquivalentDiatonicIntervalClassSegment.__repr__()`

Note: Inherited from `pitchtools._IntervalSegment`

`InversionEquivalentDiatonicIntervalClassSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

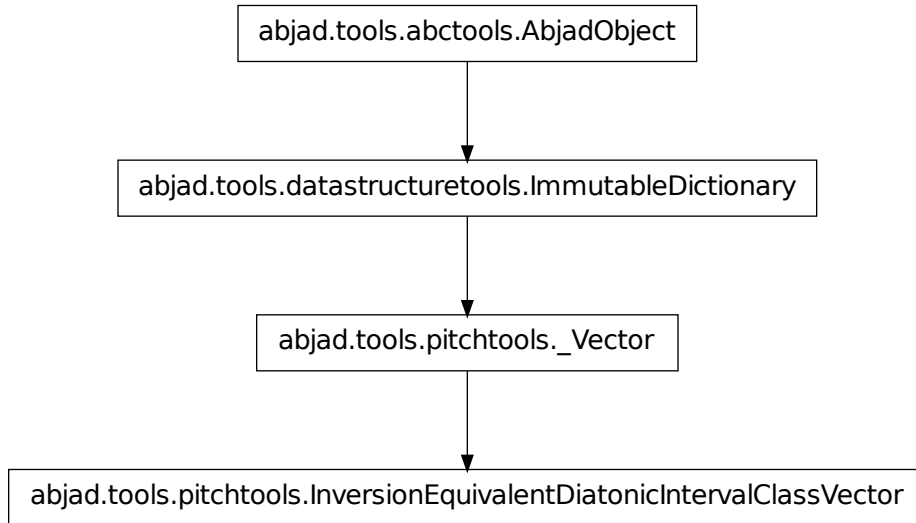
`InversionEquivalentDiatonicIntervalClassSegment.__setattr__()`
`x.__setattr__('name', value)` $\iff x.name = value$

Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClassSegment.__str__()`

Note: Inherited from `pitchtools._IntervalSegment`

pitchtools.InversionEquivalentDiatonicIntervalClassVector



class `abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector`.`InversionEquivalentDiatonicIntervalClassVector`
 New in version 2.0. Abjad model of inversion-equivalent diatonic interval-class vector:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8")
abjad> pitchtools.InversionEquivalentDiatonicIntervalClassVector(staff)
InversionEquivalentDiatonicIntervalClassVector(P1: 0, aug1: 0, m2: 1, M2: 3, aug2: 0, dim3: 0, m3: 0)
  
```

Inversion-equivalent diatonic interval-class vector are not quartertone-aware.

Inversion-equivalent diatonic interval-class vectors are immutable.

Methods

`InversionEquivalentDiatonicIntervalClassVector`.`clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector`.`copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector`.`get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.has_key(k)` → True if D has a key k,
else False

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.items()` → list of D's (key, value)
pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.iteritems()` → an iterator over the
(key, value) items of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.iterkeys()` → an iterator over the
keys of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.itervalues()` → an iterator over the
values of D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.pop(k[, d])` → v, remove specified
key and return the corresponding
value.

If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.popitem()` → (k, v), remove and re-
turn some (key, value) pair
as a

2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.setdefault(k[, d])` → D.get(k,d),
also set D[k]=d if k not
in D

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.update(E, **F) → None`. Update D from dict/iterable E and F. If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.values() → list of D's values`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewitems() → a set-like object providing a view on D's items`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewkeys() → a set-like object providing a view on D's keys`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.viewvalues() → an object providing a view on D's values`

Note: Inherited from `__builtin__.dict`

Special Methods

`InversionEquivalentDiatonicIntervalClassVector.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__contains__(k) → True if D has a key k, else False`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClassVector.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentDiatonicIntervalClassVector.__eq__(arg)`

`InversionEquivalentDiatonicIntervalClassVector.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__iter__()` `<==> iter(x)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__len__()` `<==> len(x)`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`InversionEquivalentDiatonicIntervalClassVector.__ne__(arg)`

`InversionEquivalentDiatonicIntervalClassVector.__repr__()`

`InversionEquivalentDiatonicIntervalClassVector.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

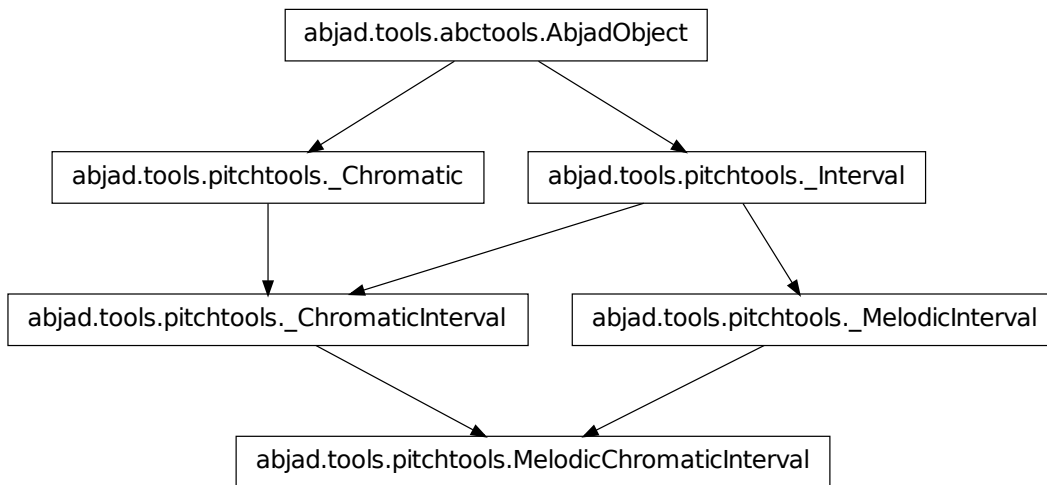
Note: Inherited from `__builtin__.object`

`InversionEquivalentDiatonicIntervalClassVector.__setitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`InversionEquivalentDiatonicIntervalClassVector.__str__()`

`pitchtools.MelodicChromaticInterval`



class `abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval`.**MelodicChromaticInterval**
 New in version 2.0. Abjad model of melodic chromatic interval:

```

abjad> pitchtools.MelodicChromaticInterval(-14)
MelodicChromaticInterval(-14)
  
```

Melodic chromatic intervals are immutable.

Read-only Properties

`MelodicChromaticInterval.cents`

Note: Inherited from `pitchtools._Interval`

`MelodicChromaticInterval.chromatic_interval_number`

Read-only chromatic interval number:

```

abjad> pitchtools.MelodicChromaticInterval(-14).chromatic_interval_number
-14
  
```

Return integer or float.

`MelodicChromaticInterval.direction_number`

Read-only numeric sign:

```
abjad> pitchtools.MelodicChromaticInterval(-14).direction_number
-1
```

Return integer.

`MelodicChromaticInterval.direction_string`

Note: Inherited from `pitchtools._MelodicInterval`

`MelodicChromaticInterval.harmonic_chromatic_interval`

Read-only harmonic chromatic interval:

```
abjad> pitchtools.MelodicChromaticInterval(-14).harmonic_chromatic_interval
HarmonicChromaticInterval(14)
```

Return harmonic chromatic interval.

`MelodicChromaticInterval.interval_class`

Note: Inherited from `pitchtools._Interval`

`MelodicChromaticInterval.melodic_chromatic_interval_class`

Read-only melodic chromatic interval-class:

```
abjad> pitchtools.MelodicChromaticInterval(-14).melodic_chromatic_interval_class
MelodicChromaticIntervalClass(-2)
```

Return melodic chromatic interval-class.

`MelodicChromaticInterval.number`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.semitones`

Note: Inherited from `pitchtools._ChromaticInterval`

Special Methods

`MelodicChromaticInterval.__abs__()`

`MelodicChromaticInterval.__add__(arg)`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MelodicChromaticInterval.__eq__(arg)`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.__float__()`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.__ge__(arg)`

`MelodicChromaticInterval.__gt__(arg)`

`MelodicChromaticInterval.__hash__()`

`MelodicChromaticInterval.__int__()`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.__le__(arg)`

`MelodicChromaticInterval.__lt__(arg)`

`MelodicChromaticInterval.__ne__(arg)`

Note: Inherited from `pitchtools._ChromaticInterval`

`MelodicChromaticInterval.__neg__()`

`MelodicChromaticInterval.__repr__()`

`MelodicChromaticInterval.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

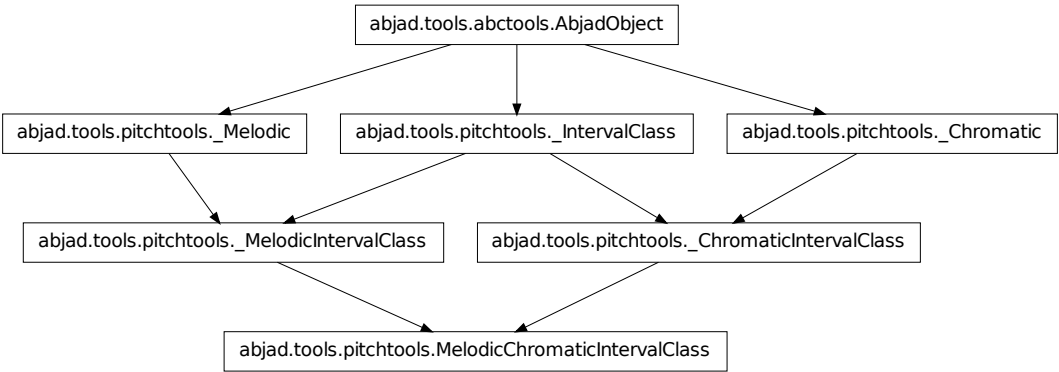
Note: Inherited from `__builtin__.object`

`MelodicChromaticInterval.__str__()`

`MelodicChromaticInterval.__sub__(arg)`

Note: Inherited from `pitchtools._ChromaticInterval`

pitchtools.MelodicChromaticIntervalClass



class `abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass`.**MelodicChromaticIntervalClass**
New in version 2.0. Abjad model of melodic chromatic interval-class:

```
abjad> pitchtools.MelodicChromaticIntervalClass(-14)
MelodicChromaticIntervalClass(-2)
```

Melodic chromatic interval-classes are immutable.

Read-only Properties

`MelodicChromaticIntervalClass.direction_number`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicChromaticIntervalClass.direction_symbol`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicChromaticIntervalClass.direction_word`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicChromaticIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

Special Methods

MelodicChromaticIntervalClass.__abs__()

Note: Inherited from `pitchtools._ChromaticIntervalClass`

MelodicChromaticIntervalClass.__delattr__()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

MelodicChromaticIntervalClass.__eq__(arg)

MelodicChromaticIntervalClass.__float__()

Note: Inherited from `pitchtools._ChromaticIntervalClass`

MelodicChromaticIntervalClass.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicChromaticIntervalClass.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

MelodicChromaticIntervalClass.__hash__()

Note: Inherited from `pitchtools._IntervalClass`

MelodicChromaticIntervalClass.__int__()

Note: Inherited from `pitchtools._ChromaticIntervalClass`

MelodicChromaticIntervalClass.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicChromaticIntervalClass.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicChromaticIntervalClass.__ne__(arg)`

`MelodicChromaticIntervalClass.__repr__()`

Note: Inherited from `pitchtools._IntervalClass`

`MelodicChromaticIntervalClass.__setattr__()`

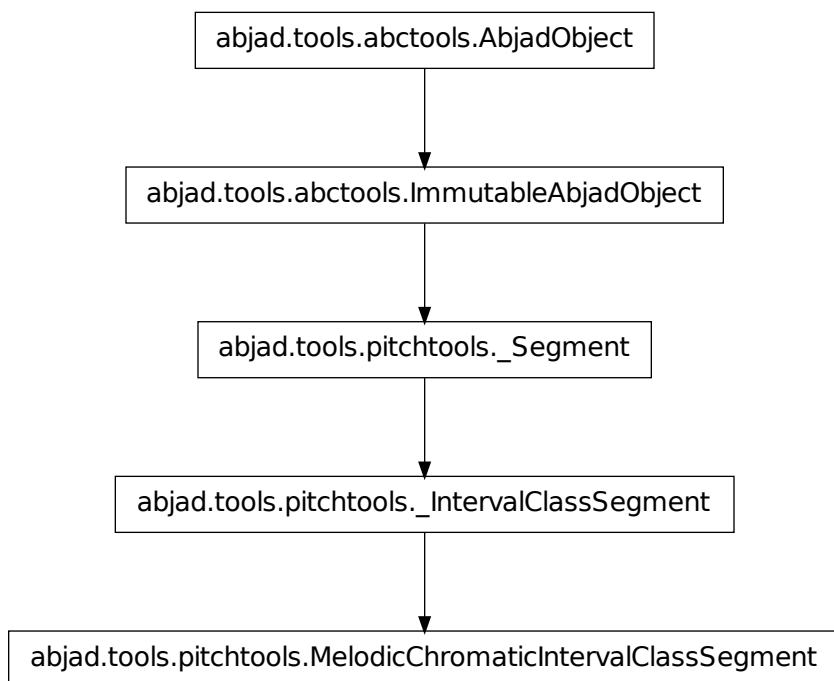
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MelodicChromaticIntervalClass.__str__()`

Note: Inherited from `pitchtools._IntervalClass`

`pitchtools.MelodicChromaticIntervalClassSegment`



class `abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment`

New in version 2.0. Abjad model of melodic chromatic interval-class segment:

```
abjad> pitchtools.MelodicChromaticIntervalClassSegment([-2, -14, 3, 5.5, 6.5])
MelodicChromaticIntervalClassSegment(-2, -2, +3, +5.5, +6.5)
```

Melodic chromatic interval-class segments are immutable.

Read-only Properties

`MelodicChromaticIntervalClassSegment.interval_class_numbers`

Note: Inherited from `pitchtools._IntervalClassSegment`

`MelodicChromaticIntervalClassSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalClassSegment`

Methods

`MelodicChromaticIntervalClassSegment.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalClassSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

Special Methods

`MelodicChromaticIntervalClassSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`MelodicChromaticIntervalClassSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalClassSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

MelodicChromaticIntervalClassSegment.__eq__()
 x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__ge__()
 x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__getitem__()
 x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__getslice__(start, stop)

Note: Inherited from `pitchtools._Segment`

MelodicChromaticIntervalClassSegment.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__len__() <==> len(x)

Note: Inherited from `__builtin__.tuple`

MelodicChromaticIntervalClassSegment.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalClassSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`MelodicChromaticIntervalClassSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalClassSegment.__repr__()`

Note: Inherited from `pitchtools._IntervalClassSegment`

`MelodicChromaticIntervalClassSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

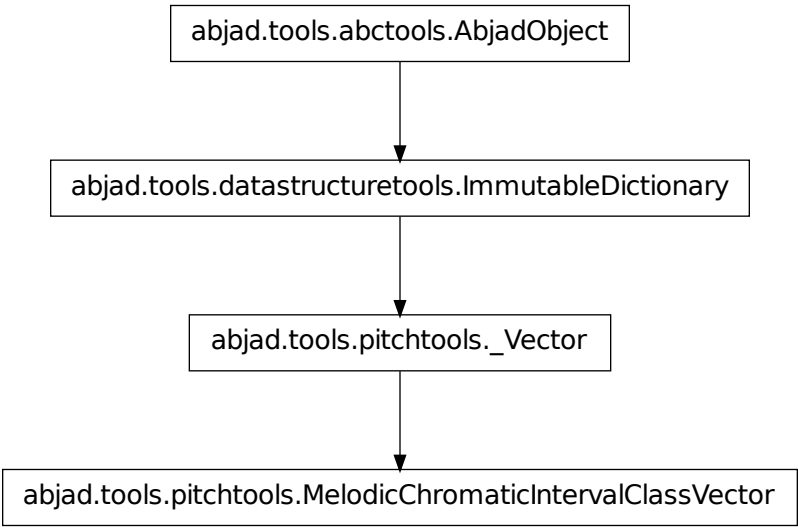
`MelodicChromaticIntervalClassSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MelodicChromaticIntervalClassSegment.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.MelodicChromaticIntervalClassVector



class `abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector`
New in version 2.0. Abjad model of melodic chromatic interval-class vector:

```
abjad> print pitchtools.MelodicChromaticIntervalClassVector([-2, -14, 3, 5.5, 6.5])
. | . . 1 . . . | . . . . .
  | . 2 . . . | . . . . .
  | . . . . 1 | 1 . . . .
  | . . . . . | . . . . .
```

Melodic chromatic interval-class vectors are immutable.

Methods

`MelodicChromaticIntervalClassVector.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.get(k, d)` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.update(E, **F) → None`. Update D from dict/iterable

If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`MelodicChromaticIntervalClassVector.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.__contains__(k) → True if D has a key k, else False`

Note: Inherited from `__builtin__.dict`

`MelodicChromaticIntervalClassVector.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MelodicChromaticIntervalClassVector.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

MelodicChromaticIntervalClassVector.__eq__()
 x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__ge__()
 x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__getitem__()
 x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__len__()
 MelodicChromaticIntervalClassVector.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__ne__()
 x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.dict`

MelodicChromaticIntervalClassVector.__repr__()
 MelodicChromaticIntervalClassVector.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

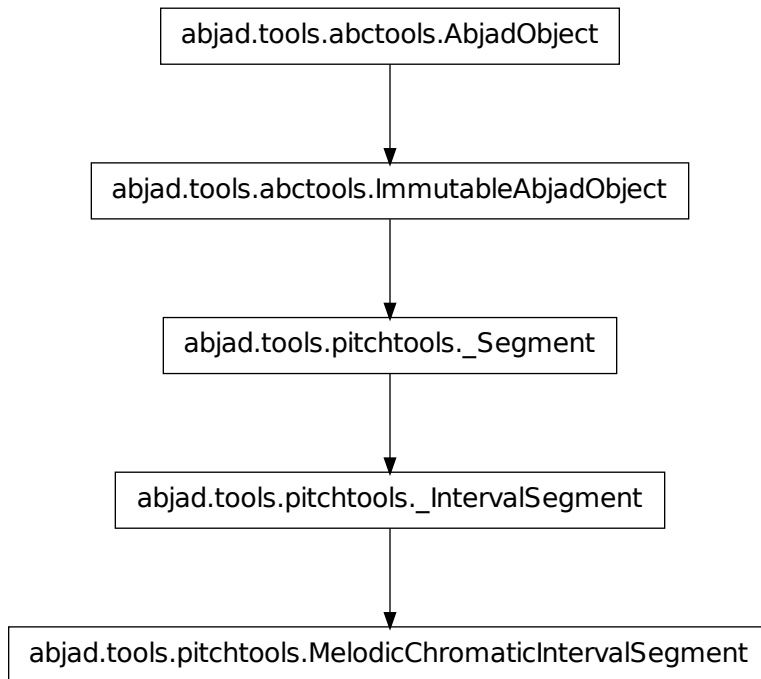
Note: Inherited from `__builtin__.object`

MelodicChromaticIntervalClassVector.__setitem__(*args)

Note: Inherited from `datastructuretools.ImmutableDictionary`

MelodicChromaticIntervalClassVector.__str__()

pitchtools.MelodicChromaticIntervalSegment



class `abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment`

New in version 2.0. Abjad model of melodic chromatic interval segment:

```

abjad> pitchtools.MelodicChromaticIntervalSegment([11, 13, 13.5, -2, 2.5])
MelodicChromaticIntervalSegment(+11, +13, +13.5, -2, +2.5)
  
```

Melodic chromatic interval segments are immutable.

Read-only Properties

`MelodicChromaticIntervalSegment.harmonic_chromatic_interval_segment`
`MelodicChromaticIntervalSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalSegment`

`MelodicChromaticIntervalSegment.intervals`

Note: Inherited from `pitchtools._IntervalSegment`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_class_segment`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_class_vector`

`MelodicChromaticIntervalSegment.melodic_chromatic_interval_numbers`

`MelodicChromaticIntervalSegment.slope`

The slope of a melodic interval segment is the sum of its intervals divided by its length:

```
abjad> pitchtools.MelodicChromaticIntervalSegment([1, 2]).slope
Fraction(3, 2)
```

Return fraction.

`MelodicChromaticIntervalSegment.spread`

The maximum harmonic interval spanned by any combination of the intervals within a harmonic chromatic interval segment:

```
abjad> pitchtools.MelodicChromaticIntervalSegment([1, 2, -3, 1, -2, 1]).spread
HarmonicChromaticInterval(4)
abjad> pitchtools.MelodicChromaticIntervalSegment([1, 1, 1, 2, -3, -2]).spread
HarmonicChromaticInterval(5)
```

Return harmonic chromatic interval.

Methods

`MelodicChromaticIntervalSegment.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.rotate(n)`

Note: Inherited from `pitchtools._IntervalSegment`

Special Methods

`MelodicChromaticIntervalSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

```
MelodicChromaticIntervalSegment.__contains__()
x.__contains__(y) <==> y in x
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__delattr__()
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
MelodicChromaticIntervalSegment.__eq__()
x.__eq__(y) <==> x==y
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__ge__()
x.__ge__(y) <==> x>=y
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__getitem__()
x.__getitem__(y) <==> x[y]
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__getslice__(start, stop)
```

Note: Inherited from `pitchtools._Segment`

```
MelodicChromaticIntervalSegment.__gt__()
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__hash__() <==> hash(x)
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__iter__() <==> iter(x)
```

Note: Inherited from `__builtin__.tuple`

```
MelodicChromaticIntervalSegment.__le__()
x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__len__()` $\iff \text{len}(x)$

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`MelodicChromaticIntervalSegment.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.tuple`

`MelodicChromaticIntervalSegment.__repr__()`

Note: Inherited from `pitchtools._IntervalSegment`

`MelodicChromaticIntervalSegment.__rmul__(n)`

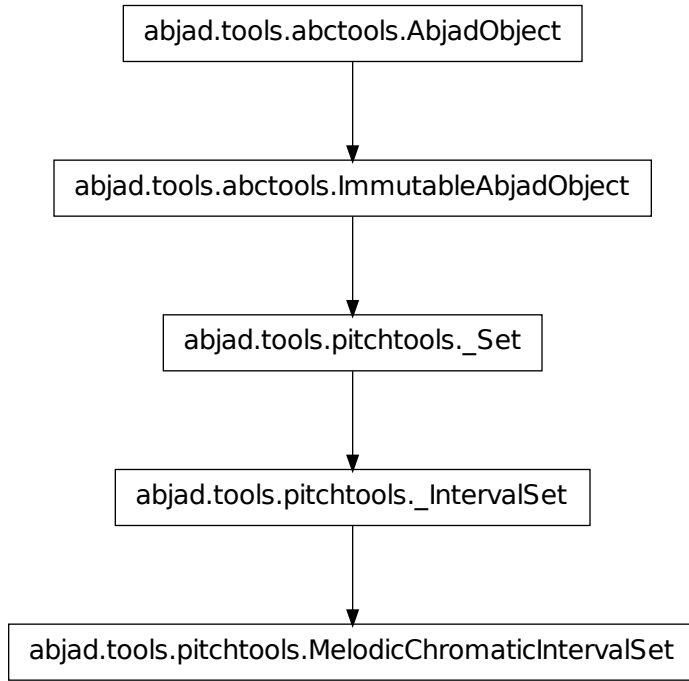
Note: Inherited from `pitchtools._Segment`

`MelodicChromaticIntervalSegment.__setattr__()`
`x.__setattr__('name', value)` $\iff x.name = value$

Note: Inherited from `__builtin__.object`

`MelodicChromaticIntervalSegment.__str__()`

Note: Inherited from `pitchtools._IntervalSegment`

pitchtools.MelodicChromaticIntervalSet

class `abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet`.**MelodicChromaticIntervalSet**

New in version 2.0. Abjad model of melodic chromatic interval set:

```
abjad> pitchtools.MelodicChromaticIntervalSet([11, 11, 13.5, 13.5])
MelodicChromaticIntervalSet(+11, +13.5)
```

Melodic chromatic interval sets are immutable.

Read-only Properties

`MelodicChromaticIntervalSet.harmonic_chromatic_interval_set`
`MelodicChromaticIntervalSet.melodic_chromatic_interval_numbers`
`MelodicChromaticIntervalSet.melodic_chromatic_intervals`

Methods

`MelodicChromaticIntervalSet.copy()`
 Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.difference()`
 Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

`MelodicChromaticIntervalSet.__and__()`

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__contains__()`
`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MelodicChromaticIntervalSet.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__iter__()` `<==> iter(x)`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.frozenset`

`MelodicChromaticIntervalSet.__len__()` `<==> len(x)`

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__ne__()
 x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__or__()
 x.__or__(y) <==> x|y

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__rand__()
 x.__rand__(y) <==> y&x

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__repr__()
 MelodicChromaticIntervalSet.__ror__()
 x.__ror__(y) <==> y|x

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__rsub__()
 x.__rsub__(y) <==> y-x

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__rxor__()
 x.__rxor__(y) <==> y^x

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

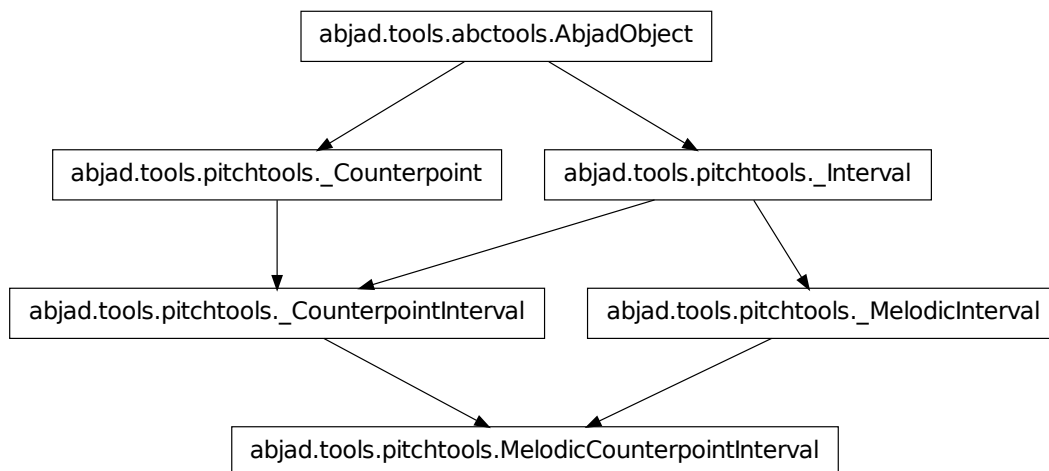
MelodicChromaticIntervalSet.__str__()
 MelodicChromaticIntervalSet.__sub__()
 x.__sub__(y) <==> x-y

Note: Inherited from `__builtin__.frozenset`

MelodicChromaticIntervalSet.__xor__()
 $x.\text{__xor__}(y) \iff x^y$

Note: Inherited from `__builtin__.frozenset`

pitchtools.MelodicCounterpointInterval



class `abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval`.**MelodicCounterpointInterval**
 New in version 2.0. Abjad model of melodic counterpoint interval:

```
abjad> pitchtools.MelodicCounterpointInterval(-9)
MelodicCounterpointInterval(-9)
```

Melodic counterpoint intervals are immutable.

Read-only Properties

`MelodicCounterpointInterval.cents`

Note: Inherited from `pitchtools._Interval`

`MelodicCounterpointInterval.direction_number`

`MelodicCounterpointInterval.direction_string`

Note: Inherited from `pitchtools._MelodicInterval`

`MelodicCounterpointInterval.interval_class`

Note: Inherited from `pitchtools._Interval`

`MelodicCounterpointInterval.melodic_counterpoint_interval_class`

`MelodicCounterpointInterval.number`

Note: Inherited from `pitchtools._CounterpointInterval`

`MelodicCounterpointInterval.semitones`

Note: Inherited from `pitchtools._CounterpointInterval`

Special Methods

`MelodicCounterpointInterval.__abs__()`

Note: Inherited from `pitchtools._CounterpointInterval`

`MelodicCounterpointInterval.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MelodicCounterpointInterval.__eq__(arg)`

Note: Inherited from `pitchtools._MelodicInterval`

`MelodicCounterpointInterval.__float__()`

Note: Inherited from `pitchtools._CounterpointInterval`

`MelodicCounterpointInterval.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__hash__()`

Note: Inherited from `pitchtools._Interval`

`MelodicCounterpointInterval.__int__()`

Note: Inherited from `pitchtools._CounterpointInterval`

`MelodicCounterpointInterval.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicCounterpointInterval.__ne__(arg)`

Note: Inherited from `pitchtools._MelodicInterval`

`MelodicCounterpointInterval.__neg__()`

Note: Inherited from `pitchtools._MelodicInterval`

`MelodicCounterpointInterval.__repr__()`

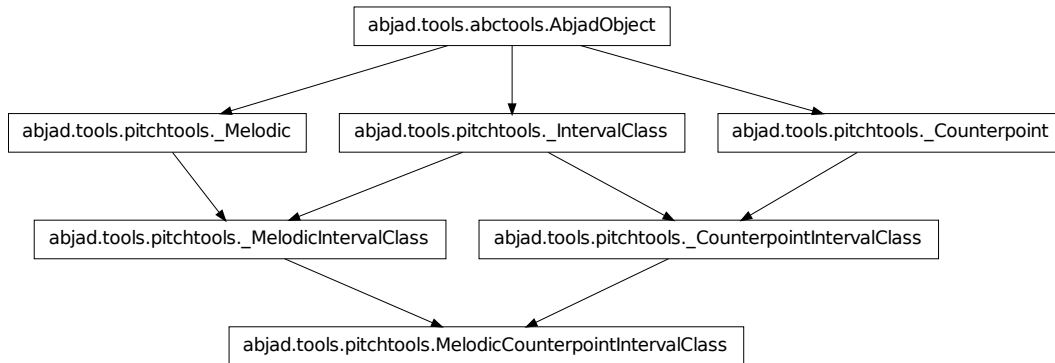
Note: Inherited from `pitchtools._Interval`

`MelodicCounterpointInterval.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MelodicCounterpointInterval.__str__()`

pitchtools.MelodicCounterpointIntervalClass



class `abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass`
 New in version 2.0. Abjad model of melodic counterpoint interval-class:

```
abjad> pitchtools.MelodicCounterpointIntervalClass(-9)
MelodicCounterpointIntervalClass(-2)
```

Melodic counterpoint interval-classes are immutable.

Read-only Properties

`MelodicCounterpointIntervalClass.direction_number`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicCounterpointIntervalClass.direction_symbol`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicCounterpointIntervalClass.direction_word`

Note: Inherited from `pitchtools._MelodicIntervalClass`

`MelodicCounterpointIntervalClass.number`

Note: Inherited from `pitchtools._IntervalClass`

Special Methods

MelodicCounterpointIntervalClass.__abs__()

Note: Inherited from `pitchtools._CounterpointIntervalClass`

MelodicCounterpointIntervalClass.__delattr__()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

MelodicCounterpointIntervalClass.__eq__(arg)

MelodicCounterpointIntervalClass.__float__()

Note: Inherited from `pitchtools._CounterpointIntervalClass`

MelodicCounterpointIntervalClass.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicCounterpointIntervalClass.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

MelodicCounterpointIntervalClass.__hash__()

Note: Inherited from `pitchtools._IntervalClass`

MelodicCounterpointIntervalClass.__int__()

Note: Inherited from `pitchtools._CounterpointIntervalClass`

MelodicCounterpointIntervalClass.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicCounterpointIntervalClass.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicCounterpointIntervalClass.__ne__(arg)`

`MelodicCounterpointIntervalClass.__repr__()`

Note: Inherited from `pitchtools._IntervalClass`

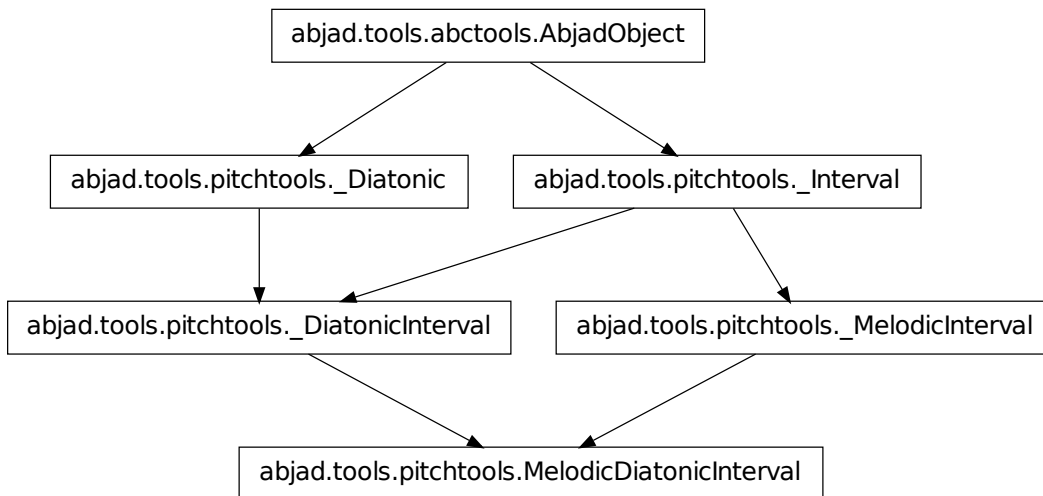
`MelodicCounterpointIntervalClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MelodicCounterpointIntervalClass.__str__()`

Note: Inherited from `pitchtools._IntervalClass`

`pitchtools.MelodicDiatonicInterval`



class `abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval` **MelodicDiatonicInterval**

New in version 2.0. Abjad model of melodic diatonic interval:

```

abjad> pitchtools.MelodicDiatonicInterval('+M9')
MelodicDiatonicInterval('+M9')
```

Melodic diatonic intervals are immutable.

Read-only Properties

`MelodicDiatonicInterval.cents`

Note: Inherited from `pitchtools._Interval`

`MelodicDiatonicInterval.diatonic_interval_class`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.direction_number`

`MelodicDiatonicInterval.direction_string`

`MelodicDiatonicInterval.harmonic_chromatic_interval`

`MelodicDiatonicInterval.harmonic_counterpoint_interval`

`MelodicDiatonicInterval.harmonic_diatonic_interval`

`MelodicDiatonicInterval.interval_class`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.interval_string`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.inversion_equivalent_chromatic_interval_class`

`MelodicDiatonicInterval.melodic_chromatic_interval`

`MelodicDiatonicInterval.melodic_counterpoint_interval`

`MelodicDiatonicInterval.melodic_diatonic_interval_class`

`MelodicDiatonicInterval.number`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.quality_string`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.semitones`

`MelodicDiatonicInterval.staff_spaces`

Special Methods

`MelodicDiatonicInterval.__abs__()`

`MelodicDiatonicInterval.__add__(arg)`

`MelodicDiatonicInterval.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MelodicDiatonicInterval.__eq__(arg)`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.__float__()`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MelodicDiatonicInterval.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MelodicDiatonicInterval.__hash__()`

Note: Inherited from `pitchtools._Interval`

`MelodicDiatonicInterval.__int__()`

Note: Inherited from `pitchtools._DiatonicInterval`

`MelodicDiatonicInterval.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicInterval.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicInterval.__mul__(arg)
MelodicDiatonicInterval.__ne__(arg)

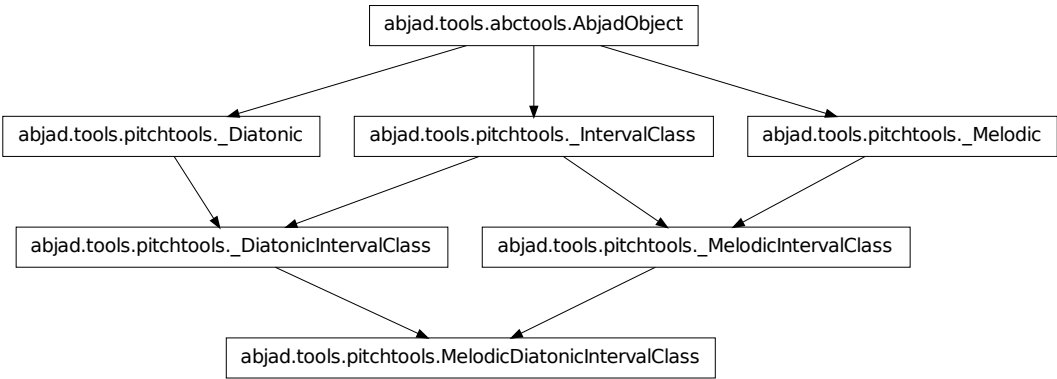
Note: Inherited from `pitchtools._DiatonicInterval`

MelodicDiatonicInterval.__neg__()
MelodicDiatonicInterval.__repr__()
MelodicDiatonicInterval.__rmul__(arg)
MelodicDiatonicInterval.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

MelodicDiatonicInterval.__str__()
MelodicDiatonicInterval.__sub__(arg)

pitchtools.MelodicDiatonicIntervalClass



class `abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass`. **Melodi**
New in version 2.0. Abjad model of melodic diatonic interval-class:

abjad> `pitchtools.MelodicDiatonicIntervalClass('-M9')`
`MelodicDiatonicIntervalClass('-M2')`

Melodic diatonic interval-classes are immutable.

Read-only Properties

MelodicDiatonicIntervalClass.**direction_number**

MelodicDiatonicIntervalClass.**direction_symbol**

MelodicDiatonicIntervalClass.**direction_word**

MelodicDiatonicIntervalClass.**number**

Note: Inherited from `pitchtools._IntervalClass`

MelodicDiatonicIntervalClass.**quality_string**

Note: Inherited from `pitchtools._DiatonicIntervalClass`

Special Methods

MelodicDiatonicIntervalClass.**__abs__**()

Note: Inherited from `pitchtools._DiatonicIntervalClass`

MelodicDiatonicIntervalClass.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

MelodicDiatonicIntervalClass.**__eq__**(*arg*)

MelodicDiatonicIntervalClass.**__float__**()

Note: Inherited from `pitchtools._DiatonicIntervalClass`

MelodicDiatonicIntervalClass.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicIntervalClass.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicIntervalClass.**__hash__**()

MelodicDiatonicIntervalClass.__int__()

Note: Inherited from `pitchtools._DiatonicIntervalClass`

MelodicDiatonicIntervalClass.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicIntervalClass.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MelodicDiatonicIntervalClass.__ne__(arg)

MelodicDiatonicIntervalClass.__repr__()

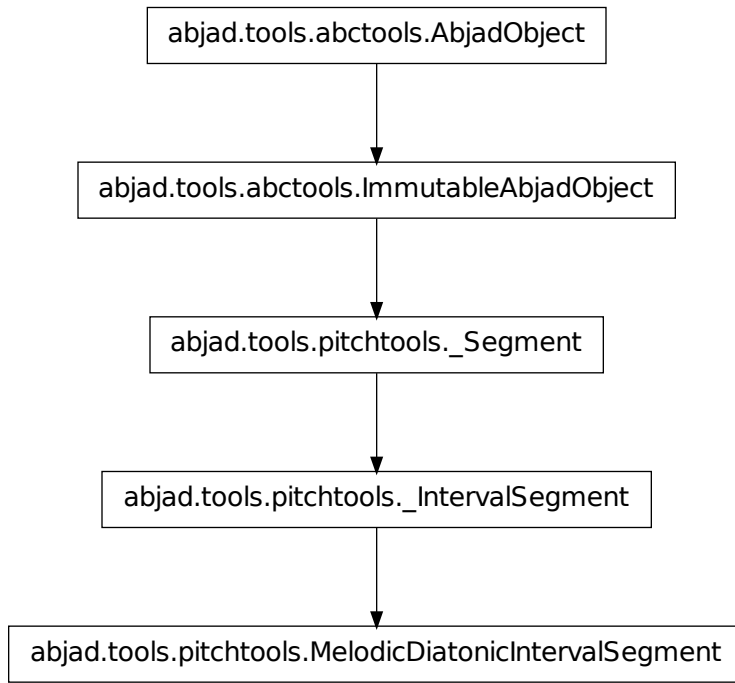
Note: Inherited from `pitchtools._DiatonicIntervalClass`

MelodicDiatonicIntervalClass.__setattr__()
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

MelodicDiatonicIntervalClass.__str__()

pitchtools.MelodicDiatonicIntervalSegment



class `abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment`. **Me**

New in version 2.0. Abjad model of melodic diatonic interval segment:

```

abjad> pitchtools.MelodicDiatonicIntervalSegment('M2 M9 -m3 -P4')
MelodicDiatonicIntervalSegment('+M2 +M9 -m3 -P4')

```

Melodic diatonic interval segments are immutable.

Read-only Properties

`MelodicDiatonicIntervalSegment.harmonic_chromatic_interval_segment`

`MelodicDiatonicIntervalSegment.harmonic_diatonic_interval_segment`

`MelodicDiatonicIntervalSegment.interval_classes`

Note: Inherited from `pitchtools._IntervalSegment`

`MelodicDiatonicIntervalSegment.intervals`

Note: Inherited from `pitchtools._IntervalSegment`

MelodicDiatonicIntervalSegment.**melodic_chromatic_interval_segment**

Methods

MelodicDiatonicIntervalSegment.**count**(*value*) → integer – return number of occurrences of *value*

Note: Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of *value*.

Raises ValueError if the value is not present.

Note: Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**rotate**(*n*)

Note: Inherited from `pitchtools._IntervalSegment`

Special Methods

MelodicDiatonicIntervalSegment.**__add__**(*arg*)

Note: Inherited from `pitchtools._Segment`

MelodicDiatonicIntervalSegment.**__contains__**()
x.**__contains__**(*y*) <==> *y* in *x*

Note: Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__delattr__**()
x.**__delattr__**('name') <==> del *x*.name

Note: Inherited from `__builtin__.object`

MelodicDiatonicIntervalSegment.**__eq__**()
x.**__eq__**(*y*) <==> *x*==*y*

Note: Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__ge__**()
x.**__ge__**(*y*) <==> *x*>=*y*

Note: Inherited from `__builtin__.tuple`

MelodicDiatonicIntervalSegment.**__getitem__**()
x.**__getitem__**(*y*) <==> *x*[*y*]

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`MelodicDiatonicIntervalSegment.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__len__() <==> len(x)`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`MelodicDiatonicIntervalSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`MelodicDiatonicIntervalSegment.__repr__()`

`MelodicDiatonicIntervalSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

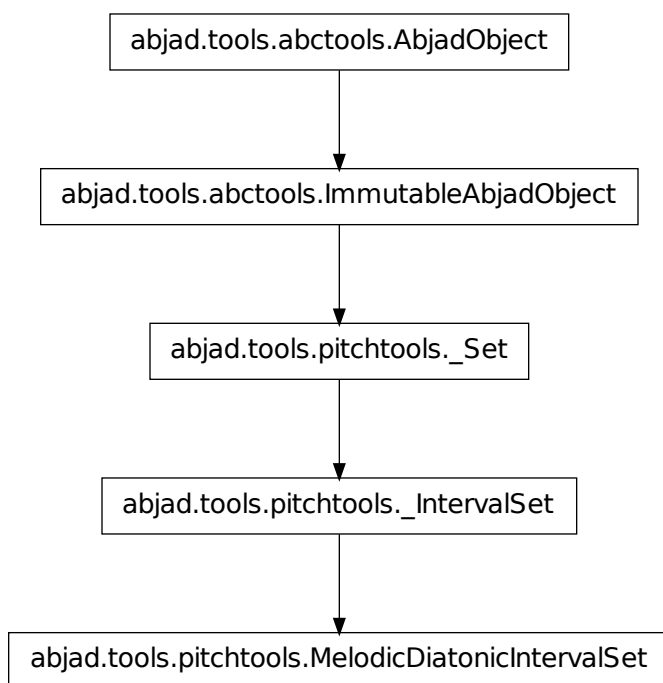
`MelodicDiatonicIntervalSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MelodicDiatonicIntervalSegment.__str__()`

Note: Inherited from `pitchtools._IntervalSegment`

`pitchtools.MelodicDiatonicIntervalSet`



class `abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet`. **MelodicDiatonicIntervalSet**

New in version 2.0. Abjad model of melodic diatonic interval set:

```
abjad> pitchtools.MelodicDiatonicIntervalSet('M2 M2 -m3 -P4')
MelodicDiatonicIntervalSet('-P4 -m3 +M2')
```

Melodic diatonic interval sets are immutable.

Read-only Properties

```
MelodicDiatonicIntervalSet.harmonic_chromatic_interval_set
MelodicDiatonicIntervalSet.harmonic_diatonic_interval_set
MelodicDiatonicIntervalSet.melodic_chromatic_interval_set
MelodicDiatonicIntervalSet.melodic_diatonic_interval_numbers
MelodicDiatonicIntervalSet.melodic_diatonic_intervals
```

Methods

MelodicDiatonicIntervalSet.**copy**()
Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**difference**()
Return the difference of two or more sets as a new set.
(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**intersection**()
Return the intersection of two or more sets as a new set.
(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**isdisjoint**()
Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**issubset**()
Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**issuperset**()
Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**symmetric_difference**()

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**union**()

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

MelodicDiatonicIntervalSet.**__and__**()

$x._\text{and}_(y) \iff x \& y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**__cmp__**(y) $\iff \text{cmp}(x, y)$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**__contains__**()

$x._\text{contains}_(y) \iff y \text{ in } x.$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**__delattr__**()

$x._\text{delattr}_('name') \iff \text{del } x.\text{name}$

Note: Inherited from `__builtin__.object`

MelodicDiatonicIntervalSet.**__eq__**()

$x._\text{eq}_(y) \iff x == y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**__ge__**()

$x._\text{ge}_(y) \iff x \geq y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.**__gt__**()

$x._\text{gt}_(y) \iff x > y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__hash__() $\iff hash(x)$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__iter__() $\iff iter(x)$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__le__()
 `x.__le__(y)` $\iff x \leq y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__len__() $\iff len(x)$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__lt__()
 `x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__ne__()
 `x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__or__()
 `x.__or__(y)` $\iff x \cup y$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__rand__()
 `x.__rand__(y)` $\iff y \& x$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__repr__()
 `x.__repr__()`

MelodicDiatonicIntervalSet.__ror__()
 `x.__ror__(y)` $\iff y \cup x$

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__rsub__()
x.__rsub__(y) <==> y-x

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__rxor__()
x.__rxor__(y) <==> y^x

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

MelodicDiatonicIntervalSet.__str__()

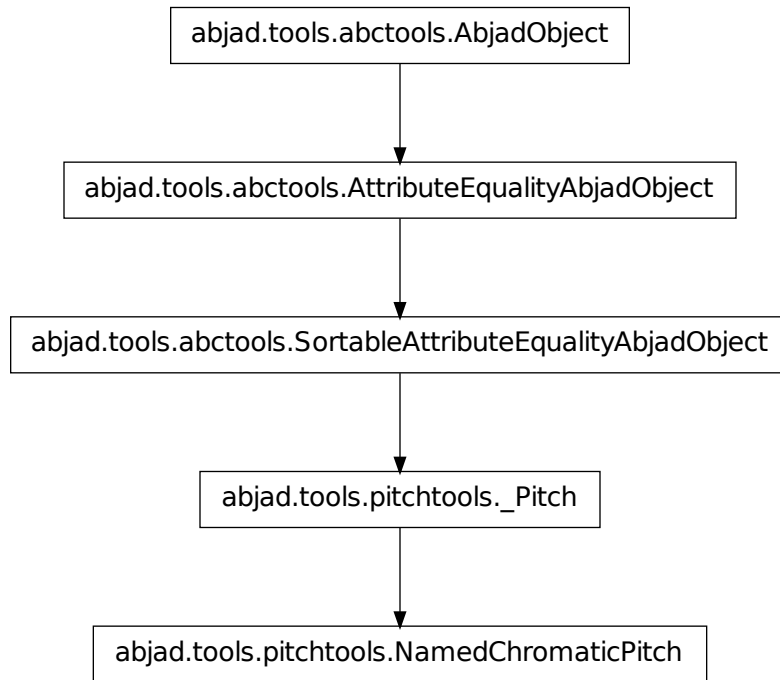
MelodicDiatonicIntervalSet.__sub__()
x.__sub__(y) <==> x-y

Note: Inherited from `__builtin__.frozenset`

MelodicDiatonicIntervalSet.__xor__()
x.__xor__(y) <==> x^y

Note: Inherited from `__builtin__.frozenset`

pitchtools.NamedChromaticPitch



class `abjad.tools.pitchtools.NamedChromaticPitch`.`NamedChromaticPitch`.**NamedChromaticPitch** (*args, **kwargs)

New in version 1.1. Abjad model of named chromatic pitch:

```
abjad> pitchtools.NamedChromaticPitch("cs'")
NamedChromaticPitch("cs'")
```

Named chromatic pitches are immutable.

Read-only Properties

`NamedChromaticPitch`.**chromatic_pitch_class_name**

Read-only chromatic pitch-class name:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.chromatic_pitch_class_name
'cs'
```

Return string.

`NamedChromaticPitch`.**chromatic_pitch_class_number**

Read-only chromatic pitch-class number:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.chromatic_pitch_class_number
1
```

Return integer or float.

`NamedChromaticPitch.chromatic_pitch_name`

Read-only chromatic pitch name:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_chromatic_pitch.chromatic_pitch_name
"cs' "
```

Return string.

`NamedChromaticPitch.chromatic_pitch_number`

Read-only chromatic pitch-class number:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_chromatic_pitch.chromatic_pitch_number
13
```

Return integer or float.

`NamedChromaticPitch.deviation_in_cents`

Read-only deviation of named chromatic pitch in cents:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_chromatic_pitch.deviation_in_cents is None
True
```

Return integer or none.

`NamedChromaticPitch.diatonic_pitch_class_name`

Read-only diatonic pitch-class name:

```
abjad> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_diatonic_pitch.diatonic_pitch_class_name
'c'
```

Return string.

`NamedChromaticPitch.diatonic_pitch_class_number`

Read-only diatonic pitch-class number:

```
abjad> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_diatonic_pitch.diatonic_pitch_class_number
0
```

Return integer.

`NamedChromaticPitch.diatonic_pitch_name`

Read-only diatonic pitch name:

```
abjad> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_diatonic_pitch.diatonic_pitch_name
"c' "
```

Return string.

`NamedChromaticPitch.diatonic_pitch_number`

Read-only diatonic pitch number:

```
abjad> named_diatonic_pitch = pitchtools.NamedChromaticPitch("cs' ")
abjad> named_diatonic_pitch.diatonic_pitch_number
7
```

Return integer.

NamedChromaticPitch.format

Read-only LilyPond input format of named chromatic pitch:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.format
"cs' "
```

Return string.

NamedChromaticPitch.named_chromatic_pitch_class

Read-only named pitch-class:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.named_chromatic_pitch_class
NamedChromaticPitchClass('cs')
```

Return named chromatic pitch-class.

NamedChromaticPitch.named_diatonic_pitch

Read-only named diatonic pitch:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.named_diatonic_pitch
NamedDiatonicPitch("c'")
```

Return named diatonic pitch.

NamedChromaticPitch.named_diatonic_pitch_class

Read-only named diatonic pitch-class:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

NamedChromaticPitch.numbered_chromatic_pitch

Read-only numbered chromatic pitch from named chromatic pitch:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

NamedChromaticPitch.numbered_chromatic_pitch_class

Read-only numbered pitch-class:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

NamedChromaticPitch.numbered_diatonic_pitch

Read-only numbered diatonic pitch:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

Return numbered diatonic pitch.

NamedChromaticPitch.numbered_diatonic_pitch_class

Read-only numbered diatonic pitch:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

NamedChromaticPitch.octave_number

Read-only integer octave number:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.octave_number
5
```

Return integer.

NamedChromaticPitch.pitch_class_octave_label

Read-only pitch-class / octave label:

```
abjad> named_chromatic_pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> named_chromatic_pitch.pitch_class_octave_label
'C#5'
```

Return string.

Special Methods

NamedChromaticPitch.__abs__()

NamedChromaticPitch.__add__(*melodic_interval*)

New in version 2.0.

NamedChromaticPitch.__delattr__()

x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

NamedChromaticPitch.__eq__(*arg*)

NamedChromaticPitch.__float__()

NamedChromaticPitch.__ge__(*arg*)

NamedChromaticPitch.__gt__(*arg*)

NamedChromaticPitch.__hash__()

NamedChromaticPitch.__int__()

NamedChromaticPitch.__le__(*arg*)

NamedChromaticPitch.__lt__(*arg*)

NamedChromaticPitch.__ne__(*arg*)

NamedChromaticPitch.__repr__()

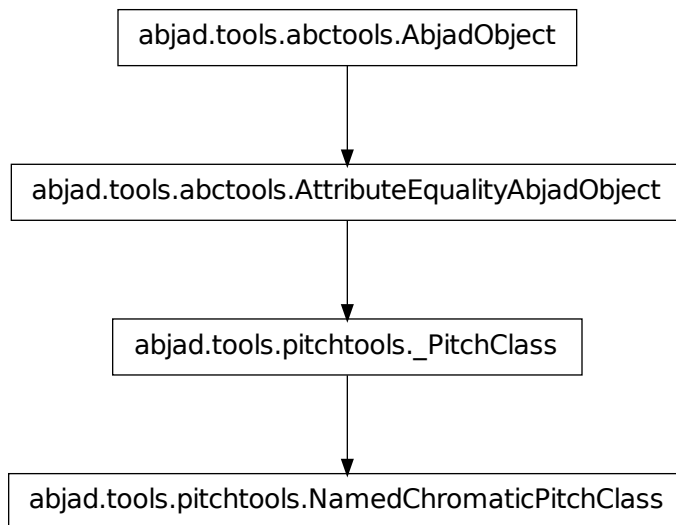
NamedChromaticPitch.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

```
NamedChromaticPitch.__str__()
NamedChromaticPitch.__sub__(arg)
```

`pitchtools.NamedChromaticPitchClass`



class `abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass`

New in version 2.0. Abjad model of named chromatic pitch-class:

```
abjad> pitchtools.NamedChromaticPitchClass('cs')
NamedChromaticPitchClass('cs')
```

Named chromatic pitch-classes are immutable.

Read-only Properties

`NamedChromaticPitchClass.numbered_chromatic_pitch_class`

Read-only numbered chromatic pitch-class:

```
abjad> named_chromatic_pitch_class = pitchtools.NamedChromaticPitchClass('cs')
abjad> named_chromatic_pitch_class.numbered_chromatic_pitch_class
NumberedChromaticPitchClass(1)
```

Return numbered chromatic pitch-class.

Methods

`NamedChromaticPitchClass.apply_accidental(accidental)`

Apply *accidental*:


```
abjad> named_chromatic_pitch_class = pitchtools.NamedChromaticPitchClass('cs')
abjad> named_chromatic_pitch_class.apply_accidental('qs')
NamedChromaticPitchClass('ctqs')
```

Return named chromatic pitch-class.

NamedChromaticPitchClass.**transpose**(*melodic_diatonic_interval*)

Transpose named chromatic pitch-class by *melodic_diatonic_interval*:

```
abjad> named_chromatic_pitch_class = pitchtools.NamedChromaticPitchClass('cs')
abjad> named_chromatic_pitch_class.transpose(pitchtools.MelodicDiatonicInterval('major', 2))
NamedChromaticPitchClass('ds')
```

Return named chromatic pitch-class.

Special Methods

NamedChromaticPitchClass.**__abs__**()

NamedChromaticPitchClass.**__add__**(*melodic_diatonic_interval*)

NamedChromaticPitchClass.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

NamedChromaticPitchClass.**__eq__**(*arg*)

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

NamedChromaticPitchClass.**__float__**()

NamedChromaticPitchClass.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

NamedChromaticPitchClass.**__hash__**()

Note: Inherited from `pitchtools._PitchClass`

NamedChromaticPitchClass.**__int__**()

`NamedChromaticPitchClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NamedChromaticPitchClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NamedChromaticPitchClass.__ne__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedChromaticPitchClass.__repr__()`

`NamedChromaticPitchClass.__setattr__()`

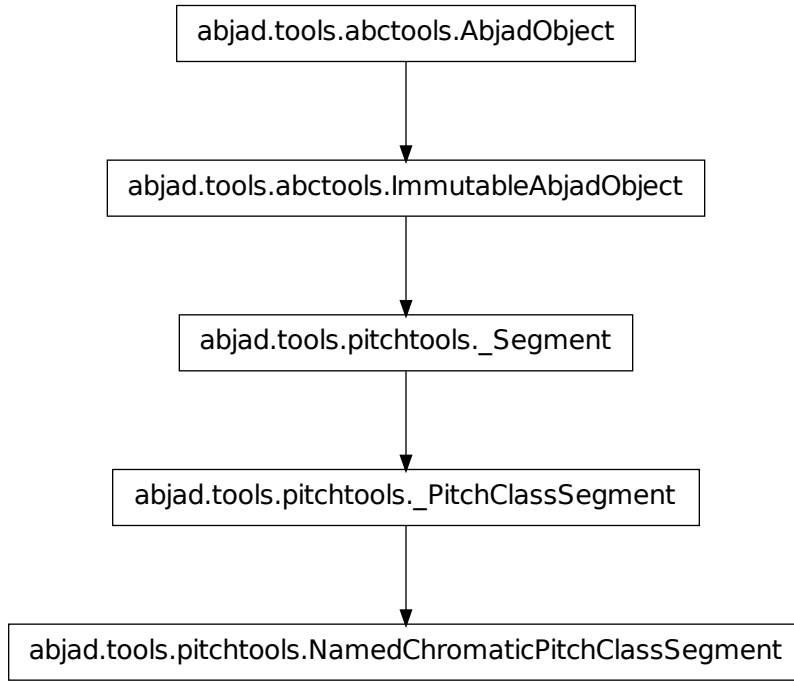
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchClass.__str__()`

`NamedChromaticPitchClass.__sub__(arg)`

pitchtools.NamedChromaticPitchClassSegment



class `abjad.tools.pitchtools.NamedChromaticPitchClassSegment`.`NamedChromaticPitchClassSegment`.

New in version 2.0. Abjad model of named chromatic pitch-class segment:

```
abjad> pitchtools.NamedChromaticPitchClassSegment(['gs', 'a', 'as', 'c', 'cs'])
NamedChromaticPitchClassSegment(['gs', 'a', 'as', 'c', 'cs'])
```

Named chromatic pitch-class segments are immutable.

Read-only Properties

`NamedChromaticPitchClassSegment.inversion_equivalent_diatonic_interval_class_segment`

`NamedChromaticPitchClassSegment.named_chromatic_pitch_class_set`

`NamedChromaticPitchClassSegment.named_chromatic_pitch_classes`

`NamedChromaticPitchClassSegment.numbered_chromatic_pitch_class_segment`

`NamedChromaticPitchClassSegment.numbered_chromatic_pitch_class_set`

`NamedChromaticPitchClassSegment.numbered_chromatic_pitch_classes`

Methods

`NamedChromaticPitchClassSegment.count` (*value*) → integer – return number of occurrences of *value*

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.is_equivalent_under_transposition(arg)`

`NamedChromaticPitchClassSegment.retrograde()`

`NamedChromaticPitchClassSegment.rotate(n)`

`NamedChromaticPitchClassSegment.transpose(melodic_diatonic_interval)`

Special Methods

`NamedChromaticPitchClassSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchClassSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchClassSegment.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchClassSegment.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__hash__()` <==> `hash(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchClassSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchClassSegment.__repr__()`

`NamedChromaticPitchClassSegment.__rmul__(n)`

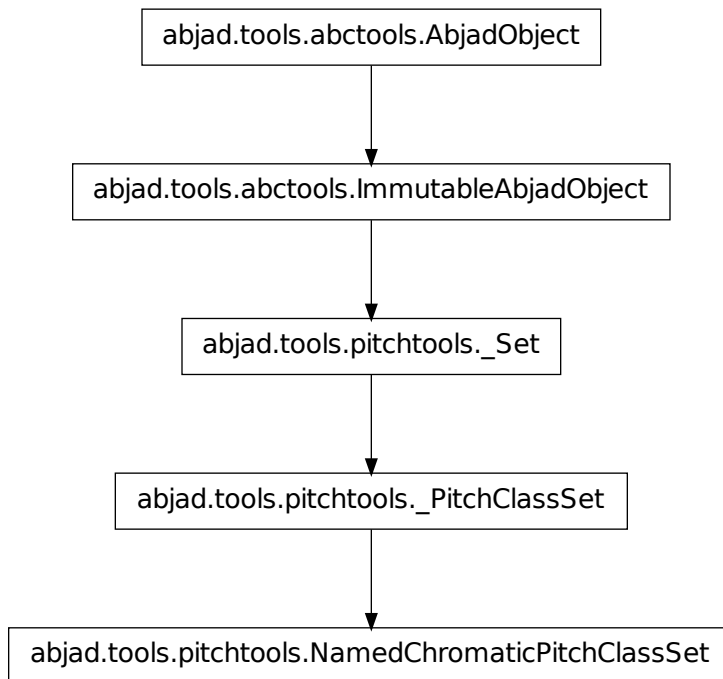
Note: Inherited from `pitchtools._Segment`

```
NamedChromaticPitchClassSegment.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
NamedChromaticPitchClassSegment.__str__()
```

`pitchtools.NamedChromaticPitchClassSet`



class `abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet` **NamedChromaticPitchClassSet**

New in version 2.0. Abjad model of a named chromatic pitch-class set:

```
abjad> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(['gs', 'g', 'as'])

abjad> named_chromatic_pitch_class_set
NamedChromaticPitchClassSet(['as', 'c', 'cs', 'g', 'gs'])

abjad> print named_chromatic_pitch_class_set
{as, c, cs, g, gs}
```

Named chromatic pitch-class sets are immutable.

Read-only Properties

`NamedChromaticPitchClassSet.inversion_equivalent_diatonic_interval_class_vector`
`NamedChromaticPitchClassSet.named_chromatic_pitch_classes`

Read-only named chromatic pitch-classes:

```
abjad> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(['gs', 'g', 'as', 'a', 'f', 'e', 'd', 'c', 'b', 'ab', 'bb', 'cb', 'db', 'eb', 'fb', 'gb', 'as', 'bs', 'cs', 'ds', 'es', 'fs', 'gs'])
abjad> named_chromatic_pitch_class_set.named_chromatic_pitch_classes # doctest: +SKIP
(NamedChromaticPitchClass('c'), NamedChromaticPitchClass('cs'), NamedChromaticPitchClass('g'), N
```

Return tuple.

`NamedChromaticPitchClassSet.numbered_chromatic_pitch_class_set`

Methods

`NamedChromaticPitchClassSet.copy()`

Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.order_by(npc_seg)`

`NamedChromaticPitchClassSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.transpose` (*melodic_diatonic_interval*)

Transpose all npcs in self by melodic diatonic interval.

`NamedChromaticPitchClassSet.union` ()

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

`NamedChromaticPitchClassSet.__and__` ()

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__cmp__` (y) <==> *cmp*(x, y)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__contains__` ()

`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__delattr__` ()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchClassSet.__eq__` (arg)

`NamedChromaticPitchClassSet.__ge__` ()

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__gt__` ()

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__hash__` ()

`NamedChromaticPitchClassSet.__iter__` () <==> *iter*(x)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__len__()` \leq `len(x)`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__ne__(arg)`
`NamedChromaticPitchClassSet.__or__()`
`x.__or__(y) <==> x|y`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__rand__()`
`x.__rand__(y) <==> y&x`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__repr__()`
`NamedChromaticPitchClassSet.__ror__()`
`x.__ror__(y) <==> y|x`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__rsub__()`
`x.__rsub__(y) <==> y-x`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__rxor__()`
`x.__rxor__(y) <==> y^x`

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchClassSet.__str__()`

`NamedChromaticPitchClassSet.__sub__()`

`x.__sub__(y) <==> x-y`

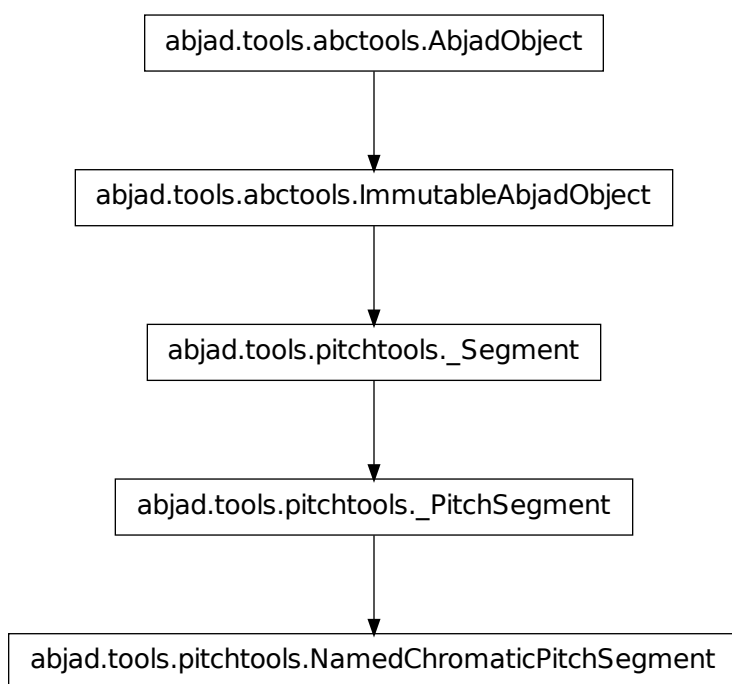
Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchClassSet.__xor__()`

`x.__xor__(y) <==> x^y`

Note: Inherited from `__builtin__.frozenset`

`pitchtools.NamedChromaticPitchSegment`



class `abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment` **NamedChrom**

New in version 2.0. Abjad model of a named chromatic pitch segment:

```

abjad> pitchtools.NamedChromaticPitchSegment(['bf', 'bqf', "fs'", "g'", 'bqf', "g'"])
NamedChromaticPitchSegment("bf bqf fs' g' bqf g'")
  
```

Named chromtic pitch segments are immutable.

Read-only Properties

`NamedChromaticPitchSegment.chromatic_pitch_numbers`
`NamedChromaticPitchSegment.harmonic_chromatic_interval_class_segment`
`NamedChromaticPitchSegment.harmonic_chromatic_interval_segment`
`NamedChromaticPitchSegment.harmonic_diatonic_interval_class_segment`
`NamedChromaticPitchSegment.harmonic_diatonic_interval_segment`
`NamedChromaticPitchSegment.inflection_point_count`
`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_segment`
`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_set`
`NamedChromaticPitchSegment.inversion_equivalent_chromatic_interval_class_vector`
`NamedChromaticPitchSegment.local_maxima`
`NamedChromaticPitchSegment.local_minima`
`NamedChromaticPitchSegment.melodic_chromatic_interval_class_segment`
`NamedChromaticPitchSegment.melodic_chromatic_interval_segment`
`NamedChromaticPitchSegment.melodic_diatonic_interval_class_segment`
`NamedChromaticPitchSegment.melodic_diatonic_interval_segment`
`NamedChromaticPitchSegment.named_chromatic_pitch_class_vector`
`NamedChromaticPitchSegment.named_chromatic_pitch_set`
`NamedChromaticPitchSegment.named_chromatic_pitch_vector`
`NamedChromaticPitchSegment.named_chromatic_pitches`
`NamedChromaticPitchSegment.numbered_chromatic_pitch_class_segment`
`NamedChromaticPitchSegment.numbered_chromatic_pitch_class_set`

Methods

`NamedChromaticPitchSegment.count` (*value*) → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.index` (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.transpose` (*melodic_interval*)
Transpose pitches in pitch segment by melodic interval and emit new pitch segment.

Special Methods

`NamedChromaticPitchSegment.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchSegment.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchSegment.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchSegment.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__iter__()` `<==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__mul__(n)`

Note: Inherited from `pitchtools._Segment`

`NamedChromaticPitchSegment.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`NamedChromaticPitchSegment.__repr__()`
`NamedChromaticPitchSegment.__rmul__(n)`

Note: Inherited from `pitchtools._Segment`

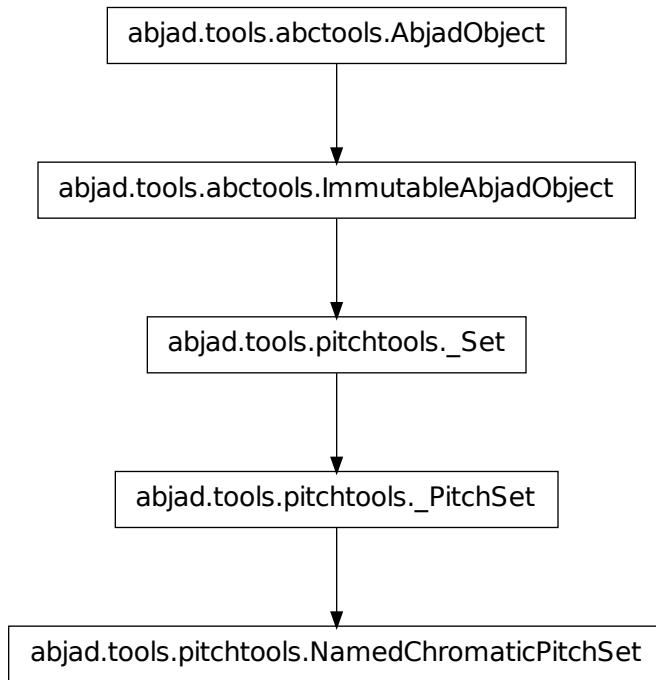
`NamedChromaticPitchSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NamedChromaticPitchSegment.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

`pitchtools.NamedChromaticPitchSet`



class `abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet`

New in version 2.0. Abjad model of a named chromatic pitch set:

```
abjad> pitchtools.NamedChromaticPitchSet(['bf', 'bqf', "fs'", "g'", 'bqf', "g'"])
NamedChromaticPitchSet(['bf', 'bqf', "fs'", "g'"])
```

Named chromatic pitch sets are immutable.

Read-only Properties

`NamedChromaticPitchSet.chromatic_pitch_numbers`

`NamedChromaticPitchSet.duplicate_pitch_classes`

`NamedChromaticPitchSet.is_pitch_class_unique`

`NamedChromaticPitchSet.named_chromatic_pitches`

`NamedChromaticPitchSet.numbered_chromatic_pitch_class_set`

`NamedChromaticPitchSet.numbered_chromatic_pitch_classes`

Methods

`NamedChromaticPitchSet.copy()`

Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NamedChromaticPitchSet.transpose(n)`

Transpose all pcs in self by n.

`NamedChromaticPitchSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

NamedChromaticPitchSet.**__and__**()
x.**__and__**(y) $\iff x \& y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__cmp__**(y) $\iff cmp(x, y)$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__contains__**()
x.**__contains__**(y) $\iff y$ in x.

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__delattr__**()
x.**__delattr__**('name') $\iff \text{del } x.\text{name}$

Note: Inherited from `__builtin__.object`

NamedChromaticPitchSet.**__eq__**(arg)

NamedChromaticPitchSet.**__ge__**()
x.**__ge__**(y) $\iff x \geq y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__gt__**()
x.**__gt__**(y) $\iff x > y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__hash__**() $\iff hash(x)$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__iter__**() $\iff iter(x)$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.**__le__**()
x.**__le__**(y) $\iff x \leq y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__len__() $\iff len(x)$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__lt__()
 $x.__lt__(y) \iff x < y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__ne__(arg)

NamedChromaticPitchSet.__or__()
 $x.__or__(y) \iff x \cup y$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__rand__()
 $x.__rand__(y) \iff y \subseteq x$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__repr__()

NamedChromaticPitchSet.__ror__()
 $x.__ror__(y) \iff y \subseteq x$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__rsub__()
 $x.__rsub__(y) \iff y - x$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__rxor__()
 $x.__rxor__(y) \iff y \oplus x$

Note: Inherited from `__builtin__.frozenset`

NamedChromaticPitchSet.__setattr__()
 $x.__setattr__('name', value) \iff x.name = value$

Note: Inherited from `__builtin__.object`

NamedChromaticPitchSet.__str__()

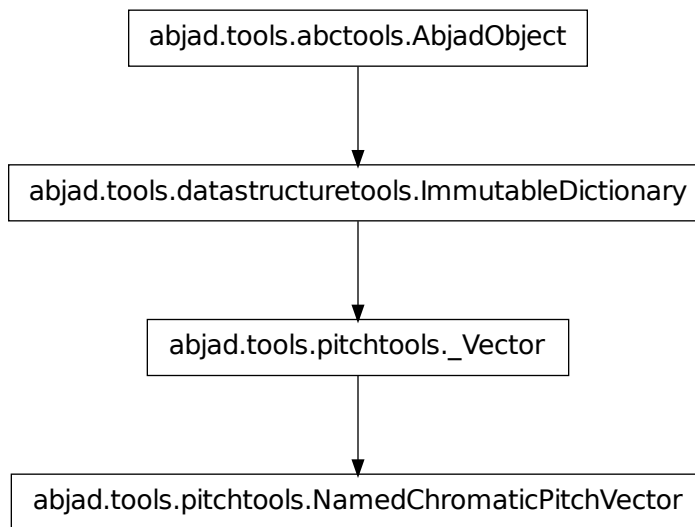
NamedChromaticPitchSet.__sub__()
 $x.__sub__(y) \iff x - y$

Note: Inherited from `__builtin__.frozenset`

```
NamedChromaticPitchSet.__xor__()
x.__xor__(y) <==> x^y
```

Note: Inherited from `__builtin__.frozenset`

`pitchtools.NamedChromaticPitchVector`



class `abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector`. **NamedChromaticPitchVector**
 New in version 2.0. Abjad model of named chromatic pitch vector:

```

abjad> named_chromatic_pitch_vector = pitchtools.NamedChromaticPitchVector(["c'", "c'", "cs'"])

abjad> named_chromatic_pitch_vector
NamedChromaticPitchVector(c': 2, cs': 3)

abjad> print named_chromatic_pitch_vector
NamedChromaticPitchVector(c': 2, cs': 3)
```

Named chromatic pitch vectors are immutable.

Read-only Properties

`NamedChromaticPitchVector.chromatic_pitch_numbers`
`NamedChromaticPitchVector.named_chromatic_pitches`

Methods

`NamedChromaticPitchVector.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.popitem()` \rightarrow (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.setdefault(k[, d])` \rightarrow D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.update(E, **F)` \rightarrow None. Update D from dict/iterable E and F. If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.values()` \rightarrow list of D's values

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewitems()` \rightarrow a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewkeys()` \rightarrow a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.viewvalues()` \rightarrow an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`NamedChromaticPitchVector.__cmp__(y)` $\leq \Rightarrow cmp(x, y)$

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.__contains__(k)` \rightarrow True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
NamedChromaticPitchVector.__delitem__(*args)
```

Note: Inherited from `datastructuretools.ImmutableDictionary`

```
NamedChromaticPitchVector.__eq__()  
x.__eq__(y) <==> x==y
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__ge__()  
x.__ge__(y) <==> x>=y
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__getitem__()  
x.__getitem__(y) <==> x[y]
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__gt__()  
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__iter__() <==> iter(x)
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__le__()  
x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__len__() <==> len(x)
```

Note: Inherited from `__builtin__.dict`

```
NamedChromaticPitchVector.__lt__()  
x.__lt__(y) <==> x<y
```

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

`NamedChromaticPitchVector.__repr__()`
`NamedChromaticPitchVector.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

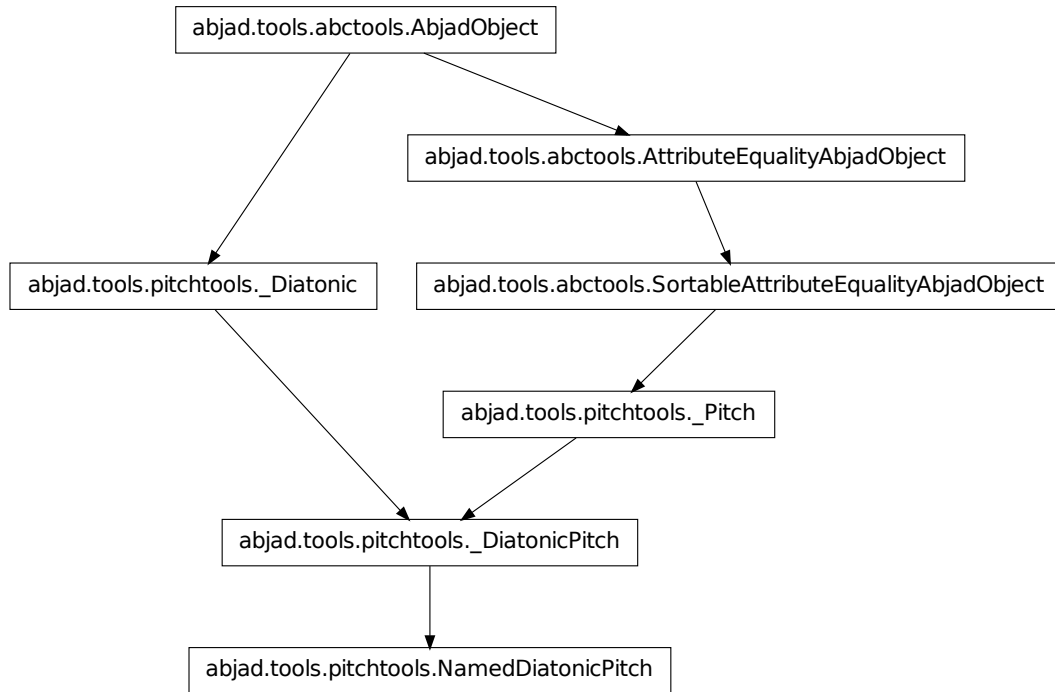
`NamedChromaticPitchVector.__setitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`NamedChromaticPitchVector.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.NamedDiatonicPitch



class `abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch`.**NamedDiatonicPitch** (*arg*)
 New in version 2.0. Abjad model of a named diatonic pitch:

```

abjad> named_diatonic_pitch = pitchtools.NamedDiatonicPitch("c' ")

abjad> named_diatonic_pitch
NamedDiatonicPitch("c' ")

abjad> print named_diatonic_pitch
c'
  
```

Named diatonic pitches are immutable.

Read-only Properties

`NamedDiatonicPitch.chromatic_pitch_class_name`

Read-only chromatic pitch-class name:

```

abjad> pitchtools.NamedDiatonicPitch("c' ").chromatic_pitch_class_name
'c'
  
```

Return string.

`NamedDiatonicPitch.chromatic_pitch_class_number`

Read-only chromatic pitch-class number:

```
abjad> pitchtools.NamedDiatonicPitch("c'").chromatic_pitch_class_number
0
```

Return integer.

NamedDiatonicPitch.chromatic_pitch_name

Read-only chromatic pitch name:

```
abjad> pitchtools.NamedDiatonicPitch("c'").chromatic_pitch_name
"c' "
```

Return string.

NamedDiatonicPitch.chromatic_pitch_number

Read-only chromatic pitch number:

```
abjad> pitchtools.NamedDiatonicPitch("c'").chromatic_pitch_number
12
```

Return integer.

NamedDiatonicPitch.diatonic_pitch_class_name

Read-only diatonic pitch-class name:

```
abjad> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_class_name
'c'
```

Return string.

NamedDiatonicPitch.diatonic_pitch_class_number

Read-only diatonic pitch-class number:

```
abjad> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_class_number
0
```

Return integer.

NamedDiatonicPitch.diatonic_pitch_name

Read-only diatonic pitch name:

```
abjad> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_name
"c' "
```

Return string.

NamedDiatonicPitch.diatonic_pitch_number

Read-only diatonic pitch number:

```
abjad> pitchtools.NamedDiatonicPitch("c'").diatonic_pitch_number
7
```

Return integer.

NamedDiatonicPitch.format

Read-only LilyPond input format of named diatonic pitch:

```
abjad> pitchtools.NamedDiatonicPitch("c'").format
"c' "
```

Return string.

NamedDiatonicPitch.named_chromatic_pitch

Read-only named chromatic pitch:


```
abjad> pitchtools.NamedDiatonicPitch("c'").named_chromatic_pitch
NamedChromaticPitch("c'")
```

Return named chromatic pitch.

NamedDiatonicPitch.named_chromatic_pitch_class

Read-only named chromatic pitch-class:

```
abjad> pitchtools.NamedDiatonicPitch("c'").named_chromatic_pitch_class
NamedChromaticPitchClass('c')
```

Return named chromatic pitch-class.

NamedDiatonicPitch.named_diatonic_pitch_class

Read-only named diatonic pitch-class:

```
abjad> pitchtools.NamedDiatonicPitch("c'").named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

NamedDiatonicPitch.numbered_chromatic_pitch

Read-only numbered chromatic pitch:

```
abjad> pitchtools.NamedDiatonicPitch("c'").numbered_chromatic_pitch
NumberedChromaticPitch(12)
```

Return numbered chromatic pitch.

NamedDiatonicPitch.numbered_chromatic_pitch_class

Read-only numbered chromatic pitch-class:

```
abjad> pitchtools.NamedDiatonicPitch("c'").numbered_chromatic_pitch_class
NumberedChromaticPitchClass(0)
```

Return numbered chromatic pitch-class.

NamedDiatonicPitch.numbered_diatonic_pitch

Read-only numbered diatonic pitch:

```
abjad> pitchtools.NamedDiatonicPitch("c'").numbered_diatonic_pitch
NumberedDiatonicPitch(7)
```

Return numbered diatonic pitch.

NamedDiatonicPitch.numbered_diatonic_pitch_class

Read-only numbered diatonic pitch-class:

```
abjad> pitchtools.NamedDiatonicPitch("c'").numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

Special Methods

NamedDiatonicPitch.__abs__()

NamedDiatonicPitch.__delattr__()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NamedDiatonicPitch.__eq__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedDiatonicPitch.__float__()`
`NamedDiatonicPitch.__ge__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__gt__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__hash__()`

Note: Inherited from `pitchtools._Pitch`

`NamedDiatonicPitch.__int__()`
`NamedDiatonicPitch.__le__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__lt__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NamedDiatonicPitch.__ne__(arg)`

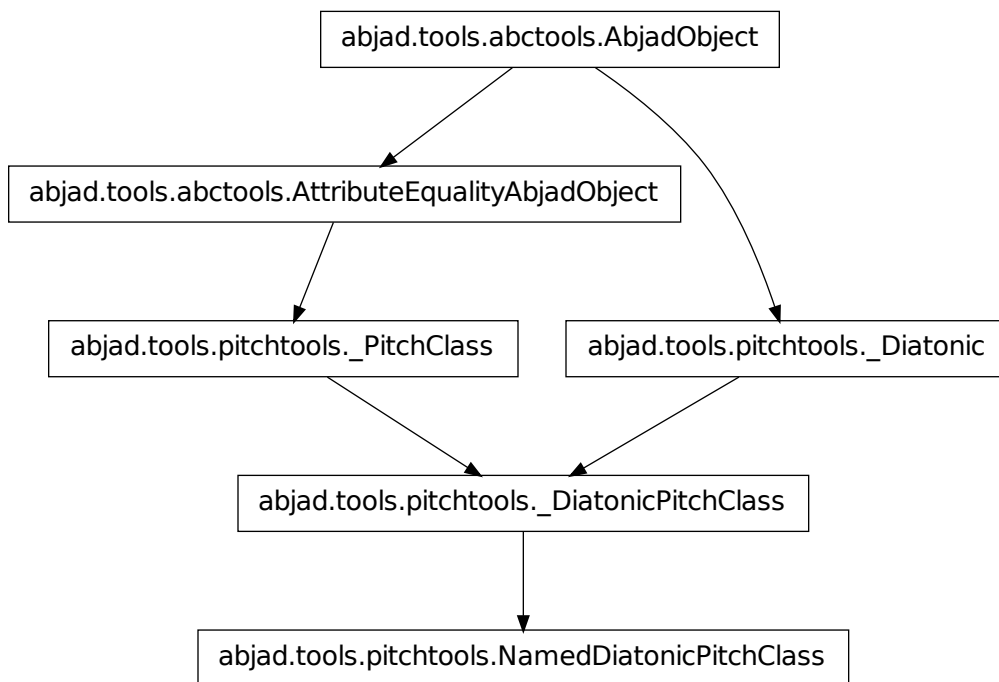
Note: Inherited from `pitchtools._Pitch`

`NamedDiatonicPitch.__repr__()`
`NamedDiatonicPitch.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NamedDiatonicPitch.__str__()`

`pitchtools.NamedDiatonicPitchClass`



class `abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass`.**NamedDiatonicPitchClass**
 New in version 2.0. Abjad model of a named diatonic pitch-class:

```
abjad> pitchtools.NamedDiatonicPitchClass('c')
NamedDiatonicPitchClass('c')
```

Named diatonic pitch-classes are immutable.

Read-only Properties

`NamedDiatonicPitchClass.numbered_diatonic_pitch_class`
 Read-only numbered diatonic pitch-class from named diatonic pitch-class:

```
abjad> named_diatonic_pitch_class = pitchtools.NamedDiatonicPitchClass('c')
abjad> named_diatonic_pitch_class.numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

Special Methods

`NamedDiatonicPitchClass.__abs__()`

`NamedDiatonicPitchClass.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NamedDiatonicPitchClass.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NamedDiatonicPitchClass.__float__()`

`NamedDiatonicPitchClass.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__hash__()`

Note: Inherited from `pitchtools._PitchClass`

`NamedDiatonicPitchClass.__int__()`

`NamedDiatonicPitchClass.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NamedDiatonicPitchClass.__ne__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

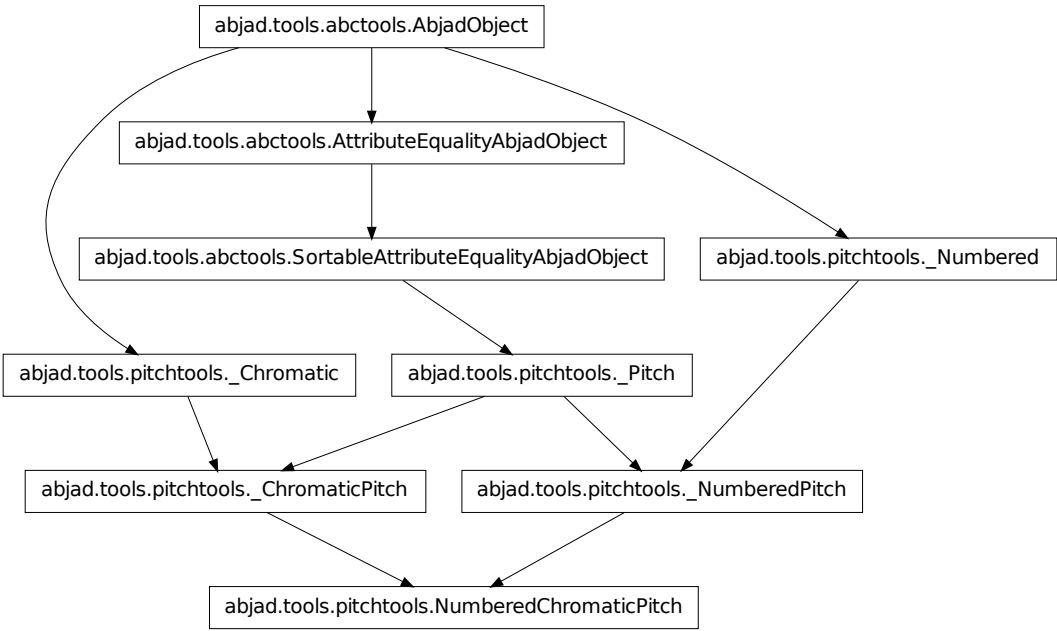
`NamedDiatonicPitchClass.__repr__()`

`NamedDiatonicPitchClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NamedDiatonicPitchClass.__str__()`

pitchtools.NumberedChromaticPitch



class `abjad.tools.pitchtools.NumberedChromaticPitch`, `NumberedChromaticPitch`, **NumberedChromaticPitch**
New in version 2.0. Abjad model of a numbered chromatic pitch:

```
abjad> pitchtools.NumberedChromaticPitch(13)
NumberedChromaticPitch(13)
```

Numbered chromatic pitches are immutable.

Read-only Properties

`NumberedChromaticPitch.chromatic_pitch_number`
Read-only chromatic pitch-class number:

```
abjad> pitchtools.NumberedChromaticPitch(13).chromatic_pitch_number
13
```

Return integer or float.

`NumberedChromaticPitch.diatonic_pitch_class_number`

Read-only diatonic pitch-class number:

```
abjad> pitchtools.NumberedChromaticPitch(13).diatonic_pitch_class_number
0
```

Return integer.

`NumberedChromaticPitch.diatonic_pitch_number`

Read-only diatonic pitch-class number:

```
abjad> pitchtools.NumberedChromaticPitch(13).diatonic_pitch_number
7
```

Return integer.

Methods

`NumberedChromaticPitch.apply_accidental (accidental=None)`

Apply *accidental*:

```
abjad> pitchtools.NumberedChromaticPitch(13).apply_accidental('flat')
NumberedChromaticPitch(12)
```

Return numbered chromatic pitch.

`NumberedChromaticPitch.transpose (n=0)`

Transpose by *n* semitones:

```
abjad> pitchtools.NumberedChromaticPitch(13).transpose(1)
NumberedChromaticPitch(14)
```

Return numbered chromatic pitch.

Special Methods

`NumberedChromaticPitch.__abs__()`

`NumberedChromaticPitch.__add__(arg)`

`NumberedChromaticPitch.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitch.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedChromaticPitch.__float__()`

`NumberedChromaticPitch.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__hash__()`

`NumberedChromaticPitch.__int__()`

`NumberedChromaticPitch.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedChromaticPitch.__ne__(arg)`

Note: Inherited from `pitchtools._Pitch`

`NumberedChromaticPitch.__neg__()`

`NumberedChromaticPitch.__repr__()`

`NumberedChromaticPitch.__setattr__()`

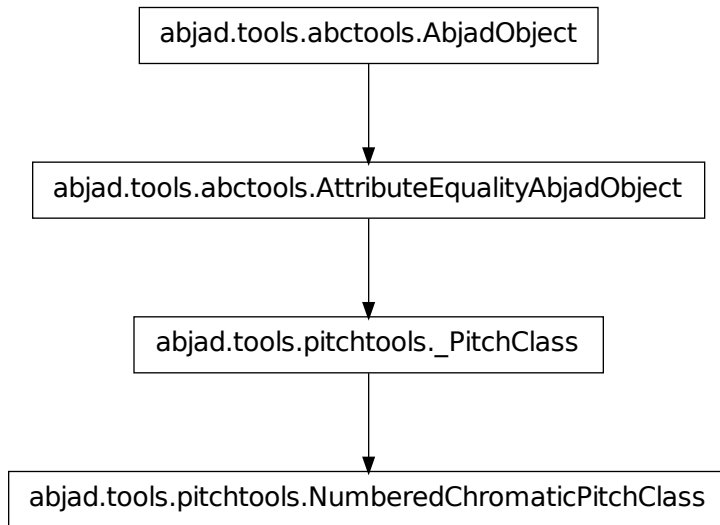
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitch.__str__()`

`NumberedChromaticPitch.__sub__(arg)`

pitchtools.NumberedChromaticPitchClass



class `abjad.tools.pitchtools.NumberedChromaticPitchClass`.`NumberedChromaticPitchClass`.**NumberedChromaticPitchClass**
 New in version 2.0. Abjad model of a numbered chromatic pitch-class:

```

abjad> pitchtools.NumberedChromaticPitchClass(13)
NumberedChromaticPitchClass(1)
  
```

Numbered chromatic pitch-classes are immutable.

Methods

`NumberedChromaticPitchClass`.**apply_accidental** (*accidental=None*)

Emit new numbered chromatic pitch-class as sum of self and accidental.

`NumberedChromaticPitchClass`.**invert** ()

Invert pitch-class.

`NumberedChromaticPitchClass`.**multiply** (*n*)

Multiply pitch-class by *n*.

`NumberedChromaticPitchClass`.**transpose** (*n*)

Transpose pitch-class by *n*.

Special Methods

`NumberedChromaticPitchClass`.**__abs__** ()

`NumberedChromaticPitchClass`.**__add__** (*arg*)

Addition defined against melodic chromatic intervals only.

`NumberedChromaticPitchClass`.**__delattr__** ()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClass.__eq__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedChromaticPitchClass.__float__()`
`NumberedChromaticPitchClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__hash__()`

Note: Inherited from `pitchtools._PitchClass`

`NumberedChromaticPitchClass.__int__()`
`NumberedChromaticPitchClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClass.__ne__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

```

NumberedChromaticPitchClass.__neg__()
NumberedChromaticPitchClass.__repr__()
NumberedChromaticPitchClass.__setattr__()
    x.__setattr__('name', value) <==> x.name = value

```

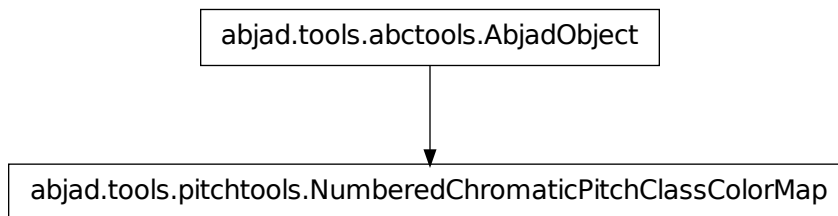
Note: Inherited from `__builtin__.object`

```

NumberedChromaticPitchClass.__str__()
NumberedChromaticPitchClass.__sub__(arg)
    Subtraction defined against both melodic chromatic intervals and against other pitch-classes.

```

`pitchtools.NumberedChromaticPitchClassColorMap`



class `abjad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap`

New in version 2.0. Abjad model of a numbered chromatic pitch-class color map:

```

abjad> chromatic_pitch_class_numbers = [[-8, 2, 10, 21], [0, 11, 32, 41], [15, 25, 42, 43]]
abjad> colors = ['red', 'green', 'blue']
abjad> pitchtools.NumberedChromaticPitchClassColorMap(chromatic_pitch_class_numbers, colors)
NumberedChromaticPitchClassColorMap([[ -8,  2, 10, 21], [ 0, 11, 32, 41], [15, 25, 42, 43]], ['red', 'green', 'blue'])

```

Numbered chromatic pitch-class color maps are immutable.

Read-only Properties

```

NumberedChromaticPitchClassColorMap.colors
NumberedChromaticPitchClassColorMap.pairs
NumberedChromaticPitchClassColorMap.pitch_iterables
NumberedChromaticPitchClassColorMap.twelve_tone_complete
NumberedChromaticPitchClassColorMap.twenty_four_tone_complete

```

Methods

```

NumberedChromaticPitchClassColorMap.get(key, alternative=None)

```

Special Methods

NumberedChromaticPitchClassColorMap.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

NumberedChromaticPitchClassColorMap.__eq__(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

NumberedChromaticPitchClassColorMap.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

NumberedChromaticPitchClassColorMap.__getitem__(pc)
NumberedChromaticPitchClassColorMap.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

NumberedChromaticPitchClassColorMap.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

NumberedChromaticPitchClassColorMap.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

NumberedChromaticPitchClassColorMap.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

NumberedChromaticPitchClassColorMap.__ne__(arg)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NumberedChromaticPitchClassColorMap.__repr__()`

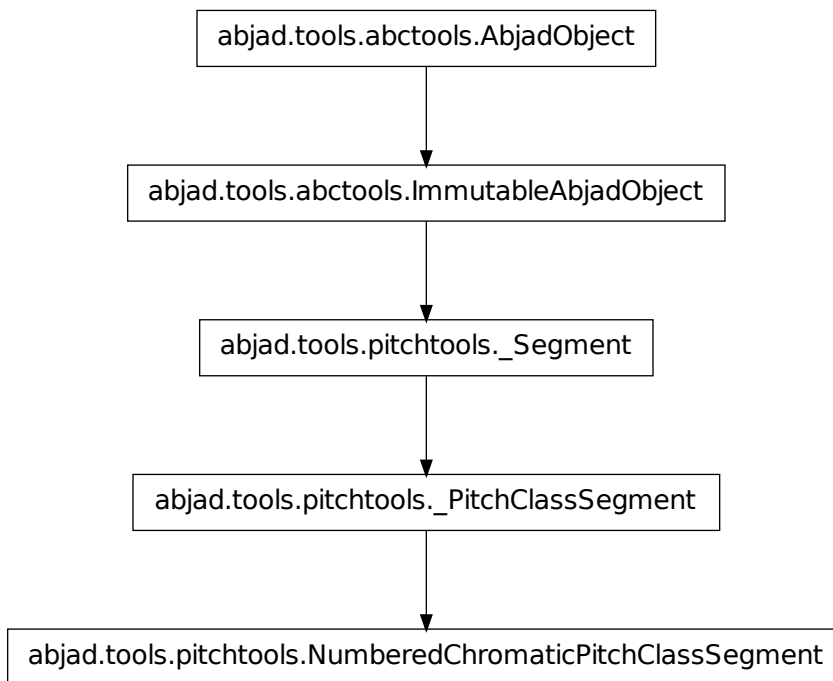
`NumberedChromaticPitchClassColorMap.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClassColorMap.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

`pitchtools.NumberedChromaticPitchClassSegment`



class `abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment`

New in version 2.0. Abjad model of a numbered chromatic pitch-class segment:

```

abjad> pitchtools.NumberedChromaticPitchClassSegment([-2, -1.5, 6, 7, -1.5, 7])
NumberedChromaticPitchClassSegment([10, 10.5, 6, 7, 10.5, 7])
  
```

Numbered chromatic pitch-class segments are immutable.

Read-only Properties

`NumberedChromaticPitchClassSegment.inversion_equivalent_chromatic_interval_class_segment`
 Read-only inversion-equivalent chromatic interval-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5])
numbered_chromatic_pitch_class_segment.inversion_equivalent_chromatic_interval_class_segment
InversionEquivalentChromaticIntervalClassSegment(0.5, 4.5, 1, 3.5, 3.5)
```

Return inversion-equivalent chromatic interval-class segment.

`NumberedChromaticPitchClassSegment.numbered_chromatic_pitch_class_set`
 Read-only numbered chromatic pitch-class set from numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5])
numbered_chromatic_pitch_class_segment.numbered_chromatic_pitch_class_set
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])
```

Return numbered chromatic pitch-class set.

Methods

`NumberedChromaticPitchClassSegment.alpha()`
 Morris alpha transform of numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5])
numbered_chromatic_pitch_class_segment.alpha()
NumberedChromaticPitchClassSegment([11, 11.5, 7, 6, 11.5, 6])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.count(value)` \rightarrow integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.index(value[, start[, stop]])` \rightarrow integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.invert()`
 Invert numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5])
numbered_chromatic_pitch_class_segment.invert()
NumberedChromaticPitchClassSegment([2, 1.5, 6, 5, 1.5, 5])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment.multiply(n)`
 Multiply numbered chromatic pitch-class segment by n :

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5])
numbered_chromatic_pitch_class_segment.multiply(5)
NumberedChromaticPitchClassSegment([2, 4.5, 6, 11, 4.5, 11])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment`.**retrograde**()

Retrograde of numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 7, 6, 10.5, 10])
numbered_chromatic_pitch_class_segment.retrograde()
NumberedChromaticPitchClassSegment([7, 10.5, 7, 6, 10.5, 10])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment`.**rotate**(*n*)

Rotate numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 7, 6, 10.5, 10])
numbered_chromatic_pitch_class_segment.rotate(1)
NumberedChromaticPitchClassSegment([7, 10, 10.5, 6, 7, 10.5])
```

Return numbered chromatic pitch-class segment.

`NumberedChromaticPitchClassSegment`.**transpose**(*n*)

Transpose numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 7, 6, 10.5, 10])
numbered_chromatic_pitch_class_segment.transpose(10)
NumberedChromaticPitchClassSegment([8, 8.5, 4, 5, 8.5, 5])
```

Return numbered chromatic pitch-class segment.

Special Methods

`NumberedChromaticPitchClassSegment`.**__add__**(*arg*)

Note: Inherited from `pitchtools._Segment`

`NumberedChromaticPitchClassSegment`.**__contains__**()

`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment`.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClassSegment`.**__eq__**()

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment`.**__ge__**()

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__getitem__()
x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__getslice__(start, stop)

Note: Inherited from `pitchtools._Segment`

NumberedChromaticPitchClassSegment.__gt__()
x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__le__()
x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__len__() <==> len(x)

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__lt__()
x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.tuple`

NumberedChromaticPitchClassSegment.__mul__(n)

Note: Inherited from `pitchtools._Segment`

NumberedChromaticPitchClassSegment.__ne__()
x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.tuple`

`NumberedChromaticPitchClassSegment.__repr__()`

`NumberedChromaticPitchClassSegment.__rmul__(n)`

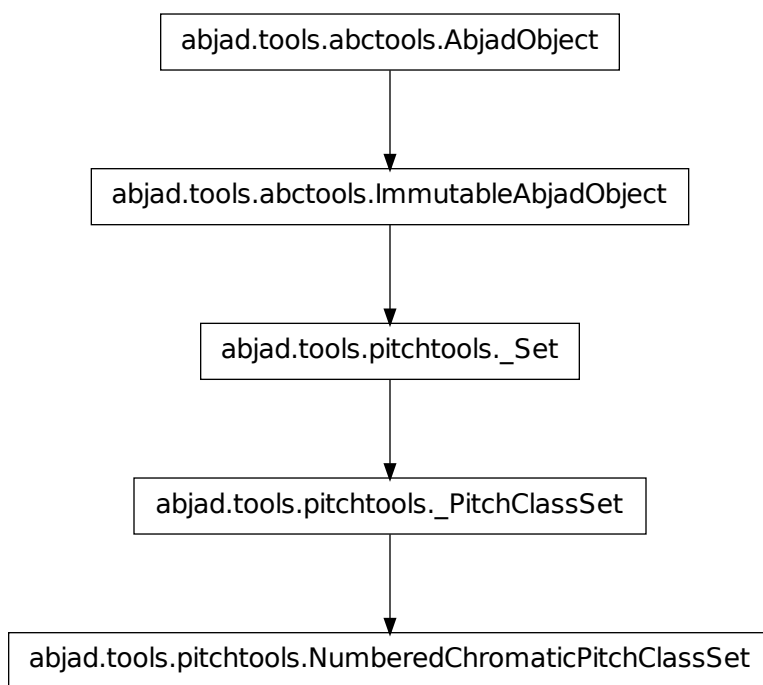
Note: Inherited from `pitchtools._Segment`

`NumberedChromaticPitchClassSegment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClassSegment.__str__()`

`pitchtools.NumberedChromaticPitchClassSet`



class `abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet` .Nu

New in version 2.0. Abjad model of a numbered chromatic pitch-class set:


```

abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])

abjad> print numbered_chromatic_pitch_class_set
{6, 7, 10, 10.5}

```

Numbered chromatic pitch-class sets are immutable.

Read-only Properties

`NumberedChromaticPitchClassSet.inversion_equivalent_chromatic_interval_class_set`
 Read-only inversion-equivalent chromatic interval-class set:

```

abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.inversion_equivalent_chromatic_interval_class_set
InversionEquivalentChromaticIntervalClassSet(0.5, 1, 3, 3.5, 4, 4.5)

```

Return inversion-equivalent chromatic interval-class set.

`NumberedChromaticPitchClassSet.inversion_equivalent_chromatic_interval_class_vector`
 Read-only inversion-equivalent chromatic interval-class vector:

```

abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.inversion_equivalent_chromatic_interval_class_vector
InversionEquivalentChromaticIntervalClassVector(0 | 1 0 1 1 0 0 1 0 0 1 1 0)

```

Return inversion-equivalent chromatic interval-class vector.

`NumberedChromaticPitchClassSet.numbered_chromatic_pitch_classes`
 Read-only numbered chromatic pitch-classes:

```

abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.numbered_chromatic_pitch_classes
(NumberedChromaticPitchClass(6), NumberedChromaticPitchClass(7), NumberedChromaticPitchClass(10))

```

Return tuple.

`NumberedChromaticPitchClassSet.prime_form`
 To be implemented.

Methods

`NumberedChromaticPitchClassSet.copy()`
 Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.difference()`
 Return the difference of two or more sets as a new set.
 (i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.intersection()`
 Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.invert()`

Invert numbered chromatic pitch-class set:

```
abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.invert()
NumberedChromaticPitchClassSet([1.5, 2, 5, 6])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.is_transposed_subset(pcset)`

True when self is transposed subset of *pcset*. False otherwise:

```
abjad> pcset_1 = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5, 6, 7, -1.5, 7])
abjad> pcset_2 = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5, 6, 7, -1.5, 7, 7.5, 8])

abjad> pcset_1.is_transposed_subset(pcset_2)
True
```

Return boolean.

`NumberedChromaticPitchClassSet.is_transposed_superset(pcset)`

True when self is transposed superset of *pcset*. False otherwise:

```
abjad> pcset_1 = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5, 6, 7, -1.5, 7])
abjad> pcset_2 = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5, 6, 7, -1.5, 7, 7.5, 8])

abjad> pcset_2.is_transposed_superset(pcset_1)
True
```

Return boolean.

`NumberedChromaticPitchClassSet.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.multiply(n)`

Multiply numbered chromatic pitch-class set by *n*:

```
abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.multiply(5)
NumberedChromaticPitchClassSet([2, 4.5, 6, 11])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.transpose(n)`

Transpose numbered chromatic pitch-class set by n :

```
abjad> numbered_chromatic_pitch_class_set = pitchtools.NumberedChromaticPitchClassSet([-2, -1.5,
abjad> numbered_chromatic_pitch_class_set.multiply(5)
NumberedChromaticPitchClassSet([2, 4.5, 6, 11])
```

Return numbered chromatic pitch-class set.

`NumberedChromaticPitchClassSet.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

`NumberedChromaticPitchClassSet.__and__()`

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__contains__()`

`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

`NumberedChromaticPitchClassSet.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClassSet.__eq__(arg)`

`NumberedChromaticPitchClassSet.__ge__()`

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__hash__()
 NumberedChromaticPitchClassSet.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__le__()
 x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__len__() <==> len(x)

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__lt__()
 x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__ne__(arg)
 NumberedChromaticPitchClassSet.__or__()
 x.__or__(y) <==> x|y

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__rand__()
 x.__rand__(y) <==> y&x

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__repr__()
 NumberedChromaticPitchClassSet.__ror__()
 x.__ror__(y) <==> y|x

Note: Inherited from `__builtin__.frozenset`

NumberedChromaticPitchClassSet.__rsub__()
 x.__rsub__(y) <==> y-x

Note: Inherited from `__builtin__.frozenset`

```
NumberedChromaticPitchClassSet.__rxor__()
x.__rxor__(y) <==> y^x
```

Note: Inherited from `__builtin__.frozenset`

```
NumberedChromaticPitchClassSet.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

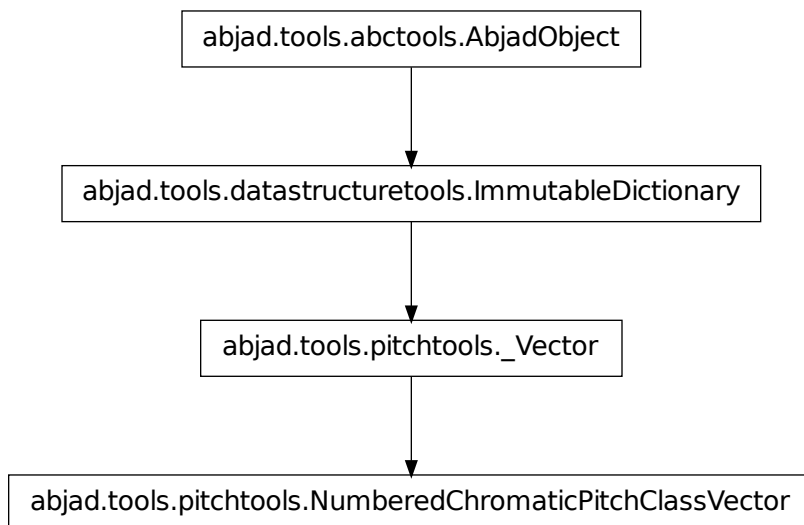
```
NumberedChromaticPitchClassSet.__str__()
NumberedChromaticPitchClassSet.__sub__()
x.__sub__(y) <==> x-y
```

Note: Inherited from `__builtin__.frozenset`

```
NumberedChromaticPitchClassSet.__xor__()
x.__xor__(y) <==> x^y
```

Note: Inherited from `__builtin__.frozenset`

`pitchtools.NumberedChromaticPitchClassVector`



class `abjad.tools.pitchtools.NumberedChromaticPitchClassVector`.`NumberedChromaticPitchClassVec`
 New in version 2.0. Abjad model of numbered chromatic pitch-class vector:

```
abjad> numbered_chromatic_pitch_class_vector = pitchtools.NumberedChromaticPitchClassVector([13,

abjad> numbered_chromatic_pitch_class_vector
NumberedChromaticPitchClassVector(0 2 0 0 0 0 | 3 0 0 0 0 0 || 0 0 3 0 0 0 | 0 0 0 0 0 0)

abjad> print numbered_chromatic_pitch_class_vector
0 2 0 0 0 0 | 3 0 0 0 0 0
0 0 3 0 0 0 | 0 0 0 0 0 0
```

Numbered chromatic pitch-class vectors are immutable.

Read-only Properties

`NumberedChromaticPitchClassVector`.**`chromatic_pitch_class_numbers`**

Read-only chromatic pitch-class numbers from numbered chromatic pitch-class vector:

```
abjad> numbered_chromatic_pitch_class_vector = pitchtools.NumberedChromaticPitchClassVector([13,
abjad> numbered_chromatic_pitch_class_vector.chromatic_pitch_class_numbers
[1, 2.5, 6]
```

Return list.

`NumberedChromaticPitchClassVector`.**`numbered_chromatic_pitch_classes`**

Read-only numbered chromatic pitch-classes from numbered chromatic pitch-class vector:

```
abjad> numbered_chromatic_pitch_class_vector = pitchtools.NumberedChromaticPitchClassVector([13,
abjad> numbered_chromatic_pitch_class_vector.numbered_chromatic_pitch_classes
[NumberedChromaticPitchClass(2.5), NumberedChromaticPitchClass(1), NumberedChromaticPitchClass(6)]
```

Return list.

Methods

`NumberedChromaticPitchClassVector`.**`clear()`** → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector`.**`copy()`** → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector`.**`get(k[, d])`** → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector`.**`has_key(k)`** → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.update(E, **F)` → None. Update D from dict/iterable E and F.
If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`NumberedChromaticPitchClassVector.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__contains__(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`NumberedChromaticPitchClassVector.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedChromaticPitchClassVector.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`NumberedChromaticPitchClassVector.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__ge__()
x.__ge__(y) <==> x>=y
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__getitem__()
x.__getitem__(y) <==> x[y]
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__gt__()
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__iter__() <==> iter(x)
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__le__()
x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__len__() <==> len(x)
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__lt__()
x.__lt__(y) <==> x<y
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__ne__()
x.__ne__(y) <==> x!=y
```

Note: Inherited from `__builtin__.dict`

```
NumberedChromaticPitchClassVector.__repr__()
```

```
NumberedChromaticPitchClassVector.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

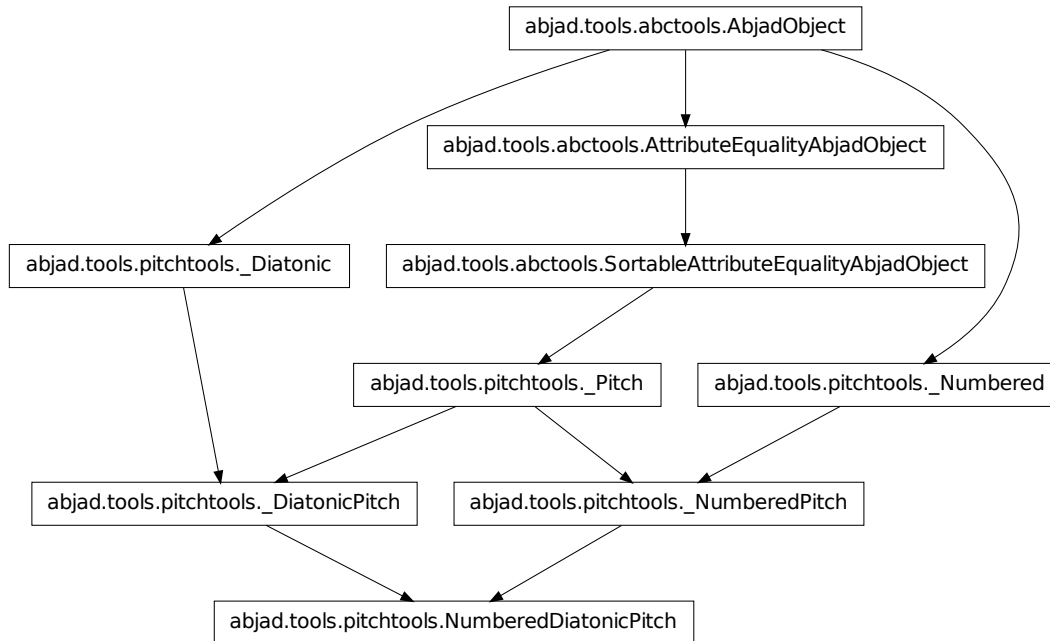
Note: Inherited from `__builtin__.object`

```
NumberedChromaticPitchClassVector.__setitem__(*args)
```

Note: Inherited from `datastructuretools.ImmutableDictionary`

`NumberedChromaticPitchClassVector.__str__()`

`pitchtools.NumberedDiatonicPitch`



class `abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch`
 New in version 2.0. Abjad model of a numbered diatonic pitch:

```
abjad> pitchtools.NumberedDiatonicPitch(7)
NumberedDiatonicPitch(7)
```

Numbered diatonic pitches are immutable.

Read-only Properties

`NumberedDiatonicPitch.chromatic_pitch_number`

Read-only chromatic pitch number:

```
abjad> pitchtools.NumberedDiatonicPitch(7).chromatic_pitch_number
12
```

Return integer.

`NumberedDiatonicPitch.diatonic_pitch_number`

Read-only diatonic pitch number:

```
abjad> pitchtools.NumberedDiatonicPitch(7).diatonic_pitch_number
7
```

Return integer.

`NumberedDiatonicPitch.named_diatonic_pitch`

Read-only named diatonic pitch:

```
abjad> pitchtools.NumberedDiatonicPitch(7).named_diatonic_pitch
NamedDiatonicPitch('c')
```

Return named diatonic pitch.

`NumberedDiatonicPitch.named_diatonic_pitch_class`

Read-only named diatonic pitch-class:

```
abjad> pitchtools.NumberedDiatonicPitch(7).named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

`NumberedDiatonicPitch.numbered_diatonic_pitch_class`

Read-only numbered diatonic pitch-class:

```
abjad> pitchtools.NumberedDiatonicPitch(7).numbered_diatonic_pitch_class
NumberedDiatonicPitchClass(0)
```

Return numbered diatonic pitch-class.

Special Methods

`NumberedDiatonicPitch.__abs__()`

`NumberedDiatonicPitch.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedDiatonicPitch.__eq__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedDiatonicPitch.__float__()`

Note: Inherited from `pitchtools._DiatonicPitch`

`NumberedDiatonicPitch.__ge__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedDiatonicPitch.__gt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedDiatonicPitch.__hash__()`

Note: Inherited from `pitchtools._Pitch`

`NumberedDiatonicPitch.__int__()`

Note: Inherited from `pitchtools._DiatonicPitch`

`NumberedDiatonicPitch.__le__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedDiatonicPitch.__lt__(arg)`

Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.SortableAttributeEqualityAbjadObject`

`NumberedDiatonicPitch.__ne__(arg)`

Note: Inherited from `pitchtools._Pitch`

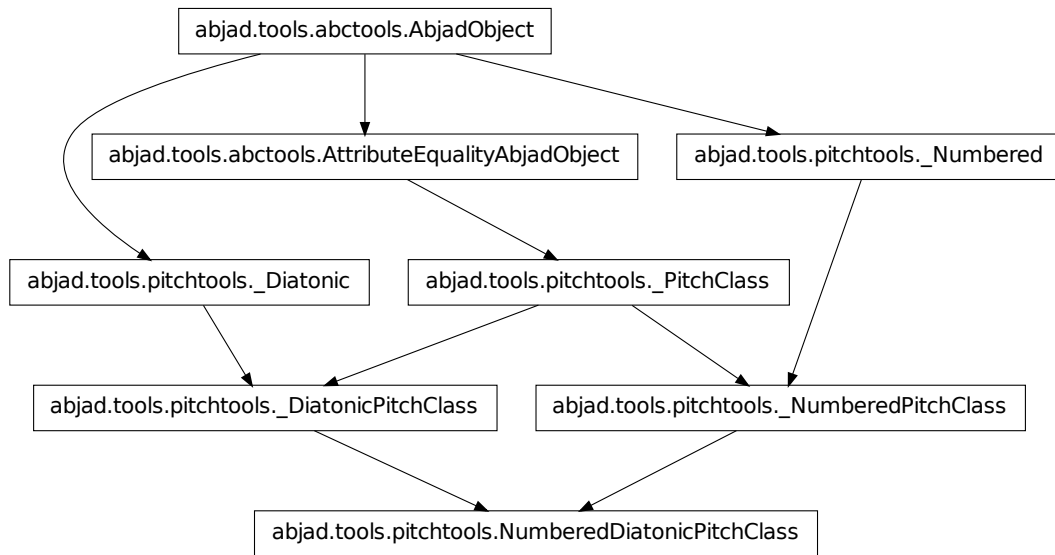
`NumberedDiatonicPitch.__repr__()`

`NumberedDiatonicPitch.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NumberedDiatonicPitch.__str__()`

pitchtools.NumberedDiatonicPitchClass

class `abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass`
 New in version 2.0. Abjad model of a numbered diatonic pitch-class:

```
abjad> pitchtools.NumberedDiatonicPitchClass(0)
NumberedDiatonicPitchClass(0)
```

Numbered diatonic pitch-classes are immutable.

Read-only Properties

`NumberedDiatonicPitchClass.named_diatonic_pitch_class`

Read-only named diatonic pitch-class from numbered diatonic pitch-class:

```
abjad> numbered_diatonic_pitch_class = pitchtools.NumberedDiatonicPitchClass(0)
abjad> numbered_diatonic_pitch_class.named_diatonic_pitch_class
NamedDiatonicPitchClass('c')
```

Return named diatonic pitch-class.

Special Methods

`NumberedDiatonicPitchClass.__abs__()`

Note: Inherited from `pitchtools._DiatonicPitchClass`

`NumberedDiatonicPitchClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NumberedDiatonicPitchClass.__eq__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.
Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedDiatonicPitchClass.__float__()`

Note: Inherited from `pitchtools._DiatonicPitchClass`

`NumberedDiatonicPitchClass.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__hash__()`

Note: Inherited from `pitchtools._PitchClass`

`NumberedDiatonicPitchClass.__int__()`

Note: Inherited from `pitchtools._DiatonicPitchClass`

`NumberedDiatonicPitchClass.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NumberedDiatonicPitchClass.__ne__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

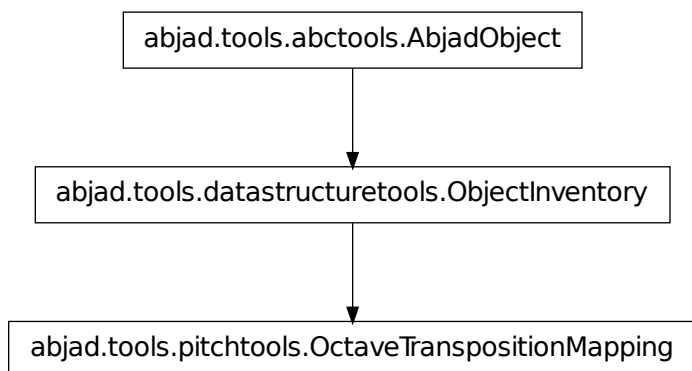
Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`NumberedDiatonicPitchClass.__repr__()`
`NumberedDiatonicPitchClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NumberedDiatonicPitchClass.__str__()`

pitchtools.OctaveTranspositionMapping



class `abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTran`

New in version 2.8. Octave transposition mapping:

```

abjad> pitchtools.OctaveTranspositionMapping([('A0, C4)', 15), ('[C4, C8)', 27)])
OctaveTranspositionMapping([('A0, C4)', 15), ('[C4, C8)', 27)])
  
```

Octave transposition mappings model `pitchtools.transpose_chromatic_pitch_number_by_octave_transpo` input.

Octave transposition mappings implement the list interface and are mutable.

Read/write Properties

OctaveTranspositionMapping.**inventory_name**

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

OctaveTranspositionMapping.**append**(*token*)

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMapping.**count**(*value*) → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**extend**(*tokens*)

Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMapping.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.

Raises ValueError if the value is not present.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**insert**()

L.insert(index, object) – insert object before index

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**pop**([*index*]) → item – remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**remove**()

L.remove(value) – remove first occurrence of value. Raises ValueError if the value is not present.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**reverse**()

L.reverse() – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.**sort**()

L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

Note: Inherited from `__builtin__.list`

Special Methods

`OctaveTranspositionMapping.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`OctaveTranspositionMapping.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OctaveTranspositionMapping.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMapping.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__iter__()` \leq `iter(x)`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__len__()` \leq `len(x)`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__mul__()`
`x.__mul__(n) <==> x*n`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.`__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.__repr__()

OctaveTranspositionMapping.__reversed__()
L.__reversed__() – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.__rmul__()
x.__rmul__(n) <==> n*x

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

OctaveTranspositionMapping.__setitem__()
x.__setitem__(i, y) <==> x[i]=y

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.__setslice__()
x.__setslice__(i, j, y) <==> x[i:j]=y

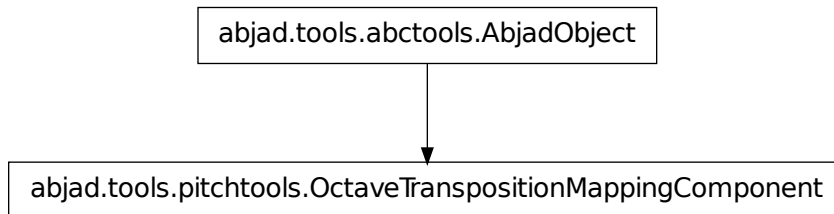
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMapping.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

pitchtools.OctaveTranspositionMappingComponent



class `abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent`
 New in version 2.8. Octave transposition mapping component:

```
abjad> pitchtools.OctaveTranspositionMappingComponent(' [A0, C8]', 15)
OctaveTranspositionMappingComponent(' [A0, C8]', 15)
```

Initialize from input parameters separately, from a pair, from a string or from another mapping component.

Model `pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping`
 input part. (See the docs for that function.)

Octave transposition mapping components are mutable.

Read/write Properties

`OctaveTranspositionMappingComponent.source_pitch_range`

Read / write source pitch range:

```
abjad> mapping_component = pitchtools.OctaveTranspositionMappingComponent(' [A0, C8]', 15)
abjad> mapping_component.source_pitch_range
PitchRange(' [A0, C8]')
```

Return pitch range or none.

`OctaveTranspositionMappingComponent.target_octave_start_pitch`

Read / write target octave start pitch:

```
abjad> mapping_component = pitchtools.OctaveTranspositionMappingComponent(' [A0, C8]', 15)
abjad> mapping_component.target_octave_start_pitch
NumberedChromaticPitch(15)
```

Return numbered chromatic pitch or none.

Special Methods

`OctaveTranspositionMappingComponent.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OctaveTranspositionMappingComponent.__eq__(other)`

OctaveTranspositionMappingComponent.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

OctaveTranspositionMappingComponent.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

OctaveTranspositionMappingComponent.**__hash__**() \Leftrightarrow *hash*(x)

Note: Inherited from `__builtin__.object`

OctaveTranspositionMappingComponent.**__le__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

OctaveTranspositionMappingComponent.**__lt__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

OctaveTranspositionMappingComponent.**__ne__**(other)

OctaveTranspositionMappingComponent.**__repr__**()

OctaveTranspositionMappingComponent.**__setattr__**()

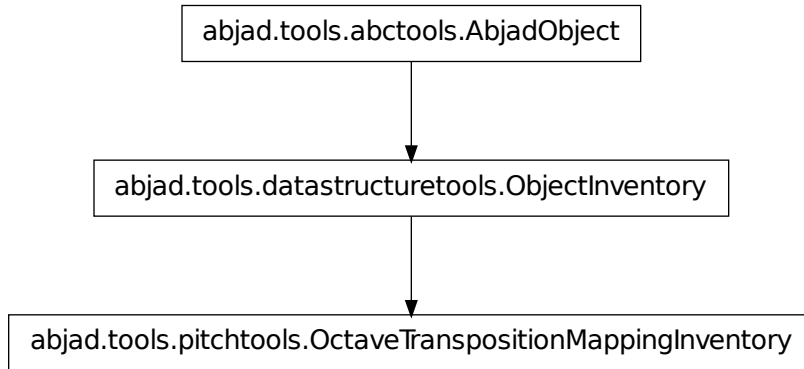
x.**__setattr__**('name', value) \Leftrightarrow *x*.name = value

Note: Inherited from `__builtin__.object`

OctaveTranspositionMappingComponent.**__str__**() \Leftrightarrow *str*(x)

Note: Inherited from `__builtin__.object`

pitchtools.OctaveTranspositionMappingInventory



class `abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory`

New in version 2.8. Model of an ordered list of octave transposition mappings:

```

abjad> mapping_1 = pitchtools.OctaveTranspositionMapping([('A0', 'C4'), 15], ('C4', 'C8'), 27))
abjad> mapping_2 = pitchtools.OctaveTranspositionMapping([('A0', 'C8'), -18])
abjad> inventory = pitchtools.OctaveTranspositionMappingInventory([mapping_1, mapping_2])

abjad> inventory
OctaveTranspositionMappingInventory([OctaveTranspositionMapping([('A0', 'C4'), 15], ('C4', 'C8'), 27),
OctaveTranspositionMapping([('A0', 'C8'), -18])])
  
```

Octave transposition mapping inventories implement list interface and are mutable.

Read/write Properties

`OctaveTranspositionMappingInventory.inventory_name`

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`OctaveTranspositionMappingInventory.append(token)`

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`OctaveTranspositionMappingInventory.count(value) → integer` – return number of occurrences of *value*

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**extend**(*tokens*)
 Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

OctaveTranspositionMappingInventory.**index**(*value*[, *start*[, *stop*]]) → integer – return first
 index of value.
 Raises ValueError if the value is not present.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**insert**()
 L.insert(index, object) – insert object before index

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**pop**([*index*]) → item – remove and return item at index
 (default last).
 Raises IndexError if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**remove**()
 L.remove(value) – remove first occurrence of value. Raises ValueError if the value is not present.

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**reverse**()
 L.reverse() – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**sort**()
 L.sort(cmp=None, key=None, reverse=False) – stable sort *IN PLACE*; cmp(x, y) -> -1, 0, 1

Note: Inherited from `__builtin__.list`

Special Methods

OctaveTranspositionMappingInventory.**__add__**()
 x.__add__(y) <==> x+y

Note: Inherited from `__builtin__.list`

OctaveTranspositionMappingInventory.**__contains__**(*token*)

Note: Inherited from `datastructuretools.ObjectInventory`

```
OctaveTranspositionMappingInventory.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
OctaveTranspositionMappingInventory.__delitem__()  
x.__delitem__(y) <==> del x[y]
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__delslice__()  
x.__delslice__(i, j) <==> del x[i:j]
```

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__eq__()  
x.__eq__(y) <==> x==y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__ge__()  
x.__ge__(y) <==> x>=y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__getitem__()  
x.__getitem__(y) <==> x[y]
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__getslice__()  
x.__getslice__(i, j) <==> x[i:j]
```

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__gt__()  
x.__gt__(y) <==> x>y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__iadd__()  
x.__iadd__(y) <==> x+=y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__imul__ ()
    x.__imul__(y) <==> x*=y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__iter__ () <==> iter(x)
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__le__ ()
    x.__le__(y) <==> x<=y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__len__ () <==> len(x)
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__lt__ ()
    x.__lt__(y) <==> x<y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__mul__ ()
    x.__mul__(n) <==> x*n
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__ne__ ()
    x.__ne__(y) <==> x!=y
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__repr__ ()
```

Note: Inherited from `datastructuretools.ObjectInventory`

```
OctaveTranspositionMappingInventory.__reversed__ ()
    L.__reversed__() – return a reverse iterator over the list
```

Note: Inherited from `__builtin__.list`

```
OctaveTranspositionMappingInventory.__rmul__ ()
    x.__rmul__(n) <==> n*x
```

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`OctaveTranspositionMappingInventory.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`

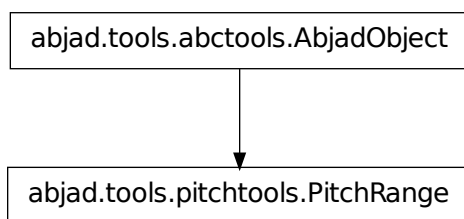
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`OctaveTranspositionMappingInventory.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.PitchRange



class `abjad.tools.pitchtools.PitchRange.PitchRange(*args, **kwargs)`

New in version 2.0. Abjad model of pitch range:

```

abjad> pitchtools.PitchRange(-12, 36)
PitchRange(' [C3, C7] ')
  
```

Initialize from pitch numbers, pitch names, pitch instances, one-line reprs or other pitch range objects.

Pitch ranges test for equality and inequality against other pitch ranges.

Pitch ranges test less than, greater than, less-equal and greater-equal against pitches.

Pitch ranges do not sort relative to other pitch ranges.

Pitch ranges are immutable.

Read-only Properties

`PitchRange.one_line_named_chromatic_pitch_repr`

Read-only one-line named chromatic pitch repr of pitch of range:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.one_line_named_chromatic_pitch_repr
'[C3, c7]'
```

Return string.

`PitchRange.one_line_numbered_chromatic_pitch_repr`

Read-only one-line numbered chromatic pitch repr of pitch of range:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.one_line_numbered_chromatic_pitch_repr
'[-12, 36]'
```

Return string.

`PitchRange.pitch_range_name`

New in version 2.7. Read-only name of pitch range:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36, pitch_range_name='four-octave range')
abjad> pitch_range.pitch_range_name
'four-octave range'
```

Return string or none.

`PitchRange.pitch_range_name_markup`

New in version 2.7. Read-only markup of pitch range name:

```
abjad> from abjad.tools.markuptools import Markup

abjad> pitch_range = pitchtools.PitchRange(-12, 36, pitch_range_name_markup=Markup('four-octave
abjad> pitch_range.pitch_range_name_markup
Markup('four-octave range')
```

Default to *pitch_range_name* when *pitch_range_name_markup* not set explicitly.

Return markup or none.

`PitchRange.start_pitch`

Read-only start pitch of range:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.start_pitch
NamedChromaticPitch('c')
```

Return pitch.

`PitchRange.start_pitch_is_included_in_range`

Read-only boolean true when start pitch is included in range. Otherwise false:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.start_pitch_is_included_in_range
True
```

Return boolean.

`PitchRange.stop_pitch`

Read-only stop pitch of range:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.stop_pitch
NamedChromaticPitch("c' ' ' ' ")
```

Return pitch.

`PitchRange.stop_pitch_is_included_in_range`

Read-only boolean true when stop pitch is included in range. Otherwise false:

```
abjad> pitch_range = pitchtools.PitchRange(-12, 36)
abjad> pitch_range.stop_pitch_is_included_in_range
True
```

Return boolean.

Special Methods

`PitchRange.__contains__ (arg)`

`PitchRange.__delattr__ ()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`PitchRange.__eq__ (arg)`

`PitchRange.__ge__ (arg)`

`PitchRange.__gt__ (arg)`

`PitchRange.__hash__ () <==> hash(x)`

Note: Inherited from `__builtin__.object`

`PitchRange.__le__ (arg)`

`PitchRange.__lt__ (arg)`

`PitchRange.__ne__ (arg)`

`PitchRange.__repr__ ()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`PitchRange.__setattr__ ()`

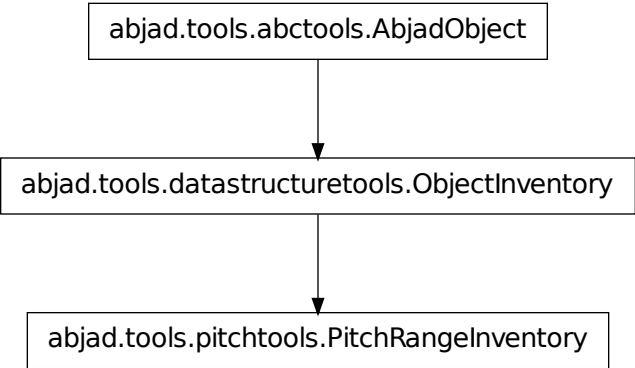
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

PitchRange.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

pitchtools.PitchRangeInventory



class `abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory`. **PitchRangeInventory** (*token inventory_*

New in version 2.7. Abjad model of an ordered list of pitch ranges:

```
abjad> pitchtools.PitchRangeInventory(['[C3, C6]', '[C4, C6]'])
PitchRangeInventory([PitchRange('[C3, C6]'), PitchRange('[C4, C6]')])
```

Pitch range inventories implement list interface and are mutable.

Read/write Properties

`PitchRangeInventory.inventory_name`
Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`PitchRangeInventory.append(token)`
Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`PitchRangeInventory.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.extend(tokens)`
Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`PitchRangeInventory.index(value[, start[, stop]])` → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.insert()`
`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.pop([index])` → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`PitchRangeInventory.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__contains__(token)

Note: Inherited from `datastructuretools.ObjectInventory`

PitchRangeInventory.__delattr__()
 x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

PitchRangeInventory.__delitem__()
 x.__delitem__(y) <==> del x[y]

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__delslice__()
 x.__delslice__(i, j) <==> del x[i:j]

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__eq__()
 x.__eq__(y) <==> x==y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__ge__()
 x.__ge__(y) <==> x>=y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__getitem__()
 x.__getitem__(y) <==> x[y]

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__getslice__()
 x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__gt__()
 x.__gt__(y) <==> x>y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__iadd__()
x.__iadd__(y) <==> x+=y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__imul__()
x.__imul__(y) <==> x*=y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__iter__() <==> iter(x)

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__le__()
x.__le__(y) <==> x<=y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__len__() <==> len(x)

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__lt__()
x.__lt__(y) <==> x<y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__mul__()
x.__mul__(n) <==> x*n

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__ne__()
x.__ne__(y) <==> x!=y

Note: Inherited from `__builtin__.list`

PitchRangeInventory.__repr__()

Note: Inherited from `datastructuretools.ObjectInventory`

PitchRangeInventory.__reversed__()
L.__reversed__() – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.__rmul__()`
`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`PitchRangeInventory.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

Note: Inherited from `__builtin__.list`

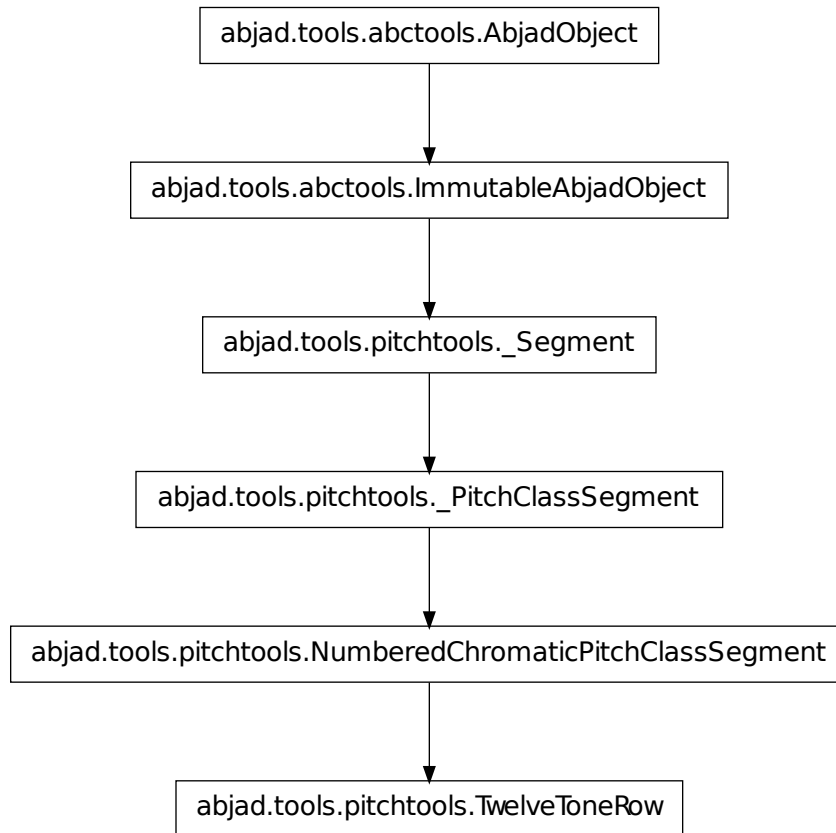
`PitchRangeInventory.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`PitchRangeInventory.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

pitchtools.TwelveToneRow



```
class abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow(TwelveToneRow (*args,
                                                                    **kwargs)
```

New in version 2.0. Abjad model of twelve-tone row:

```
abjad> pitchtools.TwelveToneRow([0, 1, 11, 9, 3, 6, 7, 5, 4, 10, 2, 8])
TwelveToneRow([0, 1, 11, 9, 3, 6, 7, 5, 4, 10, 2, 8])
```

Twelve-tone rows validate pitch-classes at initialization.

Twelve-tone rows inherit canonical operators from numbered chromatic pitch-class segment.

Twelve-tone rows return numbered chromatic pitch-class segments on calls to getslice.

Twelve-tone rows are immutable.

Read-only Properties

`TwelveToneRow.inversion_equivalent_chromatic_interval_class_segment`

Read-only inversion-equivalent chromatic interval-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5]
numbered_chromatic_pitch_class_segment.inversion_equivalent_chromatic_interval_class_segment
InversionEquivalentChromaticIntervalClassSegment(0.5, 4.5, 1, 3.5, 3.5)
```

Return inversion-equivalent chromatic interval-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.numbered_chromatic_pitch_class_set`

Read-only numbered chromatic pitch-class set from numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5]
numbered_chromatic_pitch_class_segment.numbered_chromatic_pitch_class_set
NumberedChromaticPitchClassSet([6, 7, 10, 10.5])
```

Return numbered chromatic pitch-class set.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

Methods

`TwelveToneRow.alpha()`

Morris alpha transform of numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5]
numbered_chromatic_pitch_class_segment.alpha()
NumberedChromaticPitchClassSegment([11, 11.5, 7, 6, 11.5, 6])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.count(value) → integer` – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.index(value[, start[, stop]]) → integer` – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.invert()`

Invert numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5]
numbered_chromatic_pitch_class_segment.invert()
NumberedChromaticPitchClassSegment([2, 1.5, 6, 5, 1.5, 5])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.multiply(n)`

Multiply numbered chromatic pitch-class segment by *n*:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5])
numbered_chromatic_pitch_class_segment.multiply(5)
NumberedChromaticPitchClassSegment([2, 4.5, 6, 11, 4.5, 11])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.retrograde()`

Retrograde of numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5])
numbered_chromatic_pitch_class_segment.retrograde()
NumberedChromaticPitchClassSegment([7, 10.5, 7, 6, 10.5, 10])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.rotate(n)`

Rotate numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5])
numbered_chromatic_pitch_class_segment.rotate(1)
NumberedChromaticPitchClassSegment([7, 10, 10.5, 6, 7, 10.5])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.transpose(n)`

Transpose numbered chromatic pitch-class segment:

```
numbered_chromatic_pitch_class_segment = pitchtools.NumberedChromaticPitchClassSegment([10, 10.5, 11, 11.5, 12, 12.5, 13, 13.5, 14, 14.5, 15, 15.5])
numbered_chromatic_pitch_class_segment.transpose(10)
NumberedChromaticPitchClassSegment([8, 8.5, 4, 5, 8.5, 5])
```

Return numbered chromatic pitch-class segment.

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

Special Methods

`TwelveToneRow.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`TwelveToneRow.__contains__()`

`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TwelveToneRow.__eq__(arg)`
`TwelveToneRow.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__getslice__(start, stop)`
`TwelveToneRow.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__len__() <==> len(x)`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`TwelveToneRow.__mul__(n)`

`TwelveToneRow.__ne__(arg)`

`TwelveToneRow.__repr__()`

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

`TwelveToneRow.__rmul__(n)`

`TwelveToneRow.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TwelveToneRow.__str__()`

Note: Inherited from `pitchtools.NumberedChromaticPitchClassSegment`

pitchtools functions

pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs

`abjad.tools.pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs.all_are_chromatic_pitch_class_name_octave_number_pairs(expr)`
 New in version 1.1. True when all elements of *expr* are pitch tokens. Otherwise false:

```
abjad> pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs([('c', 4), ('d', 4), ('e', 4)])
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_pitch_token_collection()` to `pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs()`.

pitchtools.all_are_named_chromatic_pitch_tokens

`abjad.tools.pitchtools.all_are_named_chromatic_pitch_tokens.all_are_named_chromatic_pitch_tokens(expr)`
 New in version 2.6. True when *expr* is a sequence of named chromatic pitch tokens:

```
abjad> named_chromatic_pitch_tokens = [('c', 4), pitchtools.NamedChromaticPitch("a")]
abjad> pitchtools.all_are_named_chromatic_pitch_tokens(named_chromatic_pitch_tokens)
True
```

```
abjad> pitchtools.all_are_named_chromatic_pitch_tokens(named_chromatic_pitch_tokens)
True
```

True when *expr* is an empty sequence:

```
abjad> pitchtools.all_are_named_chromatic_pitch_tokens([])
True
```

Otherwise false:

```
abjad> pitchtools.all_are_named_chromatic_pitch_tokens('foo')
False
```

Return boolean.

`pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string`

`abjad.tools.pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string.al`
 New in version 2.5. Change *alphabetic_accidental_abbreviation* to symbolic accidental string:

```
abjad> pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string('tqs')
'#+'
```

None when *alphabetic_accidental_abbreviation* is not a valid alphabetic accidental abbreviation.

Return string or none.

`pitchtools.apply_accidental_to_named_chromatic_pitch`

`abjad.tools.pitchtools.apply_accidental_to_named_chromatic_pitch.apply_accidental_to_named`

New in version 2.0. Apply *accidental* to *named_chromatic_pitch*:

```
abjad> pitch = pitchtools.NamedChromaticPitch("cs'")
abjad> pitchtools.apply_accidental_to_named_chromatic_pitch(pitch, 'f')
NamedChromaticPitch("c'")
```

Return new named pitch.

`pitchtools.apply_octavation_spanner_to_pitched_components`

`abjad.tools.pitchtools.apply_octavation_spanner_to_pitched_components.apply_octavation_spa`

New in version 1.1. Apply octavation spanner to pitched components in *expr*:

```
abjad> t = Measure((4, 8), notetools.make_notes([24, 26, 27, 29], [(1, 8)]))
abjad> pitchtools.apply_octavation_spanner_to_pitched_components(t, ottava_numbered_diatonic_pitch)
OctavationSpanner(|4/8(4)|)
```

```
abjad> print t.format
{
    \time 4/8
    \ottava #1
    c'''8
    d'''8
    ef'''8
    f'''8
```

```

        \ottava #0
    }

```

Apply octavation spanner according to the diatonic pitch number of the maximum pitch in *expr*.

Return octavation spanner.

pitchtools.calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier

```
abjad.tools.pitchtools.calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier
```

New in version 2.0. Calculate harmonic chromatic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```

abjad> pitchtools.calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier(pitch_carrier_1, pitch_carrier_2)
HarmonicChromaticIntervalClass(2)

```

Return harmonic chromatic interval-class.

pitchtools.calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier

```
abjad.tools.pitchtools.calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier
```

New in version 2.0. Calculate harmonic chromatic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```

abjad> pitchtools.calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier(pitch_carrier_1, pitch_carrier_2)
HarmonicChromaticInterval(14)

```

Return harmonic chromatic interval.

pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch

```
abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate harmonic counterpoint interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```

abjad> pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch(named_chromatic_pitch_1, named_chromatic_pitch_2)
HarmonicCounterpointIntervalClass(2)

```

Return harmonic counterpoint interval-class. Changed in version 2.0: renamed `pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_pchromatic_pitch_to_named_pchromatic_pitch` to `pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch`.

pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch

```
abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate harmonic counterpoint interval *pitch_carrier_1* to *pitch_carrier_2*:

```

abjad> pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch(named_chromatic_pitch_1, named_chromatic_pitch_2)
HarmonicCounterpointInterval(9)

```

Return harmonic counterpoint interval-class.

pitchtools.calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch

```
abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch
```

New in version 2.0. Calculate harmonic diatonic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch  
HarmonicDiatonicIntervalClass('M2')
```

Return harmonic diatonic interval-class.

pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch

```
abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate harmonic diatonic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch  
HarmonicDiatonicInterval('M9')
```

Return harmonic diatonic interval.

pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier

```
abjad.tools.pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier
```

New in version 2.0. Calculate melodic chromatic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier  
MelodicChromaticIntervalClass(+2)
```

Return melodic chromatic interval-class.

pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier

```
abjad.tools.pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier
```

New in version 2.0. Calculate melodic chromatic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier(pitch_carrier_1, pitch_carrier_2)  
MelodicChromaticInterval(+14)
```

Return melodic chromatic interval.

pitchtools.calculate_melodic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch

```
abjad.tools.pitchtools.calculate_melodic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate melodic counterpoint interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch  
MelodicCounterpointIntervalClass(+2)
```

Return melodic counterpoint interval-class.

`pitchtools.calculate_melodic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch`

```
abjad.tools.pitchtools.calculate_melodic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate melodic counterpoint interval *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch  
MelodicCounterpointInterval(+9)
```

Return melodic counterpoint interval.

`pitchtools.calculate_melodic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch`

```
abjad.tools.pitchtools.calculate_melodic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate melodic diatonic interval-class from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch  
MelodicDiatonicIntervalClass('+M2')
```

Return melodic diatonic interval-class.

`pitchtools.calculate_melodic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch`

```
abjad.tools.pitchtools.calculate_melodic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch
```

New in version 2.0. Calculate melodic diatonic interval from *pitch_carrier_1* to *pitch_carrier_2*:

```
abjad> pitchtools.calculate_melodic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch  
MelodicDiatonicInterval('+M9')
```

Return melodic diatonic interval.

`pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number`

```
abjad.tools.pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number.chromatic_pitch_class_name_to_chromatic_pitch_class_number
```

New in version 2.0. Change *chromatic_pitch_class_name* to chromatic pitch-class number:

```
abjad> pitchtools.chromatic_pitch_class_name_to_chromatic_pitch_class_number('cs')  
1
```

Return chromatic pitch-class number.

`pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name`

```
abjad.tools.pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name.chromatic_pitch_class_name_to_diatonic_pitch_class_name
```

New in version 2.0. Change *chromatic_pitch_class_name* to diatonic pitch-class name:

```
abjad> pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name('cs')  
'c'
```

Return string.

pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_pair

`abjad.tools.pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name_alphabetic_abbreviation_pair`
 New in version 1.1. Change *chromatic_pitch_class_name* to diatonic pitch-class name / alphabetic accidental abbreviation pair:

```
abjad> pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_pair('c', 's')
```

Return pair of strings. Changed in version 2.0: renamed `pitchtools.name_to_letter_accidental()` to `pitchtools.chromatic_pitch_class_name_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_pair()`.

pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name

`abjad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name.chromatic_pitch_class_number_to_chromatic_pitch_class_name`
 New in version 1.1. Change *chromatic_pitch_class_number* to chromatic pitch-class name:

```
abjad> for n in range(0, 13):
...     pc = n / 2.0
...     pitch_name = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name(pc)
...     print '%s    %s' % (pc, pitch_name)
...
0.0    c
0.5    cqs
1.0    cs
1.5    dqf
2.0    d
2.5    dqs
3.0    ef
3.5    eqf
4.0    e
4.5    eqs
5.0    f
5.5    fqs
6.0    fs
```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name()`.

pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats

`abjad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats`
 New in version 1.1. Change chromatic pitch-class number to chromatic pitch-class name with flats:

```
abjad> for n in range(13):
...     pc = n / 2.0
...     name = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats(pc)
...     print '%s    %s' % (pc, name)
...
0.0    c
0.5    dtqf
1.0    df
1.5    dqf
2.0    d
2.5    etqf
3.0    ef
```

```

3.5  eqf
4.0  e
4.5  fqf
5.0  f
5.5  gtqf
6.0  gf

```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name_flats()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_flats()`.

`pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps`

`abjad.tools.pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps`

New in version 1.1. Change *chromatic_pitch_class_number* to chromatic pitch-class name with sharps:

```

abjad> for n in range(13):
...     pc = n / 2.0
...     name = pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps
...     print '%s  %s' % (pc, name)
...
0.0  c
0.5  cqs
1.0  cs
1.5  ctqs
2.0  d
2.5  dqs
3.0  ds
3.5  dtqs
4.0  e
4.5  eqs
5.0  f
5.5  fqs
6.0  fs

```

Return string. Changed in version 2.0: renamed `pitchtools.pc_to_pitch_name_sharps()` to `pitchtools.chromatic_pitch_class_number_to_chromatic_pitch_class_name_with_sharps()`.

`pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number`

`abjad.tools.pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number`. **chromatic**

New in version 2.0. Change *chromatic_pitch_class_number* to diatonic pitch-class number:

```

abjad> pitchtools.chromatic_pitch_class_number_to_diatonic_pitch_class_number(1)
0

```

Return integer.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name`

`abjad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name`. **chromatic_pitch**

New in version 2.0. Change *chromatic_pitch_name* to chromatic pitch-class name:

```

abjad> pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_name("cs' ")
'cs'

```

Return string.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number`

`abjad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number.chromatic_pitch_name_to_chromatic_pitch_class_number`

New in version 2.0. Change *chromatic_class_name* to chromatic pitch-class-number:

```
abjad> pitchtools.chromatic_pitch_name_to_chromatic_pitch_class_number("cs' ")
1
```

Return integer or float.

`pitchtools.chromatic_pitch_name_to_chromatic_pitch_number`

`abjad.tools.pitchtools.chromatic_pitch_name_to_chromatic_pitch_number.chromatic_pitch_name_to_chromatic_pitch_number`

New in version 2.0. Change *chromatic_pitch_name* to chromatic pitch number:

```
abjad> pitchtools.chromatic_pitch_name_to_chromatic_pitch_number("cs' ")
13
```

Return integer or float.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name`

`abjad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name.chromatic_pitch_name_to_diatonic_pitch_class_name`

New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch name:

```
abjad> pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_name("cs' ")
'c'
```

Return string.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number`

`abjad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number.chromatic_pitch_name_to_diatonic_pitch_class_number`

New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch-class number:

```
abjad> pitchtools.chromatic_pitch_name_to_diatonic_pitch_class_number("cs' ")
0
```

Return integer.

`pitchtools.chromatic_pitch_name_to_diatonic_pitch_name`

`abjad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_name.chromatic_pitch_name_to_diatonic_pitch_name`

New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch name:

```
abjad> pitchtools.chromatic_pitch_name_to_diatonic_pitch_name("cs' ")
"c' "
```

Return string.

pitchtools.chromatic_pitch_name_to_diatonic_pitch_number

`abjad.tools.pitchtools.chromatic_pitch_name_to_diatonic_pitch_number.chromatic_pitch_name_to_diatonic_pitch_number`
New in version 2.0. Change *chromatic_pitch_name* to diatonic pitch number:

```
abjad> pitchtools.chromatic_pitch_name_to_diatonic_pitch_number("cs'")
7
```

Return integer.

pitchtools.chromatic_pitch_name_to_octave_number

`abjad.tools.pitchtools.chromatic_pitch_name_to_octave_number.chromatic_pitch_name_to_octave_number`
New in version 2.0. Change *chromatic_pitch_name* to octave number:

```
abjad> pitchtools.chromatic_pitch_name_to_octave_number('cs')
3
```

Return integer.

pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list

`abjad.tools.pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list.chromatic_pitch_names_string_to_named_chromatic_pitch_list`
New in version 2.0. Change *chromatic_pitch_names_string* to named chromatic pitch list:

```
abjad> pitchtools.chromatic_pitch_names_string_to_named_chromatic_pitch_list("cs, cs cs' cs'")
[NamedChromaticPitch('cs,'), NamedChromaticPitch('cs'), NamedChromaticPitch("cs'"), NamedChromaticPitch("cs'")]
```

Return list of named chromatic pitches.

pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number

`abjad.tools.pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number.chromatic_pitch_number_and_accidental_semitones_to_octave_number`

New in version 1.1. Change *chromatic_pitch_number* and *accidental_semitones* to octave number:

```
abjad> pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number(12, -2)
5
```

Return integer. Changed in version 2.0: renamed `pitchtools.pitch_number_and_accidental_semitones_to_octave_number` to `pitchtools.chromatic_pitch_number_and_accidental_semitones_to_octave_number()`.

pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_alphabetic_accidental_abbreviation_octave_number_pair

`abjad.tools.pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_alphabetic_accidental_abbreviation_octave_number_pair`

New in version 1.1. Change *chromatic_pitch_number* and *diatonic_pitch_class_name* to alphabetic accidental abbreviation / octave number pair:

```
abjad> pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_alphabetic_accidental_abbreviation_octave_number_triple(
('ss', 5)
```

Return pair. Changed in version 2.0: renamed `pitchtools.number_letter_to_accidental_octave()` to `pitchtools.chromatic_pitch_number_diatonic_pitch_class_name_to_alphabetic_accidental_abbreviation_octave_number_triple()`.

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number`

`abjad.tools.pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number.chromatic_pitch_number_to_chromatic_pitch_class_number`
New in version 2.0. Change *chromatic_pitch_number* to chromatic pitch-class number:

```
abjad> pitchtools.chromatic_pitch_number_to_chromatic_pitch_class_number(13)
1
```

Return integer or float.

`pitchtools.chromatic_pitch_number_to_chromatic_pitch_name`

`abjad.tools.pitchtools.chromatic_pitch_number_to_chromatic_pitch_name.chromatic_pitch_number_to_chromatic_pitch_name`

New in version 2.0. Change *chromatic_pitch_number* to chromatic pitch name:

```
abjad> pitchtools.chromatic_pitch_number_to_chromatic_pitch_name(13)
"cs'"
```

Return string.

`pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_octave_number_triple`

`abjad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_octave_number_triple`

Change *chromatic_pitch_number* to diatonic pitch-class name / alphabetic accidental abbreviation / octave number triple:

```
abjad> pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_octave_number_triple(
('c', 's', 5)
```

Return tuple. Changed in version 2.0: renamed `pitchtools.number_to_letter_accidental_octave()` to `pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_name_alphabetic_accidental_abbreviation_octave_number_triple()`.

`pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number`

`abjad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number.chromatic_pitch_number_to_diatonic_pitch_class_number`
New in version 2.0. Change *chromatic_pitch_number* to diatonic pitch-class number:

```
abjad> pitchtools.chromatic_pitch_number_to_diatonic_pitch_class_number(13)
0
```

Return integer.

`pitchtools.chromatic_pitch_number_to_diatonic_pitch_number`

`abjad.tools.pitchtools.chromatic_pitch_number_to_diatonic_pitch_number`. **chromatic_pitch_number_to_diatonic_pitch_number**

New in version 2.0. Change *chromatic_pitch_number* to diatonic pitch number:

```
abjad> pitchtools.chromatic_pitch_number_to_diatonic_pitch_number(13)
7
```

Return integer.

`pitchtools.chromatic_pitch_number_to_octave_number`

`abjad.tools.pitchtools.chromatic_pitch_number_to_octave_number`. **chromatic_pitch_number_to_octave_number**

New in version 1.1. Change *chromatic_pitch_number* to octave number:

```
abjad> pitchtools.chromatic_pitch_number_to_octave_number(13)
5
```

Return integer. Changed in version 2.0: renamed `pitchtools.pitch_number_to_octave()` to `pitchtools.chromatic_pitch_number_to_octave_number()`.

`pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch`

`abjad.tools.pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch`. **clef_and_staff_position_number_to_named_chromatic_pitch**

New in version 2.0. Change *clef* and *staff_position_number* to named chromatic pitch:

```
abjad> clef = contexttools.ClefMark('treble')
abjad> for n in range(-6, 6):
...     pitch = pitchtools.clef_and_staff_position_number_to_named_chromatic_pitch(clef, n)
...     print '%s\t%s\t%s' % (clef.clef_name, n, pitch)
treble  -6 c'
treble  -5 d'
treble  -4 e'
treble  -3 f'
treble  -2 g'
treble  -1 a'
treble   0 b'
treble   1 c''
treble   2 d''
treble   3 e''
treble   4 f''
treble   5 g''
```

Return named chromatic pitch.

`pitchtools.diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_interval`

`abjad.tools.pitchtools.diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_interval`

New in version 2.0. Change *diatonic_interval_number* and *chromatic_interval_number* to melodic diatonic interval:


```
abjad> pitchtools.diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_int
MelodicDiatonicInterval('+m2')
```

Return melodic diatonic interval.

`pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number`

```
abjad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number.diatonic_p
New in version 1.1. Change diatonic_pitch_class_name to chromatic pitch-class number:
```

```
abjad> pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number('f')
5
```

Return integer.

`pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number`

```
abjad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number.diatonic_p
New in version 2.0. Change diatonic_pitch_class_name to diatonic pitch-class number:
```

```
abjad> pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_class_number('c')
0
```

Return integer.

`pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number`

```
abjad.tools.pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number.diatonic
New in version 2.0. Change diatonic_pitch_class_number to chromatic pitch-class number:
```

```
abjad> pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number(6)
11
```

Return nonnegative integer.

`pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name`

```
abjad.tools.pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name.diatonic_p
New in version 2.0. Change diatonic_pitch_class_number to diatonic pitch-class name:
```

```
abjad> pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name(0)
'c'
```

Return string.

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name`

```
abjad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name.diatonic_pitch_na
New in version 2.0. Change diatonic_pitch_name to chromatic pitch-class name:
```

```
abjad> pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_name("c' ")
'c'
```

Return string.

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number`

`abjad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number.diatonic_pitch_name_to_chromatic_pitch_class_number`

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch-class number:

```
abjad> pitchtools.diatonic_pitch_name_to_chromatic_pitch_class_number("c'")
0
```

Return integer.

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_name`

`abjad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_name.diatonic_pitch_name_to_chromatic_pitch_name`

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch name:

```
abjad> pitchtools.diatonic_pitch_name_to_chromatic_pitch_name("c'")
"c'"
```

Return string.

`pitchtools.diatonic_pitch_name_to_chromatic_pitch_number`

`abjad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_number.diatonic_pitch_name_to_chromatic_pitch_number`

New in version 2.0. Change *diatonic_pitch_name* to chromatic pitch number:

```
abjad> pitchtools.diatonic_pitch_name_to_chromatic_pitch_number("c'")
12
```

Return integer.

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name`

`abjad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name.diatonic_pitch_name_to_diatonic_pitch_class_name`

New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch-class name:

```
abjad> pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_name("c'")
'c'
```

Return string.

`pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number`

`abjad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number.diatonic_pitch_name_to_diatonic_pitch_class_number`

New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch-class number:

```
abjad> pitchtools.diatonic_pitch_name_to_diatonic_pitch_class_number("c'")
0
```

Return integer.

pitchtools.diatonic_pitch_name_to_diatonic_pitch_number

`abjad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_number.diatonic_pitch_name_to`
New in version 2.0. Change *diatonic_pitch_name* to diatonic pitch number:

```
abjad> pitchtools.diatonic_pitch_name_to_diatonic_pitch_number("c'")
7
```

Return integer.

pitchtools.diatonic_pitch_number_to_chromatic_pitch_number

`abjad.tools.pitchtools.diatonic_pitch_number_to_chromatic_pitch_number.diatonic_pitch_numbe`
New in version 2.0. Change *diatonic_pitch_number* to chromatic pitch number:

```
abjad> pitchtools.diatonic_pitch_number_to_chromatic_pitch_number(7)
12
```

Return integer.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name

`abjad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name.diatonic_pitch_nu`
New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch-class name:

```
abjad> pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_name(7)
'c'
```

Return string.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number

`abjad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number.diatonic_pitch`
New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch-class number:

```
abjad> pitchtools.diatonic_pitch_number_to_diatonic_pitch_class_number(7)
0
```

Return nonnegative integer.

pitchtools.diatonic_pitch_number_to_diatonic_pitch_name

`abjad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_name.diatonic_pitch_number_t`
New in version 2.0. Change *diatonic_pitch_number* to diatonic pitch name:

```
abjad> pitchtools.diatonic_pitch_number_to_diatonic_pitch_name(7)
"c' "
```

Return string.

`pitchtools.expr_has_duplicate_named_chromatic_pitch`

`abjad.tools.pitchtools.expr_has_duplicate_named_chromatic_pitch.expr_has_duplicate_named_chromatic_pitch`

New in version 2.0. True when *expr* has duplicate named chromatic pitch. Otherwise false:

```
abjad> chord = Chord([13, 13, 14], (1, 4))
abjad> pitchtools.expr_has_duplicate_named_chromatic_pitch(chord)
True
```

Return boolean.

`pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class`

`abjad.tools.pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class.expr_has_duplicate_numbered_chromatic_pitch_class`

New in version 2.0. True when *expr* has duplicate numbered chromatic pitch-class. Otherwise false:

```
abjad> chord = Chord([1, 13, 14], (1, 4))
abjad> pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class(chord)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.expr_has_duplicate_numeric_chromatic_pitch_class` to `pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class`.

`pitchtools.expr_to_melodic_chromatic_interval_segment`

`abjad.tools.pitchtools.expr_to_melodic_chromatic_interval_segment.expr_to_melodic_chromatic_interval_segment`

New in version 2.0. Change *expr* to melodic chromatic interval segment:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> pitchtools.expr_to_melodic_chromatic_interval_segment(staff)
MelodicChromaticIntervalSegment(+2, +2, +1, +2, +2, +2, +1)
```

Return melodic chromatic interval segment.

`pitchtools.get_named_chromatic_pitch_from_pitch_carrier`

`abjad.tools.pitchtools.get_named_chromatic_pitch_from_pitch_carrier.get_named_chromatic_pitch_from_pitch_carrier`

New in version 1.1. Get named chromatic pitch from *pitch_carrier*:

```
abjad> pitch = pitchtools.NamedChromaticPitch('df', 5)
abjad> pitch
NamedChromaticPitch("df'")
abjad> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(pitch)
NamedChromaticPitch("df'")

abjad> note = Note(('df', 5), (1, 4))
abjad> note
Note("df''4")
abjad> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(note)
NamedChromaticPitch("df'")

abjad> note = Note(('df', 5), (1, 4))
abjad> note.note_head
NoteHead("df'")
```

```
abjad> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(note.note_head)
NamedChromaticPitch("df'")
```

```
abjad> chord = Chord([('df', 5)], (1, 4))
abjad> chord
Chord("<df' '>4")
abjad> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(chord)
NamedChromaticPitch("df'")
```

```
abjad> pitchtools.get_named_chromatic_pitch_from_pitch_carrier(13)
NamedChromaticPitch("cs'")
```

Raise missing pitch error when *pitch_carrier* carries no pitch.

Raise extra pitch error when *pitch_carrier* carries more than one pitch.

Return named chromatic pitch. Changed in version 2.0: renamed `pitchtools.get_pitch()` to `pitchtools.get_named_chromatic_pitch_from_pitch_carrier()`.

`pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier`

```
abjad.tools.pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier.get_numbered_c
```

New in version 2.0. Get numbered chromatic pitch-class from *pitch_carrier*:

```
abjad> note = Note("cs' 4")
abjad> pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier(note)
NumberedChromaticPitchClass(1)
```

Raise missing pitch error on empty chords.

Raise extra pitch error on many-note chords.

Return numbered chromatic pitch-class. Changed in version 2.0: renamed `pitchtools.get_numeric_chromatic_pitch_class_from_pitch_carrier()` to `pitchtools.get_numbered_chromatic_pitch_class_from_pitch_carrier()`.

`pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list`

```
abjad.tools.pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number
```

New in version 1.1. Insert and transpose nested subruns in *chromatic_pitch_class_number_list* according to *subrun_indicators*:

```
abjad> notes = [Note(p, (1, 4)) for p in [0, 2, 7, 9, 5, 11, 4]]
abjad> subrun_indicators = [(0, [2, 4]), (4, [3, 1])]
abjad> pitchtools.insert_and_transpose_nested_subruns_in_chromatic_pitch_class_number_list(notes)

abjad> t = []
abjad> for x in notes:
...     try:
...         t.append(x.written_pitch.chromatic_pitch_number)
...     except AttributeError:
...         t.append([y.written_pitch.chromatic_pitch_number for y in x])

abjad> t
[0, [5, 7], 2, [4, 0, 6, 11], 7, 9, 5, [10, 6, 8], 11, [7], 4]
```

Set *subrun_indicators* to a list of zero or more (index, length_list) pairs.

For each (index, length_list) pair in *subrun_indicators* the function will read *index* mod len(notes) and insert a subrun of length length_list[0] immediately after notes[index], a subrun of length length_list[1] immediately after notes[index+1], and, in general, a subrun of length_list[i] immediately after notes[index+i], for i < length(length_list).

New subruns are wrapped with lists. These wrapper lists are designed to allow inspection of the structural changes to *notes* immediately after the function returns. For this reason most calls to this function will be followed by notes = sequencetools.flatten_sequence(notes):

```
abjad> from abjad.tools import sequencetools
abjad> notes = sequencetools.flatten_sequence(notes)
abjad> notes
[Note("c'4"), Note("f'4"), Note("g'4"), Note("d'4"), Note("e'4"), Note("c'4"), Note("fs'4"), Note("a'4")]
```

This function is designed to work on a built-in Python list of notes. This function is **not** designed to work on Abjad voices, staves or other containers because the function currently implements no spanner-handling. That is, this function is designed to be used during precomposition when other, similar abstract pitch transforms may be common.

Return list of integers and / or floats. Changed in version 2.0: renamed pitchtools.insert_transposed_pc_subruns() to pitchtools.insert_and_transpose_nested_subruns()

pitchtools.instantiate_pitch_and_interval_test_collection

abjad.tools.pitchtools.instantiate_pitch_and_interval_test_collection.**instantiate_pitch_and_interval_test_collection**
New in version 2.0. Instantiate pitch and interval test collection:

```
abjad> for x in pitchtools.instantiate_pitch_and_interval_test_collection(): x
...
HarmonicChromaticInterval(1)
HarmonicChromaticIntervalClass(1)
HarmonicCounterpointInterval(1)
HarmonicCounterpointIntervalClass(1)
HarmonicDiatonicInterval('M2')
HarmonicDiatonicIntervalClass('M2')
InversionEquivalentChromaticIntervalClass(1)
InversionEquivalentDiatonicIntervalClass('M2')
MelodicChromaticInterval(+1)
MelodicChromaticIntervalClass(+1)
MelodicCounterpointInterval(1)
MelodicCounterpointIntervalClass(+1)
MelodicDiatonicInterval('+M2')
MelodicDiatonicIntervalClass('+M2')
NamedChromaticPitch('c')
NamedChromaticPitchClass('c')
NamedDiatonicPitch('c')
NamedDiatonicPitchClass('c')
NumberedChromaticPitch(1)
NumberedChromaticPitchClass(1)
NumberedDiatonicPitch(1)
NumberedDiatonicPitchClass(1)
```

Use to test pitch and interval interface consistency.

Return list.

pitchtools.inventory_aggregate_subsets

`abjad.tools.pitchtools.inventory_aggregate_subsets.inventory_aggregate_subsets()`

New in version 2.0. Inventory aggregate subsets:

```
abjad> U_star = pitchtools.inventory_aggregate_subsets()
abjad> len(U_star)
4096
abjad> for pcset in U_star[:20]:
...     pcset
NumberedChromaticPitchClassSet([])
NumberedChromaticPitchClassSet([0])
NumberedChromaticPitchClassSet([1])
NumberedChromaticPitchClassSet([0, 1])
NumberedChromaticPitchClassSet([2])
NumberedChromaticPitchClassSet([0, 2])
NumberedChromaticPitchClassSet([1, 2])
NumberedChromaticPitchClassSet([0, 1, 2])
NumberedChromaticPitchClassSet([3])
NumberedChromaticPitchClassSet([0, 3])
NumberedChromaticPitchClassSet([1, 3])
NumberedChromaticPitchClassSet([0, 1, 3])
NumberedChromaticPitchClassSet([2, 3])
NumberedChromaticPitchClassSet([0, 2, 3])
NumberedChromaticPitchClassSet([1, 2, 3])
NumberedChromaticPitchClassSet([0, 1, 2, 3])
NumberedChromaticPitchClassSet([4])
NumberedChromaticPitchClassSet([0, 4])
NumberedChromaticPitchClassSet([1, 4])
NumberedChromaticPitchClassSet([0, 1, 4])
```

There are 4096 subsets of the aggregate.

This is U^* in [Morris 1987].

Return list of numbered chromatic pitch-class sets.

pitchtools.inventory_inversion_equivalent_diatonic_interval_classes

`abjad.tools.pitchtools.inventory_inversion_equivalent_diatonic_interval_classes.inventory_inversion_equivalent_diatonic_interval_classes()`

New in version 2.0. Inventory inversion-equivalent diatonic interval-classes:

```
abjad> for dic in pitchtools.inventory_inversion_equivalent_diatonic_interval_classes():
...     dic
...
InversionEquivalentDiatonicIntervalClass('P1')
InversionEquivalentDiatonicIntervalClass('aug1')
InversionEquivalentDiatonicIntervalClass('m2')
InversionEquivalentDiatonicIntervalClass('M2')
InversionEquivalentDiatonicIntervalClass('aug2')
InversionEquivalentDiatonicIntervalClass('dim3')
InversionEquivalentDiatonicIntervalClass('m3')
InversionEquivalentDiatonicIntervalClass('M3')
InversionEquivalentDiatonicIntervalClass('dim4')
InversionEquivalentDiatonicIntervalClass('P4')
InversionEquivalentDiatonicIntervalClass('aug4')
```

There are 11 inversion-equivalent diatonic interval-classes.

It is an open question as to whether octaves should be included.

Return list of inversion-equivalent diatonic interval-classes.

pitchtools.is_alphabetic_accidental_abbreviation

`abjad.tools.pitchtools.is_alphabetic_accidental_abbreviation.is_alphabetic_accidental_abbrev`

New in version 2.0. True when *expr* is an alphabetic accidental abbreviation. Otherwise false:

```
abjad> pitchtools.is_alphabetic_accidental_abbreviation('tqs')
True
```

The regex `^([s]{1,2}|[f]{1,2}|t?q?[fs])!?$` underlies this predicate.

Return boolean.

pitchtools.is_chromatic_pitch_class_name

`abjad.tools.pitchtools.is_chromatic_pitch_class_name.is_chromatic_pitch_class_name(expr)`

New in version 2.0. True when *expr* is a chromatic pitch-class name. Otherwise false:

```
abjad> pitchtools.is_chromatic_pitch_class_name('fs')
True
```

The regex `^([a-g,A-G])((([s]{1,2}|[f]{1,2}|t?q?[fs])|)!)$` underlies this predicate.

Return boolean.

pitchtools.is_chromatic_pitch_class_name_octave_number_pair

`abjad.tools.pitchtools.is_chromatic_pitch_class_name_octave_number_pair.is_chromatic_pitch`

New in version 1.1. True when *arg* has the form of a chromatic pitch-class / octave number pair. Otherwise false:

```
abjad> pitchtools.is_chromatic_pitch_class_name_octave_number_pair(('cs', 5))
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_pair()` to `pitchtools.is_chromatic_pitch_class_name_octave_number_pair()`.

pitchtools.is_chromatic_pitch_class_number

`abjad.tools.pitchtools.is_chromatic_pitch_class_number.is_chromatic_pitch_class_number(expr)`

New in version 2.0. True *expr* is a chromatic pitch-class number. Otherwise false:

```
abjad> pitchtools.is_chromatic_pitch_class_number(1)
True
```

The chromatic pitch-class numbers are equal to the set `[0, 0.5, ..., 11, 11.5]`.

Return boolean.

`pitchtools.is_chromatic_pitch_name`

`abjad.tools.pitchtools.is_chromatic_pitch_name.is_chromatic_pitch_name(expr)`

New in version 2.0. True *expr* is a chromatic pitch name. Otherwise false:

```
abjad> pitchtools.is_chromatic_pitch_name('c,')
True
```

The regex `^([a-g,A-G])((([s]{1,2}|[f]{1,2}|t?q?[f,s])!?)|(|'+|))$` underlies this predicate.

Return boolean.

`pitchtools.is_chromatic_pitch_number`

`abjad.tools.pitchtools.is_chromatic_pitch_number.is_chromatic_pitch_number(expr)`

New in version 2.0. True *expr* is a chromatic pitch number. Otherwise false:

```
abjad> pitchtools.is_chromatic_pitch_number(13)
True
```

The chromatic pitch numbers are equal to the set of all integers in union with the set of all integers plus or minus 0.5.

Return boolean.

`pitchtools.is_diatonic_pitch_class_name`

`abjad.tools.pitchtools.is_diatonic_pitch_class_name.is_diatonic_pitch_class_name(expr)`

New in version 2.0. True when *expr* is a diatonic pitch-class name. Otherwise false:

```
abjad> pitchtools.is_diatonic_pitch_class_name('c')
True
```

The regex `^[a-g,A-G]$` underlies this predicate.

Return boolean.

`pitchtools.is_diatonic_pitch_class_number`

`abjad.tools.pitchtools.is_diatonic_pitch_class_number.is_diatonic_pitch_class_number(expr)`

New in version 2.0. True when *expr* is a diatonic pitch-class number. Otherwise false:

```
abjad> pitchtools.is_diatonic_pitch_class_number(0)
True
```

The diatonic pitch-class numbers are equal to the set `[0, 1, 2, 3, 4, 5, 6]`.

Return boolean.

`pitchtools.is_diatonic_pitch_name`

`abjad.tools.pitchtools.is_diatonic_pitch_name.is_diatonic_pitch_name(expr)`

New in version 2.0. True when *expr* is a diatonic pitch name. Otherwise false:

```
abjad> pitchtools.is_diatonic_pitch_name("c' ")
True
```

The regex `(^[a-g,A-G])(, +|' +|)$` underlies this predicate.

Return boolean.

pitchtools.is_diatonic_pitch_number

`abjad.tools.pitchtools.is_diatonic_pitch_number.is_diatonic_pitch_number(expr)`
New in version 2.0. True when *expr* is a diatonic pitch number. Otherwise false:

```
abjad> pitchtools.is_diatonic_pitch_number(7)
True
```

The diatonic pitch numbers are equal to the set of integers.

Return boolean.

pitchtools.is_diatonic_quality_abbreviation

`abjad.tools.pitchtools.is_diatonic_quality_abbreviation.is_diatonic_quality_abbreviation(expr)`
New in version 2.0. True when *expr* is a diatonic quality abbreviation. Otherwise false:

```
abjad> pitchtools.is_diatonic_quality_abbreviation('aug')
True
```

The regex `^M|m|P|aug|dim$` underlies this predicate.

Return boolean.

pitchtools.is_harmonic_diatonic_interval_abbreviation

`abjad.tools.pitchtools.is_harmonic_diatonic_interval_abbreviation.is_harmonic_diatonic_interval_abbreviation(expr)`
New in version 2.0. True when *expr* is a harmonic diatonic interval abbreviation. Otherwise false:

```
abjad> pitchtools.is_harmonic_diatonic_interval_abbreviation('M9')
True
```

The regex `^(M|m|P|aug|dim)(\d+)$` underlies this predicate.

Return boolean.

pitchtools.is_melodic_diatonic_interval_abbreviation

`abjad.tools.pitchtools.is_melodic_diatonic_interval_abbreviation.is_melodic_diatonic_interval_abbreviation(expr)`
New in version 2.0. True when *expr* is a melodic diatonic interval abbreviation. Otherwise false:

```
abjad> pitchtools.is_melodic_diatonic_interval_abbreviation('+M9')
True
```

The regex `^([+,-]?)(M|m|P|aug|dim)(\d+)$` underlies this predicate.

Return boolean.

`pitchtools.is_named_chromatic_pitch_token`

`abjad.tools.pitchtools.is_named_chromatic_pitch_token.is_named_chromatic_pitch_token(pitch_token)`
New in version 1.1. True when *pitch_token* has the form of an Abjad pitch token. Otherwise false:

```
abjad> pitchtools.is_named_chromatic_pitch_token('c', 4)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_pitch_token()` to `pitchtools.is_named_chromatic_pitch_token()`.

`pitchtools.is_octave_tick_string`

`abjad.tools.pitchtools.is_octave_tick_string.is_octave_tick_string(expr)`
New in version 2.0. True when *expr* is an octave tick string. Otherwise false:

```
abjad> pitchtools.is_octave_tick_string(',,,')
True
```

The regex `^,+|'|+$` underlies this predicate.

Return boolean.

`pitchtools.is_pitch_carrier`

`abjad.tools.pitchtools.is_pitch_carrier.is_pitch_carrier(expr)`
New in version 1.1. True when *expr* is an Abjad pitch, note, note-head of chord instance. Otherwise false:

```
abjad> note = Note("c'4")
abjad> pitchtools.is_pitch_carrier(note)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.is_carrier()` to `pitchtools.is_pitch_carrier()`.

`pitchtools.is_pitch_class_octave_number_string`

`abjad.tools.pitchtools.is_pitch_class_octave_number_string.is_pitch_class_octave_number_string(expr)`
New in version 2.5. True when *expr* is a pitch-class / octave number string. Otherwise false:

```
abjad> pitchtools.is_pitch_class_octave_number_string('C#2')
True
```

Quartertone accidentals are supported.

The regex `^([A-G])([#]{1,2}|[b]{1,2}|[#]?[+]|[b]?[~]|)([-]?[0-9]+)$` underlies this predicate.

Return boolean.

`pitchtools.is_symbolic_accidental_string`

`abjad.tools.pitchtools.is_symbolic_accidental_string.is_symbolic_accidental_string(expr)`
New in version 2.5. True when *expr* is a symbolic accidental string. Otherwise false:

```
abjad> pitchtools.is_symbolic_accidental_string('#+')
True
```

True on empty string.

The regex `^([\#\{1,2\}|\[b\]{1,2\}|[\#]?[+]|[\b]?[~]|)\$` underlies this predicate.

Return boolean.

`pitchtools.is_symbolic_pitch_range_string`

`abjad.tools.pitchtools.is_symbolic_pitch_range_string.is_symbolic_pitch_range_string(expr)`
 New in version 2.5. True when *expr* is a symbolic pitch range string. Otherwise false:

```
abjad> pitchtools.is_symbolic_pitch_range_string('[A0, C8]')
True
```

The regex that underlies this predicate matches against two comma-separated pitch indicators enclosed in some combination of square brackets and round parentheses.

Return boolean.

`pitchtools.iterate_named_chromatic_pitch_pairs_forward_in_expr`

`abjad.tools.pitchtools.iterate_named_chromatic_pitch_pairs_forward_in_expr.iterate_named_chromatic_pitch_pairs_forward_in_expr(expr)`
 New in version 2.0. Iterate left-to-right, top-to-bottom named chromatic pitch pairs in *expr*:

```
abjad> score = Score([])
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'4")]
abjad> score.append(Staff(notes))
abjad> notes = [Note(x, (1, 4)) for x in [-12, -15, -17]]
abjad> score.append(Staff(notes))
abjad> contexttools.ClefMark('bass')(score[1])
ClefMark('bass')(Staff{3})
```

```
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
    g'4
  }
  \new Staff {
    \clef "bass"
    c4
    a,4
    g,4
  }
>>
```

```
abjad> for pair in pitchtools.iterate_named_chromatic_pitch_pairs_forward_in_expr(score):
...     pair
...
(NamedChromaticPitch("c'"), NamedChromaticPitch('c'))
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'))
```

```
(NamedChromaticPitch('c'), NamedChromaticPitch("d'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch('a,'))
(NamedChromaticPitch('c'), NamedChromaticPitch("e'"))
(NamedChromaticPitch('c'), NamedChromaticPitch('a,'))
(NamedChromaticPitch("e'"), NamedChromaticPitch('a,'))
(NamedChromaticPitch("e'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch('a,'), NamedChromaticPitch("f'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch('g,'))
(NamedChromaticPitch('a,'), NamedChromaticPitch("g'"))
(NamedChromaticPitch('a,'), NamedChromaticPitch('g,'))
(NamedChromaticPitch("g'"), NamedChromaticPitch('g,'))
```

Chords are handled correctly.

```
abjad> chord_1 = Chord([0, 2, 4], (1, 4))
abjad> chord_2 = Chord([17, 19], (1, 4))
abjad> staff = Staff([chord_1, chord_2])

abjad> f(staff)
\new Staff {
    <c' d' e'>4
    <f' g'>4
}

abjad> for pair in pitchtools.iterate_named_chromatic_pitch_pairs_forward_in_expr(staff):
...     print pair
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("e'"), NamedChromaticPitch("f'"))
(NamedChromaticPitch("e'"), NamedChromaticPitch("g'"))
(NamedChromaticPitch("f'"), NamedChromaticPitch("g'"))
```

Return generator.

`pitchtools.list_chromatic_pitch_numbers_in_expr`

`abjad.tools.pitchtools.list_chromatic_pitch_numbers_in_expr.list_chromatic_pitch_numbers_in_expr`
 New in version 2.0. List chromatic pitch numbers in *expr*:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> pitchtools.list_chromatic_pitch_numbers_in_expr(tuplet)
(0, 2, 4)
```

Return tuple of zero or more numbers.

`pitchtools.list_harmonic_chromatic_intervals_in_expr`

`abjad.tools.pitchtools.list_harmonic_chromatic_intervals_in_expr.list_harmonic_chromatic_intervals_in_expr`
 New in version 2.0. List harmonic chromatic intervals in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> for interval in sorted(pitchtools.list_harmonic_chromatic_intervals_in_expr(staff)):
...     interval
...
HarmonicChromaticInterval(1)
HarmonicChromaticInterval(2)
HarmonicChromaticInterval(2)
HarmonicChromaticInterval(3)
HarmonicChromaticInterval(4)
HarmonicChromaticInterval(5)
```

Return unordered set.

`pitchtools.list_harmonic_diatonic_intervals_in_expr`

`abjad.tools.pitchtools.list_harmonic_diatonic_intervals_in_expr.list_harmonic_diatonic_intervals_in_expr`

New in version 2.0. List harmonic diatonic intervals in *expr*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> for interval in sorted(pitchtools.list_harmonic_diatonic_intervals_in_expr(staff)):
...     interval
...
HarmonicDiatonicInterval('m2')
HarmonicDiatonicInterval('M2')
HarmonicDiatonicInterval('M2')
HarmonicDiatonicInterval('m3')
HarmonicDiatonicInterval('M3')
HarmonicDiatonicInterval('P4')
```

Return unordered set.

`pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers`

`abjad.tools.pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers`

New in version 2.0. List inversion-equivalent chromatic interval-classes pairwise between *pitch_carriers*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
}

abjad> pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers(staff)
[InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(1)]
```

```

abjad> pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers(
[InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(1), InversionEquivalentChromaticIntervalClass(0)]

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8"), Note("c''8")]
abjad> notes.reverse()
abjad> notes
[Note("c''8"), Note("b'8"), Note("a'8"), Note("g'8"), Note("f'8"), Note("e'8"), Note("d'8"), Note("c'8")]

abjad> pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers(
[InversionEquivalentChromaticIntervalClass(1), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(1), InversionEquivalentChromaticIntervalClass(2)]

abjad> pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers(
[InversionEquivalentChromaticIntervalClass(1), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(1), InversionEquivalentChromaticIntervalClass(2), InversionEquivalentChromaticIntervalClass(0)]

```

When `wrap = False` do not return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

When `wrap = True` do return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

Return list.

`pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers`

`abjad.tools.pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers`

New in version 1.1. List melodic chromatic interval numbers pairwise between *pitch_carriers*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> print staff.format
\new Staff {
    c'8
    d'8
    e'8
    f'8
    g'8
    a'8
    b'8
    c''8
}

abjad> pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers(staff)
[2, 2, 1, 2, 2, 2, 1]

abjad> pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers(staff, wrap=True)
[2, 2, 1, 2, 2, 2, 1, -12]

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8"), Note("g'8"), Note("a'8"), Note("b'8"), Note("c''8")]
abjad> notes.reverse()
abjad> notes
[Note("c''8"), Note("b'8"), Note("a'8"), Note("g'8"), Note("f'8"), Note("e'8"), Note("d'8"), Note("c'8")]

```

```
abjad> pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers(notes)
[-1, -2, -2, -2, -1, -2, -2]
```

```
abjad> pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers(notes,
[-1, -2, -2, -2, -1, -2, -2, 12]
```

When `wrap = False` do not return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

When `wrap = True` do return `pitch_carriers[-1] - pitch_carriers[0]` as last in series.

Return list. Changed in version 2.0: renamed `pitchtools.get_signed_interval_series()` to `pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers()`.

`pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class`

`abjad.tools.pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class`
 New in version 2.0. List named chromatic pitch carriers in *expr* sorted by numbered chromatic pitch-class:

```
abjad> chord = Chord([9, 11, 12, 14, 16], (1, 4))
abjad> notes = chordtools.arpeggiate_chord(chord)
abjad> pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class(notes)
[Note("c' 4"), Note("d' 4"), Note("e' 4"), Note("a' 4"), Note("b' 4")]
```

The elements in *pitch_carriers* are not changed in any way.

Return list. Changed in version 2.0: renamed `pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class` to `pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class`.

`pitchtools.list_named_chromatic_pitches_in_expr`

`abjad.tools.pitchtools.list_named_chromatic_pitches_in_expr.list_named_chromatic_pitches_in_expr`
 New in version 2.0. List named chromatic pitches in *expr*:

```
abjad> t = Staff("c'4 d'4 e'4 f'4")
abjad> beam = spannertools.BeamSpanner(t[:])
abjad> pitchtools.list_named_chromatic_pitches_in_expr(beam)
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"), NamedChromaticPitch("e'"), NamedChromaticPitch("f'"))
```

Return tuple.

`pitchtools.list_numbered_chromatic_pitch_classes_in_expr`

`abjad.tools.pitchtools.list_numbered_chromatic_pitch_classes_in_expr.list_numbered_chromatic_pitch_classes_in_expr`
 New in version 2.0. List numbered chromatic pitch-classes in *expr*:

```
abjad> chord = Chord([13, 14, 15], (1, 4))
abjad> pitchtools.list_numbered_chromatic_pitch_classes_in_expr(chord)
(NumberedChromaticPitchClass(1), NumberedChromaticPitchClass(2), NumberedChromaticPitchClass(3))
```

Works with notes, chords, defective chords.

Return tuple or zero or more numbered chromatic pitch-classes. Changed in version 2.0: renamed `pitchtools.list_numeric_chromatic_pitch_classes_in_expr()` to `pitchtools.list_numbered_chromatic_pitch_classes_in_expr()`.

pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range

`abjad.tools.pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range`.**list**

New in version 1.1. List octave transpositions of *pitch_carrier* in *pitch_range*:

```
abjad> chord = Chord([0, 2, 4], (1, 4))
abjad> pitch_range = pitchtools.PitchRange(0, 48)
abjad> pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range(chord, pitch_range)
[Chord("<c' d' e'>4"), Chord("<c'' d'' e''>4"), Chord("<c''' d''' e'''>4"), Chord("<c'''' d'''' e''''>4")]
```

Return list of newly created *pitch_carrier* objects.

pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2

`abjad.tools.pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2`.**list**

New in version 2.0. List ordered named chromatic pitch pairs from *expr_1* to *expr_2*:

```
abjad> chord_1 = Chord([0, 1, 2], (1, 4))
abjad> chord_2 = Chord([3, 4], (1, 4))
abjad> for pair in pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2(chord_1, chord_2):
...     pair
(NamedChromaticPitch("c'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("e'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("e'"))
```

Return generator.

pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr

`abjad.tools.pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr`.**list_unordered_named**

New in version 2.0. List unordered named chromatic pitch pairs in *expr*:

```
abjad> for pair in pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr(Chord([0, 1, 2], (1, 4))):
...     pair
...
(NamedChromaticPitch("c'"), NamedChromaticPitch("cs'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("c'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("d'"))
(NamedChromaticPitch("cs'"), NamedChromaticPitch("ef'"))
(NamedChromaticPitch("d'"), NamedChromaticPitch("ef'"))
```

Return generator.

pitchtools.make_n_middle_c_centered_pitches

`abjad.tools.pitchtools.make_n_middle_c_centered_pitches`.**make_n_middle_c_centered_pitches**(*n*)

New in version 2.0. Make *n* middle-c centered pitches, where $0 < n$:

```

abjad> for p in pitchtools.make_n_middle_c_centered_pitches(5): p
NamedChromaticPitch('f')
NamedChromaticPitch('a')
NamedChromaticPitch("c' ")
NamedChromaticPitch("e' ")
NamedChromaticPitch("g' ")

abjad> for p in pitchtools.make_n_middle_c_centered_pitches(4): p
NamedChromaticPitch('g')
NamedChromaticPitch('b')
NamedChromaticPitch("d' ")
NamedChromaticPitch("f' ")

```

Return list of zero or more named chromatic pitches.

`pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number`

`abjad.tools.pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number`.**named_chroma**

New in version 2.0. Change named chromatic *pitch* and *clef* to staff position number:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> clef = contexttools.ClefMark('treble')
abjad> for note in staff:
...     written_pitch = note.written_pitch
...     number = pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number(written_pitch,
...     print '%s\t%s' % (written_pitch, number)
c'      -6
d'      -5
e'      -4
f'      -3
g'      -2
a'      -1
b'       0
c''      1

```

Return integer.

`pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches`

`abjad.tools.pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches`.**named_chroma**

New in version 2.0. Change named chromatic *pitch_tokens* to named chromatic pitches:

```

abjad> pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches([0, 2, ('ef', 4)])
[NamedChromaticPitch("c' "), NamedChromaticPitch("d' "), NamedChromaticPitch("ef' ")]

```

Return list of zero or more named chromatic pitches.

`pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary`

`abjad.tools.pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number`

New in version 1.1. Change named chromatic pitches to harmonic chromatic interval-class number dictionary:

```

abjad> chord = Chord([0, 2, 11], (1, 4))
abjad> vector = pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary(chord)
abjad> vector
{0: 0, 1: 0, 2: 1, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 1, 10: 0, 11: 1}

```

Return dictionary. Changed in version 2.0: renamed `pitchtools.get_interval_vector()` to `pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary()`.

`pitchtools.named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary`

`abjad.tools.pitchtools.named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary`

New in version 1.1. Change named chromatic *pitches* to inversion-equivalent chromatic interval-class number dictionary:

```

abjad> chord = Chord([0, 2, 11], (1, 4))
abjad> vector = pitchtools.named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary(chord)
abjad> for i in range(7):
...     print '\t%s\t%s' % (i, vector[i])
...
0 0
1 1
2 1
3 1
4 0
5 0
6 0

```

Changed in version 2.0: works with quartertones. Return dictionary. Changed in version 2.0: renamed `pitchtools.get_interval_class_vector()` to `pitchtools.named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary()`.

`pitchtools.octave_number_to_octave_tick_string`

`abjad.tools.pitchtools.octave_number_to_octave_tick_string.octave_number_to_octave_tick_string`

New in version 2.0. Change *octave_number* to octave tick string:

```

abjad> for octave_number in range(-1, 9):
...     print "%s\t%s" % (octave_number, pitchtools.octave_number_to_octave_tick_string(octave_number))
...
-1 ',,',,
0 ',,'
1 ',,'
2 ',,'
3 ',,'
4 ',,'
5 ',,'
6 ',,'
7 ',,'
8 ',,'

```

Raise type error on noninteger input.

Return string.

pitchtools.octave_tick_string_to_octave_number

`abjad.tools.pitchtools.octave_tick_string_to_octave_number.octave_tick_string_to_octave_number`
 New in version 2.0. Change *tick_string* to octave number:

```
abjad> pitchtools.octave_tick_string_to_octave_number("' ")
4
```

Raise type error on nonstring input.

Raise value error on input not of tick string format.

Return integer.

pitchtools.ordered_chromatic_pitch_class_numbers_are_within_ordered_chromatic_pitch_numbers

`abjad.tools.pitchtools.ordered_chromatic_pitch_class_numbers_are_within_ordered_chromatic_pitch_numbers`

New in version 1.1. True if ordered *chromatic_pitch_class_numbers* are within ordered *chromatic_pitch_numbers*:

```
abjad> pcs = [2, 7, 10]
abjad> pitches = [6, 9, 12, 13, 14, 19, 22, 27, 28, 29, 32, 35]
abjad> pitchtools.ordered_chromatic_pitch_class_numbers_are_within_ordered_chromatic_pitch_numbers(pcs, pitches)
True
```

Return boolean. Changed in version 2.0: renamed `pitchtools.are_in_octave_order()` to `pitchtools.ordered_chromatic_pitch_class_numbers_are_within_ordered_chromatic_pitch_numbers()`

pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number

`abjad.tools.pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number.pentatonic_pitch_number_to_chromatic_pitch_number`

New in version 1.1. Changed *pentatonic_scale_degree* number to chromatic pitch number:

```
abjad> for pentatonic_scale_degree in range(9): # doctest: +SKIP
...     chromatic_pitch_number = pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number(pentatonic_scale_degree)
...     print '%s\t%s' % (pentatonic_scale_degree, chromatic_pitch_number)
...
0 1
1 3
2 6
3 8
4 10
5 13
6 15
7 18
8 20
```

Pentatonic scale degrees may be negative:

```
abjad> for pentatonic_scale_degree in range(-1, -9, -1): # doctest: +SKIP
...     chromatic_pitch_number = pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number(pentatonic_scale_degree)
...     print '%s\t%s' % (pentatonic_scale_degree, chromatic_pitch_number)
```

```
...
-1 -2
-2 -4
-3 -6
-4 -9
-5 -11
-6 -14
-7 -16
-8 -18
```

Return integer. Changed in version 2.0: renamed `pitchtools.pentatonic_to_chromatic()` to `pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number()`.

`pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row`

`abjad.tools.pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row.permute`

New in version 2.0. Permute named chromatic pitch carrier list by twelve-tone *row*:

```
abjad> notes = notetools.make_notes([17, -10, -2, 11], [Duration(1, 4)])
abjad> row = pitchtools.TwelveToneRow([10, 0, 2, 6, 8, 7, 5, 3, 1, 9, 4, 11])
abjad> pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row(notes, row)
[Note('bf4'), Note('d4'), Note("f'4"), Note("b'4")]
```

Function works by reference only. No objects are cloned.

Return list.

`pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name`

`abjad.tools.pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name.pitch_class`

New in version 2.5. Change *pitch_class_octave_number_string* to chromatic pitch name:

```
abjad> pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_name('C#+2')
'ctqs,'
```

Return string.

`pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate`

`abjad.tools.pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate`

New in version 1.1. Register chromatic *pitch_class_numbers* by chromatic pitch-number *aggregate*:

```
abjad> pitchtools.register_chromatic_pitch_class_numbers_by_chromatic_pitch_number_aggregate(
...     [10, 0, 2, 6, 8, 7, 5, 3, 1, 9, 4, 11],
...     [10, 19, 20, 23, 24, 26, 27, 29, 30, 33, 37, 40])
[10, 24, 26, 30, 20, 19, 29, 27, 37, 33, 40, 23]
```

Return list of zero or more chromatic pitch numbers. Changed in version 2.0: renamed `pitchtools.registrate()` to `pitchtools.register_chromatic_pitch_class_numbers_by_chromatic`

pitchtools.respell_named_chromatic_pitches_in_expr_with_flats

abjad.tools.pitchtools.respell_named_chromatic_pitches_in_expr_with_flats.**respell_named_chromatic_pitches_in_expr_with_flats**

New in version 1.1. Respell named chromatic pitches in *expr* with flats:

```
abjad> staff = Staff(notetools.make_repeated_notes(6))
abjad> pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr(staff)
```

```
abjad> f(staff)
\new Staff {
    c'8
    cs'8
    d'8
    ef'8
    e'8
    f'8
}
```

```
abjad> pitchtools.respell_named_chromatic_pitches_in_expr_with_flats(staff)
```

```
abjad> f(staff)
\new Staff {
    c'8
    df'8
    d'8
    ef'8
    e'8
    f'8
}
```

Return `None`. Changed in version 2.0: renamed `pitchtools.make_flat()` to `pitchtools.respell_named_chromatic_pitches_in_expr_with_flats()`.

pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps

abjad.tools.pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps.**respell_named_chromatic_pitches_in_expr_with_sharps**

New in version 1.1. Respell named chromatic pitches in *expr* with sharps:

```
abjad> staff = Staff(notetools.make_repeated_notes(6))
abjad> pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr(staff)
```

```
abjad> f(staff)
\new Staff {
    c'8
    cs'8
    d'8
    ef'8
    e'8
    f'8
}
```

```
abjad> pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps(staff)
```

```
abjad> f(staff)
\new Staff {
    c'8
    cs'8
}
```

```

    d' 8
    ds' 8
    e' 8
    f' 8
}

```

Return `none`. Changed in version 2.0: renamed `pitchtools.make_sharp()` to `pitchtools.respell_named_chromatic_pitches_in_expr_with_sharps()`.

pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr

`abjad.tools.pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr`

New in version 1.1. Set ascending named chromatic pitches on nontied pitched components in *expr*:

```

abjad> staff = Voice(notetools.make_notes(0, [(5, 32)] * 4))
abjad> pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Voice {
    c' 8 ~
    c' 32
    cs' 8 ~
    cs' 32
    d' 8 ~
    d' 32
    ef' 8 ~
    ef' 32
}

```

Used primarily in generating test file examples.

Return `none`. Changed in version 2.0: renamed `pitchtools.chromaticize()` to `pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr`

pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr

`abjad.tools.pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr`

New in version 1.1. Set ascending named diatonic pitches on nontied pitched components in *expr*:

```

abjad> staff = Staff(notetools.make_notes(0, [(5, 32)] * 4))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

abjad> f(staff)
\new Staff {
    c' 8 ~
    c' 32
    d' 8 ~
    d' 32
    e' 8 ~
    e' 32
    f' 8 ~
    f' 32
}

```

Used primarily in generating test file examples. New in version 2.0: Optional *key_signature* keyword argument. Return none. Changed in version 2.0: renamed `pitchtools.diatonicize()` to `pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr()`

`pitchtools.suggest_clef_for_named_chromatic_pitches`

`abjad.tools.pitchtools.suggest_clef_for_named_chromatic_pitches.suggest_clef_for_named_chromatic_pitches`
 New in version 1.1. Suggest clef for named chromatic *pitches*:

```
abjad> staff = Staff(notetools.make_notes(range(-12, -6), [(1, 4)]))
abjad> pitchtools.suggest_clef_for_named_chromatic_pitches(staff)
ClefMark('bass')
```

Suggest clef based on minimal number of ledger lines.

Return clef mark. Changed in version 2.0: renamed `pitchtools.suggest_clef()` to `pitchtools.suggest_clef_for_named_chromatic_pitches()`.

`pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation`

`abjad.tools.pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation.symbolic_accidental_string_to_alphabetic_accidental_abbreviation`
 New in version 2.5. Change *symbolic_accidental_string* to alphabetic accidental abbreviation:

```
abjad> pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation('#+')
'tqs'
```

None when *symbolic_accidental_string* is not a valid symbolic accidental string.

Return string or none.

`pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment`

`abjad.tools.pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment`

New in version 2.0. Transpose chromatic *pitch* by melodic chromatic interval *segment*:

```
abjad> ncp = pitchtools.NumberedChromaticPitch(0)
abjad> mcis = pitchtools.MelodicChromaticIntervalSegment([0, -1, 2])
abjad> pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_segment(ncp, mcis)
[NumberedChromaticPitch(0), NumberedChromaticPitch(-1), NumberedChromaticPitch(1)]
```

Transpose by each interval in *segment* such that each transposition transposes the resulting pitch of the previous transposition.

Return list of numbered chromatic pitches.

`pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chromatic_pitch_number`

`abjad.tools.pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chromatic_pitch_number`

New in version 1.1. Transpose *chromatic_pitch_class_number* by octaves to nearest neighbor of *chromatic_pitch_number*:


```
abjad> pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chrom
16
```

Resulting chromatic pitch number must be within one tritone of *pitch_number*.

Return integer or float. Changed in version 2.0: renamed `pitchtools.nearest_neighbor()` to `pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_ch`.

`pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping`

```
abjad.tools.pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping.tra
```

New in version 1.1. Transpose *chromatic_pitch_number* by the some number of octaves up or down. Derive correct number of octaves from *mapping* where *mapping* is a list of (*range_spec*, *octave*) pairs and *range_spec* is, in turn, a (*start*, *stop*) pair suitable to pass to the built-in Python `range()` function:

```
abjad> mapping = [((-39, -13), 0), ((-12, 23), 12), ((24, 48), 24)]
```

The mapping given here comprises three (*range_spec*, *octave*) pairs. The first such pair is `((-39, -13), 0)` and can be read as follows: “any pitches between -39 and -13 should be transposed into the octave rooted at pitch 0.” The octave rooted at pitch 0 equals the twelve pitches `range(0, 0 + 12)` or `[0, 1, ..., 10, 11]`.

The second (*range_spec*, *octave*) pair is `((-12, 23), 12)` and can be read as “any pitches between -12 and 23 should be transposed into the octave rooted at pitch 12,” with the octave rooted at pitch 12 equal to the twelve pitches `range(12, 12 + 12)` or `[12, 13, ..., 22, 23]`.

The third and last (*range_spec*, *octave*) pair is `((24, 48), 24)` and can be read as “any pitches between 24 and 48 should be transposed to the octave rooted at 24,” with the octave rooted at 24 equal to the twelve pitches `range(24, 24, + 12)` or `[24, 25, ..., 34, 35]`.

The mapping given here divides the compass of the piano, from -39 to 48, into three disjunct subranges and then explains how to transpose pitches found in any of those three disjunct subranges. This means that, for example, all the f-sharps within the range of the piano now undergo a known transposition under *mapping* as defined here:

```
abjad> pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(-30, mapping)
6
```

We verify that pitch -30 should map to pitch 6 by noticing that pitch -30 falls in the first of the three subranges defined by *mapping* from -39 to -13 and then noting that *mapping* sends pitches with that subrange to the octave rooted at pitch 0. The octave transposition of -30 that falls within the octave rooted at 0 is 6:

```
abjad> pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(-18, mapping)
6
```

Likewise, *mapping* sends pitch -18 to pitch 6 because pitch -18 falls in the same subrange from -39 to -13 as did pitch -39 and so undergoes the same transposition to the octave rooted at 0.

In this way we can map all f-sharps from -39 to 48 according to *mapping*:

```
abjad> pitch_numbers = [-30, -18, -6, 6, 18, 30, 42]
abjad> for n in pitch_numbers:
...     n, pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping(n, mapping)
(-30, 6)
(-18, 6)
(-6, 18)
(6, 18)
```

```
(18, 18)
(30, 30)
(42, 30)
```

And so on.

Return chromatic pitch number. Changed in version 2.0: renamed `pitchtools.send_pitch_number_to_octave()` to `pitchtools.transpose_chromatic_pitch_number_`

`pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell`

`abjad.tools.pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_re`

New in version 1.1. Transpose named chromatic pitch by *melodic_chromatic_interval* and respell *staff_spaces* above or below:

```
abjad> pitch = pitchtools.NamedChromaticPitch(0)
abjad> pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_and_respell(pitch, 1)
NamedChromaticPitch("dtqf' ")
```

Return new named chromatic pitch. Changed in version 2.0: renamed `pitchtools.staff_space_transpose()` to `pitchtools.transpose_named_chromatic_pitch_by_melo`

`pitchtools.transpose_pitch_carrier_by_melodic_interval`

`abjad.tools.pitchtools.transpose_pitch_carrier_by_melodic_interval.transpose_pitch_carrier`

New in version 2.0. Transpose *pitch_carrier* by diatonic *melodic_interval*:

```
abjad> chord = Chord("<c' e' g'>4")

abjad> pitchtools.transpose_pitch_carrier_by_melodic_interval(chord, '+m2')
Chord("<df' f' af'>4")
```

Transpose *pitch_carrier* by chromatic *melodic_interval*:

```
abjad> chord = Chord("<c' e' g'>4")

abjad> pitchtools.transpose_pitch_carrier_by_melodic_interval(chord, 1)
Chord("<cs' f' af'>4")
```

Return non-pitch-carrying input unchanged:

```
abjad> rest = Rest('r4')

abjad> pitchtools.transpose_pitch_carrier_by_melodic_interval(rest, 1)
Rest('r4')
```

Return *pitch_carrier*.

`pitchtools.transpose_pitch_expr_into_pitch_range`

`abjad.tools.pitchtools.transpose_pitch_expr_into_pitch_range.transpose_pitch_expr_into_pit`

New in version 2.0. Transpose *pitch_expr* into *pitch_range*:

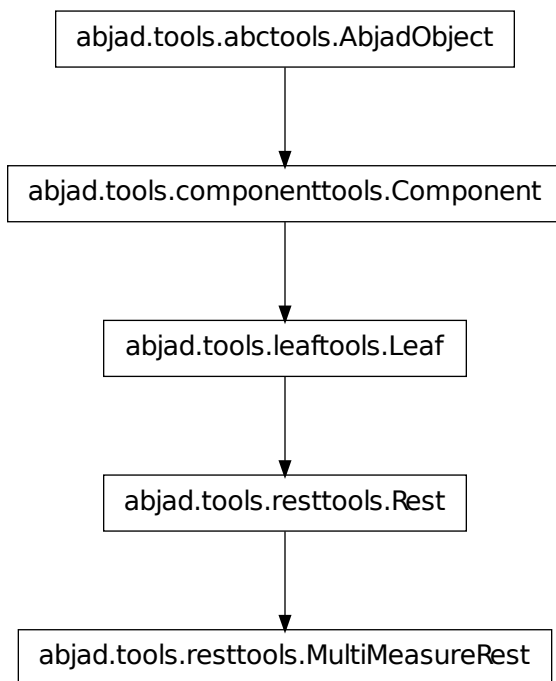
```
abjad> pitchtools.transpose_pitch_expr_into_pitch_range([-2, -1, 13, 14], pitchtools.PitchRange(
[10, 11, 1, 2])
```

Return new *pitch_expr* object.

resttools

resttools concrete classes

resttools.MultiMeasureRest



```
class abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest(*args,
                                                                **kwargs)
```

New in version 2.0. Abjad model of a multi-measure rest:

```
abjad> resttools.MultiMeasureRest((1, 4))
MultiMeasureRest('R4')
```

Multi-measure rests are immutable.

Read-only Properties

`MultiMeasureRest.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.format`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.leaf_index`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.multiplied_duration`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.parent`

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.preprolated_duration`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.prolated_duration`

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.prolation`

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`MultiMeasureRest.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_duration`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

`MultiMeasureRest.__and__(arg)`

Note: Inherited from `leaftools.Leaf`

`MultiMeasureRest.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MultiMeasureRest.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`MultiMeasureRest.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__mul__(n)`

Note: Inherited from `componenttools.Component`

`MultiMeasureRest.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MultiMeasureRest.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

MultiMeasureRest.__repr__()

Note: Inherited from `leaftools.Leaf`

MultiMeasureRest.__rmul__(*n*)

Note: Inherited from `componenttools.Component`

MultiMeasureRest.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

MultiMeasureRest.__str__()

Note: Inherited from `leaftools.Leaf`

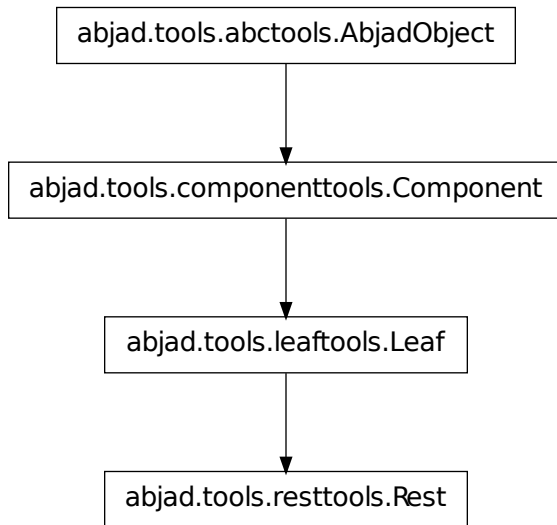
MultiMeasureRest.__sub__(*arg*)

Note: Inherited from `leaftools.Leaf`

MultiMeasureRest.__xor__(*arg*)

Note: Inherited from `leaftools.Leaf`

resttools.Rest



```

class abjad.tools.resttools.Rest.Rest.Rest (*args, **kwargs)
    Abjad model of a rest:

    abjad> Rest((3, 16))
    Rest('r8.')
  
```

Read-only Properties

Rest.**duration_in_seconds**

Note: Inherited from `leaftools.Leaf`

Rest.**format**

Note: Inherited from `leaftools.Leaf`

Rest.**leaf_index**

Note: Inherited from `leaftools.Leaf`

Rest.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Rest.**multiplied_duration**

Note: Inherited from `leaftools.Leaf`

Rest.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Rest.**parent**

Note: Inherited from `componenttools.Component`

Rest.**preprolated_duration**

Note: Inherited from `leaftools.Leaf`

Rest.**prolated_duration**

Note: Inherited from `componenttools.Component`

Rest.**prolation**

Note: Inherited from `componenttools.Component`

Rest.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Rest.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Rest.**duration_multiplier**

Note: Inherited from `leaftools.Leaf`

Rest.**written_duration**

Note: Inherited from `leaftools.Leaf`

Rest.**written_pitch_indication_is_at_sounding_pitch**

Note: Inherited from `leaftools.Leaf`

Rest.**written_pitch_indication_is_nonsemantic**

Note: Inherited from `leaftools.Leaf`

Special Methods

Rest.**__and__**(*arg*)

Note: Inherited from `leaftools.Leaf`

Rest.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Rest.**__eq__**(*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Rest.**__ge__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Rest.**__gt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

Rest.**__hash__**() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

Rest.`__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Rest.`__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Rest.`__mul__(n)`

Note: Inherited from `componenttools.Component`

Rest.`__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Rest.`__or__(arg)`

Note: Inherited from `leaftools.Leaf`

Rest.`__repr__()`

Note: Inherited from `leaftools.Leaf`

Rest.`__rmul__(n)`

Note: Inherited from `componenttools.Component`

Rest.`__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Rest.`__str__()`

Note: Inherited from `leaftools.Leaf`

`Rest.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`Rest.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

resttools functions

resttools.all_are_rests

`abjad.tools.resttools.all_are_rests.all_are_rests(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad rests:

```
abjad> rests = [Rest('r4'), Rest('r4'), Rest('r4')]
```

```
abjad> resttools.all_are_rests(rests)
```

```
True
```

True when *expr* is an empty sequence:

```
abjad> resttools.all_are_rests([])
```

```
True
```

Otherwise false:

```
abjad> resttools.all_are_rests('foo')
```

```
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

resttools.is_lilypond_rest_string

`abjad.tools.resttools.is_lilypond_rest_string.is_lilypond_rest_string(expr)`

New in version 2.0. True when *expr* is a LilyPond rest string:

```
abjad> resttools.is_lilypond_rest_string('r4.. * 1/2')
```

```
True
```

Otherwise false:

```
abjad> resttools.is_lilypond_rest_string('text')
```

```
False
```

The regex `^(r|R)\s*(1|2|4|8|16|32|64|128|\breve|\longa|\maxima)\s*(\.*)\s*(*\s*(\d+(\d+|/\d+|` underlies this predicate.

Return boolean.

`resttools.iterate_rests_backward_in_expr`

`abjad.tools.resttools.iterate_rests_backward_in_expr.iterate_rests_backward_in_expr(expr, start=0, stop=None)`

New in version 2.0. Iterate rests backward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

abjad> for rest in resttools.iterate_rests_backward_in_expr(staff):
...     rest
Rest('r2')
Rest('r8')
```

Ignore threads.

Return generator.

`resttools.iterate_rests_forward_in_expr`

`abjad.tools.resttools.iterate_rests_forward_in_expr.iterate_rests_forward_in_expr(expr, start=0, stop=None)`

New in version 2.0. Iterate rests forward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 r8 <d' f' b'>8 r2")

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    r8
    <d' f' b'>8
    r2
}

abjad> for rest in resttools.iterate_rests_forward_in_expr(staff):
...     rest
Rest('r8')
Rest('r2')
```

Ignore threads.

Return generator.

resttools.make_multi_measure_rests

abjad.tools.resttools.make_multi_measure_rests.**make_multi_measure_rests**(*duration_tokens*)

New in version 2.0. Make multi-measure rests from *duration_tokens*:

```
abjad> resttools.make_multi_measure_rests([(4, 4), (7, 4)])
[MultiMeasureRest('R1'), MultiMeasureRest('R1..')]
```

Return list.

resttools.make_repeated_rests_from_time_signature

abjad.tools.resttools.make_repeated_rests_from_time_signature.**make_repeated_rests_from_time_signature**(*time_signature*)

New in version 2.0. Make repeated rests from *time_signature*:

```
abjad> resttools.make_repeated_rests_from_time_signature((5, 32))
[Rest('r32'), Rest('r32'), Rest('r32'), Rest('r32'), Rest('r32')]
```

Return list of newly constructed rests.

resttools.make_repeated_rests_from_time_signatures

abjad.tools.resttools.make_repeated_rests_from_time_signatures.**make_repeated_rests_from_time_signatures**(*time_signatures*)

Make repeated rests from *time_signatures*:

```
resttools.make_repeated_rests_from_time_signatures([(2, 8), (3, 32)])
[[Rest('r8'), Rest('r8')], [Rest('r32'), Rest('r32'), Rest('r32')]]
```

Return two-dimensional list of newly constructed rest lists.

Use `sequencetools.flatten_sequence()` to flatten output if required.

resttools.make_rests

abjad.tools.resttools.make_rests.**make_rests**(*duration_tokens*, *direction*='big-endian', *tied*=False)

New in version 1.1. Make rests.

Make big-endian rests:

```
abjad> resttools.make_rests([(5, 16), (9, 16)], direction = 'big-endian')
[Rest('r4'), Rest('r16'), Rest('r2'), Rest('r16')]
```

Make little-endian rests:

```
abjad> resttools.make_rests([(5, 16), (9, 16)], direction = 'little-endian')
[Rest('r16'), Rest('r4'), Rest('r16'), Rest('r2')]
```

Make tied rests:

```
abjad> voice = Voice(resttools.make_rests([(5, 16), (9, 16)], tied = True))

abjad> f(voice)
\new Voice {
    r4 ~
    r16
}
```

```

    r2 ~
    r16
}

```

Return list of rests. Changed in version 2.0: renamed `construct.rests()` to `resttools.make_rests()`.

`resttools.replace_leaves_in_expr_with_rests`

`abjad.tools.resttools.replace_leaves_in_expr_with_rests.replace_leaves_in_expr_with_rests()`

New in version 1.1. Replace leaves in *expr* with rests:

```

abjad> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
abjad> resttools.replace_leaves_in_expr_with_rests(staff[0])
abjad> print staff.format
\new Staff {
  {
    \time 2/8
    r8
    r8
  }
  {
    \time 2/8
    c'8
    d'8
  }
}

```

Return none.

`resttools.set_vertical_positioning_pitch_on_rest`

`abjad.tools.resttools.set_vertical_positioning_pitch_on_rest.set_vertical_positioning_pitch_on_rest()`

New in version 2.0. Set vertical positioning *pitch* on *rest*:

```

abjad> rest = Rest((1, 4))

abjad> resttools.set_vertical_positioning_pitch_on_rest(rest, "d'")
Rest('r4')

abjad> f(rest)
d''4 \rest

```

Raise type error when *rest* is not a rest.

Return *rest*.

`resttools.yield_groups_of_rests_in_sequence`

`abjad.tools.resttools.yield_groups_of_rests_in_sequence.yield_groups_of_rests_in_sequence()`

New in version 2.0. Yield groups of rests in *sequence*:

```

abjad> staff = Staff("c'8 d'8 r8 r8 <e' g'>8 <f' a'>8 g'8 a'8 r8 r8 <b' d''>8 <c'' e''>8")

```

```

abjad> f(staff)
\new Staff {
    c'8
    d'8
    r8
    r8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    r8
    r8
    <b' d''>8
    <c'' e''>8
}

abjad> for rest in resttools.yield_groups_of_rests_in_sequence(staff):
...     rest
...
(Rest('r8'), Rest('r8'))
(Rest('r8'), Rest('r8'))

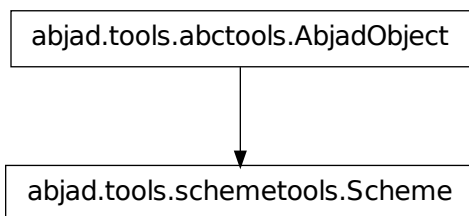
```

Return generator.

schemetools

schemetools concrete classes

schemetools.Scheme



```

class abjad.tools.schemetools.Scheme.Scheme.Scheme(*args,**kwargs)

```

Abjad model of Scheme code:

```

abjad> from abjad.tools.schemetools import Scheme
abjad> s = Scheme(True)
abjad> s.format
'##t'

```

schemetools.Scheme can represent nested structures


```
abjad> s = Scheme(('left', (1, 2, False)), ('right', (1, 2, 3.3)))
abjad> s.format
'#((left (1 2 #f)) (right (1 2 3.3)))'
```

schemetools.Scheme wraps variable-length arguments into a tuple

```
abjad> s = Scheme(1, 2, 3)
abjad> q = Scheme((1, 2, 3))
abjad> s.format == q.format
True
```

schemetools.Scheme also takes an optional *quoting* keyword, by which Scheme's various quote, unquote, unquote-splicing characters can be prepended to the formatted result

```
abjad> s = Scheme((1, 2, 3), quoting="'#")
abjad> s.format
"'#(1 2 3) "
```

Scheme is immutable.

Read-only Properties

Scheme.**format**

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> Scheme(True).format
'##t'
```

Returns string.

Special Methods

Scheme.**__delattr__**()

x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Scheme.**__eq__**(other)

Scheme.**__ge__**(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Scheme.**__gt__**(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Scheme.**__hash__**() <==> *hash*(x)

Note: Inherited from `__builtin__.object`

`Scheme.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Scheme.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Scheme.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Scheme.__repr__()`

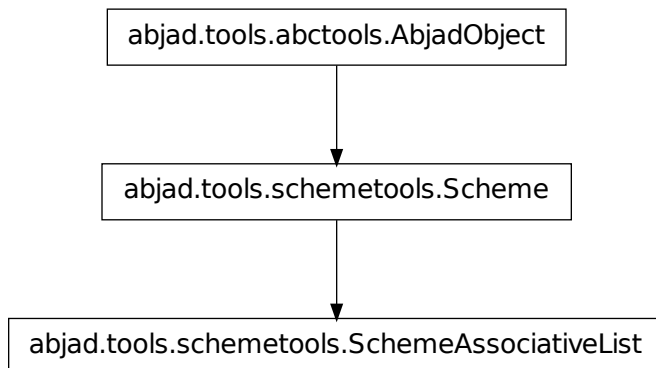
`Scheme.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Scheme.__str__()`

schemetools.SchemeAssociativeList



class abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.**SchemeAssociativeLi**

New in version 2.0. Abjad model of Scheme associative list:

```
abjad> from abjad.tools.schemetools import SchemeAssociativeList
abjad> SchemeAssociativeList(('space', 2), ('padding', 0.5))
SchemeAssociativeList((SchemePair(('space', 2)), SchemePair(('padding', 0.5))))
```

Scheme associative lists are immutable.

Read-only Properties

`SchemeAssociativeList.format`

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> Scheme(True).format
'##t'
```

Returns string.

Note: Inherited from `schemetools.Scheme`

Special Methods

`SchemeAssociativeList.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SchemeAssociativeList.__eq__(other)`

Note: Inherited from `schemetools.Scheme`

`SchemeAssociativeList.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SchemeAssociativeList.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SchemeAssociativeList.__repr__()`

Note: Inherited from `schemetools.Scheme`

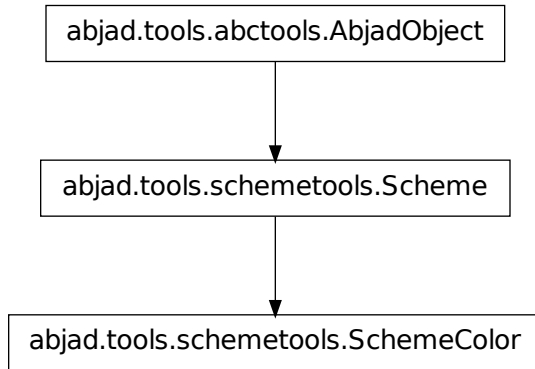
`SchemeAssociativeList.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SchemeAssociativeList.__str__()`

Note: Inherited from `schemetools.Scheme`

schemetools.SchemeColor

```
class abjad.tools.schemetools.SchemeColor.SchemeColor, SchemeColor(*args,  
                                                                **kwargs)
```

Abjad model of Scheme color:

```
abjad> schemetools.SchemeColor('ForestGreen')  
SchemeColor('ForestGreen')
```

Scheme colors are immutable.

Read-only Properties

`SchemeColor.format`

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme  
abjad> Scheme(True).format  
'##t'
```

Returns string.

Note: Inherited from `schemetools.Scheme`

Special Methods

```
SchemeColor.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

`SchemeColor.__eq__(other)`

Note: Inherited from `schemetools.Scheme`

`SchemeColor.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeColor.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SchemeColor.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SchemeColor.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeColor.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeColor.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SchemeColor.__repr__()`

Note: Inherited from `schemetools.Scheme`

`SchemeColor.__setattr__()`

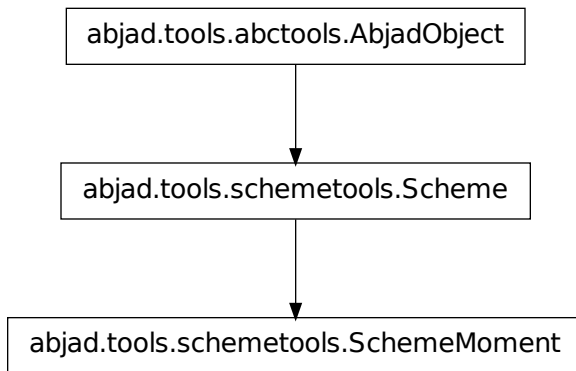
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SchemeColor.__str__()`

Note: Inherited from `schemetools.Scheme`

`schemetools.SchemeMoment`



```
class abjad.tools.schemetools.SchemeMoment.SchemeMoment(*args,
                                                         **kwargs)
```

Abjad model of LilyPond moment:

```
abjad> schemetools.SchemeMoment(1, 68)
SchemeMoment(1, 68)
```

Initialize scheme moments with a single fraction, two integers or another scheme moment.

Scheme moments are immutable.

Read-only Properties

`SchemeMoment.duration`

Duration of scheme moment:

```
abjad> scheme_moment = schemetools.SchemeMoment(1, 68)
abjad> scheme_moment.duration
Fraction(1, 68)
```

Return duration.

`SchemeMoment.format`

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> Scheme(True).format
'##t'
```

Returns string.

Note: Inherited from `schemetools.Scheme`

Special Methods

`SchemeMoment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SchemeMoment.__eq__(arg)`
`SchemeMoment.__ge__(arg)`
`SchemeMoment.__gt__(arg)`
`SchemeMoment.__hash__() <==> hash(x)`

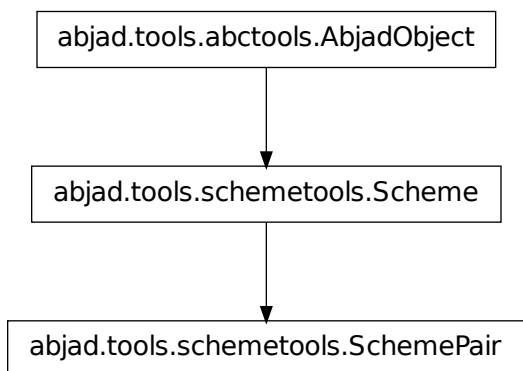
Note: Inherited from `__builtin__.object`

`SchemeMoment.__le__(arg)`
`SchemeMoment.__lt__(arg)`
`SchemeMoment.__ne__(arg)`
`SchemeMoment.__repr__()`
`SchemeMoment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SchemeMoment.__str__()`

Note: Inherited from `schemetools.Scheme`

schemetools.SchemePair

class `abjad.tools.schemetools.SchemePair.SchemePair` **SchemePair** (*args, **kwargs)
 Abjad model of Scheme pair:

```
abjad> schemetools.SchemePair('spacing', 4)
SchemePair(('spacing', 4))
```

Initialize Scheme pairs with a tuple, two separate values or another Scheme pair.

Scheme pairs are immutable.

Read-only Properties

`SchemePair.format`

Special Methods

`SchemePair.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SchemePair.__eq__(other)`

Note: Inherited from `schemetools.Scheme`

`SchemePair.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemePair.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SchemePair.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SchemePair.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemePair.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemePair.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SchemePair.__repr__()`

Note: Inherited from `schemetools.Scheme`

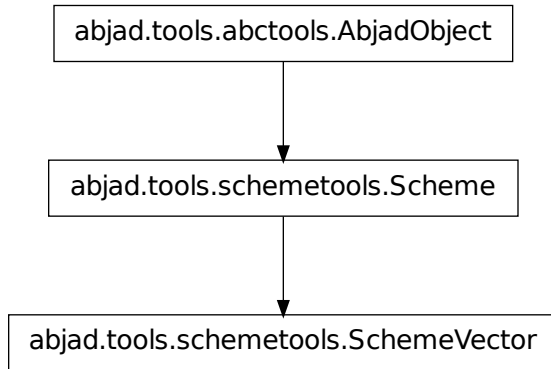
`SchemePair.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SchemePair.__str__()`

Note: Inherited from `schemetools.Scheme`

schemetools.SchemeVector

class `abjad.tools.schemetools.SchemeVector.SchemeVector`. **SchemeVector** (*args)
New in version 2.0. Abjad model of Scheme vector:

```
abjad> schemetools.SchemeVector(True, True, False)
SchemeVector((True, True, False))
```

Scheme vectors and Scheme vector constants differ in only their LilyPond input format.

Scheme vectors are immutable.

Read-only Properties

`SchemeVector.format`

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> Scheme(True).format
'##t'
```

Returns string.

Note: Inherited from `schemetools.Scheme`

Special Methods

`SchemeVector.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SchemeVector.__eq__(other)`

Note: Inherited from `schemetools.Scheme`

`SchemeVector.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVector.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SchemeVector.__hash__()` $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

`SchemeVector.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVector.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVector.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SchemeVector.__repr__()`

Note: Inherited from `schemetools.Scheme`

`SchemeVector.__setattr__()`

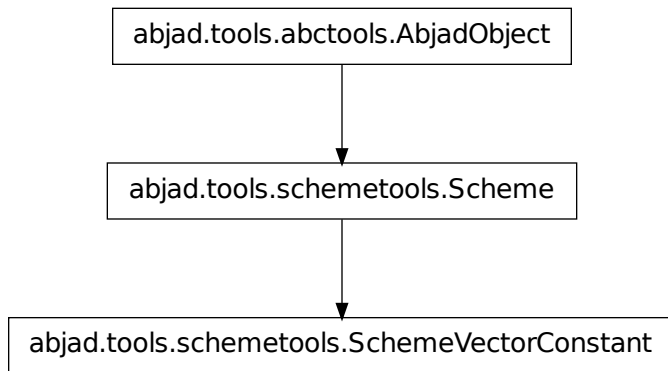
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`SchemeVector.__str__()`

Note: Inherited from `schemetools.Scheme`

`schemetools.SchemeVectorConstant`



class `abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant` **`.SchemeVectorConstant`**
 New in version 2.0. Abjad model of Scheme vector constant:

```
abjad> schemetools.SchemeVectorConstant(True, True, False)
SchemeVectorConstant((True, True, False))
```

Scheme vectors and Scheme vector constants differ in only their LilyPond input format.

Scheme vector constants are immutable.

Read-only Properties

`SchemeVectorConstant.format`

Hash-mark-prepended format of Scheme:

```
abjad> from abjad.tools.schemetools import Scheme
abjad> Scheme(True).format
'##t'
```

Returns string.

Note: Inherited from `schemetools.Scheme`

Special Methods

`SchemeVectorConstant.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SchemeVectorConstant.__eq__(other)`

Note: Inherited from `schemetools.Scheme`

`SchemeVectorConstant.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SchemeVectorConstant.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SchemeVectorConstant.__repr__()`

Note: Inherited from `schemetools.Scheme`

```
SchemeVectorConstant.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

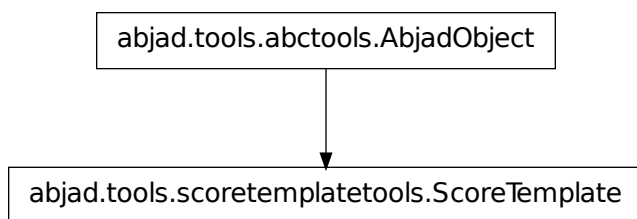
```
SchemeVectorConstant.__str__()
```

Note: Inherited from `schemetools.Scheme`

scoretemplatetools

scoretemplatetools abstract classes

scoretemplatetools.ScoreTemplate



class `abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate`
New in version 2.8. Abstract score template.

Special Methods

```
ScoreTemplate.__call__()  
ScoreTemplate.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
ScoreTemplate.__eq__(arg)  
True when id(self) equals id(arg).  
Return boolean.
```

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`ScoreTemplate.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`ScoreTemplate.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

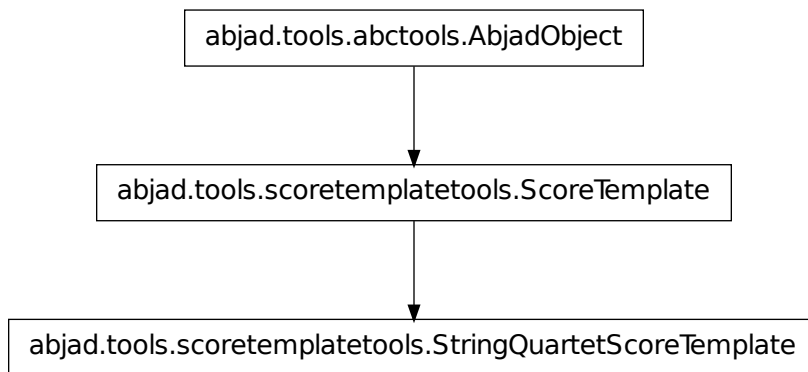
Note: Inherited from `__builtin__.object`

ScoreTemplate.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

scoretemplatetools concrete classes

scoretemplatetools.StringQuartetScoreTemplate



class `abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate.St`

New in version 2.8. String quartet score template:

```

abjad> from abjad.tools import scoretemplatetools

abjad> template = scoretemplatetools.StringQuartetScoreTemplate()
abjad> score = template()

abjad> score
Score-"String Quartet Score"<<1>>

abjad> f(score)
\context Score = "String Quartet Score" <<
  \context StaffGroup = "String Quartet Staff Group" <<
    \context Staff = "First Violin Staff" {
      \clef "treble"
      \context Voice = "First Violin Voice" {
      }
    }
    \context Staff = "Second Violin Voice" {
      \clef "treble"
    }
    \context Staff = "Viola Staff" {
      \clef "alto"
    }
  
```

```

        \context Staff = "Cello Staff" {
            \clef "bass"
        }
    >>
>>

```

Return score template.

Special Methods

StringQuartetScoreTemplate.__call__()
 StringQuartetScoreTemplate.__delattr__()
 x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

StringQuartetScoreTemplate.__eq__(arg)
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

StringQuartetScoreTemplate.__ge__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

StringQuartetScoreTemplate.__gt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

StringQuartetScoreTemplate.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

StringQuartetScoreTemplate.__le__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

StringQuartetScoreTemplate.__lt__(arg)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`StringQuartetScoreTemplate.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
 Return string.

Note: Inherited from `abctools.AbjadObject`

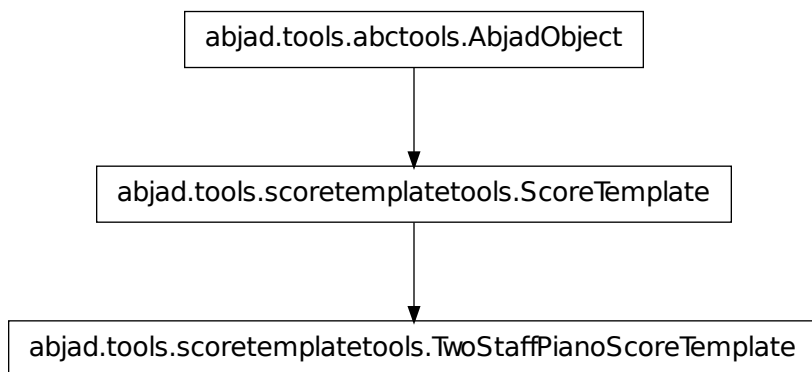
`StringQuartetScoreTemplate.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`StringQuartetScoreTemplate.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

scoretemplatetools.TwoStaffPianoScoreTemplate



class `abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate`. **TwoStaffPianoScoreTemplate**
 New in version 2.8. Two-staff piano score template:

```
abjad> from abjad.tools import scoretemplatetools

abjad> template = scoretemplatetools.TwoStaffPianoScoreTemplate()
abjad> score = template()

abjad> score
Score-"Two-Staff Piano Score"<<1>>

abjad> f(score)
\context Score = "Two-Staff Piano Score" <<
  \context PianoStaff = "Piano Staff" <<
    \context Staff = "RH Staff" {
      \clef "treble"
      \context Voice = "RH Voice" {
      }
    }
    \context Staff = "LH Staff" {
      \clef "bass"
      \context Voice = "LH Voice" {
      }
    }
  >>
>>
```

Return score template.

Special Methods

`TwoStaffPianoScoreTemplate.__call__()`
`TwoStaffPianoScoreTemplate.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TwoStaffPianoScoreTemplate.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`TwoStaffPianoScoreTemplate.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

`TwoStaffPianoScoreTemplate.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

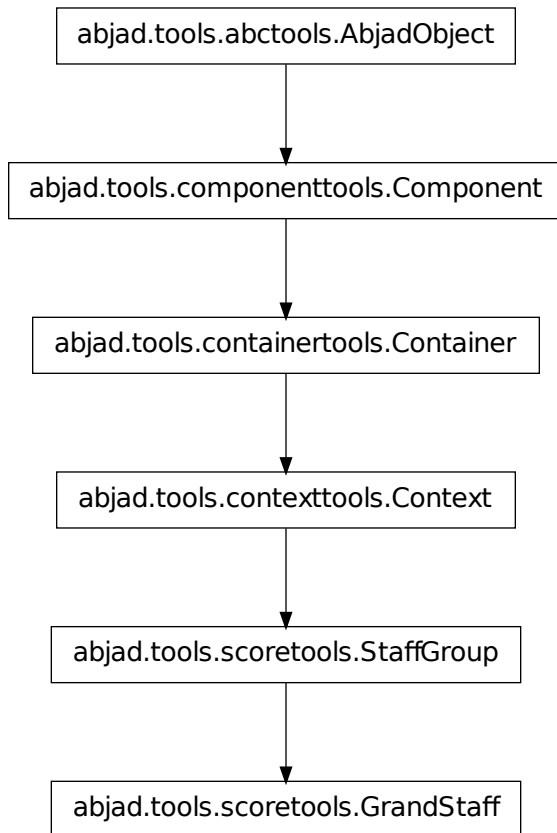
`TwoStaffPianoScoreTemplate.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

scoretools

scoretools concrete classes

scoretools.GrandStaff



class `abjad.tools.scoretools.GrandStaff.GrandStaff` (*music*)

Abjad model of grand staff:

```
abjad> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
```

```
abjad> staff_2 = Staff("g2 f2 e1")
```

```
abjad> grand_staff = scoretools.GrandStaff([staff_1, staff_2])
```

```
abjad> f(grand_staff)
```

```
\new GrandStaff <<
```

```
  \new Staff {
```

```
    c'4
```

```
    d'4
```

```
    e'4
```

```
    f'4
```

```

        g'1
    }
    \new Staff {
        g2
        f2
        e1
    }
>>

```

Return grand staff.

Read-only Properties

`GrandStaff.contents_duration`

Note: Inherited from `containertools.Container`

`GrandStaff.duration_in_seconds`

Note: Inherited from `containertools.Container`

`GrandStaff.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```

abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}

```

Note: Inherited from `contextttools.Context`

`GrandStaff.engraver_removals`

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```

abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}

```

Note: Inherited from `contextttools.Context`

`GrandStaff.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`GrandStaff.is_semantic`

Note: Inherited from `contexttools.Context`

`GrandStaff.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

`GrandStaff.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`GrandStaff.music`

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

`GrandStaff.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`GrandStaff.parent`

Note: Inherited from `componenttools.Component`

`GrandStaff.preprolated_duration`

Note: Inherited from `containertools.Container`

`GrandStaff.prolated_duration`

Note: Inherited from `componenttools.Component`

`GrandStaff.prolation`

Note: Inherited from `componenttools.Component`

`GrandStaff.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`GrandStaff.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`GrandStaff.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`GrandStaff.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'
```

```
abjad> voice.is_nonsemantic = True
```

```
abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

GrandStaff.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

`GrandStaff.name`

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

`GrandStaff.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`GrandStaff.extend(expr)`

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
}
```

```

        ds'8
        es'8
    }

```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`GrandStaff.index(component)`

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2

```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`GrandStaff.insert(i, component)`

Insert *component* in container at index *i*:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}

```

Return none.

Note: Inherited from `containertools.Container`

`GrandStaff.pop(i=-1)`

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`GrandStaff.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`GrandStaff.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new Container `c` with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`GrandStaff.__contains__(expr)`
True if `expr` is in container, otherwise False.

Note: Inherited from `containertools.Container`

`GrandStaff.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`GrandStaff.__delitem__(i)`
Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`GrandStaff.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__getitem__(i)`
Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`GrandStaff.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`GrandStaff.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`GrandStaff.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`GrandStaff.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__len__()`
Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`GrandStaff.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__mul__(n)`

Note: Inherited from `componenttools.Component`

`GrandStaff.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GrandStaff.__radd__(expr)`
Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`GrandStaff.__repr__()`
Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`GrandStaff.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`GrandStaff.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

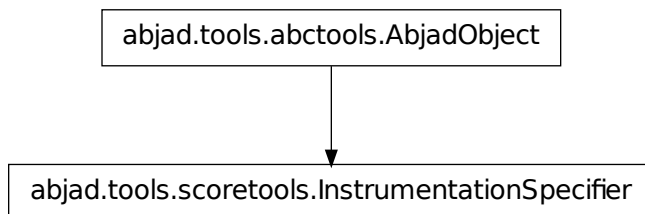
`GrandStaff.__setitem__(i, expr)`
Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`GrandStaff.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

scoretools.InstrumentationSpecifier



class `abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier`. **InstrumentationSpecifier**

New in version 2.5. Abjad model of score instrumentation:

```
abjad> flute = scoretools.Performer('Flute')
abjad> flute.instruments.append(instrumenttools.Flute())
abjad> flute.instruments.append(instrumenttools.AltoFlute())

abjad> guitar = scoretools.Performer('Guitar')
abjad> guitar.instruments.append(instrumenttools.Guitar())
```



```
abjad> instrumentation_specifier = scoretools.InstrumentationSpecifier([flute, guitar])
```

```
abjad> instrumentation_specifier
InstrumentationSpecifier(performers=PerformerInventory([Performer(name='Flute', instruments=Inst
```

Return instrumentation specifier.

Read-only Properties

`InstrumentationSpecifier.instrument_count`

Read-only number of instruments in score:

```
abjad> instrumentation_specifier.instrument_count
3
```

Return nonnegative integer.

`InstrumentationSpecifier.instruments`

Read-only list of instruments derived from performers:

```
abjad> instrumentation_specifier.instruments
[Flute(), AltoFlute(), Guitar()]
```

Return list.

`InstrumentationSpecifier.performer_count`

Read-only number of performers in score:

```
abjad> instrumentation_specifier.performer_count
2
```

Return nonnegative integer.

`InstrumentationSpecifier.performer_name_string`

Read-only string of performer names:

```
abjad> instrumentation_specifier.performer_name_string
'Flute, Guitar'
```

Return string.

Read/write Properties

`InstrumentationSpecifier.performers`

Read / write list of performers in score:

```
abjad> instrumentation_specifier.performers
PerformerInventory([Performer(name='Flute', instruments=InstrumentInventory([Flute(), AltoFlute()
```

Return list.

Special Methods

`InstrumentationSpecifier.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InstrumentationSpecifier.__eq__ (other)`

`InstrumentationSpecifier.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`InstrumentationSpecifier.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__ne__(other)`

`InstrumentationSpecifier.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

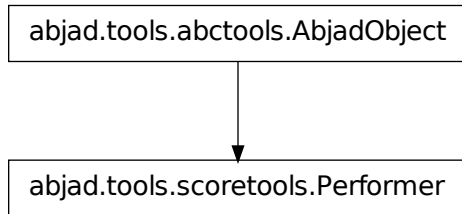
Note: Inherited from `abctools.AbjadObject`

`InstrumentationSpecifier.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`InstrumentationSpecifier.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

scoretools.Performer

class `abjad.tools.scoretools.Performer.Performer` (*name=None, instruments=None*)

New in version 2.5. Abjad model of performer:

```
abjad> scoretools.Performer(name='flutist')
Performer(name='flutist', instruments=InstrumentInventory([]))
```

The purpose of the class is to model things like flute I doubling piccolo and alto flute.

At present the class is a list of instruments.

Read-only Properties

`Performer.instrument_count`

Read-only number of instruments to be played by performer:

```
abjad> performer = scoretools.Performer('flutist')

abjad> performer.instruments.append(instrumenttools.Flute())
abjad> performer.instruments.append(instrumenttools.Piccolo())

abjad> performer.instrument_count
2
```

Return nonnegative integer

`Performer.is_doubling`

Is performer to play more than one instrument?

```
abjad> performer = scoretools.Performer('flutist')

abjad> performer.instruments.append(instrumenttools.Flute())
abjad> performer.instruments.append(instrumenttools.Piccolo())

abjad> performer.is_doubling
True
```

Return boolean.

`Performer.likely_instruments_based_on_performer_name`

New in version 2.5. Likely instruments based on performer name:

```
abjad> flutist = scoretools.Performer(name='flutist')
abjad> for likely_instrument in flutist.likely_instruments_based_on_performer_name:
...     likely_instrument
...
<class 'abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute'>
<class 'abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute'>
<class 'abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute'>
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
<class 'abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo'>
```

Return list.

Performer.most_likely_instrument_based_on_performer_name

New in version 2.5. Most likely instrument based on performer name:

```
abjad> flutist = scoretools.Performer(name='flutist')
abjad> flutist.most_likely_instrument_based_on_performer_name
<class 'abjad.tools.instrumenttools.Flute.Flute.Flute'>
```

Return instrument class.

Read/write Properties

Performer.instruments

List of instruments to be played by performer:

```
abjad> performer = scoretools.Performer('flutist')

abjad> performer.instruments.append(instrumenttools.Flute())
abjad> performer.instruments.append(instrumenttools.Piccolo())

abjad> performer.instruments
InstrumentInventory([Flute(), Piccolo()])
```

Return list.

Performer.name

Score name of performer:

```
abjad> performer = scoretools.Performer('flutist')

abjad> performer.name
'flutist'
```

Return string.

Methods

Performer.make_performer_name_instrument_dictionary (*locale=None*)

New in version 2.5. Make performer name / instrument dictionary.

Return dictionary.

Special Methods

Performer.__delattr__ ()

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Performer.__eq__(*other*)

Performer.__ge__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Performer.__gt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Performer.__hash__()

Performer.__le__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Performer.__lt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Performer.__ne__(*other*)

Performer.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

Performer.__setattr__()

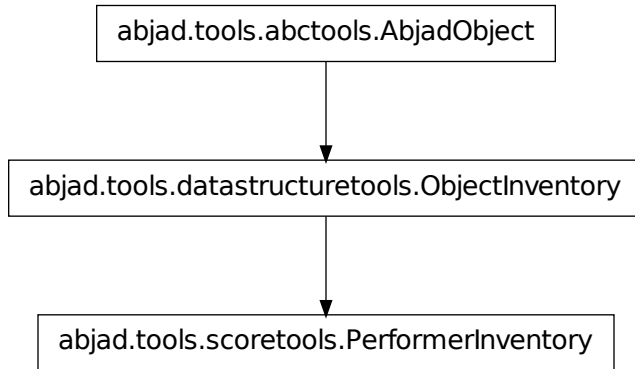
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Performer.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

scoretools.PerformerInventory



class `abjad.tools.scoretools.PerformerInventory.PerformerInventory` (*tokens=None, inventory_name=None*)

New in version 2.8. Abjad model of an ordered list of performers.

Performer inventories implement the list interface and are mutable.

Read/write Properties

`PerformerInventory.inventory_name`

Read / write name of inventory.

Note: Inherited from `datastructuretools.ObjectInventory`

Methods

`PerformerInventory.append(token)`

Change *token* to item and append.

Note: Inherited from `datastructuretools.ObjectInventory`

`PerformerInventory.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`PerformerInventory.extend(tokens)`

Change *tokens* to items and extend.

Note: Inherited from `datastructuretools.ObjectInventory`

`PerformerInventory.index(value[, start[, stop]])` → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`PerformerInventory.insert()`
`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`PerformerInventory.pop([index])` → item – remove and return item at index (default last).
Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`PerformerInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`PerformerInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`PerformerInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`PerformerInventory.__add__()`
`x.__add__(y)` <==> `x+y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__contains__(token)`

Note: Inherited from `datastructuretools.ObjectInventory`

`PerformerInventory.__delattr__()`
`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`PerformerInventory.__delitem__()`
`x.__delitem__(y)` <==> `del x[y]`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`PerformerInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`PerformerInventory.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__mul__()`
`x.__mul__(n) <==> x*n`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__repr__()`

Note: Inherited from `datastructuretools.ObjectInventory`

`PerformerInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`PerformerInventory.__rmul__()`
`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`PerformerInventory.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

PerformerInventory.__setitem__()
x.__setitem__(i, y) <==> x[i]=y

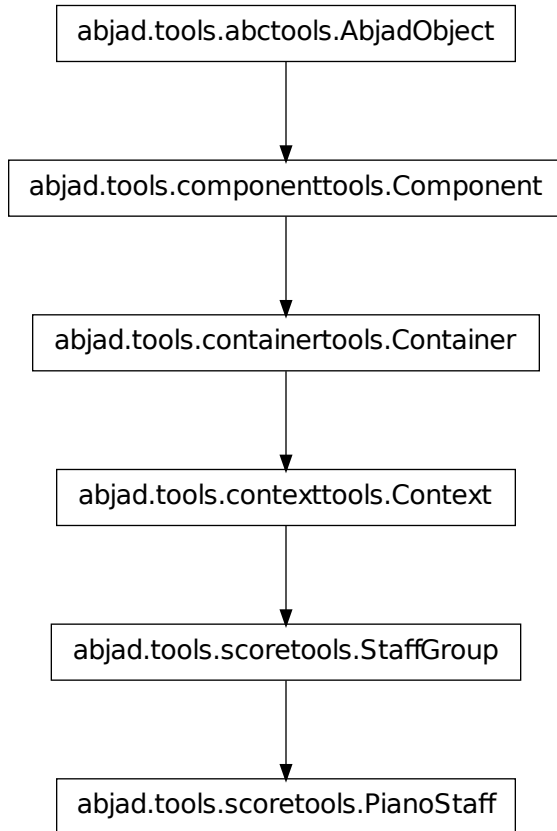
Note: Inherited from `__builtin__.list`

PerformerInventory.__setslice__()
x.__setslice__(i, j, y) <==> x[i:j]=y
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

PerformerInventory.__str__() <==> *str*(x)

Note: Inherited from `__builtin__.object`

scoretools.PianoStaff

class `abjad.tools.scoretools.PianoStaff.PianoStaff`(*music=None*,
***kwargs*)

Abjad model of piano staff:

```

abjad> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
abjad> staff_2 = Staff("g2 f2 e1")

abjad> piano_staff = scoretools.PianoStaff([staff_1, staff_2])

abjad> f(piano_staff)
\new PianoStaff <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
    g'1
  }
  \new Staff {
    g2
  }

```

```
        f2
        e1
    }
>>
```

Return piano staff.

Read-only Properties

PianoStaff.contents_duration

Note: Inherited from `containertools.Container`

PianoStaff.duration_in_seconds

Note: Inherited from `containertools.Container`

PianoStaff.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

PianoStaff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

PianoStaff.format

Read-only LilyPond input format of component.

Note: Inherited from `componentttools.Component`

PianoStaff.**is_semantic**

Note: Inherited from `contexttools.Context`

PianoStaff.**leaves**

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

PianoStaff.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

PianoStaff.**music**

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

PianoStaff.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

PianoStaff.**parent**

Note: Inherited from `componenttools.Component`

PianoStaff.**preprolated_duration**

Note: Inherited from `containertools.Container`

PianoStaff.**prolated_duration**

Note: Inherited from `componenttools.Component`

`PianoStaff.prolation`

Note: Inherited from `componenttools.Component`

`PianoStaff.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`PianoStaff.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`PianoStaff.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`PianoStaff.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'

abjad> voice.is_nonsemantic = True

abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}

abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

PianoStaff.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

PianoStaff.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

PianoStaff.**append**(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.append(Note("f'8"))
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

PianoStaff.**extend**(*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```


Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`PianoStaff.index(component)`

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> note = container[-1]
```

```
abjad> note
```

```
Note("e'8")
```

```
abjad> container.index(note)
```

```
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`PianoStaff.insert(i, component)`

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
```

```
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.insert(1, Note("cs'8"))
```

```
abjad> f(container)
```

```
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`PianoStaff.pop(i=-1)`

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
```

```
{
    c'8 [
```

```
        d'8
        e'8 ]
    }

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

PianoStaff.**remove**(*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

PianoStaff.**__add__**(*expr*)

Concatenate containers self and *expr*. The operation $c = a + b$ returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`PianoStaff.__contains__(expr)`
 True if `expr` is in container, otherwise False.

Note: Inherited from `containertools.Container`

`PianoStaff.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`PianoStaff.__delitem__(i)`
 Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`PianoStaff.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__getitem__(i)`
 Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`PianoStaff.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`PianoStaff.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`PianoStaff.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`PianoStaff.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`PianoStaff.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__mul__(n)`

Note: Inherited from `componenttools.Component`

`PianoStaff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`PianoStaff.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`PianoStaff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`PianoStaff.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`PianoStaff.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

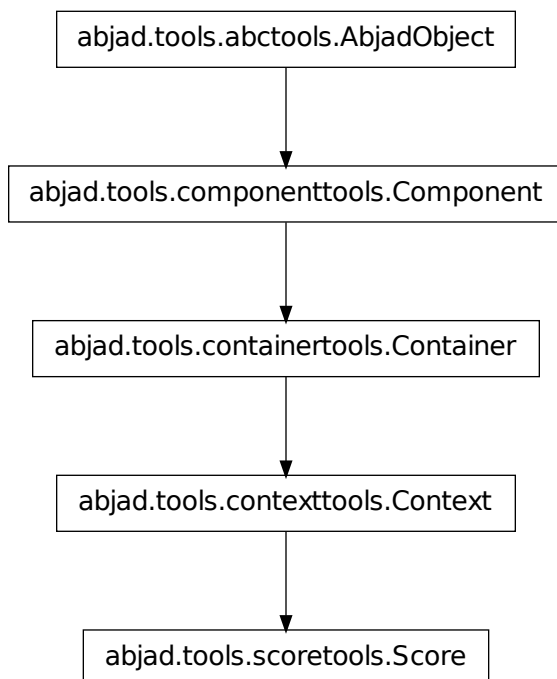
`PianoStaff.__setitem__(i, expr)`
 Set 'expr' in self at nonnegative integer index i. Or, set 'expr' in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`PianoStaff.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

scoretools.Score



class `abjad.tools.scoretools.Score.Score.Score` (*music=None, **kwargs*)

Abjad model of a score:

```
abjad> staff_1 = Staff("c'8 d'8 e'8 f'8")
abjad> staff_2 = Staff("c'8 d'8 e'8 f'8")
abjad> score = Score([staff_1, staff_2])
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>
```

Return score object.

Read-only Properties

`Score.contents_duration`

Note: Inherited from `containertools.Container`

`Score.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Score.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

`Score.engraver_removals`

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

Score.format

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

Score.is_semantic

Note: Inherited from `contexttools.Context`

Score.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Score.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Score.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Score.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`Score.parent`

Note: Inherited from `componenttools.Component`

`Score.preprolated_duration`

Note: Inherited from `containertools.Container`

`Score.prolated_duration`

Note: Inherited from `componenttools.Component`

`Score.prolation`

Note: Inherited from `componenttools.Component`

`Score.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`Score.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`Score.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`Score.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'

abjad> voice.is_nonsemantic = True

abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
```



```
{
    \time 1/8
    s1 * 1/8
}
{
    \time 5/16
    s1 * 5/16
}
{
    \time 5/16
    s1 * 5/16
}
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

Score.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True
```

```
abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Score.**name**

Read-write name of context. Must be string or none.

Note: Inherited from `contextttools.Context`

Methods

Score.**append**(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

Score.**extend**(*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`Score.index` (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`Score.insert` (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
```

```

        c' 8 [
        cs' 8
        d' 8
        e' 8 ]
    }

```

Return none.

Note: Inherited from `containertools.Container`

`Score.pop(i=-1)`

Pop component at index *i* from container:

```

abjad> container = Container("c' 8 d' 8 e' 8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c' 8 [
    d' 8
    e' 8 ]
}

abjad> container.pop(-1)
Note("e' 8")

abjad> f(container)
{
    c' 8 [
    d' 8 ]
}

```

Return component.

Note: Inherited from `containertools.Container`

`Score.remove(component)`

Remove *component* from container:

```

abjad> container = Container("c' 8 d' 8 e' 8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c' 8 [
    d' 8
    e' 8 ]
}

abjad> note = container[-1]
abjad> note
Note("e' 8")

abjad> container.remove(note)

```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`Score.__add__(expr)`

Concatenate containers self and *expr*. The operation `c = a + b` returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`Score.__contains__(expr)`

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Score.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Score.__delitem__(i)`

Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Score.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Score.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Score.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Score.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Score.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Score.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Score.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Score.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Score.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Score.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Score.__mul__(n)`

Note: Inherited from `componenttools.Component`

Score.**__ne__**(*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Score.**__radd__**(*expr*)

Extend container by contents of *expr* to the right.

Note: Inherited from `containertools.Container`

Score.**__repr__**()

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contextttools.Context`

Score.**__rmul__**(*n*)

Note: Inherited from `componentttools.Component`

Score.**__setattr__**()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Score.**__setitem__**(*i, expr*)

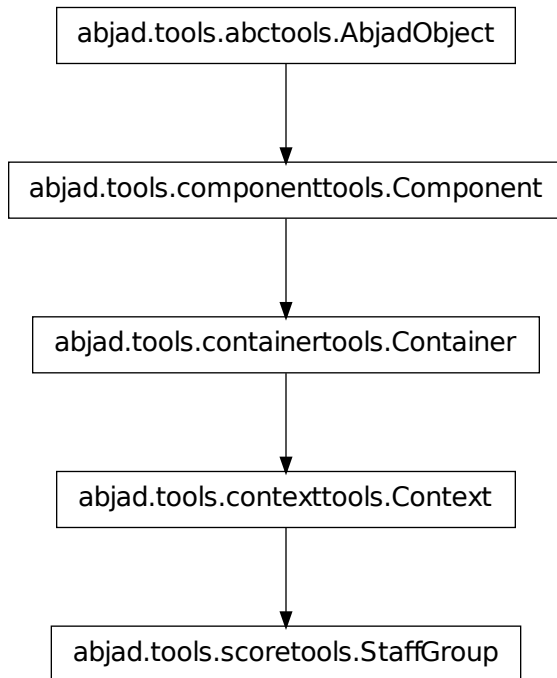
Set 'expr' in self at nonnegative integer index *i*. Or, set 'expr' in self at slice *i*. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

Score.**__str__**() <==> `str(x)`

Note: Inherited from `__builtin__.object`

scoretools.StaffGroup



class `abjad.tools.scoretools.StaffGroup.StaffGroup`.**StaffGroup** (*music=None*,
**kwargs)

Abjad model of staff group:

```

abjad> staff_1 = Staff("c'4 d'4 e'4 f'4 g'1")
abjad> staff_2 = Staff("g2 f2 e1")

abjad> staff_group = scoretools.StaffGroup([staff_1, staff_2])

abjad> f(staff_group)
\new StaffGroup <<
  \new Staff {
    c'4
    d'4
    e'4
    f'4
    g'1
  }
  \new Staff {
    g2
    f2
    e1
  }
>>
  
```


Return staff group.

Read-only Properties

`StaffGroup.contents_duration`

Note: Inherited from `containertools.Container`

`StaffGroup.duration_in_seconds`

Note: Inherited from `containertools.Container`

`StaffGroup.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

`StaffGroup.engraver_removals`

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

`StaffGroup.format`

Read-only LilyPond input format of component.

Note: Inherited from `componentttools.Component`

`StaffGroup.is_semantic`

Note: Inherited from `contextttools.Context`

StaffGroup.**leaves**

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

StaffGroup.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

StaffGroup.**music**

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

StaffGroup.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

StaffGroup.**parent**

Note: Inherited from `componenttools.Component`

StaffGroup.**preprolated_duration**

Note: Inherited from `containertools.Container`

StaffGroup.**prolated_duration**

Note: Inherited from `componenttools.Component`

StaffGroup.**prolation**

Note: Inherited from `componenttools.Component`

`StaffGroup.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`StaffGroup.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`StaffGroup.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`StaffGroup.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'
```

```
abjad> voice.is_nonsemantic = True
```

```
abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])
```

```
abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

StaffGroup.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

StaffGroup.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

`StaffGroup.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`StaffGroup.extend(expr)`

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`StaffGroup.index` (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`StaffGroup.insert` (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`StaffGroup.pop` (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.pop(-1)
Note("e'8")
```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`StaffGroup.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.remove(note)
```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`StaffGroup.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new `Container` c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`StaffGroup.__contains__(expr)`

True if expr is in container, otherwise False.

Note: Inherited from `containertools.Container`

```
StaffGroup.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
StaffGroup.__delitem__(i)  
Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).
```

Note: Inherited from `containertools.Container`

```
StaffGroup.__eq__(arg)  
True when id(self) equals id(arg).  
  
Return boolean.
```

Note: Inherited from `abctools.AbjadObject`

```
StaffGroup.__ge__(arg)  
Abjad objects by default do not implement this method.  
  
Raise exception.
```

Note: Inherited from `abctools.AbjadObject`

```
StaffGroup.__getitem__(i)  
Return component at index i in container. Shallow traversal of container for numeric indices only.
```

Note: Inherited from `containertools.Container`

```
StaffGroup.__gt__(arg)  
Abjad objects by default do not implement this method.  
  
Raise exception
```

Note: Inherited from `abctools.AbjadObject`

```
StaffGroup.__hash__() <==> hash(x)
```

Note: Inherited from `__builtin__.object`

```
StaffGroup.__iadd__(expr)  
__iadd__ avoids unnecessary copying of structures.
```

Note: Inherited from `containertools.Container`

```
StaffGroup.__imul__(total)  
Multiply contents of container 'total' times. Return multiplied container.
```

Note: Inherited from `containertools.Container`

`StaffGroup.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffGroup.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`StaffGroup.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffGroup.__mul__(n)`

Note: Inherited from `componenttools.Component`

`StaffGroup.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`StaffGroup.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`StaffGroup.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`StaffGroup.__rmul__(n)`

Note: Inherited from `componenttools.Component`

```
StaffGroup.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
StaffGroup.__setitem__(i, expr)
```

Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

```
StaffGroup.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

scoretools functions

scoretools.add_double_bar_to_end_of_score

```
abjad.tools.scoretools.add_double_bar_to_end_of_score.add_double_bar_to_end_of_score(score)
```

New in version 2.0. Add double bar to end of *score*:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")  
  
abjad> scoretools.add_double_bar_to_end_of_score(staff)  
BarLine('|.')(f'4)  
  
abjad> f(staff)  
\new Staff {  
    c'4  
    d'4  
    e'4  
    f'4  
    \bar "|."  
}
```

Return double bar.

scoretools.add_markup_to_end_of_score

```
abjad.tools.scoretools.add_markup_to_end_of_score.add_markup_to_end_of_score(score,  
                                                                              markup,  
                                                                              ex-  
                                                                              tra_offset=None)
```

New in version 2.0. Add *markup* to end of *score*:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")  
abjad> markup = r'\italic \right-column { "Bremen - Boston - Los Angeles." "Jul 2010 - May 2011."  
abjad> markup = markuptools.Markup(markup, 'down')
```

```

abjad> scoretools.add_markup_to_end_of_score(staff, markup, (4, -2))
Markup('\\italic \\right-column { "Bremen - Boston - Los Angeles." "Jul 2010 - May 2011." }', di

abjad> f(staff)
\\new Staff {
    c'4
    d'4
    e'4
    \\once \\override TextScript #'extra-offset = #'(4 . -2)
    f'4 _ \\markup { \\italic \\right-column { "Bremen - Boston - Los Angeles." "Jul 2010 - May 2011." } }
}

```

Return *markup*.

scoretools.all_are_scores

`abjad.tools.scoretools.all_are_scores.all_are_scores(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad scores:

```

abjad> score = Score([Staff([Note("c'4")])])

abjad> score
Score<<1>>

```

```

abjad> scoretools.all_are_scores([score])
True

```

True when *expr* is an empty sequence:

```

abjad> scoretools.all_are_scores([])
True

```

Otherwise false:

```

abjad> scoretools.all_are_scores('foo')
False

```

Return boolean.

Function wraps `componenttools.all_are_components()`.

scoretools.get_first_score_in_improper_parentage_of_component

`abjad.tools.scoretools.get_first_score_in_improper_parentage_of_component.get_first_score_in_improper_parentage_of_component(component)`

New in version 2.0. Get first score in improper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score = Score([staff])

abjad> f(score)
\\new Score <<
  \\new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

```

```
abjad> scoretools.get_first_score_in_improper_parentage_of_component(score.leaves[0])
Score<<1>>
```

Return score or none.

scoretools.get_first_score_in_proper_parentage_of_component

abjad.tools.scoretools.get_first_score_in_proper_parentage_of_component.**get_first_score_in**
 New in version 2.0. Get first score in proper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> score = Score([staff])

abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
>>

abjad> scoretools.get_first_score_in_proper_parentage_of_component(score.leaves[0])
Score<<1>>
```

Return score or none.

scoretools.iterate_scores_backward_in_expr

abjad.tools.scoretools.iterate_scores_backward_in_expr.**iterate_scores_backward_in_expr** (*expr*,
start=
stop=

New in version 2.0. Iterate scores backward in *expr*:

```
abjad> score_1 = Score([Staff("c'8 d'8 e'8 f'8")])
abjad> score_2 = Score([Staff("c'1"), Staff("g'1")])
abjad> scores = [score_1, score_2]

abjad> for score in scoretools.iterate_scores_backward_in_expr(scores):
...     score
Score<<2>>
Score<<1>>
```

Ignore threads.

Return generator.

scoretools.iterate_scores_forward_in_expr

abjad.tools.scoretools.iterate_scores_forward_in_expr.**iterate_scores_forward_in_expr** (*expr*,
start=0,
stop=No

New in version 2.0. Iterate scores forward in *expr*:

```
abjad> score_1 = Score([Staff("c'8 d'8 e'8 f'8")])
abjad> score_2 = Score([Staff("c'1"), Staff("g'1")])
abjad> scores = [score_1, score_2]

abjad> for score in scoretools.iterate_scores_forward_in_expr(scores):
...     score
Score<<1>>
Score<<2>>
```

Ignore threads.

Return generator.

scoretools.list_performer_names

`abjad.tools.scoretools.list_performer_names.list_performer_names(locale='en-us')`

New in version 2.5. List performer names:

```
abjad> for performer_name in scoretools.list_performer_names():
...     performer_name
...
'accordionist'
'baritone'
'bass'
'bassist'
'bassoonist'
'cellist'
'clarinetist'
'contralto'
'flutist'
'guitarist'
'harpist'
'harpsichordist'
'hornist'
'mezzo-soprano'
'oboist'
'percussionist'
'pianist'
'saxophonist'
'soprano'
'tenor'
'trombonist'
'trumpeter'
'tubist'
'vibraphonist'
'violinist'
'violist'
'xylophonist'
```

Available values for *locale* are 'en-us' and 'en-uk'.

scoretools.list_primary_performer_names

`abjad.tools.scoretools.list_primary_performer_names.list_primary_performer_names()`

New in version 2.5. List performer names:

```
abjad> for pair in scoretools.list_primary_performer_names():
...     pair
...
('accordionist', 'acc.')
('baritone', 'baritone')
('bass', 'bass')
('bassist', 'vb.')
('bassoonist', 'bsn.')
('cellist', 'vc.')
('clarinetist', 'cl. in B-flat')
('contralto', 'contralto')
('flutist', 'fl.')
('guitarist', 'gt.')
('harpist', 'hp.')
('harpsichordist', 'hpschd.')
('hornist', 'hn.')
('mezzo-soprano', 'mezzo-soprano')
('oboist', 'ob.')
('pianist', 'pf.')
('saxophonist', 'alto sax.')
('soprano', 'soprano')
('tenor', 'tenor')
('trombonist', 'ten. trb.')
('trumpeter', 'tp.')
('tubist', 'tb.')
('violinist', 'vn.')
('violist', 'va.')
```

Return list.

scoretools.make_empty_piano_score

abjad.tools.scoretools.make_empty_piano_score.make_empty_piano_score()

New in version 1.1. Make empty piano score:

```
abjad> score, treble, bass = scoretools.make_empty_piano_score()
```

```
abjad> f(score)
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
    }
    \context Staff = "bass" {
      \clef "bass"
    }
  >>
>>
```

Return score, treble staff, bass staff. Changed in version 2.0: renamed scoretools.make_piano_staff() to scoretools.make_empty_piano_score().

scoretools.make_piano_score_from_leaves

abjad.tools.scoretools.make_piano_score_from_leaves.**make_piano_score_from_leaves** (*leaves*,
low-
est_treble_pit)

New in version 2.0. Make piano score from *leaves*:

```
abjad> notes = [Note(x, (1, 4)) for x in [-12, 37, -10, 2, 4, 17]]
abjad> score, treble_staff, bass_staff = scoretools.make_piano_score_from_leaves(notes)

abjad> f(score)
\new Score <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      r4
      cs''''4
      r4
      d'4
      e'4
      f''4
    }
    \context Staff = "bass" {
      \clef "bass"
      c4
      r4
      d4
      r4
      r4
      r4
    }
  >>
>>
```

Return score, treble staff, bass staff.

scoretools.make_piano_sketch_score_from_leaves

abjad.tools.scoretools.make_piano_sketch_score_from_leaves.**make_piano_sketch_score_from_leaves**

New in version 2.0. Make piano sketch score from *leaves*:

```
abjad> notes = notetools.make_notes([-12, -10, -8, -7, -5, 0, 2, 4, 5, 7], [(1, 4)])
abjad> score, treble_staff, bass_staff = scoretools.make_piano_sketch_score_from_leaves(notes)

abjad> f(score)
\new Score \with {
  \override BarLine #'stencil = ##f
  \override BarNumber #'transparent = ##t
  \override SpanBar #'stencil = ##f
  \override TimeSignature #'stencil = ##f
} <<
  \new PianoStaff <<
    \context Staff = "treble" {
      \clef "treble"
      #(set-accidental-style 'forget)
```

```

        r4
        r4
        r4
        r4
        r4
        c'4
        d'4
        e'4
        f'4
        g'4
    }
    \context Staff = "bass" {
        \clef "bass"
        #(set-accidental-style 'forget)
        c4
        d4
        e4
        f4
        g4
        r4
        r4
        r4
        r4
        r4
    }
>>
>>

```

Make time signatures and bar numbers transparent.

Do not print bar lines or span bars.

Set all staff accidental styles to forget.

Return score, treble staff, bass staff.

scoretools.make_pitch_array_score_from_pitch_arrays

abjad.tools.scoretools.make_pitch_array_score_from_pitch_arrays.**make_pitch_array_score_from**

New in version 2.0. Make pitch-array score from *pitch_arrays*:

```

abjad> from abjad.tools import pitcharraytools

abjad> array_1 = pitcharraytools.PitchArray([
...     [1, (2, 1), ([-2, -1.5], 2)],
...     [(7, 2), (6, 1), 1]])

abjad> array_2 = pitcharraytools.PitchArray([
...     [1, 1, 1],
...     [1, 1, 1]])

abjad> score = scoretools.make_pitch_array_score_from_pitch_arrays([array_1, array_2])

abjad> f(score)
\new Score <<
  \new StaffGroup <<
    \new Staff {
      {

```



```
        \time 4/8
        r8
        d'8
        <bf bqf>4
    }
    {
        \time 3/8
        r8
        r8
        r8
    }
}
\new Staff {
    {
        \time 4/8
        g'4
        fs'8
        r8
    }
    {
        \time 3/8
        r8
        r8
        r8
    }
}
>>
>>
```

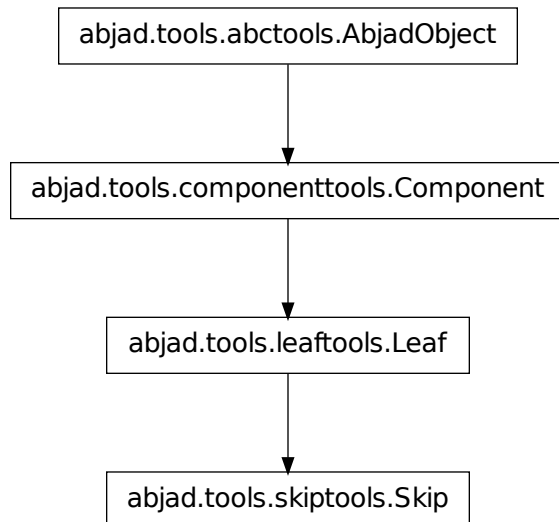
Create one staff per pitch-array row.

Return score.

skiptools

skiptools concrete classes

skiptools.Skip



class `abjad.tools.skiptools.Skip.Skip`.**Skip**(*args, **kwargs)

Abjad model of a LilyPond skip:

```
abjad> skiptools.Skip((3, 16))
Skip('s8.')
```

Return skip.

Read-only Properties

`Skip.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`Skip.format`

Note: Inherited from `leaftools.Leaf`

`Skip.leaf_index`

Note: Inherited from `leaftools.Leaf`

Skip.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Skip.**multiplied_duration**

Note: Inherited from `leaftools.Leaf`

Skip.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Skip.**parent**

Note: Inherited from `componenttools.Component`

Skip.**preprolated_duration**

Note: Inherited from `leaftools.Leaf`

Skip.**prolated_duration**

Note: Inherited from `componenttools.Component`

Skip.**prolation**

Note: Inherited from `componenttools.Component`

Skip.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Skip.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Skip.`duration_multiplier`

Note: Inherited from `leaftools.Leaf`

Skip.`written_duration`

Note: Inherited from `leaftools.Leaf`

Skip.`written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

Skip.`written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

Skip.`__and__`(*arg*)

Note: Inherited from `leaftools.Leaf`

Skip.`__delattr__`()
x.`__delattr__`('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Skip.`__eq__`(*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

Skip.`__ge__`(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Skip.`__gt__`(*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`Skip.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`Skip.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Skip.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Skip.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Skip.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Skip.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`Skip.__repr__()`

Note: Inherited from `leaftools.Leaf`

`Skip.__rmul__(n)`

Note: Inherited from `componenttools.Component`

```
Skip.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
Skip.__str__()
```

Note: Inherited from `leaftools.Leaf`

```
Skip.__sub__(arg)
```

Note: Inherited from `leaftools.Leaf`

```
Skip.__xor__(arg)
```

Note: Inherited from `leaftools.Leaf`

skiptools functions

skiptools.all_are_skips

`abjad.tools.skiptools.all_are_skips.all_are_skips(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad skips:

```
abjad> from abjad.tools import skiptools
```

```
abjad> skips = 3 * skiptools.Skip('s4')
```

```
abjad> skips  
[Skip('s4'), Skip('s4'), Skip('s4')]
```

```
abjad> skiptools.all_are_skips(skips)  
True
```

True when *expr* is an empty sequence:

```
abjad> skiptools.all_are_skips([])  
True
```

Otherwise false:

```
abjad> skiptools.all_are_skips('foo')  
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

skiptools.iterate_skips_backward_in_expr

abjad.tools.skiptools.iterate_skips_backward_in_expr.**iterate_skips_backward_in_expr**(*expr*,
start=0,
stop=None)

New in version 2.0. Iterate skips backward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 s8 <d' f' b'>8 s2")

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    s8
    <d' f' b'>8
    s2
}

abjad> for skip in skiptools.iterate_skips_backward_in_expr(staff):
...     skip
Skip('s2')
Skip('s8')
```

Ignore threads.

Return generator.

skiptools.iterate_skips_forward_in_expr

abjad.tools.skiptools.iterate_skips_forward_in_expr.**iterate_skips_forward_in_expr**(*expr*,
start=0,
stop=None)

New in version 2.0. Iterate skips forward in *expr*:

```
abjad> staff = Staff("<e' g' c''>8 a'8 s8 <d' f' b'>8 s2")

abjad> f(staff)
\new Staff {
    <e' g' c''>8
    a'8
    s8
    <d' f' b'>8
    s2
}

abjad> for skip in skiptools.iterate_skips_forward_in_expr(staff):
...     skip
Skip('s8')
Skip('s2')
```

Ignore threads.

Return generator.

`skiptools.make_repeated_skips_from_time_signature`

`abjad.tools.skiptools.make_repeated_skips_from_time_signature`.**`make_repeated_skips_from_time_signature`**
 New in version 2.0. Make repeated skips from *time_signature*:

```
abjad> skiptools.make_repeated_skips_from_time_signature((5, 32))
[Skip('s32'), Skip('s32'), Skip('s32'), Skip('s32'), Skip('s32')]
```

Return list of skips.

`skiptools.make_repeated_skips_from_time_signatures`

`abjad.tools.skiptools.make_repeated_skips_from_time_signatures`.**`make_repeated_skips_from_time_signatures`**
 Make repeated skips from *time_signatures*:

```
skiptools.make_repeated_skips_from_time_signatures([(2, 8), (3, 32)])
[[Skip('s8'), Skip('s8')], [Skip('s32'), Skip('s32'), Skip('s32')]]
```

Return list of skip lists.

`skiptools.make_skips_with_multiplied_durations`

`abjad.tools.skiptools.make_skips_with_multiplied_durations`.**`make_skips_with_multiplied_durations`**

New in version 2.0. Make *written_duration* skips with *multiplied_durations*:

```
abjad> skiptools.make_skips_with_multiplied_durations(Duration(1, 4), [(1, 2), (1, 3), (1, 4), (1, 5)])
[Skip('s4 * 2'), Skip('s4 * 4/3'), Skip('s4 * 1'), Skip('s4 * 4/5')]
```

Useful for making invisible layout voices.

Return list of skips. Changed in version 2.0: renamed `construct.skips_with_multipliers()` to `skiptools.make_skips_with_multiplied_durations()`.

`skiptools.replace_leaves_in_expr_with_skips`

`abjad.tools.skiptools.replace_leaves_in_expr_with_skips`.**`replace_leaves_in_expr_with_skips`**
 New in version 1.1. Replace leaves in *expr* with skips:

```
abjad> staff = Staff(Measure((2, 8), "c'8 d'8") * 2)
abjad> skiptools.replace_leaves_in_expr_with_skips(staff[0])
abjad> print staff.format
\new Staff {
    {
        \time 2/8
        s8
        s8
    }
    {
        \time 2/8
        c'8
        d'8
    }
}
```



```
}
}
```

Return none. Changed in version 2.0: renamed `leaftools.replace_leaves_with_skips_in()` to `skiptools.replace_leaves_in_expr_with_skips()`.

skiptools.yield_groups_of_skips_in_sequence

`abjad.tools.skiptools.yield_groups_of_skips_in_sequence.yield_groups_of_skips_in_sequence(s`

New in version 2.0. Yield groups of skips in *sequence*:

```
abjad> staff = Staff("c'8 d'8 s8 s8 <e' g'>8 <f' a'>8 g'8 a'8 s8 s8 <b' d''>8 <c'' e''>8")
```

```
abjad> f(staff)
```

```
\new Staff {
    c'8
    d'8
    s8
    s8
    <e' g'>8
    <f' a'>8
    g'8
    a'8
    s8
    s8
    <b' d''>8
    <c'' e''>8
}
```

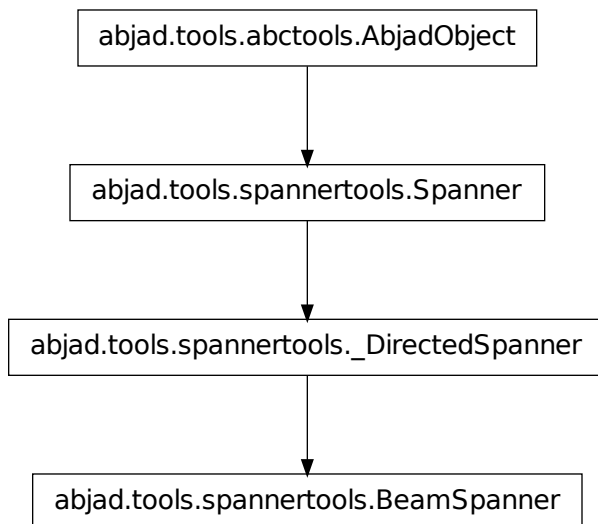
```
abjad> for skip in skiptools.yield_groups_of_skips_in_sequence(staff):
...     skip
...
(Skip('s8'), Skip('s8'))
(Skip('s8'), Skip('s8'))
```

Return generator.

`spannertools`

`spannertools` concrete classes

`spannertools.BeamSpanner`



class `abjad.tools.spannertools.BeamSpanner.BeamSpanner` (*components=None, direction=None*)

Abjad beam spanner:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'2")
```

```
abjad> f(staff)
```

```
\new Staff {
  c'8
  d'8
  e'8
  f'8
  g'2
}
```

```
abjad> spannertools.BeamSpanner(staff[:4])
```

```
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> f(staff)
```

```
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
```

```

        g'2
    }

```

Return beam spanner.

Read-only Properties

`BeamSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(e'8, f'8)

```

```

abjad> spanner._offset.start
Offset(1, 4)

```

```

abjad> spanner._offset.stop
Offset(1, 2)

```

Return duration.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`BeamSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

Methods

`BeamSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}
```

```
abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`BeamSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`BeamSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BeamSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BeamSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`BeamSpanner.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`BeamSpanner.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BeamSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.__lt__(other)`
 Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`BeamSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BeamSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

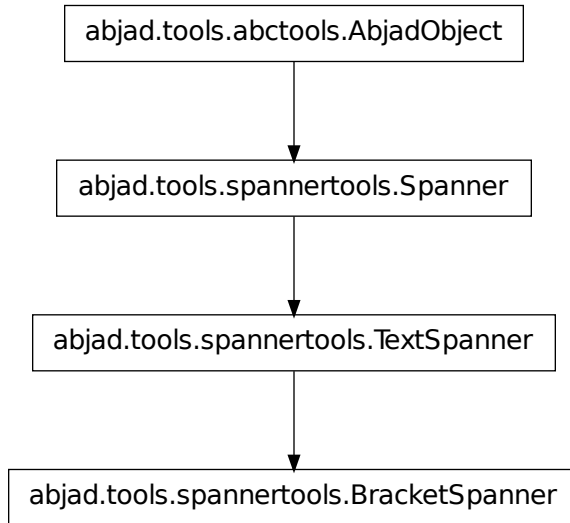
`BeamSpanner.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`BeamSpanner.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

spannertools.BracketSpanner



class `abjad.tools.spannertools.BracketSpanner.BracketSpanner` (*components=None*)
 Abjad bracket spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.BracketSpanner(staff[:])
BracketSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = #(markup #:draw-line '(0 . -1))
  \override TextSpanner #'bound-details #'left-broken #'text = ##f
  \override TextSpanner #'bound-details #'right #'text = #(markup #:draw-line '(0 . -1))
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'color = #red
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'staff-padding = #2
  \override TextSpanner #'thickness = #1.5
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'left-broken #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'bound-details #'right-broken #'text
  \revert TextSpanner #'color
  \revert TextSpanner #'dash-fraction
  \revert TextSpanner #'staff-padding
  \revert TextSpanner #'thickness

```

```
}
```

Render 1.5-unit thick solid red spanner.

Draw nibs at beginning and end of spanner.

Do not draw nibs at line breaks.

Return bracket spanner.

Read-only Properties

`BracketSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop  
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

`BracketSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:2])  
abjad> spanner  
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])  
abjad> spanner  
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
```

```
c'8 [  
d'8  
e'8  
f'8 ]  
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[2:])  
abjad> spanner  
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])  
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:])  
abjad> spanner  
Spanner(c'8, d'8, e'8, f'8)  
  
abjad> spanner.pop()  
Note("f'8")  
  
abjad> spanner  
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:])  
abjad> spanner  
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()  
Note("c'8")
```

```
abjad> spanner  
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`BracketSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`BracketSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BracketSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BracketSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`BracketSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`BracketSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BracketSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BracketSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

`BracketSpanner.__setattr__()`

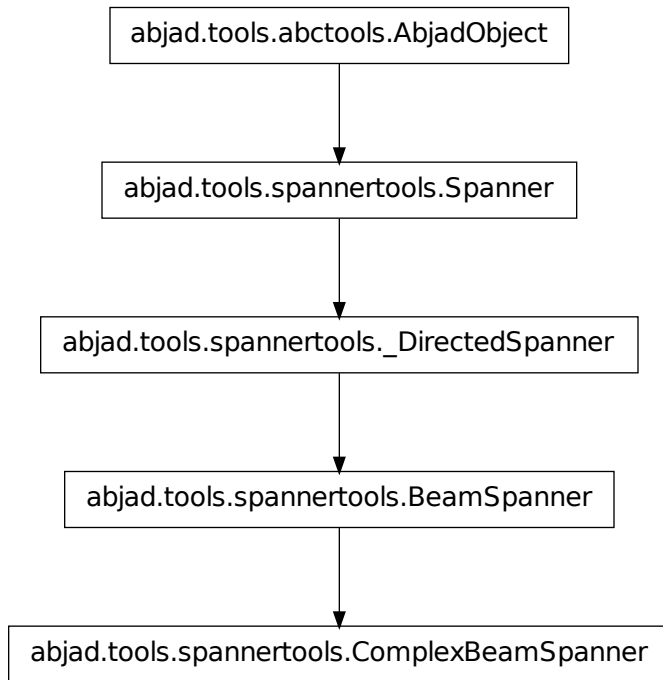
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BracketSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.ComplexBeamSpanner



class `abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner` (*complex beam spanner*)

Abjad complex beam spanner:

```

abjad> staff = Staff("c'16 e'16 r16 f'16 g'2")

abjad> f(staff)
\new Staff {
    c'16
    e'16
    r16
    f'16
    g'2
}

abjad> spannertools.ComplexBeamSpanner(staff[:4])
ComplexBeamSpanner(c'16, e'16, r16, f'16)

abjad> f(staff)
\new Staff {
    \set stemLeftBeamCount = #0
    \set stemRightBeamCount = #2

```

```
c'16 [  
  \set stemLeftBeamCount = #2  
  \set stemRightBeamCount = #2  
e'16 ]  
r16  
  \set stemLeftBeamCount = #2  
  \set stemRightBeamCount = #0  
f'16 [ ]  
g'2  
}
```

Return complex beam spanner.

Read-only Properties

`ComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:2])  
abjad> spanner.components  
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:2])  
abjad> spanner.leaves  
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[2:])
```

```
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner._offset.start
Offset(1, 4)

abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.override`
LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.preprolated_duration`
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.prolated_duration`
Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.set`
LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.written_duration`
Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`ComplexBeamSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

`ComplexBeamSpanner.lone`
Beam lone leaf and force beam nibs to left:

```
abjad> note = Note("c'16")
```

```
abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'left')
```

```
abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'right')

abjad> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'both')

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = True)

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = False)

abjad> f(note)
c'16
```

Set to 'left', 'right', 'both', true or false as shown above.

Ignore this setting when spanner contains more than one leaf.

Methods

`ComplexBeamSpanner.append(component)`
Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.append_left(component)`
Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.clear()`
Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.extend(components)`
Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\nnew Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])
```

```
abjad> print voice.format
\\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from [spannertools.Spanner](#)

`ComplexBeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from [spannertools.Spanner](#)

`ComplexBeamSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`ComplexBeamSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ComplexBeamSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__hash__()` $\iff hash(x)$

Note: Inherited from `__builtin__.object`

`ComplexBeamSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ComplexBeamSpanner.__repr__()`

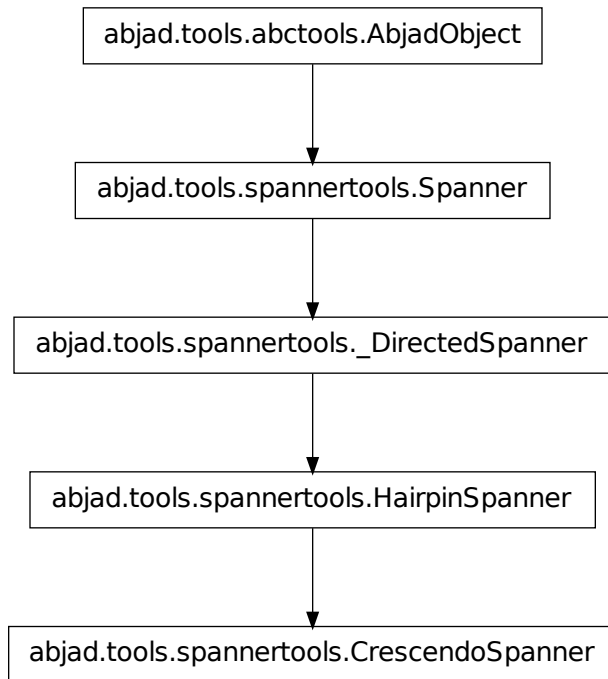
Note: Inherited from `spannertools.Spanner`

`ComplexBeamSpanner.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`

Note: Inherited from `__builtin__.object`

`ComplexBeamSpanner.__str__()` $\iff str(x)$

Note: Inherited from `__builtin__.object`

spannertools.CrescendoSpanner

class `abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner` (*components=None*, *include_rests=True*, *direction=None*)

Abjad crescendo spanner that includes rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")
```

```
abjad> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}
```

```
abjad> spannertools.CrescendoSpanner(staff[:], include_rests = True)
CrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)
```

```
abjad> f(staff)
\new Staff {
    r4 \<
    c'8
    d'8
    e'8
    f'8
    r4 \!
}
```

Abjad crescendo spanner that does not include rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

abjad> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

abjad> spannertools.CrescendoSpanner(staff[:], include_rests = False)
CrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)

abjad> f(staff)
\new Staff {
    r4
    c'8 \<
    d'8
    e'8
    f'8 \!
    r4
}
```

Return crescendo spanner.

Read-only Properties

CrescendoSpanner.components

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

CrescendoSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`CrescendoSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

`CrescendoSpanner.include_rests`

Get boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests = False
abjad> hairpin.include_rests
False
```

Set boolean.

Note: Inherited from `spannertools.HairpinSpanner`

`CrescendoSpanner.shape_string`

Get hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string = '>'
abjad> hairpin.shape_string
'>'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

CrescendoSpanner.start_dynamic_string

Get hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string = 'mf'
abjad> hairpin.start_dynamic_string
'mf'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

CrescendoSpanner.stop_dynamic_string

Get hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string = 'mf'
abjad> hairpin.stop_dynamic_string
'mf'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

Methods**CrescendoSpanner.append** (*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```



```

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

```

Return none.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}

```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {

```

```
c'8 [
d'8
e'8
f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`CrescendoSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`CrescendoSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

`CrescendoSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`CrescendoSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`CrescendoSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

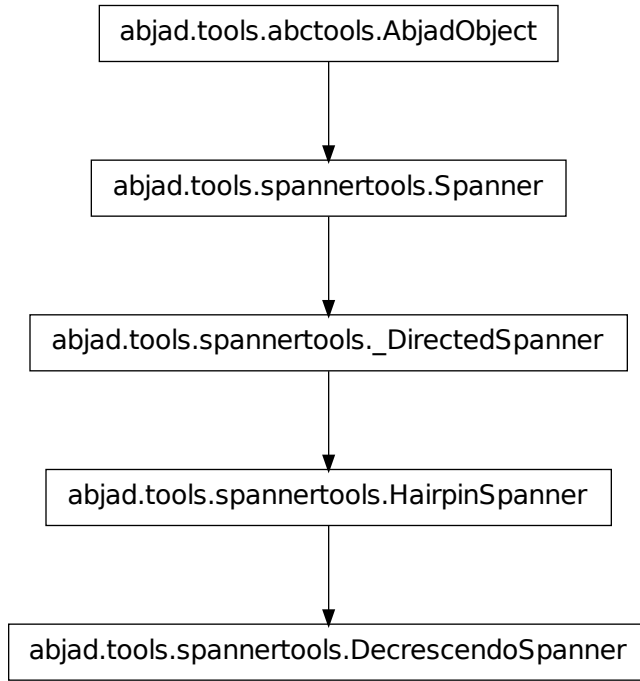
`CrescendoSpanner.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`CrescendoSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.DecrescendoSpanner

class `abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner`.**DecrescendoSpanner** *(composition=...*
include_
di-
rec-
tion=N

Abjad decrescendo spanner that includes rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")
```

```
abjad> f(staff)
\new Staff {
  r4
  c'8
  d'8
  e'8
  f'8
  r4
}
```

```
abjad> spannertools.DecrescendoSpanner(staff[:], include_rests = True)
DecrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)
```

```
abjad> f(staff)
\new Staff {
  r4 \>
  c'8
  d'8
  e'8
  f'8
  r4 \!
}
```

Abjad decrescendo spanner that does not include rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")
```

```
abjad> f(staff)
\new Staff {
  r4
  c'8
  d'8
  e'8
  f'8
  r4
}
```

```
abjad> spannertools.DecrescendoSpanner(staff[:], include_rests = False)
DecrescendoSpanner(r4, c'8, d'8, e'8, f'8, r4)
```

```
abjad> f(staff)
\new Staff {
  r4
  c'8 \>
  d'8
  e'8
  f'8 \!
  r4
}
```

Return decrescendo spanner.

Read-only Properties

`DecrescendoSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

DecrescendoSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`DecrescendoSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

`DecrescendoSpanner.include_rests`

Get boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests = False
abjad> hairpin.include_rests
False
```

Set boolean.

Note: Inherited from `spannertools.HairpinSpanner`

`DecrescendoSpanner.shape_string`

Get hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string = '>'
abjad> hairpin.shape_string
'>'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

DecrescendoSpanner.start_dynamic_string

Get hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string = 'mf'
abjad> hairpin.start_dynamic_string
'mf'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

DecrescendoSpanner.stop_dynamic_string

Get hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string = 'mf'
abjad> hairpin.stop_dynamic_string
'mf'
```

Set string.

Note: Inherited from `spannertools.HairpinSpanner`

Methods**DecrescendoSpanner.append(component)**

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

```

Return none.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}

```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {

```

```
c'8 [  
d'8  
e'8  
f'8 ]  
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[2:])  
abjad> spanner  
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])  
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:])  
abjad> spanner  
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()  
Note("f'8")
```

```
abjad> spanner  
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")  
abjad> spanner = spannertools.Spanner(voice[:])  
abjad> spanner  
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`DecrescendoSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DecrescendoSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__hash__()` <==> `hash(x)`

Note: Inherited from `__builtin__.object`

`DecrescendoSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DecrescendoSpanner.__repr__()`

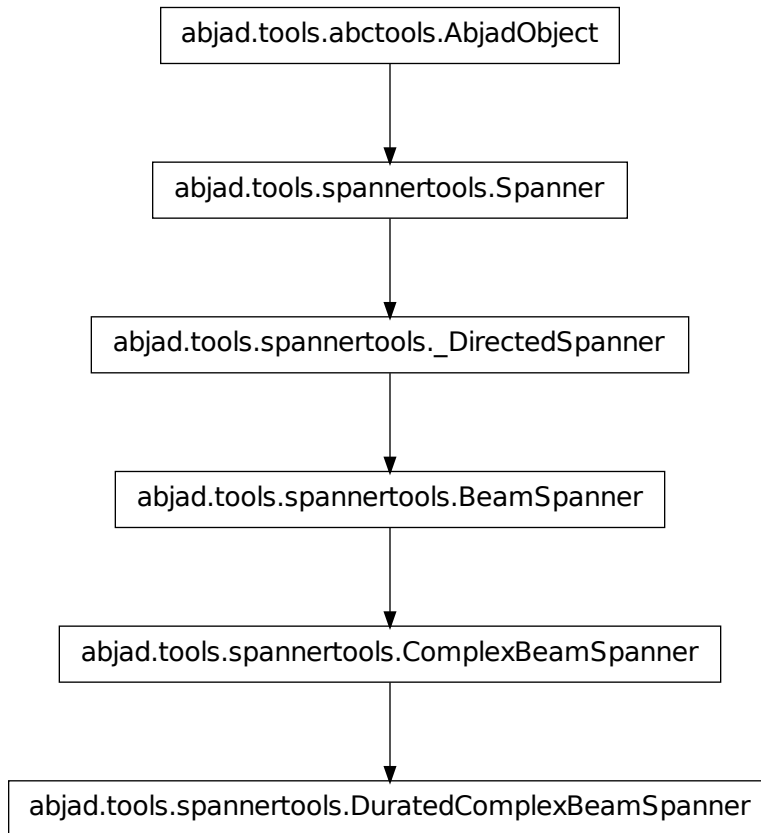
Note: Inherited from `spannertools.Spanner`

`DecrescendoSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DecrescendoSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.DuratedComplexBeamSpanner

class `abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner`

Abjad durated complex beam spanner:

```

staff = Staff("c'16 d'16 e'16 f'16")

durations = [Duration(1, 8), Duration(1, 8)]
beam = spannertools.DuratedComplexBeamSpanner(staff[:], durations, 1)

f(staff)
\new Staff {
    \set stemLeftBeamCount = #0

```

```

\set stemRightBeamCount = #2
c'16 [
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #1
d'16
\set stemLeftBeamCount = #1
\set stemRightBeamCount = #2
e'16
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
f'16 ]
}

```

Beam all beamable leaves in spanner explicitly.

Group leaves in spanner according to *durations*.

Span leaves between duration groups according to *span*.

Return durated complex beam spanner.

Read-only Properties

`DuratedComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements start and stop attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

DuratedComplexBeamSpanner.written_duration

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties**DuratedComplexBeamSpanner.direction**

Note: Inherited from `spannertools._DirectedSpanner`

DuratedComplexBeamSpanner.durations

Get spanner leaf group durations:

```
abjad> staff = Staff("c'16 d'16 e'16 f'16")
abjad> durations = [Duration(1, 8), Duration(1, 8)]
abjad> beam = spannertools.DuratedComplexBeamSpanner(staff[:], durations)
abjad> beam.durations
[Duration(1, 8), Duration(1, 8)]
```

Set spanner leaf group durations:

```
abjad> staff = Staff("c'16 d'16 e'16 f'16")
abjad> durations = [Duration(1, 8), Duration(1, 8)]
abjad> beam = spannertools.DuratedComplexBeamSpanner(staff[:], durations)
abjad> beam.durations = [Duration(1, 4)]
abjad> beam.durations
[Duration(1, 4)]
```

Set iterable.

DuratedComplexBeamSpanner.lone

Beam lone leaf and force beam nibs to left:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'left')

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'right')

abjad> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'both')

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = True)
```

```
abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = False)

abjad> f(note)
c'16
```

Set to 'left', 'right', 'both', true or false as shown above.

Ignore this setting when spanner contains more than one leaf.

Note: Inherited from `spannertools.ComplexBeamSpanner`

`DuratedComplexBeamSpanner`. **span**

Get top-level beam count:

```
abjad> staff = Staff("c'16 d'16 e'16 f'16")
abjad> durations = [Duration(1, 8), Duration(1, 8)]
abjad> beam = spannertools.DuratedComplexBeamSpanner(staff[:], durations, 1)
abjad> beam.span
1
```

Set top-level beam count:

```
abjad> staff = Staff("c'16 d'16 e'16 f'16")
abjad> durations = [Duration(1, 8), Duration(1, 8)]
abjad> beam = spannertools.DuratedComplexBeamSpanner(staff[:], durations, 1)
abjad> beam.span = 2
abjad> beam.span
2
```

Set nonnegative integer.

Methods

`DuratedComplexBeamSpanner`. **append** (*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
```

```
c'8 [
d'8
e'8
f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()  
Note("c'8")
```

```
abjad> spanner  
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`DuratedComplexBeamSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DuratedComplexBeamSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DuratedComplexBeamSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DuratedComplexBeamSpanner.__repr__()`

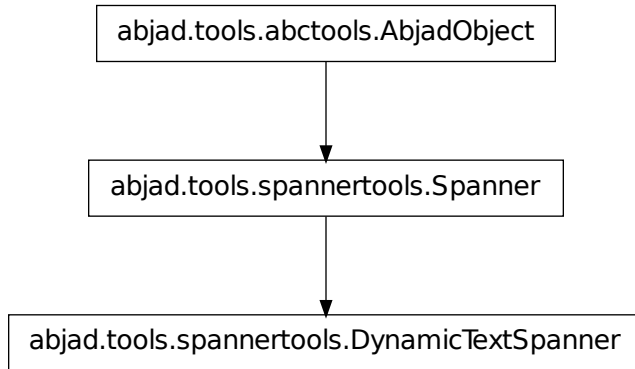
Note: Inherited from `spannertools.Spanner`

`DuratedComplexBeamSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DuratedComplexBeamSpanner.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.DynamicTextSpanner

class `abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner` (*component*, *mark=*)

Abjad dynamic text spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.DynamicTextSpanner(staff[:], 'f')
DynamicTextSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 \f
    d'8
    e'8
    f'8
}
  
```

Format dynamic *mark* at first leaf in spanner.

Return dynamic text spanner.

Read-only Properties

`DynamicTextSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`DynamicTextSpanner.mark`

Get dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> dynamic_text_spanner = spannertools.DynamicTextSpanner(staff[:], 'f')
abjad> dynamic_text_spanner.mark
'f'
```

Set dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> dynamic_text_spanner = spannertools.DynamicTextSpanner(staff[:], 'f')
abjad> dynamic_text_spanner.mark = 'p'
abjad> dynamic_text_spanner.mark
'p'
```

Set string.

Methods

`DynamicTextSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`DynamicTextSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DynamicTextSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DynamicTextSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DynamicTextSpanner.__repr__()`

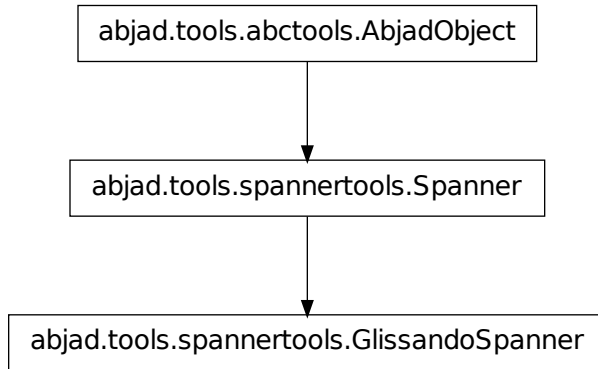
Note: Inherited from `spannertools.Spanner`

`DynamicTextSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DynamicTextSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.GlissandoSpanner

class `abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner` (*components=None*)
 Abjad glissando spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.GlissandoSpanner(staff[:])
GlissandoSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 \glissando
    d'8 \glissando
    e'8 \glissando
    f'8
}
  
```

Format nonlast leaves in spanner with LilyPond `glissando` command.

Return glissando spanner.

Read-only Properties

`GlissandoSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

GlissandoSpanner.**set**

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

GlissandoSpanner.**written_duration**

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

GlissandoSpanner.**append**(*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

GlissandoSpanner.**append_left**(*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

GlissandoSpanner.**clear**()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))
```

```
abjad> print voice.format
\nnew Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\nnew Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\nnew Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`GlissandoSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`GlissandoSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`GlissandoSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`GlissandoSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlissandoSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

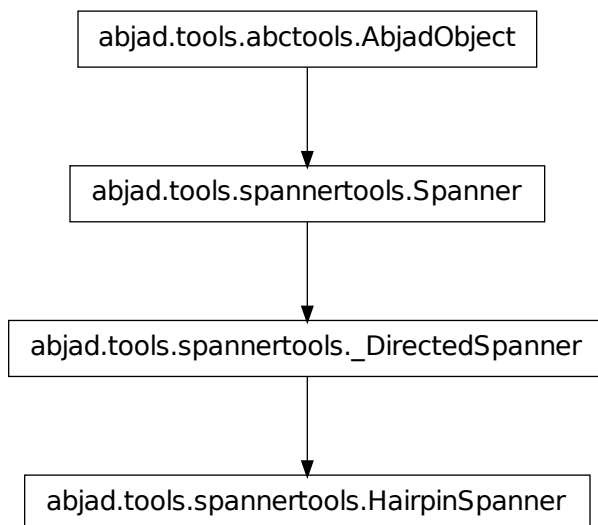
`GlissandoSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`GlissandoSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.HairpinSpanner



class `abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner` (*components=None, de-scrip-tor='<', in-clude_rests=True, di-rec-tion=None*)

Abjad hairpin spanner that includes rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

abjad> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

abjad> spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
HairpinSpanner(r4, c'8, d'8, e'8, f'8, r4)

abjad> f(staff)
\new Staff {
    r4 \< \p
    c'8
    d'8
    e'8
    f'8
    r4 \f
}
```

Abjad hairpin spanner that does not include rests:

```
abjad> staff = Staff("r4 c'8 d'8 e'8 f'8 r4")

abjad> f(staff)
\new Staff {
    r4
    c'8
    d'8
    e'8
    f'8
    r4
}

abjad> spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = False)
HairpinSpanner(r4, c'8, d'8, e'8, f'8, r4)

abjad> f(staff)
\new Staff {
    r4
    c'8 \< \p
    d'8
```

```
e'8
f'8 \f
r4
}
```

Return hairpin spanner.

Read-only Properties

HairpinSpanner.components

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

HairpinSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

HairpinSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

HairpinSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`HairpinSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

`HairpinSpanner.include_rests`

Get boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests
True
```

Set boolean hairpin rests setting:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f', include_rests = True)
abjad> hairpin.include_rests = False
abjad> hairpin.include_rests
False
```

Set boolean.

HairpinSpanner.shape_string

Get hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string
'<'
```

Set hairpin shape string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.shape_string = '>'
abjad> hairpin.shape_string
'>'
```

Set string.

HairpinSpanner.start_dynamic_string

Get hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string
'p'
```

Set hairpin start dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.start_dynamic_string = 'mf'
abjad> hairpin.start_dynamic_string
'mf'
```

Set string.

HairpinSpanner.stop_dynamic_string

Get hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string
'f'
```

Set hairpin stop dynamic string:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> hairpin = spannertools.HairpinSpanner(staff[:], 'p < f')
abjad> hairpin.stop_dynamic_string = 'mf'
abjad> hairpin.stop_dynamic_string
'mf'
```

Set string.

Methods

`HairpinSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`HairpinSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HairpinSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HairpinSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HairpinSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`HairpinSpanner.__hash__()` \leq \Rightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`HairpinSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HairpinSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`HairpinSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HairpinSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

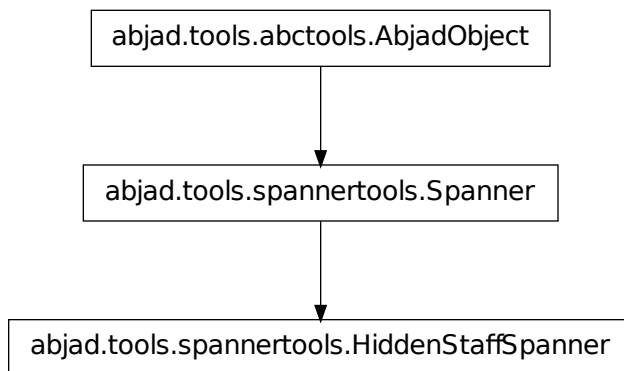
`HairpinSpanner.__setattr__()`
`x.__setattr__('name', value) \leq \Rightarrow x.name = value`

Note: Inherited from `__builtin__.object`

HairpinSpanner.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

spannertools.HiddenStaffSpanner



class `abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner` (*components*)
 Abjad hidden staff spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.HiddenStaffSpanner(staff[:2])
HiddenStaffSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
  \stopStaff
  c'8
  d'8
  \startStaff
  e'8
  f'8
}
```

Hide staff behind leaves in spanner.

Return hidden staff spanner.

Read-only Properties

`HiddenStaffSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

HiddenStaffSpanner.**duration_in_seconds**

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

HiddenStaffSpanner.**leaves**

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

HiddenStaffSpanner.**offset**

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(e'8, f'8)

```

```

abjad> spanner._offset.start
Offset(1, 4)

```

```

abjad> spanner._offset.stop
Offset(1, 2)

```

Return duration.

Note: Inherited from `spannertools.Spanner`

HiddenStaffSpanner.**override**

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

`HiddenStaffSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`HiddenStaffSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HiddenStaffSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`HiddenStaffSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`HiddenStaffSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HiddenStaffSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

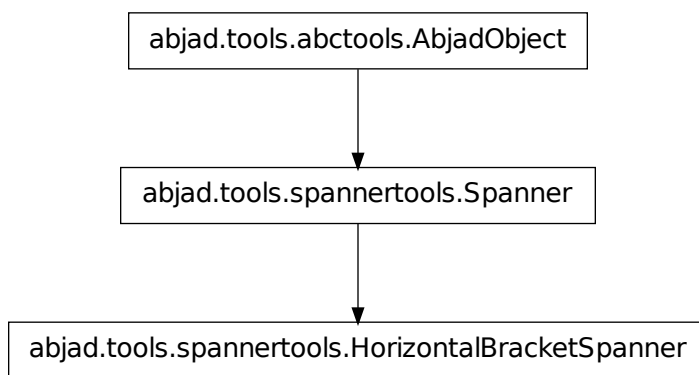
`HiddenStaffSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`HiddenStaffSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`spannertools.HorizontalBracketSpanner`



class `abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner`. **HorizontalBr**
 New in version 2.4. Abjad horizontal bracket spanner:

```

abjad> voice = Voice("c'4 d'4 e'4 f'4")
abjad> voice.engraver_consists.add('Horizontal_bracket_engraver')
```

```
abjad> horizontal_bracket_spanner = spannertools.HorizontalBracketSpanner(voice[:])

abjad> horizontal_bracket_spanner
HorizontalBracketSpanner(c'4, d'4, e'4, f'4)

abjad> f(voice)
\new Voice \with {
  \consists Horizontal_bracket_engraver
} {
  c'4 \startGroup
  d'4
  e'4
  f'4 \stopGroup
}
```

Return horizontal bracket spanner.

Read-only Properties

HorizontalBracketSpanner.components

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

HorizontalBracketSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner._offset.start
Offset(1, 4)

abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.override`
LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.preprolated_duration`
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.prolated_duration`
Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.set`
LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.written_duration`
Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

`HorizontalBracketSpanner.append(component)`
Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.append_left(component)`
Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.clear()`
Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.extend(components)`
Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
```

```

        f'8 ]
    }

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}

```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

```

```

abjad> spanner.index(voice[-2])
0

```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.pop()`

Remove and return rightmost component in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

```

```

abjad> spanner.pop()
Note("f'8")

```

```

abjad> spanner
Spanner(c'8, d'8, e'8)

```

Return component.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`HorizontalBracketSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`HorizontalBracketSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`HorizontalBracketSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`HorizontalBracketSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`HorizontalBracketSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

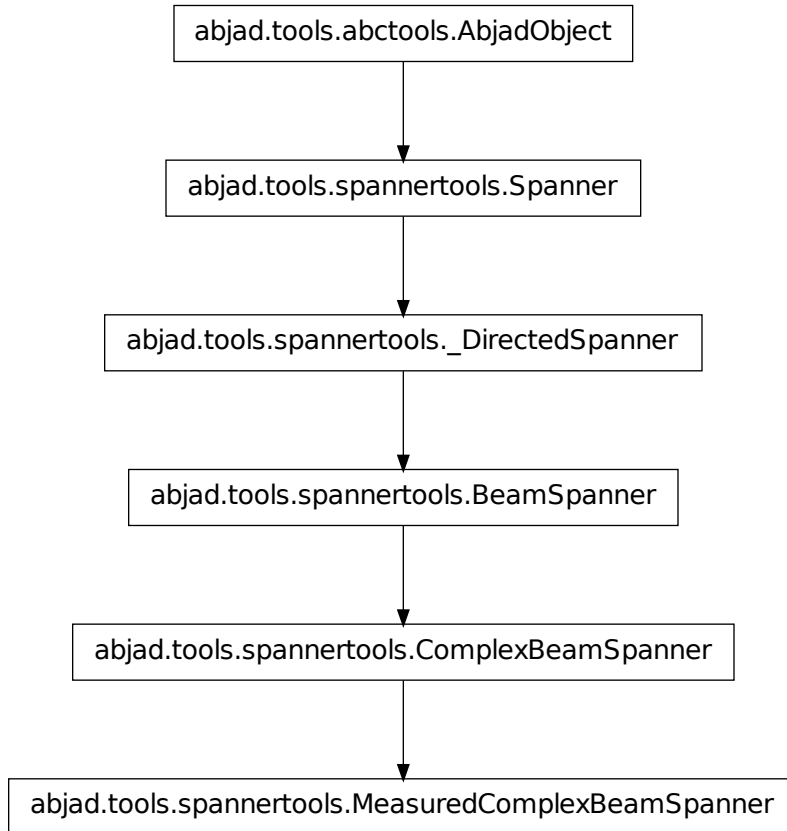
`HorizontalBracketSpanner.__setattr__()`
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`HorizontalBracketSpanner.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

spannertools.MeasuredComplexBeamSpanner



class `abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner`.**MeasuredComplexBeamSpanner**

Abjad measured complex beam spanner:

```
abjad> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
```

```
abjad> spannertools.MeasuredComplexBeamSpanner(staff.leaves)
MeasuredComplexBeamSpanner(c'16, d'16, e'16, f'16)
```

```
abjad> f(staff)
\new Staff {
  {
    \time 2/16
    \set stemLeftBeamCount = #0
    \set stemRightBeamCount = #2
```

```

        c'16 [
        \set stemLeftBeamCount = #2
        \set stemRightBeamCount = #1
        d'16
    ]
}
{
    \time 2/16
    \set stemLeftBeamCount = #1
    \set stemRightBeamCount = #2
    e'16
    \set stemLeftBeamCount = #2
    \set stemRightBeamCount = #0
    f'16 ]
}
}

```

Beam leaves in spanner explicitly.

Group leaves by measures.

Format top-level *span* beam between measures.

Return measured complex beam spanner.

Read-only Properties

`MeasuredComplexBeamSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements start and stop attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`MeasuredComplexBeamSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

`MeasuredComplexBeamSpanner.lone`

Beam lone leaf and force beam nibs to left:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'left')

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #0
c'16 [ ]
```

Beam lone leaf and force beam nibs to right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'right')

abjad> f(note)
\set stemLeftBeamCount = #0
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and force beam nibs to both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = 'both')

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Beam lone leaf and accept LilyPond default nibs at both left and right:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = True)

abjad> f(note)
\set stemLeftBeamCount = #2
\set stemRightBeamCount = #2
c'16 [ ]
```

Do not beam lone leaf:

```
abjad> note = Note("c'16")

abjad> beam = spannertools.ComplexBeamSpanner([note], lone = False)

abjad> f(note)
c'16
```

Set to `'left'`, `'right'`, `'both'`, `true` or `false` as shown above.

Ignore this setting when spanner contains more than one leaf.

Note: Inherited from `spannertools.ComplexBeamSpanner`

`MeasuredComplexBeamSpanner`.**span**

Get top-level beam count:

```
abjad> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
abjad> beam = spannertools.MeasuredComplexBeamSpanner(staff.leaves)
abjad> beam.span
1
```

Set top-level beam count:

```
abjad> staff = Staff([Measure((2, 16), "c'16 d'16"), Measure((2, 16), "e'16 f'16")])
abjad> beam = spannertools.MeasuredComplexBeamSpanner(staff.leaves)
abjad> beam.span = 2
abjad> beam.span
2
```

Set nonnegative integer.

Methods

`MeasuredComplexBeamSpanner`.**append**(*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner`.**append_left**(*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner`.**clear**()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.pop()`
Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.pop_left()`
Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`MeasuredComplexBeamSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MeasuredComplexBeamSpanner.__eq__(arg)`

True when `id(self) equals id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__hash__()` $\leq \Rightarrow hash(x)$

Note: Inherited from `__builtin__.object`

`MeasuredComplexBeamSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`MeasuredComplexBeamSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MeasuredComplexBeamSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

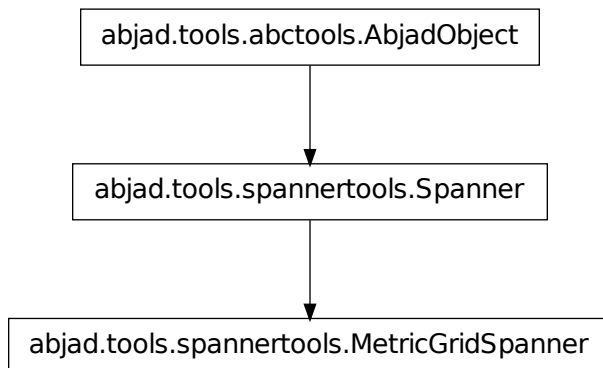
`MeasuredComplexBeamSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MeasuredComplexBeamSpanner.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

`spannertools.MetricGridSpanner`



class `abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner` (*components=*
me-
ters=None)

Abjad metric grid spanner:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
```

```
abjad> spannertools.MetricGridSpanner(staff.leaves, meters = [(1, 8), (1, 4)])
MetricGridSpanner(c'8, d'8, e'8, f'8, g'8, a'8, b'8, c'8)
```

```
abjad> f(staff)
\new Staff {
  \time 1/8
  c'8
  \time 1/4
  d'8
  e'8
  \time 1/8
  f'8
  \time 1/4
  g'8
  a'8
  \time 1/8
  b'8
  \time 1/4
  c'8
}
```

Format leaves in spanner cyclically with *meters*.

Return metric grid spanner.

Read-only Properties

MetricGridSpanner.components

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.written_duration

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

MetricGridSpanner.meters

Get metric grid meters:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
abjad> metric_grid_spanner = spannertools.MetricGridSpanner(staff.leaves, meters = [(1, 8), (1, 8)])
abjad> list(metric_grid_spanner.meters)
[(TimeSignatureMark((1, 8)), 0, False), (TimeSignatureMark((1, 4)), Duration(1, 8), False), (TimeSignatureMark((1, 4)), 0, False)]
```

Set metric grid meters:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c'8")
abjad> metric_grid_spanner = spannertools.MetricGridSpanner(staff.leaves, meters = [(1, 8), (1, 8)])
abjad> metric_grid_spanner.meters = [Duration(1, 4)]
abjad> list(metric_grid_spanner.meters)
[(TimeSignatureMark((1, 4)), 0, False), (TimeSignatureMark((1, 4)), Duration(1, 4), True), (TimeSignatureMark((1, 4)), 0, False)]
```

Set iterable.

Methods**MetricGridSpanner.append** (*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.append_left (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

MetricGridSpanner.clear ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
```

```
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.split_on_bar()`

Temporarily unavailable.

`MetricGridSpanner.splitting_condition(leaf)`

User-definable boolean function to determine whether leaf should be split:

```
abjad> voice = Voice("c'4 r4 c'4")
```

```
abjad> f(voice)
\new Voice {
  c'4
```



```
        r4
        c' 4
    }

abjad> def cond(leaf):
...     if not isinstance(leaf, Rest): return True
...     else: return False
abjad> metric_grid_spanner = spannertools.MetricGridSpanner(voice.leaves, [Duration(1, 8)])
abjad> metric_grid_spanner.splitting_condition = cond

abjad> metric_grid_spanner.split_on_bar()

abjad> f(voice)
\new Voice {
    \time 1/8
    c' 8 ~
    c' 8
    r4
    c' 8 ~
    c' 8
}
```

Function defaults to return true.

Special Methods

`MetricGridSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MetricGridSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`MetricGridSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`MetricGridSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MetricGridSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

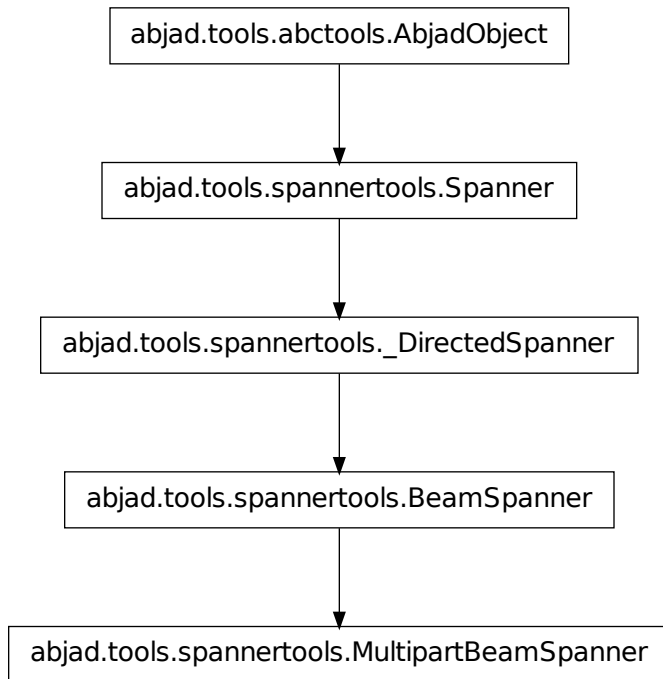
`MetricGridSpanner.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MetricGridSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.MultipartBeamSpanner

```
class abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner
```

New in version 2.0. Abjad multipart beam spanner:

```
abjad> staff = Staff("c'8 d'8 e'4 f'8 g'8 r4")

abjad> spannertools.MultipartBeamSpanner(staff[:])
MultipartBeamSpanner(c'8, d'8, e'4, f'8, g'8, r4)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8 ]
  e'4
  f'8 [
  g'8 ]
  r4
}
```

Avoid rests.

Avoid large-duration notes.

Return multipart beam spanner.

Read-only Properties

`MultipartBeamSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.override`
LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.preprolated_duration`
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.prolated_duration`
Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.set`
LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.written_duration`
Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`MultipartBeamSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

Methods

`MultipartBeamSpanner.append(component)`
Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
```

```
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```



```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`MultipartBeamSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`MultipartBeamSpanner.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__hash__()` <==> `hash(x)`

Note: Inherited from `__builtin__.object`

`MultipartBeamSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MultipartBeamSpanner.__repr__()`

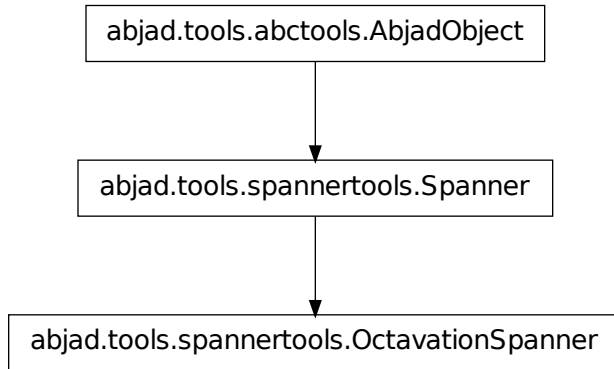
Note: Inherited from `spannertools.Spanner`

`MultipartBeamSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MultipartBeamSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.OctavationSpanner

class `abjad.tools.spannertools.OctavationSpanner.OctavationSpanner` (*components*, *start=0*, *stop=0*)

Abjad octavation spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spanner = spannertools.OctavationSpanner(staff[:], start = 1)

abjad> f(staff)
\new Staff {
  \ottava #1
  c'8
  d'8
  e'8
  f'8
  \ottava #0
}
  
```

Return octavation spanner.

Read-only Properties

`OctavationSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.duration_in_seconds
Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.leaves
Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.offset
New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.override
LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.preprolated_duration
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

OctavationSpanner.prolated_duration
Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`OctavationSpanner.start`

Get octavation start:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> octavation = spannertools.OctavationSpanner(staff[:], start = 1)
abjad> octavation.start
1
```

Set octavation start:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> octavation = spannertools.OctavationSpanner(staff[:], start = 1)
abjad> octavation.start
1
```

Set integer.

`OctavationSpanner.stop`

Get octavation stop:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> octavation = spannertools.OctavationSpanner(staff[:], start = 2, stop = 1)
abjad> octavation.stop
1
```

Set octavation stop:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> octavation = spannertools.OctavationSpanner(staff[:], start = 2, stop = 1)
abjad> octavation.stop = 0
abjad> octavation.stop
0
```

Set integer.

Methods

`OctavationSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
```

```
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
```

```

        c'8 [
        d'8 ]
        e'8 [
        f'8 ]
    }

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\nnew Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.pop()`

Remove and return rightmost component in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`OctavationSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OctavationSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OctavationSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OctavationSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`OctavationSpanner.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`OctavationSpanner.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OctavationSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OctavationSpanner.__repr__()`

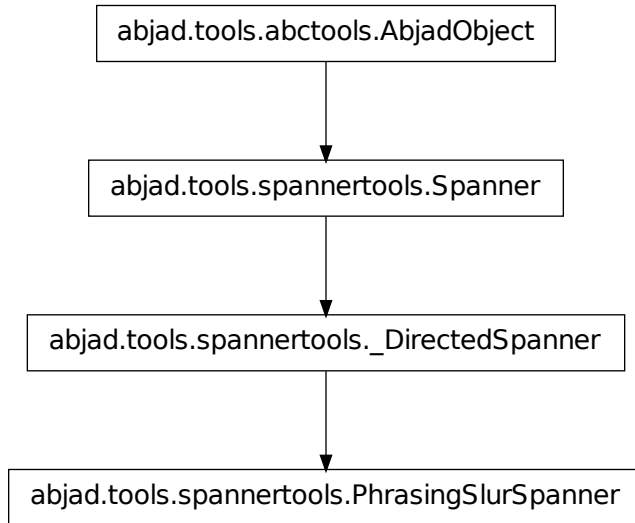
Note: Inherited from `spannertools.Spanner`

`OctavationSpanner.__setattr__()`
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`OctavationSpanner.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

spannertools.PhrasingSlurSpanner

class `abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner` **PhrasingSlurSpanner** (co

Abjad phrasing slur spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.PhrasingSlurSpanner(staff[:])
PhrasingSlurSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  c'8 \{
  d'8
  e'8
  f'8 \}
}

```

Return phrasing slur spanner.

Read-only Properties

`PhrasingSlurSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.duration_in_seconds`
Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.leaves`
Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.offset`
New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.override`
LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.preprolated_duration`
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.**prolated_duration**

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.**set**

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.**written_duration**

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

PhrasingSlurSpanner.**direction**

Note: Inherited from `spannertools._DirectedSpanner`

Methods

PhrasingSlurSpanner.**append**(*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.**append_left**(*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

PhrasingSlurSpanner.__contains__(*expr*)

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.__delattr__()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

PhrasingSlurSpanner.__eq__(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

PhrasingSlurSpanner.__ge__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PhrasingSlurSpanner.__getitem__(*expr*)

Note: Inherited from `spannertools.Spanner`

PhrasingSlurSpanner.__gt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

PhrasingSlurSpanner.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

PhrasingSlurSpanner.__le__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PhrasingSlurSpanner.__len__()

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`PhrasingSlurSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`PhrasingSlurSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

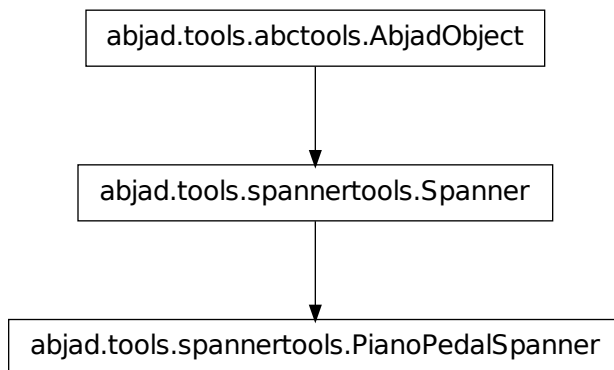
`PhrasingSlurSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`PhrasingSlurSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`spannertools.PianoPedalSpanner`



class `abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner` (*components*)
 Abjad piano pedal spanner:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.PianoPedalSpanner(staff[:])
PianoPedalSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
  \set Staff.pedalSustainStyle = #'mixed
  c'8 \sustainOn
  d'8
  e'8
  f'8 \sustainOff
}
```

Return piano pedal spanner.

Read-only Properties

`PianoPedalSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements start and stop attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`PianoPedalSpanner.kind`

Get piano pedal spanner kind:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.PianoPedalSpanner(staff[:])
abjad> spanner.kind
'sustain'
```

Set piano pedal spanner kind:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.PianoPedalSpanner(staff[:])
abjad> spanner.kind = 'sostenuto'
abjad> spanner.kind
'sostenuto'
```

Acceptable values 'sustain', 'sostenuto', 'corda'.

PianoPedalSpanner.style

Get piano pedal spanner style:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.PianoPedalSpanner(staff[:])
abjad> spanner.style
'mixed'
```

Set piano pedal spanner style:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.PianoPedalSpanner(staff[:])
abjad> spanner.style = 'bracket'
abjad> spanner.style
'bracket'
```

Acceptable values 'mixed', 'bracket', 'text'.

Methods

PianoPedalSpanner.append (*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from [spannertools.Spanner](#)

PianoPedalSpanner.append_left (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`PianoPedalSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

PianoPedalSpanner.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

PianoPedalSpanner.__eq__(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

PianoPedalSpanner.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

PianoPedalSpanner.__getitem__(expr)

Note: Inherited from `spannertools.Spanner`

PianoPedalSpanner.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

PianoPedalSpanner.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

PianoPedalSpanner.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

PianoPedalSpanner.__len__()

Note: Inherited from `spannertools.Spanner`

PianoPedalSpanner.__lt__(other)
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`PianoPedalSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`PianoPedalSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

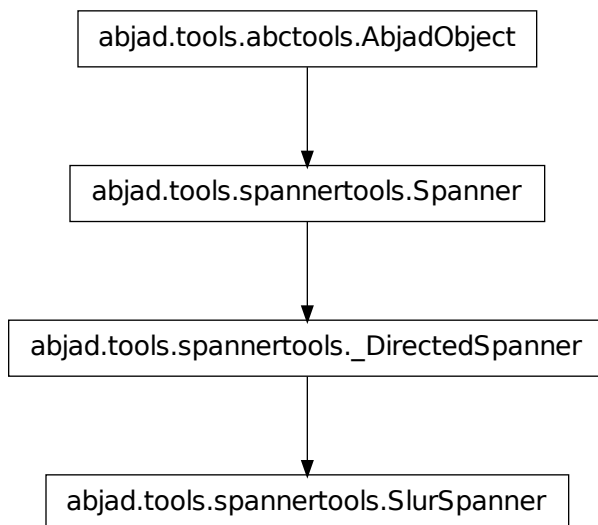
`PianoPedalSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`PianoPedalSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.SlurSpanner



class `abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner` (*components=None, direction=None*)

Abjad slur spanner:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.SlurSpanner(staff[:])
SlurSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 (
    d'8
    e'8
    f'8 )
}
```

Return slur spanner.

Read-only Properties

`SlurSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

SlurSpanner.written_duration

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties**SlurSpanner.direction**

Note: Inherited from `spannertools._DirectedSpanner`

Methods

`SlurSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`SlurSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SlurSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SlurSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SlurSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SlurSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SlurSpanner.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SlurSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`SlurSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`SlurSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

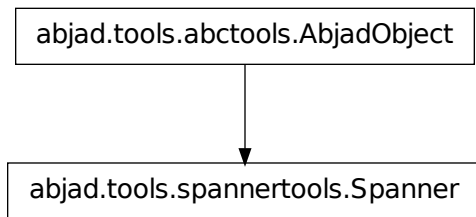
`SlurSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SlurSpanner.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`spannertools.Spanner`



class `abjad.tools.spannertools.Spanner.Spanner` (*components=None*)

Any type of notation object that stretches horizontally and encompasses some number of notes, rest, chords, tuplets, measures, voices or other Abjad components.

Beams, slurs, hairpins, trills, glissandi and piano pedal brackets all stretch horizontally on the page to encompass multiple notes and all implement as Abjad spanners. That is, these spanner all have an obvious graphic reality with definite start-, stop- and midpoints.

Abjad also implements a number of spanners of a different type, such as tempo and instrument spanners, which mark a group of notes, rests, chords or measures as carrying a certain tempo or being played by a certain instrument.

The spanner class described here abstracts the functionality that all such spanners, both graphic and nongraphics, share. This shared functionality includes methods to add, remove, inspect and test components governed by the spanner, as well as basic formatting properties. The other spanner classes, such as beam and glissando, all inherit from this class and receive the functionality implemented here.

Read-only Properties

`Spanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

`Spanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

`Spanner.leaves`

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use `spanner-` specific iteration tools.

Spanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(e'8, f'8)

```

```

abjad> spanner._offset.start
Offset(1, 4)

```

```

abjad> spanner._offset.stop
Offset(1, 2)

```

Return duration.

Spanner.override

LilyPond grob override component plug-in.

Spanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Spanner.prolated_duration

Sum of prolated duration of all components in spanner.

Spanner.set

LilyPond context setting component plug-in.

Spanner.written_duration

Sum of written duration of all components in spanner.

Methods

Spanner.append(*component*)

Add *component* to right of spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

```

```

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)

```

Return `none`.

Spanner.append_left (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Spanner.clear ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Spanner.extend (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Spanner.extend_left (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Spanner.fracture (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

`Spanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

`Spanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

`Spanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

`Spanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Special Methods

`Spanner.__contains__(expr)`

`Spanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Spanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Spanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Spanner.__getitem__(expr)`

`Spanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Spanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Spanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Spanner.__len__()`

`Spanner.__lt__(other)`

Trivial comparison to allow doctests to work.

`Spanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Spanner.__repr__()`

`Spanner.__setattr__()`

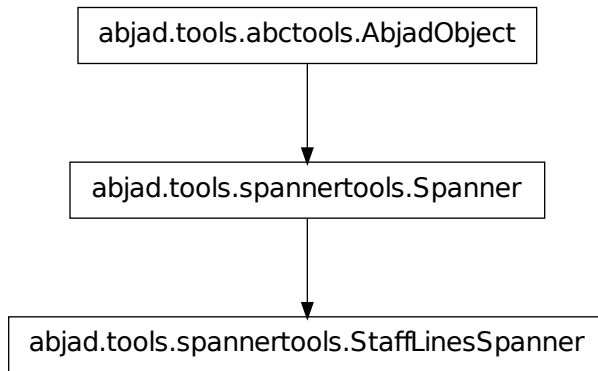
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Spanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.StaffLinesSpanner



class `abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner` (*components*, *arg=5*)

Abjad staff lines spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.StaffLinesSpanner(staff[:2], 1)
StaffLinesSpanner(c'8, d'8)

abjad> f(staff)
\new Staff {
  \stopStaff
  \override Staff.StaffSymbol #'line-count = #1
  \startStaff
  c'8
  d'8
  \stopStaff
  \revert Staff.StaffSymbol #'line-count
  \startStaff
  e'8
  f'8
}
  
```

Staff lines spanner handles changing either the line-count or the line-positions property of the StaffSymbol grob, as well as automatically stopping and restarting the staff so that the change may take place.

Return staff lines spanner.

Read-only Properties

`StaffLinesSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
  
```



```
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

StaffLinesSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

StaffLinesSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

StaffLinesSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

StaffLinesSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.preprolated_duration`
Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.prolated_duration`
Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.set`
LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.written_duration`
Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`StaffLinesSpanner.lines`

Get staff lines spanner line count:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.StaffLinesSpanner(staff[:2], 1)
abjad> spanner.lines
1
```

Set staff lines spanner line count:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.StaffLinesSpanner(staff[:2], 1)
abjad> spanner.lines = 2
abjad> spanner.lines
2
```

Set integer.

Methods

`StaffLinesSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.append_left` (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.clear` ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.extend` (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}
```

```
abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`StaffLinesSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`StaffLinesSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`StaffLinesSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`StaffLinesSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`StaffLinesSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

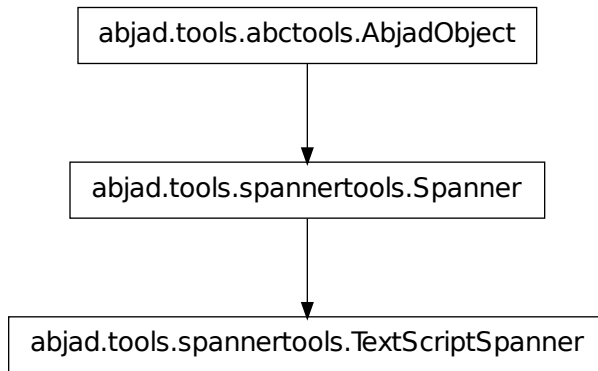
`StaffLinesSpanner.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`StaffLinesSpanner.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

spannertools.TextScriptSpanner



class `abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner` (*components*)
 New in version 2.0. Abjad text script spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spanner = spannertools.TextScriptSpanner(staff[:])
abjad> spanner.override.text_script.color = 'red'
abjad> markuptools.Markup(r'\italic { espressivo }', 'up')(staff[1])
Markup('\italic { espressivo }', direction='^')(d'8)

abjad> f(staff)
\new Staff {
  \override TextScript #'color = #red
  c'8
  d'8 ^ \markup { \italic { espressivo } }
  e'8
  f'8
  \revert TextScript #'color
}
  
```

Override LilyPond TextScript grob.

Return text script spanner.

Read-only Properties

`TextScriptSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
  
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.leaves`

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.offset`

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.override`

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.preprolated_duration`

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

`TextScriptSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)
```

```
abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\\new Voice {
    c'8 [ ]
    d'8 [
    e'8
    f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]

abjad> print voice.format
\\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop()
Note("f'8")
```

```
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`TextScriptSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TextScriptSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`TextScriptSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`TextScriptSpanner.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TextScriptSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

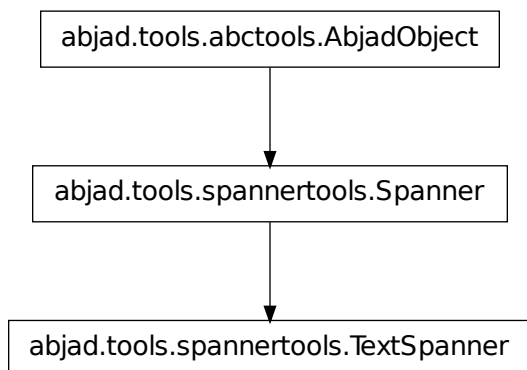
`TextScriptSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TextScriptSpanner.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.TextSpanner



class `abjad.tools.spannertools.TextSpanner.TextSpanner(components=None)`
 New in version 2.0. Abjad text spanner:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> text_spanner = spannertools.TextSpanner(staff[:])

```

```

abjad> markup = markuptools.Markup(' (markup #:bold #:italic "foo")', style_string = 'scheme')
abjad> text_spanner.override.text_spanner.bound_details__left__text = markup

```

```

abjad> markup = markuptools.Markup("(markup #:draw-line '(0 . -1))", style_string = 'scheme')
abjad> text_spanner.override.text_spanner.bound_details__right__text = markup
abjad> text_spanner.override.text_spanner.dash_fraction = 1

abjad> f(staff)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = #(markup #:bold #:italic "foo")
  \override TextSpanner #'bound-details #'right #'text = #(markup #:draw-line '(0 . -1))
  \override TextSpanner #'dash-fraction = #1
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'dash-fraction
}

```

Override LilyPond TextSpanner grob.

Return text spanner.

Read-only Properties

`TextSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.duration_in_seconds`

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.leaves`

Return read-only tuple of leaves in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

TextSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

TextSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

TextSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

TextSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

TextSpanner.set

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

TextSpanner.written_duration

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Methods

TextSpanner.append (*component*)

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

TextSpanner.**append_left** (*component*)

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

TextSpanner.**clear** ()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

TextSpanner.**extend** (*components*)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.extend_left` (*components*)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8 ]
    e'8 [
    f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from [spannertools.Spanner](#)

`TextSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from [spannertools.Spanner](#)

`TextSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.pop_left()
Note("c'8")
```

```
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`TextSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`TextSpanner.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TextSpanner.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TextSpanner.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TextSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`TextSpanner.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TextSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TextSpanner.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TextSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`TextSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`TextSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TextSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

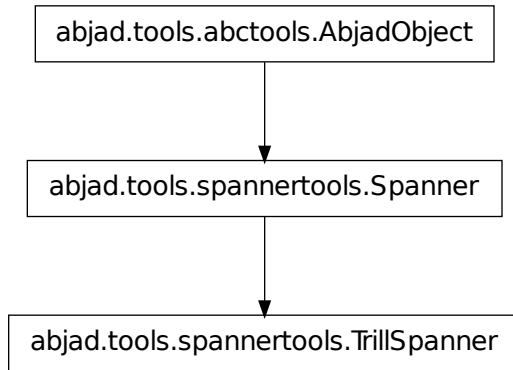
`TextSpanner.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TextSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools.TrillSpanner

class `abjad.tools.spannertools.TrillSpanner.TrillSpanner` (*components=None*)
Abjad trill spanner:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> spannertools.TrillSpanner(staff[:])
TrillSpanner(c'8, d'8, e'8, f'8)

abjad> f(staff)
\new Staff {
    c'8 \startTrillSpan
    d'8
    e'8
    f'8 \stopTrillSpan
}
```

Override LilyPond TrillSpanner grob.

Return trill spanner.

Read-only Properties

`TrillSpanner.components`

Return read-only tuple of components in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.prolated_duration

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`TrillSpanner.pitch`

Optional read / write pitch for pitched trills.

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> trill = spannertools.TrillSpanner(t[:2])
abjad> trill.pitch = pitchtools.NamedChromaticPitch('cs', 4)

abjad> f(t)
\new Staff {
  \pitchedTrill c'8 \startTrillSpan cs'
  d'8 \stopTrillSpan
  e'8
  f'8
}
```

Set pitch.

`TrillSpanner.written_pitch`

Methods

`TrillSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.clear()

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.extend(components)

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)

abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

TrillSpanner.extend_left(components)

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.fracture` (*i*, *direction*='both')

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.fuse` (*spanner*)

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.index` (*component*)

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.pop` ()

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.pop_left` ()

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`TrillSpanner.__contains__` (*expr*)

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.__delattr__` ()
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TrillSpanner.__eq__` (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TrillSpanner.__ge__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TrillSpanner.__getitem__` (*expr*)

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.__gt__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`TrillSpanner.__hash__` () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

`TrillSpanner.__le__` (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TrillSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.__lt__(other)`
Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TrillSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

`TrillSpanner.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TrillSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

spannertools functions

spannertools.all_are_spanners

`abjad.tools.spannertools.all_are_spanners.all_are_spanners(expr)`
New in version 2.6. True when *expr* is a sequence of Abjad spanners:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.BeamSpanner(voice[:2])

abjad> spannertools.all_are_spanners([spanner])
True
```

True when *expr* is an empty sequence:

```
abjad> spannertools.all_are_spanners([])
True
```

Otherwise false:

```
abjad> spannertools.all_are_spanners('foo')
False
```

Return boolean.

spannertools.destroy_all_spanners_attached_to_component

`abjad.tools.spannertools.destroy_all_spanners_attached_to_component`.**destroy_all_spanners_attached_to_component**

New in version 1.1. Destroy all spanners attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

abjad> spannertools.destroy_all_spanners_attached_to_component(staff[0])
abjad> f(staff)
\new Staff {
    c'8 \startTrillSpan
    d'8
    e'8
    f'8 \stopTrillSpan
}
```

Return none.

spannertools.find_index_of_spanner_component_at_score_offset

`abjad.tools.spannertools.find_index_of_spanner_component_at_score_offset`.**find_index_of_spanner_component_at_score_offset**

Return index of component in ‘spanner’ that begins at exactly ‘score_offset’:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)

abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
abjad> spannertools.find_index_of_spanner_component_at_score_offset (beam, Duration(3, 8))
3
```

Raise *spanner* population error when no component in *spanner* begins at exactly *score_offset*.
 Changed in version 2.0: renamed `spannertools.find_index_at_score_offset()` to `spannertools.find_index_of_spanner_component_at_score_offset()`.

`spannertools.find_spanner_component_starting_at_exactly_score_offset`

`abjad.tools.spannertools.find_spanner_component_starting_at_exactly_score_offset`. **`find_spanner`**

Find *spanner* component starting at exactly *score_offset*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
```

```
abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

```
abjad> spannertools.find_spanner_component_starting_at_exactly_score_offset (beam, Duration(3, 8))
Note("f'8")
```

When no *spanner* component starts at exactly *score_offset* return none.

Return *spanner* component or none. Changed in version 2.0: re-named `spannertools.find_component_at_score_offset()` to `spannertools.find_spanner_component_starting_at_exactly_score_offset()`.

`spannertools.fracture_all_spanners_attached_to_component`

`abjad.tools.spannertools.fracture_all_spanners_attached_to_component`. **`fracture_all_spanners`**

New in version 1.1. Fracture all spanners attached to *component* according to *direction*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8
  e'8
  f'8 ] ) \stopTrillSpan
}

abjad> spannertools.fracture_all_spanners_attached_to_component (staff[1], 'right')
[(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8)), (SlurSpanner(c
```



```

abjad> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8 ] )
  e'8 [ (
  f'8 ] ) \stopTrillSpan
}

```

Set *direction* to left, right or both.

spannertools.fracture_spanners_that_cross_components

`abjad.tools.spannertools.fracture_spanners_that_cross_components.fracture_spanners_that_cross_components`

Fracture to the left of the leftmost component. Fracture to the right of the rightmost component. Do not fracture spanners of any components at higher levels of score. Do not fracture spanners of any components at lower levels of score. Return components.

Components must be thread-contiguous. Some spanners may copy during fracture. This helper is public-safe.

Example:

```

t = Staff(Container(notetools.make_repeated_notes(2)) * 3)
pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
spannertools.CrescendoSpanner(t)
spannertools.BeamSpanner(t[:])
spannertools.TrillSpanner(t.leaves)

```

```

\new Staff {
  {
    c'8 [ \< \startTrillSpan
    d'8
  }
  {
    e'8
    f'8
  }
  {
    g'8
    a'8 ] \! \stopTrillSpan
  }
}

```

```
spannertools.fracture_spanners_that_cross_components(t[1:2])
```

```

\new Staff {
  {
    c'8 [ \< \startTrillSpan
    d'8 ]
  }
  {
    e'8 [
    f'8 ]
  }
  {
    g'8 [
    a'8 ] \! \stopTrillSpan
  }
}

```

Changed in version 2.0: renamed `spannertools.fracture_crossing()` to `spannertools.fracture_spanners_that_cross_components()`.

`spannertools.get_beam_spanner_attached_to_component`

`abjad.tools.spannertools.get_beam_spanner_attached_to_component.get_beam_spanner_attached_to_component`

New in version 2.0. Get the only beam spanner attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)

abjad> f(staff)
\new Staff {
  c'8 [
  d'8
  e'8
  f'8 ]
}

abjad> spannertools.get_beam_spanner_attached_to_component(staff[0])
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> _ is beam
True
```

Return beam spanner.

Raise missing spanner error when no beam spanner attached to *component*.

Raise extra spanner error when more than one beam spanner attached to *component*. Changed in version 2.0: renamed `beamtools.get_beam_spanner()` to `spannertools.get_beam_spanner_attached_to_component()`. Changed in version 2.0: renamed `beamtools.get_beam_spanner_attached_to_component()` to `spannertools.get_beam_spanner_attached_to_component()`.

`spannertools.get_nth_leaf_in_spanner`

`abjad.tools.spannertools.get_nth_leaf_in_spanner.get_nth_leaf_in_spanner(spanner, idx)`

Get *nth* leaf in spanner, no matter how complicated the nesting situation. Changed in version 2.0: renamed `spannertools.get_nth_leaf()` to `spannertools.get_nth_leaf_in_spanner()`.

`spannertools.get_spanners_attached_to_any_improper_child_of_component`

`abjad.tools.spannertools.get_spanners_attached_to_any_improper_child_of_component.get_spanners_attached_to_any_improper_child_of_component`

New in version 2.0. Get all spanners attached to any improper children of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
abjad> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
abjad> trill = spannertools.TrillSpanner(staff)
```

```

abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8 )
    e'8 (
    f'8 ] ) \stopTrillSpan
}

abjad> len(spannertools.get_spanners_attached_to_any_improper_child_of_component(staff)) == 4
True

```

Get all spanners of *klass* attached to any proper children of *component*:

```

abjad> spanner_klass = spannertools.SlurSpanner
abjad> spannertools.get_spanners_attached_to_any_proper_child_of_component(staff, spanner_klass)
set([SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)])

```

Get all spanners of any *klass* attached to any proper children of *component*:

```

abjad> spanner_klasses = (spannertools.SlurSpanner, spannertools.BeamSpanner)
abjad> spannertools.get_spanners_attached_to_any_proper_child_of_component(staff, spanner_klasses)
set([BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)])

```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_any_improper_children_of_component()` to `spannertools.get_spanners_attached_to_any_improper_child_of_component()`.

spannertools.get_spanners_attached_to_any_improper_parent_of_component

`abjad.tools.spannertools.get_spanners_attached_to_any_improper_parent_of_component.get_spanners_attached_to_any_improper_parent_of_component`

New in version 1.1. Get all spanners attached to improper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

abjad> spannertools.get_spanners_attached_to_any_improper_parent_of_component(staff[0]) # doctest
set([BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8, e'8, f'8), TrillSpanner({c'8, d'8, e'8, f'8})])

```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_improper_parentage_of_component()` to `spannertools.get_spanners_attached_to_any_improper_parent_of_component()`.

spannertools.get_spanners_attached_to_any_proper_child_of_component

`abjad.tools.spannertools.get_spanners_attached_to_any_proper_child_of_component.get_spanners_attached_to_any_proper_child_of_component`

New in version 2.0. Get all spanners attached to any proper children of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
abjad> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
abjad> trill = spannertools.TrillSpanner(staff)

abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8 )
    e'8 (
    f'8 ] ) \stopTrillSpan
}

abjad> len(spannertools.get_spanners_attached_to_any_proper_child_of_component(staff)) == 3
True

```

Get all spanners of *klass* attached to any proper children of *component*:

```

abjad> spanner_klass = spannertools.SlurSpanner
abjad> spannertools.get_spanners_attached_to_any_proper_child_of_component(staff, spanner_klass)
set([SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)])

```

Get all spanners of any *klass* attached to any proper children of *component*:

```

abjad> spanner_klasses = (spannertools.SlurSpanner, spannertools.BeamSpanner)
abjad> spannertools.get_spanners_attached_to_any_proper_child_of_component(staff, spanner_klasses)
set([BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8), SlurSpanner(e'8, f'8)])

```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_any_proper_children_of_component()` to `spannertools.get_spanners_attached_to_any_proper_child_of_component()`.

`spannertools.get_spanners_attached_to_any_proper_parent_of_component`

`abjad.tools.spannertools.get_spanners_attached_to_any_proper_parent_of_component.get_spanners_attached_to_any_proper_parent_of_component`

New in version 2.0. Get all spanners attached to any proper parent of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

abjad> spannertools.get_spanners_attached_to_any_proper_parent_of_component(staff[0])
set([TrillSpanner({c'8, d'8, e'8, f'8})])

```

Return unordered set of zero or more spanners. Changed in version 2.0: renamed `spannertools.get_all_spanners_attached_to_any_proper_parent_of_component()` to `spannertools.get_spanners_attached_to_any_proper_parent_of_component()`.

spannertools.get_spanners_attached_to_component

`abjad.tools.spannertools.get_spanners_attached_to_component.get_spanners_attached_to_compon`

New in version 2.0. Get all spanners attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> first_slur = spannertools.SlurSpanner(staff.leaves[:2])
abjad> second_slur = spannertools.SlurSpanner(staff.leaves[2:])
abjad> crescendo = spannertools.CrescendoSpanner(staff.leaves)
```

```
abjad> f(staff)
\new Staff {
    c'8 [ \< (
    d'8 )
    e'8 (
    f'8 ] \! )
}
```

```
abjad> spannertools.get_spanners_attached_to_component(staff.leaves[0]) # doctest: +SKIP
set([BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8), CrescendoSpanner(c'8, d'8, e'8, f'8)])
```

Get spanners of *klass* attached to *component*:

```
abjad> klass = spannertools.BeamSpanner
abjad> spannertools.get_spanners_attached_to_component(staff.leaves[0], klass) # doctest: +SKIP
set([BeamSpanner(c'8, d'8, e'8, f'8)])
```

Get spanners of any *klass* attached to *component*:

```
abjad> classes = (spannertools.BeamSpanner, spannertools.SlurSpanner)
abjad> spannertools.get_spanners_attached_to_component(staff.leaves[0], classes) # doctest: +SKIP
set([BeamSpanner(c'8, d'8, e'8, f'8), SlurSpanner(c'8, d'8)])
```

Return unordered set of zero or more spanners. Changed in version 2.0: re-named `spannertools.get_all_spanners_attached_to_component()` to `spannertools.get_spanners_attached_to_component()`.

spannertools.get_spanners_contained_by_components

`abjad.tools.spannertools.get_spanners_contained_by_components.get_spanners_contained_by_compon`

Return unordered set of spanners contained within any component in list of thread-contiguous components. Getter for `t.spanners.contained` across thread-contiguous components. Changed in version 2.0: renamed `spannertools.get_contained()` to `spannertools.get_spanners_contained_by_components()`.

spannertools.get_spanners_covered_by_components

`abjad.tools.spannertools.get_spanners_covered_by_components.get_spanners_covered_by_compon`

Return unordered set of spanners completely contained within the time bounds of thread-contiguous components.

Compare ‘covered’ spanners with ‘contained’ spanners. Compare ‘covered’ spanners with ‘dominant’ spanners.

Changed in version 2.0: renamed `spannertools.get_covered()` to `spannertools.get_spanners_covered_by_components()`.

`spannertools.get_spanners_on_components_or_component_children`

`abjad.tools.spannertools.get_spanners_on_components_or_component_children.get_spanners_on_components_or_component_children`
 Return unordered set of all spanners attaching to any component in *components* or attaching to any of the children of any of the components in *components*. Changed in version 2.0: renamed `spannertools.get_attached()` to `spannertools.get_spanners_on_components_or_component_children()`.

`spannertools.get_spanners_that_cross_components`

`abjad.tools.spannertools.get_spanners_that_cross_components.get_spanners_that_cross_components`
 Assert thread-contiguous components. Collect spanners that attach to any component in ‘components’. Return unordered set of crossing spanners. A spanner P crosses a list of thread-contiguous components C when P and C share at least one component and when it is the case that NOT ALL of the components in P are also in C. In other words, there is some intersection – but not total intersection – between the components of P and C.
 Compare ‘crossing’ spanners with ‘covered’ spanners. Compare ‘crossing’ spanners with ‘dominant’ spanners. Compare ‘crossing’ spanners with ‘contained’ spanners. Compare ‘crossing’ spanners with ‘attached’ spanners. Changed in version 2.0: renamed `spannertools.get_crossing()` to `spannertools.get_spanners_that_cross_components()`.

`spannertools.get_spanners_that_dominate_component_pair`

`abjad.tools.spannertools.get_spanners_that_dominate_component_pair.get_spanners_that_dominate_component_pair`
 Return Python list of (spanner, index) pairs. ‘left’ must be either an Abjad component or None. ‘right’ must be either an Abjad component or None.
 If both ‘left’ and ‘right’ are components, then ‘left’ and ‘right’ must be thread-contiguous.
 This is a special version of `spannertools.get_spanners_that_dominate_components()`. This version is useful for finding spanners that dominate a zero-length ‘crack’ between components, as in `t[2:2]`. Changed in version 2.0: renamed `spannertools.get_dominant_between()` to `spannertools.get_spanners_that_dominate_component_pair()`.

`spannertools.get_spanners_that_dominate_components`

`abjad.tools.spannertools.get_spanners_that_dominate_components.get_spanners_that_dominate_components`
 Return Python list of (spanner, index) pairs. Each (spanner, index) pair gives a spanner which dominates all components in ‘components’ together with the start-index at which spanner first encounters ‘components’.
 Use this helper to ‘lift’ any and all spanners temporarily from ‘components’, perform some action to the underlying score tree, and then reattach all spanners to new score components.
 This operation always leaves all expressions in tact. Changed in version 2.0: renamed `spannertools.get_dominant()` to `spannertools.get_spanners_that_dominate_components()`.

spannertools.get_spanners_that_dominate_container_components_from_to

`abjad.tools.spannertools.get_spanners_that_dominate_container_components_from_to.get_spanners_that_dominate_container_components_from_to`

Return Python list of (spanner, index) pairs. Each spanner dominates the components specified by slice with start index 'start' and stop index 'stop'. Generalization of dominant spanner-finding functions for slices. This exists for slices like `t[2:2]` that are empty lists. Changed in version 2.0: renamed `spannertools.get_dominant_slice()` to `spannertools.get_spanners_that_dominate_container_components_from_to()`.

spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component

`abjad.tools.spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component`

New in version 1.1. Get the only spanner attached to any improper parent *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
    f'8 ] ) \stopTrillSpan
}

abjad> print spannertools.get_the_only_spanner_attached_to_component(staff)
TrillSpanner({c'8, d'8, e'8, f'8})
```

Raise missing spanner error when no spanner attached to *component*.

Raise extra spanner error when more than one spanner attached to *component*.

Return a single spanner.

Note: function will usually be called with *klass* specifier set.

spannertools.get_the_only_spanner_attached_to_component

`abjad.tools.spannertools.get_the_only_spanner_attached_to_component.get_the_only_spanner_attached_to_component`

New in version 1.1. Get the only spanner attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
    c'8 [ ( \startTrillSpan
    d'8
    e'8
```

```
f'8 ] ) \stopTrillSpan
}
```

```
abjad> print spannertools.get_the_only_spanner_attached_to_component(staff)
TrillSpanner({c'8, d'8, e'8, f'8})
```

Raise missing spanner error when no spanner attached to *component*.

Raise extra spanner error when more than one spanner attached to *component*.

Return a single spanner.

Note: function will usually be called with *klass* specifier set.

spannertools.is_component_with_beam_spanner_attached

`abjad.tools.spannertools.is_component_with_beam_spanner_attached.is_component_with_beam_sp`

New in version 2.0. True when *expr* is component with beam spanner attached:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
```

```
abjad> spannertools.is_component_with_beam_spanner_attached(staff[0])
True
```

Otherwise false:

```
abjad> note = Note("c'8")
```

```
abjad> spannertools.is_component_with_beam_spanner_attached(note)
False
```

Return boolean. Changed in version 2.0: renamed `beamtools.is_component_with_beam_spanner_attached()` to `spannertools.is_component_with_beam_spanner_attached()`.

spannertools.is_component_with_spanner_attached

`abjad.tools.spannertools.is_component_with_spanner_attached.is_component_with_spanner_atta`

New in version 2.0. True when *expr* is a component with spanner attached:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> f(staff)
\new Staff {
    c'8 [
    d'8
    e'8
    f'8 ]
}
```

```
abjad> spannertools.is_component_with_spanner_attached(staff[0])
True
```

Otherwise false:


```
abjad> spannertools.is_component_with_spanner_attached(staff)
False
```

When *klass* is not none then true when *expr* is a component with a spanner of *klass* attached.

Return true or false.

spannertools.iterate_components_backward_in_spanner

```
abjad.tools.spannertools.iterate_components_backward_in_spanner.iterate_components_backward_in_spanner
```

New in version 2.0. Yield components in *spanner* one at a time from left to right.

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> p = spannertools.BeamSpanner(t[2:])
abjad> notes = spannertools.iterate_components_backward_in_spanner(p, klass = Note)
abjad> for note in notes:
...     note
Note("f'8")
Note("e'8")
```

Changed in version 2.0: renamed `spannertools.iterate_components_backward()` to `spannertools.iterate_components_backward_in_spanner()`.

spannertools.iterate_components_forward_in_spanner

```
abjad.tools.spannertools.iterate_components_forward_in_spanner.iterate_components_forward_in_spanner
```

New in version 2.0. Yield components in *spanner* one at a time from left to right.

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> p = spannertools.BeamSpanner(t[2:])
abjad> notes = spannertools.iterate_components_forward_in_spanner(p, klass = Note)
abjad> for note in notes:
...     note
Note("e'8")
Note("f'8")
```

Changed in version 2.0: renamed `spannertools.iterate_components_forward()` to `spannertools.iterate_components_forward_in_spanner()`.

spannertools.make_covered_spanner_schema

```
abjad.tools.spannertools.make_covered_spanner_schema.make_covered_spanner_schema(components)
New in version 2.0. Make schema of spanners covered by components:
```

```
abjad> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> slur = spannertools.SlurSpanner(voice[-2:])
```

```

abjad> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    \time 2/8
    e'8
    f'8 ]
  }
  {
    \time 2/8
    g'8 (
    a'8
  )
  {
    \time 2/8
    b'8
    c''8 )
  }
}

abjad> spannertools.make_covered_spanner_schema([voice]) # doctest: +SKIP
{BeamSpanner(c'8, d'8, e'8, f'8): [2, 3, 5, 6], SlurSpanner(|2/8(2)|, |2/8(2)|): [7, 10]}

```

Return dictionary.

spannertools.make_dynamic_spanner_below_with_nib_at_right

`abjad.tools.spannertools.make_dynamic_spanner_below_with_nib_at_right.make_dynamic_spanner`

New in version 2.0. Span *components* with text spanner. Position spanner below staff and configure with *dynamic_text*, solid line and upward-pointing nib at right.

```

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.make_dynamic_spanner_below_with_nib_at_right('mp', t[:])
TextSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { \dynamic { mp } }
  \override TextSpanner #'bound-details #'right #'text = #(markup #:draw-line '(0 . 1))
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'direction = #down
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'bound-details #'right-broken #'text
  \revert TextSpanner #'dash-fraction
}

```

```
\revert TextSpanner #'direction
}
```

Changed in version 2.0: renamed `spanners.dynamic_spanner_below_with_nib_at_right()` to `spannertools.make_dynamic_spanner_below_with_nib_at_right()`.

spannertools.make_solid_text_spanner_above_with_nib_at_right

`abjad.tools.spannertools.make_solid_text_spanner_above_with_nib_at_right.make_solid_text_sp`

New in version 2.0. Span *components* with text spanner. Position spanner above staff and configure with *left_text*, solid line and downward-pointing nib at right.

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.make_solid_text_spanner_above_with_nib_at_right('foo', t[:])
TextSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { foo }
  \override TextSpanner #'bound-details #'right #'text = #(markup #:draw-line '(0 . -1))
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
  \override TextSpanner #'dash-fraction = #1
  \override TextSpanner #'direction = #up
  c'8 \startTextSpan
  d'8
  e'8
  f'8 \stopTextSpan
  \revert TextSpanner #'bound-details #'left #'text
  \revert TextSpanner #'bound-details #'right #'text
  \revert TextSpanner #'bound-details #'right-broken #'text
  \revert TextSpanner #'dash-fraction
  \revert TextSpanner #'direction
}
```

Changed in version 2.0: renamed `spanners.solid_text_spanner_above_with_nib_at_right()` to `spannertools.make_solid_text_spanner_above_with_nib_at_right()`.

spannertools.make_solid_text_spanner_below_with_nib_at_right

`abjad.tools.spannertools.make_solid_text_spanner_below_with_nib_at_right.make_solid_text_sp`

New in version 2.0. Span *components* with text spanner. Position spanner below staff and configure with *left_text*, solid line and upward-pointing nib at right.

```
abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> spannertools.make_solid_text_spanner_below_with_nib_at_right('foo', t[:])
TextSpanner(c'8, d'8, e'8, f'8)
abjad> f(t)
\new Staff {
  \override TextSpanner #'bound-details #'left #'text = \markup { foo }
  \override TextSpanner #'bound-details #'right #'text = #(markup #:draw-line '(0 . 1))
  \override TextSpanner #'bound-details #'right-broken #'text = ##f
```

```

\override TextSpanner #'dash-fraction = #1
\override TextSpanner #'direction = #down
c'8 \startTextSpan
d'8
e'8
f'8 \stopTextSpan
\revert TextSpanner #'bound-details #'left #'text
\revert TextSpanner #'bound-details #'right #'text
\revert TextSpanner #'bound-details #'right-broken #'text
\revert TextSpanner #'dash-fraction
\revert TextSpanner #'direction
}

```

Changed in version 2.0: renamed `spanners.solid_text_spanner_below_with_nib_at_right()` to `spannertools.make_solid_text_spanner_below_with_nib_at_right()`.

spannertools.make_spanner_schema

`abjad.tools.spannertools.make_spanner_schema.make_spanner_schema(components)`

New in version 2.0. Make schema of spanners contained by *components*:

```

abjad> voice = Voice(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(voice)
abjad> beam = spannertools.BeamSpanner(voice.leaves[:4])
abjad> slur = spannertools.SlurSpanner(voice[-2:])

```

```

abjad> f(voice)
\new Voice {
  {
    \time 2/8
    c'8 [
    d'8
  ]
  {
    \time 2/8
    e'8
    f'8 ]
  }
  {
    \time 2/8
    g'8 (
    a'8
  )
  {
    \time 2/8
    b'8
    c''8 )
  }
}

```

```

abjad> spannertools.make_spanner_schema(voice.leaves[2:4])
{BeamSpanner(c'8, d'8, e'8, f'8): [0, 1]}

```

Return dictionary.

spannertools.move_spanners_from_component_to_children_of_component

`abjad.tools.spannertools.move_spanners_from_component_to_children_of_component.move_spanners`

Give spanners attaching directly to donor to recipients. Usual use is to give attached spanners from parent to children, which is a composer-safe operation. Changed in version 2.0: renamed `spannertools.give_attached_to_children()` to `spannertools.move_spanners_from_component_to_children_of_component()`.

spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_component

`abjad.tools.spannertools.report_as_string_format_contributions_of_all_spanners_attached_to`

New in version 1.1. Report as string format contributions of all spanners attached to *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8
  e'8
  f'8 ] ) \stopTrillSpan
}
```

```
abjad> spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_component (
'BeamSpanner\n\t\t\t[\nSlurSpanner\n\t\t\t[\n\t\t\t(\n'
```

Return string.

spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_improper_parentage_of_component

`abjad.tools.spannertools.report_as_string_format_contributions_of_all_spanners_attached_to`

New in version 1.1. Report as string format contributions of all spanners attached to improper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(staff.leaves)
abjad> slur = spannertools.SlurSpanner(staff.leaves)
abjad> trill = spannertools.TrillSpanner(staff)
abjad> f(staff)
\new Staff {
  c'8 [ ( \startTrillSpan
  d'8
  e'8
  f'8 ] ) \stopTrillSpan
}
```

```
abjad> spannertools.report_as_string_format_contributions_of_all_spanners_attached_to_component (
'BeamSpanner\n\t\t\t[\nSlurSpanner\n\t\t\t[\n\t\t\t(\n'
```

Return string.

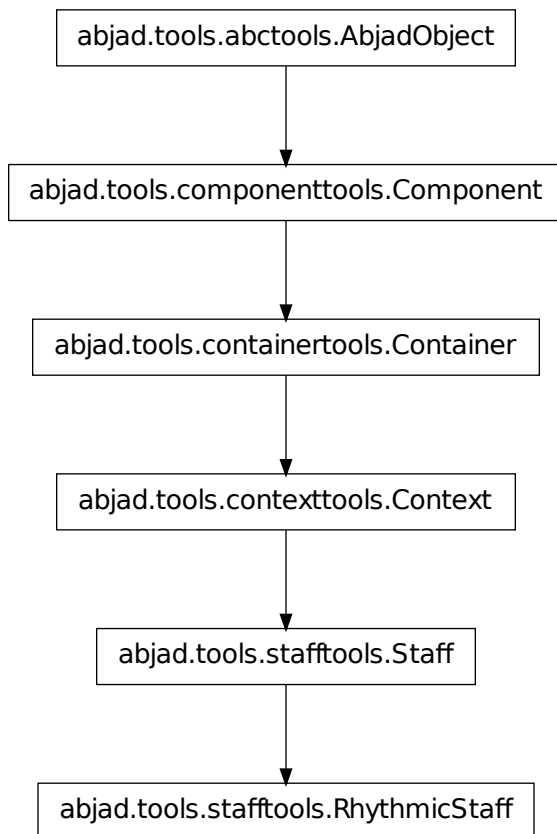
`spannertools.withdraw_components_from_spanners_covered_by_components`

`abjad.tools.spannertools.withdraw_components_from_spanners_covered_by_components.withdraw_`
 Find every spanner covered by ‘components’. Withdraw all components in ‘components’ from covered spanners. Return ‘components’. The operation always leaves all score trees in tact. Changed in version 2.0: renamed `spannertools.withdraw_from_covered()` to `spannertools.withdraw_components_from_spanners_covered_by_components()`.

`stafftools`

`stafftools` concrete classes

`stafftools.RhythmicStaff`



class `abjad.tools.stafftools.RhythmicStaff.RhythmicStaff` (*music=None*,
***kwargs*
 Abjad model of a rhythmic staff.

Read-only Properties

`RhythmicStaff.contents_duration`

Note: Inherited from `containertools.Container`

`RhythmicStaff.duration_in_seconds`

Note: Inherited from `containertools.Container`

`RhythmicStaff.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
  \consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

`RhythmicStaff.engraver_removals`

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

`RhythmicStaff.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`RhythmicStaff.is_semantic`

Note: Inherited from `contexttools.Context`

`RhythmicStaff.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.leaves  
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

RhythmicStaff.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

RhythmicStaff.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.music  
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

RhythmicStaff.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

RhythmicStaff.parent

Note: Inherited from `componenttools.Component`

RhythmicStaff.preprolated_duration

Note: Inherited from `containertools.Container`

RhythmicStaff.prolated_duration

Note: Inherited from `componenttools.Component`

RhythmicStaff.prolation

Note: Inherited from `componenttools.Component`

`RhythmicStaff.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`RhythmicStaff.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`RhythmicStaff.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`RhythmicStaff.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'
```

```
abjad> voice.is_nonsemantic = True
```

```
abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])
```

```
abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

RhythmicStaff.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

RhythmicStaff.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

RhythmicStaff.**append** (*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

RhythmicStaff.**extend** (*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

RhythmicStaff.**index** (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`RhythmicStaff.insert(i, component)`

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.insert(1, Note("cs'8"))
```

```
abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`RhythmicStaff.pop(i=-1)`

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.pop(-1)
Note("e'8")
```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`RhythmicStaff.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`RhythmicStaff.__add__(expr)`

Concatenate containers self and *expr*. The operation `c = a + b` returns a new `Container` *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__contains__(expr)`

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`RhythmicStaff.__delitem__(i)`
Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`RhythmicStaff.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__getitem__(i)`
Return component at index i in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`RhythmicStaff.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__imul__(total)`
Multiply contents of container 'total' times. Return multiplied container.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__mul__(n)`

Note: Inherited from `componenttools.Component`

`RhythmicStaff.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`RhythmicStaff.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`RhythmicStaff.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`RhythmicStaff.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`RhythmicStaff.__setitem__(i, expr)`

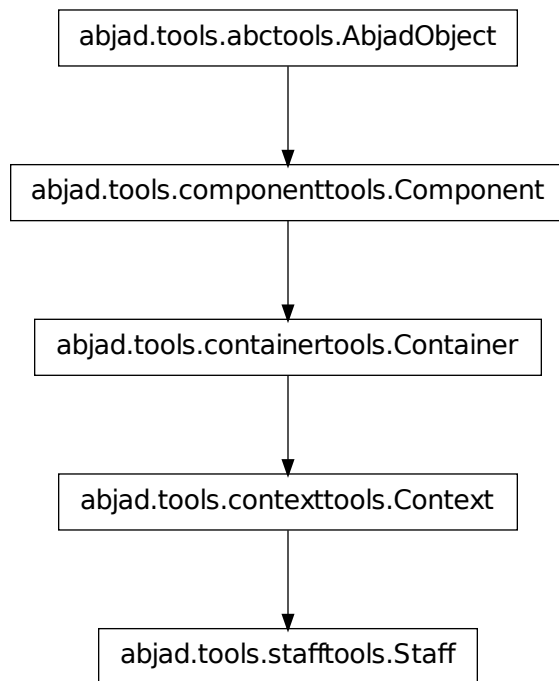
Set ‘expr’ in self at nonnegative integer index i. Or, set ‘expr’ in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with ‘expr’. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`RhythmicStaff.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

stafftools.Staff



class `abjad.tools.stafftools.Staff.Staff.Staff` (*music=None, **kwargs*)

Abjad model of a staff:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> f(staff)

```



```
\new Staff {  
  c'8  
  d'8  
  e'8  
  f'8  
}
```

Return staff object.

Read-only Properties

Staff.contents_duration

Note: Inherited from `containertools.Container`

Staff.duration_in_seconds

Note: Inherited from `containertools.Container`

Staff.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])  
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')  
abjad> f(staff)  
\new Staff \with {  
  \consists Horizontal_bracket_engraver  
} {  
}
```

Note: Inherited from `contextttools.Context`

Staff.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])  
abjad> staff.engraver_removals.add('Time_signature_engraver')  
abjad> f(staff)  
\new Staff \with {  
  \remove Time_signature_engraver  
} {  
}
```

Note: Inherited from `contextttools.Context`

Staff.format

Read-only LilyPond input format of component.

Note: Inherited from `componentttools.Component`

Staff.is_semantic

Note: Inherited from `contexttools.Context`

Staff.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Staff.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Staff.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Staff.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Staff.parent

Note: Inherited from `componenttools.Component`

Staff.preprolated_duration

Note: Inherited from `containertools.Container`

Staff.**prolated_duration**

Note: Inherited from `componenttools.Component`

Staff.**prolation**

Note: Inherited from `componenttools.Component`

Staff.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Staff.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Staff.**context_name**

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

Staff.**is_nonsemantic**

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'

abjad> voice.is_nonsemantic = True

abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice iteration and other functions.

Note: Inherited from `contexttools.Context`

Staff.**is_parallel**

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Staff.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

Staff.append(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.append(Note("f'8"))

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

Staff.extend(*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
}
```

```
        ds'8
        es'8
    }
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Staff.index (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Staff.insert (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Staff.pop (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

Staff.remove (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

Staff.__add__ (*expr*)

Concatenate containers self and *expr*. The operation `c = a + b` returns a new Container *c* with the content of both

a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`Staff.__contains__(expr)`

True if `expr` is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Staff.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Staff.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Staff.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Staff.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Staff.__getitem__(i)`

Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Staff.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Staff.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Staff.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Staff.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Staff.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Staff.__len__()`
Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Staff.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Staff.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Staff.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Staff.__radd__(expr)`
Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`Staff.__repr__()`
Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`Staff.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`Staff.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Staff.__setitem__(i, expr)`
Set 'expr' in self at nonnegative integer index i. Or, set 'expr' in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`Staff.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

stafftools functions

stafftools.all_are_staves

`abjad.tools.stafftools.all_are_staves.all_are_staves(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad staves:

```
abjad> staff = Staff("c'4 d'4 e'4 f'4")

abjad> stafftools.all_are_staves([staff])
True
```

True when *expr* is an empty sequence:

```
abjad> stafftools.all_are_staves([])
True
```

Otherwise false:

```
abjad> stafftools.all_are_staves('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

stafftools.get_first_staff_in_improper_parentage_of_component

`abjad.tools.stafftools.get_first_staff_in_improper_parentage_of_component.get_first_staff_in_`

New in version 2.0. Get first staff in improper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> stafftools.get_first_staff_in_improper_parentage_of_component(staff[1])
Staff{4}
```

Return staff or none.

stafftools.get_first_staff_in_proper_parentage_of_component

`abjad.tools.stafftools.get_first_staff_in_proper_parentage_of_component.get_first_staff_in_`

New in version 2.0. Get first staff in proper parentage of *component*:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")

abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}

abjad> stafftools.get_first_staff_in_proper_parentage_of_component(staff[1])
Staff{4}
```

Return staff or none.

stafftools.iterate_staves_backward_in_expr

`abjad.tools.stafftools.iterate_staves_backward_in_expr.iterate_staves_backward_in_expr` (*expr*,
start=
stop=

New in version 2.0. Iterate staves backward in *expr*:

```
abjad> score = Score(4 * Staff([]))

abjad> f(score)
\new Score <<
    \new Staff {
    }
    \new Staff {
    }
    \new Staff {
```

```

    }
    \new Staff {
    }
>>

abjad> for staff in stafftools.iterate_staves_backward_in_expr(score):
...     staff
...
Staff{}
Staff{}
Staff{}
Staff{}

```

Return generator.

stafftools.iterate_staves_forward_in_expr

`abjad.tools.stafftools.iterate_staves_forward_in_expr.iterate_staves_forward_in_expr(expr, start=0, stop=None)`

New in version 2.0. Iterate staves forward in *expr*:

```

abjad> score = Score(4 * Staff([]))

abjad> f(score)
\new Score <<
  \new Staff {
  }
  \new Staff {
  }
  \new Staff {
  }
  \new Staff {
  }
>>

abjad> for staff in stafftools.iterate_staves_forward_in_expr(score):
...     staff
...
Staff{}
Staff{}
Staff{}
Staff{}

```

Return generator.

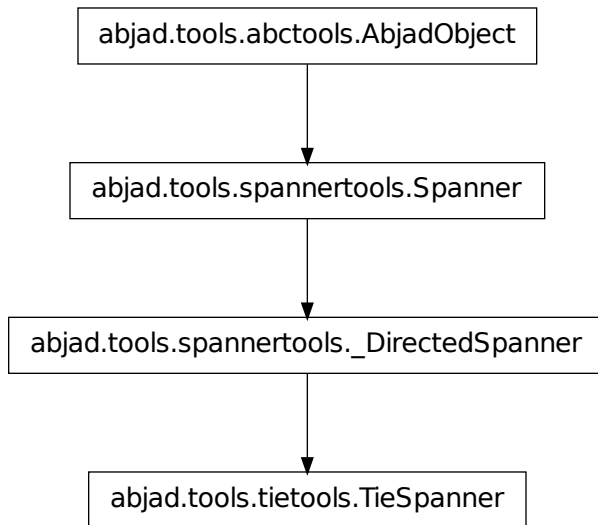
stafftools.make_rhythmic_sketch_staff

`abjad.tools.stafftools.make_rhythmic_sketch_staff.make_rhythmic_sketch_staff(music)`
 Make rhythmic staff with transparent meter and transparent bar lines.

tietools

tietools concrete classes

tietools.TieSpanner



class `abjad.tools.tietools.TieSpanner.TieSpanner` **TieSpanner** (*music=None*, *direction=None*)

Abjad tie spanner:

```

abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tietools.TieSpanner(staff[:])
TieSpanner(c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
    c'8 ~
    c'8 ~
    c'8 ~
    c'8
}

```

Return tie spanner.

Read-only Properties

`TieSpanner.components`

Return read-only tuple of components in spanner.

```

abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.components
(Note("c'8"), Note("d'8"))

```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: Inherited from `spannertools.Spanner`

TieSpanner.duration_in_seconds

Sum of duration of all leaves in spanner, in seconds.

Note: Inherited from `spannertools.Spanner`

TieSpanner.leaves

Return read-only tuple of leaves in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner.leaves
(Note("c'8"), Note("d'8"))
```

Changed in version 1.1: Now returns an (immutable) tuple instead of a (mutable) list.

Note: When dealing with large, complex scores accessing this attribute can take some time. Best to make a local copy with `leaves = spanner.leaves` first. Or use spanner- specific iteration tools.

Note: Inherited from `spannertools.Spanner`

TieSpanner.offset

New in version 1.1. Return read-only reference to spanner offset interface.

Spanner offset interface implements `start` and `stop` attributes.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner._offset.start
Offset(1, 4)
```

```
abjad> spanner._offset.stop
Offset(1, 2)
```

Return duration.

Note: Inherited from `spannertools.Spanner`

TieSpanner.override

LilyPond grob override component plug-in.

Note: Inherited from `spannertools.Spanner`

TieSpanner.preprolated_duration

Sum of preprolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.prolated_duration`

Sum of prolated duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.set`

LilyPond context setting component plug-in.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.written_duration`

Sum of written duration of all components in spanner.

Note: Inherited from `spannertools.Spanner`

Read/write Properties

`TieSpanner.direction`

Note: Inherited from `spannertools._DirectedSpanner`

Methods

`TieSpanner.append(component)`

Add *component* to right of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.append(voice[2])
abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.append_left(component)`

Add *component* to left of spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.append_left(voice[1])
abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.clear()`

Remove all components from spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

```
abjad> spanner.clear()
abjad> spanner
Spanner()
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.extend(components)`

Add iterable *components* to right of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:2])
abjad> spanner
Spanner(c'8, d'8)
```

```
abjad> spanner.extend(voice[2:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.extend_left(components)`

Add iterable *components* to left of spanner:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)
```

```
abjad> spanner.extend_left(voice[:2])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)
```

Return none.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.fracture(i, direction='both')`

Fracture spanner at *direction* of component at index *i*.

Valid values for *direction* are 'left', 'right' and 'both'.

Return original, left and right spanners.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> beam = spannertools.BeamSpanner(voice[:])
abjad> beam
BeamSpanner(c'8, d'8, e'8, f'8)

abjad> beam.fracture(1, direction = 'left')
(BeamSpanner(c'8, d'8, e'8, f'8), BeamSpanner(c'8), BeamSpanner(d'8, e'8, f'8))

abjad> print voice.format
\new Voice {
  c'8 [ ]
  d'8 [
  e'8
  f'8 ]
}
```

Return tuple.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.fuse(spanner)`

Fuse contiguous spanners.

Return new spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> left_beam = spannertools.BeamSpanner(voice[:2])
abjad> right_beam = spannertools.BeamSpanner(voice[2:])

abjad> print voice.format
\new Voice {
  c'8 [
  d'8 ]
  e'8 [
  f'8 ]
}

abjad> left_beam.fuse(right_beam)
[(BeamSpanner(c'8, d'8), BeamSpanner(e'8, f'8), BeamSpanner(c'8, d'8, e'8, f'8))]
```

```
abjad> print voice.format
\new Voice {
  c'8 [
  d'8
  e'8
  f'8 ]
}
```

Todo

Return (immutable) tuple instead of (mutable) list.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.index(component)`

Return nonnegative integer index of *component* in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[2:])
abjad> spanner
Spanner(e'8, f'8)

abjad> spanner.index(voice[-2])
0
```

Return nonnegative integer.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.pop()`

Remove and return rightmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop()
Note("f'8")

abjad> spanner
Spanner(c'8, d'8, e'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.pop_left()`

Remove and return leftmost component in spanner.

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> spanner = spannertools.Spanner(voice[:])
abjad> spanner
Spanner(c'8, d'8, e'8, f'8)

abjad> spanner.pop_left()
Note("c'8")

abjad> spanner
Spanner(d'8, e'8, f'8)
```

Return component.

Note: Inherited from `spannertools.Spanner`

Special Methods

`TieSpanner.__contains__(expr)`

Note: Inherited from `spannertools.Spanner`

`TieSpanner.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TieSpanner.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TieSpanner.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TieSpanner.__getitem__(expr)`

Note: Inherited from `spannertools.Spanner`

`TieSpanner.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`TieSpanner.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TieSpanner.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TieSpanner.__len__()`

Note: Inherited from `spannertools.Spanner`

`TieSpanner.__lt__(other)`

Trivial comparison to allow doctests to work.

Note: Inherited from `spannertools.Spanner`

`TieSpanner.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TieSpanner.__repr__()`

Note: Inherited from `spannertools.Spanner`

`TieSpanner.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TieSpanner.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

tietools functions

`tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration`

`abjad.tools.tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration()`

Scale tie chain by multiplier. Wraps `tie_chain_duration_change`. Returns tie chain. Changed in version 2.0: renamed `tietools.duration_scale()` to `tietools.add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration()`.

`tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration`

`abjad.tools.tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration.add_or_remove_tie_chain_notes_to_achieve_written_duration()`

Change the written duration of tie chain, adding and subtracting notes as necessary.

Return newly modified tie chain. Changed in version 2.0: renamed `tietools.duration_change()` to `tietools.add_or_remove_tie_chain_notes_to_achieve_written_duration()`.

tietools.apply_tie_spanner_to_leaf_pair

`abjad.tools.tietools.apply_tie_spanner_to_leaf_pair.apply_tie_spanner_to_leaf_pair` (*left*, *right*)

Apply tie spanner to *left* leaf and *right* leaf:

```
abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'8)
abjad> f(staff)
\new Staff {
    c'8 ~
    c'8
    c'8
    c'8
}

abjad> tietools.apply_tie_spanner_to_leaf_pair(staff[1], staff[2])

abjad> f(staff)
\new Staff {
    c'8 ~
    c'8 ~
    c'8
    c'8
}
```

Handle existing tie spanners intelligently.

Return `none`. Changed in version 2.0: renamed `tietools.span_leaf_pair()` to `tietools.apply_tie_spanner_to_leaf_pair()`.

tietools.are_components_in_same_tie_spanner

`abjad.tools.tietools.are_components_in_same_tie_spanner.are_components_in_same_tie_spanner`
True if all components in list share same tie spanner, otherwise False.
Changed in version 2.0: renamed `tietools.are_in_same_spanner()` to `tietools.are_components_in_same_tie_spanner()`.

tietools.get_leaves_in_tie_chain

`abjad.tools.tietools.get_leaves_in_tie_chain.get_leaves_in_tie_chain` (*tie_chain*)
Return Python list of leaves in tie chain.

tietools.get_preprolated_tie_chain_duration

`abjad.tools.tietools.get_preprolated_tie_chain_duration.get_preprolated_tie_chain_duration`
Get sum of preprolated duration of all leaves in *tie_chain*.

Todo

write `tietools.get_preprolated_tie_chain_duration()` tests.

Changed in version 2.0: renamed `tietools.get_duration_preprolated()` to `tietools.get_preprolated_tie_chain_duration()`.

`tietools.get_prolated_tie_chain_duration`

`abjad.tools.tietools.get_prolated_tie_chain_duration.get_prolated_tie_chain_duration(tie_chain)`
 Return sum of prolated duration of all leaves in chain.

Todo

Write `tietools.get_prolated_tie_chain_duration()` tests.

Changed in version 2.0: renamed `tietools.get_duration_prolated()` to `tietools.get_prolated_tie_chain_duration()`. Changed in version 2.0: renamed `tietools.get_tie_chain_prolated_duration()` to `tietools.get_prolated_tie_chain_duration()`.

`tietools.get_tie_chain`

`abjad.tools.tietools.get_tie_chain.get_tie_chain(component)`
 New in version 2.0. Get tie chain from *component*.

`tietools.get_tie_chain_duration_in_seconds`

`abjad.tools.tietools.get_tie_chain_duration_in_seconds.get_tie_chain_duration_in_seconds(tie_chain)`
 Return sum of seconds duration of all leaves in chain.

Todo

Write `tietools.get_tie_chain_duration_in_seconds()` tests.

Changed in version 2.0: renamed `tietools.get_duration_seconds()` to `tietools.get_tie_chain_duration_in_seconds()`.

`tietools.get_tie_chains_in_expr`

`abjad.tools.tietools.get_tie_chains_in_expr.get_tie_chains_in_expr(components)`
 This function returns all tie chains in components. A tie chain may not encompass all the leaves spanned by its corresponding Tie spanner, but only those found in the given list. i.e. the function returns the intersection between all the leaves spanned by all tie spanners touching the components given and the leaves found in the given components list. Changed in version 2.0: renamed `tietools.get_tie_chains()` to `tietools.get_tie_chains_in_expr()`.

`tietools.get_written_tie_chain_duration`

`abjad.tools.tietools.get_written_tie_chain_duration.get_written_tie_chain_duration(tie_chain)`
 Return sum of written duration of all leaves in chain.

tietools.group_leaves_in_tie_chain_by_immediate_parents

`abjad.tools.tietools.group_leaves_in_tie_chain_by_immediate_parents.group_leaves_in_tie_chain_by_immediate_parents`

Group leaves in *tie_chain* by immediate parent:

```
abjad> staff = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> tietools.TieSpanner(staff.leaves)
TieSpanner(c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
  {
    \time 2/8
    c'8 ~
    c'8 ~
  }
  {
    \time 2/8
    c'8 ~
    c'8
  }
}
```

```
abjad> tie_chain = tietools.get_tie_chain(staff.leaves[0])
abjad> tietools.group_leaves_in_tie_chain_by_immediate_parents(tie_chain)
[[Note("c'8"), Note("c'8")], [Note("c'8"), Note("c'8")]]
```

Return list of leaf group lists. Changed in version 2.0: renamed `tietools.group_by_parent()` to `tietools.group_leaves_in_tie_chain_by_immediate_parents()`.

tietools.is_component_with_tie_spanner_attached

`abjad.tools.tietools.is_component_with_tie_spanner_attached.is_component_with_tie_spanner_attached`

New in version 2.0. True when *expr* is component with tie spanner attached:

```
abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tietools.TieSpanner(staff[:])
TieSpanner(c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
  c'8 ~
  c'8 ~
  c'8 ~
  c'8
}
abjad> tietools.is_component_with_tie_spanner_attached(staff)
False
```

Otherwise false:

```
abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tietools.TieSpanner(staff[:])
TieSpanner(c'8, c'8, c'8, c'8)
abjad> f(staff)
\new Staff {
  c'8 ~
  c'8 ~
  c'8 ~
}
```

```

        c'8
    }
    abjad> tietools.is_component_with_tie_spanner_attached(staff[1])
    True

```

Return boolean.

tietools.is_tie_chain

`abjad.tools.tietools.is_tie_chain.is_tie_chain(expr)`
 True when *expr* is a tie chain, otherwise False.

tietools.is_tie_chain_with_all_leaves_in_same_parent

`abjad.tools.tietools.is_tie_chain_with_all_leaves_in_same_parent.is_tie_chain_with_all_leaves_in_same_parent(expr)`
 True when *expr* is a tie chain with all leaves in same parent.

That is, True when tie chain crosses no container boundaries, otherwise False.

Example:

```

abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 2)
abjad> tietools.TieSpanner(t.leaves[1:3])
TieSpanner(c'8, c'8)

\new Staff {
    \time 2/8
    c'8
    c'8 ~
    \time 2/8
    c'8
    c'8
}

abjad> tie_chain = tietools.get_tie_chain(t.leaves[0])
abjad> assert tietools.is_tie_chain_with_all_leaves_in_same_parent(tie_chain)
abjad> tie_chain = tietools.get_tie_chain(t.leaves[1])
abjad> assert not tietools.is_tie_chain_with_all_leaves_in_same_parent(tie_chain)
abjad> tie_chain = tietools.get_tie_chain(t.leaves[2])
abjad> assert not tietools.is_tie_chain_with_all_leaves_in_same_parent(tie_chain)
abjad> tie_chain = tietools.get_tie_chain(t.leaves[3])
abjad> assert tietools.is_tie_chain_with_all_leaves_in_same_parent(tie_chain)

```

Changed in version 2.0: renamed `tietools.is_in_same_parent()` to `tietools.is_tie_chain_with_all_leaves_in_same_parent()`.

tietools.iterate_tie_chains_backward_in_expr

`abjad.tools.tietools.iterate_tie_chains_backward_in_expr.iterate_tie_chains_backward_in_expr(expr)`
 Yield right-to-left tie chains in *expr*:

```

abjad> notes = notetools.make_notes([0], [(5, 16), (1, 8), (1, 8), (5, 16)])
abjad> staff = Staff(notes)
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 16), staff[1:3])
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)

```



```

abjad> print staff.format
\new Staff {
  c'4 ~
  \times 2/3 {
    c'16
    d'8
  }
  e'8
  f'4 ~
  f'16
}

abjad> for x in tietools.iterate_tie_chains_backward_in_expr(staff):
...     x
...
(Note("f'4"), Note("f'16"))
(Note("e'8"),)
(Note("d'8"),)
(Note("c'4"), Note("c'16"))

```

Note that one-note tie chains yield the same as other tie chains.

Note also that nested structures are no problem. Changed in version 2.0: renamed `iterate.tie_chains_backward_in()` to `tietools.iterate_tie_chains_backward_in_expr()`. Changed in version 2.0: renamed `iterate.tie_chains_backward_in_expr()` to `tietools.iterate_tie_chains_backward_in_expr()`.

`tietools.iterate_tie_chains_forward_in_expr`

`abjad.tools.tietools.iterate_tie_chains_forward_in_expr.iterate_tie_chains_forward_in_expr`
Yield left-to-right tie chains in *expr*:

```

abjad> notes = notetools.make_notes([0], [(5, 16), (1, 8), (1, 8), (5, 16)])
abjad> staff = Staff(notes)
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 16), staff[1:3])
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> print staff.format
\new Staff {
  c'4 ~
  \times 2/3 {
    c'16
    d'8
  }
  e'8
  f'4 ~
  f'16
}

abjad> for x in tietools.iterate_tie_chains_forward_in_expr(staff):
...     x
...
(Note("c'4"), Note("c'16"))
(Note("d'8"),)
(Note("e'8"),)
(Note("f'4"), Note("f'16"))

```

Note that one-note tie chains yield the same as other tie chains.

Note also that nested structures are no problem. Changed in version 2.0: renamed `iterate.tie_chains_forward_in()` to `tietools.iterate_tie_chains_forward_in_expr()`. Changed in version 2.0: renamed `iterate.tie_chains_forward_in_expr()` to `tietools.iterate_tie_chains_forward_in_expr()`.

`tietools.iterate_topmost_tie_chains_and_components_forward_in_expr`

`abjad.tools.tietools.iterate_topmost_tie_chains_and_components_forward_in_expr.iterate_topmost_tie_chains_and_components_forward_in_expr`
Yield the left-to-right, top-level contents of *expr* with chain-wrapped leaves.

```
abjad> t = Staff(notetools.make_notes(0, [(5, 32)] * 4))
abjad> t.insert(4, tuplettools.FixedDurationTuplet(Duration(2, 8), notetools.make_repeated_notes(2, 8)))
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> f(t)
\new Staff {
  c'8 ~
  c'32
  d'8 ~
  d'32
  \times 2/3 {
    e'8
    f'8
    g'8
  }
  a'8 ~
  a'32
  b'8 ~
  b'32
}

abjad> for x in tietools.iterate_topmost_tie_chains_and_components_forward_in_expr(t):
...     x
...
(Note("c'8"), Note("c'32"))
(Note("d'8"), Note("d'32"))
FixedDurationTuplet(1/4, [e'8, f'8, g'8])
(Note("a'8"), Note("a'32"))
(Note("b'8"), Note("b'32"))
```

Crossing ties raise `TieChainError`. Changed in version 2.0: renamed `iterate.chained_contents()` to `tietools.iterate_topmost_tie_chains_and_components_forward_in_expr()`. Changed in version 2.0: renamed `iterate.topmost_tie_chains_and_components_forward_in_expr()` to `tietools.iterate_topmost_tie_chains_and_components_forward_in_expr()`.

`tietools.label_tie_chains_in_expr_with_prolated_tie_chain_duration`

`abjad.tools.tietools.label_tie_chains_in_expr_with_prolated_tie_chain_duration.label_tie_chains_in_expr_with_prolated_tie_chain_duration`

Label tie chains in *expr* with prolated tie chain duration:

```
abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])
FixedDurationTuplet(1/4, [c'8, c'8, c'8])
abjad> tietools.TieSpanner(staff.leaves[:2])
TieSpanner(c'8, c'8)
abjad> tietools.TieSpanner(staff.leaves[2:])
```

```

TieSpanner(c'8, c'8)
abjad> tietools.label_tie_chains_in_expr_with_prolated_tie_chain_duration(staff)
abjad> f(staff)
\new Staff {
  \times 2/3 {
    c'8 _ \markup { \small 1/6 } ~
    c'8
    c'8 _ \markup { \small 5/24 } ~
  }
  c'8
}

```

Return none.

tietools.label_tie_chains_in_expr_with_tie_chain_durations

abjad.tools.tietools.label_tie_chains_in_expr_with_tie_chain_durations.**label_tie_chains_in**

Label tie chains in *expr* with both written tie chain duration and prolated tie chain duration:

```

abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])
FixedDurationTuplet(1/4, [c'8, c'8, c'8])
abjad> tietools.TieSpanner(staff.leaves[:2])
TieSpanner(c'8, c'8)
abjad> tietools.TieSpanner(staff.leaves[2:])
TieSpanner(c'8, c'8)
abjad> tietools.label_tie_chains_in_expr_with_tie_chain_durations(staff)
abjad> f(staff)
\new Staff {
  \times 2/3 {
    c'8 _ \markup { \column { \small 1/4 \small 1/6 } } ~
    c'8
    c'8 _ \markup { \column { \small 1/4 \small 5/24 } } ~
  }
  c'8
}

```

Return none.

tietools.label_tie_chains_in_expr_with_written_tie_chain_duration

abjad.tools.tietools.label_tie_chains_in_expr_with_written_tie_chain_duration.**label_tie_cha**

Label tie chains in *expr* with written tie chain duration.:

```

abjad> staff = Staff(notetools.make_repeated_notes(4))
abjad> tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])
FixedDurationTuplet(1/4, [c'8, c'8, c'8])
abjad> tietools.TieSpanner(staff.leaves[:2])
TieSpanner(c'8, c'8)
abjad> tietools.TieSpanner(staff.leaves[2:])
TieSpanner(c'8, c'8)
abjad> tietools.label_tie_chains_in_expr_with_written_tie_chain_duration(staff)
abjad> f(staff)
\new Staff {

```

```

\times 2/3 {
    c'8 _ \markup { \small 1/4 } ~
    c'8
    c'8 _ \markup { \small 1/4 } ~
}
c'8
}

```

Return none.

tietools.remove_all_leaves_in_tie_chain_except_first

`abjad.tools.tietools.remove_all_leaves_in_tie_chain_except_first.remove_all_leaves_in_tie_chain_except_first`.
Detach all leaves of tie chain after the first. Unspan and return length-1 tie chain. Changed in version 2.0: renamed `tietools.truncate()` to `tietools.remove_all_leaves_in_tie_chain_except_first()`.

tietools.remove_tie_spanners_from_components_in_expr

`abjad.tools.tietools.remove_tie_spanners_from_components_in_expr.remove_tie_spanners_from_components_in_expr`.
Remove tie spanners components in *expr*:

```

abjad> staff = Staff("c'4 ~ c'16 d'4 ~ d'16")
abjad> f(staff)
\new Staff {
    c'4 ~
    c'16
    d'4 ~
    d'16
}

abjad> tietools.remove_tie_spanners_from_components_in_expr(staff[:])
[Note("c'4"), Note("c'16"), Note("d'4"), Note("d'16")]
abjad> f(staff)
\new Staff {
    c'4
    c'16
    d'4
    d'16
}

```

Return *expr*. Changed in version 2.0: renamed `componenttools.untie_shallow()` to `tietools.remove_tie_spanners_from_components_in_expr()`.

tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots

`abjad.tools.tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots`.

New in version 2.0. Divide *tie_chain* into fixed-duration tuplet according to arbitrary integer *proportions*.

Interpret *proportions* as a ratio. That is, reduce integers in *proportions* relative to each other.

Return non-trivial tuplet as augmentation.

Where `proportions[i] == 1` for `i < len(proportions)`, do not allow tupletted notes to carry dots.

```

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots(tie_chain, [1])
FixedDurationTuplet(3/16, [c'8])
abjad> f(staff)
\new Staff {
    \fraction \times 3/2 {
        c'8 [
    ]
    c'16 ]
}

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots(tie_chain, [1, 2])
FixedDurationTuplet(3/16, [c'16, c'8])
abjad> f(staff)
\new Staff {
    {
        c'16 [
        c'8
    ]
    c'16 ]
}

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots(tie_chain, [1, 2])
FixedDurationTuplet(3/16, [c'32, c'16, c'16])
abjad> f(staff)
\new Staff {
    \fraction \times 6/5 {
        c'32 [
        c'16
        c'16
    ]
    c'16 ]
}

```

Changed in version 2.0: renamed `divide.tie_chain_into_arbitrary_augmentation_undotted()` to `tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots()`.

tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots

abjad.tools.tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots.**tie**

New in version 2.0. Divide *tie_chain* into fixed-duration tuplet according to arbitrary integer *proportions*.

Interpret *proportions* as a ratio. That is, reduce integers in *proportions* relative to each other.

Return non-trivial tuplet as augmentation.

Where `proportions[i] == 1` for `i < len(proportions)`, allow tupletted notes to carry dots.

```
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots(tie_chain, [1])
FixedDurationTuplet(3/16, [c'8.])
abjad> f(staff)
\new Staff {
    {
        c'8. [
    ]
    c'16 ]
}
```

```
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots(tie_chain, [1, 1])
FixedDurationTuplet(3/16, [c'16, c'8])
abjad> f(staff)
\new Staff {
    {
        c'16 [
        c'8
    ]
    c'16 ]
}
```

```
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots(tie_chain, [1, 1, 1])
FixedDurationTuplet(3/16, [c'64., c'32., c'32.])
abjad> f(staff)
\new Staff {
    \fraction \times 8/5 {
        c'64. [
    ]
    c'16 ]
}
```

```

        c'32.
        c'32.
    }
    c'16 ]
}

```

Changed in version 2.0: renamed `divide.tie_chain_into_arbitrary_augmentation_dotted()` to `tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots()`.

tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots

`abjad.tools.tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots.tie_cha`

New in version 2.0. Divide *tie_chain* into fixed-duration tuplet according to arbitrary integer *proportions*.

Interpret *proportions* as a ratio. That is, reduce integers in *proportions* relative to each other.

Return non-trivial tuplet as diminution.

Where `proportions[i] == 1` for `i < len(proportions)`, do not allow tupletted notes to carry dots.

```

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots(tie_chain, [1])
FixedDurationTuplet(3/16, [c'4])
abjad> f(staff)
\new Staff {
    \fraction \times 3/4 {
        c'4 [
    ]
    c'16 ]
}

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots(tie_chain, [1, 2])
FixedDurationTuplet(3/16, [c'16, c'8])
abjad> f(staff)
\new Staff {
    {
        c'16 [
        c'8
    ]
    c'16 ]
}

abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])

```

```
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots(tie_chain, [1, 2,
FixedDurationTuplet(3/16, [c'16, c'8, c'8])
abjad> f(staff)
\new Staff {
  \fraction \times 3/5 {
    c'16 [
      c'8
      c'8
    ]
  }
  c'16 ]
}
```

Changed in version 2.0: renamed `divide.tie_chain_into_arbitrary_diminution_undotted()` to `tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots()`.

`tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots`

```
abjad.tools.tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots.ti
```

New in version 2.0. Divide *tie_chain* into fixed-duration tuplet according to arbitrary integer *proportions*.

Interpret *proportions* as a ratio. That is, reduce integers in *proportions* relative to each other.

Return non-trivial tuplet as diminution.

Where `proportions[i] == 1` for `i < len(proportions)`, allow tupletted notes to carry dots.

```
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots(tie_chain, [1
FixedDurationTuplet(3/16, [c'8.])
abjad> f(staff)
\new Staff {
  {
    c'8. [
  ]
  c'16 ]
}
```

```
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots(tie_chain, [1
FixedDurationTuplet(3/16, [c'16, c'8])
abjad> f(staff)
```



```
\new Staff {
  {
    c'16 [
      c'8
    ]
  }
  c'16 ]
}

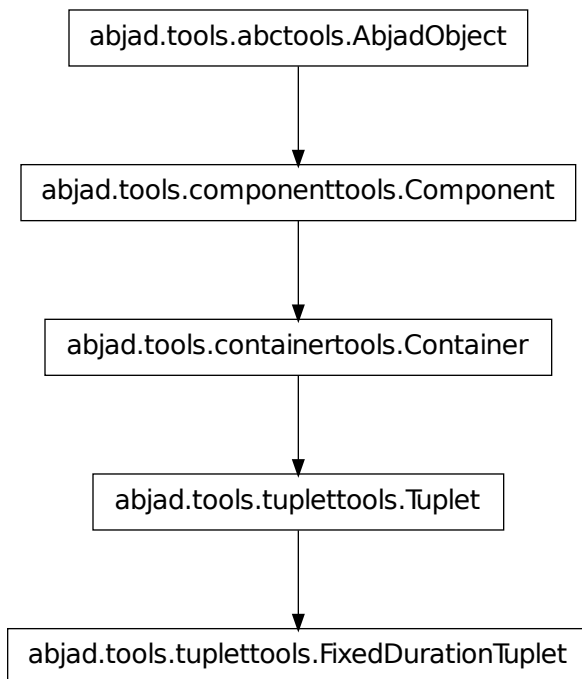
abjad> staff = Staff([Note(0, (1, 8)), Note(0, (1, 16)), Note(0, (1, 16))])
abjad> tietools.TieSpanner(staff[:2])
TieSpanner(c'8, c'16)
abjad> spannertools.BeamSpanner(staff[:])
BeamSpanner(c'8, c'16, c'16)
abjad> tie_chain = tietools.get_tie_chain(staff[0])
abjad> tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots(tie_chain, [1
FixedDurationTuplet(3/16, [c'32., c'16., c'16.])
abjad> f(staff)
\new Staff {
  \times 4/5 {
    c'32. [
      c'16.
      c'16.
    ]
  }
  c'16 ]
}
```

Changed in version 2.0: renamed `divide.tie_chain_into_arbitrary_diminution_dotted()` to `tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots()`.

`tuplettools`

`tuplettools` concrete classes

`tuplettools.FixedDurationTuplet`



```

class abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet(FixedDurationTuplet (duration, multiplier, contents, sic=0, **kwargs))

```

Abjad tuplet of fixed duration and variable multiplier:

```

abjad> tuplettools.FixedDurationTuplet(Fraction(2, 8), "c'8 d'8 e'8")
FixedDurationTuplet(1/4, [c'8, d'8, e'8])

```

Return fixed-duration tuplet.

Read-only Properties

`FixedDurationTuplet.contents_duration`

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.duration_in_seconds`

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.is_augmentation`

True when multiplier is greater than 1. Otherwise false:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.is_augmentation
False
```

Return boolean.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_binary`

Read-only boolean true when multiplier numerator is power of two. Otherwise false:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.is_binary
True
```

Return boolean.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_diminution`

True when multiplier is less than 1. Otherwise false:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.is_diminution
True
```

Return boolean.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_nonbinary`

Read-only boolean true when multiplier numerator is not power of two. Otherwise false:

```
abjad> tuplet = Tuplet((3, 5), "c'8 d'8 e'8 f'8 g'8")
abjad> tuplet.is_nonbinary
True
```

Return boolean.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_trivial`

True when tuplet multiplier is one. Otherwise false:

```
abjad> tuplet = Tuplet((1, 1), "c'8 d'8 e'8")
abjad> tuplet.is_trivial
True
```

Return boolean.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.leaves`

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.multiplied_duration`

Read-only multiplied duration of tuplet:

```
abjad> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")
abjad> tuplet.multiplied_duration
Duration(1, 4)
```

Return duration.

`FixedDurationTuplet.music`

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.parent`

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.preprolated_duration`

Duration prior to prolation:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.preprolated_duration
Duration(1, 4)
```

Return duration.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.prolated_duration`

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.prolation`

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.ratio_string`

Read-only tuplet multiplier formatted with colon as ratio:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.ratio_string
'3:2'
```

Return string.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`FixedDurationTuplet.force_fraction`

Read / write boolean to force $n:m$ fraction in LilyPond format:

```
abjad> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```

abjad> tuplet.force_fraction is None
True

abjad> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}

abjad> tuplet.force_fraction = True

abjad> f(tuplet)
\fraction \times 2/3 {
    c'8
    d'8
    e'8
}

```

Return boolean or none.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_invisible`

Read / write boolean to render as LilyPond `\scaledDurations` construct instead of tuplet:

```

abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")

abjad> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}

abjad> tuplet.is_invisible = True

\scaledDurations #'(2 . 3) {
    c'8
    d'8
    e'8
}

```

This has the effect of rendering no no tuplet bracket and no tuplet number while preserving the rhythmic value of the tuplet and the contents of the tuplet.

Return boolean or none.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.is_parallel`

Get parallel container:

```

abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')]

abjad> f(container)
{

```

```
\new Voice {  
  c'8  
  d'8  
  e'8  
}  
\new Voice {  
  g4.  
}  
}  
  
abjad> container.is_parallel  
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True  
  
abjad> f(container)  
<<  
  \new Voice {  
    c'8  
    d'8  
    e'8  
  }  
  \new Voice {  
    g4.  
  }  
>>
```

Return none.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.multiplier`

Read-only multiplier of tuplet:

```
abjad> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")  
abjad> tuplet.multiplier  
Fraction(2, 3)
```

Return fraction.

`FixedDurationTuplet.preferred_denominator`

New in version 2.0. Integer denominator in terms of which tuplet fraction should format:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")  
abjad> tuplet.preferred_denominator = 4  
  
abjad> f(tuplet)  
\times 4/6 {  
  c'8  
  d'8  
  e'8  
}
```

Return positive integer or none.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.target_duration`

Read / write target duration of fixed-duration tuplet:

```
abjad> tuplet = tuplettools.FixedDurationTuplet((1, 4), "c'8 d'8 e'8")
abjad> tuplet.target_duration
Duration(1, 4)
```

```
abjad> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}
```

```
abjad> tuplet.target_duration = Duration(5, 8)
abjad> f(tuplet)
\fraction \times 5/3 {
    c'8
    d'8
    e'8
}
```

Return duration.

Methods

`FixedDurationTuplet.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.append(Note("f'8"))
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.extend(expr)`

Extend *expr* against container:


```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.index(component)`

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.insert(i, component)`

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.pop(i=-1)`

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.remove(note)
```

```
abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.trim(start, stop='unused')`

Trim fixed-duration tuplet elements from *start* to *stop*:

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Fraction(2, 8), "c'8 d'8 e'8")
abjad> tuplet
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
```

```
abjad> tuplet.trim(2)
abjad> tuplet
FixedDurationTuplet(1/6, [c'8, d'8])
```

Preserve fixed-duration tuplet multiplier.

Adjust fixed-duration tuplet duration.

Return none.

Special Methods

`FixedDurationTuplet.__add__(arg)`

Add two tuplets of same type and with same multiplier.

Note: Inherited from `tuplettools.Tuplet`

`FixedDurationTuplet.__contains__(expr)`

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`FixedDurationTuplet.__delitem__(i)`

Find component(s) at index or slice *i* in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`FixedDurationTuplet.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__mul__(n)`

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FixedDurationTuplet.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__repr__()`

`FixedDurationTuplet.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`FixedDurationTuplet.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

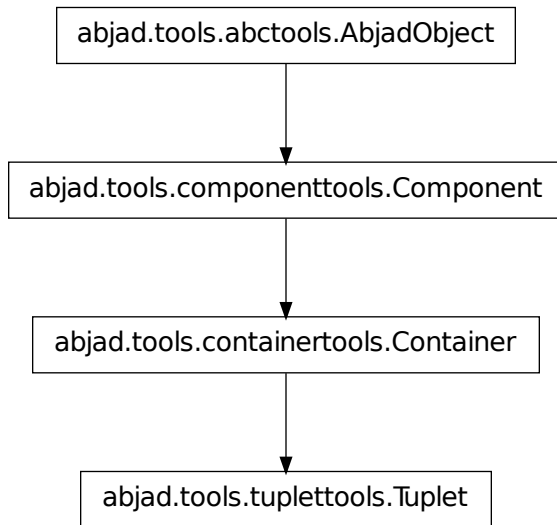
`FixedDurationTuplet.__setitem__(i, expr)`

Set 'expr' in self at nonnegative integer index i. Or, set 'expr' in self at slice i. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`FixedDurationTuplet.__str__()`

`tuplettools.Tuplet`



class `abjad.tools.tuplettools.Tuplet.Tuplet` (*multiplier*, *music=None*, ***kwargs*)

Abjad model of a tuplet:

```

abjad> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
abjad> f(tuplet)
\times 2/3 {
    c'8
    d'8
    e'8
}
  
```

Return tuplet object.

Read-only Properties

`Tuplet.contents_duration`

Note: Inherited from `containertools.Container`

`Tuplet.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Tuplet.format`

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

Tuplet.is_augmentation

True when multiplier is greater than 1. Otherwise false:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.is_augmentation
False
```

Return boolean.

Tuplet.is_binary

Read-only boolean true when multiplier numerator is power of two. Otherwise false:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.is_binary
True
```

Return boolean.

Tuplet.is_diminution

True when multiplier is less than 1. Otherwise false:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.is_diminution
True
```

Return boolean.

Tuplet.is_nonbinary

Read-only boolean true when multiplier numerator is not power of two. Otherwise false:

```
abjad> tuplet = Tuplet((3, 5), "c'8 d'8 e'8 f'8 g'8")
abjad> tuplet.is_nonbinary
True
```

Return boolean.

Tuplet.is_trivial

True when tuplet multiplier is one. Otherwise false:

```
abjad> tuplet = Tuplet((1, 1), "c'8 d'8 e'8")
abjad> tuplet.is_trivial
True
```

Return boolean.

Tuplet.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Tuplet.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Tuplet.multiplied_duration

Read-only multiplied duration of tuplet:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.multiplied_duration
Duration(1, 4)
```

Return duration.

Tuplet.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Tuplet.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Tuplet.parent

Note: Inherited from `componenttools.Component`

Tuplet.preprolated_duration

Duration prior to prolation:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> t.preprolated_duration
Duration(1, 4)
```

Return duration.

Tuplet.prolated_duration

Note: Inherited from `componenttools.Component`

Tuplet.prolation

Note: Inherited from `componenttools.Component`

Tuplet.`ratio_string`

Read-only tuplet multiplier formatted with colon as ratio:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.ratio_string
'3:2'
```

Return string.

Tuplet.`set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Tuplet.`spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties**Tuplet.`force_fraction`**

Read / write boolean to force `n:m` fraction in LilyPond format:

```
abjad> tuplet = Tuplet(Fraction(2, 3), "c'8 d'8 e'8")
```

```
abjad> tuplet.force_fraction is None
True
```

```
abjad> f(tuplet)
```

```
\times 2/3 {
  c'8
  d'8
  e'8
}
```

```
abjad> tuplet.force_fraction = True
```

```
abjad> f(tuplet)
```

```
\fraction \times 2/3 {
  c'8
  d'8
  e'8
}
```

Return boolean or none.

Tuplet.`is_invisible`

Read / write boolean to render as LilyPond `\scaledDurations` construct instead of tuplet:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
```

```
abjad> f(tuplet)
```

```
\times 2/3 {
  c'8
  d'8
  e'8
}
```

```
abjad> tuplet.is_invisible = True

\scaleDurations #'(2 . 3) {
  c'8
  d'8
  e'8
}
```

This has the effect of rendering no no tuplet bracket and no tuplet number while preserving the rhythmic value of the tuplet and the contents of the tuplet.

Return boolean or none.

Tuplet.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Tuplet.multiplier

Read / write tuplet multiplier:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.multiplier
Fraction(2, 3)
```

Return fraction.

`Tuplet.preferred_denominator`

New in version 2.0. Integer denominator in terms of which tuplet fraction should format:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
abjad> tuplet.preferred_denominator = 4
```

```
abjad> f(tuplet)
\times 4/6 {
    c'8
    d'8
    e'8
}
```

Return positive integer or none.

Methods

`Tuplet.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}
```

```
abjad> container.append(Note("f'8"))
```

```
abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`Tuplet.extend(expr)`

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
{
```

```

        c'8 [
        d'8
        e'8 ]
    }

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`Tuplet.index(component)`

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`Tuplet.insert(i, component)`

Insert *component* in container at index *i*:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
```

```
        d'8
        e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`Tuplet.pop` (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`Tuplet.remove` (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
```

```
        d'8 ]
    }
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`Tuplet.__add__(arg)`

Add two tuplets of same type and with same multiplier.

`Tuplet.__contains__(expr)`

True if expr is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Tuplet.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Tuplet.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Tuplet.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Tuplet.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`Tuplet.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Tuplet.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Tuplet.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__len__()`
Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Tuplet.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Tuplet.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Tuplet.__radd__(expr)`
Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`Tuplet.__repr__()`

`Tuplet.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`Tuplet.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Tuplet.__setitem__(i, expr)`

Set 'expr' in self at nonnegative integer index i. Or, set 'expr' in self at slice i. Find spanners that dominate self[i] and children of self[i]. Replace contents at self[i] with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`Tuplet.__str__()`

tuplettools functions

tuplettools.all_are_tuplets

`abjad.tools.tuplettools.all_are_tuplets.all_are_tuplets(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad tuplets:

```
abjad> tuplet = Tuplet((2, 3), "c'8 d'8 e'8")
```

```
abjad> tuplettools.all_are_tuplets([tuplet])
```

```
True
```

True when *expr* is an empty sequence:

```
abjad> tuplettools.all_are_tuplets([])
```

```
True
```

Otherwise false:

```
abjad> tuplettools.all_are_tuplets('foo')
```

```
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

tuplettools.beam_bottommost_tuplets_in_expr

`abjad.tools.tuplettools.beam_bottommost_tuplets_in_expr.beam_bottommost_tuplets_in_expr(expr)`

Beam bottommost tuplets in *expr*:

```
abjad> staff = Staff(3 * Tuplet(Fraction(2, 3), "c'8 d'8 e'8"))
```

```
f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  \times 2/3 {
    c'8
    d'8
    e'8
  }
}
```

```
abjad> tuplettools.beam_bottommost_tuplets_in_expr(staff)
```

```
abjad> f(staff)
\new Staff {
  \times 2/3 {
    c'8 [
    d'8
    e'8 ]
  }
  \times 2/3 {
    c'8 [
    d'8
    e'8 ]
  }
  \times 2/3 {
    c'8 [
    d'8
    e'8 ]
  }
}
```

Return none.

tuplettools.change_augmented_tuplets_in_expr_to_diminished

`abjad.tools.tuplettools.change_augmented_tuplets_in_expr_to_diminished.change_augmented_tuplets_in_expr_to_diminished(expr)`

New in version 2.0. Multiply the written duration of the leaves in *tuplet* by the least power of 2 necessary to diminished *tuplet*.

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 4), "c'8 d'8 e'8")
abjad> tuplet
```

```
FixedDurationTuplet(1/2, [c'8, d'8, e'8])
abjad> tuplettools.change_augmented_tuplets_in_expr_to_diminished(tuplet)
FixedDurationTuplet(1/2, [c'4, d'4, e'4])
```

Todo

make work with nested tuplets.

Changed in version 2.0: renamed `tuplettools.augmentation_to_diminution()` to `tuplettools.change_augmented_tuplets_in_expr_to_diminished()`.

`tuplettools.change_diminished_tuplets_in_expr_to_augmented`

`abjad.tools.tuplettools.change_diminished_tuplets_in_expr_to_augmented`. **`change_diminished_tuplets_in_expr_to_augmented`**
 New in version 2.0. Divide the written duration of the leaves in *tuplet* by the least power of 2 necessary to augment *tuplet*.

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> tuplet
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
abjad> tuplettools.change_diminished_tuplets_in_expr_to_augmented(tuplet)
FixedDurationTuplet(1/4, [c'16, d'16, e'16])
```

Todo

make work with nested tuplets.

Changed in version 2.0: renamed `tuplettools.diminution_to_augmentation()` to `tuplettools.change_diminished_tuplets_in_expr_to_augmented()`.

`tuplettools.fix_contents_of_tuplets_in_expr`

`abjad.tools.tuplettools.fix_contents_of_tuplets_in_expr`. **`fix_contents_of_tuplets_in_expr`** (*tuplet*)
 Scale *tuplet* contents by power of two if tuplet multiplier less than 1/2 or greater than 2. Return *tuplet*.

```
abjad> tuplet = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'4 d'4 e'4")
abjad> tuplet
FixedDurationTuplet(1/4, [c'4, d'4, e'4])
abjad> tuplettools.fix_contents_of_tuplets_in_expr(tuplet)
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
```

Changed in version 2.0: renamed `tuplettools.contents_fix()` to `tuplettools.fix_contents_of_tuplets_in_expr()`.

`tuplettools.fuse_tuplets`

`abjad.tools.tuplettools.fuse_tuplets`. **`fuse_tuplets`** (*tuplets*)
 Fuse parent-contiguous *tuplets*:

```
abjad> t1 = tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8")
abjad> spannertools.BeamSpanner(t1[:])
BeamSpanner(c'8, d'8, e'8)
```

```

abjad> t2 = tuplettools.FixedDurationTuplet(Duration(2, 16), "c'16 d'16 e'16")
abjad> spannertools.SlurSpanner(t2[:])
SlurSpanner(c'16, d'16, e'16)
abjad> staff = Staff([t1, t2])
abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [
            d'8
            e'8 ]
    }
    \times 2/3 {
        c'16 (
            d'16
            e'16 )
    }
}

```

```

abjad> tuplettools.fuse_tuplets(staff[:])
FixedDurationTuplet(3/8, [c'8, d'8, e'8, c'16, d'16, e'16])

```

```

abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8 [
            d'8
            e'8 ]
        c'16 (
            d'16
            e'16 )
    }
}

```

Return new tuplet.

Fuse zero or more parent-contiguous *tuplets*.

Allow in-score *tuplets*.

Allow outside-of-score *tuplets*.

All *tuplets* must carry the same multiplier.

All *tuplets* must be of the same type. Changed in version 2.0: renamed `fuse.tuplets_by_reference()` to `tuplettools.fuse_tuplets()`.

tuplettools.get_first_tuplet_in_improper_parentage_of_component

`abjad.tools.tuplettools.get_first_tuplet_in_improper_parentage_of_component.get_first_tuplet`

New in version 2.0. Get first tuplet in improper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])

abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8

```

```

        d'8
        e'8
    }
    f'8
}

```

```

abjad> tuplettools.get_first_tuplet_in_improper_parentage_of_component(staff.leaves[1])
Tuplet(2/3, [c'8, d'8, e'8])

```

Return tuplet or none.

tuplettools.get_first_tuplet_in_proper_parentage_of_component

`abjad.tools.tuplettools.get_first_tuplet_in_proper_parentage_of_component.get_first_tuplet_in_proper_parentage_of_component`

New in version 2.0. Get first tuplet in proper parentage of *component*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])

```

```

abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    f'8
}

```

```

abjad> tuplettools.get_first_tuplet_in_proper_parentage_of_component(staff.leaves[1])
Tuplet(2/3, [c'8, d'8, e'8])

```

Return tuplet or none.

tuplettools.is_proper_tuplet_multiplier

`abjad.tools.tuplettools.is_proper_tuplet_multiplier.is_proper_tuplet_multiplier` (*multiplier*)

True when $1/2 < \text{multiplier} < 2$.

```

abjad> for n in range(17):
...     rational = Fraction(n, 8)
...     multiplier = tuplettools.is_proper_tuplet_multiplier(rational)
...     print '%s    %s' % (rational, multiplier)
...
0        False
1/8      False
1/4      False
3/8      False
1/2      False
5/8      True
3/4      True
7/8      True
1        True
9/8      True
5/4      True

```

```

11/8    True
3/2     True
13/8    True
7/4     True
15/8    True
2       False

```

This function models the idea that 4:3, 4:5, 4:6, 4:7 are valid tuplet multipliers while 4:2 and 4:8 aren't. Changed in version 2.0: renamed `durationtools.is_tuplet_multiplier()` to `tuplettools.is_proper_tuplet_multiplier()`.

`tuplettools.iterate_tuplets_backward_in_expr`

`abjad.tools.tuplettools.iterate_tuplets_backward_in_expr`.**`iterate_tuplets_backward_in_expr`** (*exp*

New in version 2.0. Iterate tuplets backward in *expr*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])
abjad> Tuplet(Fraction(2, 3), staff[-3:])
Tuplet(2/3, [a'8, b'8, c''8])

```

```

abjad> f(staff)
\new Staff {
    \times 2/3 {
        c'8
        d'8
        e'8
    }
    f'8
    g'8
    \times 2/3 {
        a'8
        b'8
        c''8
    }
}

```

```

abjad> for tuplet in tuplettools.iterate_tuplets_backward_in_expr(staff):
...     tuplet
...
Tuplet(2/3, [a'8, b'8, c''8])
Tuplet(2/3, [c'8, d'8, e'8])

```

Return generator.

`tuplettools.iterate_tuplets_forward_in_expr`

`abjad.tools.tuplettools.iterate_tuplets_forward_in_expr`.**`iterate_tuplets_forward_in_expr`** (*exp*
start
stop

New in version 2.0. Iterate tuplets forward in *expr*:

```

abjad> staff = Staff("c'8 d'8 e'8 f'8 g'8 a'8 b'8 c''8")
abjad> Tuplet(Fraction(2, 3), staff[:3])
Tuplet(2/3, [c'8, d'8, e'8])
abjad> Tuplet(Fraction(2, 3), staff[-3:])
Tuplet(2/3, [a'8, b'8, c''8])

abjad> f(staff)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  f'8
  g'8
  \times 2/3 {
    a'8
    b'8
    c''8
  }
}

abjad> for tuplet in tuplettools.iterate_tuplets_forward_in_expr(staff):
...     tuplet
...
Tuplet(2/3, [c'8, d'8, e'8])
Tuplet(2/3, [a'8, b'8, c''8])

```

Return generator.

tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots

```
abjad.tools.tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots
```

New in version 2.0. Make augmented tuplet from *duration* and *proportions* and avoid dots.

Return tupletted leaves strictly without dots when all *proportions* equal 1:

```

abjad> print tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots(
... Fraction(3, 16), [1, 1, 1, -1, -1])
{@ 5:6 c'32, c'32, c'32, r32, r32 @}

```

Allow tupletted leaves to return with dots when some *proportions* do not equal 1:

```

abjad> print tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots(
... Fraction(3, 16), [1, -2, -2, 3, 3])
{@ 11:12 c'64, r32, r32, c'32., c'32. @}

```

Interpret nonassignable *proportions* according to *direction*:

```

abjad> print tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots(
... Fraction(3, 16), [5, -1, 5], direction = 'little-endian')
{@ 11:12 c'64, c'16, r64, c'64, c'16 @}

```

Reduce *proportions* relative to each other.

Interpret negative *proportions* as rests.

Return	fixed-duration	tuplet.	Changed	in	version	2.0:	renamed
							to

```
divide.duration_into_arbitrary_augmentation_undotted()
tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots().
```

tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots

```
abjad.tools.tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots
```

New in version 2.0. Make augmented tuplet from *duration* and *proportions* and encourage dots:

```
abjad> print tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots(
... Fraction(3, 16), [1, 1, 1, -1, -1])
{@ 5:8 c'64., c'64., c'64., r64., r64. @}
```

Interpret nonassignable *proportions* according to *direction*:

```
abjad> print tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots(
... Fraction(3, 16), [5, -1, 5], direction = 'little-endian')
{@ 11:16 c'32..., r128., c'32... @}
```

Reduce *proportions* relative to each other.

Interpret negative *proportions* as rests.

Return	fixed-duration	tuplet.	Changed	in	version	2.0:	renamed
							to

```
divide.duration_into_arbitrary_augmentation_dotted()
tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots().
```

tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots

```
abjad.tools.tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots
```

New in version 2.0. Make diminished tuplet from *duration* and nonzero integer *proportions*.

Return tupletted leaves strictly without dots when all *proportions* equal 1:

```
abjad> print tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots(
... Fraction(3, 16), [1, 1, 1, -1, -1])
{@ 5:3 c'16, c'16, c'16, r16, r16 @}
```

Allow tupletted leaves to return with dots when some *proportions* do not equal 1:

```
abjad> print tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots(
... Fraction(3, 16), [1, -2, -2, 3, 3])
{@ 11:6 c'32, r16, r16, c'16., c'16. @}
```



```

abjad> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4])

abjad> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16])

abjad> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1, 2], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16, c'8])

abjad> tuplettools.make_tuplet_from_proportions_and_pair([1, 2, 4, 1, 2, 4], (7, 16))
FixedDurationTuplet(7/16, [c'16, c'8, c'4, c'16, c'8, c'4])

```

Note: function accepts a pair rather than a rational.

Note: function interprets d as tuplet denominator.

Changed in version 2.0: renamed `divide.pair()` to `tuplettools.make_tuplet_from_proportions_and_pair`

`tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet`

`abjad.tools.tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet.m`

Scale tuplet contents and then bequeath in-score position of tuplet to contents.

Return orphaned tuplet emptied of all contents.

```

abjad> t = Staff(tuplettools.FixedDurationTuplet(Duration(3, 8), "c'8 d'8") * 2)
abjad> spannertools.BeamSpanner(t.leaves)
BeamSpanner(c'8, d'8, c'8, d'8)
abjad> print t.format
\new Staff {
  \fraction \times 3/2 {
    c'8 [
      d'8
    ]
  }
  \fraction \times 3/2 {
    c'8
    d'8 ]
  }
}

abjad> tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet(t[0])
FixedDurationTuplet(3/8, [])
abjad> print t.format
\new Staff {
  c'8. [
    d'8.
    \fraction \times 3/2 {
      c'8
      d'8 ]
    }
}

```

Changed in version 2.0: renamed `tuplettools.subsume()` to `tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet()`.

`tuplettools.remove_trivial_tuplets_in_expr`

`abjad.tools.tuplettools.remove_trivial_tuplets_in_expr.remove_trivial_tuplets_in_expr(expr)`

Remove trivial tuplets in *expr*:

```
abjad> t = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8 e'8")
abjad> u = tuplettools.FixedDurationTuplet(Duration(1, 4), "c'8 d'8")
abjad> s = Staff([t, u])
abjad> len(s)
2
```

```
abjad> s[0]
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
abjad> s[1]
FixedDurationTuplet(1/4, [c'8, d'8])
```

```
abjad> tuplettools.remove_trivial_tuplets_in_expr(s)
abjad> len(s)
3
```

```
abjad> s[0]
FixedDurationTuplet(1/4, [c'8, d'8, e'8])
abjad> s[1]
Note("c'8")
abjad> s[2]
Note("d'8")
```

```
abjad> f(s)
\new Staff {
  \times 2/3 {
    c'8
    d'8
    e'8
  }
  c'8
  d'8
}
```

Replace trivial tuplets with plain leaves.

Return `none`. Changed in version 2.0: renamed `tuplettools.slip_trivial()` to `tuplettools.remove_trivial_tuplets_in_expr()`.

`tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier`

`abjad.tools.tuplettools.scale_contents_of_tuplets_in_expr_by_multiplier.scale_contents_of_tuplets_in_expr_by_multiplier(expr, multiplier)`

Scale fixed-duration tuplet by multiplier. Preserve tuplet multiplier. Return tuplet.

`tuplettools.set_denominator_of_tuplets_in_expr_to_at_least`

`abjad.tools.tuplettools.set_denominator_of_tuplets_in_expr_to_at_least.set_denominator_of_tuplets_in_expr_to_at_least(expr, n)`

New in version 2.0. Set denominator of tuplets in *expr* to at least *n*:

```
abjad> tuplet = Tuplet(Fraction(3, 5), "c'4 d'8 e'8 f'4 g'2")

abjad> f(tuplet)
\fraction \times 3/5 {
    c'4
    d'8
    e'8
    f'4
    g'2
}

abjad> tuplettools.set_denominator_of_tuplets_in_expr_to_at_least(tuplet, 8)

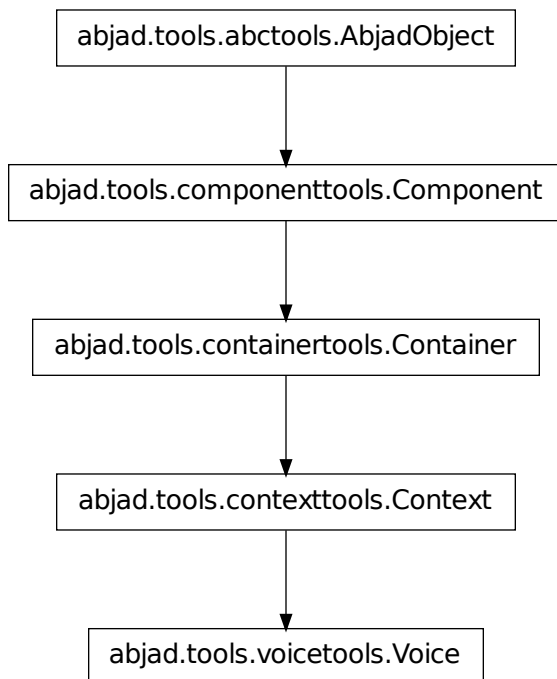
abjad> f(tuplet)
\fraction \times 6/10 {
    c'4
    d'8
    e'8
    f'4
    g'2
}
```

Return none.

voicetools

voicetools concrete classes

voicetools.Voice



class `abjad.tools.voicetools.Voice.Voice.Voice` (*music=None*, ***kwargs*)

Abjad model of a voice:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> f(voice)
\new Voice {
  c'8
  d'8
  e'8
  f'8
}
```

Return voice object.

Read-only Properties

`Voice.contents_duration`

Note: Inherited from `containertools.Container`

Voice.duration_in_seconds

Note: Inherited from `containertools.Container`

Voice.engraver_consists

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {
    \consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

Voice.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
    \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contextttools.Context`

Voice.format

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

Voice.is_semantic

Note: Inherited from `contextttools.Context`

Voice.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Voice.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Voice.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Voice.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Voice.parent

Note: Inherited from `componenttools.Component`

Voice.preprolated_duration

Note: Inherited from `containertools.Container`

Voice.prolated_duration

Note: Inherited from `componenttools.Component`

Voice.prolation

Note: Inherited from `componenttools.Component`

Voice.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`Voice.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`Voice.context_name`

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

`Voice.is_nonsemantic`

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'
```

```
abjad> voice.is_nonsemantic = True
```

```
abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}
```

```
abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice interaction and other functions.

Note: Inherited from `contexttools.Context`

Voice.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
}

abjad> container.is_parallel
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True

abjad> f(container)
<<
    \new Voice {
        c'8
        d'8
        e'8
    }
    \new Voice {
        g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Voice.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

Voice.**append**(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
```



```
        c'8 [
        d'8
        e'8 ]
    }

    abjad> container.append(Note("f'8"))

    abjad> f(container)
    {
        c'8 [
        d'8
        e'8 ]
        f'8
    }
```

Return none.

Note: Inherited from `containertools.Container`

Voice.**extend**(*expr*)

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

Voice.**index**(*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")
```

```
abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

Voice.**insert** (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Voice.**pop** (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

Voice.**remove**(*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

Voice.**__add__**(*expr*)

Concatenate containers self and *expr*. The operation $c = a + b$ returns a new Container *c* with the content of both *a* and *b*. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

Voice.**__contains__**(*expr*)

True if *expr* is in container, otherwise False.

Note: Inherited from `containertools.Container`

Voice.**__delattr__**()

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Voice.**__delitem__**(*i*)

Find component(s) at index or slice '*i*' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Voice.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Voice.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Voice.__getitem__(i)`
Return component at index `i` in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Voice.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`Voice.__hash__()` $\leq \Rightarrow hash(x)$

Note: Inherited from `__builtin__.object`

`Voice.__iadd__(expr)`
`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Voice.__imul__(total)`
Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Voice.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

Voice.__len__()

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

Voice.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Voice.__mul__(n)

Note: Inherited from `componenttools.Component`

Voice.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Voice.__radd__(expr)

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

Voice.__repr__()

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

Voice.__rmul__(n)

Note: Inherited from `componenttools.Component`

Voice.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

Voice.__setitem__(i, expr)

Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`Voice.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

voicetools functions

voicetools.all_are_voices

`abjad.tools.vocetools.all_are_voices.all_are_voices(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad voices:

```
abjad> voice = Voice("c'4 d'4 e'4 f'4")

abjad> voicetools.all_are_voices([voice])
True
```

True when *expr* is an empty sequence:

```
abjad> voicetools.all_are_voices([])
True
```

Otherwise false:

```
abjad> voicetools.all_are_voices('foo')
False
```

Return boolean.

Function wraps `componenttools.all_are_components()`.

voicetools.get_first_voice_in_improper_parentage_of_component

`abjad.tools.vocetools.get_first_voice_in_improper_parentage_of_component.get_first_voice_in_improper_parentage_of_component(component)`

New in version 2.0. Get first voice in improper parentage of *component*:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> staff = Staff([voice])

abjad> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
}

abjad> voicetools.get_first_voice_in_improper_parentage_of_component(staff.leaves[0])
Voice{4}
```

Return voice or none.

voicetools.get_first_voice_in_proper_parentage_of_component

`abjad.tools.voicetools.get_first_voice_in_proper_parentage_of_component.get_first_voice_in`

New in version 2.0. Get first voice in proper parentage of *component*:

```
abjad> voice = Voice("c'8 d'8 e'8 f'8")
abjad> staff = Staff([voice])

abjad> f(staff)
\new Staff {
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
}

abjad> voicetools.get_first_voice_in_proper_parentage_of_component(staff.leaves[0])
Voice{4}
```

Return voice or none.

voicetools.iterate_semantic_voices_backward_in_expr

`abjad.tools.voicetools.iterate_semantic_voices_backward_in_expr.iterate_semantic_voices_backward_in_expr`

New in version 2.0. Iterate semantic voices backward in *expr*:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(3, 8), (5, 16), (5, 16)])
abjad> time_signature_voice = Voice(measures)
abjad> time_signature_voice.name = 'TimeSignatureVoice'
abjad> time_signature_voice.is_nonsemantic = True
abjad> music_voice = Voice("c'4. d'4 e'16 f'4 g'16")
abjad> music_voice.name = 'MusicVoice'
abjad> staff = Staff([time_signature_voice, music_voice])
abjad> staff.is_parallel = True

abjad> f(staff)
\new Staff <<
  \context Voice = "TimeSignatureVoice" {
    {
      \time 3/8
      s1 * 3/8
    }
    {
      \time 5/16
      s1 * 5/16
    }
    {
      \time 5/16
      s1 * 5/16
    }
  }
  \context Voice = "MusicVoice" {
    c'4.
    d'4
    e'16
  }
}
```

```

        f'4
        g'16
    }
>>

```

```

abjad> for voice in voicetools.iterate_semantic_voices_backward_in_expr(staff):
...     voice
Voice-"MusicVoice"{5}

```

Return generator.

voicetools.iterate_semantic_voices_forward_in_expr

`abjad.tools.voicetools.iterate_semantic_voices_forward_in_expr`. **iterate_semantic_voices_forward_in_expr**
 New in version 2.0. Iterate semantic voices forward in *expr*:

```

abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(3, 8), (5, 16), (5, 16)])
abjad> meter_voice = Voice(measures)
abjad> meter_voice.name = 'TimeSignatuerVoice'
abjad> meter_voice.is_nonsemantic = True
abjad> music_voice = Voice("c'4. d'4 e'16 f'4 g'16")
abjad> music_voice.name = 'MusicVoice'
abjad> staff = Staff([meter_voice, music_voice])
abjad> staff.is_parallel = True

```

```

abjad> f(staff)
\new Staff <<
  \context Voice = "TimeSignatuerVoice" {
    {
      \time 3/8
      s1 * 3/8
    }
    {
      \time 5/16
      s1 * 5/16
    }
    {
      \time 5/16
      s1 * 5/16
    }
  }
  \context Voice = "MusicVoice" {
    c'4.
    d'4
    e'16
    f'4
    g'16
  }
>>

```

```

abjad> for voice in voicetools.iterate_semantic_voices_forward_in_expr(staff):
...     voice
Voice-"MusicVoice"{5}

```

Return generator.

voicetools.iterate_voices_backward_in_expr

`abjad.tools.voicetools.iterate_voices_backward_in_expr.iterate_voices_backward_in_expr(expr)`

New in version 2.0. Iterate voices backward in *expr*:

```
abjad> voice_1 = Voice("c'8 d'8 e'8 f'8")
abjad> voice_2 = Voice("c'4 b4")
abjad> staff = Staff([voice_1, voice_2])
abjad> staff.is_parallel = True
```

```
abjad> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
  \new Voice {
    c'4
    b4
  }
>>
```

```
abjad> for voice in voicetools.iterate_voices_backward_in_expr(staff):
...     voice
Voice{2}
Voice{4}
```

Return generator.

voicetools.iterate_voices_forward_in_expr

`abjad.tools.voicetools.iterate_voices_forward_in_expr.iterate_voices_forward_in_expr(expr)`

New in version 2.0. Iterate voices forward in *expr*:

```
abjad> voice_1 = Voice("c'8 d'8 e'8 f'8")
abjad> voice_2 = Voice("c'4 b4")
abjad> staff = Staff([voice_1, voice_2])
abjad> staff.is_parallel = True
```

```
abjad> f(staff)
\new Staff <<
  \new Voice {
    c'8
    d'8
    e'8
    f'8
  }
  \new Voice {
    c'4
    b4
  }
>>
```

```
abjad> for voice in voicetools.iterate_voices_forward_in_expr(staff):
...     voice
```

```
Voice{4}
Voice{2}
```

Return generator.

51.1.2 Additional Abjad composition packages (load manually)

abctools

abctools abstract classes

abctools.AbjadObject

`abjad.tools.abctools.AbjadObject`

class `abjad.tools.abctools.AbjadObject.AbjadObject` (**args, **kwargs*)
 New in version 2.0. Abstract base class from which all custom classes should inherit.

Abjad objects raise exceptions on `__gt__`, `__ge__`, `__lt__`, `__le__`.

Abjad objects compare equal only with equal object IDs.

Authors of custom classes should override these behaviors as required.

Special Methods

`AbjadObject.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`AbjadObject.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

`AbjadObject.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

`AbjadObject.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

`AbjadObject.__hash__()` <==> `hash(x)`

Note: Inherited from `__builtin__.object`

`AbjadObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

`AbjadObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

`AbjadObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

`AbjadObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

`AbjadObject.__setattr__()`

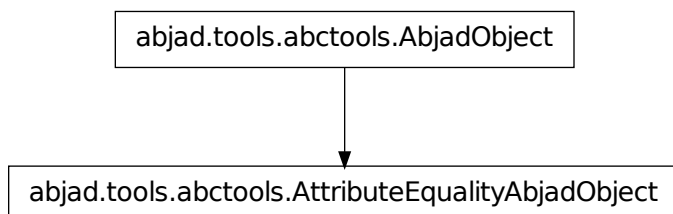
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`AbjadObject.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

abctools.AttributeEqualityAbjadObject



class `abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject`. **Attribut**

New in version 2.0. Abstract base class to confer nonsorting attribute-equality to any custom class.

Nonsorting objects raise exceptions on `__gt__`, `__ge__`, `__lt__`, `__le__`.

Attribute-equality objects compare equal only with equal comparison attributes.

Special Methods

AttributeEqualityAbjadObject.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

AttributeEqualityAbjadObject.**__eq__**(arg)
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

AttributeEqualityAbjadObject.**__ge__**(arg)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AttributeEqualityAbjadObject.**__gt__**(arg)
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

AttributeEqualityAbjadObject.**__hash__**() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

AttributeEqualityAbjadObject.**__le__**(arg)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AttributeEqualityAbjadObject.**__lt__**(arg)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AttributeEqualityAbjadObject.**__ne__**(arg)
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

AttributeEqualityAbjadObject.**__repr__**()
Interpreter representation of object defined equal to class name and format string.

Return string.

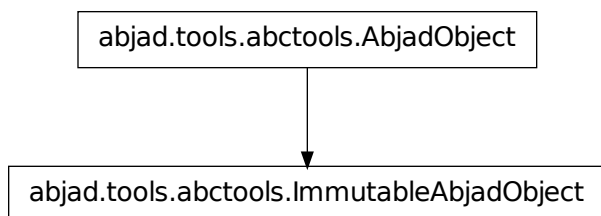
```
AttributeEqualityAbjadObject.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
AttributeEqualityAbjadObject.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

abctools.ImmutableAbjadObject



```
class abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject(*arg  
**k
```

New in version 2.8. Abstract base class from which all custom classes which also subclass immutable builtin classes, such as tuple and frozenset, should inherit.

Special Methods

```
ImmutableAbjadObject.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
ImmutableAbjadObject.__eq__(arg)  
True when id(self) equals id(arg).  
Return boolean.
```

Note: Inherited from `abctools.AbjadObject`

```
ImmutableAbjadObject.__ge__(arg)  
Abjad objects by default do not implement this method.  
Raise exception.
```

Note: Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`ImmutableAbjadObject.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ImmutableAbjadObject.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

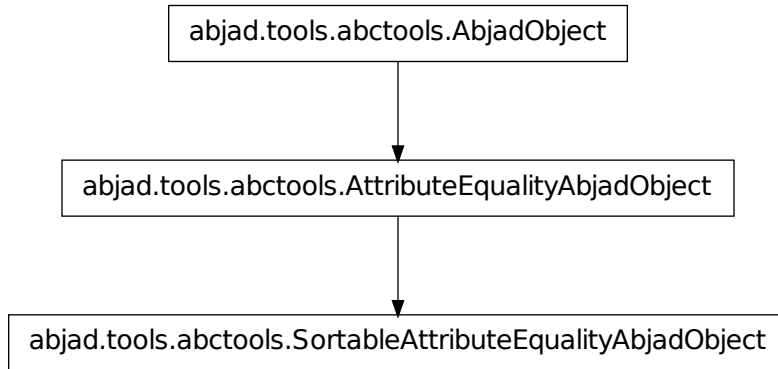
`ImmutableAbjadObject.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ImmutableAbjadObject.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

abctools.SortableAttributeEqualityAbjadObject

class `abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject`

New in version 2.0. Abstract base class to confer sortable attribute-equality to any custom class.

Sortable attribute-equality comparators implement `__eq__`, `__ne__`, `__gt__`, `__ge__`, `__lt__`, `__le__` against a single comparison attribute.

Special Methods

`SortableAttributeEqualityAbjadObject.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SortableAttributeEqualityAbjadObject.__eq__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`SortableAttributeEqualityAbjadObject.__ge__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__gt__(arg)`
 Initialize new object from *arg* and evaluate comparison attributes.
 Return boolean.

`SortableAttributeEqualityAbjadObject.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

`SortableAttributeEqualityAbjadObject.__le__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

`SortableAttributeEqualityAbjadObject.__lt__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

`SortableAttributeEqualityAbjadObject.__ne__(arg)`
Initialize new object from *arg* and evaluate comparison attributes.

Return boolean.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`SortableAttributeEqualityAbjadObject.__repr__()`
Interpreter representation of object defined equal to class name and format string.

Return string.

Note: Inherited from `abctools.AttributeEqualityAbjadObject`

`SortableAttributeEqualityAbjadObject.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SortableAttributeEqualityAbjadObject.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

configurationtools

configurationtools functions

configurationtools.get_abjad_revision_string

`abjad.tools.configurationtools.get_abjad_revision_string.get_abjad_revision_string()`
New in version 2.0. Get Abjad revision string:

```
abjad> configurationtools.get_abjad_revision_string() # doctest: +SKIP
'5280'
```

Return string.

configurationtools.get_abjad_version_string

abjad.tools.configurationtools.get_abjad_version_string.get_abjad_version_string()

New in version 2.0. Get Abjad version string:

```
abjad> from abjad.tools import configurationtools

abjad> configurationtools.get_abjad_version_string()
'2.8'
```

Return string.

configurationtools.get_lilypond_version_string

abjad.tools.configurationtools.get_lilypond_version_string.get_lilypond_version_string()

New in version 2.0. Get LilyPond version string:

```
abjad> configurationtools.get_lilypond_version_string() # doctest: +SKIP
'2.13.61'
```

Return string.

configurationtools.get_python_version_string

abjad.tools.configurationtools.get_python_version_string.get_python_version_string()

New in version 2.0. Get Python version string:

```
abjad> from abjad.tools import configurationtools

abjad> configurationtools.get_python_version_string() # doctest: +SKIP
'2.6.1'
```

Return string.

configurationtools.list_abjad_environment_variables

abjad.tools.configurationtools.list_abjad_environment_variables.list_abjad_environment_variables()

New in version 1.1. List Abjad environment variables.

Return tuple of zero or more environment variable / setting pairs.

Abjad environment variables are defined in abjad/cfg/cfg.py. Changed in version 2.0: renamed configurationtools.list_settings() to configurationtools.list_abjad_environment_variables().

configurationtools.list_abjad_templates

abjad.tools.configurationtools.list_abjad_templates.list_abjad_templates()

New in version 2.0. List Abjad templates:

```
abjad> from abjad.tools import configurationtools

abjad> configurationtools.list_abjad_templates()
('coventry.ly', 'lagos.ly', 'oedo.ly', 'paris.ly', 'tangiers.ly', 'thebes.ly', 'tirnaveni.ly')
```

Return tuple of zero or more strings.

Abjad templates are housed in `abjad/templates`.

configurationtools.list_package_dependency_versions

`abjad.tools.configurationtools.list_package_dependency_versions.list_package_dependency_versions`

configurationtools.set_default_accidental_spelling

`abjad.tools.configurationtools.set_default_accidental_spelling.set_default_accidental_spelling`

New in version 1.1. Set default accidental spelling to sharps:

```
abjad> from abjad.tools import configurationtools

abjad> configurationtools.set_default_accidental_spelling('sharps')

abjad> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("cs''4"), Note("ds''4")]
```

Set default accidental spelling to flats:

```
abjad> configurationtools.set_default_accidental_spelling('flats')

abjad> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("df''4"), Note("ef''4")]
```

Set default accidental spelling to mixed:

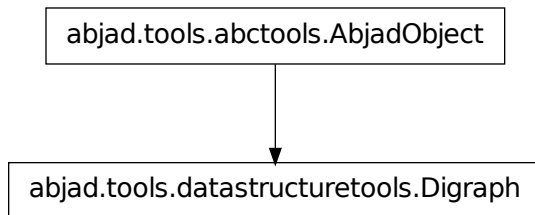
```
abjad> configurationtools.set_default_accidental_spelling()

abjad> [Note(13, (1, 4)), Note(15, (1, 4))]
[Note("cs''4"), Note("ef''4")]
```

Mixed is system default.

Mixed test case must appear last here for doc tests to check correctly.

Return none. Changed in version 2.0: renamed `pitchtools.change_default_accidental_spelling()` to `configurationtools.set_default_accidental_spelling()`.

datastructuretools**datastructuretools concrete classes****datastructuretools.Digraph**

class `abjad.tools.datastructuretools.Digraph.Digraph`(*edges=None*)

A digraph, built out of edges - pairs of hashable objects:

```
abjad> from abjad.tools.datastructuretools import Digraph
```

```
abjad> edges = [('a', 'b'), ('a', 'c'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')]
abjad> digraph = Digraph(edges)
abjad> digraph
Digraph(edges=[('a', 'c'), ('a', 'b'), ('a', 'f'), ('c', 'd'), ('d', 'e'), ('e', 'c')])
```

```
abjad> digraph.root_nodes
('a',)
abjad> digraph.terminal_nodes
('b', 'f')
abjad> digraph.cyclic_nodes
('c', 'd', 'e')
abjad> digraph.is_cyclic
True
```

Return *Digraph* instance.

Read-only Properties**`Digraph.child_graph`**

A dictionary representation of the digraph where the keys are child nodes, and where each value is the set of that child's parents.

`Digraph.cyclic_nodes`

A tuple of those nodes which partake in a cycle.

`Digraph.edges`

A tuple of all edges in the graph.

`Digraph.is_cyclic`

Return True if the digraph contains any cycles.

`Digraph.nodes`

A tuple of all nodes in the graph.

Digraph.**parent_graph**

A dictionary representation of the digraph where the keys are parent nodes, and where each value is the set of that parent's children.

Digraph.**root_nodes**

A tuple of those nodes which have no parents.

Digraph.**terminal_nodes**

A tuple of those nodes which have no children.

Methods

Digraph.**partition()**

Partition the digraph into a list of digraphs according to connectivity:

```
abjad> from abjad.tools.datastructuretools import Digraph

abjad> edges = [('a', 'b'), ('a', 'c'), ('b', 'c'), ('b', 'd'), ('d', 'e')]
abjad> edges.extend([('f', 'h'), ('g', 'h')])
abjad> edges.append(('i', 'j'))
abjad> digraph = Digraph(edges)
abjad> for graph in digraph.partition(): graph
...
Digraph(edges=[('a', 'c'), ('a', 'b'), ('b', 'c'), ('b', 'd'), ('d', 'e')])
Digraph(edges=[('f', 'h'), ('g', 'h')])
Digraph(edges=[('i', 'j')])
```

Return list of *Digraph* instances.

Digraph.**reverse()**

Reverse all edges in the graph:

```
abjad> from abjad.tools.datastructuretools import Digraph

abjad> edges = [('a', 'b'), ('b', 'c'), ('b', 'd')]
abjad> digraph = Digraph(edges)
abjad> digraph
Digraph(edges=[('a', 'b'), ('b', 'c'), ('b', 'd')])

abjad> digraph.reverse()
Digraph(edges=[('b', 'a'), ('c', 'b'), ('d', 'b')])
```

Return *Digraph* instance.

Special Methods

Digraph.**__delattr__()**

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Digraph.**__eq__**(*other*)

Digraph.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Digraph.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Digraph.__hash__() \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

Digraph.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Digraph.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Digraph.__ne__(other)

Digraph.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

Digraph.__setattr__()

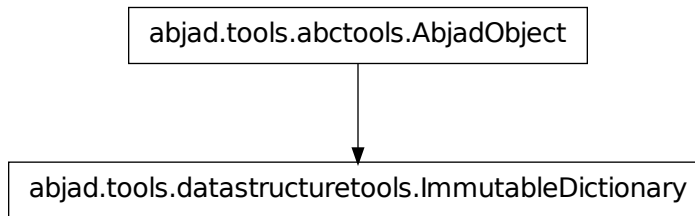
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

Digraph.__str__() \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

datastructuretools.ImmutableDictionary



class `abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary`
 New in version 2.0. Immutable dictionary:

```

abjad> from abjad.tools import datastructuretools

abjad> dictionary = datastructuretools.ImmutableDictionary({'color': 'red', 'number': 9})

abjad> dictionary
{'color': 'red', 'number': 9}

abjad> dictionary['color']
'red'

abjad> dictionary.size = 'large' # doctest: +SKIP
AttributeError: ImmutableDictionary objects are immutable.

abjad> dictionary['size'] = 'large' # doctest: +SKIP
AttributeError: ImmutableDictionary objects are immutable.
```

Return immutable dictionary.

Methods

`ImmutableDictionary.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.update(E, **F)` → None. Update D from dict/iterable E and F.
If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`ImmutableDictionary.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__contains__(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ImmutableDictionary.__delitem__(*args)`

`ImmutableDictionary.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

`ImmutableDictionary.__repr__()` <==> `repr(x)`

Note: Inherited from `__builtin__.dict`

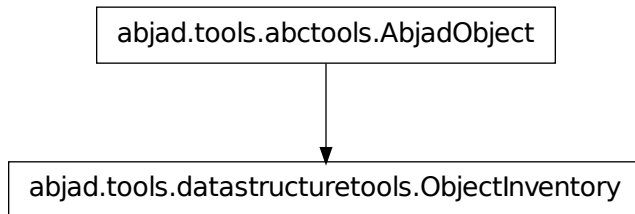
`ImmutableDictionary.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ImmutableDictionary.__setitem__(*args)`
`ImmutableDictionary.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`datastructuretools.ObjectInventory`



class `abjad.tools.datastructuretools.ObjectInventory.ObjectInventory` (`tokens=None`, `inventory_name=None`)

New in version 2.8. Ordered collection of custom objects.

Object inventories extend `append()`, `extend()` and `__contains__()` and allow token input.

Object inventories inherit from list and are mutable.

This class is an abstract base class that can not instantiate and should be subclassed.

Read/write Properties

`ObjectInventory.inventory_name`
Read / write name of inventory.

Methods

`ObjectInventory.append(token)`
Change *token* to item and append.

`ObjectInventory.count(value)` \rightarrow integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`ObjectInventory.extend(tokens)`
Change *tokens* to items and extend.

`ObjectInventory.index(value[, start[, stop]])` \rightarrow integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ObjectInventory.insert()`
`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`ObjectInventory.pop([index])` → item – remove and return item at index (default last).
 Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`ObjectInventory.remove()`
`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`ObjectInventory.reverse()`
`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`ObjectInventory.sort()`
`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`ObjectInventory.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__contains__(token)`

`ObjectInventory.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ObjectInventory.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`

Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ObjectInventory.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__getslice__()`
`x.__getslice__(i, j) <==> x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ObjectInventory.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.list`

`ObjectInventory.__len__()` $\iff \text{len}(x)$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__lt__()`
`x.__lt__(y)` $\iff x < y$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__mul__()`
`x.__mul__(n)` $\iff x * n$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__ne__()`
`x.__ne__(y)` $\iff x \neq y$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__repr__()`

`ObjectInventory.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`ObjectInventory.__rmul__()`
`x.__rmul__(n)` $\iff n * x$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__setattr__()`
`x.__setattr__('name', value)` $\iff x.name = value$

Note: Inherited from `__builtin__.object`

`ObjectInventory.__setitem__()`
`x.__setitem__(i, y)` $\iff x[i] = y$

Note: Inherited from `__builtin__.list`

`ObjectInventory.__setslice__()`
`x.__setslice__(i, j, y)` $\iff x[i:j] = y$
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`ObjectInventory.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

decoratortools

decoratortools functions

decoratortools.requires

`abjad.tools.decoratortools.requires.requires(*tests)`

New in version 2.6. Function decorator to require input parameter *tests*.

Example:

```
abjad> from abjad.tools import mathtools
abjad> from abjad.tools.decoratortools import requires

abjad> @requires(mathtools.is_nonnegative_integer, string) # doctest: +SKIP
abjad> def multiply_string(n, string): return n * string

abjad> multiply_string(2, 'bar') # doctest: +SKIP
'barbar'

abjad> multiply_string(2.5, 'bar') # doctest: +SKIP
...
AssertionError: is_nonnegative_integer(2.5) does not return true.
```

Decorator target is available like this:

```
abjad> multiply_string.func_closure[1].cell_contents # doctest: +SKIP
<function multiply_string at 0x104e512a8>
```

Decorator tests are available like this:

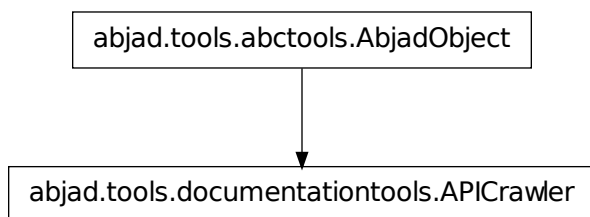
```
abjad> multiply_string.func_closure[0].cell_contents # doctest: +SKIP
(<function is_nonnegative_integer at 0x104725d70>, <type 'str'>)
```

Return decorated function in the form of function wrapper.

documentationtools

documentationtools concrete classes

documentationtools.APICrawler



```

class abjad.tools.documentationtools.APICrawler.APICrawler(code_root,
                                                            docs_root,
                                                            root_package_name,
                                                            ig-
                                                            nored_directories=['test',
                                                            '.svn',
                                                            '__py-  
cache__'],
                                                            pre-
                                                            fix='abjad.tools.')
```

Generates directories containing ReST to parallel directories containing code.

Read-only Properties

APICrawler.**code_root**

APICrawler.**docs_root**

APICrawler.**module_crawler**

APICrawler.**prefix**

Special Methods

APICrawler.**__call__**()

Crawl *code_root* and generate corresponding ReST in *docs_root* while ignoring ignored directories.

APICrawler.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

APICrawler.**__eq__**(*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`APICrawler.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`APICrawler.__setattr__()`

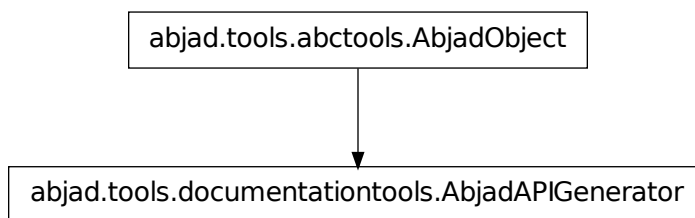
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`APICrawler.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.AbjadAPIGenerator



class `abjad.tools.documentationtools.AbjadAPIGenerator`, `AbjadAPIGenerator`, **AbjadAPIGenerator**
 Creates Abjad's API ReST:

- writes ReST pages for individual classes and functions
- writes the API index ReST
- handles sorting tools packages into composition, manual-loading and unstable
- handles ignoring private tools packages

Returns *AbjadAPIGenerator* instance.

Read-only Properties

`AbjadAPIGenerator.code_tools_path`

Path to Abjad tools package.

`AbjadAPIGenerator.docs_api_index_path`

Path to index.rst for Abjad API.

`AbjadAPIGenerator.docs_tools_path`

Path to tools directory inside docs.

Special Methods

`AbjadAPIGenerator.__call__(verbose=False)`

`AbjadAPIGenerator.__delattr__()`

`x.__delattr__('name')` \Leftrightarrow `del x.name`

Note: Inherited from `__builtin__.object`

AbjadAPIGenerator.__eq__(arg)
True when id(self) equals id(arg).
Return boolean.

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

AbjadAPIGenerator.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__ne__(arg)
True when id(self) does not equal id(arg).
Return boolean.

Note: Inherited from `abctools.AbjadObject`

AbjadAPIGenerator.__repr__()
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

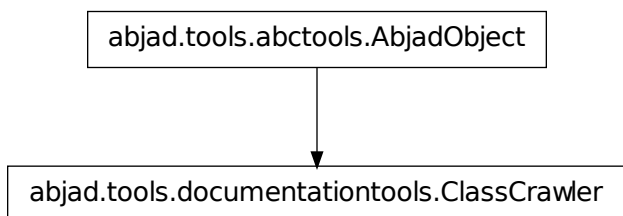
```
AbjadAPIGenerator.__setattr__ ()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
AbjadAPIGenerator.__str__ () <==> str(x)
```

Note: Inherited from `__builtin__.object`

documentationtools.ClassCrawler



```
class abjad.tools.documentationtools.ClassCrawler.ClassCrawler (code_root,
                                                                in-
                                                                clude_private_objects=F
                                                                root_package_name='ab
```

Read-only Properties

```
ClassCrawler.code_root
ClassCrawler.include_private_objects
ClassCrawler.module_crawler
ClassCrawler.root_package_name
```

Special Methods

```
ClassCrawler.__call__ ()
ClassCrawler.__delattr__ ()
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
ClassCrawler.__eq__ (arg)
True when id(self) equals id(arg).
```

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__hash__()` $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

`ClassCrawler.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ClassCrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

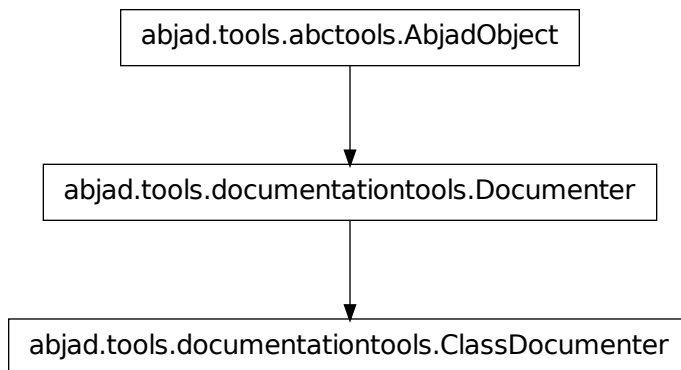
```
ClassCrawler.__setattr__()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
ClassCrawler.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

documentationtools.ClassDocumenter



```
class abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter(obj,
                                                                    pre-
                                                                    fix='abjad.
```

ClassDocumenter generates an ReST API entry for a given class:

```
abjad> from abjad import Note
abjad> from abjad.tools.documentationtools import ClassDocumenter

abjad> documenter = ClassDocumenter(Note)
abjad> rest = documenter()
```

Returns ClassDocumenter instance.

Read-only Properties

ClassDocumenter.**data**

ClassDocumenter.**inherited_attributes**

ClassDocumenter.**is_abstract**

ClassDocumenter.**methods**

`ClassDocumenter.module_name`

Note: Inherited from `documentationtools.Documenter`

`ClassDocumenter.object`

Note: Inherited from `documentationtools.Documenter`

`ClassDocumenter.prefix`

Note: Inherited from `documentationtools.Documenter`

`ClassDocumenter.readonly_properties`

`ClassDocumenter.readwrite_properties`

`ClassDocumenter.special_methods`

Special Methods

`ClassDocumenter.__call__()`

Generate documentation.

Returns string.

`ClassDocumenter.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ClassDocumenter.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`ClassDocumenter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`ClassDocumenter.__setattr__()`

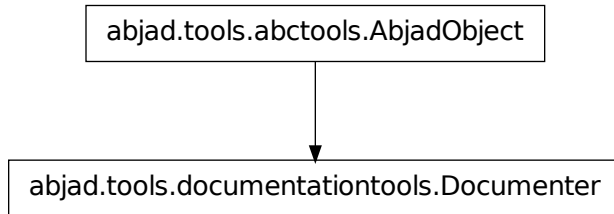
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`ClassDocumenter.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.Documenter



class `abjad.tools.documentationtools.Documenter.Documenter`(*obj*, *pre-fix*=*'abjad.tools.'*)

Documenter is an abstract base class for documentation classes.

Read-only Properties

`Documenter.module_name`

`Documenter.object`

`Documenter.prefix`

Special Methods

`Documenter.__call__()`

`Documenter.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Documenter.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Documenter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Documenter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Documenter.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`Documenter.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Documenter.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Documenter.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Documenter.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

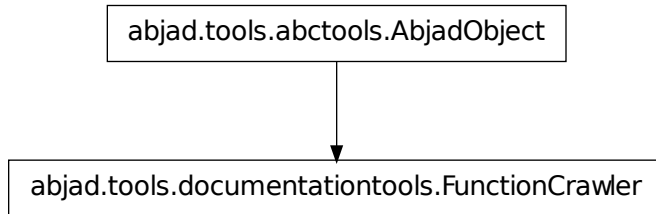
`Documenter.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`Documenter.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.FunctionCrawler



```
class abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler(code_root,
                                                                    in-
                                                                    clude_private
                                                                    root_package)
```

Read-only Properties

`FunctionCrawler.code_root`

`FunctionCrawler.include_private_objects`

`FunctionCrawler.module_crawler`

`FunctionCrawler.root_package_name`

Special Methods

`FunctionCrawler.__call__()`

`FunctionCrawler.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`FunctionCrawler.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`FunctionCrawler.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FunctionCrawler.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

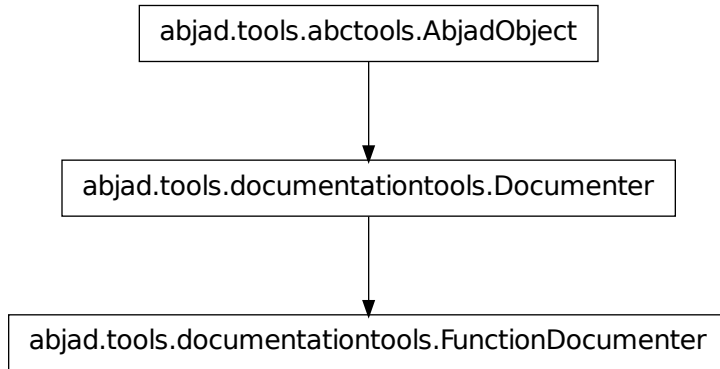
`FunctionCrawler.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`FunctionCrawler.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.FunctionDocumenter



```
class abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter
```

FunctionDocumenter generates an ReST entry for a given function:

```
abjad> from abjad.tools.documentationtools import *

abjad> from abjad.tools.notetools import make_notes
abjad> documenter = FunctionDocumenter(make_notes)
abjad> print documenter()
notetools.make_notes
=====
<BLANKLINE>
.. autofunction:: abjad.tools.notetools.make_notes.make_notes
<BLANKLINE>
```

Returns FunctionDocumenter `instance`.

Read-only Properties

FunctionDocumenter.**module_name**

Note: Inherited from `documentationtools.Documenter`

FunctionDocumenter.**object**

Note: Inherited from `documentationtools.Documenter`

FunctionDocumenter.**prefix**

Note: Inherited from `documentationtools.Documenter`

Special Methods

`FunctionDocumenter.__call__()`

Generate documentation.

Returns string.

`FunctionDocumenter.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`FunctionDocumenter.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`FunctionDocumenter.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FunctionDocumenter.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

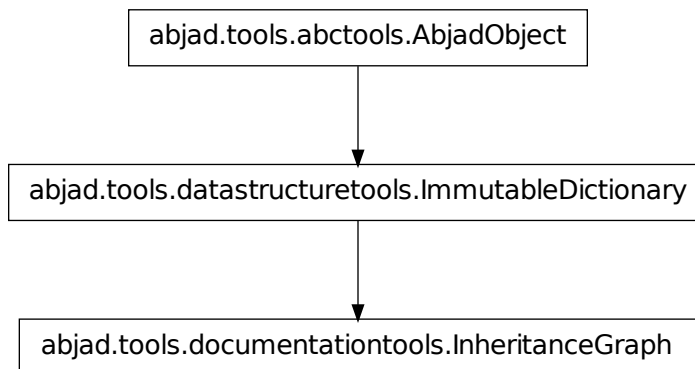
`FunctionDocumenter.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`FunctionDocumenter.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.InheritanceGraph



class `abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph(*args, **kwargs)`
Generates a graph of a class or collection of classes as a dictionary of parent:children relationships:

```

abjad> from abjad.tools.documentationtools import InheritanceGraph

abjad> class A(object): pass
...
abjad> class B(A): pass
...
abjad> class C(B): pass
...
abjad> class D(B): pass
...
abjad> class E(C, D): pass
...
abjad> class F(A): pass
...

abjad> graph = InheritanceGraph(F, E)

abjad> for parent, children in sorted(graph.items()):
...     parent, tuple(sorted(children))
(<type 'object'>, (<class '__main__.A'>,))
(<class '__main__.A'>, (<class '__main__.B'>, <class '__main__.F'>))
(<class '__main__.B'>, (<class '__main__.C'>, <class '__main__.D'>))
(<class '__main__.C'>, (<class '__main__.E'>,))
(<class '__main__.D'>, (<class '__main__.E'>,))
(<class '__main__.E'>, ())
(<class '__main__.F'>, ())

```

`InheritanceGraph` may be instantiated from one or more instances, classes or modules. If instantiated from a module, all public classes in that module will be taken into the graph.

A *root_class* keyword may be defined at instantiation, which filters out all classes from the graph which do not inherit from that *root_class* (or are not already the *root_class*):

```

abjad> graph = InheritanceGraph(A, B, C, D, E, F, root_class=B)
abjad> for parent, children in sorted(graph.items()):
...     parent, tuple(sorted(children))
(<class '__main__.B'>, (<class '__main__.C'>, <class '__main__.D'>))
(<class '__main__.C'>, (<class '__main__.E'>,))
(<class '__main__.D'>, (<class '__main__.E'>,))
(<class '__main__.E'>, ())

```

Returns `InheritanceGraph` instance.

Read-only Properties

`InheritanceGraph.root_class`

Methods

`InheritanceGraph.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.get(k[, d])` → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.has_key(k)` → `True` if `D` has a key `k`, else `False`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.items()` → list of `D`'s (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.iteritems()` → an iterator over the (key, value) items of `D`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.iterkeys()` → an iterator over the keys of `D`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.itervalues()` → an iterator over the values of `D`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.keys()` → list of `D`'s keys

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.pop(k[, d])` → `v`, remove specified key and return the corresponding value.
If key is not found, `d` is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if `D` is empty.

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.setdefault(k[, d])` → `D.get(k, d)`, also set `D[k]=d` if `k` not in `D`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.update(E, **F) → None`. Update D from dict/iterable E and F.

If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`InheritanceGraph.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__contains__(k) → True if D has a key k, else False`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InheritanceGraph.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`InheritanceGraph.__eq__()`

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__ge__()`

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__gt__()`

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__iter__()` $\leq \Rightarrow$ `iter(x)`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__le__()`

`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__len__()` $\leq \Rightarrow$ `len(x)`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__lt__()`

`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`InheritanceGraph.__ne__()`

`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

InheritanceGraph.__repr__() <==> repr(x)

Note: Inherited from `__builtin__.dict`

InheritanceGraph.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

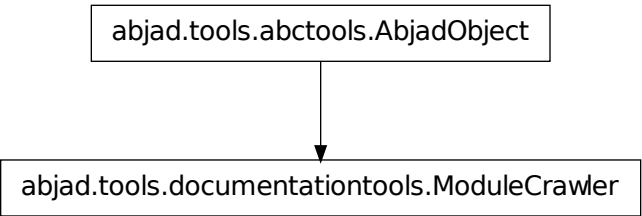
InheritanceGraph.__setitem__(*args)

Note: Inherited from `datastructuretools.ImmutableDictionary`

InheritanceGraph.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

documentationtools.ModuleCrawler



class `abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler`(*code_root*,
ignored_directories=
 `['svn',`
 `'__py-`
 `cache__']`,
 root_package_name)

Crawls *code_root*, yielding all module objects whose name begins with *root_package_name*.
Return *ModuleCrawler* instance.

Read-only Properties

`ModuleCrawler.code_root`
`ModuleCrawler.ignored_directories`

ModuleCrawler.**root_package_name**

Special Methods

ModuleCrawler.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

ModuleCrawler.**__eq__**(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

ModuleCrawler.**__ge__**(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

ModuleCrawler.**__gt__**(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

ModuleCrawler.**__hash__**() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

ModuleCrawler.**__iter__**()

ModuleCrawler.**__le__**(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

ModuleCrawler.**__lt__**(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

ModuleCrawler.**__ne__**(arg)
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ModuleCrawler.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`ModuleCrawler.__setattr__()`

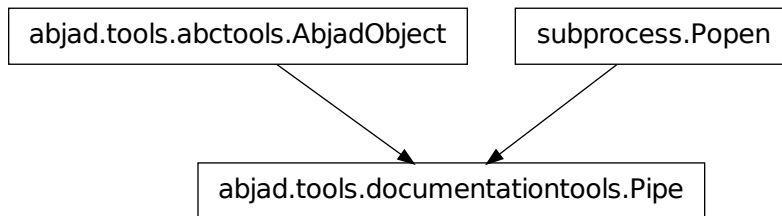
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ModuleCrawler.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

documentationtools.Pipe



class `abjad.tools.documentationtools.Pipe.Pipe` `(executable='python', arguments=['-i'], timeout=0)`

A two-way, non-blocking pipe for interprocess communication:

```
abjad> from abjad.tools.documentationtools import Pipe
```

```
abjad> pipe = Pipe('python', ['-i'])
abjad> pipe.writeline('my_list = [1, 2, 3]')
abjad> pipe.writeline('print my_list')
```

Return *Pipe* instance.

Read-only Properties

`Pipe.arguments`

`Pipe.executable`

`Pipe.timeout`

Methods

`Pipe.close()`

Close the pipe.

`Pipe.communicate(input=None)`

Interact with process: Send data to stdin. Read data from stdout and stderr, until end-of-file is reached. Wait for process to terminate. The optional input argument should be a string to be sent to the child process, or None, if no data should be sent to the child.

`communicate()` returns a tuple (stdout, stderr).

Note: Inherited from *subprocess.Popen*

`Pipe.kill()`

Kill the process with SIGKILL

Note: Inherited from *subprocess.Popen*

`Pipe.poll()`

Note: Inherited from *subprocess.Popen*

`Pipe.read(n=1)`

Read from the pipe.

`Pipe.send_signal(sig)`

Send a signal to the process

Note: Inherited from *subprocess.Popen*

`Pipe.terminate()`

Terminate the process with SIGTERM

Note: Inherited from *subprocess.Popen*

`Pipe.wait()`

Wait for child process to terminate. Returns returncode attribute.

Note: Inherited from *subprocess.Popen*

`Pipe.write(data)`

Write *data* into the pipe.

`Pipe.writeline(data)`

Write *data* into the pipe, terminated by a newline.

Special Methods

Pipe.**__del__** (*_maxint=2147483647, _active=[]*)

Note: Inherited from *subprocess.Popen*

Pipe.**__delattr__** ()
x.__delattr__('name') <==> *del x.name*

Note: Inherited from *__builtin__.object*

Pipe.**__eq__** (*arg*)
 True when *id(self)* equals *id(arg)*.
 Return boolean.

Note: Inherited from *abctools.AbjadObject*

Pipe.**__ge__** (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from *abctools.AbjadObject*

Pipe.**__gt__** (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from *abctools.AbjadObject*

Pipe.**__hash__** () <==> *hash(x)*

Note: Inherited from *__builtin__.object*

Pipe.**__le__** (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from *abctools.AbjadObject*

Pipe.**__lt__** (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from *abctools.AbjadObject*

Pipe.**__ne__**(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Pipe.**__repr__**()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

Pipe.**__setattr__**()

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

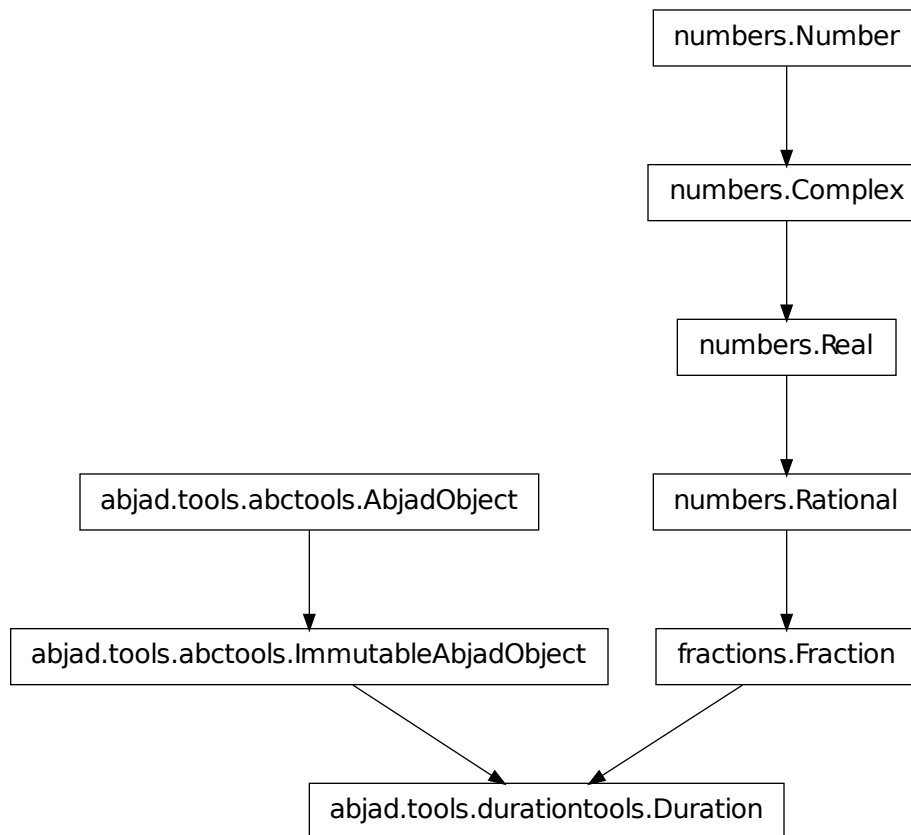
Pipe.**__str__**() <==> `str(x)`

Note: Inherited from `__builtin__.object`

`durationtools`

`durationtools` concrete classes

`durationtools.Duration`



class `abjad.tools.durationtools.Duration.Duration`.**Duration** (*args, **kwargs)

New in version 2.0. Abjad model of musical duration:

```

abjad> Duration(15, 16)
Duration(15, 16)

```

Durations inherit from built-in `Fraction`.

Read-only Properties

`Duration.denominator`

Note: Inherited from `fractions.Fraction`

`Duration.imag`

Real numbers have no imaginary component.

Note: Inherited from *numbers.Real*

`Duration.numerator`

Note: Inherited from *fractions.Fraction*

`Duration.real`

Real numbers are their real component.

Note: Inherited from *numbers.Real*

Methods

`Duration.conjugate()`

Conjugate is a no-op for Reals.

Note: Inherited from *numbers.Real*

`Duration.limit_denominator(max_denominator=1000000)`

Closest Fraction to self with denominator at most max_denominator.

```
>>> Fraction('3.141592653589793').limit_denominator(10)
Fraction(22, 7)
>>> Fraction('3.141592653589793').limit_denominator(100)
Fraction(311, 99)
>>> Fraction(4321, 8765).limit_denominator(10000)
Fraction(4321, 8765)
```

Note: Inherited from *fractions.Fraction*

Special Methods

`Duration.__abs__(*args)`

`Duration.__add__(*args)`

`Duration.__complex__()`

`complex(self) == complex(float(self), 0)`

Note: Inherited from *numbers.Real*

`Duration.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from *__builtin__.object*

`Duration.__div__(*args)`

Duration.__divmod__(*args)

Duration.__eq__(arg)

Duration.__float__()
float(self) = self.numerator / self.denominator

It's important that this conversion use the integer's "true" division rather than casting one side to float before dividing so that ratios of huge integers convert without overflowing.

Note: Inherited from *numbers.Rational*

Duration.__floordiv__(a, b)
a // b

Note: Inherited from *fractions.Fraction*

Duration.__ge__(arg)

Duration.__gt__(arg)

Duration.__hash__()
hash(self)

Tricky because values that are exactly representable as a float must have the same hash as that float.

Note: Inherited from *fractions.Fraction*

Duration.__le__(arg)

Duration.__lt__(arg)

Duration.__mod__(*args)

Duration.__mul__(*args)

Duration.__ne__(arg)

Duration.__neg__(*args)

Duration.__nonzero__(a)
a != 0

Note: Inherited from *fractions.Fraction*

Duration.__pos__(*args)

Duration.__pow__(*args)

Duration.__radd__(*args)

Duration.__rdiv__(*args)

Duration.__rdivmod__(*args)

Duration.__repr__()

Duration.__rfloordiv__(b, a)
a // b

Note: Inherited from *fractions.Fraction*

```
Duration.__rmod__(*args)
Duration.__rmul__(*args)
Duration.__rpow__(*args)
Duration.__rsub__(*args)
Duration.__rtruediv__(*args)
Duration.__setattr__()
    x.__setattr__('name', value) <==> x.name = value
```

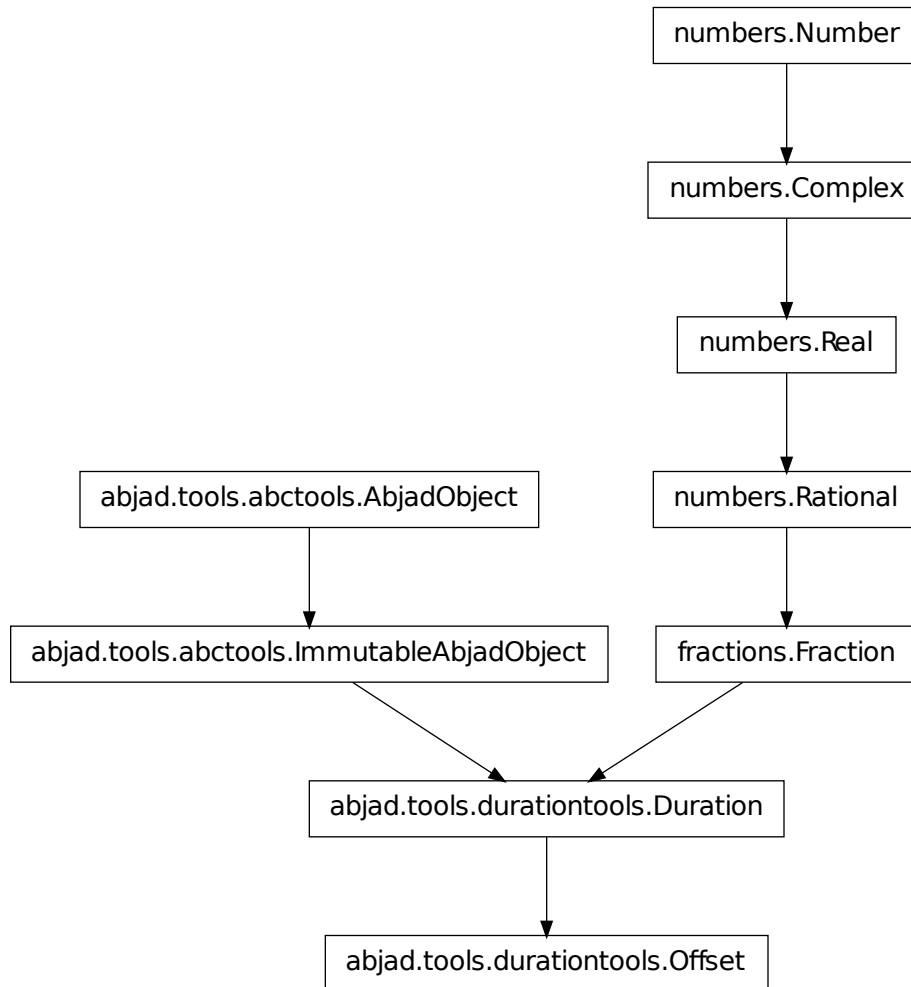
Note: Inherited from *__builtin__.object*

```
Duration.__str__()
    str(self)
```

Note: Inherited from *fractions.Fraction*

```
Duration.__sub__(*args)
Duration.__truediv__(*args)
Duration.__trunc__(a)
    trunc(a)
```

Note: Inherited from *fractions.Fraction*

durationtools.Offset

class `abjad.tools.durationtools.Offset.Offset` **Offset** (*args, **kwargs)

New in version 2.0. Abjad model of offset value of musical time:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.Offset(121, 16)
Offset(121, 16)
```

Offset inherits from duration (which inherits from built-in `Fraction`).

Read-only Properties

`Offset.denominator`

Note: Inherited from *fractions.Fraction*

Offset.**imag**

Real numbers have no imaginary component.

Note: Inherited from *numbers.Real*

Offset.**numerator**

Note: Inherited from *fractions.Fraction*

Offset.**real**

Real numbers are their real component.

Note: Inherited from *numbers.Real*

Methods

Offset.**conjugate**()

Conjugate is a no-op for Reals.

Note: Inherited from *numbers.Real*

Offset.**limit_denominator**(*max_denominator=1000000*)

Closest Fraction to self with denominator at most *max_denominator*.

```
>>> Fraction('3.141592653589793').limit_denominator(10)
Fraction(22, 7)
>>> Fraction('3.141592653589793').limit_denominator(100)
Fraction(311, 99)
>>> Fraction(4321, 8765).limit_denominator(10000)
Fraction(4321, 8765)
```

Note: Inherited from *fractions.Fraction*

Special Methods

Offset.**__abs__**(*args)

Note: Inherited from *durationtools.Duration*

Offset.**__add__**(*args)

Note: Inherited from *durationtools.Duration*

Offset.**__complex__**()

`complex(self) == complex(float(self), 0)`

Note: Inherited from *numbers.Real*

Offset.**__delattr__**()
x.__delattr__('name') <==> del x.name

Note: Inherited from *__builtin__.object*

Offset.**__div__**(*args)

Note: Inherited from *durationtools.Duration*

Offset.**__divmod__**(*args)

Note: Inherited from *durationtools.Duration*

Offset.**__eq__**(arg)

Note: Inherited from *durationtools.Duration*

Offset.**__float__**()
float(self) = self.numerator / self.denominator

It's important that this conversion use the integer's "true" division rather than casting one side to float before dividing so that ratios of huge integers convert without overflowing.

Note: Inherited from *numbers.Rational*

Offset.**__floordiv__**(a, b)
a // b

Note: Inherited from *fractions.Fraction*

Offset.**__ge__**(arg)

Note: Inherited from *durationtools.Duration*

Offset.**__gt__**(arg)

Note: Inherited from *durationtools.Duration*

Offset.**__hash__**()
hash(self)

Tricky because values that are exactly representable as a float must have the same hash as that float.

Note: Inherited from *fractions.Fraction*

`Offset.__le__(arg)`

Note: Inherited from `durationtools.Duration`

`Offset.__lt__(arg)`

Note: Inherited from `durationtools.Duration`

`Offset.__mod__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__mul__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__ne__(arg)`

Note: Inherited from `durationtools.Duration`

`Offset.__neg__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__nonzero__(a)`
`a != 0`

Note: Inherited from *fractions.Fraction*

`Offset.__pos__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__pow__(*args)`

Note: Inherited from `durationtools.Duration`

Offset.__**radd**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rdiv**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rdivmod**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**repr**()

Note: Inherited from `durationtools.Duration`

Offset.__**rfloordiv**__(b, a)
a // b

Note: Inherited from *fractions.Fraction*

Offset.__**rmod**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rmul**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rpow**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rsub**__(*args)

Note: Inherited from `durationtools.Duration`

Offset.__**rtruediv**__(*args)

Note: Inherited from `durationtools.Duration`

`Offset.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Offset.__str__()`
`str(self)`

Note: Inherited from `fractions.Fraction`

`Offset.__sub__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__truediv__(*args)`

Note: Inherited from `durationtools.Duration`

`Offset.__trunc__(a)`
`trunc(a)`

Note: Inherited from `fractions.Fraction`

durationtools functions

durationtools.all_are_duration_tokens

`abjad.tools.durationtools.all_are_duration_tokens.all_are_duration_tokens(expr)`
New in version 2.6. True when *expr* is a sequence of Abjad duration tokens:

```
abjad> from abjad.tools import durationtools
```

```
abjad> duration_tokens = ['8.', (3, 16), Fraction(3, 16), Duration(3, 16)]
```

```
abjad> durationtools.all_are_duration_tokens(duration_tokens)
True
```

True when *expr* is an empty sequence:

```
abjad> durationtools.all_are_duration_tokens([])
True
```

Otherwise false:

```
abjad> durationtools.all_are_durations('foo')
False
```

Return boolean.

durationtools.all_are_durations

`abjad.tools.durationtools.all_are_durations.all_are_durations(expr)`
 New in version 2.6. True when *expr* is a sequence of Abjad durations:

```
abjad> from abjad.tools import durationtools

abjad> durations = [Duration((3, 16)), Duration((4, 16))]

abjad> durationtools.all_are_durations(durations)
True
```

True when *expr* is an empty sequence:

```
abjad> durationtools.all_are_durations([])
True
```

Otherwise false:

```
abjad> durationtools.all_are_durations('foo')
False
```

Return boolean.

durationtools.assignable_rational_to_dot_count

`abjad.tools.durationtools.assignable_rational_to_dot_count.assignable_rational_to_dot_count(rational)`
 New in version 2.0. Change assignable *rational* to dot count:

```
abjad> from abjad.tools import durationtools

abjad> for n in range(1, 9):
...     try:
...         rational = Fraction(n, 16)
...         dot_count = durationtools.assignable_rational_to_dot_count(rational)
...         print '%s\t%s' % (rational, dot_count)
...     except AssignabilityError:
...         pass
...
1/16    0
1/8     0
3/16    1
1/4     0
3/8     1
7/16    2
1/2     0
```

Raise assignability error when *rational* not assignable.

Return nonnegative integer.

`durationtools.assignable_rational_to_lilypond_duration_string`

`abjad.tools.durationtools.assignable_rational_to_lilypond_duration_string`.**assignable_rational_to_lilypond_duration_string**

New in version 2.0. Change assignable *rational* to LilyPond duration string:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.assignable_rational_to_lilypond_duration_string(Fraction(3, 16))
'8.'
```

Raise assignability error when *rational* not assignable.

Return string.

`durationtools.duration_pair_to_prolation_string`

`abjad.tools.durationtools.duration_pair_to_prolation_string`.**duration_pair_to_prolation_string**

New in version 2.0. Change positive integer duration *pair* to colon-separated prolation string:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.duration_pair_to_prolation_string((2, 3))
'3:2'
```

Return string.

`durationtools.duration_token_to_big_endian_list_of_assignable_duration_pairs`

`abjad.tools.durationtools.duration_token_to_big_endian_list_of_assignable_duration_pairs`.**duration_token_to_big_endian_list_of_assignable_duration_pairs**

New in version 1.1. Change *duration_token* to big-endian tuple of assignable duration pairs:

```
abjad> from abjad.tools import durationtools
```

```
abjad> duration_tokens = [(n, 16) for n in range(10, 20)]
abjad> for duration_token in duration_tokens:
...     print duration_token, durationtools.duration_token_to_big_endian_list_of_assignable_dura
...
(10, 16) ((8, 16), (2, 16))
(11, 16) ((8, 16), (3, 16))
(12, 16) ((12, 16),)
(13, 16) ((12, 16), (1, 16))
(14, 16) ((14, 16),)
(15, 16) ((15, 16),)
(16, 16) ((16, 16),)
(17, 16) ((16, 16), (1, 16))
(18, 16) ((16, 16), (2, 16))
(19, 16) ((16, 16), (3, 16))
```

Return tuple of integer pairs. Changed in version 2.0: renamed `durationtools.token_decompose()` to `durationtools.duration_token_to_big_endian_list_of_assignable_duration_pairs()`.

`durationtools.duration_token_to_duration_pair`

`abjad.tools.durationtools.duration_token_to_duration_pair`.**duration_token_to_duration_pair**

New in version 1.1. Change *duration_token* to duration pair:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.duration_token_to_duration_pair(Fraction(2, 4))
(1, 2)
```

New in version 2.0: Change LilyPond duration string to duration pair:

```
abjad> durationtools.duration_token_to_duration_pair('8.')
(3, 16)
```

Return pair. Changed in version 2.0: renamed `durationtools.token_unpack()` to `durationtools.duration_token_to_duration_pair()`.

durationtools.duration_token_to_rational

`abjad.tools.durationtools.duration_token_to_rational.duration_token_to_rational(duration_token)`

New in version 2.0. Change *duration_token* to rational:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.duration_token_to_rational((4, 16))
Fraction(1, 4)
```

```
abjad> durationtools.duration_token_to_rational('4.')
Fraction(3, 8)
```

Return fraction.

durationtools.duration_tokens_to_duration_pairs

`abjad.tools.durationtools.duration_tokens_to_duration_pairs.duration_tokens_to_duration_pairs(duration_tokens)`

New in version 2.0. Change *duration_tokens* to duration pairs:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.duration_tokens_to_duration_pairs([Fraction(2, 4), 3, '8.', (5, 16)])
[(1, 2), (3, 1), (3, 16), (5, 16)]
```

Return new object of *duration_tokens* type.

durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator

`abjad.tools.durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator.duration_tokens_to_duration_pairs_with_least_common_denominator(duration_tokens)`

New in version 2.0. Change *duration_tokens* to duration pairs with least common denominator:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator([Fraction(2, 4), 3, '8.', (5, 16)])
[(8, 16), (48, 16), (3, 16), (5, 16)]
```

Return new object of *duration_tokens* type.

`durationtools.duration_tokens_to_least_common_denominator`

`abjad.tools.durationtools.duration_tokens_to_least_common_denominator.duration_tokens_to_least_common_denominator`

New in version 2.0. Change *duration_tokens* to least common denominator:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.duration_tokens_to_least_common_denominator([Fraction(2, 4), 3, '8.', (5, 16)])
16
```

Return positive integer.

`durationtools.duration_tokens_to_rationals`

`abjad.tools.durationtools.duration_tokens_to_rationals.duration_tokens_to_rationals` (*duration_tokens*)

New in version 2.0. Change *duration_tokens* to rationals:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.duration_tokens_to_rationals([Fraction(2, 4), 3, '8.', (5, 16)])
[Fraction(1, 2), Fraction(3, 1), Fraction(3, 16), Fraction(5, 16)]
```

Return new object of *duration_tokens* type.

`durationtools.group_duration_tokens_by_implied_prolation`

`abjad.tools.durationtools.group_duration_tokens_by_implied_prolation.group_duration_tokens_by_implied_prolation`

New in version 1.1. Group *durations* by implied prolation:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.group_duration_tokens_by_implied_prolation([(1, 4), (1, 8), (1, 3), (1, 6)],
[[[1, 4), (1, 8)], [(1, 3), (1, 6)], [(1, 4)]]
```

Return list of integer pair lists. Changed in version 2.0: renamed `durationtools.agglomerate_by_prolation()` to `durationtools.group_duration_tokens_by_implied_prolation()`

`durationtools.is_assignable_rational`

`abjad.tools.durationtools.is_assignable_rational.is_assignable_rational` (*expr*)

New in version 1.1. True when *expr* is assignable rational. Otherwise false:

```
abjad> from abjad.tools import durationtools

abjad> for numerator in range(0, 16 + 1):
...     duration = Fraction(numerator, 16)
...     print '%s\t%s' % (duration, durationtools.is_assignable_rational(duration))
...
0      False
1/16   True
1/8    True
3/16   True
1/4    True
5/16   False
3/8    True
```

```

7/16  True
1/2   True
9/16  False
5/8   False
11/16 False
3/4   True
13/16 False
7/8   True
15/16 True
1      True

```

Return boolean. Changed in version 2.0: renamed `durationtools.is_assignable()` to `durationtools.is_assignable_rational()`.

`durationtools.is_binary_rational`

`abjad.tools.durationtools.is_binary_rational.is_binary_rational(rational)`

New in version 1.1. True when *rational* is of the form $1/2**n$. Otherwise false:

```

abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17): # doctest: +SKIP
...     rational = Fraction(1, n)
...     print '%s\t%s' % (rational, durationtools.is_binary_rational(rational))
...
1      True
1/2    True
1/3    False
1/4    True
1/5    False
1/6    False
1/7    False
1/8    True
1/9    False
1/10   False
1/11   False
1/12   False
1/13   False
1/14   False
1/15   False
1/16   True

```

Return boolean.

`durationtools.is_duration_pair`

`abjad.tools.durationtools.is_duration_pair.is_duration_pair(arg)`

New in version 1.1. True when *arg* has the form of a pair of integers that initialize a positive rational:

```

abjad> from abjad.tools import durationtools

abjad> durationtools.is_duration_pair((5, 16))
True

```

Otherwise false:

```
abjad> durationtools.is_duration_pair((-5, 16))
False
```

Return boolean. Changed in version 2.0: renamed `durationtools.is_pair()` to `durationtools.is_duration_pair()`.

`durationtools.is_duration_token`

`abjad.tools.durationtools.is_duration_token.is_duration_token(expr)`

New in version 2.0. True when *expr* has the form of an Abjad duration token:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.is_duration_token('8.')
True
```

Otherwise false:

```
abjad> durationtools.is_duration_token('foo')
False
```

Return boolean.

`durationtools.is_lilypond_duration_name`

`abjad.tools.durationtools.is_lilypond_duration_name.is_lilypond_duration_name(expr)`

New in version 2.0. True when *expr* is a LilyPond duration name:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.is_lilypond_duration_name('\\breve')
True
```

Otherwise false:

```
abjad> durationtools.is_lilypond_duration_name('foo')
False
```

The regex `^(\\breve|\\longa|\\maxima)$` underlies this predicate.

Return boolean.

`durationtools.is_lilypond_duration_string`

`abjad.tools.durationtools.is_lilypond_duration_string.is_lilypond_duration_string(expr)`

New in version 2.0. True when *expr* is a LilyPond duration string:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.is_lilypond_duration_string('4.. * 1/2')
True
```

Otherwise false:

```
abjad> durationtools.is_lilypond_duration_string('foo')
False
```


The regex `^(1|2|4|8|16|32|64|128|\breve|\longa|\maxima)\s*(\.*)\s*(*\s*(\d+(\d+)?))?$` underlies this predicate.

Return boolean.

`durationtools.lilypond_duration_string_to_rational`

`abjad.tools.durationtools.lilypond_duration_string_to_rational.lilypond_duration_string_to_rational`
New in version 2.0. Change LilyPond *duration_string* to rational:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.lilypond_duration_string_to_rational('8.')
Fraction(3, 16)
```

Return fraction.

`durationtools.lilypond_duration_string_to_rational_list`

`abjad.tools.durationtools.lilypond_duration_string_to_rational_list.lilypond_duration_string_to_rational_list`
New in version 2.0. Change LilyPond *duration_string* to rational list:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.lilypond_duration_string_to_rational_list('8.. 32 8.. 32')
[Fraction(7, 32), Fraction(1, 32), Fraction(7, 32), Fraction(1, 32)]
```

Return list of fractions.

`durationtools.multiply_duration_pair`

`abjad.tools.durationtools.multiply_duration_pair.multiply_duration_pair` (*pair*, *multiplier*)

New in version 1.1. Multiply duration *pair* by rational *multiplier*:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.multiply_duration_pair((4, 8), Fraction(4, 5))
(16, 40)
```

Naive multiplication with no simplification of anything intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_naive()` to `durationtools.multiply_duration_pair()`.

`durationtools.multiply_duration_pair_and_reduce_factors`

`abjad.tools.durationtools.multiply_duration_pair_and_reduce_factors.multiply_duration_pair_and_reduce_factors`

New in version 1.1. Multiply *pair* by rational *multiplier* and reduce factors:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.multiply_duration_pair_and_reduce_factors((4, 8), Fraction(2, 3))
(4, 12)
```

Intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_reduce_factors()` to `durationtools.multiply_duration_pair_and_reduce_factors()`.

`durationtools.multiply_duration_pair_and_try_to_preserve_numerator`

```
abjad.tools.durationtools.multiply_duration_pair_and_try_to_preserve_numerator.multiply_du
```

New in version 1.1. Multiply duration *pair* by rational *multiplier* and try to preserve numerator:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.multiply_duration_pair_and_try_to_preserve_numerator((9, 16), Fraction(2, 3))
(9, 24)
```

Intended for certain types of meter multiplication.

Return integer pair. Changed in version 2.0: renamed `durationtools.pair_multiply_constant_numerator()` to `durationtools.multiply_duration_pair_and_try_to_preserve_numerator()`.

`durationtools.numeric_seconds_to_clock_string`

```
abjad.tools.durationtools.numeric_seconds_to_clock_string.numeric_seconds_to_clock_string(s)
```

New in version 2.0. Change numeric *seconds* to clock string:

```
abjad> from abjad.tools import durationtools
```

```
abjad> durationtools.numeric_seconds_to_clock_string(117)
'1\57''
```

Return string.

`durationtools.numeric_seconds_to_escaped_clock_string`

```
abjad.tools.durationtools.numeric_seconds_to_escaped_clock_string.numeric_seconds_to_escaped
```

New in version 2.0. Change numeric *seconds* to escaped clock string:

```
abjad> from abjad.tools import durationtools
```

```
abjad> note = Note("c'4")
abjad> clock_string = durationtools.numeric_seconds_to_escaped_clock_string(117)
abjad> markuptools.Markup('%s' % clock_string, 'up')(note)
Markup('1\57\'\'', direction='^')(c'4)
```

```
abjad> f(note)
c'4 ^ \markup { "1'57\'" }
```

Escape seconds indicator for output as LilyPond markup.

Return string.

`durationtools.positive_integer_to_implied_prolation_multiplier`

`abjad.tools.durationtools.positive_integer_to_implied_prolation_multiplier`.**positive_integer**

New in version 1.1. Change positive integer *n* to implied porlation multiplier:

```
abjad> from abjad.tools import durationtools

abjad> for denominator in range(1, 17): # doctest: +SKIP
...     multiplier = durationtools.positive_integer_to_implied_prolation_multiplier(denominator)
...     print '%s\t%s' % (denominator, multiplier)
...
1         1
2         1
3         2/3
4         1
5         4/5
6         2/3
7         4/7
8         1
9         8/9
10        4/5
11        8/11
12        2/3
13        8/13
14        4/7
15        8/15
16        1
```

Return positive fraction less than or equal to 1. Changed in version 2.0: renamed `durationtools.denominator_to_multiplier()` to `durationtools.positive_integer_to_implied_prolation_multiplier()`.

`durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator`

`abjad.tools.durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator`

Change *duration* to duration pair with multiple of specified *integer_denominator*:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fr
(1, 2)
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fr
(2, 4)
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fr
(4, 8)
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fr
(8, 16)
```

```

abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(3, 6))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(3, 6))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(6, 12))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(12, 24))

abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(5, 10))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(5, 10))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(10, 20))
abjad> durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator(Fraction(20, 40))

```

Return integer pair. Changed in version 2.0: renamed `durationtools.in_terms_of_binary_multiple()` to `durationtools.rational_to_duration_pair_with_multiple_of_specified_integer_denominator()`

`durationtools.rational_to_duration_pair_with_specified_integer_denominator`

```
abjad.tools.durationtools.rational_to_duration_pair_with_specified_integer_denominator.rat...
```

New in version 1.1. Change *duration* to *duraiton* pair with specified *integer_denominator*:

```

abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17):
...     rational = Fraction(n, 16)
...     pair = durationtools.rational_to_duration_pair_with_specified_integer_denominator(rational)
...     print '%s\t%s' % (rational, pair)
...
1/16      (1, 16)
1/8       (2, 16)
3/16      (3, 16)
1/4       (4, 16)
5/16      (5, 16)
3/8       (6, 16)
7/16      (7, 16)
1/2       (8, 16)
9/16      (9, 16)
5/8       (10, 16)
11/16     (11, 16)
3/4       (12, 16)
13/16     (13, 16)
7/8       (14, 16)
15/16     (15, 16)
1         (16, 16)

```

Return integer pair. Changed in version 2.0: renamed `durationtools.in_terms_of()` to `durationtools.rational_to_duration_pair_with_specified_integer_denominator()`.

durationtools.rational_to_equal_or_greater_assignable_rational

`abjad.tools.durationtools.rational_to_equal_or_greater_assignable_rational.rational_to_equal_or_greater_assignable_rational`

New in version 1.1. Change *rational* to equal or greater assignable rational:

```
abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17): # doctest: +SKIP
...     prolated = Fraction(n, 16)
...     written = durationtools.rational_to_equal_or_greater_assignable_rational(prolated)
...     print '%s/16\t%s' % (n, written)
...
1/16    1/16
2/16    1/8
3/16    3/16
4/16    1/4
5/16    3/8
6/16    3/8
7/16    7/16
8/16    1/2
9/16    3/4
10/16   3/4
11/16   3/4
12/16   3/4
13/16   7/8
14/16   7/8
15/16   15/16
16/16   1
```

Return fraction.

Function returns dotted and double dotted durations where possible. Changed in version 2.0: Fixed to produce monotonically increasing output in response to monotonically increasing input. Changed in version 2.0: renamed `durationtools.prolated_to_written_not_less_than()` to `durationtools.rational_to_equal_or_greater_assignable_rational()`.

durationtools.rational_to_equal_or_greater_binary_rational

`abjad.tools.durationtools.rational_to_equal_or_greater_binary_rational.rational_to_equal_or_greater_binary_rational`

New in version 1.1. Change *rational* to equal to greater binary rational:

```
abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17): # doctest: +SKIP
...     rational = Fraction(n, 16)
...     written_duration = durationtools.rational_to_equal_or_greater_binary_rational(rational)
...     print '%s/16\t%s' % (n, written_duration)
...
1/16    1/16
2/16    1/8
3/16    1/4
4/16    1/4
5/16    1/2
6/16    1/2
7/16    1/2
8/16    1/2
9/16    1
```

```
10/16 1
11/16 1
12/16 1
13/16 1
14/16 1
15/16 1
16/16 1
```

```
abjad> durationtools.rational_to_equal_or_greater_binary_rational(Fraction(1, 80))
Fraction(1, 64)
```

```
abjad> durationtools.rational_to_equal_or_greater_binary_rational(Fraction(17, 16))
Fraction(2, 1)
```

Use to find written duration of tupletted leaves.

Return fraction. Changed in version 2.0: renamed `durationtools.naive_prolated_to_written_not_less_than` to `durationtools.rational_to_equal_or_greater_binary_rational()`.

`durationtools.rational_to_equal_or_lesser_assignable_rational`

`abjad.tools.durationtools.rational_to_equal_or_lesser_assignable_rational`. **`rational_to_equal_or_lesser_assignable_rational`**
 New in version 1.1. Change *rational* to equal or lesser assignable rational:

```
abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17): # doctest: +SKIP
...     rational = Fraction(n, 16)
...     written = durationtools.rational_to_equal_or_lesser_assignable_rational(rational)
...     print '%s/16\t%s' % (n, written)
...
1/16 1/16
2/16 1/8
3/16 3/16
4/16 1/4
5/16 1/4
6/16 3/8
7/16 7/16
8/16 1/2
9/16 1/2
10/16 1/2
11/16 1/2
12/16 3/4
13/16 3/4
14/16 7/8
15/16 15/16
16/16 1
```

Return fraction.

Function returns dotted and double dotted durations where possible. Changed in version 2.0: Fixed to produce monotonically increasing output in response to monotonically increasing input. Changed in version 2.0: renamed `durationtools.prolated_to_written_not_greater_than()` to `durationtools.rational_to_equal_or_lesser_assignable_rational()`.

durationtools.rational_to_equal_or_lesser_binary_rational

`abjad.tools.durationtools.rational_to_equal_or_lesser_binary_rational.rational_to_equal_or_lesser_binary_rational`
 New in version 1.1. Change *rational* to equal or lesser binary rational:

```
abjad> from abjad.tools import durationtools

abjad> for n in range(1, 17): # doctest: +SKIP
...     rational = Fraction(n, 16)
...     written_duration = durationtools.rational_to_equal_or_lesser_binary_rational(rational)
...     print '%s/16\t%s' % (n, written_duration)
...
1/16    1/16
2/16    1/8
3/16    1/8
4/16    1/4
5/16    1/4
6/16    1/4
7/16    1/4
8/16    1/2
9/16    1/2
10/16   1/2
11/16   1/2
12/16   1/2
13/16   1/2
14/16   1/2
15/16   1/2
16/16   1

abjad> durationtools.rational_to_equal_or_lesser_binary_rational(Fraction(1, 80))
Fraction(1, 128)
```

Return fraction.

Function intended to find written duration of notes inside tuplet. Changed in version 2.0: re-named `durationtools.naive_prolated_to_written_not_greater_than()` to `durationtools.rational_to_equal_or_lesser_binary_rational()`.

durationtools.rational_to_flag_count

`abjad.tools.durationtools.rational_to_flag_count.rational_to_flag_count` (*rational*)
 New in version 2.0. Change *rational* to number of flags required to notate:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.rational_to_flag_count(Fraction(1, 32))
3
```

Return nonnegative integer.

durationtools.rational_to_fraction_string

`abjad.tools.durationtools.rational_to_fraction_string.rational_to_fraction_string` (*rational*)
 New in version 1.1. Change *rational* to fraction string:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.rational_to_fraction_string(Fraction(2, 4))
'1/2'
```

Return string.

durationtools.rational_to_prolation_string

`abjad.tools.durationtools.rational_to_prolation_string.rational_to_prolation_string(rational)`
New in version 2.0. Change *rational* to prolotion string:

```
abjad> from abjad.tools import durationtools

abjad> generator = durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order_unique
abjad> for n in range(16): # doctest: +SKIP
...     rational = generator.next()
...     prolotion_string = durationtools.rational_to_prolation_string(rational)
...     print '%s\\t%s' % (rational, prolotion_string)
...
1         1:1
2         1:2
1/2       2:1
1/3       3:1
3         1:3
4         1:4
3/2       2:3
2/3       3:2
1/4       4:1
1/5       5:1
5         1:5
6         1:6
5/2       2:5
4/3       3:4
3/4       4:3
2/5       5:2
```

Return string.

durationtools.rational_to_proper_fraction

`abjad.tools.durationtools.rational_to_proper_fraction.rational_to_proper_fraction(rational)`
New in version 2.0. Change *rational* to proper fraction:

```
abjad> from abjad.tools import durationtools

abjad> durationtools.rational_to_proper_fraction(Fraction(116, 8))
(14, Fraction(1, 2))
```

Return pair.

durationtools.rewrite_rational_under_new_tempo

`abjad.tools.durationtools.rewrite_rational_under_new_tempo.rewrite_rational_under_new_tempo`

New in version 2.0. Given *prolated_duration_1* governed by *tempo_mark_1*, return *prolated_duration_2* governed by *tempo_mark_2* such that *prolated_duration_1* and *prolated_duration_2* consume exactly the same amount of time in seconds.

Consider the two tempo indications below.

```
abjad> from abjad.tools import durationtools

abjad> tempo_mark_1 = contexttools.TempoMark(Duration(1, 4), 60)
abjad> tempo_mark_2 = contexttools.TempoMark(Duration(1, 4), 90)
```

The first tempo indication specifies quarter = 60 MM. The second tempo indication specifies quarter = 90 MM.

The second tempo is 1 1/2 times as fast as the first.

```
abjad> tempo_mark_2 / tempo_mark_1
Duration(3, 2)
```

An triplet eighth note at tempo 1 equals a regular eighth note at tempo 2.

```
abjad> durationtools.rewrite_rational_under_new_tempo(Duration(1, 12), tempo_mark_1, tempo_mark_2)
Duration(1, 8)
```

Conversely, a regular eighth note at tempo 1 equals a dotted sixteenth at tempo 2.

```
abjad> durationtools.rewrite_rational_under_new_tempo(Duration(1, 8), tempo_mark_1, tempo_mark_2)
Duration(3, 16)
```

Return fraction.

durationtools.yield_all_assignable_rationals_in_cantor_diagonalized_order

`abjad.tools.durationtools.yield_all_assignable_rationals_in_cantor_diagonalized_order.yield_all_assignable_rationals_in_cantor_diagonalized_order`

New in version 2.0. Yield all assignable rationals in Cantor diagonalized order:

```
abjad> from abjad.tools import durationtools

abjad> generator = durationtools.yield_all_assignable_rationals_in_cantor_diagonalized_order()
abjad> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
Fraction(1, 4)
Fraction(6, 1)
Fraction(3, 4)
Fraction(7, 1)
Fraction(8, 1)
Fraction(7, 2)
```

```
Fraction(1, 8)
Fraction(7, 4)
Fraction(3, 8)
Fraction(12, 1)
```

Return fraction generator.

durationtools.yield_all_positive_integer_pairs_in_cantor_diagonalized_order

`abjad.tools.durationtools.yield_all_positive_integer_pairs_in_cantor_diagonalized_order.yield_all_positive_integer_pairs_in_cantor_diagonalized_order`

New in version 2.0. Yield all positive integer pairs in Cantor diagonalized order:

```
abjad> from abjad.tools import durationtools

abjad> generator = durationtools.yield_all_positive_integer_pairs_in_cantor_diagonalized_order()
abjad> for n in range(16):
...     generator.next()
...
(1, 1)
(2, 1)
(1, 2)
(1, 3)
(2, 2)
(3, 1)
(4, 1)
(3, 2)
(2, 3)
(1, 4)
(1, 5)
(2, 4)
(3, 3)
(4, 2)
(5, 1)
(6, 1)
```

Return pair generator.

durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order

`abjad.tools.durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order.yield_all_positive_rationals_in_cantor_diagonalized_order`

New in version 2.0. Yield all positive rationals in Cantor diagonalized order:

```
abjad> from abjad.tools import durationtools

abjad> generator = durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order()
abjad> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(1, 3)
Fraction(1, 1)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
```

```
Fraction(2, 3)
Fraction(1, 4)
Fraction(1, 5)
Fraction(1, 2)
Fraction(1, 1)
Fraction(2, 1)
Fraction(5, 1)
Fraction(6, 1)
```

Return fraction generator.

durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order_uniquely

`abjad.tools.durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order_uniquely`
New in version 2.0. Yield all positive rationals in Cantor diagonalized order uniquely:

```
abjad> from abjad.tools import durationtools

abjad> generator = durationtools.yield_all_positive_rationals_in_cantor_diagonalized_order_uniquely
abjad> for n in range(16):
...     generator.next()
...
Fraction(1, 1)
Fraction(2, 1)
Fraction(1, 2)
Fraction(1, 3)
Fraction(3, 1)
Fraction(4, 1)
Fraction(3, 2)
Fraction(2, 3)
Fraction(1, 4)
Fraction(1, 5)
Fraction(5, 1)
Fraction(6, 1)
Fraction(5, 2)
Fraction(4, 3)
Fraction(3, 4)
Fraction(2, 5)
```

Return fraction generator.

durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalized_order

`abjad.tools.durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalized_order`

New in version 2.0. Yield all prolation rewrite pairs of *prolated_duration* in Cantor diagonalized order.

Ensure written duration never less than *minimum_written_duration*.

The different ways to notate a prolated duration of 1/8:

```
abjad> from abjad.tools import durationtools
```

```
abjad> pairs = durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalize
abjad> for pair in pairs: pair
...
(Fraction(1, 1), Fraction(1, 8))
(Fraction(2, 3), Fraction(3, 16))
(Fraction(4, 3), Fraction(3, 32))
(Fraction(4, 7), Fraction(7, 32))
(Fraction(8, 7), Fraction(7, 64))
(Fraction(8, 15), Fraction(15, 64))
(Fraction(16, 15), Fraction(15, 128))
(Fraction(16, 31), Fraction(31, 128))
```

The different ways to notate a prolated duration of 1/12.

```
abjad> pairs = durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalize
abjad> for pair in pairs: pair
...
(Fraction(2, 3), Fraction(1, 8))
(Fraction(4, 3), Fraction(1, 16))
(Fraction(8, 9), Fraction(3, 32))
(Fraction(16, 9), Fraction(3, 64))
(Fraction(16, 21), Fraction(7, 64))
(Fraction(32, 21), Fraction(7, 128))
(Fraction(32, 45), Fraction(15, 128))
```

The different ways to notate a prolated duration of 5/48.

```
abjad> pairs = durationtools.yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalize
abjad> for pair in pairs: pair
...
(Fraction(5, 6), Fraction(1, 8))
(Fraction(5, 3), Fraction(1, 16))
(Fraction(5, 9), Fraction(3, 16))
(Fraction(10, 9), Fraction(3, 32))
(Fraction(20, 21), Fraction(7, 64))
(Fraction(40, 21), Fraction(7, 128))
(Fraction(8, 9), Fraction(15, 128))
```

Return generator of paired fractions.

iotools

iotools functions

iotools.capitalize_string_start

`abjad.tools.iotools.capitalize_string_start.capitalize_string_start(string)`

New in version 2.5. Capitalize *string*:

```
abjad> string = 'violin I'

abjad> iotools.capitalize_string_start(string)
'Violin I'
```

Function differs from built-in `string.capitalize()`.

This function affects only `string[0]` and leaves noninitial characters as-is.

Built-in `string.capitalize()` forces noninitial characters to lowercase.

```
abjad> string.capitalize() 'Violin i'
```

Return newly constructed string.

iotools.clear_terminal

```
abjad.tools.iotools.clear_terminal.clear_terminal()
```

New in version 2.0. Run `clear` if OS is POSIX-compliant (UNIX / Linux / MacOS).

Run `cls` if OS is not POSIX-compliant (Windows):

```
abjad> iotools.clear_terminal()
```

Return none.

iotools.f

```
abjad.tools.iotools.f.f(expr)
```

Format *expr* and print to standard out:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
```

```
abjad> f(staff)
\new Staff {
    c'8
    d'8
    e'8
    f'8
}
```

Return none.

iotools.format_input_lines_as_doc_string

```
abjad.tools.iotools.format_input_lines_as_doc_string.format_input_lines_as_doc_string(input_lines)
```

New in version 2.0. Format *input_lines* as doc string.

Format expressions intelligently.

Treat blank lines intelligently.

Capture hash-suffixed line output.

Use when writing docstrings.

Example skipped because docstring goes crazy on example input.

iotools.format_input_lines_as_regression_test

```
abjad.tools.iotools.format_input_lines_as_regression_test.format_input_lines_as_regression_test(input_lines)
```

New in version 2.0. Format *input_lines* as regression test:

```

abjad> input_lines = '''
... staff = Staff("c'8 d'8 e'8 f'8")
... spannertools.BeamSpanner(staff.leaves)
... f(staff)
...
... tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])
... f(staff)
... '''
abjad> iotools.format_input_lines_as_regression_test(input_lines) # doctest: +SKIP

    staff = Staff("c'8 d'8 e'8 f'8")
    spannertools.BeamSpanner(staff.leaves)

    r'''
    \new Staff {
      c'8 [
        d'8
        e'8
        f'8 ]
    }
    '''

    tuplettools.FixedDurationTuplet(Duration(2, 8), staff[:3])

    r'''
    \new Staff {
      \times 2/3 {
        c'8 [
          d'8
          e'8
        ]
      }
      f'8 ]
    }

    assert componenttools.is_well_formed_component(staff)
    assert staff.format == "\\new Staff {\\n\\t\\times 2/3 {\\n\\t\\tc'8 [\\n\\t\\td'8\\n\\t\\te'8\\n\\t}\\n\\t\\t
    '''

```

Format expressions intelligently.

Treat blank lines intelligently.

Remove line-final hash characters.

Used when writing tests.

iotools.get_last_output_file_name

`abjad.tools.iotools.get_last_output_file_name.get_last_output_file_name()`

Get last output file name like 6222.ly.

Return string.

iotools.get_next_output_file_name

`abjad.tools.iotools.get_next_output_file_name.get_next_output_file_name()`

Get next output file name like 6223.ly.

Return string.

iotools.is_lowercamelcase_string

`abjad.tools.iotools.is_lowercamelcase_string.is_lowercamelcase_string(expr)`

New in version 2.5. True when *expr* is a string and is lowercamelcase:

```
abjad> iotools.is_lowercamelcase_string('fooBar')
True
```

False otherwise:

```
abjad> iotools.is_lowercamelcase_string('FooBar')
False
```

Return boolean.

iotools.is_space_delimited_lowercase_string

`abjad.tools.iotools.is_space_delimited_lowercase_string.is_space_delimited_lowercase_string(expr)`

New in version 2.5. True when *expr* is a string and is space-delimited lowercase:

```
abjad> iotools.is_space_delimited_lowercase_string('foo bar')
True
```

False otherwise:

```
abjad> iotools.is_space_delimited_lowercase_string('foo_bar')
False
```

Return boolean.

iotools.is_underscore_delimited_lowercase_file_name

`abjad.tools.iotools.is_underscore_delimited_lowercase_file_name.is_underscore_delimited_lowercase_file_name(expr)`

New in version 2.7. True when *expr* is a string and is underscore-delimited lowercase file name with extension:

```
abjad> iotools.is_underscore_delimited_lowercase_file_name('foo_bar')
True
```

False otherwise:

```
abjad> iotools.is_underscore_delimited_lowercase_file_name('foo.bar.blah')
False
```

Return boolean.

iotools.is_underscore_delimited_lowercase_file_name_with_extension

`abjad.tools.iotools.is_underscore_delimited_lowercase_file_name_with_extension.is_underscore_delimited_lowercase_file_name_with_extension(expr)`

New in version 2.7. True when *expr* is a string and is underscore-delimited lowercase file name with extension:

```
abjad> iotools.is_underscore_delimited_lowercase_file_name_with_extension('foo_bar.blah')
True
```

False otherwise:

```
abjad> iotools.is_underscore_delimited_lowercase_file_name_with_extension('foo.bar.blah')
False
```

Return boolean.

iotools.is_underscore_delimited_lowercase_package_name

`abjad.tools.iotools.is_underscore_delimited_lowercase_package_name.is_underscore_delimited_lowercase_package_name`
New in version 2.5. True when *expr* is a string and is underscore-delimited lowercase package name:

```
abjad> iotools.is_underscore_delimited_lowercase_package_name('foo.bar.blah_package')
True
```

False otherwise:

```
abjad> iotools.is_underscore_delimited_lowercase_package_name('foo.bar.BlahPackage')
False
```

Return boolean.

iotools.is_underscore_delimited_lowercase_string

`abjad.tools.iotools.is_underscore_delimited_lowercase_string.is_underscore_delimited_lowercase_string`
New in version 2.5. True when *expr* is a string and is underscore delimited lowercase:

```
abjad> iotools.is_underscore_delimited_lowercase_string('foo_bar')
True
```

False otherwise:

```
abjad> iotools.is_underscore_delimited_lowercase_string('foo bar')
False
```

Return boolean.

iotools.is_uppercamelcase_string

`abjad.tools.iotools.is_uppercamelcase_string.is_uppercamelcase_string` (*expr*)
New in version 2.5. True when *expr* is a string and is uppercamelcase:

```
abjad> iotools.is_uppercamelcase_string('FooBar')
True
```

False otherwise:

```
abjad> iotools.is_uppercamelcase_string('fooBar')
False
```

Return boolean.

iotools.log

`abjad.tools.iotools.log.log()`
Open the LilyPond log file in operating system-specific text editor:


```
abjad> iotools.log() # doctest: +SKIP
```

```
GNU LilyPond 2.12.2
Processing `0440.ly'
Parsing...
Interpreting music...
Preprocessing graphical objects...
Finding the ideal number of pages...
Fitting music on 1 page...
Drawing systems...
Layout output to `0440.ps'...
Converting to `./0440.pdf'...
```

Exit text editor in the usual way.

Return none.

iotools.ly

`abjad.tools.iotools.ly.ly(target=-1)`

Open the last LilyPond output file in text editor:

```
abjad> iotools.ly() # doctest: +SKIP
```

```
% Abjad revision 2162
% 2009-05-31 14:29
```

```
\version "2.12.2"
\include "english.ly"
\include "/Path/to/abjad/trunk/abjad/cfg/abjad.scm"
```

```
{
  c'4
}
```

Open the next-to-last LilyPond output file in text editor:

```
abjad> iotools.ly(-2) # doctest: +SKIP
```

Exit text editor in the usual way.

Return none.

iotools.p

`abjad.tools.iotools.p.p(*args)`

New in version 2.7. Parse *args* as LilyPond string:

```
abjad> p("{c'4 d'4 e'4 f'4}")
{c'4, d'4, e'4, f'4}
```

```
abjad> container = _
abjad> f(container)
{
  c'4
  d'4
  e'4
```

```
f'4  
}
```

A pitch-name language may also be specified.

```
abjad> p("{c'8 des' e' fis'}", 'nederlands')  
{c'8, df'8, e'8, fs'8}
```

Return Abjad expression.

iotools.parse_lilypond_input_string

`abjad.tools.iotools.parse_lilypond_input_string.parse_lilypond_input_string(note_entry_string)`

New in version 2.0. Parse LilyPond *note_entry_string*:

```
abjad> note_entry_string = "g'2 a'2 g'4. fs'8 e'4 d'4"  
abjad> iotools.parse_lilypond_input_string(note_entry_string)  
{g'2, a'2, g'4., fs'8, e'4, d'4}
```

Return container of note, rest and chord instances.

Handle simple beaming, slurs and articulations.

Do not parse tuplets, measures or other complex LilyPond input.

iotools.pdf

`abjad.tools.iotools.pdf.pdf(target=-1)`

Open the last PDF generated by Abjad with `iotools.pdf()`.

Open the next-to-last PDF generated by Abjad with `iotools.pdf(-2)`.

Return none.

Abjad writes PDFs to the `~/ .abjad/output` directory by default.

You may change this by setting the `abjad_output` variable in the `config.py` file.

iotools.play

`abjad.tools.iotools.play.play(expr)`

Play *expr*:

```
abjad> note = Note("c'4")  
  
abjad> iotools.play(note) # doctest: +SKIP
```

This input creates and opens a one-note MIDI file.

Abjad outputs MIDI files of the format `filename.mid` under Windows.

Abjad outputs MIDI files of the format `filename.midi` under other operating systems.

iotools.profile_expr

```
abjad.tools.iotools.profile_expr.profile_expr(expr, sort_by='cum', num_lines=12,
                                              strip_dirs=True)
```

Profile *expr*:

```
abjad> iotools.profile_expr('Staff(notetools.make_repeated_notes(8))') # doctest: +SKIP
Tue Apr  5 20:32:40 2011      _tmp_abj_profile
```

2852 function calls (2829 primitive calls) in 0.006 CPU seconds

Ordered by: cumulative time

List reduced from 118 to 12 due to restriction <12>

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.006	0.006	<string>:1(<module>)
1	0.000	0.000	0.003	0.003	make_repeated_notes.py:5(make_repeated_notes)
1	0.001	0.001	0.003	0.003	make_notes.py:12(make_notes)
1	0.000	0.000	0.003	0.003	Staff.py:21(__init__)
1	0.000	0.000	0.003	0.003	Context.py:11(__init__)
1	0.000	0.000	0.003	0.003	Container.py:23(__init__)
1	0.000	0.000	0.003	0.003	Container.py:271(_initialize_music)
2	0.000	0.000	0.002	0.001	all_are_thread_contiguous_components.py:9(all_are_
52	0.001	0.000	0.002	0.000	component_to_thread_signature.py:5(component_to_th
1	0.000	0.000	0.002	0.002	_construct_unprolated_notes.py:4(_construct_unprol
8	0.000	0.000	0.002	0.000	_construct_tied_note.py:5(_construct_tied_note)
8	0.000	0.000	0.002	0.000	_construct_tied_leaf.py:5(_construct_tied_leaf)

Function wraps the built-in Python `cProfile` module.

Set *expr* to any string of Abjad input.

Set *sort_by* to 'cum', 'time' or 'calls'.

Set *num_lines* to any positive integer.

Set *strip_dirs* to `True` to strip directory names from output lines.

Note: This function fails on some Linux distros. Some Linux distributions do not include the Python `pstats` module.

Note: This function creates the file `_tmp_abj_profile` in the directory from which it is run.

Note: For information on reading the output of the different Python profilers, see [the Python docs](#).

Changed in version 2.0: renamed `check.profile()` to `iotools.profile_expr()`.

iotools.redo

```
abjad.tools.iotools.redo.redo(target=-1, lily_time=10)
```

Rerender the last .ly file created in Abjad and then show the resulting PDF:

```
abjad> iotools.redo() # doctest: +SKIP
```

Rerender the next-to-last .ly file created in Abjad and then show the resulting PDF:

```
abjad> iotools.redo(-2) # doctest: +SKIP
```

Return none.

iotools.remove_abjad__pycache__directories

```
abjad.tools.iotools.remove_abjad__pycache__directories.remove_abjad__pycache__director:
```

Remove __pycache__ directories from Abjad source tree:

```
abjad> iotools.remove_abjad__pycache__directories() # doctest: +SKIP
```

Return none.

iotools.remove_abjad_pyc_files

```
abjad.tools.iotools.remove_abjad_pyc_files.remove_abjad_pyc_files()
```

Remove .pyc files from Abjad source tree:

```
abjad> iotools.remove_abjad_pyc_files() # doctest: +SKIP
```

Return none.

iotools.save_last_ly_as

```
abjad.tools.iotools.save_last_ly_as.save_last_ly_as(file_name)
```

New in version 2.0. Save last ly file as *file_name*:

```
abjad> iotools.save_last_ly_as('/project/output/example-1.ly') # doctest: +SKIP
```

Return none.

iotools.save_last_pdf_as

```
abjad.tools.iotools.save_last_pdf_as.save_last_pdf_as(file_name)
```

New in version 2.0. Save last PDF as *file_name*:

```
abjad> iotools.save_last_pdf_as('/project/output/example-1.pdf') # doctest: +SKIP
```

Return none.

iotools.show

```
abjad.tools.iotools.show.show(expr, template=None, return_timing=False, suppress_pdf=False)
```

Show *expr*:

```
abjad> note = Note("c'4")
abjad> show(note) # doctest: +SKIP
```

Show *expr* with *template*:

```
abjad> note = Note("c'4")
abjad> show(note, template = 'tangiers') # doctest: +SKIP
```

Show *expr* and return both Abjad and LilyPond processing time in seconds:

```
abjad> staff = Staff(Note("c'4") * 200)
abjad> show(note, return_timing = True) # doctest: +SKIP
(0, 3)
```

Return none or timing tuple.

Abjad writes LilyPond input files to the ~/ .abjad/output directory by default.

You may change this by setting the `abjad_output` variable in the `config.py` file.

iotools.space_delimited_lowercase_to_uppercamelcase

`abjad.tools.iotools.space_delimited_lowercase_to_uppercamelcase.space_delimited_lowercase_to_uppercamelcase(string)`

New in version 2.6. Change space-delimited lowercase *string* to uppercamelcase:

```
abjad> string = 'bass figure alignment positioning'
abjad> iotools.space_delimited_lowercase_to_uppercamelcase(string)
'BassFigureAlignmentPositioning'
```

Return string.

iotools.string_to_strict_directory_name

`abjad.tools.iotools.string_to_strict_directory_name.string_to_strict_directory_name(string)`

New in version 2.6. Change *string* to strict directory name:

```
abjad> iotools.string_to_strict_directory_name('Déja vu')
'deja_vu'
```

Strip accents from accented characters. Change all punctuation (including spaces) to underscore. Set to lowercase.

Return string.

iotools.strip_diacritics_from_binary_string

`abjad.tools.iotools.strip_diacritics_from_binary_string.strip_diacritics_from_binary_string(binary_string)`

New in version 2.5. Strip diacritics from *binary_string*:

```
abjad> binary_string = 'Dvo\x5c\x99\x3\xa1k'

abjad> print binary_string
Dvořák

abjad> iotools.strip_diacritics_from_binary_string(binary_string)
'Dvorak'
```

Return ASCII string.

iotools.underscore_delimited_lowercase_to_lowercamelcase

`abjad.tools.iotools.underscore_delimited_lowercase_to_lowercamelcase.underscore_delimited_lowercase_to_lowercamelcase(string)`

New in version 2.0. Change underscore-delimited lowercase *string* to lowercamelcase:

```
abjad> string = 'bass_figure_alignment_positioning'
abjad> iotools.underscore_delimited_lowercase_to_lowercamelcase(string)
'bassFigureAlignmentPositioning'
```

Changed in version 2.0: renamed `stringtools.underscore_delimited_lowercase_to_lowercamelcase()` to `iotools.underscore_delimited_lowercase_to_lowercamelcase()`.

iotools.underscore_delimited_lowercase_to_uppercamelcase

`abjad.tools.iotools.underscore_delimited_lowercase_to_uppercamelcase.underscore_delimited_`
New in version 2.0. Change underscore-delimited lowercase *string* to uppercamelcase:

```
abjad> string = 'bass_figure_alignment_positioning'
abjad> iotools.underscore_delimited_lowercase_to_uppercamelcase(string)
'BassFigureAlignmentPositioning'
```

Changed in version 2.0: renamed `stringtools.underscore_delimited_lowercase_to_uppercamelcase()` to `iotools.underscore_delimited_lowercase_to_uppercamelcase()`.

iotools.uppercamelcase_to_space_delimited_lowercase

`abjad.tools.iotools.uppercamelcase_to_space_delimited_lowercase.uppercamelcase_to_space_de`
New in version 2.6. Change uppercamelcase *string* to space-delimited lowercase:

```
abjad> string = 'KeySignatureMark'

abjad> iotools.uppercamelcase_to_space_delimited_lowercase(string)
'key signature mark'
```

Return string.

iotools.uppercamelcase_to_underscore_delimited_lowercase

`abjad.tools.iotools.uppercamelcase_to_underscore_delimited_lowercase.uppercamelcase_to_unde`
New in version 2.6. Change uppercamelcase *string* to underscore-delimited lowercase:

```
abjad> string = 'KeySignatureMark'

abjad> iotools.uppercamelcase_to_underscore_delimited_lowercase(string)
'key_signature_mark'
```

Return string.

iotools.write_expr_to_ly

`abjad.tools.iotools.write_expr_to_ly.write_expr_to_ly(expr, file_name,`
template=None,
print_status=True,
tagline=False)

Write *expr* to *file_name*:

```
abjad> note = Note("c'4")
abjad> iotools.write_expr_to_ly(note, '/home/user/foo.ly') # doctest: +SKIP
```

Write *expr* to *file_name* with *template*:

```
abjad> note = Note("c'4")
abjad> iotools.write_expr_to_ly(note, '/home/user/foo.ly', 'paris') # doctest: +SKIP
```

Return none. Changed in version 2.0: renamed `io.write_ly()` to `io.write_expr_to_ly()`.

iotools.write_expr_to_ly_and_to_pdf_and_show

```
abjad.tools.iotools.write_expr_to_ly_and_to_pdf_and_show.write_expr_to_ly_and_to_pdf_and_sl
```

Write *expr* to named `.ly` and to PDF and then open the resulting PDF:

```
abjad> iotools.write_expr_to_ly_and_to_pdf_and_show(Note("c'8"), 'file_name_stem') # doctest: +S
```

Write *expr* to temporary `.ly` and to PDF and then open the resulting PDF:

```
abjad> iotools.write_expr_to_ly_and_to_pdf_and_show(Note("c'8"), 'file_name_stem', write = False
```

Return none.

The purpose of this function is to save named `.ly` and PDF output. Changed in version 2.0: renamed `io.write_and_show()` to `io.write_expr_to_ly_and_to_pdf_and_show()`.

iotools.write_expr_to_pdf

```
abjad.tools.iotools.write_expr_to_pdf.write_expr_to_pdf(expr, file_name,
                                                         template=None,
                                                         print_status=True,
                                                         tagline=False)
```

Write *expr* to pdf *file_name*:

```
abjad> note = Note("c'4")
abjad> iotools.write_expr_to_pdf(note, 'one_note.pdf') # doctest: +SKIP
```

Write *expr* to pdf *file_name* with *template*:

```
abjad> note = Note("c'4")
abjad> iotools.write_expr_to_pdf(note, 'one_note.pdf', template='paris') # doctest: +SKIP
```

Return none.

iotools.z

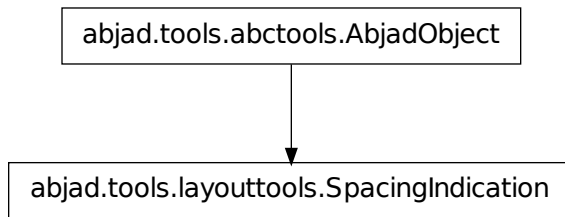
```
abjad.tools.iotools.z.z(expr)
```

New in version 2.8. Print the tools package-qualified indented repr of *z*.

layouttools

layouttools concrete classes

layouttools.SpacingIndication



class `abjad.tools.layouttools.SpacingIndication.SpacingIndication`. **SpacingIndication** (*tempo_indication*, *proportional_notation_duration*)

Spacing indication token.

LilyPond `Score.proportionalNotationDuration` will equal `proportional_notation_duration` when `tempo` equals `tempo_indication`.

```
abjad> from abjad.tools import layouttools
```

```
abjad> tempo = contexttools.TempoMark(Duration(1, 8), 44)
abjad> spacing_indication = layouttools.SpacingIndication(tempo, Duration(1, 68))
abjad> spacing_indication
SpacingIndication(TempoMark(Duration(1, 8), 44), 1/68)
```

Spacing indications are immutable.

Read-only Properties

`SpacingIndication.normalized_spacing_duration`

Read-only proportional notation duration at 60 MM.

`SpacingIndication.proportional_notation_duration`

LilyPond proportional notation duration context setting.

`SpacingIndication.tempo_indication`

Abjad tempo indication object.

Special Methods

`SpacingIndication.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

SpacingIndication.__eq__(*expr*)

Spacing indications compare equal when normalized spacing durations compare equal.

SpacingIndication.__ge__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

SpacingIndication.__gt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

SpacingIndication.__hash__() \leq \Rightarrow *hash*(*x*)

Note: Inherited from `__builtin__.object`

SpacingIndication.__le__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

SpacingIndication.__lt__(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

SpacingIndication.__ne__(*expr*)

Spacing indications compare unequal when normalized spacing durations compare unequal.

SpacingIndication.__repr__()

SpacingIndication.__setattr__()

x.__setattr__('name', value) \leq \Rightarrow *x*.name = value

Note: Inherited from `__builtin__.object`

SpacingIndication.__str__() \leq \Rightarrow *str*(*x*)

Note: Inherited from `__builtin__.object`

layouttools functions

layouttools.make_spacing_vector

abjad.tools.layouttools.make_spacing_vector.**make_spacing_vector** (*basic_distance*,
mini-
mum_distance,
padding,
stretchability)

New in version 2.0. Make spacing vector:

```
abjad> from abjad.tools import layouttools
```

```
abjad> layouttools.make_spacing_vector(0, 0, 12, 0)
SchemeVector((SchemePair(('basic_distance', 0)), SchemePair(('minimum_distance', 0)), SchemePair(
```

Use to set paper block spacing attributes:

```
abjad> staff = Staff("c'8 d'8 e'8 f'8")
abjad> lilypond_file = lilypondfiletools.make_basic_lilypond_file(staff)
abjad> lilypond_file.paper_block.system_system_spacing = layouttools.make_spacing_vector(0, 0, 12, 0)
```

```
abjad> f(lilypond_file) # doctest: +SKIP
% Abjad revision 4229
% 2011-04-07 15:19
```

```
\version "2.13.44"
\include "english.ly"
\include "/abjad/trunk/abjad/cfg/abjad.scm"

\paper {
  system-system-spacing = #'((basic_distance . 0) (minimum_distance . 0) (padding . 12) (stretchability . 1))
}

\score {
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
}
```

Return scheme vector.

layouttools.set_line_breaks_cyclically_by_line_duration_ge

abjad.tools.layouttools.set_line_breaks_cyclically_by_line_duration_ge.**set_line_breaks_cyc**

Iterate *klass* instances in *expr* and accumulate prolated duration. Add line break after every total less than or

equal to *line_duration*:

```
abjad> from abjad.tools import layouttools

abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
  }
  {
    \time 2/8
    g'8
    a'8
  }
  {
    \time 2/8
    b'8
    c''8
  }
}

abjad> layouttools.set_line_breaks_cyclically_by_line_duration_ge(t, Duration(4, 8))
abjad> f(t)
\new Staff {
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
    \break
  }
  {
    \time 2/8
    g'8
    a'8
  }
  {
    \time 2/8
    b'8
    c''8
    \break
  }
}
```

Set *adjust_eol* to *True* to include a magic Scheme incantation to move end-of-line LilyPond TimeSignature and BarLine grobs to the right. Changed in version 2.0: renamed `layout.line_break_every_prolated()`

```
to layout.set_line_breaks_cyclically_by_line_duration_ge().
```

layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge

```
abjad.tools.layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge.set_line
```

Iterate *klass* instances in *expr* and accumulate duration in seconds. Add line break after every total less than or equal to *line_duration*:

```
abjad> from abjad.tools import layouttools
```

```
abjad> t = Staff(Measure((2, 8), notetools.make_repeated_notes(2)) * 4)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(t)
abjad> tempo_mark = contexttools.TempoMark(Duration(1, 8), 44, target_context = Staff)(t)
abjad> f(t)
\new Staff {
  \tempo 8=44
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
    f'8
  }
  {
    \time 2/8
    g'8
    a'8
  }
  {
    \time 2/8
    b'8
    c''8
  }
}
```

```
abjad> layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge(t, Duration(6))
abjad> f(t)
\new Staff {
  \tempo 8=44
  {
    \time 2/8
    c'8
    d'8
  }
  {
    \time 2/8
    e'8
  }
}
```

```

        f'8
        \break
    }
    {
        \time 2/8
        g'8
        a'8
    }
    {
        \time 2/8
        b'8
        c''8
    }
}

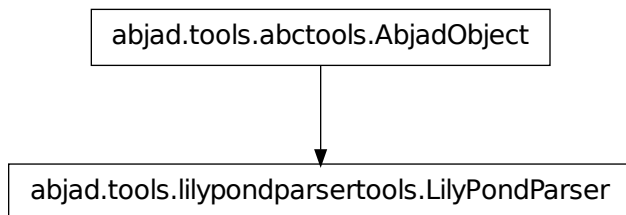
```

Set `adjust_eol = True` to include a magic Scheme incantation to move end-of-line LilyPond TimeSignature and BarLine grobs to the right. Changed in version 2.0: renamed `layout.line_break_every_seconds()` to `layout.set_line_breaks_cyclically_by_line_duration_in_seconds_ge()`.

lilypondparsertools

lilypondparsertools concrete classes

lilypondparsertools.LilyPondParser



class `abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser` (*default_language=de-bug=False*)

Parses a subset of LilyPond input syntax:

```

abjad> from abjad.tools.lilypondparsertools import LilyPondParser
abjad> parser = LilyPondParser( )
abjad> input = r"\new Staff { c'4 ( d'8 e' fs'2) \fermata }"
abjad> result = parser(input)
abjad> f(result)
\new Staff {
    c'4 (
    d'8
    e'8

```

```

    fs'2 -\fermata )
}

```

LilyPondParser defaults to English note names, but any of the other languages supported by LilyPond may be used:

```

abjad> parser = LilyPondParser('nederlands')
abjad> input = '{ c des e fis }'
abjad> result = parser(input)
abjad> f(result)
{
    c4
    df4
    e4
    fs4
}

```

Briefly, LilyPondParser understands theses aspects of LilyPond syntax:

- Notes, chords, rests, skips and multi-measure rests
- Durations, dots, and multipliers
- All pitchnames, and octave ticks
- Simple markup (i.e. `c'4 ^ "hello!"`)
- Most articulations
- Most spanners, including beams, slurs, phrasing slurs, ties, and glissandi
- Most context types via `\new` and `\context`, as well as context ids (i.e. `\new Staff = "foo" { }`)
- Variable assignment (i.e. `global = { \time 3/4 } \new Staff { \global }`)
- Many music functions:
 - `\acciaccatura`
 - `\appoggiatura`
 - `\bar`
 - `\breathe`
 - `\clef`
 - `\grace`
 - `\key`
 - `\transpose`
 - `\language`
 - `\makeClusters`
 - `\mark`
 - `\oneVoice`
 - `\relative`
 - `\skip`
 - `\slashedGrace`

- \time
- \times
- \transpose
- \voiceOne, \voiceTwo, \voiceThree, \voiceFour

LilyPondParser currently **DOES NOT** understand many other aspects of LilyPond syntax:

- \markup
- \book, \bookpart, \header, \layout, \midi and \paper
- \repeat and \alternative
- Lyrics
- \chordmode, \drummode or \figuremode
- Property operations, such as \override, \revert, \set, \unset, and \once
- Music functions which generate or extensively mutate musical structures
- Embedded Scheme statements (anything beginning with #)

Returns LilyPondParser instance.

Read-only Properties

LilyPondParser.**available_languages**

Tuple of pitch-name languages supported by LilyPondParser:

```
abjad> from abjad.tools.lilypondparsertools import LilyPondParser
abjad> parser = LilyPondParser( )
abjad> parser.available_languages
('catalan', 'deutsch', 'english', 'espanol', 'italiano', 'nederlands', 'norsk', 'portugues', 'su
```

Return tuple.

Read/write Properties

LilyPondParser.**default_language**

Read/write attribute to set parser's default pitch-name language:

```
abjad> from abjad.tools.lilypondparsertools import LilyPondParser
abjad> parser = LilyPondParser( )
abjad> parser.default_language
'english'
abjad> parser('{ c df e fs }')
{c4, df4, e4, fs4}
abjad> parser.default_language = 'nederlands'
abjad> parser.default_language
'nederlands'
abjad> parser('{ c des e fis }')
{c4, df4, e4, fs4}
```

Special Methods

LilyPondParser.**__call__**(*input_string*)

LilyPondParser.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

`LilyPondParser.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`LilyPondParser.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`LilyPondParser.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`LilyPondParser.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

mathtools

mathtools functions

mathtools.are_relatively_prime

`abjad.tools.mathtools.are_relatively_prime.are_relatively_prime(expr)`

New in version 2.5. True when *expr* is a sequence comprising zero or more numbers, all of which are relatively prime:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.are_relatively_prime([13, 14, 15])
True
```

Otherwise false:

```
abjad> mathtools.are_relatively_prime([13, 14, 15, 16])
False
```

Note that function returns true when *expr* is an empty sequence:

```
abjad> mathtools.are_relatively_prime([])
True
```

Function returns false when *expr* is nonsensical type:

```
abjad> mathtools.are_relatively_prime('foo')
False
```

Return boolean.

mathtools.arithmetic_mean

`abjad.tools.mathtools.arithmetic_mean.arithmetic_mean(sequence)`

New in version 1.1. Arithmetic means of *sequence* as an exact integer:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.arithmetic_mean([1, 2, 2, 20, 30])
11
```

As a rational:

```
abjad> mathtools.arithmetic_mean([1, 2, 20])
Fraction(23, 3)
```

As a float:

```
abjad> mathtools.arithmetic_mean([2, 2, 20.0])
8.0
```

Return number. Changed in version 2.0: renamed `sequencetools.arithmetic_mean()` to `mathtools.arithmetic_mean()`.

mathtools.binomial_coefficient

`abjad.tools.mathtools.binomial_coefficient`.**binomial_coefficient** (*n*, *k*)

New in version 2.0. Binomial coefficient of *n* choose *k*:

```
abjad> from abjad.tools import mathtools

abjad> for k in range(8):
...     print k, '\t', mathtools.binomial_coefficient(8, k)
...
0 1
1 8
2 28
3 56
4 70
5 56
6 28
7 8
```

Return positive integer.

mathtools.cumulative_products

`abjad.tools.mathtools.cumulative_products`.**cumulative_products** (*sequence*)

Cumulative products of *sequence*:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.cumulative_products([1, 2, 3, 4, 5, 6, 7, 8])
[1, 2, 6, 24, 120, 720, 5040, 40320]

abjad> mathtools.cumulative_products([1, -2, 3, -4, 5, -6, 7, -8])
[1, -2, -6, 24, 120, -720, -5040, 40320]
```

Raise type error when *sequence* is neither list nor tuple.

Raise value error on empty *sequence*.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_products()` to `mathtools.cumulative_products()`.

mathtools.cumulative_signed_weights

abjad.tools.mathtools.cumulative_signed_weights.**cumulative_signed_weights**(*sequence*)

Cumulative signed weights of *sequence*:

```
abjad> from abjad.tools import mathtools

abjad> l = [1, -2, -3, 4, -5, -6, 7, -8, -9, 10]
abjad> mathtools.cumulative_signed_weights(l)
[1, -3, -6, 10, -15, -21, 28, -36, -45, 55]
```

Raise type error when *sequence* is not a list.

For cumulative (unsigned) weights use `mathtools.cumulative_sums([abs(x) for x in l])`.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_weights_signed()` to `mathtools.cumulative_signed_weights()`.

mathtools.cumulative_sums

abjad.tools.mathtools.cumulative_sums.**cumulative_sums**(*sequence*)

Cumulative sums of *sequence*:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.cumulative_sums([1, 2, 3, 4, 5, 6, 7, 8])
[1, 3, 6, 10, 15, 21, 28, 36]
```

Raise type error when *sequence* is neither list nor tuple.

Raise value error on empty *sequence*.

Return list. Changed in version 2.0: renamed `sequencetools.cumulative_sums()` to `mathtools.cumulative_sums()`.

mathtools.cumulative_sums_zero

abjad.tools.mathtools.cumulative_sums_zero.**cumulative_sums_zero**(*sequence*)

Cumulative sums of *sequence* starting from 0:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.cumulative_sums_zero([1, 2, 3, 4, 5, 6, 7, 8])
[0, 1, 3, 6, 10, 15, 21, 28, 36]
```

Return `[0]` on empty *sequence*:

```
abjad> mathtools.cumulative_sums_zero([])
[0]
```

Return list. Changed in version 2.0: renamed `mathtools.cumulative_sums_zero()` to `mathtools.cumulative_sums_zero()`.

mathtools.cumulative_sums_zero_pairwise

abjad.tools.mathtools.cumulative_sums_zero_pairwise.**cumulative_sums_zero_pairwise**(*sequence*)

List pairwise cumulative sums of *sequence* from 0:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.cumulative_sums_zero_pairwise([1, 2, 3, 4, 5, 6])
[(0, 1), (1, 3), (3, 6), (6, 10), (10, 15), (15, 21)]
```

Return list of pairs. Changed in version 2.0: renamed `sequencetools.pairwise_cumulative_sums_zero()` to `mathtools.cumulative_sums_zero_pairwise()`.

mathtools.difference_series

`abjad.tools.mathtools.difference_series.difference_series(sequence)`

Difference series of *sequence*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.difference_series([1, 1, 2, 3, 5, 5, 6])
[0, 1, 1, 2, 0, 1]
```

Return list. Changed in version 2.0: renamed `sequencetools.difference_series()` to `mathtools.difference_series()`.

mathtools.divide_number_by_ratio

`abjad.tools.mathtools.divide_number_by_ratio.divide_number_by_ratio(number, ratio)`

Divide integer by *ratio*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.divide_number_by_ratio(1, [1, 1, 3])
[Fraction(1, 5), Fraction(1, 5), Fraction(3, 5)]
```

Divide fraction by *ratio*:

```
abjad> mathtools.divide_number_by_ratio(Fraction(1), [1, 1, 3])
[Fraction(1, 5), Fraction(1, 5), Fraction(3, 5)]
```

Divide float by *ratio*:

```
abjad> mathtools.divide_number_by_ratio(1.0, [1, 1, 3]) # doctest: +SKIP
[0.20000000000000001, 0.20000000000000001, 0.60000000000000009]
```

Raise type error on nonnumeric *number*.

Raise type error on noninteger in *ratio*.

Return list of fractions or list of floats. Changed in version 2.0: renamed `mathtools.divide_number_by_ratio()` to `mathtools.divide_number_by_ratio()`.

mathtools.divisors

`abjad.tools.mathtools.divisors.divisors(n)`

Positive divisors of integer *n* in increasing order:

```
abjad> from abjad.tools import mathtools
```

```

abjad> mathtools.divisors(84)
[1, 2, 3, 4, 6, 7, 12, 14, 21, 28, 42, 84]

abjad> for x in range(10, 20):
...     print x, mathtools.divisors(x)
...
10 [1, 2, 5, 10]
11 [1, 11]
12 [1, 2, 3, 4, 6, 12]
13 [1, 13]
14 [1, 2, 7, 14]
15 [1, 3, 5, 15]
16 [1, 2, 4, 8, 16]
17 [1, 17]
18 [1, 2, 3, 6, 9, 18]
19 [1, 19]

```

Allow nonpositive n :

```

abjad> mathtools.divisors(-27)
[1, 3, 9, 27]

```

Raise type error on noninteger n .

Raise not implemented error on 0.

Return list of positive integers.

mathtools.factors

`abjad.tools.mathtools.factors.factors(n)`
 Integer factors of positive integer n in increasing order:

```

abjad> from abjad.tools import mathtools

abjad> mathtools.factors(84)
[1, 2, 2, 3, 7]

abjad> for n in range(10, 20):
...     print n, mathtools.factors(n)
...
10 [1, 2, 5]
11 [1, 11]
12 [1, 2, 2, 3]
13 [1, 13]
14 [1, 2, 7]
15 [1, 3, 5]
16 [1, 2, 2, 2, 2]
17 [1, 17]
18 [1, 2, 3, 3]
19 [1, 19]

```

Raise type error on noninteger n .

Raise value error on nonpositive n .

Return list of one or more positive integers.

mathtools.get_shared_numeric_sign

`abjad.tools.mathtools.get_shared_numeric_sign.get_shared_numeric_sign(sequence)`

Return 1 when all *sequence* elements are positive:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.get_shared_numeric_sign([1, 2, 3])
1
```

Return -1 when all *sequence* elements are negative:

```
abjad> mathtools.get_shared_numeric_sign([-1, -2, -3])
-1
```

Return 0 on empty *sequence*:

```
abjad> mathtools.get_shared_numeric_sign([])
0
```

Otherwise return none:

```
abjad> mathtools.get_shared_numeric_sign([1, 2, -3]) is None
True
```

Return 1, -1, 0 or none. Changed in version 2.0: renamed `sequencetools.sign()` to `mathtools.get_shared_numeric_sign()`.

mathtools.greatest_common_divisor

`abjad.tools.mathtools.greatest_common_divisor.greatest_common_divisor(*integers)`

New in version 2.0. Greatest common divisor of *integers*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.greatest_common_divisor(84, -94, -144)
2
```

Allow nonpositive *integers*.

Raise type error on noninteger *integers*.

Raise not implemented error when 0 in *integers*.

Return positive integer.

mathtools.greatest_multiple_less_equal

`abjad.tools.mathtools.greatest_multiple_less_equal.greatest_multiple_less_equal(m, n)`

Greatest integer multiple of *m* less than or equal to *n*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.greatest_multiple_less_equal(10, 47)
40
```

```
abjad> for m in range(1, 10):
...     print m, mathtools.greatest_multiple_less_equal(m, 47)
...
1 47
2 46
3 45
4 44
5 45
6 42
7 42
8 40
9 45

abjad> for n in range(10, 100, 10):
...     print mathtools.greatest_multiple_less_equal(7, n), n
...
7 10
14 20
28 30
35 40
49 50
56 60
70 70
77 80
84 90
```

Raise type error on nonnumeric *m*.

Raise type error on nonnumeric *n*.

Return nonnegative integer.

mathtools.greatest_power_of_two_less_equal

`abjad.tools.mathtools.greatest_power_of_two_less_equal.greatest_power_of_two_less_equal(n, i=0)`

Greatest integer power of two less than or equal to positive *n*:

```
abjad> from abjad.tools import mathtools

abjad> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.greatest_power_of_two_less_equal(n))
...
    10 8
    11 8
    12 8
    13 8
    14 8
    15 8
    16 16
    17 16
    18 16
    19 16
```

Greatest-but-*i* integer power of 2 less than or equal to positive *n*:

```
abjad> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.greatest_power_of_two_less_equal(n, i = 1))
```

```
...
10 4
11 4
12 4
13 4
14 4
15 4
16 8
17 8
18 8
19 8
```

Raise type error on nonnumeric n .

Raise value error on nonpositive n .

Return positive integer.

`mathtools.integer_equivalent_number_to_integer`

`abjad.tools.mathtools.integer_equivalent_number_to_integer.integer_equivalent_number_to_integer`

New in version 2.0. Integer-equivalent $number$ to integer:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.integer_equivalent_number_to_integer(17.0)
17
```

Return noninteger-equivalent number unchanged:

```
abjad> mathtools.integer_equivalent_number_to_integer(17.5)
17.5
```

Raise type error on nonnumber input.

Return number.

`mathtools.integer_to_base_k_tuple`

`abjad.tools.mathtools.integer_to_base_k_tuple.integer_to_base_k_tuple(n, k)`

New in version 2.0. Nonnegative integer n to base- k tuple:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.integer_to_base_k_tuple(1066, 10)
(1, 0, 6, 6)
```

Return tuple of one or more positive integers.

`mathtools.integer_to_binary_string`

`abjad.tools.mathtools.integer_to_binary_string.integer_to_binary_string(n)`

Positive integer n to binary string:

```
abjad> from abjad.tools import mathtools
```



```

abjad> mathtools.integer_to_binary_string(5)
'101'

abjad> for n in range(1, 17):
...     print '\t%s\t%s' % (n, mathtools.integer_to_binary_string(n))
...
1 1
2 10
3 11
4 100
5 101
6 110
7 111
8 1000
9 1001
10 1010
11 1011
12 1100
13 1101
14 1110
15 1111
16 10000

```

Return string. Changed in version 2.0: renamed `mathtools.binary_string()` to `mathtools.integer_to_binary_string()`.

mathtools.interpolate_cosine

`abjad.tools.mathtools.interpolate_cosine.interpolate_cosine(y1, y2, mu)`
Cosine interpolate *y1* and *y2* with *mu* normalized [0, 1]:

```

abjad> from abjad.tools import mathtools

abjad> mathtools.interpolate_cosine(0, 1, 0.5)
0.49999999999999994

```

Return float. Changed in version 2.0: renamed `interpolate.cosine()` to `mathtools.interpolate_cosine()`.

mathtools.interpolate_divide

`abjad.tools.mathtools.interpolate_divide.interpolate_divide(total, start_frac, stop_frac, exp='cosine')`
Divide *total* into segments of sizes computed from interpolating between *start_frac* and *stop_frac*:

```

abjad> from abjad.tools import mathtools

abjad> mathtools.interpolate_divide(10, 1, 1, exp=1)
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
abjad> sum(mathtools.interpolate_divide(10, 1, 1, exp=1))
10.0

abjad> mathtools.interpolate_divide(10, 5, 1) # doctest: +SKIP
[4.7986734489043181, 2.8792040693425909, 1.3263207210948171,
0.99580176065827419]

```

```
abjad> sum(mathtools.interpolate_divide(10, 5, 1))
10.0
```

Set `exp='cosine'` for cosine interpolation.

Set `exp` to a numeric value for exponential interpolation with `exp` as the exponent.

Scale resulting segments so that their sum equals exactly *total*.

Return a list of floats. Changed in version 2.0: renamed `interpolate.divide()` to `mathtools.interpolate_divide()`.

mathtools.interpolate_divide_multiple

```
abjad.tools.mathtools.interpolate_divide_multiple.interpolate_divide_multiple(totals,
                                                                              key_values,
                                                                              exp='cosine')
```

New in version 2.0. Interpolate *key_values* such that the sum of the resulting interpolated values equals the given *totals*:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.interpolate_divide_multiple([100, 50], [20, 10, 20]) # doctest: +SKIP
[19.4487, 18.5201, 16.2270, 13.7156, 11.7488, 10.4879,
 9.8515, 9.5130, 10.4213, 13.0736, 16.9918]
```

The operation is the same as `mathtools.interpolate_divide()`. But this function takes multiple *totals* and *key_values* at once.

Precondition: `len(totals) == len(key_values) - 1`.

Set *totals* equal to a list or tuple of the total sum of interpolated values.

Set *key_values* equal to a list or tuple of key values to interpolate.

Set `exp` to *consine* for consine interpolation.

Set `exp` to a number for exponential interpolation.

Returns a list of floats. Changed in version 2.0: renamed `interpolate.divide_multiple()` to `mathtools.interpolate_divide_multiple()`.

mathtools.interpolate_exponential

```
abjad.tools.mathtools.interpolate_exponential.interpolate_exponential(y1,
                              y2,
                              mu,
                              exp=1)
```

Exponential interpolate *y1* and *y2* with *mu* normalized `[0, 1]`:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.interpolate_exponential(0, 1, 0.5, 4)
0.0625
```

Set `exp` equal to the exponent of interpolation.

Return float. Changed in version 2.0: renamed `interpolate.exponential()` to `mathtools.interpolate_exponential()`.

mathtools.interpolate_linear

abjad.tools.mathtools.interpolate_linear.**interpolate_linear**(*y1*, *y2*, *mu*)

Linear interpolate *y1* and *y2* with *mu* normalized [0, 1]:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.interpolate_linear(0, 1, 0.5)
0.5
```

Return float. Changed in version 2.0: renamed `interpolate.linear()` to `mathtools.interpolate_linear()`.

mathtools.is_assignable_integer

abjad.tools.mathtools.is_assignable_integer.**is_assignable_integer**(*expr*)

New in version 2.0. True when *expr* is equivalent to an integer and can be written without recourse to ties:

```
abjad> from abjad.tools import mathtools
```

```
abjad> for n in range(0, 16 + 1):
...     print '%s\t%s' % (n, mathtools.is_assignable_integer(n))
...
0 False
1 True
2 True
3 True
4 True
5 False
6 True
7 True
8 True
9 False
10 False
11 False
12 True
13 False
14 True
15 True
16 True
```

Otherwise false.

Return boolean. Changed in version 2.0: renamed `mathtools.is_assignable()` to `mathtools.is_assignable_integer()`.

mathtools.is_dotted_integer

abjad.tools.mathtools.is_dotted_integer.**is_dotted_integer**(*expr*)

New in version 2.0. True when *expr* is equivalent to a positive integer and can be written with zero or more dots:

```
abjad> from abjad.tools import mathtools
```

```
abjad> for expr in range(16):
...     print '%s\t%s' % (expr, mathtools.is_dotted_integer(expr))
...
```

```

0      False
1      False
2      False
3      True
4      False
5      False
6      True
7      True
8      False
9      False
10     False
11     False
12     True
13     False
14     True
15     True

```

Otherwise false.

Return boolean.

Integer n qualifies as dotted when $\text{abs}(n)$ is of the form $2^{**j} * (2^{**k} - 1)$ with integers $0 \leq j, 2 < k$.

mathtools.is_integer_equivalent_expr

`abjad.tools.mathtools.is_integer_equivalent_expr.is_integer_equivalent_expr(expr)`

New in version 2.5. True when *expr* is an integer-equivalent number:

```

abjad> from abjad.tools import mathtools

abjad> mathtools.is_integer_equivalent_expr(12.0)
True

```

True when *expr* evaluates to an integer:

```

abjad> mathtools.is_integer_equivalent_expr('12')
True

```

Otherwise false:

```

abjad> mathtools.is_integer_equivalent_expr('foo')
False

```

Return boolean.

mathtools.is_integer_equivalent_number

`abjad.tools.mathtools.is_integer_equivalent_number.is_integer_equivalent_number(expr)`

New in version 2.0. True when *expr* is a number and *expr* is equivalent to an integer:

```

abjad> from abjad.tools import mathtools

abjad> mathtools.is_integer_equivalent_number(12.0)
True

```

Otherwise false:

```
abjad> mathtools.is_integer_equivalent_number(Duration(1, 2))
False
```

Return boolean.

mathtools.is_negative_integer

`abjad.tools.mathtools.is_negative_integer.is_negative_integer(expr)`
New in version 2.0. True when *expr* equals a negative integer:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.is_negative_integer(-1)
True
```

Otherwise false:

```
abjad> mathtools.is_negative_integer(0)
False

abjad> mathtools.is_negative_integer(99)
False
```

Return boolean.

mathtools.is_nonnegative_integer

`abjad.tools.mathtools.is_nonnegative_integer.is_nonnegative_integer(expr)`
New in version 2.0. True when *expr* equals a nonnegative integer:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.is_nonnegative_integer(99)
True

abjad> mathtools.is_nonnegative_integer(0)
True
```

Otherwise false:

```
abjad> mathtools.is_nonnegative_integer(-1)
False
```

Return boolean.

mathtools.is_nonnegative_integer_equivalent_number

`abjad.tools.mathtools.is_nonnegative_integer_equivalent_number.is_nonnegative_integer_equivalent_number(expr)`
New in version 2.0. True when *expr* is a nonnegative integer-equivalent number. Otherwise false:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.is_nonnegative_integer_equivalent_number(Duration(4, 2))
True
```

Return boolean.

`mathtools.is_nonnegative_integer_power_of_two`

`abjad.tools.mathtools.is_nonnegative_integer_power_of_two.is_nonnegative_integer_power_of_two`

True when *expr* is a nonnegative integer power of 2:

```
abjad> from abjad.tools import mathtools
```

```
abjad> for n in range(10):
...     print n, mathtools.is_nonnegative_integer_power_of_two(n)
...
0 True
1 True
2 True
3 False
4 True
5 False
6 False
7 False
8 True
9 False
```

Otherwise false.

Return boolean. Changed in version 2.0: renamed `mathtools.is_power_of_two()` to `mathtools.is_nonnegative_integer_power_of_two()`.

`mathtools.is_positive_integer`

`abjad.tools.mathtools.is_positive_integer.is_positive_integer` (*expr*)

New in version 2.0. True when *expr* equals a positive integer:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.is_positive_integer(99)
True
```

Otherwise false:

```
abjad> mathtools.is_positive_integer(0)
False
```

```
abjad> mathtools.is_positive_integer(-1)
False
```

Return boolean.

`mathtools.is_positive_integer_equivalent_number`

`abjad.tools.mathtools.is_positive_integer_equivalent_number.is_positive_integer_equivalent_number`

New in version 2.0. True when *expr* is a positive integer-equivalent number. Otherwise false:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.is_positive_integer_equivalent_number(Duration(4, 2))
True
```

Return boolean.

mathtools.is_positive_integer_power_of_two

`abjad.tools.mathtools.is_positive_integer_power_of_two.is_positive_integer_power_of_two(expr)`
True when *expr* is a positive integer power of 2:

```
abjad> from abjad.tools import mathtools

abjad> for n in range(10):
...     print n, mathtools.is_positive_integer_power_of_two(n)
...
0 False
1 True
2 True
3 False
4 True
5 False
6 False
7 False
8 True
9 False
```

Otherwise false.

Return boolean.

mathtools.least_common_multiple

`abjad.tools.mathtools.least_common_multiple.least_common_multiple(*integers)`
Least common multiple of positive *integers*:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.least_common_multiple(2, 4, 5, 10, 20)
20
```

Return positive integer.

mathtools.least_multiple_greater_equal

`abjad.tools.mathtools.least_multiple_greater_equal.least_multiple_greater_equal(m,
n)`

Return the least integer multiple of *m* greater than or equal to *n*.

```
abjad> from abjad.tools import mathtools

abjad> mathtools.least_multiple_greater_equal(10, 47)
50

abjad> for m in range(1, 10):
...     print m, mathtools.least_multiple_greater_equal(m, 47)
...
1 47
2 48
3 48
4 48
5 50
6 48
```

```

7 49
8 48
9 54

abjad> for n in range(10, 100, 10):
...     print mathtools.least_multiple_greater_equal(7, n), n
...
14 10
21 20
35 30
42 40
56 50
63 60
70 70
84 80
91 90

```

Return integer.

mathtools.least_power_of_two_greater_equal

`abjad.tools.mathtools.least_power_of_two_greater_equal.least_power_of_two_greater_equal(n, i=0)`

Return least integer power of two greater than or equal to positive n :

```

abjad> from abjad.tools import mathtools

abjad> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.least_power_of_two_greater_equal(n))
...
    10 16
    11 16
    12 16
    13 16
    14 16
    15 16
    16 16
    17 32
    18 32
    19 32

```

When $i = 1$, return the first integer power of 2 greater than the least integer power of 2 greater than or equal to n .

```

abjad> for n in range(10, 20):
...     print '\t%s\t%s' % (n, mathtools.least_power_of_two_greater_equal(n, i = 1))
...
    10 32
    11 32
    12 32
    13 32
    14 32
    15 32
    16 32
    17 64
    18 64
    19 64

```


When $i = 2$, return the second integer power of 2 greater than the least integer power of 2 greater than or equal to n , and, in general, return the i th integer power of 2 greater than the least integer power of 2 greater than or equal to n .

Raise type error on nonnumeric n .

Raise value error on nonpositive n .

Return integer.

mathtools.next_integer_partition

`abjad.tools.mathtools.next_integer_partition.next_integer_partition(integer_partition)`
New in version 2.0. Next integer partition following *integer_partition* in descending lex order:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.next_integer_partition((8, 3))
(8, 2, 1)

abjad> mathtools.next_integer_partition((8, 2, 1))
(8, 1, 1, 1)

abjad> mathtools.next_integer_partition((8, 1, 1, 1))
(7, 4)
```

Input *integer_partition* must be sequence of positive integers.

Return integer partition as tuple of positive integers.

mathtools.partition_integer_by_ratio

`abjad.tools.mathtools.partition_integer_by_ratio.partition_integer_by_ratio(n, ratio)`

Partition positive integer-equivalent n by *ratio*:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.partition_integer_by_ratio(10, [1, 2])
[3, 7]
```

Partition positive integer-equivalent n by *ratio* with negative parts:

```
abjad> mathtools.partition_integer_by_ratio(10, [1, -2])
[3, -7]
```

Partition negative integer-equivalent n by *ratio*:

```
abjad> mathtools.partition_integer_by_ratio(-10, [1, 2])
[-3, -7]
```

Partition negative integer-equivalent n by *ratio* with negative parts:

```
abjad> mathtools.partition_integer_by_ratio(-10, [1, -2])
[-3, 7]
```

Return result with weight equal to absolute value of n .

Raise type error on noninteger n .

Return list of integers.

`mathtools.partition_integer_into_canonic_parts`

`abjad.tools.mathtools.partition_integer_into_canonic_parts.partition_integer_into_canonic_parts`

Partition integer n into big-endian or small-endian parts.

Return all parts positive on positive n :

```
abjad> from abjad.tools import mathtools

abjad> for n in range(1, 11):
...     print n, mathtools.partition_integer_into_canonic_parts(n)
...
1 (1,)
2 (2,)
3 (3,)
4 (4,)
5 (4, 1)
6 (6,)
7 (7,)
8 (8,)
9 (8, 1)
10 (8, 2)
```

Return all parts negative on negative n :

```
abjad> for n in reversed(range(-20, -10)):
...     print n, mathtools.partition_integer_into_canonic_parts(n)
...
-11 (-8, -3)
-12 (-12,)
-13 (-12, -1)
-14 (-14,)
-15 (-15,)
-16 (-16,)
-17 (-16, -1)
-18 (-16, -2)
-19 (-16, -3)
-20 (-16, -4)
```

Return little-endian tuple When `direction = 'little-endian'`:

```
abjad> for n in range(11, 21):
...     print n, mathtools.partition_integer_into_canonic_parts(n, direction = 'little-endian')
...
11 (3, 8)
12 (12,)
13 (1, 12)
14 (14,)
15 (15,)
16 (16,)
17 (1, 16)
```

```
18 (2, 16)
19 (3, 16)
20 (4, 16)
```

Return big-endian tuple $t = (t_0, \dots, t_j)$ such that

- $\text{sum}(t) == n$
- t_i can be written without recourse to ties, and
- $t_{i+1} < t_i$ for every t_i in t .

Raise type error on noninteger n .

Return tuple of one or more integers.

mathtools.partition_integer_into_halves

abjad.tools.mathtools.partition_integer_into_halves.**partition_integer_into_halves**(n ,
bigger='left',
even='allow'

Write positive integer n as the pair $t = (\text{left}, \text{right})$ such that $n == \text{left} + \text{right}$.

When n is odd the greater part of t corresponds to the value of *bigger*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.partition_integer_into_halves(7, bigger = 'left')
(4, 3)
abjad> mathtools.partition_integer_into_halves(7, bigger = 'right')
(3, 4)
```

Likewise when n is even and *even* = 'disallowed':

```
abjad> mathtools.partition_integer_into_halves(8, bigger = 'left', even = 'disallowed')
(5, 3)
abjad> mathtools.partition_integer_into_halves(8, bigger = 'right', even = 'disallowed')
(3, 5)
```

But when n is even and *even* = 'allowed' then $\text{left} == \text{right}$ and *bigger* is ignored:

```
abjad> mathtools.partition_integer_into_halves(8)
(4, 4)
abjad> mathtools.partition_integer_into_halves(8, bigger = 'left')
(4, 4)
abjad> mathtools.partition_integer_into_halves(8, bigger = 'right')
(4, 4)
```

When n is 0 return (0, 0):

```
abjad> mathtools.partition_integer_into_halves(0)
(0, 0)
```

When n is 0 and *even* = 'disallowed' raise partition error.

Raise type error on noninteger n .

Raise value error on negative n .

Return pair of positive integers.

mathtools.partition_integer_into_thirds

abjad.tools.mathtools.partition_integer_into_thirds.**partition_integer_into_thirds**(*n*, *smallest='middle'*, *biggest='middle'*)

Partition positive integer *n* into left, middle, right parts.

When $n \% 3 == 0$, $\text{left} == \text{middle} == \text{right}$:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.partition_integer_into_thirds(9)
(3, 3, 3)
```

When $n \% 3 == 1$, set biggest part to *biggest*:

```
abjad> mathtools.partition_integer_into_thirds(10, biggest = 'left')
(4, 3, 3)
abjad> mathtools.partition_integer_into_thirds(10, biggest = 'middle')
(3, 4, 3)
abjad> mathtools.partition_integer_into_thirds(10, biggest = 'right')
(3, 3, 4)
```

When $n \% 3 == 2$, set smallest part to *smallest*:

```
abjad> mathtools.partition_integer_into_thirds(11, smallest = 'left')
(3, 4, 4)
abjad> mathtools.partition_integer_into_thirds(11, smallest = 'middle')
(4, 3, 4)
abjad> mathtools.partition_integer_into_thirds(11, smallest = 'right')
(4, 4, 3)
```

Raise type error on noninteger *n*.

Raise value error on nonpositive *n*.

Return triple of positive integers.

mathtools.partition_integer_into_units

abjad.tools.mathtools.partition_integer_into_units.**partition_integer_into_units**(*n*)

Partition positive integer into units:

```
abjad> from abjad.tools import mathtools

abjad> mathtools.partition_integer_into_units(6)
[1, 1, 1, 1, 1, 1]
```

Partition negative integer into units:

```
abjad> mathtools.partition_integer_into_units(-5)
[-1, -1, -1, -1, -1]
```

Partition 0 into units:

```
abjad> mathtools.partition_integer_into_units(0)
[]
```

Return list of zero or more parts with absolute value equal to 1.

mathtools.remove_powers_of_two

`abjad.tools.mathtools.remove_powers_of_two.remove_powers_of_two(n)`

Remove powers of 2 from the factors of positive integer *n*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> for n in range(10, 100, 10):
...     print '\t%s\t%s' % (n, mathtools.remove_powers_of_two(n))
...
    10 5
    20 5
    30 15
    40 5
    50 25
    60 15
    70 35
    80 5
    90 45
```

Raise type error on noninteger *n*.

Raise value error on nonpositive *n*.

Return positive integer.

mathtools.sign

`abjad.tools.mathtools.sign.sign(n)`

Return -1 on negative *n*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.sign(-96.2)
-1
```

Return 0 when *n* is 0:

```
abjad> mathtools.sign(0)
0
```

Return 1 on positive *n*:

```
abjad> mathtools.sign(Duration(9, 8))
1
```

Return -1, 0 or 1.

mathtools.weight

`abjad.tools.mathtools.weight.weight(sequence)`

Sum of the absolute value of the elements in *sequence*:

```
abjad> from abjad.tools import mathtools
```

```
abjad> mathtools.weight([-1, -2, 3, 4, 5])
15
```

Return nonnegative integer. Changed in version 2.0: renamed `sequencetools.weight()` to `mathtools.weight()`.

`mathtools.yield_all_compositions_of_integer`

`abjad.tools.mathtools.yield_all_compositions_of_integer.yield_all_compositions_of_integer(n)`
 New in version 2.0. Yield all compositions of positive integer n in descending lex order:

```
abjad> from abjad.tools import mathtools

abjad> for integer_composition in mathtools.yield_all_compositions_of_integer(5):
...     integer_composition
...
(5,)
(4, 1)
(3, 2)
(3, 1, 1)
(2, 3)
(2, 2, 1)
(2, 1, 2)
(2, 1, 1, 1)
(1, 4)
(1, 3, 1)
(1, 2, 2)
(1, 2, 1, 1)
(1, 1, 3)
(1, 1, 2, 1)
(1, 1, 1, 2)
(1, 1, 1, 1, 1)
```

Integer compositions are ordered integer partitions.

Return generator of positive integer tuples of length at least 1. Changed in version 2.0: renamed `mathtools.integer_compositions()` to `mathtools.yield_all_compositions_of_integer()`.

`mathtools.yield_all_partitions_of_integer`

`abjad.tools.mathtools.yield_all_partitions_of_integer.yield_all_partitions_of_integer(n)`
 New in version 2.0. Yield all partitions of positive integer n in descending lex order:

```
abjad> from abjad.tools import mathtools

abjad> for partition in mathtools.yield_all_partitions_of_integer(7):
...     partition
...
(7,)
(6, 1)
(5, 2)
(5, 1, 1)
(4, 3)
(4, 2, 1)
(4, 1, 1, 1)
(3, 3, 1)
(3, 2, 2)
(3, 2, 1, 1)
```

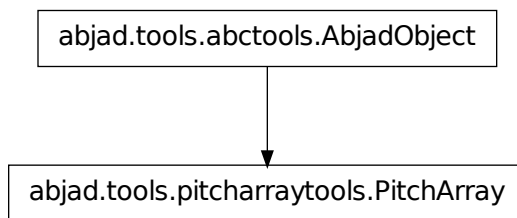
```
(3, 1, 1, 1, 1)
(2, 2, 2, 1)
(2, 2, 1, 1, 1)
(2, 1, 1, 1, 1, 1)
(1, 1, 1, 1, 1, 1, 1)
```

Return generator of positive integer tuples of length at least 1. Changed in version 2.0: renamed `mathtools.integer_partitions()` to `mathtools.yield_all_partitions_of_integer()`.

pitcharraytools

pitcharraytools concrete classes

pitcharraytools.PitchArray



class `abjad.tools.pitcharraytools.PitchArray.PitchArray(*args)`
 New in version 2.0. Two-dimensional array of pitches.

Read-only Properties

```
PitchArray.cell_tokens_by_row
PitchArray.cell_widths_by_row
PitchArray.cells
PitchArray.columns
PitchArray.depth
PitchArray.dimensions
PitchArray.has_voice_crossing
PitchArray.is_rectangular
PitchArray.pitches
PitchArray.pitches_by_row
PitchArray.rows
PitchArray.size
```

PitchArray.**voice_crossing_count**

PitchArray.**weight**

PitchArray.**width**

Methods

PitchArray.**append_column** (*column*)

PitchArray.**append_row** (*row*)

PitchArray.**apply_pitches_by_row** (*pitch_lists*)

PitchArray.**copy_subarray** (*upper_left_pair*, *lower_right_pair*)

PitchArray.**has_spanning_cell_over_index** (*index*)

PitchArray.**pad_to_depth** (*depth*)

PitchArray.**pad_to_width** (*width*)

PitchArray.**pop_column** (*column_index*)

PitchArray.**pop_row** (*row_index=-1*)

PitchArray.**remove_row** (*row*)

Special Methods

PitchArray.**__add__** (*arg*)

PitchArray.**__contains__** (*arg*)

PitchArray.**__delattr__** ()

x.**__delattr__** ('name') <==> del *x*.name

Note: Inherited from `__builtin__.object`

PitchArray.**__eq__** (*arg*)

PitchArray.**__ge__** (*arg*)

 Abjad objects by default do not implement this method.

 Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArray.**__getitem__** (*arg*)

PitchArray.**__gt__** (*arg*)

 Abjad objects by default do not implement this method.

 Raise exception

Note: Inherited from `abctools.AbjadObject`

PitchArray.**__hash__** () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

PitchArray.__iadd__(arg)

PitchArray.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArray.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArray.__ne__(arg)

PitchArray.__repr__()

PitchArray.__setattr__()

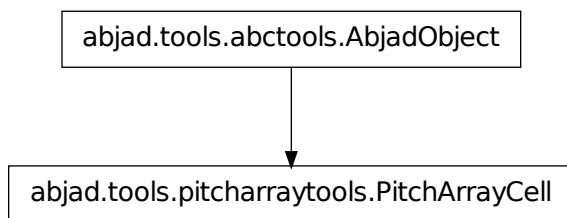
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

PitchArray.__setitem__(i, arg)

PitchArray.__str__()

`pitcharraytools.PitchArrayCell`



class `abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell`(*cell_token=None*)

One cell in a pitch array.

```
abjad> from abjad.tools import pitcharraytools
```

```
abjad> array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
```

```
abjad> print array
```

```
[ ] [   ] [ ]
```

```
[   ] [ ] [ ]
```

```
abjad> cell = array[0][1]
```

```

abjad> cell
PitchArrayCell(x2)

abjad> cell.column_indices
(1, 2)

abjad> cell.indices
(0, (1, 2))

abjad> cell.is_first_in_row
False

abjad> cell.is_last_in_row
False

abjad> cell.next
PitchArrayCell(x1)

abjad> cell.parent_array
PitchArray(PitchArrayRow(x1, x2, x1), PitchArrayRow(x2, x1, x1))

abjad> cell.parent_column
PitchArrayColumn(x2, x2)

abjad> cell.parent_row
PitchArrayRow(x1, x2, x1)

abjad> cell.pitches
[]

abjad> cell.prev
PitchArrayCell(x1)

abjad> cell.row_index
0

abjad> cell.token
2

abjad> cell.width
2

```

Return pitch array cell.

Read-only Properties

`PitchArrayCell.column_indices`

Read-only tuple of one or more nonnegative integer indices.

`PitchArrayCell.indices`

`PitchArrayCell.is_first_in_row`

`PitchArrayCell.is_last_in_row`

`PitchArrayCell.next`

`PitchArrayCell.parent_array`

`PitchArrayCell.parent_column`

PitchArrayCell.**parent_row**

PitchArrayCell.**prev**

PitchArrayCell.**row_index**

PitchArrayCell.**token**

PitchArrayCell.**weight**

PitchArrayCell.**width**

Read/write Properties

PitchArrayCell.**pitches**

Methods

PitchArrayCell.**matches_cell**(*arg*)

Special Methods

PitchArrayCell.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

PitchArrayCell.**__eq__**(*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

PitchArrayCell.**__ge__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayCell.**__gt__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

PitchArrayCell.**__hash__**() <==> *hash*(x)

Note: Inherited from `__builtin__.object`

PitchArrayCell.**__le__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PitchArrayCell.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PitchArrayCell.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

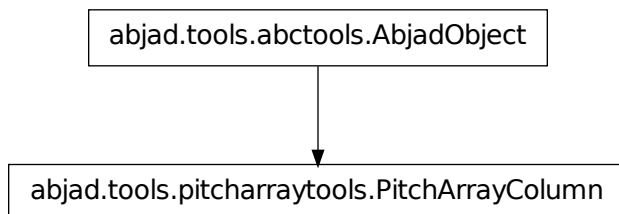
Note: Inherited from `abctools.AbjadObject`

`PitchArrayCell.__repr__()`
`PitchArrayCell.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`PitchArrayCell.__str__()`

`pitcharraytools.PitchArrayColumn`



class `abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn(cells)`
 New in version 2.0. Column in a pitch array:

```

abjad> from abjad.tools import pitcharraytools

abjad> array = pitcharraytools.PitchArray([
...     [1, (2, 1), (-1.5, 2)],
...     [(7, 2), (6, 1), 1]])
  
```

```
abjad> print array
[ ] [d'] [bqf ]
[g'    ] [fs'] [ ]

abjad> array.columns[0]
PitchArrayColumn(x1, g' x2)

abjad> print array.columns[0]
[ ]
[g'    ]
```

Return pitch array column.

Read-only Properties

`PitchArrayColumn.cell_tokens`
`PitchArrayColumn.cell_widths`
`PitchArrayColumn.cells`
`PitchArrayColumn.column_index`
`PitchArrayColumn.depth`
`PitchArrayColumn.dimensions`
`PitchArrayColumn.has_voice_crossing`
`PitchArrayColumn.is_defective`
`PitchArrayColumn.parent_array`
`PitchArrayColumn.pitches`
`PitchArrayColumn.start_cells`
`PitchArrayColumn.start_pitches`
`PitchArrayColumn.stop_cells`
`PitchArrayColumn.stop_pitches`
`PitchArrayColumn.weight`
`PitchArrayColumn.width`

Methods

`PitchArrayColumn.append(cell)`
`PitchArrayColumn.extend(cells)`
`PitchArrayColumn.remove_pitches()`

Special Methods

`PitchArrayColumn.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`PitchArrayColumn.__eq__(arg)`

PitchArrayColumn.__ge__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayColumn.__getitem__(arg)

PitchArrayColumn.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

PitchArrayColumn.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

PitchArrayColumn.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayColumn.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayColumn.__ne__(arg)

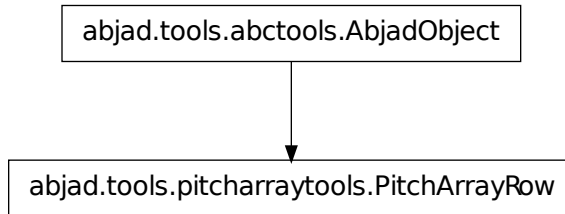
PitchArrayColumn.__repr__()

PitchArrayColumn.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

PitchArrayColumn.__str__()

pitcharraytools.PitchArrayRow

class `abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow(cells)`
 New in version 2.0. One row in pitch array.

```

abjad> from abjad.tools import pitcharraytools

abjad> array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
abjad> array[0].cells[0].pitches.append(0)
abjad> array[0].cells[1].pitches.append(2)
abjad> array[1].cells[2].pitches.append(4)
abjad> print array
[c'] [d'   ] [  ]
[      ] [  ] [e']

abjad> array[0]
PitchArrayRow(c', d' x2, x1)

abjad> array[0].cell_widths
(1, 2, 1)

abjad> array[0].dimensions
(1, 4)

abjad> array[0].pitches
(NamedChromaticPitch("c'"), NamedChromaticPitch("d'"))
  
```

Return pitch array row.

Read-only Properties

`PitchArrayRow.cell_tokens`
`PitchArrayRow.cell_widths`
`PitchArrayRow.cells`
`PitchArrayRow.depth`
`PitchArrayRow.dimensions`
`PitchArrayRow.is_defective`
`PitchArrayRow.is_in_range`
`PitchArrayRow.parent_array`

`PitchArrayRow.pitches`
`PitchArrayRow.row_index`
`PitchArrayRow.weight`
`PitchArrayRow.width`

Read/write Properties

`PitchArrayRow.pitch_range`

Methods

`PitchArrayRow.append` (*cell_token*)
`PitchArrayRow.apply_pitches` (*pitch_tokens*)
`PitchArrayRow.copy_subrow` (*start=None, stop=None*)
`PitchArrayRow.empty_pitches` ()
`PitchArrayRow.extend` (*cell_tokens*)
`PitchArrayRow.has_spanning_cell_over_index` (*i*)
`PitchArrayRow.index` (*cell*)
`PitchArrayRow.merge` (*cells*)
`PitchArrayRow.pad_to_width` (*width*)
`PitchArrayRow.pop` (*cell_index*)
`PitchArrayRow.remove` (*cell*)
`PitchArrayRow.withdraw` ()

Special Methods

`PitchArrayRow.__add__` (*arg*)
`PitchArrayRow.__delattr__` ()
 x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

`PitchArrayRow.__eq__` (*arg*)
`PitchArrayRow.__ge__` (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`PitchArrayRow.__getitem__` (*arg*)
`PitchArrayRow.__gt__` (*arg*)
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

PitchArrayRow.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

PitchArrayRow.__iadd__(arg)

PitchArrayRow.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayRow.__len__()

PitchArrayRow.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

PitchArrayRow.__ne__(arg)

PitchArrayRow.__repr__()

PitchArrayRow.__setattr__()

x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

PitchArrayRow.__str__()

pitcharraytools functions

pitcharraytools.all_are_pitch_arrays

`abjad.tools.pitcharraytools.all_are_pitch_arrays.all_are_pitch_arrays(expr)`

New in version 2.6. True when *expr* is a sequence of Abjad pitch arrays:

```
abjad> from abjad.tools import pitcharraytools
```

```
abjad> pitch_array = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
```

```
abjad> pitcharraytools.all_are_pitch_arrays([pitch_array])
```

```
True
```

True when *expr* is an empty sequence:

```
abjad> pitcharraytools.all_are_pitch_arrays([])
```

```
True
```

Otherwise false:

```
abjad> pitcharraytools.all_are_pitch_arrays('foo')
False
```

Return boolean.

`pitcharraytools.concatenate_pitch_arrays`

`abjad.tools.pitcharraytools.concatenate_pitch_arrays.concatenate_pitch_arrays(pitch_arrays)`
 New in version 2.0. Concatenate *pitch_arrays*:

```
abjad> from abjad.tools import pitcharraytools

abjad> array_1 = pitcharraytools.PitchArray([[1, 2, 1], [2, 1, 1]])
abjad> print array_1
[ ] [   ] [ ]
[   ] [ ] [ ]

abjad> array_2 = pitcharraytools.PitchArray([[3, 4], [4, 3]])
abjad> print array_2
[   ] [   ]
[   ] [   ]

abjad> array_3 = pitcharraytools.PitchArray([[1, 1], [1, 1]])
abjad> print array_3
[ ] [ ]
[ ] [ ]

abjad> merged_array = pitcharraytools.concatenate_pitch_arrays([array_1, array_2, array_3])
abjad> print merged_array
[ ] [   ] [ ] [   ] [   ] [ ] [ ] [ ]
[   ] [ ] [ ] [   ] [   ] [ ] [ ] [ ]
```

Return pitch array.

`pitcharraytools.list_nonspanning_subarrays_of_pitch_array`

`abjad.tools.pitcharraytools.list_nonspanning_subarrays_of_pitch_array.list_nonspanning_subarrays_of_pitch_array(pitch_array)`
 New in version 2.0. List nonspanning subarrays of *pitch_array*:

```
abjad> from abjad.tools import pitcharraytools

abjad> array = pitcharraytools.PitchArray([
...     [2, 2, 3, 1],
...     [1, 2, 1, 1, 2, 1],
...     [1, 1, 1, 1, 1, 1, 1, 1]])
abjad> print array
[   ] [   ] [   ] [ ]
[ ] [   ] [ ] [ ] [   ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

abjad> subarrays = pitcharraytools.list_nonspanning_subarrays_of_pitch_array(array)
abjad> len(subarrays)
3
```

```

abjad> print subarrays[0]
[      ] [      ]
[ ] [      ] [ ]
[ ] [ ] [ ] [ ]

abjad> print subarrays[1]
[      ]
[ ] [      ]
[ ] [ ] [ ]

abjad> print subarrays[2]
[ ]
[ ]
[ ]

```

Return list.

pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists

`abjad.tools.pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists`.**make_empty_pitch_array**

New in version 2.0. Make empty pitch array from *leaf_iterables*:

```

abjad> from abjad.tools import pitcharraytools

abjad> score = Score([])
abjad> score.append(Staff("c'8 d'8 e'8 f'8"))
abjad> score.append(Staff("c'4 d'4"))
abjad> score.append(Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2))
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'4
    d'4
  }
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      c'8
      d'8
      e'8
    }
  }
>>

abjad> array = pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists(score)
abjad> print array
[      ] [      ] [      ] [      ]

```

```
[          ] [          ]
[ ] [    ] [ ] [ ] [    ] [ ]
```

Return pitch array.

`pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists`

`abjad.tools.pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists`.**make_populated**

New in version 2.0. Make populated pitch array from *leaf_iterables*:

```
abjad> from abjad.tools import pitcharraytools
```

```
abjad> score = Score([])
abjad> score.append(Staff("c'8 d'8 e'8 f'8"))
abjad> score.append(Staff("c'4 d'4"))
abjad> score.append(Staff(tuplettools.FixedDurationTuplet(Duration(2, 8), "c'8 d'8 e'8") * 2))
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8
    e'8
    f'8
  }
  \new Staff {
    c'4
    d'4
  }
  \new Staff {
    \times 2/3 {
      c'8
      d'8
      e'8
    }
    \times 2/3 {
      c'8
      d'8
      e'8
    }
  }
>>
```

```
abjad> array = pitcharraytools.make_populated_pitch_array_from_list_of_pitch_lists(score)
abjad> print array
[c'      ] [d'      ] [e'      ] [f'      ]
[c'      ] [d'      ]
[c'] [d'      ] [e'] [c'] [d'      ] [e']
```

Return pitch array.

rhythmtreertools

rhythmtreertools functions

rhythmtreertools.parse_rtm_syntax

`abjad.tools.rhythmtreertools.parse_rtm_syntax.parse_rtm_syntax(rtm)`

Parse RTM syntax:

```
abjad> from abjad.tools.rhythmtreertools import parse_rtm_syntax
```

```
abjad> rtm = '(1 (1 (1 (1 1)) 1))'
```

```
abjad> result = parse_rtm_syntax(rtm)
```

```
abjad> result
```

```
FixedDurationTuplet(1/4, [c'8, c'16, c'16, c'8])
```

Returns *FixedDurationTuplet* or *Container* instance.

sequencetools

sequencetools concrete classes

sequencetools.CyclicList

`abjad.tools.sequencetools.CyclicList`

class `abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList`

New in version 2.0. Abjad model of cyclic list:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> cyclic_list = sequencetools.CyclicList('abcd')
```

```
abjad> cyclic_list
```

```
CyclicList([a, b, c, d])
```

```
abjad> for x in range(8):
...     print x, cyclic_list[x]
```

```
...
```

```
0 a
```

```
1 b
```

```
2 c
```

```
3 d
```

```
4 a
```

```
5 b
```

```
6 c
```

```
7 d
```

Cyclic lists overload the item-getting method of built-in lists.

Cyclic lists return a value for any integer index.

Cyclic lists otherwise behave exactly like built-in lists.

Methods

`CyclicList.append()`

`L.append(object)` – append object to end

Note: Inherited from `__builtin__.list`

`CyclicList.count(value)` → integer – return number of occurrences of value

Note: Inherited from `__builtin__.list`

`CyclicList.extend()`

`L.extend(iterable)` – extend list by appending elements from the iterable

Note: Inherited from `__builtin__.list`

`CyclicList.index(value[, start[, stop]])` → integer – return first index of value.

Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`CyclicList.insert()`

`L.insert(index, object)` – insert object before index

Note: Inherited from `__builtin__.list`

`CyclicList.pop([index])` → item – remove and return item at index (default last).

Raises `IndexError` if list is empty or index is out of range.

Note: Inherited from `__builtin__.list`

`CyclicList.remove()`

`L.remove(value)` – remove first occurrence of value. Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.list`

`CyclicList.reverse()`

`L.reverse()` – reverse *IN PLACE*

Note: Inherited from `__builtin__.list`

`CyclicList.sort()`

`L.sort(cmp=None, key=None, reverse=False)` – stable sort *IN PLACE*; `cmp(x, y) -> -1, 0, 1`

Note: Inherited from `__builtin__.list`

Special Methods

`CyclicList.__add__()`
`x.__add__(y) <==> x+y`

Note: Inherited from `__builtin__.list`

`CyclicList.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.list`

`CyclicList.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`CyclicList.__delitem__()`
`x.__delitem__(y) <==> del x[y]`

Note: Inherited from `__builtin__.list`

`CyclicList.__delslice__()`
`x.__delslice__(i, j) <==> del x[i:j]`
Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`CyclicList.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.list`

`CyclicList.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.list`

`CyclicList.__getitem__(expr)`
`CyclicList.__getslice__(start_index, stop_index)`
`CyclicList.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.list`

`CyclicList.__iadd__()`
`x.__iadd__(y) <==> x+=y`

Note: Inherited from `__builtin__.list`

`CyclicList.__imul__()`
`x.__imul__(y) <==> x*=y`

Note: Inherited from `__builtin__.list`

`CyclicList.__iter__()` $\langle == \rangle$ `iter(x)`

Note: Inherited from `__builtin__.list`

`CyclicList.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.list`

`CyclicList.__len__()` $\langle == \rangle$ `len(x)`

Note: Inherited from `__builtin__.list`

`CyclicList.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.list`

`CyclicList.__mul__()`
`x.__mul__(n) <==> x*n`

Note: Inherited from `__builtin__.list`

`CyclicList.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.list`

`CyclicList.__repr__()`

`CyclicList.__reversed__()`
`L.__reversed__()` – return a reverse iterator over the list

Note: Inherited from `__builtin__.list`

`CyclicList.__rmul__()`
`x.__rmul__(n) <==> n*x`

Note: Inherited from `__builtin__.list`

`CyclicList.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`CyclicList.__setitem__()`
`x.__setitem__(i, y) <==> x[i]=y`

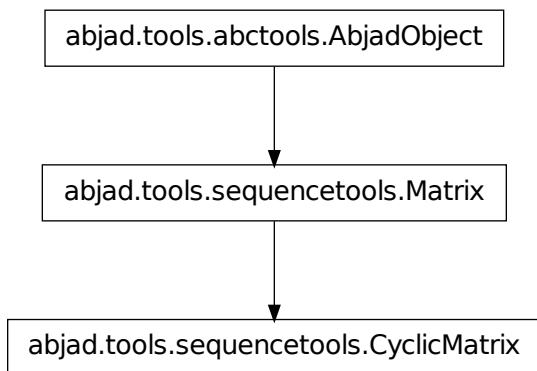
Note: Inherited from `__builtin__.list`

`CyclicList.__setslice__()`
`x.__setslice__(i, j, y) <==> x[i:j]=y`
 Use of negative indices is not supported.

Note: Inherited from `__builtin__.list`

`CyclicList.__str__()`

sequencetools.CyclicMatrix



class `abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix(*args, **kwargs)`

New in version 2.0. Abjad model of cyclic matrix.

Initialize from rows:

```
abjad> from abjad.tools import sequencetools
```

```

abjad> cyclic_matrix = sequencetools.CyclicMatrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22,
abjad> cyclic_matrix
CyclicMatrix(3x4)
abjad> cyclic_matrix[2]
CyclicTuple([20, 21, 22, 23])
abjad> cyclic_matrix[2][2]
22
abjad> cyclic_matrix[99]
CyclicTuple([0, 1, 2, 3])
abjad> cyclic_matrix[99][99]
3

```

Initialize from columns:

```

abjad> cyclic_matrix = sequencetools.CyclicMatrix(columns = [[0, 10, 20], [1, 11, 21], [2, 12, 22],
abjad> cyclic_matrix
CyclicMatrix(3x4)
abjad> cyclic_matrix[2]
CyclicTuple([20, 21, 22, 23])
abjad> cyclic_matrix[2][2]
22
abjad> cyclic_matrix[99]
CyclicTuple([0, 1, 2, 3])
abjad> cyclic_matrix[99][99]
3

```

CyclicMatrix implements only item retrieval in this revision.

Concatenation and division remain to be implemented.

Standard transforms of linear algebra remain to be implemented.

Read-only Properties

CyclicMatrix.columns

Read-only columns:

```

abjad> cyclic_matrix = sequencetools.CyclicMatrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22,
abjad> cyclic_matrix.columns
CyclicTuple([[0, 10, 20], [1, 11, 21], [2, 12, 22], [3, 13, 23]])

```

Return cyclic tuple.

CyclicMatrix.rows

Read-only rows:

```

abjad> cyclic_matrix = sequencetools.CyclicMatrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22,

```

```
abjad> cyclic_matrix.rows
CyclicTuple([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

Return cyclic tuple.

Special Methods

`CyclicMatrix.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`CyclicMatrix.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`CyclicMatrix.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicMatrix.__getitem__(expr)`
`CyclicMatrix.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`CyclicMatrix.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`CyclicMatrix.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicMatrix.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicMatrix.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

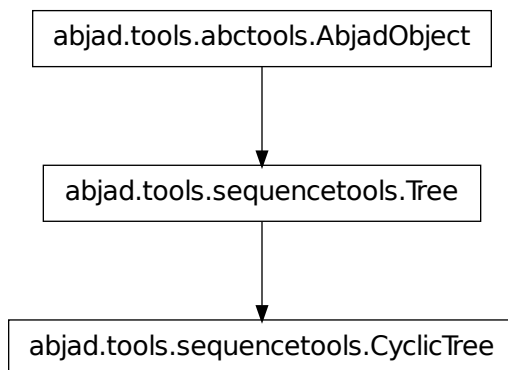
`CyclicMatrix.__repr__()`
`CyclicMatrix.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`CyclicMatrix.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`sequencetools.CyclicTree`



class `abjad.tools.sequencetools.CyclicTree.CyclicTree`(*expr*)
 New in version 2.5. Like `Tree` but with cyclic s Abjad data structure to work with a sequence whose elements have been grouped into arbitrarily many levels of **cyclic** containment.

Exactly like the `Tree` class but with the additional affordance that all integer indices of any size work at every level of structure; like `CyclicTuple`, `CyclicList` and `CyclicMatrix`, no index errors raises in working with objects of this class.

```
abjad> from abjad.tools import sequencetools
```

Here is a cyclic tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> cyclic_tree = sequencetools.CyclicTree(sequence)
```

```
abjad> cyclic_tree
CyclicTree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

Here's an internal node:

```
abjad> cyclic_tree[2]
CyclicTree([4, 5])
```

Here's the same node indexed with a different way:

```
abjad> cyclic_tree[2]
CyclicTree([4, 5])
```

With a negative index:

```
abjad> cyclic_tree[-2]
CyclicTree([4, 5])
```

And another negative index:

```
abjad> cyclic_tree[-6]
CyclicTree([4, 5])
```

Here's a leaf node:

```
abjad> cyclic_tree[2][0]
CyclicTree(4)
```

And here's the same node indexed a different way:

```
abjad> cyclic_tree[2][20]
CyclicTree(4)
```

All other interface attributes function as in `Tree`.

Read-only Properties

`CyclicTree.children`

New in version 2.4. Children of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].children
(Tree(2), Tree(3))
```

Return tuple of zero or more nodes.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.depth`

New in version 2.4. Depth of subtree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].depth
2
```

Return nonnegative integer.

Note: Inherited from `sequencetools.Tree`

CyclicTree.improper_parentage

New in version 2.4. Improper parentage of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].improper_parentage
(Tree([2, 3]), Tree([[0, 1], [2, 3], [4, 5], [6, 7]]))
```

Return tuple of one or more nodes.

Note: Inherited from `sequencetools.Tree`

CyclicTree.index_in_parent

New in version 2.4. Index of node in parent:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].index_in_parent
1
```

Return nonnegative integer.

Note: Inherited from `sequencetools.Tree`

CyclicTree.level

New in version 2.4. Level of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].level
1
```

Return nonnegative integer.

Note: Inherited from `sequencetools.Tree`

CyclicTree.negative_level

New in version 2.4. Negative level of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].negative_level
-2
```

Return negative integer.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.position`

New in version 2.4. Position of node relative to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].position
(1,)
```

Return tuple of zero or more nonnegative integers.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.proper_parentage`

New in version 2.4. Proper parentage of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return tuple of zero or more nodes.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.root`

New in version 2.4. Root of tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return node.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.width`

New in version 2.4. Number of leaves in subtree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].width
2
```

Return nonnegative integer.

Note: Inherited from `sequencetools.Tree`

Methods

`CyclicTree.get_next_n_complete_nodes_at_level` (*n*, *level*)

New in version 2.5. Get next *n* complete nodes at *level* from node.

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

Trim first node if necessary.

Return list of nodes.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.get_next_n_nodes_at_level` (*n*, *level*)

New in version 2.4. Get next *n* nodes at *level* from node.

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
abjad> tree[0][0].get_next_n_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
abjad> tree[0][0].get_next_n_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Get next node at level 0:

```
abjad> tree[0][0].get_next_n_nodes_at_level(1, 0)
[Tree([1], [2, 3], [4, 5], [6, 7])]
```


With negative *level*:

Get next 4 nodes at level -1:

```
abjad> tree[0][0].get_next_n_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
abjad> tree[0][0].get_next_n_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Trim first node if necessary.

Return list of nodes.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.get_node_at_position` (*position*)

New in version 2.4. Get node at *position*:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree.get_node_at_position((2, 1))
Tree(5)
```

Return node.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.get_position_of_descendant` (*descendant*)

New in version 2.4. Get position of *descendent* relative to node rather than relative to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[3].get_position_of_descendant(tree[3][0])
(0,)
```

Return tuple of zero or more nonnegative integers.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.is_at_level` (*level*)

New in version 2.4. True when node is at *level* in tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1][1].is_at_level(-1)
True
```

False otherwise:

```
abjad> tree[1][1].is_at_level(0)
False
```

Return boolean.

Predicate works for positive, negative and zero-valued *level*.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.iterate_at_level` (*level*)

New in version 2.4. Iterate depth at *level*:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> for x in tree.iterate_at_level(0): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

```
abjad> for x in tree.iterate_at_level(1): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
abjad> for x in tree.iterate_at_level(2): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
abjad> for x in tree.iterate_at_level(-1): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
abjad> for x in tree.iterate_at_level(-2): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
abjad> for x in tree.iterate_at_level(-3): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

Return node generator.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.iterate_depth_first()`

New in version 2.4. Iterate tree depth-first:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> for node in tree.iterate_depth_first(): node
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
Tree([0, 1])
Tree(0)
Tree(1)
Tree([2, 3])
Tree(2)
Tree(3)
Tree([4, 5])
Tree(4)
Tree(5)
Tree([6, 7])
Tree(6)
Tree(7)
```

Return node generator.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.iterate_payload()`

New in version 2.4. Iterate tree payload:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> for element in tree.iterate_payload():
...     element
...
0
1
2
3
4
5
6
7
```

Return payload generator.

Note: Inherited from `sequencetools.Tree`

`CyclicTree.remove(node)`

New in version 2.4. Remove *node* from tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree.remove(tree[1])

abjad> tree
Tree([[0, 1], [4, 5], [6, 7]])
```

Return none.

Note: Inherited from `sequencetools.Tree`

CyclicTree.remove_to_root()

New in version 2.4. Remove node and all nodes left of node to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]

abjad> tree = sequencetools.Tree(sequence)
abjad> tree[0][0].remove_to_root()
abjad> tree
Tree([[1], [2, 3], [4, 5], [6, 7]])

abjad> tree = sequencetools.Tree(sequence)
abjad> tree[0][1].remove_to_root()
abjad> tree
Tree([[2, 3], [4, 5], [6, 7]])

abjad> tree = sequencetools.Tree(sequence)
abjad> tree[1].remove_to_root()
abjad> tree
Tree([[4, 5], [6, 7]])
```

Modify in-place to root.

Return none.

Note: Inherited from `sequencetools.Tree`

CyclicTree.to_nested_lists()

New in version 2.5. Change tree to nested lists:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

abjad> tree.to_nested_lists()
[[0, 1], [2, 3], [4, 5], [6, 7]]
```

Return list of lists.

Note: Inherited from `sequencetools.Tree`

Special Methods

CyclicTree.__contains__ (*expr*)

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`CyclicTree.__eq__(other)`

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicTree.__getitem__(expr)`

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`CyclicTree.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`CyclicTree.__iter__()`

`CyclicTree.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicTree.__len__()`

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`CyclicTree.__ne__(other)`

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__repr__()`

Note: Inherited from `sequencetools.Tree`

`CyclicTree.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`CyclicTree.__str__()`

Note: Inherited from `sequencetools.Tree`

`sequencetools.CyclicTuple`

`abjad.tools.sequencetools.CyclicTuple`

class `abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple`

New in version 2.0. Abjad model of cyclic tuple:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> cyclic_tuple = sequencetools.CyclicTuple('abcd')
```

```
abjad> cyclic_tuple
CyclicTuple([a, b, c, d])
```

```
abjad> for x in range(8):
...     print x, cyclic_tuple[x]
```

```
...
0 a
1 b
2 c
3 d
4 a
5 b
6 c
7 d
```

Cyclic tuples overload the item-getting method of built-in tuples.

Cyclic tuples return a value for any integer index.

Cyclic tuples otherwise behave exactly like built-in tuples.

Methods

`CyclicTuple.count(value)` \rightarrow integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.index(value[, start[, stop]])` \rightarrow integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

Special Methods

`CyclicTuple.__add__()`
`x.__add__(y)` \iff `x+y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__contains__()`
`x.__contains__(y)` \iff `y in x`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__delattr__()`
`x.__delattr__('name')` \iff `del x.name`

Note: Inherited from `__builtin__.object`

`CyclicTuple.__eq__()`
`x.__eq__(y)` \iff `x==y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__ge__()`
`x.__ge__(y)` \iff `x>=y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__getitem__(expr)`

`CyclicTuple.__getslice__(start_index, stop_index)`

`CyclicTuple.__gt__()`

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__le__()`

`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__len__() <==> len(x)`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__lt__()`

`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__mul__()`

`x.__mul__(n) <==> x*n`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__ne__()`

`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__repr__()`

`CyclicTuple.__rmul__()`

`x.__rmul__(n) <==> n*x`

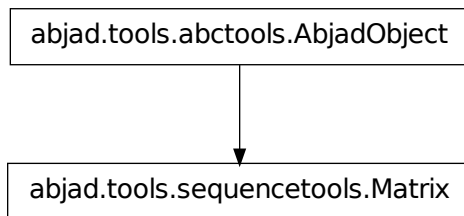
Note: Inherited from `__builtin__.tuple`

`CyclicTuple.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`CyclicTuple.__str__()`

sequencetools.Matrix



class `abjad.tools.sequencetools.Matrix.Matrix(*args, **kwargs)`

New in version 2.0. Abjad model of matrix.

Initialize from rows:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

```
abjad> matrix
Matrix(3x4)
```

```
abjad> matrix[:]
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))
```

```
abjad> matrix[2]
(20, 21, 22, 23)
```

```
abjad> matrix[2][0]
20
```

Initialize from columns:

```
abjad> matrix = sequencetools.Matrix(columns = [[0, 10, 20], [1, 11, 21], [2, 12, 22], [3, 13, 23]])
```

```
abjad> matrix
Matrix(3x4)
```

```

abjad> matrix[:]
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))

abjad> matrix[2]
(20, 21, 22, 23)

abjad> matrix[2][0]
20

```

Matrix implements only item retrieval in this revision.

Concatenation and division remain to be implemented.

Standard transforms of linear algebra remain to be implemented.

Read-only Properties

`Matrix.columns`

Read-only columns:

```

abjad> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

abjad> matrix.columns
((0, 10, 20), (1, 11, 21), (2, 12, 22), (3, 13, 23))

```

Return tuple.

`Matrix.rows`

Read-only rows:

```

abjad> matrix = sequencetools.Matrix([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

abjad> matrix.rows
((0, 1, 2, 3), (10, 11, 12, 13), (20, 21, 22, 23))

```

Return tuple.

Special Methods

`Matrix.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Matrix.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Matrix.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Matrix.__getitem__ (expr)`

`Matrix.__gt__ (arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Matrix.__hash__ () <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Matrix.__le__ (arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Matrix.__lt__ (arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Matrix.__ne__ (arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Matrix.__repr__ ()`

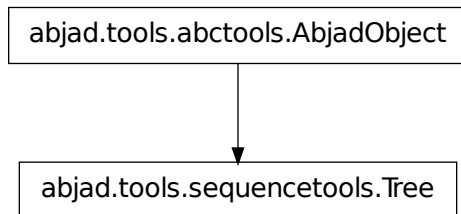
`Matrix.__setattr__ ()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Matrix.__str__ () <==> str(x)`

Note: Inherited from `__builtin__.object`

sequencetools.Tree

class `abjad.tools.sequencetools.Tree.Tree`. **Tree** (*expr*)

New in version 2.4. Abjad data structure to work with a sequence whose elements have been grouped into arbitrarily many levels of containment.

Example: a list of pitches that have been grouped into cells that have, in turn, been grouped into groups of cells that have, in turn, been grouped into groups of groups of cells.

```
abjad> from abjad.tools import sequencetools
```

Here is a tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

abjad> tree.parent is None
True

abjad> tree.children
(Tree([0, 1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7]))

abjad> tree.depth
3
```

Here's an internal node:

```
abjad> tree[2]
Tree([4, 5])

abjad> tree[2].parent
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])

abjad> tree[2].children
(Tree(4), Tree(5))

abjad> tree[2].depth
2
```

```
abjad> tree[2].level
1
```

Here's a leaf node:

```
abjad> tree[2][0]
Tree(4)
```

```
abjad> tree[2][0].parent
Tree([4, 5])
```

```
abjad> tree[2][0].children
()
```

```
abjad> tree[2][0].depth
1
```

```
abjad> tree[2][0].level
2
```

```
abjad> tree[2][0].position
(2, 0)
```

```
abjad> tree[2][0].payload
4
```

Only leaf nodes carry payload. Internal nodes carry no payload.

Negative levels are available to work with trees bottom-up instead of top-down.

Trees do not yet implement append or extend methods.

Read-only Properties

Tree.children

New in version 2.4. Children of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].children
(Tree(2), Tree(3))
```

Return tuple of zero or more nodes.

Tree.depth

New in version 2.4. Depth of subtree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].depth
2
```

Return nonnegative integer.

Tree.improper_parentage

New in version 2.4. Improper parentage of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].improper_parentage
(Tree([2, 3]), Tree([[0, 1], [2, 3], [4, 5], [6, 7]]))
```

Return tuple of one or more nodes.

Tree.**index_in_parent**

New in version 2.4. Index of node in parent:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].index_in_parent
1
```

Return nonnegative integer.

Tree.**level**

New in version 2.4. Level of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].level
1
```

Return nonnegative integer.

Tree.**negative_level**

New in version 2.4. Negative level of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].negative_level
-2
```

Return negative integer.

Tree.**position**

New in version 2.4. Position of node relative to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].position
(1,)
```

Return tuple of zero or more nonnegative integers.

Tree.**proper_parentage**

New in version 2.4. Proper parentage of node:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return tuple of zero or more nodes.

Tree.**root**

New in version 2.4. Root of tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].proper_parentage
(Tree([[0, 1], [2, 3], [4, 5], [6, 7]]),)
```

Return node.

Tree.**width**

New in version 2.4. Number of leaves in subtree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree[1].width
2
```

Return nonnegative integer.

Methods

Tree.**get_next_n_complete_nodes_at_level** (*n*, *level*)

New in version 2.5. Get next *n* complete nodes at *level* from node.

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
abjad> tree[0][0].get_next_n_complete_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5]), Tree([6, 7])]
```

Trim first node if necessary.

Return list of nodes.

Tree.**get_next_n_nodes_at_level** (*n*, *level*)

New in version 2.4. Get next *n* nodes at *level* from node.

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

With nonnegative *level*:

Get next 4 nodes at level 2:

```
abjad> tree[0][0].get_next_n_nodes_at_level(4, 2)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level 1:

```
abjad> tree[0][0].get_next_n_nodes_at_level(3, 1)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Get next node at level 0:

```
abjad> tree[0][0].get_next_n_nodes_at_level(1, 0)
[Tree([[1], [2, 3], [4, 5], [6, 7]])]
```

With negative *level*:

Get next 4 nodes at level -1:

```
abjad> tree[0][0].get_next_n_nodes_at_level(4, -1)
[Tree(1), Tree(2), Tree(3), Tree(4)]
```

Get next 3 nodes at level -2:

```
abjad> tree[0][0].get_next_n_nodes_at_level(3, -2)
[Tree([1]), Tree([2, 3]), Tree([4, 5])]
```

Trim first node if necessary.

Return list of nodes.

Tree.get_node_at_position (*position*)

New in version 2.4. Get node at *position*:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree.get_node_at_position((2, 1))
Tree(5)
```

Return node.

Tree.get_position_of_descendant (*descendant*)

New in version 2.4. Get position of *descendent* relative to node rather than relative to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[3].get_position_of_descendant(tree[3][0])
(0,)
```

Return tuple of zero or more nonnegative integers.

Tree.is_at_level (*level*)

New in version 2.4. True when node is at *level* in tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree[1][1].is_at_level(-1)
True
```


False otherwise:

```
abjad> tree[1][1].is_at_level(0)
False
```

Return boolean.

Predicate works for positive, negative and zero-valued *level*.

Tree.**iterate_at_level**(*level*)

New in version 2.4. Iterate depth at *level*:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> for x in tree.iterate_at_level(0): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

```
abjad> for x in tree.iterate_at_level(1): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
abjad> for x in tree.iterate_at_level(2): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
abjad> for x in tree.iterate_at_level(-1): x
...
Tree(0)
Tree(1)
Tree(2)
Tree(3)
Tree(4)
Tree(5)
Tree(6)
Tree(7)
```

```
abjad> for x in tree.iterate_at_level(-2): x
...
Tree([0, 1])
Tree([2, 3])
Tree([4, 5])
Tree([6, 7])
```

```
abjad> for x in tree.iterate_at_level(-3): x
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

Return node generator.

`Tree.iterate_depth_first()`

New in version 2.4. Iterate tree depth-first:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> for node in tree.iterate_depth_first(): node
...
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
Tree([0, 1])
Tree(0)
Tree(1)
Tree([2, 3])
Tree(2)
Tree(3)
Tree([4, 5])
Tree(4)
Tree(5)
Tree([6, 7])
Tree(6)
Tree(7)
```

Return node generator.

`Tree.iterate_payload()`

New in version 2.4. Iterate tree payload:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> for element in tree.iterate_payload():
...     element
...
0
1
2
3
4
5
6
7
```

Return payload generator.

`Tree.remove(node)`

New in version 2.4. Remove *node* from tree:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)

abjad> tree.remove(tree[1])

abjad> tree
Tree([[0, 1], [4, 5], [6, 7]])
```

Return none.

`Tree.remove_to_root()`

New in version 2.4. Remove node and all nodes left of node to root:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
```

```
abjad> tree = sequencetools.Tree(sequence)
abjad> tree[0][0].remove_to_root()
abjad> tree
Tree([[1], [2, 3], [4, 5], [6, 7]])
```

```
abjad> tree = sequencetools.Tree(sequence)
abjad> tree[0][1].remove_to_root()
abjad> tree
Tree([[2, 3], [4, 5], [6, 7]])
```

```
abjad> tree = sequencetools.Tree(sequence)
abjad> tree[1].remove_to_root()
abjad> tree
Tree([[4, 5], [6, 7]])
```

Modify in-place to root.

Return none.

Tree.**to_nested_lists**()

New in version 2.5. Change tree to nested lists:

```
abjad> sequence = [[0, 1], [2, 3], [4, 5], [6, 7]]
abjad> tree = sequencetools.Tree(sequence)
```

```
abjad> tree
Tree([[0, 1], [2, 3], [4, 5], [6, 7]])
```

```
abjad> tree.to_nested_lists()
[[0, 1], [2, 3], [4, 5], [6, 7]]
```

Return list of lists.

Special Methods

Tree.**__contains__**(*expr*)

Tree.**__delattr__**()

x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

Tree.**__eq__**(*other*)

Tree.**__ge__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Tree.**__getitem__**(*expr*)

Tree.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Tree.__hash__()` $\iff hash(x)$

Note: Inherited from `__builtin__.object`

`Tree.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tree.__len__()`

`Tree.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Tree.__ne__(other)`

`Tree.__repr__()`

`Tree.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`

Note: Inherited from `__builtin__.object`

`Tree.__str__()`

sequencetools functions

sequencetools.all_are_assignable_integers

`abjad.tools.sequencetools.all_are_assignable_integers.all_are_assignable_integers(expr)`
New in version 2.0. True when *expr* is a sequence and all elements in *expr* are notehead-assignable integers:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_assignable_integers([1, 2, 3, 4, 6, 7, 8, 12, 14, 15, 16])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_assignable_integers([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_assignable_integers('foo')
False
```

Return boolean.

sequencetools.all_are_equal

`abjad.tools.sequencetools.all_are_equal.all_are_equal(expr)`
New in version 2.0. True when *expr* is a sequence and all elements in *expr* are equal:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.all_are_equal([99, 99, 99, 99, 99, 99])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_equal([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_equal(17)
False
```

Return boolean.

sequencetools.all_are_integer_equivalent_numbers

`abjad.tools.sequencetools.all_are_integer_equivalent_numbers.all_are_integer_equivalent_numbers(expr)`
New in version 2.0. True when *expr* is a sequence and all elements in *expr* are integer-equivalent numbers:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.all_are_integer_equivalent_numbers([1, 2, 3.0, Fraction(4, 1)])
True
```

Otherwise false:

```
abjad> sequencetools.all_are_integer_equivalent_numbers([1, 2, 3.5, 4])
False
```

Return boolean.

sequencetools.all_are_nonnegative_integer_equivalent_numbers

`abjad.tools.sequencetools.all_are_nonnegative_integer_equivalent_numbers.all_are_nonnegative_integer_equivalent_numbers(expr)`
New in version 2.0. True *expr* is a sequence and when all elements in *expr* are nonnegative integer-equivalent numbers. Otherwise false:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.all_are_nonnegative_integer_equivalent_numbers([0, 0.0, Fraction(0), 2, 2.0])
True
```

Return boolean.

sequencetools.all_are_nonnegative_integer_powers_of_two

`abjad.tools.sequencetools.all_are_nonnegative_integer_powers_of_two.all_are_nonnegative_int`
New in version 2.0. True when *expr* is a sequence and all elements in *expr* are nonnegative integer powers of two:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_nonnegative_integer_powers_of_two([0, 1, 1, 1, 2, 4, 32, 32])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_nonnegative_integer_powers_of_two([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_nonnegative_integer_powers_of_two(17)
False
```

Return boolean.

sequencetools.all_are_nonnegative_integers

`abjad.tools.sequencetools.all_are_nonnegative_integers.all_are_nonnegative_integers(expr)`
New in version 2.0. True when *expr* is a sequence and all elements in *expr* are nonnegative integers:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_nonnegative_integers([0, 1, 2, 99])
True
```

Otherwise false:

```
abjad> sequencetools.all_are_nonnegative_integers([0, 1, 2, -99])
False
```

Return boolean.

sequencetools.all_are_numbers

`abjad.tools.sequencetools.all_are_numbers.all_are_numbers(expr)`
New in version 1.1. True when *expr* is a sequence and all elements in *expr* are numbers:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_numbers([1, 2, 3.0, Fraction(13, 8)])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_numbers([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_numbers(17)
False
```

Return boolean. Changed in version 2.0: renamed `sequencetools.is_numeric()` to `sequencetools.all_are_numbers()`.

sequencetools.all_are_pairs

`abjad.tools.sequencetools.all_are_pairs.all_are_pairs(expr)`

True when *expr* is a sequence whose members are all sequences of length 2:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_pairs([(1, 2), (3, 4), (5, 6), (7, 8)])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_pairs([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_pairs('foo')
False
```

Return boolean.

sequencetools.all_are_pairs_of_types

`abjad.tools.sequencetools.all_are_pairs_of_types.all_are_pairs_of_types(expr,
 first_type,
 sec-
 ond_type)`

True when *expr* is a sequence whose members are all sequences of length 2, and where the first member of each pair is an instance of *first_type* and where the second member of each pair is an instance of *second_type*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_pairs_of_types([(1., 'a'), (2.1, 'b'), (3.45, 'c')], float, str)
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_pairs_of_types([], float, str)
True
```

False otherwise:

```
abjad> sequencetools.all_are_pairs_of_types('foo', float, str)
False
```

Return boolean.

sequencetools.all_are_positive_integer_equivalent_numbers

`abjad.tools.sequencetools.all_are_positive_integer_equivalent_numbers.all_are_positive_integer_equivalent_numbers(expr)`
 New in version 2.0. True when *expr* is a sequence and all elements in *expr* are positive integer-equivalent numbers. Otherwise false:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_positive_integer_equivalent_numbers([Fraction(4, 2), 2.0, 2])
True
```

Return boolean.

sequencetools.all_are_positive_integers

`abjad.tools.sequencetools.all_are_positive_integers.all_are_positive_integers(expr)`
 New in version 2.0. True when *expr* is a sequence and all elements in *expr* are positive integers:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_positive_integers([1, 2, 3, 99])
True
```

Otherwise false:

```
abjad> sequencetools.all_are_positive_integers(17)
False
```

Return boolean.

sequencetools.all_are_unequal

`abjad.tools.sequencetools.all_are_unequal.all_are_unequal(expr)`
 New in version 1.1. True when *expr* is a sequence all elements in *expr* are unequal:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.all_are_unequal([1, 2, 3, 4, 9])
True
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.all_are_unequal([])
True
```

False otherwise:

```
abjad> sequencetools.all_are_unequal(17)
False
```

Return boolean. Changed in version 2.0: renamed `sequencetools.is_unique()` to `sequencetools.all_are_unequal()`.

sequencetools.count_length_two_runs_in_sequence

`abjad.tools.sequencetools.count_length_two_runs_in_sequence`.**count_length_two_runs_in_sequence**

New in version 1.1. Count length-2 runs in *sequence*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.count_length_two_runs_in_sequence([0, 0, 1, 1, 1, 2, 3, 4, 5])
3
```

Return nonnegative integer. Changed in version 2.0: renamed `sequencetools.count_repetitions()` to `sequencetools.count_length_two_runs_in_sequence()`.

sequencetools.divide_sequence_elements_by_greatest_common_divisor

`abjad.tools.sequencetools.divide_sequence_elements_by_greatest_common_divisor`.**divide_sequence_elements_by_greatest_common_divisor**

New in version 2.0. Divide *sequence* elements by greatest common divisor:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.divide_sequence_elements_by_greatest_common_divisor([2, 2, -8, -16])
[1, 1, -4, -8]
```

Allow negative *sequence* elements.

Raise type error on noninteger *sequence* elements.

Raise not implemented error when 0 in *sequence*.

Return new *sequence* object.

sequencetools.flatten_sequence

`abjad.tools.sequencetools.flatten_sequence`.**flatten_sequence** (*sequence*,
klases=None,
depth=-1)

New in version 1.1. Flatten *sequence*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]])
[1, 2, 3, 4, 5, 6, 7, 8]
```

Flatten *sequence* to depth 1:

```
abjad> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]], depth = 1)
[1, 2, 3, [4], 5, 6, 7, [8]]
```

Flatten *sequence* to depth 2:

```
abjad> sequencetools.flatten_sequence([1, [2, 3, [4]], 5, [6, 7, [8]]], depth = 2)
[1, 2, 3, 4, 5, 6, 7, 8]
```

Leave *sequence* unchanged.

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.flatten()` to `sequencetools.flatten_sequence()`.

sequencetools.flatten_sequence_at_indices

`abjad.tools.sequencetools.flatten_sequence_at_indices.flatten_sequence_at_indices(sequence, indices, classes=None, depth=-1)`

New in version 2.0. Flatten *sequence* at *indices*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.flatten_sequence_at_indices([0, 1, [2, 3, 4], [5, 6, 7]], [3])
[0, 1, [2, 3, 4], 5, 6, 7]
```

Flatten *sequence* at negative *indices*:

```
abjad> sequencetools.flatten_sequence_at_indices([0, 1, [2, 3, 4], [5, 6, 7]], [-1])
[0, 1, [2, 3, 4], 5, 6, 7]
```

Leave *sequence* unchanged.

Return newly constructed *sequence* object.

sequencetools.get_indices_of_sequence_elements_equal_to_true

`abjad.tools.sequencetools.get_indices_of_sequence_elements_equal_to_true.get_indices_of_sequence_elements_equal_to_true(sequence)`

New in version 1.1. Get indices of *sequence* elements equal to true:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.get_indices_of_sequence_elements_equal_to_true([0, 0, 0, 1, 1, 1, 0, 0, 0,
(3, 4, 5, 9, 10, 11, 12)
```

Return newly constructed tuple of zero or more nonnegative integers. Changed in version 2.0: renamed `listtools.true_indices()` to `sequencetools.get_indices_of_sequence_elements_equal_to_true()`.

sequencetools.get_sequence_degree_of_rotational_symmetry

`abjad.tools.sequencetools.get_sequence_degree_of_rotational_symmetry.get_sequence_degree_of_rotational_symmetry(sequence)`

New in version 2.0. Change *sequence* to degree of rotational symmetry:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 3, 4, 5, 6])
1
```

```
abjad> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 3, 1, 2, 3])
2
```

```
abjad> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 2, 1, 2, 1, 2])
3
```

```
abjad> sequencetools.get_sequence_degree_of_rotational_symmetry([1, 1, 1, 1, 1, 1])
6
```

Return positive integer.

`sequencetools.get_sequence_element_at_cyclic_index`

`abjad.tools.sequencetools.get_sequence_element_at_cyclic_index.get_sequence_element_at_cyclic_index`

New in version 2.0. Get *sequence* element at nonnegative cyclic *index*:

```
abjad> from abjad.tools import sequencetools

abjad> for index in range(10):
...     print '%s\t%s' % (index, sequencetools.get_sequence_element_at_cyclic_index('string', index))
...
0  s
1  t
2  r
3  i
4  n
5  g
6  s
7  t
8  r
9  i
```

Get *sequence* element at negative cyclic *index*:

```
abjad> for index in range(1, 11):
...     print '%s\t%s' % (-index, sequencetools.get_sequence_element_at_cyclic_index('string', -index))
...
-1  g
-2  n
-3  i
-4  r
-5  t
-6  s
-7  g
-8  n
-9  i
-10 r
```

Return reference to *sequence* element.

`sequencetools.get_sequence_elements_at_indices`

`abjad.tools.sequencetools.get_sequence_elements_at_indices.get_sequence_elements_at_indices`

New in version 2.0. Get *sequence* elements at *indices*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.get_sequence_elements_at_indices('string of text', (2, 3, 10, 12))
('r', 'i', 't', 'x')
```

Return newly constructed tuple of references to *sequence* elements.

sequencetools.get_sequence_elements_frequency_distribution

abjad.tools.sequencetools.get_sequence_elements_frequency_distribution.get_sequence_element

New in version 2.0. Get *sequence* elements frequency distribution:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.get_sequence_elements_frequency_distribution([1, 3, 3, 3, 2, 1, 1, 2, 3, 3,
[(1, 4), (2, 3), (3, 5)]
```

Return list of element / count pairs.

sequencetools.get_sequence_period_of_rotation

abjad.tools.sequencetools.get_sequence_period_of_rotation.get_sequence_period_of_rotation(s

New in version 2.0. Change *sequence* to period of rotation:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 1)
3
```

```
abjad> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 2)
3
```

```
abjad> sequencetools.get_sequence_period_of_rotation([1, 2, 3, 1, 2, 3], 3)
1
```

Return positive integer.

sequencetools.increase_sequence_elements_at_indices_by_addenda

abjad.tools.sequencetools.increase_sequence_elements_at_indices_by_addenda.increase_sequenc

New in version 1.1. Increase *sequence* by *addenda* at *indices*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [1, 1, 2, 3, 5, 5, 1, 2, 5, 5, 6]
```

```
abjad> sequencetools.increase_sequence_elements_at_indices_by_addenda(sequence, [0.5, 0.5], [0,
[1.5, 1.5, 2, 3, 5.5, 5.5, 1, 2, 5.5, 5.5, 6]
```

Return list. Changed in version 2.0: renamed `sequencetools.increase_at_indices()` to `sequencetools.increase_sequence_elements_at_indices_by_addenda()`.

sequencetools.increase_sequence_elements_cyclically_by_addenda

abjad.tools.sequencetools.increase_sequence_elements_cyclically_by_addenda.increase_sequenc

New in version 1.1.. Increase *sequence* cyclically by *addenda*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.increase_sequence_elements_cyclically_by_addenda(range(10), [10, -10], shie
[10, -9, 12, -7, 14, -5, 16, -3, 18, -1]
```

Increase *sequence* cyclically by addenda and map nonpositive values to 1:

```
abjad> sequencetools.increase_sequence_elements_cyclically_by_addenda(range(10), [10, -10], shie
[10, 1, 12, 1, 14, 1, 16, 1, 18, 1]
```

Return list. Changed in version 2.0: renamed `sequencetools.increase_cyclic()` to `sequencetools.increase_sequence_elements_cyclically_by_addenda()`.

sequencetools.interlace_sequences

`abjad.tools.sequencetools.interlace_sequences.interlace_sequences(*sequences)`

New in version 1.1. Interlace *sequences*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> k = range(100, 103)
abjad> l = range(200, 201)
abjad> m = range(300, 303)
abjad> n = range(400, 408)
abjad> sequencetools.interlace_sequences(k, l, m, n)
[100, 200, 300, 400, 101, 301, 401, 102, 302, 402, 403, 404, 405, 406, 407]
```

Return list. Changed in version 2.0: renamed `sequencetools.interlace()` to `sequencetools.interlace_sequences()`.

sequencetools.is_monotonically_decreasing_sequence

`abjad.tools.sequencetools.is_monotonically_decreasing_sequence.is_monotonically_decreasing`

New in version 2.0. True when *expr* is a sequence and the elements in *expr* decrease monotonically:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

```
abjad> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

```
abjad> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not decrease monotonically:

```
abjad> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
False
```

```
abjad> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
False
```

True when *expr* is a sequence and *expr* is empty:

```
abjad> expr = []
abjad> sequencetools.is_monotonically_decreasing_sequence(expr)
True
```

False when *expr* is not a sequence:

```
abjad> sequencetools.is_monotonically_decreasing_sequence(17)
False
```

Return boolean.

sequencetools.is_monotonically_increasing_sequence

`abjad.tools.sequencetools.is_monotonically_increasing_sequence.is_monotonically_increasing`

New in version 2.0. True when *expr* is a sequence and the elements in *expr* increase monotonically:

```
abjad> from abjad.tools import sequencetools

abjad> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
True

abjad> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
True

abjad> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not increase monotonically:

```
abjad> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
False

abjad> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
False
```

True when *expr* is a sequence and *expr* is empty:

```
abjad> expr = []
abjad> sequencetools.is_monotonically_increasing_sequence(expr)
True
```

False when *expr* is not a sequence:

```
abjad> sequencetools.is_monotonically_increasing_sequence(17)
False
```

Return boolean.

sequencetools.is_permutation

`abjad.tools.sequencetools.is_permutation.is_permutation(expr, length=None)`

New in version 2.0. True when *expr* is a permutation:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.is_permutation([4, 5, 0, 3, 2, 1])
True
```

Otherwise false:

```
abjad> sequencetools.is_permutation([1, 1, 5, 3, 2, 1])
False
```

True when *expr* is a permutation of first *length* nonnegative integers:

```
abjad> sequencetools.is_permutation([4, 5, 0, 3, 2, 1], length = 6)
True
```

Otherwise false:

```
abjad> sequencetools.is_permutation([4, 0, 3, 2, 1], length = 6)
False
```

Return boolean.

sequencetools.is_repetition_free_sequence

`abjad.tools.sequencetools.is_repetition_free_sequence.is_repetition_free_sequence(expr)`

New in version 2.0. True when *expr* is a sequence and *expr* is repetition free:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.is_repetition_free_sequence([0, 1, 2, 6, 7, 8])
True
```

False when *expr* is a sequence and *expr* is not repetition free:

```
abjad> sequencetools.is_repetition_free_sequence([0, 1, 2, 2, 7, 8])
False
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.is_repetition_free_sequence([])
True
```

False *expr* is not a sequence:

```
abjad> sequencetools.is_repetition_free_sequence(17)
False
```

Return boolean.

sequencetools.is_restricted_growth_function

`abjad.tools.sequencetools.is_restricted_growth_function.is_restricted_growth_function(expr)`

New in version 2.0. True when *expr* is a sequence and *expr* meets the criteria for a restricted growth function:

```

abjad> from abjad.tools import sequencetools

abjad> sequencetools.is_restricted_growth_function([1, 1, 1, 1])
True

abjad> sequencetools.is_restricted_growth_function([1, 1, 1, 2])
True

abjad> sequencetools.is_restricted_growth_function([1, 1, 2, 1])
True

abjad> sequencetools.is_restricted_growth_function([1, 1, 2, 2])
True

```

Otherwise false:

```

abjad> sequencetools.is_restricted_growth_function([1, 1, 1, 3])
False

abjad> sequencetools.is_restricted_growth_function(17)
False

```

A restricted growth function is a sequence l such that $l[0] == 1$ and such that $l[i] \leq \max(l[:i]) + 1$ for $1 \leq i \leq \text{len}(l)$.

Return boolean.

sequencetools.is_strictly_decreasing_sequence

`abjad.tools.sequencetools.is_strictly_decreasing_sequence`. **`is_strictly_decreasing_sequence`** (New in version 2.0. True when *expr* is a sequence and the elements in *expr* decrease strictly:

```

abjad> from abjad.tools import sequencetools

abjad> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
abjad> sequencetools.is_strictly_decreasing_sequence(expr)
True

```

False when *expr* is a sequence and the elements in *expr* do not decrease strictly:

```

abjad> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
abjad> sequencetools.is_strictly_decreasing_sequence(expr)
False

abjad> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_strictly_decreasing_sequence(expr)
False

abjad> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_strictly_decreasing_sequence(expr)
False

abjad> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
abjad> sequencetools.is_strictly_decreasing_sequence(expr)
False

```

True when *expr* is an empty sequence:


```
abjad> sequencetools.is_strictly_decreasing_sequence([])
True
```

False *expr* is not a sequence:

```
abjad> sequencetools.is_strictly_decreasing_sequence(17)
False
```

Return boolean.

sequencetools.is_strictly_increasing_sequence

`abjad.tools.sequencetools.is_strictly_increasing_sequence.is_strictly_increasing_sequence` (*sequence*, *step*, *start*, *length*)

New in version 2.0. True when *expr* is a sequence and the elements in *expr* increase strictly:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> expr = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
abjad> sequencetools.is_strictly_increasing_sequence(expr)
True
```

False when *expr* is a sequence and the elements in *expr* do not increase strictly:

```
abjad> expr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
abjad> sequencetools.is_strictly_increasing_sequence(expr)
False
```

```
abjad> expr = [3, 3, 3, 3, 3, 3, 3, 2, 1, 0]
abjad> sequencetools.is_strictly_increasing_sequence(expr)
False
```

```
abjad> expr = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_strictly_increasing_sequence(expr)
False
```

```
abjad> expr = [0, 1, 2, 3, 3, 3, 3, 3, 3, 3]
abjad> sequencetools.is_strictly_increasing_sequence(expr)
False
```

True when *expr* is an empty sequence:

```
abjad> sequencetools.is_strictly_increasing_sequence([])
True
```

False when *expr* is not a sequence:

```
abjad> sequencetools.is_strictly_increasing_sequence(17)
False
```

Return boolean.

sequencetools.iterate_sequence_cyclically

`abjad.tools.sequencetools.iterate_sequence_cyclically.iterate_sequence_cyclically` (*sequence*, *step*, *start*, *length*)

sequence, *step*, *start*, *length*

sequence: sequence to iterate over

step: step to iterate by (default 1)

start: start index (default 0)

length: length of sequence to iterate over (default 'inf')

New in version 1.1. Iterate *sequence* cyclically according to *step*, *start* and *length*:

```

abjad> from abjad.tools import sequencetools

abjad> sequence = [1, 2, 3, 4, 5, 6, 7]

abjad> list(sequencetools.iterate_sequence_cyclically(sequence, length = 20))
[1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6]

abjad> list(sequencetools.iterate_sequence_cyclically(sequence, 2, length = 20))
[1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4]

abjad> list(sequencetools.iterate_sequence_cyclically(sequence, 2, 3, length = 20))
[4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7, 2, 4, 6, 1, 3, 5, 7]

abjad> list(sequencetools.iterate_sequence_cyclically(sequence, -2, 5, length = 20))
[6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3]

```

Changed in version 2.0: allows generator input.

```

abjad> list(sequencetools.iterate_sequence_cyclically(xrange(1, 8), -2, 5, length = 20))
[6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3, 1, 6, 4, 2, 7, 5, 3]

```

Set *step* to jump size and direction across sequence.

Set *start* to the index of *sequence* where the function begins iterating.

Set *length* to number of elements to return. Set to 'inf' to return infinitely.

Return generator. Changed in version 2.0: renamed `sequencetools.phasor()` to `sequencetools.iterate_sequence_cyclically()`.

`sequencetools.iterate_sequence_cyclically_from_start_to_stop`

`abjad.tools.sequencetools.iterate_sequence_cyclically_from_start_to_stop.iterate_sequence_cyclically_from_start_to_stop`

New in version 1.1. Iterate *sequence* cyclically from *start* to *stop*:

```

abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.iterate_sequence_cyclically_from_start_to_stop(range(20), 18, 10))
[18, 19, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

```

Return generator of references to *sequence* elements. Changed in version 2.0: renamed `sequencetools.get_cyclic()` to `sequencetools.iterate_sequence_cyclically_from_start_to_stop`.

`sequencetools.iterate_sequence_forward_and_backward_nonoverlapping`

`abjad.tools.sequencetools.iterate_sequence_forward_and_backward_nonoverlapping.iterate_sequence_forward_and_backward_nonoverlapping`

New in version 2.0. Iterate *sequence* first forward and then backward, with first and last elements repeated:

```

abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.iterate_sequence_forward_and_backward_nonoverlapping([1, 2, 3, 4, 5]))
[1, 2, 3, 4, 5, 5, 4, 3, 2, 1]

```

Return generator.

`sequencetools.iterate_sequence_forward_and_backward_overlapping`

`abjad.tools.sequencetools.iterate_sequence_forward_and_backward_overlapping.iterate_sequence_forward_and_backward_overlapping`. New in version 2.0. Iterate *sequence* first forward and then backward, with first and last elements appearing only once:

```
abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.iterate_sequence_forward_and_backward_overlapping([1, 2, 3, 4, 5]))
[1, 2, 3, 4, 5, 4, 3, 2]
```

Return generator.

`sequencetools.iterate_sequence_nwise_cyclic`

`abjad.tools.sequencetools.iterate_sequence_nwise_cyclic.iterate_sequence_nwise_cyclic` (*sequence*, *n*)

New in version 2.0. Iterate elements in *sequence* cyclically *n* at a time:

```
abjad> from abjad.tools import sequencetools

abjad> g = sequencetools.iterate_sequence_nwise_cyclic(range(6), 3)
abjad> for n in range(10):
...     print g.next()
(0, 1, 2)
(1, 2, 3)
(2, 3, 4)
(3, 4, 5)
(4, 5, 0)
(5, 0, 1)
(0, 1, 2)
(1, 2, 3)
(2, 3, 4)
(3, 4, 5)
```

Return generator.

`sequencetools.iterate_sequence_nwise_strict`

`abjad.tools.sequencetools.iterate_sequence_nwise_strict.iterate_sequence_nwise_strict` (*sequence*, *n*)

New in version 2.0. Iterate elements in *sequence* *n* at a time:

```
abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.iterate_sequence_nwise_strict(range(10), 4))
[(0, 1, 2, 3), (1, 2, 3, 4), (2, 3, 4, 5), (3, 4, 5, 6), (4, 5, 6, 7), (5, 6, 7, 8), (6, 7, 8, 9)]
```

Return generator.

`sequencetools.iterate_sequence_nwise_wrapped`

`abjad.tools.sequencetools.iterate_sequence_nwise_wrapped.iterate_sequence_nwise_wrapped` (*sequence*, *n*)

New in version 2.0. Iterate elements in *sequence* *n* at a time wrapped to beginning:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.iterate_sequence_nwise_wrapped(range(6), 3))
[(0, 1, 2), (1, 2, 3), (2, 3, 4), (3, 4, 5), (4, 5, 0), (5, 0, 1)]
```

Return generator.

sequencetools.iterate_sequence_pairwise_cyclic

`abjad.tools.sequencetools.iterate_sequence_pairwise_cyclic.iterate_sequence_pairwise_cyclic`

New in version 1.1. Iterate *sequence* pairwise cyclic:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> generator = sequencetools.iterate_sequence_pairwise_cyclic(range(6))
```

```
abjad> generator.next()
(0, 1)
abjad> generator.next()
(1, 2)
abjad> generator.next()
(2, 3)
abjad> generator.next()
(3, 4)
abjad> generator.next()
(4, 5)
abjad> generator.next()
(5, 0)
abjad> generator.next()
(0, 1)
abjad> generator.next()
(1, 2)
```

Return pair generator.

sequencetools.iterate_sequence_pairwise_strict

`abjad.tools.sequencetools.iterate_sequence_pairwise_strict.iterate_sequence_pairwise_strict`

New in version 1.1. Iterate *sequence* pairwise strict:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.iterate_sequence_pairwise_strict(range(6)))
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]
```

Return pair generator.

sequencetools.iterate_sequence_pairwise_wrapped

`abjad.tools.sequencetools.iterate_sequence_pairwise_wrapped.iterate_sequence_pairwise_wrapped`

New in version 1.1. Iterate *sequence* pairwise wrapped:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.iterate_sequence_pairwise_wrapped(range(6)))
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 0)]
```

Return pair generator.

sequencetools.join_subsequences

abjad.tools.sequencetools.join_subsequences.**join_subsequences**(*sequence*)
 New in version 2.4. Join subsequences in *sequence*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.join_subsequences([(1, 2, 3), (), (4, 5), (), (6,)])
(1, 2, 3, 4, 5, 6)
```

Return newly constructed object of subsequence type.

sequencetools.join_subsequences_by_sign_of_subsequence_elements

abjad.tools.sequencetools.join_subsequences_by_sign_of_subsequence_elements.**join_subsequences_by_sign_of_subsequence_elements**(*sequence*)
 New in version 1.1. Join subsequences in *sequence* by sign:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [[1, 2], [3, 4], [-5, -6, -7], [-8, -9, -10], [11, 12]]
abjad> sequencetools.join_subsequences_by_sign_of_subsequence_elements(sequence)
[[1, 2, 3, 4], [-5, -6, -7, -8, -9, -10], [11, 12]]

abjad> sequence = [[1, 2], [], [], [3, 4, 5], [6, 7]]
abjad> sequencetools.join_subsequences_by_sign_of_subsequence_elements(sequence)
[[1, 2], [], [3, 4, 5, 6, 7]]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.join_sublists_by_sign()` to `sequencetools.join_subsequences_by_sign_of_subsequence_elements()`.

sequencetools.map_sequence_elements_to_canonic_tuples

abjad.tools.sequencetools.map_sequence_elements_to_canonic_tuples.**map_sequence_elements_to_canonic_tuples**(*sequence*)

New in version 1.1. Partition *sequence* elements into canonic big-endian parts:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.map_sequence_elements_to_canonic_tuples(range(10))
[(0,), (1,), (2,), (3,), (4,), (4, 1), (6,), (7,), (8,), (8, 1)]
```

Partition *sequence* elements into canonic little-endian parts:

```
abjad> sequencetools.map_sequence_elements_to_canonic_tuples(range(10), direction = 'little-endian')
[(0,), (1,), (2,), (3,), (4,), (1, 4), (6,), (7,), (8,), (1, 8)]
```

Raise type error when *sequence* is not a list.

Raise value error on noninteger elements in *sequence*.

Return list of tuples. Changed in version 2.0: renamed `sequencetools.partition_elements_into_canonic_parts` to `sequencetools.map_sequence_elements_to_canonic_tuples()`.

`sequencetools.map_sequence_elements_to_numbered_sublists`

`abjad.tools.sequencetools.map_sequence_elements_to_numbered_sublists.map_sequence_elements`
 New in version 1.1. Map *sequence* elements to numbered sublists:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.map_sequence_elements_to_numbered_sublists([1, 2, -3, -4, 5])
[[1], [2, 3], [-4, -5, -6], [-7, -8, -9, -10], [11, 12, 13, 14, 15]]

abjad> sequencetools.map_sequence_elements_to_numbered_sublists([1, 0, -3, -4, 5])
[[1], [], [-2, -3, -4], [-5, -6, -7, -8], [9, 10, 11, 12, 13]]
```

Note that numbering starts at 1.

Return newly constructed list of lists. Changed in version 2.0: renamed `sequencetools.lengths_to_counts()` to `sequencetools.map_sequence_elements_to_numbered_sublists()`.

`sequencetools.negate_absolute_value_of_sequence_elements_at_indices`

`abjad.tools.sequencetools.negate_absolute_value_of_sequence_elements_at_indices.negate_abs`

New in version 1.1. Negate the absolute value of *sequence* elements at *indices*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

abjad> sequencetools.negate_sequence_elements_at_indices(sequence, [0, 1, 2])
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.negate_elements_at_indices_absolutely()` to `sequencetools.negate_absolute_value_of_sequence_elements_at_indices()`.

`sequencetools.negate_absolute_value_of_sequence_elements_cyclically`

`abjad.tools.sequencetools.negate_absolute_value_of_sequence_elements_cyclically.negate_abs`

New in version 2.0. Negate the absolute value of *sequence* elements at *indices* cyclically according to *period*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

abjad> sequencetools.negate_absolute_value_of_sequence_elements_cyclically(sequence, [0, 1, 2],
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10])
```

Return newly constructed list.

`sequencetools.negate_sequence_elements_at_indices`

`abjad.tools.sequencetools.negate_sequence_elements_at_indices`.**`negate_sequence_elements_at_`**

New in version 1.1. Negate *sequence* elements at *indices*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

abjad> sequencetools.negate_sequence_elements_at_indices(sequence, [0, 1, 2])
[-1, -2, -3, 4, 5, -6, -7, -8, -9, -10]
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.negate_elements_at_indices()` to `sequencetools.negate_sequence_elements_at_`

`sequencetools.negate_sequence_elements_cyclically`

`abjad.tools.sequencetools.negate_sequence_elements_cyclically`.**`negate_sequence_elements_cyc`**

New in version 2.0. Negate *sequence* elements at *indices* cyclically according to *period*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [1, 2, 3, 4, 5, -6, -7, -8, -9, -10]

abjad> sequencetools.negate_sequence_elements_cyclically(sequence, [0, 1, 2], 5)
[-1, -2, -3, 4, 5, 6, 7, 8, -9, -10]
```

Return newly constructed list.

`sequencetools.overwrite_sequence_elements_at_indices`

`abjad.tools.sequencetools.overwrite_sequence_elements_at_indices`.**`overwrite_sequence_element`**

New in version 1.1. Overwrite *sequence* elements at indices according to *pairs*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.overwrite_sequence_elements_at_indices(range(10), [(0, 3), (5, 3)])
[0, 0, 0, 3, 4, 5, 5, 5, 8, 9]
```

Set *pairs* to a list of (anchor_index, length) pairs.

Return new list. Changed in version 2.0: renamed `sequencetools.overwrite_slices_at()` to `sequencetools.overwrite_sequence_elements_at_indices()`.

`sequencetools.partition_sequence_by_ratio_of_lengths`

`abjad.tools.sequencetools.partition_sequence_by_ratio_of_lengths`.**`partition_sequence_by_rat`**

New in version 2.0. Partition *sequence* by ratio of *lengths*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_lengths(tuple(range(10)), [1, 1, 2])
[(0, 1, 2), (3, 4), (5, 6, 7, 8, 9)]
```

Use rounding magic to avoid fractional part lengths.

Return list of *sequence* objects.

sequencetools.partition_sequence_by_ratio_of_weights

abjad.tools.sequencetools.partition_sequence_by_ratio_of_weights.**partition_sequence_by_ratio_of_weights**

New in version 2.0. Partition *sequence* by ratio of *weights*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [1, 1, 1])
[[1, 1, 1], [1, 1, 1, 1], [1, 1, 1]]
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [1, 1, 1, 1])
[[1, 1, 1], [1, 1], [1, 1, 1], [1, 1]]
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [2, 2, 3])
[[1, 1, 1], [1, 1, 1], [1, 1, 1, 1]]
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1] * 10, [3, 2, 2])
[[1, 1, 1, 1], [1, 1, 1], [1, 1, 1]]
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2])
[[1, 1, 1, 1, 1, 1, 2, 2], [2, 2, 2, 2]]
```

```
abjad> sequencetools.partition_sequence_by_ratio_of_weights([1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2])
[[1, 1, 1, 1, 1, 1], [2, 2, 2], [2, 2, 2]]
```

Weights of parts of returned list equal *weights_ratio* proportions with some rounding magic.

Return list of lists.

sequencetools.partition_sequence_by_restricted_growth_function

abjad.tools.sequencetools.partition_sequence_by_restricted_growth_function.**partition_sequence_by_restricted_growth_function**

New in version 2.0. Partition *sequence* by *restricted_growth_function*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> l = range(10)
abjad> rgf = [1, 1, 2, 2, 1, 2, 3, 3, 2, 4]
abjad> sequencetools.partition_sequence_by_restricted_growth_function(l, rgf)
[[0, 1, 4], [2, 3, 5, 8], [6, 7], [9]]
```

Raise value error when *sequence* length does not equal *restricted_growth_function* length.

Return list of lists.

sequencetools.partition_sequence_by_sign_of_elements

`abjad.tools.sequencetools.partition_sequence_by_sign_of_elements.partition_sequence_by_sign_of_elements`

New in version 1.1. Partition *sequence* elements by sign:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [0, 0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence))
[[0, 0], [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [-1]))
[0, 0, [-1, -1], 2, 3, [-5], 1, 2, 5, [-5, -6]]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [0]))
[[0, 0], -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [1]))
[0, 0, -1, -1, [2, 3], -5, [1, 2, 5], -5, -6]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [-1, 0]))
[[0, 0], [-1, -1], 2, 3, [-5], 1, 2, 5, [-5, -6]]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [-1, 1]))
[0, 0, [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [0, 1]))
[[0, 0], -1, -1, [2, 3], -5, [1, 2, 5], -5, -6]

abjad> list(sequencetools.partition_sequence_by_sign_of_elements(sequence, sign = [-1, 0, 1]))
[[0, 0], [-1, -1], [2, 3], [-5], [1, 2, 5], [-5, -6]]
```

When -1 in sign, group negative elements.

When 0 in sign, group 0 elements.

When 1 in sign, group positive elements.

Return list of tuples of *sequence* element references. Changed in version 2.0: renamed `listtools.group_by_sign()` to `sequencetools.partition_sequence_by_sign_of_elements()`.

sequencetools.partition_sequence_by_value_of_elements

`abjad.tools.sequencetools.partition_sequence_by_value_of_elements.partition_sequence_by_value_of_elements`

New in version 1.1. Group *sequence* elements by equality:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.partition_sequence_by_value_of_elements([0, 0, -1, -1, 2, 3, -5, 1, 1, 5, -5, -6])
[(0, 0), (-1, -1), (2,), (3,), (-5,), (1, 1), (5,), (-5,)]
```

Return list of tuples of *sequence* element references. Changed in version 2.0: renamed `sequencetools.group_by_equality()` to `sequencetools.partition_sequence_by_value_of_elements()`.

`sequencetools.partition_sequence_cyclically_by_counts_with_overhang`

`abjad.tools.sequencetools.partition_sequence_cyclically_by_counts_with_overhang.partition_`

New in version 1.1. Partition *sequence* cyclically by *counts* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_cyclically_by_counts_with_overhang(range(16), [4, 6])
[[0, 1, 2, 3], [4, 5, 6, 7, 8, 9], [10, 11, 12, 13], [14, 15]]
```

Return list of *sequence* objects. Changed in version 2.0: renamed `listtools.partition_sequence_cyclically_by_counts_with_overhang()` to `sequencetools.partition_sequence_cyclically_by_counts_with_overhang()`.

`sequencetools.partition_sequence_cyclically_by_counts_without_overhang`

`abjad.tools.sequencetools.partition_sequence_cyclically_by_counts_without_overhang.partition_`

New in version 1.1. Partition *sequence* cyclically by *counts* without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_cyclically_by_counts_without_overhang(range(16), [4, 6])
[[0, 1, 2, 3], [4, 5, 6, 7, 8, 9], [10, 11, 12, 13]]
```

Return list of *sequence* objects. Changed in version 2.0: renamed `listtools.partition_sequence_cyclically_by_counts_without_overhang()` to `sequencetools.partition_sequence_cyclically_by_counts_without_overhang()`.

`sequencetools.partition_sequence_cyclically_by_weights_at_least_with_overhang`

`abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_at_least_with_overhang.p`

New in version 1.1. Partition *sequence* elements cyclically by *weights* at least with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_at_least_with_overhang(sequence, [
[[3, 3, 3, 3], [4], [4, 4, 4], [5], [5]]
```

Return list *sequence* element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_at_least_with_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_at_least_with_overhang()`.

sequencetools.partition_sequence_cyclically_by_weights_at_least_without_overhang

```
abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_at_least_without_overhang
```

New in version 1.1. Partition *sequence* elements cyclically by *weights* at least without overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_at_least_without_overhang(sequence)
[[3, 3, 3, 3], [4], [4, 4, 4], [5]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_at_least_without_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_at_least_without_overhang()`.

sequencetools.partition_sequence_cyclically_by_weights_at_most_with_overhang

```
abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_at_most_with_overhang.p
```

New in version 1.1. Partition *sequence* elements cyclically by *weights* at most with overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_at_most_with_overhang(sequence, [1])
[[3, 3, 3], [3], [4, 4], [4], [4, 5], [5]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_at_most_with_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_at_most_with_overhang()`.

sequencetools.partition_sequence_cyclically_by_weights_at_most_without_overhang

```
abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_at_most_without_overhang
```

New in version 1.1. Partition *sequence* elements cyclically by *weights* at most without overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_at_most_without_overhang(sequence, [1])
[[3, 3, 3], [3], [4, 4], [4]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_at_most_without_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_at_most_without_overhang()`.

sequencetools.partition_sequence_cyclically_by_weights_exactly_with_overhang

```
abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_exactly_with_overhang.p
```

New in version 1.1. Partition *sequence* elements cyclically by *weights* exactly with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_exactly_with_overhang(sequence, [1, 1, 1, 1, 1, 1, 1, 1, 1])
[[3, 3, 3, 3], [4, 4, 4], [4, 5]]
```

Return list of sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_exactly_with_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_exactly_with_overhang()`.

`sequencetools.partition_sequence_cyclically_by_weights_exactly_without_overhang`

```
abjad.tools.sequencetools.partition_sequence_cyclically_by_weights_exactly_without_overhang
```

New in version 1.1. Partition *sequence* elements cyclically by *weights* exactly without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5]
abjad> sequencetools.partition_sequence_cyclically_by_weights_exactly_without_overhang(sequence, [1, 1, 1, 1, 1, 1, 1, 1, 1])
[[3, 3, 3, 3], [4, 4, 4]]
```

Return list of sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_cyclically_by_weights_exactly_without_overhang()` to `sequencetools.partition_sequence_cyclically_by_weights_exactly_without_overhang()`.

`sequencetools.partition_sequence_extended_to_counts_with_overhang`

```
abjad.tools.sequencetools.partition_sequence_extended_to_counts_with_overhang.partition_se
```

New in version 2.0. Partition *sequence* extended to *counts* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_extended_to_counts_with_overhang([1, 2, 3, 4], [6, 6, 6])
[[1, 2, 3, 4, 1, 2], [3, 4, 1, 2, 3, 4], [1, 2, 3, 4, 1, 2], [3, 4]]
```

Return new object of *sequence* type.

`sequencetools.partition_sequence_extended_to_counts_without_overhang`

```
abjad.tools.sequencetools.partition_sequence_extended_to_counts_without_overhang.partition
```

New in version 2.0. Partition *sequence* extended to *counts* without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_extended_to_counts_without_overhang([1, 2, 3, 4], [6, 6])
[[1, 2, 3, 4, 1, 2], [3, 4, 1, 2, 3, 4], [1, 2, 3, 4, 1, 2]]
```

Return new object of *sequence* type.

sequencetools.partition_sequence_once_by_counts_with_overhang

`abjad.tools.sequencetools.partition_sequence_once_by_counts_with_overhang`. **partition_sequenc**

New in version 1.1. Partition *sequence* once by *counts* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_once_by_counts_with_overhang(range(16), [4, 6])
[[0, 1, 2, 3], [4, 5, 6, 7, 8, 9], [10, 11, 12, 13, 14, 15]]
```

Return list of *sequence* objects. Changed in version 2.0: renamed
`listtools.partition_sequence_once_by_counts_with_overhang()` to
`sequencetools.partition_sequence_once_by_counts_with_overhang()`.

sequencetools.partition_sequence_once_by_counts_without_overhang

`abjad.tools.sequencetools.partition_sequence_once_by_counts_without_overhang`. **partition_sequenc**

New in version 1.1. Partition *sequence* once by *counts* without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.partition_sequence_once_by_counts_without_overhang(range(16), [4, 6])
[[0, 1, 2, 3], [4, 5, 6, 7, 8, 9]]
```

Return list of *sequence* objects. Changed in version 2.0: renamed
`listtools.partition_sequence_once_by_counts_without_overhang()` to
`sequencetools.partition_sequence_once_by_counts_without_overhang()`.

sequencetools.partition_sequence_once_by_weights_at_least_with_overhang

`abjad.tools.sequencetools.partition_sequence_once_by_weights_at_least_with_overhang`. **partiti**

New in version 1.1. Partition *sequence* elements once by *weights* at least with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_at_least_with_overhang(sequence, [10, 4])
[[3, 3, 3, 3], [4], [4, 4, 4, 5, 5]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed
`sequencetools.group_sequence_elements_once_by_weights_at_least_with_overhang()`
to `sequencetools.partition_sequence_once_by_weights_at_least_with_overhang()`.

sequencetools.partition_sequence_once_by_weights_at_least_without_overhang

`abjad.tools.sequencetools.partition_sequence_once_by_weights_at_least_without_overhang`. **partiti**

New in version 1.1. Partition *sequence* elements once by *weights* at least without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_at_least_without_overhang(sequence, [10],
[[3, 3, 3, 3], [4]])
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_once_by_weights_at_least_without_overhang()` to `sequencetools.partition_sequence_once_by_weights_at_least_without_overhang()`.

`sequencetools.partition_sequence_once_by_weights_at_most_with_overhang`

```
abjad.tools.sequencetools.partition_sequence_once_by_weights_at_most_with_overhang.partiti
```

New in version 1.1. Partition *sequence* elements once by *weights* at most with overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_at_most_with_overhang(sequence, [10, 4])
[[3, 3, 3], [3], [4, 4, 4, 4, 5, 5]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_once_by_weights_at_most_with_overhang()` to `sequencetools.partition_sequence_once_by_weights_at_most_with_overhang()`.

`sequencetools.partition_sequence_once_by_weights_at_most_without_overhang`

```
abjad.tools.sequencetools.partition_sequence_once_by_weights_at_most_without_overhang.partiti
```

New in version 1.1. Partition *sequence* elements once by *weights* at most without overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_at_most_without_overhang(sequence, [10],
[[3, 3, 3], [3]])
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_once_by_weights_at_most_without_overhang()` to `sequencetools.partition_sequence_once_by_weights_at_most_without_overhang()`.

`sequencetools.partition_sequence_once_by_weights_exactly_with_overhang`

```
abjad.tools.sequencetools.partition_sequence_once_by_weights_exactly_with_overhang.partiti
```

New in version 1.1. Partition *sequence* elements once by *weights* exactly with overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_exactly_with_overhang(sequence, [3, 9])
[[3], [3, 3, 3], [4, 4, 4, 4, 5, 5]]
```

Return list sequence element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_once_by_weights_exactly_with_overhang()` to `sequencetools.partition_sequence_once_by_weights_exactly_with_overhang()`.

sequencetools.partition_sequence_once_by_weights_exactly_without_overhang

`abjad.tools.sequencetools.partition_sequence_once_by_weights_exactly_without_overhang.partition`

New in version 1.1. Partition *sequence* elements once by *weights* exactly without overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [3, 3, 3, 3, 4, 4, 4, 4, 5, 5]
abjad> sequencetools.partition_sequence_once_by_weights_exactly_without_overhang(sequence, [3, 9])
[[3], [3, 3, 3]]
```

Return list *sequence* element reference lists. Changed in version 2.0: renamed `sequencetools.group_sequence_elements_once_by_weights_exactly_without_overhang()` to `sequencetools.partition_sequence_once_by_weights_exactly_without_overhang()`.

sequencetools.permute_sequence

`abjad.tools.sequencetools.permute_sequence.permute_sequence(sequence, permutation)`

New in version 2.0. Permute *sequence* by *permutation*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.permute_sequence([10, 11, 12, 13, 14, 15], [5, 4, 0, 1, 2, 3])
[15, 14, 10, 11, 12, 13]
```

Return newly constructed *sequence* object.

sequencetools.remove_sequence_elements_at_indices

`abjad.tools.sequencetools.remove_sequence_elements_at_indices.remove_sequence_elements_at_indices`

New in version 2.0. Remove *sequence* elements at *indices*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.remove_sequence_elements_at_indices(range(20), [1, 16, 17, 18])
[0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19]
```

Ignore negative indices.

Return list.

sequencetools.remove_sequence_elements_at_indices_cyclically

`abjad.tools.sequencetools.remove_sequence_elements_at_indices_cyclically.remove_sequence_e`

New in version 2.0. Remove *sequence* elements at *indices mod period* plus *offset*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.remove_sequence_elements_at_indices_cyclically(range(20), [0, 1], 5, 3)
[0, 1, 2, 5, 6, 7, 10, 11, 12, 15, 16, 17]
```

Ignore negative indices.

Return list.

sequencetools.remove_subsequence_of_weight_at_index

```
abjad.tools.sequencetools.remove_subsequence_of_weight_at_index.remove_subsequence_of_weight
```

New in version 1.1. Remove subsequence of *weight* at *index*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.remove_subsequence_of_weight_at_index((1, 1, 2, 3, 5, 5, 1, 2, 5, 5, 6), 13)
(1, 1, 2, 3, 5, 5, 6)
```

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.remove_weighted_subrun_at()` to `sequencetools.remove_subsequence_of_weight_at_index()`.

sequencetools.repeat_runs_in_sequence_to_count

```
abjad.tools.sequencetools.repeat_runs_in_sequence_to_count.repeat_runs_in_sequence_to_count
```

New in version 1.1. Repeat subruns in *sequence* according to *indicators*. The *indicators* input parameter must be a list of zero or more (start, length, count) triples. For every (start, length, count) indicator in *indicators*, the function copies `sequence[start:start+length]` and inserts count new copies of `sequence[start:start+length]` immediately after `sequence[start:start+length]` in *sequence*.

Note: The function reads the value of count in every (start, length, count) triple not as the total number of occurrences of `sequence[start:start+length]` to appear in *sequence* after execution, but rather as the number of new occurrences of `sequence[start:start+length]` to appear in *sequence* after execution.

Note: The function wraps newly created subruns in tuples. That is, this function returns output with one more level of nesting than given in input.

To insert 10 count of `sequence[:2]` at `sequence[2:2]`:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.repeat_runs_in_sequence_to_count(range(20), [(0, 2, 10)])
[0, 1, (0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1),
 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```


To insert 5 count of `sequence[10:12]` at `sequence[12:12]` and then insert 5 count of `sequence[:2]` at `sequence[2:2]`:

```
abjad> sequence = range(20)
```

```
abjad> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(0, 2, 5), (10, 2, 5)])
[0, 1, (0, 1, 0, 1, 0, 1, 0, 1, 0, 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, (10, 11, 10, 11, 10, 11,
```

Note: This function wraps around the end of *sequence* whenever `len(sequence) < start + length`.

To insert 2 count of `[18, 19, 0, 1]` at `sequence[2:2]`:

```
abjad> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(18, 4, 2)])
[0, 1, (18, 19, 0, 1, 18, 19, 0, 1), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
```

To insert 2 count of `[18, 19, 0, 1, 2, 3, 4]` at `sequence[4:4]`:

```
abjad> sequencetools.repeat_runs_in_sequence_to_count(sequence, [(18, 8, 2)])
[0, 1, 2, 3, 4, 5, (18, 19, 0, 1, 2, 3, 4, 5, 18, 19, 0, 1, 2, 3, 4, 5), 6, 7, 8, 9, 10, 11, 12,
```

Todo

Implement an optional *wrap* keyword to specify whether this function should wrap around the end of *sequence* whenever `len(sequence) < start + length` or not.

Todo

Reimplement this function to return a generator.

Generalizations of this function would include functions to repeat subruns in *sequence* to not only a certain count, as implemented here, but to a certain length, weight or sum. That is, `sequencetools.repeat_subruns_to_length()`, `sequencetools.repeat_subruns_to_weight()` and `sequencetools.repeat_subruns_to_sum()`. Changed in version 2.0: renamed `sequencetools.repeat_subruns_to_count()` to `sequencetools.repeat_runs_in_sequence_to_count()`.

sequencetools.repeat_sequence_elements_at_indices

```
abjad.tools.sequencetools.repeat_sequence_elements_at_indices.repeat_sequence_elements_at_
```

New in version 2.0. Repeat *sequence* elements at *indices* to *total* length:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.repeat_sequence_elements_at_indices(range(10), [6, 7, 8], 3)
[0, 1, 2, 3, 4, 5, [6, 6, 6], [7, 7, 7], [8, 8, 8], 9]
```

Return list.

sequencetools.repeat_sequence_elements_at_indices_cyclically

`abjad.tools.sequencetools.repeat_sequence_elements_at_indices_cyclically.repeat_sequence_e...`

New in version 2.0. Repeat *sequence* elements at indices specified by *cycle_token* to *total* length:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_elements_at_indices_cyclically(range(10), (5, [1, 2]), 3)
[0, [1, 1, 1], [2, 2, 2], 3, 4, 5, [6, 6, 6], [7, 7, 7], 8, 9]
```

The *cycle_token* may be a sieve:

```
abjad> from abjad.tools import sievetools

abjad> sieve = sievetools.cycle_tokens_to_sieve((5, [1, 2]))
abjad> sequencetools.repeat_sequence_elements_at_indices_cyclically(range(10), sieve, 3)
[0, [1, 1, 1], [2, 2, 2], 3, 4, 5, [6, 6, 6], [7, 7, 7], 8, 9]
```

Return list.

sequencetools.repeat_sequence_elements_n_times_each

`abjad.tools.sequencetools.repeat_sequence_elements_n_times_each.repeat_sequence_elements_n...`

New in version 1.1. Repeat *sequence* elements *n* times each:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_elements_n_times_each((1, -1, 2, -3, 5, -5, 6), 2)
(1, 1, -1, -1, 2, 2, -3, -3, 5, 5, -5, -5, 6, 6)
```

Return newly constructed *sequence* object with copied *sequence* elements. Changed in version 2.0: renamed `listtools.repeat_elements_to_count()` to `sequencetools.repeat_sequence_elements_n_times_each()`.

sequencetools.repeat_sequence_n_times

`abjad.tools.sequencetools.repeat_sequence_n_times.repeat_sequence_n_times(sequence, n)`

New in version 2.0. Repeat *sequence* *n* times:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_n_times((1, 2, 3, 4, 5), 3)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

Repeat *sequence* 0 times:

```
abjad> sequencetools.repeat_sequence_n_times((1, 2, 3, 4, 5), 0)
()
```

Return newly constructed *sequence* object of copied *sequence* elements.

sequencetools.repeat_sequence_to_length

`abjad.tools.sequencetools.repeat_sequence_to_length.repeat_sequence_to_length(sequence, length, start=0)`

New in version 1.1. Repeat *sequence* to nonnegative integer *length*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_to_length(range(5), 11)
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0]
```

Repeat *sequence* to nonnegative integer *length* from *start*:

```
abjad> sequencetools.repeat_sequence_to_length(range(5), 11, start = 2)
[2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2]
```

Return newly constructed *sequence* object. Changed in version 2.0: renamed `listtools.repeat_list_to_length()` to `sequencetools.repeat_sequence_to_length()`.

sequencetools.repeat_sequence_to_weight_at_least

`abjad.tools.sequencetools.repeat_sequence_to_weight_at_least.repeat_sequence_to_weight_at_least(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* at least:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_to_weight_at_least((5, -5, -5), 23)
(5, -5, -5, 5, -5)
```

Return newly constructed *sequence* object.

sequencetools.repeat_sequence_to_weight_at_most

`abjad.tools.sequencetools.repeat_sequence_to_weight_at_most.repeat_sequence_to_weight_at_most(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* at most:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.repeat_sequence_to_weight_at_most((5, -5, -5), 23)
(5, -5, -5, 5)
```

Return newly constructed *sequence* object.

sequencetools.repeat_sequence_to_weight_exactly

`abjad.tools.sequencetools.repeat_sequence_to_weight_exactly.repeat_sequence_to_weight_exactly(sequence, weight)`

New in version 1.1. Repeat *sequence* to *weight* exactly:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.repeat_sequence_to_weight_exactly((5, -5, -5), 23)
(5, -5, -5, 5, -3)
```

Return newly constructed *sequence* object.

sequencetools.replace_sequence_elements_cyclically_with_new_material

```
abjad.tools.sequencetools.replace_sequence_elements_cyclically_with_new_material.replace_s
```

New in version 1.1. Replace *sequence* elements cyclically at *indices* with *new_material*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.replace_sequence_elements_cyclically_with_new_material(range(20), ([0], 2),
['A', 1, 'B', 3, 4, 5, 'A', 7, 'B', 9, 10, 11, 'A', 13, 'B', 15, 16, 17, 'A', 19])

abjad> sequencetools.replace_sequence_elements_cyclically_with_new_material(range(20), ([0], 2),
['*', 1, '*', 3, '*', 5, '*', 7, '*', 9, '*', 11, '*', 13, '*', 15, '*', 17, '*', 19])

abjad> sequencetools.replace_sequence_elements_cyclically_with_new_material(range(20), ([0], 2),
['A', 1, 'B', 3, 'C', 5, 'D', 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

abjad> sequencetools.replace_sequence_elements_cyclically_with_new_material(range(20), ([0, 1, 8], 2),
['A', 'B', 2, 3, 4, 5, 6, 7, 'C', 9, 10, 11, 12, 'D', 14, 15, 16, 17, 18, 19])
```

Raise type error when *sequence* not a list.

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.replace_elements_cyclic()` to `sequencetools.replace_sequence_elements_cyclically()`

sequencetools.retain_sequence_elements_at_indices

```
abjad.tools.sequencetools.retain_sequence_elements_at_indices.retain_sequence_elements_at_
```

New in version 2.0. Retain *sequence* elements at *indices*:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.retain_sequence_elements_at_indices(range(20), [1, 16, 17, 18])
[1, 16, 17, 18]
```

Return sequence elements in the order they appear in *sequence*.

Ignore negative indices.

Return list.

sequencetools.retain_sequence_elements_at_indices_cyclically

`abjad.tools.sequencetools.retain_sequence_elements_at_indices_cyclically`.**retain_sequence_e**

New in version 2.0. Retain *sequence* elements at *indices* mod *period* plus *offset*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.retain_sequence_elements_at_indices_cyclically(range(20), [0, 1], 5, 3)
[3, 4, 8, 9, 13, 14, 18, 19]
```

Ignore negative values in *indices*.

Return list.

sequencetools.reverse_sequence

`abjad.tools.sequencetools.reverse_sequence`.**reverse_sequence** (*sequence*)

New in version 2.0. Reverse *sequence*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.reverse_sequence((1, 2, 3, 4, 5))
(5, 4, 3, 2, 1)
```

Return new *sequence* object.

sequencetools.reverse_sequence_elements

`abjad.tools.sequencetools.reverse_sequence_elements`.**reverse_sequence_elements** (*sequence*)

New in version 2.0. Reverse *sequence* elements:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.reverse_sequence_elements([1, (2, 3, 4), 5, (6, 7)])
[1, (4, 3, 2), 5, (7, 6)]
```

Return new *sequence* object.

sequencetools.rotate_sequence

`abjad.tools.sequencetools.rotate_sequence`.**rotate_sequence** (*sequence*, *n*)

New in version 1.1. Rotate *sequence* to the right:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.rotate_sequence(range(10), 4)
[6, 7, 8, 9, 0, 1, 2, 3, 4, 5]
```

Rotate *sequence* to the left:

```
abjad> sequencetools.rotate_sequence(range(10), -3)
[3, 4, 5, 6, 7, 8, 9, 0, 1, 2]
```

Rotate *sequence* neither to the right nor the left:

```
abjad> sequencetools.rotate_sequence(range(10), 0)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Return newly created *sequence* object. Changed in version 2.0: renamed `sequencetools.rotate()` to `sequencetools.rotate_sequence()`.

sequencetools.splice_new_elements_between_sequence_elements

`abjad.tools.sequencetools.splice_new_elements_between_sequence_elements.splice_new_elements`

New in version 1.1. Splice copies of *new_elements* between each of the elements of *sequence*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [0, 1, 2, 3, 4]
abjad> new_elements = ['A', 'B']

abjad> sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements)
[0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4]
```

Splice copies of *new_elements* between each of the elements of *sequence* and after the last element of *sequence*:

```
abjad> sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements, overhang=1)
[0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4, 'A', 'B']
```

Splice copies of *new_elements* before the first element of *sequence* and between each of the other elements of *sequence*:

```
abjad> sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements, overhang=0,
['A', 'B', 0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4])
```

Splice copies of *new_elements* before the first element of *sequence*, after the last element of *sequence* and between each of the other elements of *sequence*:

```
abjad> sequencetools.splice_new_elements_between_sequence_elements(sequence, new_elements, overhang=1,
['A', 'B', 0, 'A', 'B', 1, 'A', 'B', 2, 'A', 'B', 3, 'A', 'B', 4, 'A', 'B'])
```

Return newly constructed list. Changed in version 2.0: renamed `sequencetools.insert_slice_cyclic()` to `sequencetools.splice_new_elements_between_sequence_elements()`.

sequencetools.split_sequence_cyclically_by_weights_with_overhang

`abjad.tools.sequencetools.split_sequence_cyclically_by_weights_with_overhang.split_sequence`

New in version 2.0. Split *sequence* cyclically by *weights* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.split_sequence_cyclically_by_weights_with_overhang((10, -10, 10, -10), [3,
[(3,), (7, -8), (-2, 1), (3,), (6, -9), (-1,)]
```

Return list of *sequence* objects.

sequencetools.split_sequence_cyclically_by_weights_without_overhang

```
abjad.tools.sequencetools.split_sequence_cyclically_by_weights_without_overhang.split_sequ
```

New in version 2.0. Split *sequence* cyclically by *weights* without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.split_sequence_cyclically_by_weights_without_overhang((10, -10, 10, -10), [
[(3,), (7, -8), (-2, 1), (3,), (6, -9)]
```

Return list of *sequence* objects.

sequencetools.split_sequence_extended_to_weights_with_overhang

```
abjad.tools.sequencetools.split_sequence_extended_to_weights_with_overhang.split_sequence_e
```

New in version 2.0. Split *sequence* extended to *weights* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.split_sequence_extended_to_weights_with_overhang([1, 2, 3, 4, 5], [7, 7, 7]
[[1, 2, 3, 1], [3, 4], [1, 1, 2, 3], [4, 5]]
```

Return new object of *sequence* type.

sequencetools.split_sequence_extended_to_weights_without_overhang

```
abjad.tools.sequencetools.split_sequence_extended_to_weights_without_overhang.split_sequenc
```

New in version 2.0. Split *sequence* extended to *weights* without overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.split_sequence_extended_to_weights_without_overhang([1, 2, 3, 4, 5], [7, 7,
[[1, 2, 3, 1], [3, 4], [1, 1, 2, 3]]
```

Return new object of *sequence* type.

sequencetools.split_sequence_once_by_weights_with_overhang

```
abjad.tools.sequencetools.split_sequence_once_by_weights_with_overhang.split_sequence_once
```

New in version 2.0. Split *sequence* once by *weights* with overhang:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.split_sequence_once_by_weights_with_overhang((10, -10, 10, -10), [3, 15, 3]
[(3,), (7, -8), (-2, 1), (9, -10)]
```

Return list of *sequence* objects.

sequencetools.split_sequence_once_by_weights_without_overhang

`abjad.tools.sequencetools.split_sequence_once_by_weights_without_overhang`.**split_sequence_once_by_weights_without_overhang**

New in version 2.0. Split *sequence* once by *weights* without overhang:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.split_sequence_once_by_weights_without_overhang((10, -10, 10, -10), [3, 15, 10, 10])
[(3,), (7, -8), (-2, 1)]
```

Return list of *sequence* objects.

sequencetools.sum_consecutive_sequence_elements_by_sign

`abjad.tools.sequencetools.sum_consecutive_sequence_elements_by_sign`.**sum_consecutive_sequence_elements_by_sign**

New in version 1.1. Sum consecutive *sequence* elements by *sign*:

```
abjad> from abjad.tools import sequencetools

abjad> sequence = [0, 0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence)
[0, -2, 5, -5, 8, -11]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [-1])
[0, 0, -2, 2, 3, -5, 1, 2, 5, -11]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [0])
[0, -1, -1, 2, 3, -5, 1, 2, 5, -5, -6]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [1])
[0, 0, -1, -1, 5, -5, 8, -5, -6]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [-1, 0])
[0, -2, 2, 3, -5, 1, 2, 5, -11]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [-1, 1])
[0, 0, -2, 5, -5, 8, -11]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [0, 1])
[0, -1, -1, 5, -5, 8, -5, -6]

abjad> sequencetools.sum_consecutive_sequence_elements_by_sign(sequence, sign = [-1, 0, 1])
[0, -2, 5, -5, 8, -11]
```

When `-1` in *sign*, sum consecutive negative elements.

When `0` in *sign*, sum consecutive 0 elements.

When `1` in *sign*, sum consecutive positive elements.

Return list. Changed in version 2.0: renamed `sequencetools.sum_by_sign()` to `sequencetools.sum_consecutive_sequence_elements_by_sign()`.

sequencetools.sum_sequence_elements_at_indices

`abjad.tools.sequencetools.sum_sequence_elements_at_indices.sum_sequence_elements_at_indices`

New in version 1.1. Sum *sequence* elements at indices according to *pairs*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.sum_sequence_elements_at_indices(range(10), [(0, 3)])
[3, 3, 4, 5, 6, 7, 8, 9]
```

Sum *sequence* elements cyclically at indices according to *pairs* and *period*:

```
abjad> sequencetools.sum_sequence_elements_at_indices(range(10), [(0, 3)], period = 4)
[3, 3, 15, 7, 17]
```

Sum *sequence* elements cyclically at indices according to *pairs* and *period* and do not return incomplete final sum:

```
abjad> sequencetools.sum_sequence_elements_at_indices(range(10), [(0, 3)], period = 4, overhang = 1)
[3, 3, 15, 7]
```

Replace `sequence[i:i+count]` with `sum(sequence[i:i+count])` for each `(i, count)` in *pairs*.

Indices in *pairs* must be less than *period* when *period* is not none.

Return new list. Changed in version 2.0: renamed `sequencetools.sum_slices_at()` to `sequencetools.sum_sequence_elements_at_indices()`.

sequencetools.truncate_runs_in_sequence

`abjad.tools.sequencetools.truncate_runs_in_sequence.truncate_runs_in_sequence(sequence)`

New in version 1.1. Truncate subruns of like elements in *sequence* to length 1:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> sequencetools.truncate_runs_in_sequence([1, 1, 2, 3, 3, 3, 9, 4, 4, 4])
[1, 2, 3, 9, 4]
```

Return empty list when *sequence* is empty:

```
abjad> sequencetools.truncate_runs_in_sequence([])
[]
```

Raise type error when *sequence* is not a list.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_subruns()` to `sequencetools.truncate_runs_in_sequence()`.

sequencetools.truncate_sequence_to_sum

`abjad.tools.sequencetools.truncate_sequence_to_sum.truncate_sequence_to_sum(sequence, target_sum)`

New in version 1.1. Truncate *sequence* to *target_sum*:

```
abjad> from abjad.tools import sequencetools

abjad> for n in range(10):
...     print n, sequencetools.truncate_sequence_to_sum([-1, 2, -3, 4, -5, 6, -7, 8, -9, 10], n)
...
0 []
1 [-1, 2]
2 [-1, 2, -3, 4]
3 [-1, 2, -3, 4, -5, 6]
4 [-1, 2, -3, 4, -5, 6, -7, 8]
5 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
6 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
7 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
8 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
9 [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
```

Return empty list when *target_sum* is 0:

```
abjad> sequencetools.truncate_sequence_to_sum([1, 2, 3, 4, 5], 0)
[]
```

Raise type error when *sequence* is not a list.

Raise value error on negative *target_sum*.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_to_sum()` to `sequencetools.truncate_sequence_to_sum()`.

sequencetools.truncate_sequence_to_weight

`abjad.tools.sequencetools.truncate_sequence_to_weight.truncate_sequence_to_weight(sequence, weight)`

New in version 1.1. Truncate *sequence* to *weight*:

```
abjad> from abjad.tools import sequencetools

abjad> l = [-1, 2, -3, 4, -5, 6, -7, 8, -9, 10]
abjad> for x in range(10):
...     print x, sequencetools.truncate_sequence_to_weight(l, x)
...
0 []
1 [-1]
2 [-1, 1]
3 [-1, 2]
4 [-1, 2, -1]
5 [-1, 2, -2]
6 [-1, 2, -3]
7 [-1, 2, -3, 1]
8 [-1, 2, -3, 2]
9 [-1, 2, -3, 3]
```

Return empty list when *weight* is 0:

```
abjad> sequencetools.truncate_sequence_to_weight([1, 2, 3, 4, 5], 0)
[]
```

Raise type error when *sequence* is not a list.

Raise value error on negative *weight*.

Return new list. Changed in version 2.0: renamed `sequencetools.truncate_to_weight()` to `sequencetools.truncate_sequence_to_weight()`.

sequencetools.yield_all_combinations_of_sequence_elements

`abjad.tools.sequencetools.yield_all_combinations_of_sequence_elements.yield_all_combinations_of`

New in version 2.0. Yield all combinations of *sequence* in binary string order:

```
abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.yield_all_combinations_of_sequence_elements([1, 2, 3, 4]))
[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3], [4], [1, 4],
 [2, 4], [1, 2, 4], [3, 4], [1, 3, 4], [2, 3, 4], [1, 2, 3, 4]]
```

Yield all combinations of *sequence* greater than or equal to *min_length* in binary string order:

```
abjad> list(sequencetools.yield_all_combinations_of_sequence_elements([1, 2, 3, 4], min_length =
[1, 2, 3], [1, 2, 4], [1, 3, 4], [2, 3, 4], [1, 2, 3, 4]]
```

Yield all combinations of *sequence* less than or equal to *max_length* in binary string order:

```
abjad> list(sequencetools.yield_all_combinations_of_sequence_elements([1, 2, 3, 4], max_length =
[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [4], [1, 4], [2, 4], [3, 4]]
```

Yield all combinations of *sequence* greater than or equal to *min_length* and less than or equal to *max_length* in lex order:

```
abjad> list(sequencetools.yield_all_combinations_of_sequence_elements([1, 2, 3, 4], min_length =
[1, 2], [1, 3], [2, 3], [1, 4], [2, 4], [3, 4]]
```

Return generator of newly created *sequence* objects. Changed in version 2.0: renamed `sequencetools.sublists()` to `sequencetools.yield_all_combinations_of_sequence_elements()`.

sequencetools.yield_all_k_ary_sequences_of_length

`abjad.tools.sequencetools.yield_all_k_ary_sequences_of_length.yield_all_k_ary_sequences_of`

New in version 2.0. Generate all *k*-ary sequences of *length*:

```
abjad> from abjad.tools import sequencetools

abjad> for sequence in sequencetools.yield_all_k_ary_sequences_of_length(2, 3):
...     sequence
...
(0, 0, 0)
(0, 0, 1)
(0, 1, 0)
```

```
(0, 1, 1)
(1, 0, 0)
(1, 0, 1)
(1, 1, 0)
(1, 1, 1)
```

Return generator of tuples.

sequencetools.yield_all_pairs_between_sequences

`abjad.tools.sequencetools.yield_all_pairs_between_sequences.yield_all_pairs_between_sequences`

New in version 2.0. Yield all pairs between sequences *l* and *m*:

```
abjad> from abjad.tools import sequencetools

abjad> for pair in sequencetools.yield_all_pairs_between_sequences([1, 2, 3], [4, 5]):
...     pair
...
(1, 4)
(1, 5)
(2, 4)
(2, 5)
(3, 4)
(3, 5)
```

Return pair generator.

sequencetools.yield_all_partitions_of_sequence

`abjad.tools.sequencetools.yield_all_partitions_of_sequence.yield_all_partitions_of_sequence`

New in version 2.0. Yield all partitions of *sequence*:

```
abjad> from abjad.tools import sequencetools

abjad> for partition in sequencetools.yield_all_partitions_of_sequence([0, 1, 2, 3]):
...     partition
...
[[0, 1, 2, 3]]
[[0, 1, 2], [3]]
[[0, 1], [2, 3]]
[[0, 1], [2], [3]]
[[0], [1, 2, 3]]
[[0], [1, 2], [3]]
[[0], [1], [2, 3]]
[[0], [1], [2], [3]]
```

Return generator of newly created lists.

sequencetools.yield_all_permutations_of_sequence

`abjad.tools.sequencetools.yield_all_permutations_of_sequence.yield_all_permutations_of_sequence`

New in version 1.1. Yield all permutations of *sequence* in lex order:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.yield_all_permutations_of_sequence((1, 2, 3)))
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

Return generator of *sequence* objects. Changed in version 2.0: renamed `listtools.permutations()` to `sequencetools.yield_all_permutations_of_sequence()`.

sequencetools.yield_all_permutations_of_sequence_in_orbit

```
abjad.tools.sequencetools.yield_all_permutations_of_sequence_in_orbit.yield_all_permutation
```

New in version 2.0. Yield all permutations of *sequence* in orbit of *permutation* in lex order:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.yield_all_permutations_of_sequence_in_orbit((1, 2, 3, 4), [1, 2, 3, 0])
[(1, 2, 3, 4), (2, 3, 4, 1), (3, 4, 1, 2), (4, 1, 2, 3)]
```

Return generator of *sequence* objects.

sequencetools.yield_all_restricted_growth_functions_of_length

```
abjad.tools.sequencetools.yield_all_restricted_growth_functions_of_length.yield_all_restric
```

New in version 2.0. Generate all restricted growth functions of *length* in lex order:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> for rgf in sequencetools.yield_all_restricted_growth_functions_of_length(4):
...     rgf
...
(1, 1, 1, 1)
(1, 1, 1, 2)
(1, 1, 2, 1)
(1, 1, 2, 2)
(1, 1, 2, 3)
(1, 2, 1, 1)
(1, 2, 1, 2)
(1, 2, 1, 3)
(1, 2, 2, 1)
(1, 2, 2, 2)
(1, 2, 2, 3)
(1, 2, 3, 1)
(1, 2, 3, 2)
(1, 2, 3, 3)
(1, 2, 3, 4)
```

Return generator of tuples.

sequencetools.yield_all_rotations_of_sequence

`abjad.tools.sequencetools.yield_all_rotations_of_sequence.yield_all_rotations_of_sequence(sequence, n)`

New in version 2.0. Yield all n -rotations of *sequence* up to identity:

```
abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.yield_all_rotations_of_sequence([1, 2, 3, 4], -1))
[[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3]]
```

Return generator of *sequence* objects.

sequencetools.yield_all_set_partitions_of_sequence

`abjad.tools.sequencetools.yield_all_set_partitions_of_sequence.yield_all_set_partitions_of_sequence(sequence)`

New in version 2.0. Yield all set partitions of *sequence* in restricted growth function order:

```
abjad> from abjad.tools import sequencetools

abjad> for set_partition in sequencetools.yield_all_set_partitions_of_sequence([21, 22, 23, 24]):
...     set_partition
...
[[21, 22, 23, 24]]
[[21, 22, 23], [24]]
[[21, 22, 24], [23]]
[[21, 22], [23, 24]]
[[21, 22], [23], [24]]
[[21, 23, 24], [22]]
[[21, 23], [22, 24]]
[[21, 23], [22], [24]]
[[21, 24], [22, 23]]
[[21], [22, 23, 24]]
[[21], [22, 23], [24]]
[[21, 24], [22], [23]]
[[21], [22, 24], [23]]
[[21], [22], [23, 24]]
[[21], [22], [23], [24]]
```

Return generator of list of lists.

sequencetools.yield_all_subsequences_of_sequence

`abjad.tools.sequencetools.yield_all_subsequences_of_sequence.yield_all_subsequences_of_sequence(sequence, min_length)`

New in version 2.0. Yield all subsequences of *sequence* in lex order:

```
abjad> from abjad.tools import sequencetools

abjad> list(sequencetools.yield_all_subsequences_of_sequence([0, 1, 2]))
[[], [0], [0, 1], [0, 1, 2], [1], [1, 2], [2]]
```

Yield all subsequences of *sequence* greater than or equal to *min_length* in lex order:

```
abjad> list(sequencetools.yield_all_subsequences_of_sequence([0, 1, 2, 3, 4], min_length = 3))
[[0, 1, 2], [0, 1, 2, 3], [0, 1, 2, 3, 4], [1, 2, 3], [1, 2, 3, 4], [2, 3, 4]]
```

Yield all subsequences of *sequence* less than or equal to *max_length* in lex order:

```
abjad> list(sequencetools.yield_all_subsequences_of_sequence([0, 1, 2, 3, 4], max_length = 3))
[[], [0], [0, 1], [0, 1, 2], [1], [1, 2], [1, 2, 3], [2], [2, 3], [2, 3, 4], [3], [3, 4], [4]]
```

Yield all subsequences of *sequence* greater than or equal to *min_length* and less than or equal to *max_length* in lex order:

```
abjad> list(sequencetools.yield_all_subsequences_of_sequence([0, 1, 2, 3, 4], min_length = 3, max_length = 4))
[[0, 1, 2], [1, 2, 3], [2, 3, 4]]
```

Return generator of newly created *sequence* slices.

sequencetools.yield_all_unordered_pairs_of_sequence

`abjad.tools.sequencetools.yield_all_unordered_pairs_of_sequence.yield_all_unordered_pairs_of_sequence`
New in version 2.0. Yield all unordered pairs of *sequence*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.yield_all_unordered_pairs_of_sequence([1, 2, 3, 4]))
[(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)]
```

Yield all unordered pairs of length-1 *sequence*:

```
abjad> list(sequencetools.yield_all_unordered_pairs_of_sequence([1]))
[]
```

Yield all unordered pairs of empty *sequence*:

```
abjad> list(sequencetools.yield_all_unordered_pairs_of_sequence([]))
[]
```

Yield all unordered pairs of *sequence* with duplicate elements:

```
abjad> list(sequencetools.yield_all_unordered_pairs_of_sequence([1, 1, 1]))
[(1, 1), (1, 1), (1, 1)]
```

Pairs are tuples instead of sets to accommodate duplicate *sequence* elements.

Return generator.

sequencetools.yield_outer_product_of_sequences

`abjad.tools.sequencetools.yield_outer_product_of_sequences.yield_outer_product_of_sequences`
New in version 1.1. Yield outer product of *sequences*:

```
abjad> from abjad.tools import sequencetools
```

```
abjad> list(sequencetools.yield_outer_product_of_sequences([[1, 2, 3], ['a', 'b']]))
[[1, 'a'], [1, 'b'], [2, 'a'], [2, 'b'], [3, 'a'], [3, 'b']]
```

```
abjad> list(sequencetools.yield_outer_product_of_sequences([[1, 2, 3], ['a', 'b'], ['X', 'Y']]))
[[1, 'a', 'X'], [1, 'a', 'Y'], [1, 'b', 'X'], [1, 'b', 'Y'],
 [2, 'a', 'X'], [2, 'a', 'Y'], [2, 'b', 'X'], [2, 'b', 'Y'],
 [3, 'a', 'X'], [3, 'a', 'Y'], [3, 'b', 'X'], [3, 'b', 'Y']]

abjad> list(sequencetools.yield_outer_product_of_sequences([[1, 2, 3], [4, 5], [6, 7, 8]]))
[[1, 4, 6], [1, 4, 7], [1, 4, 8], [1, 5, 6], [1, 5, 7], [1, 5, 8],
 [2, 4, 6], [2, 4, 7], [2, 4, 8], [2, 5, 6], [2, 5, 7], [2, 5, 8],
 [3, 4, 6], [3, 4, 7], [3, 4, 8], [3, 5, 6], [3, 5, 7], [3, 5, 8]]
```

Return generator. Changed in version 2.0: renamed `sequencetools.outer_product()` to `sequencetools.yield_outer_product_of_sequences()`.

sequencetools.zip_sequences_cyclically

`abjad.tools.sequencetools.zip_sequences_cyclically.zip_sequences_cyclically(*sequences)`
New in version 1.1. Zip *sequences* cyclically:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.zip_sequences_cyclically([1, 2, 3], ['a', 'b'])
[(1, 'a'), (2, 'b'), (3, 'a')]
```

New in version 1.1: Arbitrary number of input sequences now allowed.

```
abjad> sequencetools.zip_sequences_cyclically([10, 11, 12], [20, 21], [30, 31, 32, 33])
[(10, 20, 30), (11, 21, 31), (12, 20, 32), (10, 21, 33)]
```

Cycle over the elements of the sequences of shorter length.

Return list of length equal to sequence of greatest length in *sequences*. Changed in version 2.0: renamed `sequencetools.zip_cyclic()` to `sequencetools.zip_sequences_cyclically()`.

sequencetools.zip_sequences_without_truncation

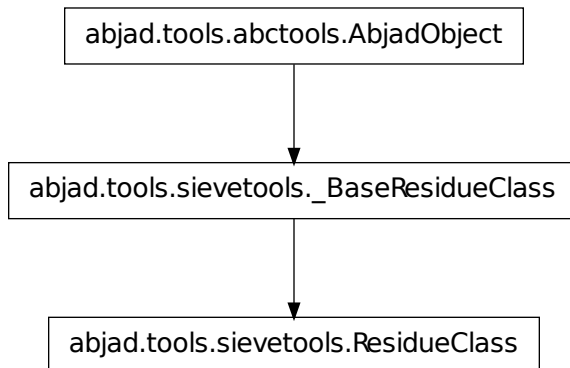
`abjad.tools.sequencetools.zip_sequences_without_truncation.zip_sequences_without_truncation(*sequences)`
New in version 1.1. Zip *sequences* nontruncating:

```
abjad> from abjad.tools import sequencetools

abjad> sequencetools.zip_sequences_without_truncation([1, 2, 3, 4], [11, 12, 13], [21, 22, 23])
[(1, 11, 21), (2, 12, 22), (3, 13, 23), (4,)]
```

Lengths of the tuples returned may differ but will always be greater than or equal to 1.

Return list of tuples. Changed in version 2.0: renamed `sequencetools.zip_nontruncating()` to `sequencetools.zip_sequences_without_truncation()`.

sievetools**sievetools concrete classes****sievetools.ResidueClass**

class `abjad.tools.sievetools.ResidueClass.ResidueClass(*args)`

Residue class (or congruence class). Residue classes form the basis of Xenakis sieves. They can be used to construct any complex periodic integer (or boolean) sequence as a combination of simple periodic sequences.

Example from the opening of Xenakis's *Psappha* for solo percussion:

```

abjad> from abjad.tools.sievetools import ResidueClass as RC

abjad> s1 = (RC(8, 0) | RC(8, 1) | RC(8, 7)) & (RC(5, 1) | RC(5, 3))
abjad> s2 = (RC(8, 0) | RC(8, 1) | RC(8, 2)) & RC(5, 0)
abjad> s3 = RC(8, 3)
abjad> s4 = RC(8, 4)
abjad> s5 = (RC(8, 5) | RC(8, 6)) & (RC(5, 2) | RC(5, 3) | RC(5, 4))
abjad> s6 = (RC(8, 1) & RC(5, 2))
abjad> s7 = (RC(8, 6) & RC(5, 1))

abjad> y = s1 | s2 | s3 | s4 | s5 | s6 | s7
abjad> y
{{ResidueClass(8, 0) | ResidueClass(8, 1) | ResidueClass(8, 7)} & {ResidueClass(5, 1) | ResidueClass(5, 3) | ResidueClass(5, 0) | ResidueClass(8, 3) | ResidueClass(8, 4) | ResidueClass(8, 5) | ResidueClass(8, 6)} & {ResidueClass(8, 1) & ResidueClass(5, 2)} & {ResidueClass(8, 6) & ResidueClass(5, 1)}}

abjad> y.get_congruent_bases(40)
[0, 1, 3, 4, 6, 8, 10, 11, 12, 13, 14, 16, 17, 19, 20, 22, 23, 25, 27,
 28, 29, 31, 33, 35, 36, 37, 38, 40]
abjad> y.get_boolean_train(40)
[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0]
  
```

Return residue class.

Read-only Properties

`ResidueClass.modulo`

Period of residue class.

`ResidueClass.residue`

Residue of residue class.

Methods

`ResidueClass.get_boolean_train(*min_max)`

Returns a boolean train with 0s mapped to the integers that are not congruent bases of the residue class and 1s mapped to those that are. The method takes one or two integer arguments. If only one is given, it is taken as the max range and the min is assumed to be 0.

Example:

```
abjad> from abjad.tools.sievetools import ResidueClass as RC

abjad> r = RC(3, 0)
abjad> r.get_boolean_train(6)
[1, 0, 0, 1, 0, 0]
abjad> r.get_congruent_bases(-6, 6)
[-6, -3, 0, 3, 6]
```

Return list.

`ResidueClass.get_congruent_bases(*min_max)`

Returns all the congruent bases of this residue class within the given range. The method takes one or two integer arguments. If only one it given, it is taken as the max range and the min is assumed to be 0.

Example:

```
abjad> from abjad.tools.sievetools import ResidueClass as RC

abjad> r = RC(3, 0)
abjad> r.get_congruent_bases(6)
[0, 3, 6]
abjad> r.get_congruent_bases(-6, 6)
[-6, -3, 0, 3, 6]
```

Return list.

Special Methods

`ResidueClass.__and__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

`ResidueClass.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ResidueClass.__eq__(exp)`

`ResidueClass.__ge__(expr)`

`ResidueClass.__gt__(expr)`

`ResidueClass.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`ResidueClass.__le__(expr)`

`ResidueClass.__lt__(expr)`

`ResidueClass.__ne__(expr)`

`ResidueClass.__or__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

`ResidueClass.__repr__()`

`ResidueClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

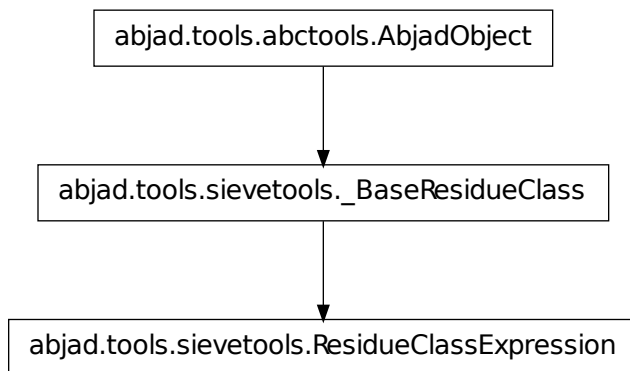
`ResidueClass.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`ResidueClass.__xor__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

sievetools.ResidueClassExpression



`class abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression`

Read-only Properties

`ResidueClassExpression.operator`

Operator of residue class expression.

`ResidueClassExpression.period`

`ResidueClassExpression.rcs`

Residue classes of expression.

`ResidueClassExpression.representative_boolean_train`

`ResidueClassExpression.representative_congruent_bases`

Methods

`ResidueClassExpression.get_boolean_train(*min_max)`

Returns a boolean train with 0s mapped to the integers that are not congruent bases of the RC expression and 1s mapped to those that are. The method takes one or two integer arguments. If only one is given, it is taken as the max range and min is assumed to be 0.

Example:

```
abjad> from abjad.tools.sievetools import ResidueClass as RC

abjad> e = RC(3, 0) | RC(2, 0)
abjad> e.get_boolean_train(6)
[1, 0, 1, 1, 1, 0]
abjad> e.get_congruent_bases(-6, 6)
[-6, -4, -3, -2, 0, 2, 3, 4, 6]
```

Return list.

`ResidueClassExpression.get_congruent_bases(*min_max)`

Returns all the congruent bases of this RC expression within the given range. The method takes one or two integer arguments. If only one is given, it is taken as the max range and min is assumed to be 0.

Example:

```
abjad> from abjad.tools.sievetools import ResidueClass as RC

abjad> e = RC(3, 0) | RC(2, 0)
abjad> e.get_congruent_bases(6)
[0, 2, 3, 4, 6]
abjad> e.get_congruent_bases(-6, 6)
[-6, -4, -3, -2, 0, 2, 3, 4, 6]
```

Return list.

`ResidueClassExpression.is_congruent_base(integer)`

Special Methods

`ResidueClassExpression.__and__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

`ResidueClassExpression.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ResidueClassExpression.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`ResidueClassExpression.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ResidueClassExpression.__or__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

`ResidueClassExpression.__repr__()`

`ResidueClassExpression.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ResidueClassExpression.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`ResidueClassExpression.__xor__(arg)`

Note: Inherited from `sievetools._BaseResidueClass`

sievetools functions

sievetools.all_are_residue_class_expressions

`abjad.tools.sievetools.all_are_residue_class_expressions.all_are_residue_class_expressions`
New in version 2.6. True when *expr* is a sequence of Abjad residue class expressions:

```
abjad> from abjad.tools import sievetools

abjad> sieve = sievetools.ResidueClass(3, 0) | sievetools.ResidueClass(2, 0)

abjad> sieve
{ResidueClass(2, 0) | ResidueClass(3, 0)}

abjad> sievetools.all_are_residue_class_expressions([sieve])
True
```

True when *expr* is an empty sequence:

```
abjad> sievetools.all_are_residue_class_expressions([])
True
```

Otherwise false:

```
abjad> sievetools.all_are_residue_class_expressions('foo')
False
```

Return boolean.

sievetools.cycle_tokens_to_sieve

abjad.tools.sievetools.cycle_tokens_to_sieve.**cycle_tokens_to_sieve**(*cycle_tokens)

New in version 2.0. Make Xenakis sieve from arbitrarily many *cycle_tokens*.

```
abjad> from abjad.tools import sievetools
```

```
abjad> cycle_token_1 = (6, [0, 4, 5])
```

```
abjad> cycle_token_2 = (10, [0, 1, 2], 6)
```

```
abjad> sievetools.cycle_tokens_to_sieve(cycle_token_1, cycle_token_2)
```

```
{ResidueClass(6, 0) | ResidueClass(6, 4) | ResidueClass(6, 5) | ResidueClass(10, 6) | ResidueClass(10, 2)}
```

Cycle token comprises mandatory *modulo*, mandatory *residues* and optional *offset*.

tempotools

tempotools functions

tempotools.integer_tempo_to_multiplier_tempo_pairs

abjad.tools.tempotools.integer_tempo_to_multiplier_tempo_pairs.**integer_tempo_to_multiplier_tempo_pairs**(integer_tempo, multiplier_tempo)

New in version 2.0. Return all multiplier, tempo pairs possible from *integer_tempo*.

Tempi must be no less than $\text{integer_tempo} / 2$ and not greater than $2 * \text{integer_tempo}$:

```
abjad> from abjad.tools import tempotools
```

```
abjad> pairs = tempotools.integer_tempo_to_multiplier_tempo_pairs(58, 8, 8)
```

```
abjad> for pair in pairs:
```

```
...     pair
```

```
...
```

```
(Fraction(1, 2), Fraction(29, 1))
```

```
(Fraction(1, 1), Fraction(58, 1))
```

```
(Fraction(3, 2), Fraction(87, 1))
```

```
(Fraction(2, 1), Fraction(116, 1))
```

Return list.

tempotools.integer_tempo_to_multiplier_tempo_pairs_report

abjad.tools.tempotools.integer_tempo_to_multiplier_tempo_pairs_report.**integer_tempo_to_multiplier_tempo_pairs_report**(integer_tempo, multiplier_tempo)

New in version 2.0. Print all multiplier, tempo pairs possible from *integer_tempo*.

Allow no tempi less than `integer_tempo / 2` nor greater than `2 * integer_tempo`:

```
abjad> from abjad.tools import tempotools
```

```
abjad> tempotools.integer_tempo_to_multiplier_tempo_pairs_report(58, 8, 8)
```

```
2:1      29
1:1      58
2:3      87
1:2     116
```

With more lenient numerator and denominator.

```
abjad> tempotools.integer_tempo_to_multiplier_tempo_pairs_report(58, 30, 30)
```

```
2:1      29
29:15    30
29:16    32
29:17    34
29:18    36
29:19    38
29:20    40
29:21    42
29:22    44
29:23    46
29:24    48
29:25    50
29:26    52
29:27    54
29:28    56
1:1      58
29:30    60
2:3      87
1:2     116
```

Return none.

threadtools

threadtools functions

threadtools.component_to_thread_signature

`abjad.tools.threadtools.component_to_thread_signature.component_to_thread_signature(component)`

Return `_ContainmentSignature` giving the root and first voice, staff and score in parentage of component.

threadtools.iterate_thread_backward_from_component

`abjad.tools.threadtools.iterate_thread_backward_from_component.iterate_thread_backward_from_component(component, klass)`

New in version 2.0. Yield right-to-left components in the thread of *component* starting from *component*.

When `klass = None` return all components in the thread of *component*.

When *klass* is set to some other Abjad class, yield only *klass* instances in the thread of *component*:

```
abjad> from abjad.tools import threadtools
```



```

abjad> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'voice 1'
abjad> container[1].name = 'voice 2'
abjad> staff = Staff(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> print staff.format
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "voice 2" {
      b'8
      c''8
    }
  >>
}

```

Starting from the last leaf in score.

```

abjad> for x in threadtools.iterate_thread_backward_from_component(staff.leaves[-1], Note):
...     x
Note("c'8")
Note("b'8")
Note("f'8")
Note("e'8")

```

Yield all components in thread:

```

abjad> for x in threadtools.iterate_thread_backward_from_component(staff.leaves[-1]):
...     x
Note("c'8")
Voice-"voice 2"{2}
Note("b'8")
Voice-"voice 2"{2}
Note("f'8")
Note("e'8")

```

Note that this function is a special type of depth-first search.

Compare	with	threadtools.iterate_thread_backward_in_expr().
Changed in version 2.0:	renamed	iterate.thread_backward_from() to
threadtools.iterate_thread_backward_from_component().	Changed	in
version 2.0:	renamed	iterate.thread_backward_from_component() to
threadtools.iterate_thread_backward_from_component().		

threadtools.iterate_thread_backward_in_expr

abjad.tools.threadtools.iterate_thread_backward_in_expr.**iterate_thread_backward_in_expr** (*expr*, *klass*, *thread_signature*)

New in version 2.0. Yield right-to-left instances of *klass* in *expr* with *thread_signature*:

```
abjad> from abjad.tools import threadtools

abjad> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'voice 1'
abjad> container[1].name = 'vocie 2'
abjad> staff = Staff(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> f(staff)
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "vocie 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "vocie 2" {
      b'8
      c''8
    }
  >>
}

abjad> signature = threadtools.component_to_thread_signature(staff[0])
abjad> for x in threadtools.iterate_thread_backward_in_expr(staff, Note, signature): # doctest:
...     x
Note("c'8")
Note("b'8")
Note("f'8")
Note("e'8")
```

The important thing to note is that the function yields only those leaves that sit in the same thread.

Compare with `componenttools.iterate_components_backward_in_expr()`.
 Changed in version 2.0: renamed `iterate_thread_backward_in()` to `threadtools.iterate_thread_backward_in_expr()`.

threadtools.iterate_thread_forward_from_component

abjad.tools.threadtools.iterate_thread_forward_from_component.**iterate_thread_forward_from_component** (*component*)

New in version 1.1. Yield left-to-right components in the thread of *component* starting from *component*.

When `klass = None` return all components in the thread of *component*.

When *klass* is set to some other Abjad class, yield only *klass* instances in the thread of *component*:

```
abjad> from abjad.tools import threadtools

abjad> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'voice 1'
abjad> container[1].name = 'voice 2'
abjad> staff = Staff(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> print staff.format
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "voice 2" {
      b'8
      c''8
    }
  >>
}
```

Starting from the first leaf in score.

```
abjad> for x in threadtools.iterate_thread_forward_from_component(staff.leaves[0], Note):
...     x
...
Note("c'8")
Note("d'8")
Note("g'8")
Note("a'8")
```

Starting from the second leaf in score.

```
abjad> for x in threadtools.iterate_thread_forward_from_component(staff.leaves[1], Note):
...     x
...
Note("d'8")
Note("g'8")
Note("a'8")
```

Yield all components in thread.

```
abjad> for x in threadtools.iterate_thread_forward_from_component(staff.leaves[0]):
...     x
...
```

```
Note("c'8")
Voice="voice 1"{2}
Note("d'8")
Voice="voice 1"{2}
Note("g'8")
Note("a'8")
```

Note that this function is a special type of depth-first search.

Compare with `threadtools.iterate_thread_forward_in_expr()`.
 Changed in version 2.0: renamed `iterate.thread_forward_from()` to
`threadtools.iterate_thread_forward_from_component()`. Changed in
 version 2.0: renamed `iterate.thread_forward_from_component()` to
`threadtools.iterate_thread_forward_from_component()`.

`threadtools.iterate_thread_forward_in_expr`

`abjad.tools.threadtools.iterate_thread_forward_in_expr.iterate_thread_forward_in_expr(expr,
klass,
thread)`

New in version 1.1. Yield left-to-right instances of *klass* in *expr* with *thread_signature*:

```
abjad> from abjad.tools import threadtools
```

```
abjad> container = Container(Voice(notetools.make_repeated_notes(2)) * 2)
abjad> container.is_parallel = True
abjad> container[0].name = 'voice 1'
abjad> container[1].name = 'voice 2'
abjad> staff = Staff(container * 2)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(staff)
abjad> print staff.format
\new Staff {
  <<
    \context Voice = "voice 1" {
      c'8
      d'8
    }
    \context Voice = "voice 2" {
      e'8
      f'8
    }
  >>
  <<
    \context Voice = "voice 1" {
      g'8
      a'8
    }
    \context Voice = "voice 2" {
      b'8
      c''8
    }
  >>
}
```

```
abjad> signature = threadtools.component_to_thread_signature(staff.leaves[0])
abjad> for x in threadtools.iterate_thread_forward_in_expr(staff, Note, signature):
...     x
```

```
...
Note("c'8")
Note("d'8")
Note("g'8")
Note("a'8")
```

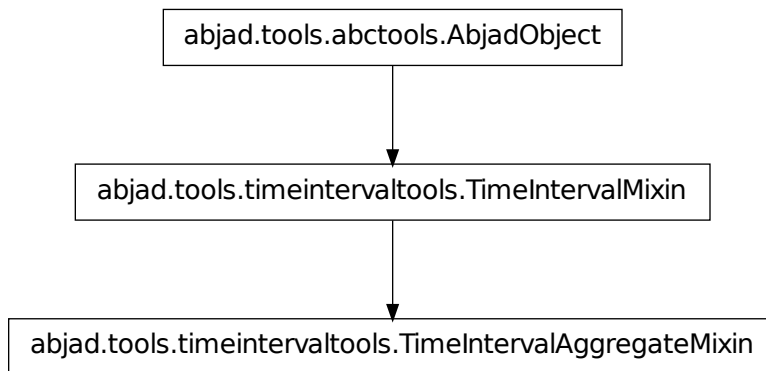
The important thing to note is that the function yields only those leaves that sit in the same thread.

```
Compare      with      componenttools.iterate_components_forward_in_expr().
Changed      in      version      2.0:      renamed      iterate.thread_forward_in()
to      threadtools.iterate_thread_forward_in_expr().Changed      in
version      2.0:      renamed      iterate.thread_forward_in_expr()      to
threadtools.iterate_thread_forward_in_expr().
```

timeintervaltools

timeintervaltools abstract classes

timeintervaltools.TimeIntervalAggregateMixin



```
class abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.Tin
```

Read-only Properties

TimeIntervalAggregateMixin.**bounds**

Start and stop of self returned as TimeInterval instance:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> bounds = interval.bounds
abjad> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
abjad> bounds == interval
True
```

```
abjad> bounds is interval
False
```

Returns *TimeInterval* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin`.**center**

Center offset of start and stop offsets:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin`.**duration**

Duration of the time interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin`.**earliest_start**

`TimeIntervalAggregateMixin`.**earliest_stop**

`TimeIntervalAggregateMixin`.**intervals**

`TimeIntervalAggregateMixin`.**latest_start**

`TimeIntervalAggregateMixin`.**latest_stop**

`TimeIntervalAggregateMixin`.**signature**

Tuple of start bound and stop bound.

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin`.**start**

Starting offset of interval:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.start
Offset(2, 1)

```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.stop`

Stopping offset of interval:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.stop
Offset(10, 1)

```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

Methods

`TimeIntervalAggregateMixin.find_intervals_intersecting_or_tangent_to_interval()`
`TimeIntervalAggregateMixin.find_intervals_intersecting_or_tangent_to_offset()`
`TimeIntervalAggregateMixin.find_intervals_starting_after_offset()`
`TimeIntervalAggregateMixin.find_intervals_starting_and_stopping_within_interval()`
`TimeIntervalAggregateMixin.find_intervals_starting_at_offset()`
`TimeIntervalAggregateMixin.find_intervals_starting_before_offset()`
`TimeIntervalAggregateMixin.find_intervals_starting_or_stopping_at_offset()`
`TimeIntervalAggregateMixin.find_intervals_starting_within_interval()`
`TimeIntervalAggregateMixin.find_intervals_stopping_after_offset()`
`TimeIntervalAggregateMixin.find_intervals_stopping_at_offset()`
`TimeIntervalAggregateMixin.find_intervals_stopping_before_offset()`
`TimeIntervalAggregateMixin.find_intervals_stopping_within_interval()`
`TimeIntervalAggregateMixin.get_overlap_with_interval(interval)`
 Return amount of overlap with *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_contained_by_interval(interval)`
 True if interval is contained by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_container_of_interval(interval)`
True if interval contains *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_overlapped_by_interval(interval)`
True if interval is overlapped by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.is_tangent_to_interval(interval)`
True if interval is tangent to *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.quantize_to_rational(rational)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.scale_by_rational(rational)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.scale_to_rational(rational)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.shift_by_rational(rational)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.shift_to_rational(rational)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.split_at_rationals(*rationals)`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

Special Methods

TimeIntervalAggregateMixin.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

TimeIntervalAggregateMixin.__eq__(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalAggregateMixin.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalAggregateMixin.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

TimeIntervalAggregateMixin.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

TimeIntervalAggregateMixin.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalAggregateMixin.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalAggregateMixin.__ne__(arg)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalAggregateMixin.__nonzero__()`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalAggregateMixin.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

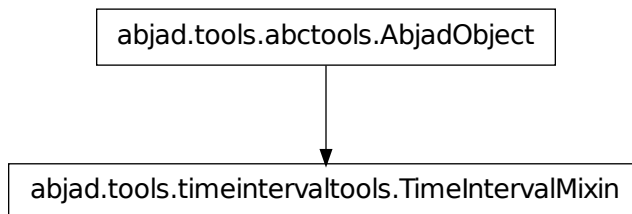
`TimeIntervalAggregateMixin.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TimeIntervalAggregateMixin.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

timeintervaltools.TimeIntervalMixin



```

class abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin(*args, **kwargs)
  
```

Read-only Properties

`TimeIntervalMixin.bounds`

Start and stop of self returned as `TimeInterval` instance:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> bounds = interval.bounds
abjad> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
  
```

```
abjad> bounds == interval
True
abjad> bounds is interval
False
```

Returns *TimeInterval* instance.

TimeIntervalMixin.center

Center offset of start and stop offsets:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

TimeIntervalMixin.duration

Duration of the time interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

TimeIntervalMixin.signature

Tuple of start bound and stop bound.

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

TimeIntervalMixin.start

Starting offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

TimeIntervalMixin.stop

Stopping offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Methods

TimeIntervalMixin.get_overlap_with_interval (*interval*)

Return amount of overlap with *interval*.

`TimeIntervalMixin.is_contained_by_interval(interval)`
 True if interval is contained by *interval*.

`TimeIntervalMixin.is_container_of_interval(interval)`
 True if interval contains *interval*.

`TimeIntervalMixin.is_overlapped_by_interval(interval)`
 True if interval is overlapped by *interval*.

`TimeIntervalMixin.is_tangent_to_interval(interval)`
 True if interval is tangent to *interval*.

`TimeIntervalMixin.quantize_to_rational(rational)`

`TimeIntervalMixin.scale_by_rational(rational)`

`TimeIntervalMixin.scale_to_rational(rational)`

`TimeIntervalMixin.shift_by_rational(rational)`

`TimeIntervalMixin.shift_to_rational(rational)`

`TimeIntervalMixin.split_at_rationals(*rationals)`

Special Methods

`TimeIntervalMixin.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TimeIntervalMixin.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalMixin.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

TimeIntervalMixin.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalMixin.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalMixin.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalMixin.__nonzero__()

TimeIntervalMixin.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

TimeIntervalMixin.__setattr__()

`x.__setattr__('name', value) <==> x.name = value`

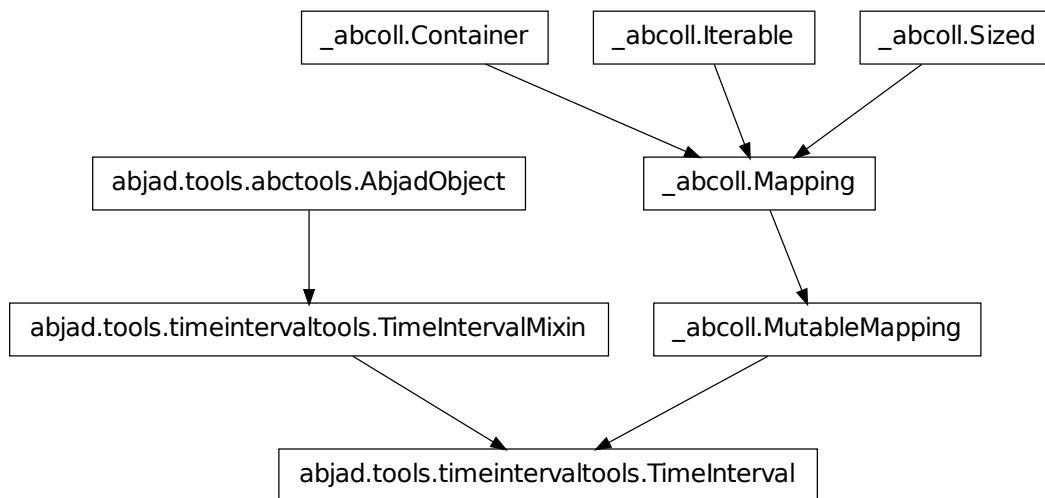
Note: Inherited from `__builtin__.object`

TimeIntervalMixin.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

timeintervaltools concrete classes

timeintervaltools.TimeInterval



class `abjad.tools.timeintervaltools.TimeInterval.TimeInterval(*args)`
 A start / stop pair, carrying some metadata.

Read-only Properties

`TimeInterval.bounds`

Start and stop of self returned as `TimeInterval` instance:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> bounds = interval.bounds
abjad> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
abjad> bounds == interval
True
abjad> bounds is interval
False
  
```

Returns *TimeInterval* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.center`

Center point of start and stop bounds.

`TimeInterval.duration`

stop bound minus start bound.

`TimeInterval.signature`

Tuple of start bound and stop bound.

`TimeInterval.start`
start bound.

`TimeInterval.stop`
stop bound.

Methods

`TimeInterval.clear()`

Note: Inherited from `_abcoll.MutableMapping`

`TimeInterval.get(key, default=None)`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.get_overlap_with_interval(interval)`
Return amount of overlap with *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_contained_by_interval(interval)`
True if interval is contained by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_container_of_interval(interval)`
True if interval contains *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_overlapped_by_interval(interval)`
True if interval is overlapped by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.is_tangent_to_interval(interval)`
True if interval is tangent to *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.items()`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.iteritems()`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.iterkeys()`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.intervalvalues()`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.keys()`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.pop(key, default=<object object at 0x4042f048>)`

Note: Inherited from `_abcoll.MutableMapping`

`TimeInterval.popitem()`

Note: Inherited from `_abcoll.MutableMapping`

`TimeInterval.quantize_to_rational(rational)`

`TimeInterval.scale_by_rational(rational)`

`TimeInterval.scale_to_rational(rational)`

`TimeInterval.setdefault(key, default=None)`

Note: Inherited from `_abcoll.MutableMapping`

`TimeInterval.shift_by_rational(rational)`

`TimeInterval.shift_to_rational(rational)`

`TimeInterval.split_at_rationals(*rationals)`

`TimeInterval.update(*args, **kws)`

Note: Inherited from `_abcoll.MutableMapping`

`TimeInterval.values()`

Note: Inherited from `_abcoll.Mapping`

Special Methods

`TimeInterval.__contains__(key)`

Note: Inherited from `_abcoll.Mapping`

`TimeInterval.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TimeInterval.__delitem__(item)`

`TimeInterval.__eq__(other)`

`TimeInterval.__ge__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeInterval.__getitem__(item)`

`TimeInterval.__gt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TimeInterval.__hash__()`

`TimeInterval.__iter__()`

`TimeInterval.__le__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeInterval.__len__()`

`TimeInterval.__lt__(arg)`
 Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeInterval.__ne__` (*other*)

`TimeInterval.__nonzero__` ()

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeInterval.__repr__` ()

`TimeInterval.__setattr__` ()

`x.__setattr__('name', value) <==> x.name = value`

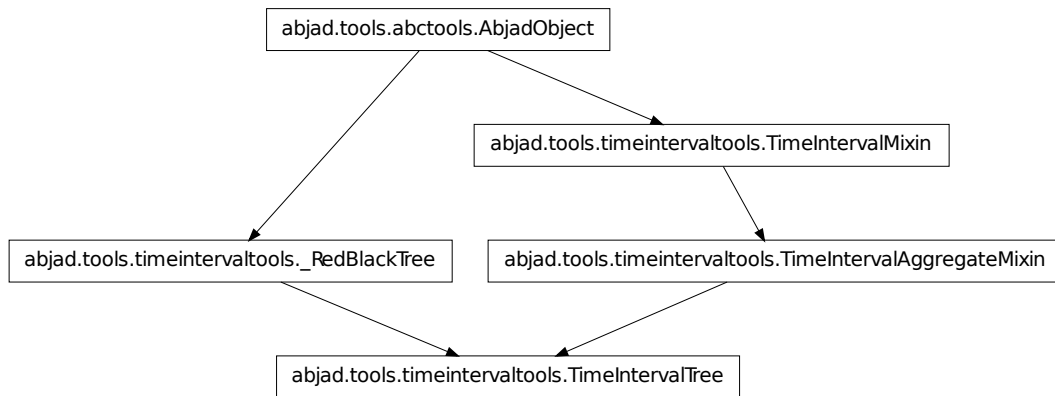
Note: Inherited from `__builtin__.object`

`TimeInterval.__setitem__` (*item*, *value*)

`TimeInterval.__str__` () <==> `str(x)`

Note: Inherited from `__builtin__.object`

`timeintervaltools.TimeIntervalTree`



class `abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree` (*intervals*)

An augmented red-black tree for storing and searching for intervals of time (rather than pitch).

This allows for the arbitrary placement of blocks of material along a time-line. While this functionality could be achieved with Python's built-in collections, this class reduces the complexity of the search process, such as locating overlapping intervals.

`TimeIntervalTrees` can be instantiated without contents, or from a mixed collection of other `TimeIntervalTrees` and / or `TimeIntervals`. The input will be parsed recursively:

```

abjad> from abjad.tools.timeintervaltools import TimeIntervalTree
abjad> from abjad.tools.timeintervaltools import TimeInterval

```

```
abjad> interval_one = TimeInterval(0, 10)
abjad> interval_two = TimeInterval(1, 8)
abjad> interval_three = TimeInterval(3, 13)
abjad> tree = TimeIntervalTree([interval_one, interval_two, interval_three])

abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(10, 1), {}),
    TimeInterval(Offset(1, 1), Offset(8, 1), {}),
    TimeInterval(Offset(3, 1), Offset(13, 1), {})
])
```

Return *TimeIntervalTree* instance.

Read-only Properties

`TimeIntervalTree.bounds`

Start and stop of self returned as *TimeInterval* instance:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> bounds = interval.bounds
abjad> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
abjad> bounds == interval
True
abjad> bounds is interval
False
```

Returns *TimeInterval* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.center`

Center offset of start and stop offsets:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.duration`

Absolute difference of the stop and start values of the tree:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> ti1 = TimeInterval(1, 2)
abjad> ti2 = TimeInterval(3, (7, 2))
abjad> tree = TimeIntervalTree([ti1, ti2])
abjad> tree.duration
Duration(5, 2)
```

Empty trees have a duration of 0.

Return *Duration* instance.

TimeIntervalTree.earliest_start

The minimum start value of all intervals in the tree:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> ti1 = TimeInterval(1, 2)
abjad> ti2 = TimeInterval(3, (7, 2))
abjad> tree = TimeIntervalTree([ti1, ti2])
abjad> tree.earliest_start
Offset(1, 1)
```

Return *Offset* instance, or *None* if tree is empty.

TimeIntervalTree.earliest_stop

The minimum stop value of all intervals in the tree:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> ti1 = TimeInterval(1, 2)
abjad> ti2 = TimeInterval(3, (7, 2))
abjad> tree = TimeIntervalTree([ti1, ti2])
abjad> tree.earliest_stop
Offset(2, 1)
```

Return *Offset* instance, or *None* if tree is empty.

TimeIntervalTree.intervals**TimeIntervalTree.latest_start**

The maximum start value of all intervals in the tree:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> ti1 = TimeInterval(1, 2)
abjad> ti2 = TimeInterval(3, (7, 2))
abjad> tree = TimeIntervalTree([ti1, ti2])
abjad> tree.latest_start
Offset(3, 1)
```

Return *Offset* instance, or *None* if tree is empty.

TimeIntervalTree.latest_stop

The maximum stop value of all intervals in the tree:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> ti1 = TimeInterval(1, 2)
abjad> ti2 = TimeInterval(3, (7, 2))
abjad> tree = TimeIntervalTree([ti1, ti2])
abjad> tree.latest_stop
Offset(7, 2)
```

Return *Offset* instance, or *None* if tree is empty.

TimeIntervalTree.signature

Tuple of start bound and stop bound.

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.start

Starting offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTree.stop

Stopping offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

Methods**TimeIntervalTree.find_intervals_intersecting_or_tangent_to_interval(*args)**

Find all intervals in tree intersecting or tangent to the interval defined in *args*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> interval = TimeInterval(0, 1)
abjad> found = tree.find_intervals_intersecting_or_tangent_to_interval(interval)
abjad> sorted([x['name'] for x in found])
['a', 'b', 'c']

abjad> interval = TimeInterval(3, 4)
abjad> found = tree.find_intervals_intersecting_or_tangent_to_interval(interval)
abjad> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_intersecting_or_tangent_to_offset(offset)

Find all intervals in tree intersecting or tangent to *offset*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])
```

```
abjad> offset = 1
abjad> found = tree.find_intervals_intersecting_or_tangent_to_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'b', 'c']

abjad> offset = 3
abjad> found = tree.find_intervals_intersecting_or_tangent_to_offset(offset)
abjad> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_after_offset (*offset*)

Find all intervals in tree starting after *offset*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 0
abjad> found = tree.find_intervals_starting_after_offset(offset)
abjad> sorted([x['name'] for x in found])
['b', 'd']

abjad> offset = 1
abjad> found = tree.find_intervals_starting_after_offset(offset)
abjad> sorted([x['name'] for x in found])
['d']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_and_stopping_within_interval (**args*)

Find all intervals in tree starting and stopping within the interval defined by *args*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> interval = TimeInterval(1, 3)
abjad> found = tree.find_intervals_starting_and_stopping_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['b', 'd']

abjad> interval = TimeInterval(-1, 2)
abjad> found = tree.find_intervals_starting_and_stopping_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['a', 'b']
```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_at_offset (*offset*)

Find all intervals in tree starting at *offset*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
```

```

abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 0
abjad> found = tree.find_intervals_starting_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'c']

abjad> offset = 1
abjad> found = tree.find_intervals_starting_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['b']

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_before_offset (*offset*)

Find all intervals in tree starting before *offset*:

```

abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 1
abjad> found = tree.find_intervals_starting_before_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'c']

abjad> offset = 2
abjad> found = tree.find_intervals_starting_before_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'b', 'c']

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_or_stopping_at_offset (*offset*)

Find all intervals in tree starting or stopping at *offset*:

```

abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 2
abjad> found = tree.find_intervals_starting_or_stopping_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['b', 'd']

abjad> offset = 1
abjad> found = tree.find_intervals_starting_or_stopping_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'b']

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_starting_within_interval (**args*)

Find all intervals in tree starting within the interval defined by *args*:

```

abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> interval = TimeInterval((-1, 2), (1, 2))
abjad> found = tree.find_intervals_starting_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['a', 'c']

abjad> interval = TimeInterval((1, 2), (5, 2))
abjad> found = tree.find_intervals_starting_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['b', 'd']

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_stopping_after_offset (*offset*)

Find all intervals in tree stopping after *offset*:

```

abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 1
abjad> found = tree.find_intervals_stopping_after_offset(offset)
abjad> sorted([x['name'] for x in found])
['b', 'c', 'd']

abjad> offset = 2
abjad> found = tree.find_intervals_stopping_after_offset(offset)
abjad> sorted([x['name'] for x in found])
['c', 'd']

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.find_intervals_stopping_at_offset (*offset*)

Find all intervals in tree stopping at *offset*:

```

abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 3
abjad> found = tree.find_intervals_stopping_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['c', 'd']

abjad> offset = 1
abjad> found = tree.find_intervals_stopping_at_offset(offset)
abjad> sorted([x['name'] for x in found])
['a']

```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_stopping_before_offset` (*offset*)

Find all intervals in tree stopping before *offset*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> offset = 3
abjad> found = tree.find_intervals_stopping_before_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'b']

abjad> offset = (7, 2)
abjad> found = tree.find_intervals_stopping_before_offset(offset)
abjad> sorted([x['name'] for x in found])
['a', 'b', 'c', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.find_intervals_stopping_within_interval` (**args*)

Find all intervals in tree stopping within the interval defined by *args*:

```
abjad> a = TimeInterval(0, 1, {'name': 'a'})
abjad> b = TimeInterval(1, 2, {'name': 'b'})
abjad> c = TimeInterval(0, 3, {'name': 'c'})
abjad> d = TimeInterval(2, 3, {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])

abjad> interval = TimeInterval((3, 2), (5, 2))
abjad> found = tree.find_intervals_stopping_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['b']

abjad> interval = TimeInterval((5, 2), (7, 2))
abjad> found = tree.find_intervals_stopping_within_interval(interval)
abjad> sorted([x['name'] for x in found])
['c', 'd']
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.get_overlap_with_interval` (*interval*)

Return amount of overlap with *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.is_contained_by_interval` (*interval*)

True if interval is contained by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.is_container_of_interval` (*interval*)

True if interval contains *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.is_overlapped_by_interval(interval)`
 True if interval is overlapped by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.is_tangent_to_interval(interval)`
 True if interval is tangent to *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTree.quantize_to_rational(rational)`
 Quantize all intervals in tree to a multiple (1 or more) of *rational*:

```
abjad> a = TimeInterval((1, 16), (1, 8), {'name': 'a'})
abjad> b = TimeInterval((2, 7), (13, 7), {'name': 'b'})
abjad> c = TimeInterval((3, 5), (8, 5), {'name': 'c'})
abjad> d = TimeInterval((2, 3), (5, 3), {'name': 'd'})
abjad> tree = TimeIntervalTree([a, b, c, d])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(1, 16), Offset(1, 8), {'name': 'a'}),
    TimeInterval(Offset(2, 7), Offset(13, 7), {'name': 'b'}),
    TimeInterval(Offset(3, 5), Offset(8, 5), {'name': 'c'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'})
])

abjad> rational = (1, 4)
abjad> tree.quantize_to_rational(rational)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'a'}),
    TimeInterval(Offset(1, 4), Offset(7, 4), {'name': 'b'}),
    TimeInterval(Offset(1, 2), Offset(3, 2), {'name': 'c'}),
    TimeInterval(Offset(3, 4), Offset(7, 4), {'name': 'd'})
])

abjad> rational = (1, 3)
abjad> tree.quantize_to_rational(rational)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 3), {'name': 'a'}),
    TimeInterval(Offset(1, 3), Offset(2, 1), {'name': 'b'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'c'}),
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'})
])
```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.scale_by_rational(rational)`
 Scale aggregate duration of tree by *rational*:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> one = TimeInterval(0, 1, {'name': 'one'})
abjad> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
abjad> three = TimeInterval(2, 4, {'name': 'three'})
abjad> tree = TimeIntervalTree([one, two, three])
abjad> tree
```

```

TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> result = tree.scale_by_rational((2, 3))
abjad> result
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(2, 3), {'name': 'one'}),
    TimeInterval(Offset(1, 3), Offset(5, 3), {'name': 'two'}),
    TimeInterval(Offset(4, 3), Offset(8, 3), {'name': 'three'})
])

```

Scaling works regardless of the starting offset of the *TimeIntervalTree*:

```

abjad> zero = TimeInterval(-4, 0, {'name': 'zero'})
abjad> tree = TimeIntervalTree([zero, one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> result = tree.scale_by_rational(2)
abjad> result
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(4, 1), {'name': 'zero'}),
    TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'one'}),
    TimeInterval(Offset(5, 1), Offset(9, 1), {'name': 'two'}),
    TimeInterval(Offset(8, 1), Offset(12, 1), {'name': 'three'})
])

abjad> result.start == tree.start
True
abjad> result.duration == tree.duration * 2
True

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.scale_to_rational (*rational*)

Scale aggregate duration of tree to *rational*:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> one = TimeInterval(0, 1, {'name': 'one'})
abjad> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
abjad> three = TimeInterval(2, 4, {'name': 'three'})
abjad> tree = TimeIntervalTree([one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

```

```

abjad> result = tree.scale_to_rational(1)
abjad> result
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'one'}),
    TimeInterval(Offset(1, 8), Offset(5, 8), {'name': 'two'}),
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'three'})
])

abjad> result.scale_to_rational(10)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(5, 2), {'name': 'one'}),
    TimeInterval(Offset(5, 4), Offset(25, 4), {'name': 'two'}),
    TimeInterval(Offset(5, 1), Offset(10, 1), {'name': 'three'})
])

```

Scaling works regardless of the starting offset of the *TimeIntervalTree*:

```

abjad> zero = TimeInterval(-4, 0, {'name': 'zero'})
abjad> tree = TimeIntervalTree([zero, one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> tree.scale_to_rational(4)
TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(-2, 1), {'name': 'zero'}),
    TimeInterval(Offset(-2, 1), Offset(-3, 2), {'name': 'one'}),
    TimeInterval(Offset(-7, 4), Offset(-3, 4), {'name': 'two'}),
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

`TimeIntervalTree.shift_by_rational(rational)`

Shift aggregate offset of tree by *rational*:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> one = TimeInterval(0, 1, {'name': 'one'})
abjad> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
abjad> three = TimeInterval(2, 4, {'name': 'three'})
abjad> tree = TimeIntervalTree([one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> result = tree.shift_by_rational(-2.5)
abjad> result
TimeIntervalTree([
    TimeInterval(Offset(-5, 2), Offset(-3, 2), {'name': 'one'}),
    TimeInterval(Offset(-2, 1), Offset(0, 1), {'name': 'two'}),

```

```

    TimeInterval(Offset(-1, 2), Offset(3, 2), {'name': 'three'})
])
abjad> result.shift_by_rational(6)
TimeIntervalTree([
    TimeInterval(Offset(7, 2), Offset(9, 2), {'name': 'one'}),
    TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'two'}),
    TimeInterval(Offset(11, 2), Offset(15, 2), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.**shift_to_rational**(*rational*)

Shift aggregate offset of tree to *rational*:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> one = TimeInterval(0, 1, {'name': 'one'})
abjad> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
abjad> three = TimeInterval(2, 4, {'name': 'three'})
abjad> tree = TimeIntervalTree([one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> result = tree.shift_to_rational(100)
abjad> result
TimeIntervalTree([
    TimeInterval(Offset(100, 1), Offset(101, 1), {'name': 'one'}),
    TimeInterval(Offset(201, 2), Offset(205, 2), {'name': 'two'}),
    TimeInterval(Offset(102, 1), Offset(104, 1), {'name': 'three'})
])

```

Return *TimeIntervalTree* instance.

TimeIntervalTree.**split_at_rationals**(**rationals*)

Split tree at each rational in *rationals*:

```

abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> one = TimeInterval(0, 1, {'name': 'one'})
abjad> two = TimeInterval((1, 2), (5, 2), {'name': 'two'})
abjad> three = TimeInterval(2, 4, {'name': 'three'})
abjad> tree = TimeIntervalTree([one, two, three])
abjad> tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'})
])

abjad> result = tree.split_at_rationals(1, 2, 3)
abjad> len(result)
4

```

```

abjad> result[0]
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'two'})
])

abjad> result[1]
TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'two'})
])

abjad> result[2]
TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(5, 2), {'name': 'two'}),
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'three'})
])

abjad> result[3]
TimeIntervalTree([
    TimeInterval(Offset(3, 1), Offset(4, 1), {'name': 'three'})
])

```

Return tuple of *TimeIntervalTree* instances.

Special Methods

`TimeIntervalTree.__contains__(item)`

`TimeIntervalTree.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TimeIntervalTree.__eq__(other)`

`TimeIntervalTree.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__getitem__(item)`

`TimeIntervalTree.__getslice__(start, end)`

`TimeIntervalTree.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TimeIntervalTree.__iter__()`

`TimeIntervalTree.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__len__()`

`TimeIntervalTree.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeIntervalTree.__ne__(other)`

`TimeIntervalTree.__nonzero__()`

TimeIntervalTree evaluates to True if it contains any intervals:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> true_tree = TimeIntervalTree([TimeInterval(0, 1)])
abjad> false_tree = TimeIntervalTree([])
```

```
abjad> bool(true_tree)
True
abjad> bool(false_tree)
False
```

Return boolean.

`TimeIntervalTree.__repr__()`

`TimeIntervalTree.__setattr__()`

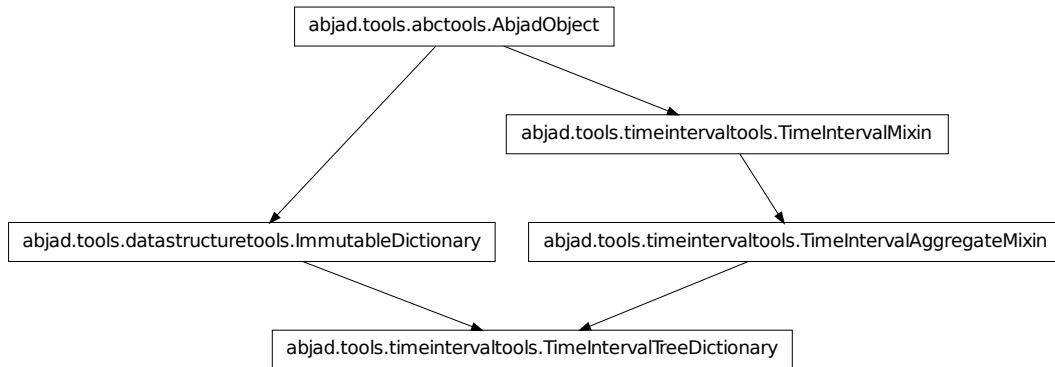
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TimeIntervalTree.__str__()`

Note: Inherited from `timeintervaltools._RedBlackTree`

timeintervaltools.TimeIntervalTreeDictionary



```
class abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.Tim
    A dictionary of TimeIntervalTrees:
```

```

abjad> from abjad.tools.timeintervaltools import TimeIntervalTreeDictionary

abjad> from abjad.tools.timeintervaltools import TimeIntervalTree
abjad> from abjad.tools.timeintervaltools import TimeInterval

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

```

TimeIntervalTreeDictionary can be instantiated from one or more other *TimeIntervalTreeDictionary* instances, whose trees will be fused if they share keys. It can also be instantiated from a regular dictionary whose values are *TimeIntervalTree* instances, or from a list of pairs where the second value of each pair is a *TimeIntervalTree* instance.

TimeIntervalTreeDictionary supports the same set of methods and properties as *TimeIntervalTree* and *TimeInterval*, including searching for intervals, quantizing, scaling, shifting and splitting.

TimeIntervalTreeDictionary is immutable.

Return *TimeIntervalTreeDictionary* instance.

Read-only Properties

TimeIntervalTreeDictionary.**bounds**

Start and stop of self returned as *TimeInterval* instance:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> bounds = interval.bounds
abjad> bounds
TimeInterval(Offset(2, 1), Offset(10, 1), {})
abjad> bounds == interval
True
abjad> bounds is interval
False
```

Returns *TimeInterval* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTreeDictionary.**center**

Center offset of start and stop offsets:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.center
Offset(6, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

TimeIntervalTreeDictionary.**composite_tree**

The *TimeIntervalTree* composed of all the intervals in all trees in self:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

abjad> treedict.composite_tree
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'})
])
```

Return *TimeIntervalTree* instance.

TimeIntervalTreeDictionary.**duration**

Duration of the time interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.duration
Duration(8, 1)
```

Returns *Duration* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.earliest_start`

The earliest start offset of all intervals in all trees in self:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

abjad> treedict.earliest_start
Offset(0, 1)
```

Return *Offset* instance.

`TimeIntervalTreeDictionary.earliest_stop`

The earliest stop offset of all intervals in all trees in self:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

abjad> treedict.earliest_stop
Offset(1, 1)
```

Return *Offset* instance.

`TimeIntervalTreeDictionary.intervals`

`TimeIntervalTreeDictionary.latest_start`

The latest start offset of all intervals in all trees in self:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

abjad> treedict.latest_start
Offset(2, 1)
```

Return *Offset* instance.

`TimeIntervalTreeDictionary.latest_stop`

The latest stop offset of all intervals in all trees in self:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
```

```
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})

abjad> treedict.latest_stop
Offset(3, 1)
```

Return *Offset* instance.

`TimeIntervalTreeDictionary.signature`

Tuple of start bound and stop bound.

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.signature
(Offset(2, 1), Offset(10, 1))
```

Returns 2-tuple of *Offset* instances.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.start`

Starting offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.start
Offset(2, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.stop`

Stopping offset of interval:

```
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> interval = TimeInterval(2, 10)
abjad> interval.stop
Offset(10, 1)
```

Returns *Offset* instance.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

Methods

`TimeIntervalTreeDictionary.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.find_intervals_intersecting_or_tangent_to_interval(*args)`
Find all intervals in dictionary intersecting or tangent to the interval defined in *args*:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> interval = TimeInterval(0, 1)
abjad> treedict.find_intervals_intersecting_or_tangent_to_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

abjad> interval = TimeInterval(3, 4)
abjad> treedict.find_intervals_intersecting_or_tangent_to_interval(interval)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_intersecting_or_tangent_to_offset(offset)`
Find all intervals in dictionary intersecting or tangent to *offset*:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
```

```

abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 1
abjad> treedict.find_intervals_intersecting_or_tangent_to_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

abjad> offset = 3
abjad> treedict.find_intervals_intersecting_or_tangent_to_offset(offset)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_starting_after_offset**(*offset*)

Find all intervals in dictionary starting after *offset*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

```
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 0
abjad> treedict.find_intervals_starting_after_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 1
abjad> treedict.find_intervals_starting_after_offset(offset)
TimeIntervalTreeDictionary({
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.find_intervals_starting_and_stopping_within_interval(*args)
Find all intervals in dictionary starting and stopping within the interval defined by *args*:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> interval = TimeInterval(1, 3)
abjad> treedict.find_intervals_starting_and_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})
```

```

    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> interval = TimeInterval(-1, 2)
abjad> treedict.find_intervals_starting_and_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_starting_at_offset` (*offset*)

Find all intervals in dictionary starting at *offset*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 0
abjad> treedict.find_intervals_starting_at_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

abjad> offset = 1
abjad> treedict.find_intervals_starting_at_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})

```

```
    ]),  
  })
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_starting_before_offset` (*offset*)

Find all intervals in dictionary starting before *offset*:

```
abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])  
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])  
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])  
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])  
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})  
abjad> treedict  
TimeIntervalTreeDictionary({  
    'a': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),  
    ]),  
    'b': TimeIntervalTree([  
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),  
    ]),  
    'c': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),  
    ]),  
    'd': TimeIntervalTree([  
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),  
    ]),  
})  
  
abjad> offset = 1  
abjad> treedict.find_intervals_starting_before_offset(offset)  
TimeIntervalTreeDictionary({  
    'a': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),  
    ]),  
    'c': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),  
    ]),  
})  
  
abjad> offset = 2  
abjad> treedict.find_intervals_starting_before_offset(offset)  
TimeIntervalTreeDictionary({  
    'a': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),  
    ]),  
    'b': TimeIntervalTree([  
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),  
    ]),  
    'c': TimeIntervalTree([  
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),  
    ]),  
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_starting_or_stopping_at_offset` (*offset*)

Find all intervals in dictionary starting or stopping at *offset*:


```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 2
abjad> treedict.find_intervals_starting_or_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 1
abjad> treedict.find_intervals_starting_or_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.find_intervals_starting_within_interval(*args)
Find all intervals in dictionary starting within the interval defined by *args*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([

```

```

        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ],
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> interval = TimeInterval((-1, 2), (1, 2))
abjad> treedict.find_intervals_starting_within_interval(interval)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
})

abjad> interval = TimeInterval((1, 2), (5, 2))
abjad> treedict.find_intervals_starting_within_interval(interval)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_stopping_after_offset` (*offset*)

Find all intervals in dictionary stopping after *offset*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

```

abjad> offset = 1
abjad> treedict.find_intervals_stopping_after_offset(offset)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 2
abjad> treedict.find_intervals_stopping_after_offset(offset)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.find_intervals_stopping_at_offset(offset)`

Find all intervals in dictionary stopping at *offset*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> offset = 3
abjad> treedict.find_intervals_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

```

    ]),
  })

abjad> offset = 1
abjad> treedict.find_intervals_stopping_at_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_stopping_before_offset**(*offset*)

Find all intervals in dictionary stopping before *offset*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
  ]),
})

abjad> offset = 3
abjad> treedict.find_intervals_stopping_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
})

```

```

abjad> offset = (7, 2)
abjad> treedict.find_intervals_stopping_before_offset(offset)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
  ]),
})

```

```

    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**find_intervals_stopping_within_interval**(*args)

Find all intervals in dictionary stopping within the interval defined by *args*:

```

abjad> a = TimeIntervalTree([TimeInterval(0, 1, {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval(1, 2, {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval(0, 3, {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval(2, 3, {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

abjad> interval = TimeInterval((3, 2), (5, 2))
abjad> treedict.find_intervals_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'b'}),
    ]),
})

abjad> interval = TimeInterval((5, 2), (7, 2))
abjad> treedict.find_intervals_stopping_within_interval(interval)
TimeIntervalTreeDictionary({
    'c': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(3, 1), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**get**(*k*[, *d*]) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to None.

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.get_overlap_with_interval(interval)`
Return amount of overlap with *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.is_contained_by_interval(interval)`
True if interval is contained by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.is_container_of_interval(interval)`
True if interval contains *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.is_overlapped_by_interval(interval)`
True if interval is overlapped by *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.is_tangent_to_interval(interval)`
True if interval is tangent to *interval*.

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.quantize_to_rational(rational)`

Quantize all intervals in dictionary to a multiple (1 or more) of *rational*:

```
abjad> a = TimeIntervalTree([TimeInterval((1, 16), (1, 8), {'name': 'a'})])
abjad> b = TimeIntervalTree([TimeInterval((2, 7), (13, 7), {'name': 'b'})])
abjad> c = TimeIntervalTree([TimeInterval((3, 5), (8, 5), {'name': 'c'})])
abjad> d = TimeIntervalTree([TimeInterval((2, 3), (5, 3), {'name': 'd'})])
abjad> treedict = TimeIntervalTreeDictionary({'a': a, 'b': b, 'c': c, 'd': d})
abjad> treedict
```

```
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(1, 16), Offset(1, 8), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(2, 7), Offset(13, 7), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(3, 5), Offset(8, 5), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
    TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'}),
  ]),
})
```

```
abjad> rational = (1, 4)
abjad> treedict.quantize_to_rational(rational)
TimeIntervalTreeDictionary({
  'a': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'a'}),
  ]),
  'b': TimeIntervalTree([
    TimeInterval(Offset(1, 4), Offset(7, 4), {'name': 'b'}),
  ]),
  'c': TimeIntervalTree([
    TimeInterval(Offset(1, 2), Offset(3, 2), {'name': 'c'}),
  ]),
  'd': TimeIntervalTree([
```

```

        TimeInterval(Offset(3, 4), Offset(7, 4), {'name': 'd'}),
    ]),
})

abjad> rational = (1, 3)
abjad> treedict.quantize_to_rational(rational)
TimeIntervalTreeDictionary({
    'a': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 3), {'name': 'a'}),
    ]),
    'b': TimeIntervalTree([
        TimeInterval(Offset(1, 3), Offset(2, 1), {'name': 'b'}),
    ]),
    'c': TimeIntervalTree([
        TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'c'}),
    ]),
    'd': TimeIntervalTree([
        TimeInterval(Offset(2, 3), Offset(5, 3), {'name': 'd'}),
    ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**scale_by_rational**(*rational*)

Scale aggregate duration of dictionary by *rational*:

```

abjad> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
abjad> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
abjad> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
abjad> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

abjad> result = treedict.scale_by_rational((2, 3))
abjad> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(2, 3), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(4, 3), Offset(8, 3), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 3), Offset(5, 3), {'name': 'two'}),
    ]),
})

```

Scaling works regardless of the starting offset of the *TimeIntervalTreeDictionary*:


```

abjad> zero = TimeIntervalTree([TimeInterval(-4, 0, {'name': 'zero'})])
abjad> treedict = TimeIntervalTreeDictionary({'zero': zero, 'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
    'zero': TimeIntervalTree([
        TimeInterval(Offset(-4, 1), Offset(0, 1), {'name': 'zero'}),
    ]),
})

abjad> result = treedict.scale_by_rational(2)
abjad> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(8, 1), Offset(12, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(5, 1), Offset(9, 1), {'name': 'two'}),
    ]),
    'zero': TimeIntervalTree([
        TimeInterval(Offset(-4, 1), Offset(4, 1), {'name': 'zero'}),
    ]),
})

abjad> result.start == treedict.start
True
abjad> result.duration == treedict.duration * 2
True

```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**scale_to_rational**(*rational*)

Scale aggregate duration of dictionary to *rational*:

```

abjad> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
abjad> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
abjad> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
abjad> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

```

```

    ]),
  })

abjad> result = treedict.scale_to_rational(1)
abjad> result
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 4), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(1, 8), Offset(5, 8), {'name': 'two'}),
  ]),
})

abjad> result.scale_to_rational(10)
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(5, 2), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(5, 1), Offset(10, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(5, 4), Offset(25, 4), {'name': 'two'}),
  ]),
})

```

Scaling works regardless of the starting offset of the *TimeIntervalTreeDictionary*:

```

abjad> zero = TimeIntervalTree([TimeInterval(-4, 0, {'name': 'zero'})])
abjad> treedict = TimeIntervalTreeDictionary({'zero': zero, 'one': one, 'two': two, 'three': thr

abjad> treedict.scale_to_rational(4)
TimeIntervalTreeDictionary({
  'one': TimeIntervalTree([
    TimeInterval(Offset(-2, 1), Offset(-3, 2), {'name': 'one'}),
  ]),
  'three': TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'name': 'three'}),
  ]),
  'two': TimeIntervalTree([
    TimeInterval(Offset(-7, 4), Offset(-3, 4), {'name': 'two'}),
  ]),
  'zero': TimeIntervalTree([
    TimeInterval(Offset(-4, 1), Offset(-2, 1), {'name': 'zero'}),
  ]),
})

```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.setdefault(k[, d])` → `D.get(k,d)`, also set `D[k]=d` if `k` not in `D`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.shift_by_rational(rational)`

Shift aggregate offset of dictionary by *rational*:

```
abjad> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
abjad> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
abjad> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
abjad> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

abjad> result = treedict.shift_by_rational(-2.5)
abjad> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(-5, 2), Offset(-3, 2), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(-1, 2), Offset(3, 2), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(-2, 1), Offset(0, 1), {'name': 'two'}),
    ]),
})

abjad> result.shift_by_rational(6)
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(7, 2), Offset(9, 2), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(11, 2), Offset(15, 2), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(4, 1), Offset(6, 1), {'name': 'two'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

`TimeIntervalTreeDictionary.shift_to_rational(rational)`

Shift aggregate offset of dictionary to *rational*:

```
abjad> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
abjad> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
abjad> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
abjad> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
```

```
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

abjad> result = treedict.shift_to_rational(100)
abjad> result
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(100, 1), Offset(101, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(102, 1), Offset(104, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(201, 2), Offset(205, 2), {'name': 'two'}),
    ]),
})
```

Return *TimeIntervalTreeDictionary* instance.

TimeIntervalTreeDictionary.**split_at_rationals**(*rationals)

Split dictionary at each rational in *rationals*:

```
abjad> one = TimeIntervalTree([TimeInterval(0, 1, {'name': 'one'})])
abjad> two = TimeIntervalTree([TimeInterval((1, 2), (5, 2), {'name': 'two'})])
abjad> three = TimeIntervalTree([TimeInterval(2, 4, {'name': 'three'})])
abjad> treedict = TimeIntervalTreeDictionary({'one': one, 'two': two, 'three': three})
abjad> treedict
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(4, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(5, 2), {'name': 'two'}),
    ]),
})

abjad> result = treedict.split_at_rationals(1, 2, 3)
abjad> len(result)
4

abjad> result[0]
TimeIntervalTreeDictionary({
    'one': TimeIntervalTree([
        TimeInterval(Offset(0, 1), Offset(1, 1), {'name': 'one'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 2), Offset(1, 1), {'name': 'two'}),
    ]),
})
```

```

abjad> result[1]
TimeIntervalTreeDictionary({
    'two': TimeIntervalTree([
        TimeInterval(Offset(1, 1), Offset(2, 1), {'name': 'two'}),
    ]),
})

abjad> result[2]
TimeIntervalTreeDictionary({
    'three': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(3, 1), {'name': 'three'}),
    ]),
    'two': TimeIntervalTree([
        TimeInterval(Offset(2, 1), Offset(5, 2), {'name': 'two'}),
    ]),
})

abjad> result[3]
TimeIntervalTreeDictionary({
    'three': TimeIntervalTree([
        TimeInterval(Offset(3, 1), Offset(4, 1), {'name': 'three'}),
    ]),
})

```

Return tuple of *TimeIntervalTreeDictionary* instances.

TimeIntervalTreeDictionary.**update** (*E*, ***F*) → None. Update D from dict/iterable E and F. If E has a .keys() method, does: for k in E: D[k] = E[k] If E lacks .keys() method, does: for (k, v) in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**values** () → list of D's values

Note: Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewitems** () → a set-like object providing a view on D's items

Note: Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewkeys** () → a set-like object providing a view on D's keys

Note: Inherited from *__builtin__.dict*

TimeIntervalTreeDictionary.**viewvalues** () → an object providing a view on D's values

Note: Inherited from *__builtin__.dict*

Special Methods

`TimeIntervalTreeDictionary.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__contains__(k) → True if D has a key k, else False`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TimeIntervalTreeDictionary.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`TimeIntervalTreeDictionary.__eq__()`

`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__ge__()`

`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__getitem__()`

`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__gt__()`

`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__le__()`

`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__len__()` $\Leftrightarrow \text{len}(x)$

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__lt__()`
`x.__lt__(y)` $\Leftrightarrow x < y$

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__ne__()`
`x.__ne__(y)` $\Leftrightarrow x \neq y$

Note: Inherited from `__builtin__.dict`

`TimeIntervalTreeDictionary.__nonzero__()`

Note: Inherited from `timeintervaltools.TimeIntervalMixin`

`TimeIntervalTreeDictionary.__repr__()`
`TimeIntervalTreeDictionary.__setattr__()`
`x.__setattr__('name', value)` $\Leftrightarrow x.name = value$

Note: Inherited from `__builtin__.object`

`TimeIntervalTreeDictionary.__setitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`TimeIntervalTreeDictionary.__str__()` $\Leftrightarrow \text{str}(x)$

Note: Inherited from `__builtin__.object`

timeintervaltools functions

timeintervaltools.all_are_intervals_or_trees_or_empty

`abjad.tools.timeintervaltools.all_are_intervals_or_trees_or_empty.all_are_intervals_or_trees_or_empty`
 Recursively test if all elements of *input* are TimeIntervals or TimeIntervalTrees. An empty result also return as True.

`timeintervaltools.all_intervals_are_contiguous`

`abjad.tools.timeintervaltools.all_intervals_are_contiguous.all_intervals_are_contiguous` (*interval*)
True when all intervals in *intervals* are contiguous and non-overlapping.

`timeintervaltools.all_intervals_are_nonoverlapping`

`abjad.tools.timeintervaltools.all_intervals_are_nonoverlapping.all_intervals_are_nonoverlapping` (*intervals*)
True when all intervals in *intervals* in tree are non-overlapping.

`timeintervaltools.calculate_density_of_attacks_in_interval`

`abjad.tools.timeintervaltools.calculate_density_of_attacks_in_interval.calculate_density_of_attacks_in_interval` (*interval*)

Return a Fraction of number of attacks in *interval* over the duration of *interval*.

`timeintervaltools.calculate_density_of_releases_in_interval`

`abjad.tools.timeintervaltools.calculate_density_of_releases_in_interval.calculate_density_of_releases_in_interval` (*interval*)

Return a Fraction of the number of releases in *interval* divided by the duration of *interval*.

`timeintervaltools.calculate_depth_centroid_of_intervals`

`abjad.tools.timeintervaltools.calculate_depth_centroid_of_intervals.calculate_depth_centroid_of_intervals` (*intervals*)
Return a weighted mean, such that the centroids of each interval in the depth tree of *intervals* are the values, and the depth of each interval in the depth tree of *intervals* are the weights.

`timeintervaltools.calculate_depth_centroid_of_intervals_in_interval`

`abjad.tools.timeintervaltools.calculate_depth_centroid_of_intervals_in_interval.calculate_depth_centroid_of_intervals_in_interval` (*interval*)

Return the weighted mean of the depth tree of *intervals* in *interval*, such that the centroids of each interval of the depth tree are the values, and the weights are the depths at each interval of the depth tree.

`timeintervaltools.calculate_depth_density_of_intervals`

`abjad.tools.timeintervaltools.calculate_depth_density_of_intervals.calculate_depth_density_of_intervals` (*intervals*)
Return a Fraction, of the duration of each interval in the depth tree of *intervals*, multiplied by the depth at that interval, divided by the overall duration of *intervals*.

The depth density of a single interval is 1


```

abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(0, 1)
abjad> b = TimeInterval(0, 1)
abjad> c = TimeInterval(Fraction(1, 2), 1)
abjad> timeintervaltools.calculate_depth_density_of_intervals(a)
Duration(1, 1)
abjad> timeintervaltools.calculate_depth_density_of_intervals([a, b])
Duration(2, 1)
abjad> timeintervaltools.calculate_depth_density_of_intervals([a, c])
Duration(3, 2)
abjad> timeintervaltools.calculate_depth_density_of_intervals([a, b, c])
Duration(5, 2)

```

Return fraction.

`timeintervaltools.calculate_depth_density_of_intervals_in_interval`

`abjad.tools.timeintervaltools.calculate_depth_density_of_intervals_in_interval.calculate_depth_density_of_intervals_in_interval`

Return a Fraction, of the duration of each interval in the depth tree of *intervals* within *interval*, multiplied by the depth at that interval, divided by the overall duration of *intervals*.

`timeintervaltools.calculate_mean_attack_of_intervals`

`abjad.tools.timeintervaltools.calculate_mean_attack_of_intervals.calculate_mean_attack_of_intervals`
Return Fraction of the average attack offset of *intervals*

`timeintervaltools.calculate_mean_release_of_intervals`

`abjad.tools.timeintervaltools.calculate_mean_release_of_intervals.calculate_mean_release_of_intervals`
Return a Fraction of the average release offset of *intervals*.

`timeintervaltools.calculate_min_mean_and_max_depth_of_intervals`

`abjad.tools.timeintervaltools.calculate_min_mean_and_max_depth_of_intervals.calculate_min_mean_and_max_depth_of_intervals`
Return a 3-tuple of the minimum, mean and maximum depth of *intervals*. If *intervals* is empty, return None.
“Mean” in this case is a weighted mean, where the durations of the intervals in depth tree of *intervals* are the weights

`timeintervaltools.calculate_min_mean_and_max_durations_of_intervals`

`abjad.tools.timeintervaltools.calculate_min_mean_and_max_durations_of_intervals.calculate_min_mean_and_max_durations_of_intervals`
Return a 3-tuple of the minimum, mean and maximum duration of all intervals in *intervals*. If *intervals* is empty, return None.

`timeintervaltools.calculate_sustain_centroid_of_intervals`

`abjad.tools.timeintervaltools.calculate_sustain_centroid_of_intervals.calculate_sustain_centroid_of_intervals(intervals)`
 Return a weighted mean, such that the centroid of each interval in *intervals* are the values, and the weights are their durations.

`timeintervaltools.clip_interval_durations_to_range`

`abjad.tools.timeintervaltools.clip_interval_durations_to_range.clip_interval_durations_to_range(intervals, start, end)`

`timeintervaltools.compute_depth_of_intervals`

`abjad.tools.timeintervaltools.compute_depth_of_intervals.compute_depth_of_intervals(intervals)`
 Compute a tree whose intervals represent the depth (level of overlap) in each boundary pair of *intervals*:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> a = TimeInterval(0, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 15)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> compute_depth_of_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(3, 1), {'depth': 1}),
    TimeInterval(Offset(3, 1), Offset(6, 1), {'depth': 0}),
    TimeInterval(Offset(6, 1), Offset(9, 1), {'depth': 1}),
    TimeInterval(Offset(9, 1), Offset(12, 1), {'depth': 2}),
    TimeInterval(Offset(12, 1), Offset(15, 1), {'depth': 1})
])
```

Return interval tree.

`timeintervaltools.compute_depth_of_intervals_in_interval`

`abjad.tools.timeintervaltools.compute_depth_of_intervals_in_interval.compute_depth_of_intervals_in_interval(intervals, interval)`

Compute a tree whose intervals represent the depth (level of overlap) in each boundary pair of *intervals*, cropped within *interval*:

```
abjad> from abjad.tools.timeintervaltools import *
abjad> a = TimeInterval(0, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 15)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> d = TimeInterval(-1, 16)
abjad> compute_depth_of_intervals_in_interval(tree, d)
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(0, 1), {'depth': 0}),
    TimeInterval(Offset(0, 1), Offset(3, 1), {'depth': 1}),
    TimeInterval(Offset(3, 1), Offset(6, 1), {'depth': 0}),
    TimeInterval(Offset(6, 1), Offset(9, 1), {'depth': 1}),
])
```

```
    TimeInterval(Offset(9, 1), Offset(12, 1), {'depth': 2}),
    TimeInterval(Offset(12, 1), Offset(15, 1), {'depth': 1}),
    TimeInterval(Offset(15, 1), Offset(16, 1), {'depth': 0})
])
```

Return interval tree.

timeintervaltools.compute_logical_and_of_intervals

`abjad.tools.timeintervaltools.compute_logical_and_of_intervals.compute_logical_and_of_intervals`
Compute the logical AND of a collection of intervals.

timeintervaltools.compute_logical_and_of_intervals_in_interval

`abjad.tools.timeintervaltools.compute_logical_and_of_intervals_in_interval.compute_logical_and_of_intervals_in_interval`

Compute the logical AND of a collection of intervals, cropped within *interval*.

timeintervaltools.compute_logical_not_of_intervals

`abjad.tools.timeintervaltools.compute_logical_not_of_intervals.compute_logical_not_of_intervals`
Compute the logical NOT of some collection of intervals.

timeintervaltools.compute_logical_not_of_intervals_in_interval

`abjad.tools.timeintervaltools.compute_logical_not_of_intervals_in_interval.compute_logical_not_of_intervals_in_interval`

Compute the logical NOT of some collection of intervals, cropped within *interval*.

timeintervaltools.compute_logical_or_of_intervals

`abjad.tools.timeintervaltools.compute_logical_or_of_intervals.compute_logical_or_of_intervals`
Compute the logical OR of a collection of intervals.

timeintervaltools.compute_logical_or_of_intervals_in_interval

`abjad.tools.timeintervaltools.compute_logical_or_of_intervals_in_interval.compute_logical_or_of_intervals_in_interval`

Compute the logical OR of a collection of intervals, cropped within *interval*.

`timeintervaltools.compute_logical_xor_of_intervals`

`abjad.tools.timeintervaltools.compute_logical_xor_of_intervals.compute_logical_xor_of_intervals`
 Compute the logical XOR of a collections of intervals.

`timeintervaltools.compute_logical_xor_of_intervals_in_interval`

`abjad.tools.timeintervaltools.compute_logical_xor_of_intervals_in_interval.compute_logical_xor_of_intervals_in_interval`

Compute the logical XOR of a collections of intervals, cropped within *interval*.

`timeintervaltools.concatenate_trees`

`abjad.tools.timeintervaltools.concatenate_trees.concatenate_trees` (*trees*,
padding=0
 Merge all trees in *trees*, offsetting each subsequent tree to start after the previous.

`timeintervaltools.explode_intervals_compactly`

`abjad.tools.timeintervaltools.explode_intervals_compactly.explode_intervals_compactly` (*intervals*)
 Explode the intervals in *intervals* into n non-overlapping trees, where n is the maximum depth of *intervals*.
 Returns an array of *TimeIntervalTree* instances.
 The algorithm will attempt to insert the exploded intervals into the lowest-indexed resultant tree with free space.

`timeintervaltools.explode_intervals_into_n_trees_heuristically`

`abjad.tools.timeintervaltools.explode_intervals_into_n_trees_heuristically.explode_intervals_into_n_trees_heuristically` (*intervals*,
n)
 Explode *intervals* into *n* trees, avoiding overlap when possible, and distributing intervals so as to equalize density across the trees.

`timeintervaltools.explode_intervals_uncompactly`

`abjad.tools.timeintervaltools.explode_intervals_uncompactly.explode_intervals_uncompactly` (*intervals*,
n)
 Explode the intervals in *intervals* into n non-overlapping trees, where n is the maximum depth of *intervals*.
 Returns an array of *TimeIntervalTree* instances.
 The algorithm will attempt to insert the exploded intervals cyclically, making its insertion attempt at the next resultant tree in the array, rather than always beginning its search from index 0.

`timeintervaltools.fuse_overlapping_intervals`

`abjad.tools.timeintervaltools.fuse_overlapping_intervals.fuse_overlapping_intervals` (*intervals*)
 Fuse the overlapping intervals in *intervals* and return an *TimeIntervalTree* of the result

```

abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(0, 10)
abjad> b = TimeInterval(5, 15)
abjad> c = TimeInterval(15, 25)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.fuse_overlapping_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(15, 1), {}),
    TimeInterval(Offset(15, 1), Offset(25, 1), {})
])

```

Return interval tree.

`timeintervaltools.fuse_tangent_or_overlapping_intervals`

`abjad.tools.timeintervaltools.fuse_tangent_or_overlapping_intervals.fuse_tangent_or_overlapping_intervals`
 Fuse all tangent or overlapping intervals and return an *TimeIntervalTree* of the result

```

abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(0, 10)
abjad> b = TimeInterval(5, 15)
abjad> c = TimeInterval(15, 25)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.fuse_tangent_or_overlapping_intervals(tree)
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(25, 1), {})
])

```

Return interval tree.

`timeintervaltools.get_all_unique_bounds_in_intervals`

`abjad.tools.timeintervaltools.get_all_unique_bounds_in_intervals.get_all_unique_bounds_in_intervals`
 Return all unique starting and ending boundaries in *intervals*.

`timeintervaltools.group_overlapping_intervals_and_yield_groups`

`abjad.tools.timeintervaltools.group_overlapping_intervals_and_yield_groups.group_overlapping_intervals_and_yield_groups`
 Group overlapping intervals in *intervals* and return tuples.

`timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups`

`abjad.tools.timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups.group_tangent_or_overlapping_intervals_and_yield_groups`
 Group tangent or overlapping intervals in *intervals* and return tuples.

`timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals`

```
abjad.tools.timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals
```

Create a monophonic percussion score from nonoverlapping interval collection *intervals*.

`timeintervaltools.make_polyphonic_percussion_score_from_nonoverlapping_trees`

```
abjad.tools.timeintervaltools.make_polyphonic_percussion_score_from_nonoverlapping_trees.m
```

Make a polyphonic percussion score from a collections of non-overlapping trees.

`timeintervaltools.mask_intervals_with_intervals`

```
abjad.tools.timeintervaltools.mask_intervals_with_intervals.mask_intervals_with_intervals (
```

Clip or remove all intervals in *masked_intervals* outside of the bounds defined in *mask_intervals*, while maintaining *masked_intervals*' payload contents

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(0, 10, {'a': 1})
abjad> b = TimeInterval(5, 15, {'b': 2})
abjad> tree = TimeIntervalTree([a, b])
abjad> mask = TimeInterval(4, 11)
abjad> timeintervaltools.mask_intervals_with_intervals(tree, mask)
TimeIntervalTree([
    TimeInterval(Offset(4, 1), Offset(10, 1), {'a': 1}),
    TimeInterval(Offset(5, 1), Offset(11, 1), {'b': 2})
])
```

Return interval tree.

`timeintervaltools.resolve_overlaps_between_nonoverlapping_trees`

```
abjad.tools.timeintervaltools.resolve_overlaps_between_nonoverlapping_trees.resolve_overlap
```

Create a nonoverlapping `TimeIntervalTree` from *trees*. Intervals in higher-indexed trees in *trees* only appear in part or whole where they do not overlap intervals from starter-indexed trees

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeIntervalTree(TimeInterval(0, 4, {'a': 1}))
abjad> b = TimeIntervalTree(TimeInterval(1, 5, {'b': 2}))
abjad> c = TimeIntervalTree(TimeInterval(2, 6, {'c': 3}))
abjad> d = TimeIntervalTree(TimeInterval(1, 3, {'d': 4}))
abjad> timeintervaltools.resolve_overlaps_between_nonoverlapping_trees([a, b, c, d])
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(4, 1), {'a': 1}),
```

```

    TimeInterval(Offset(4, 1), Offset(5, 1), {'b': 2}),
    TimeInterval(Offset(5, 1), Offset(6, 1), {'c': 3})
])

```

Return interval tree.

`timeintervaltools.resolve_overlaps_between_nonoverlapping_trees_excluding_remainders_less_than_rational`

`abjad.tools.timeintervaltools.resolve_overlaps_between_nonoverlapping_trees_excluding_remainders_less_than_rational`

Create a nonoverlapping `TimeIntervalTree` from *trees*. Intervals in higher-indexed trees in *trees* only appear in part or whole where they do not overlap intervals from starter-indexed trees, and then only where their durations are equal to or greater than *rational*

```

abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeIntervalTree(TimeInterval(0, 1, {'a': 1}))
abjad> b = TimeIntervalTree(TimeInterval(Fraction(1, 32), Fraction(33, 32), {'b': 2}))
abjad> c = TimeIntervalTree(TimeInterval(Fraction(1, 16), Fraction(17, 16), {'c': 3}))
abjad> timeintervaltools.resolve_overlaps_between_nonoverlapping_trees_excluding_remainders_less_than_rational(
TimeIntervalTree([
    TimeInterval(Offset(0, 1), Offset(1, 1), {'a': 1}),
    TimeInterval(Offset(1, 1), Offset(17, 16), {'c': 3})
]))

```

Return interval tree.

`timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational`

`abjad.tools.timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational.round_interval_bounds_to_nearest_multiple_of_rational`

`timeintervaltools.scale_aggregate_duration_by_rational`

`abjad.tools.timeintervaltools.scale_aggregate_duration_by_rational.scale_aggregate_duration_by_rational`

Scale the aggregate duration of all intervals in *intervals* by *rational*, maintaining the original start offset

```

abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])

```

```
abjad> timeintervaltools.scale_aggregate_duration_by_rational(tree, Fraction(1, 3))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(1, 3), {}),
    TimeInterval(Offset(4, 3), Offset(10, 3), {}),
    TimeInterval(Offset(7, 3), Offset(14, 3), {})
])
```

Return interval tree.

timeintervaltools.scale_aggregate_duration_to_rational

`abjad.tools.timeintervaltools.scale_aggregate_duration_to_rational.scale_aggregate_duration`

Scale the aggregate duration of all intervals in *intervals* to *rational*, maintaining the original start offset

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.scale_aggregate_duration_to_rational(tree, Fraction(16, 7))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(-55, 119), {}),
    TimeInterval(Offset(-1, 17), Offset(89, 119), {}),
    TimeInterval(Offset(41, 119), Offset(9, 7), {})
])
```

Return interval tree.

timeintervaltools.scale_interval_durations_by_rational

`abjad.tools.timeintervaltools.scale_interval_durations_by_rational.scale_interval_durations`

Scale the duration of each interval in *intervals* by *rational*, maintaining their start offsets

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.scale_interval_durations_by_rational(tree, Fraction(6, 5))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(19, 5), {}),
    TimeInterval(Offset(6, 1), Offset(66, 5), {}),
    TimeInterval(Offset(9, 1), Offset(87, 5), {})
])
```


Return interval tree.

`timeintervaltools.scale_interval_durations_to_rational`

`abjad.tools.timeintervaltools.scale_interval_durations_to_rational.scale_interval_durations_to_rational`

Scale the duration of each interval in *intervals* to *rational*, maintaining their start offsets

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.scale_interval_durations_to_rational(tree, Fraction(1, 7))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(-6, 7), {}),
    TimeInterval(Offset(6, 1), Offset(43, 7), {}),
    TimeInterval(Offset(9, 1), Offset(64, 7), {})
])
```

Return interval tree.

`timeintervaltools.scale_interval_offsets_by_rational`

`abjad.tools.timeintervaltools.scale_interval_offsets_by_rational.scale_interval_offsets_by_rational`

Scale the offset of each interval in *intervals* by *rational*, maintaining the startest offset in *intervals*

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.scale_interval_offsets_by_rational(tree, Fraction(4, 5))
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(3, 1), {}),
    TimeInterval(Offset(23, 5), Offset(53, 5), {}),
    TimeInterval(Offset(7, 1), Offset(14, 1), {})
])
```

Return interval tree.

timeintervaltools.shift_aggregate_offset_by_rational

abjad.tools.timeintervaltools.shift_aggregate_offset_by_rational.**shift_aggregate_offset_by_rational**

Shift the aggregate offset of *intervals* by *rational*

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.shift_aggregate_offset_by_rational(tree, Fraction(1, 3))
TimeIntervalTree([
    TimeInterval(Offset(-2, 3), Offset(10, 3), {}),
    TimeInterval(Offset(19, 3), Offset(37, 3), {}),
    TimeInterval(Offset(28, 3), Offset(49, 3), {})
])
```

Return interval tree.

timeintervaltools.shift_aggregate_offset_to_rational

abjad.tools.timeintervaltools.shift_aggregate_offset_to_rational.**shift_aggregate_offset_to_rational**

Shift the aggregate offset of *intervals* to *rational*

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.shift_aggregate_offset_to_rational(tree, Fraction(10, 7))
TimeIntervalTree([
    TimeInterval(Offset(10, 7), Offset(38, 7), {}),
    TimeInterval(Offset(59, 7), Offset(101, 7), {}),
    TimeInterval(Offset(80, 7), Offset(129, 7), {})
])
```

Return interval tree.

timeintervaltools.split_intervals_at_rationals

abjad.tools.timeintervaltools.split_intervals_at_rationals.**split_intervals_at_rationals** (*interval tree*)

Split *intervals* at each rational in *rationals*

```
abjad> from abjad.tools import timeintervaltools
abjad> from abjad.tools.timeintervaltools import TimeInterval
abjad> from abjad.tools.timeintervaltools import TimeIntervalTree

abjad> a = TimeInterval(-1, 3)
abjad> b = TimeInterval(6, 12)
abjad> c = TimeInterval(9, 16)
abjad> tree = TimeIntervalTree([a, b, c])
abjad> timeintervaltools.split_intervals_at_rationals(tree, [1, Fraction(19, 2)])
TimeIntervalTree([
    TimeInterval(Offset(-1, 1), Offset(1, 1), {}),
    TimeInterval(Offset(1, 1), Offset(3, 1), {}),
    TimeInterval(Offset(6, 1), Offset(19, 2), {}),
    TimeInterval(Offset(9, 1), Offset(19, 2), {}),
    TimeInterval(Offset(19, 2), Offset(12, 1), {}),
    TimeInterval(Offset(19, 2), Offset(16, 1), {})
])
```

Return interval tree.

timesignaturetools**timesignaturetools functions****timesignaturetools.duration_and_possible_denominators_to_time_signature**

abjad.tools.timesignaturetools.duration_and_possible_denominators_to_time_signature.**duration_and_possible_denominators_to_time_signature**

Make new meter equal to *duration*:

```
abjad> from abjad.tools import timesignaturetools

abjad> timesignaturetools.duration_and_possible_denominators_to_time_signature(Duration(3, 2))
TimeSignatureMark((3, 2))
```

Make new meter equal to *duration* with denominator equal to the first possible element in *denominators*:

```
abjad> timesignaturetools.duration_and_possible_denominators_to_time_signature(Duration(3, 2), denominators=[3, 6, 9])
TimeSignatureMark((9, 6))
```

Make new meter equal to *duration* with denominator divisible by *factor*:

```
abjad> timesignaturetools.duration_and_possible_denominators_to_time_signature(Duration(3, 2), f
TimeSignatureMark((15, 10))
```

Return new meter. Changed in version 2.0: renamed `timesignaturetools.make_best()` to `timesignaturetools.duration_and_possible_denominators_to_time_signature()`.

`timesignaturetools.get_nonbinary_factor_from_time_signature_denominator`

```
abjad.tools.timesignaturetools.get_nonbinary_factor_from_time_signature_denominator.get_non
```

Get nonbinary factor from nonbinary *meter* denominator:

```
abjad> from abjad.tools import timesignaturetools
```

```
abjad> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(contexttools.Time
3
```

```
abjad> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(contexttools.Time
13
```

```
abjad> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(contexttools.Time
7
```

```
abjad> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(contexttools.Time
15
```

Get 1 from binary *meter* denominator:

```
abjad> timesignaturetools.get_nonbinary_factor_from_time_signature_denominator(contexttools.Time
1
```

Return nonnegative integer.

`timesignaturetools.is_time_signature_with_equivalent_binary_representation`

```
abjad.tools.timesignaturetools.is_time_signature_with_equivalent_binary_representation.is_t
```

True when *expr* is a meter with binary-valued duration:

```
abjad> from abjad.tools import timesignaturetools
```

```
abjad> timesignaturetools.is_time_signature_with_equivalent_binary_representation(contexttools.T
True
```

Otherwise false:

```
abjad> timesignaturetools.is_time_signature_with_equivalent_binary_representation(contexttools.T
False
```

```
abjad> timesignaturetools.is_time_signature_with_equivalent_binary_representation('text')
False
```

Return boolean.

timesignaturetools.time_signature_to_binary_time_signature

abjad.tools.timesignaturetools.time_signature_to_binary_time_signature.time_signature_to_b

Change nonbinary *meter* to binary meter:

```
abjad> from abjad.tools import timesignaturetools

abjad> timesignaturetools.time_signature_to_binary_time_signature(contexttools.TimeSignatureMark(
TimeSignatureMark((2, 8))
```

Preserve binary *meter*:

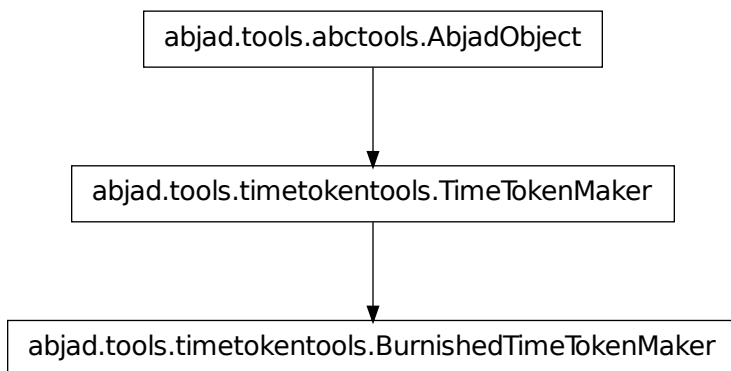
```
abjad> timesignaturetools.time_signature_to_binary_time_signature(contexttools.TimeSignatureMark(
TimeSignatureMark((2, 8))
```

Return newly constructed meter. Changed in version 2.0: renamed timesignaturetools.make_binary() to timesignaturetools.time_signature_to_binary_time_sig

timetokentools

timetokentools abstract classes

timetokentools.BurnishedTimeTokenMaker



class abjad.tools.timetoken.tools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.**BurnishedTimeTokenMaker**

New in version 2.8. Abstract base class for time token makers that burnish some of all of the time tokens they produce.

‘Burnishing’ means to forcibly cast the first or last (or both first and last) elements of a time token to be either a note or rest.

‘Token-burnishing’ time token makers burnish every time token they produce.

‘Output-burnishing’ time token makers burnish only the first and last time tokens they produce and leave interior time tokens unchanged.

Special Methods

BurnishedTimeTokenMaker.**__call__** (*duration_tokens*, *seeds=None*)

BurnishedTimeTokenMaker.**__delattr__** ()

x.**__delattr__** (‘name’) <==> del x.name

Note: Inherited from `__builtin__.object`

BurnishedTimeTokenMaker.**__eq__** (*other*)

BurnishedTimeTokenMaker.**__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`BurnishedTimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BurnishedTimeTokenMaker.__ne__(other)`

`BurnishedTimeTokenMaker.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

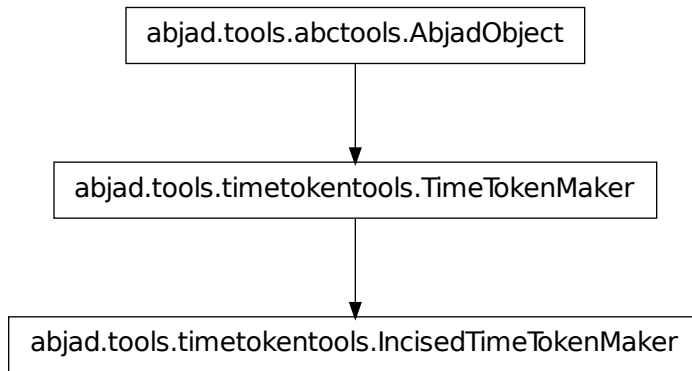
`BurnishedTimeTokenMaker.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BurnishedTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`timetokentools.IncisedTimeTokenMaker`



```
class abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker(IncisedTimeToken
```

Abstract base class for time token makers that incise some or all of the time tokens they produce.

Time token makers can incise the edge of every time token.

Or time token makers can incise only the start of the first time token and the end of the last time token.

Special Methods

`IncisedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

`IncisedTimeTokenMaker.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`IncisedTimeTokenMaker.__eq__(other)`

`IncisedTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`IncisedTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`IncisedTimeTokenMaker.__setattr__()`

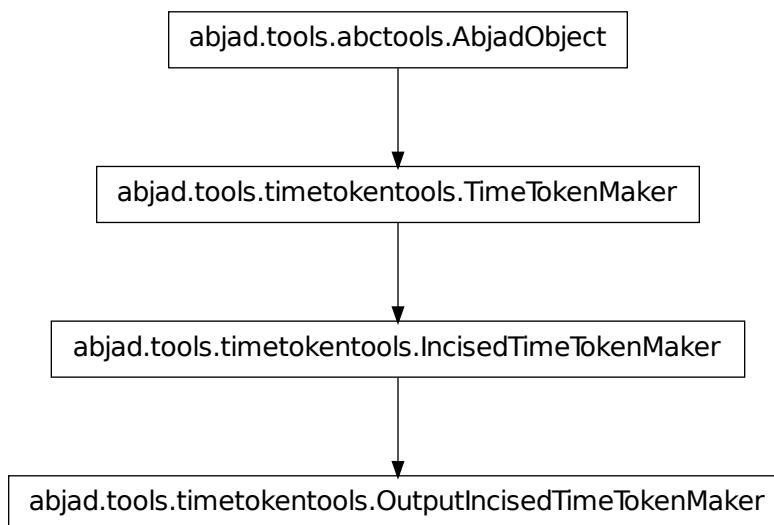
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`IncisedTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`timetokentools.OutputIncisedTimeTokenMaker`



class `abjad.tools.timetoken.tools.OutputIncisedTimeTokenMaker`.`OutputIncisedTimeTokenMaker`.`OutputIncisedTimeTokenMaker`

New in version 2.8. Abstract base class for time token makers that incise only the first and last time tokens they produce.

Special Methods

`OutputIncisedTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`OutputIncisedTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OutputIncisedTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

`OutputIncisedTimeTokenMaker.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`OutputIncisedTimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedTimeTokenMaker.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

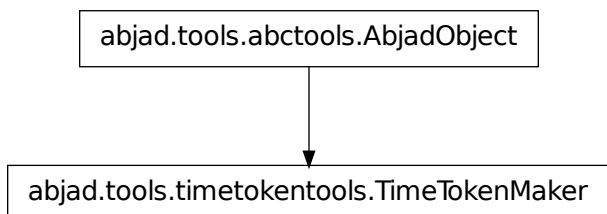
`OutputIncisedTimeTokenMaker.__setattr__()`
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`OutputIncisedTimeTokenMaker.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

timetokentools.TimeTokenMaker



class `abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker`. **TimeTokenMaker**
 New in version 2.8. Time token maker abstract base class.

Special Methods

`TimeTokenMaker.__call__(duration_tokens, seeds=None)`

`TimeTokenMaker.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TimeTokenMaker.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

TimeTokenMaker.__hash__() <==> hash(x)

Note: Inherited from `__builtin__.object`

TimeTokenMaker.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeTokenMaker.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

TimeTokenMaker.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

TimeTokenMaker.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

TimeTokenMaker.__setattr__()

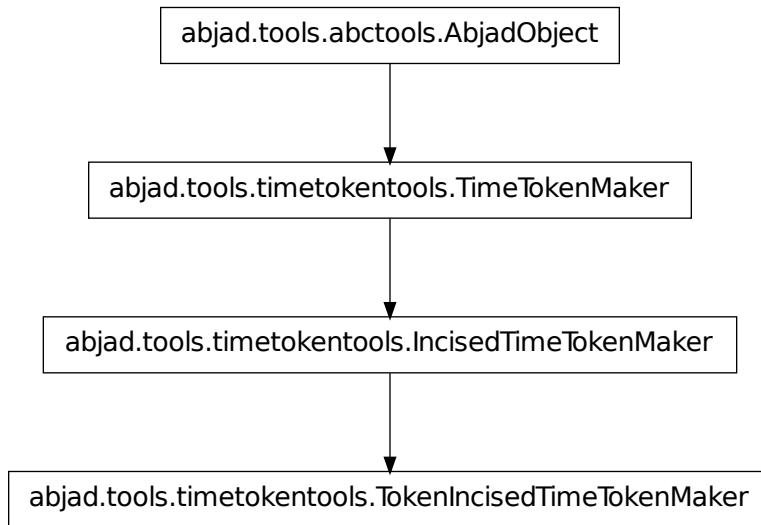
`x.__setattr__('name', value)` <==> `x.name = value`

Note: Inherited from `__builtin__.object`

TimeTokenMaker.__str__() <==> str(x)

Note: Inherited from `__builtin__.object`

`timetokentools.TokenIncisedTimeTokenMaker`



class abjad.tools.timetoken.tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.**TokenI**

New in version 2.8. Abstract base class for time token makers that incise every time token they produce.

Special Methods

TokenIncisedTimeTokenMaker.**__call__**(*duration_tokens*, *seeds=None*)

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

TokenIncisedTimeTokenMaker.**__delattr__**()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

TokenIncisedTimeTokenMaker.**__eq__**(*other*)

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

TokenIncisedTimeTokenMaker.**__ge__**(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TokenIncisedTimeTokenMaker.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedTimeTokenMaker.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

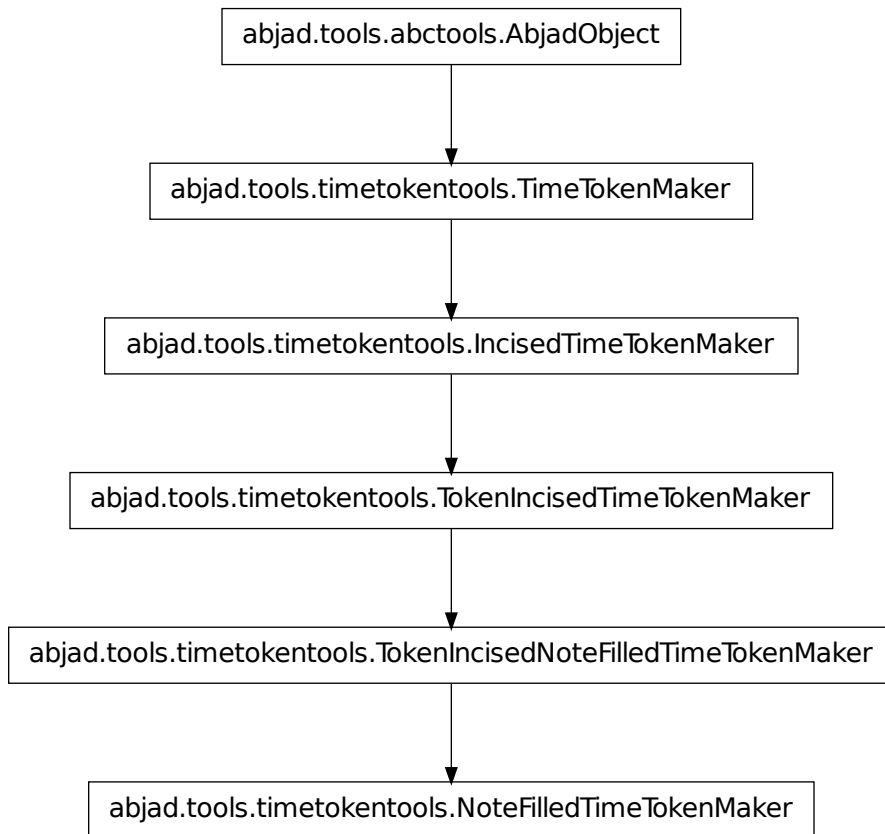
Note: Inherited from `__builtin__.object`

`TokenIncisedTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

timetokentools concrete classes

timetokentools.NoteFilledTimeTokenMaker



class `abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker`. **NoteFilledTimeTokenMaker**. New in version 2.8. Note-filled time-token maker:

```

abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> maker = timetokentools.NoteFilledTimeTokenMaker()

abjad> duration_tokens = [(5, 16), (3, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)
  
```

```
abjad> f(staff)
\new Staff {
  {
    \time 5/16
    c'4
    c'16
  }
  {
    \time 3/8
    c'4.
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

NoteFilledTimeTokenMaker.**__call__** (*duration_tokens*, *seeds=None*)

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

NoteFilledTimeTokenMaker.**__delattr__** ()
x.**__delattr__** ('name') <==> del x.name

Note: Inherited from `__builtin__.object`

NoteFilledTimeTokenMaker.**__eq__** (*other*)

Note: Inherited from `timetoken.tools.IncisedTimeTokenMaker`

NoteFilledTimeTokenMaker.**__ge__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

NoteFilledTimeTokenMaker.**__gt__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

NoteFilledTimeTokenMaker.**__hash__** () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

`NoteFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NoteFilledTimeTokenMaker.__repr__()`

`NoteFilledTimeTokenMaker.__setattr__()`

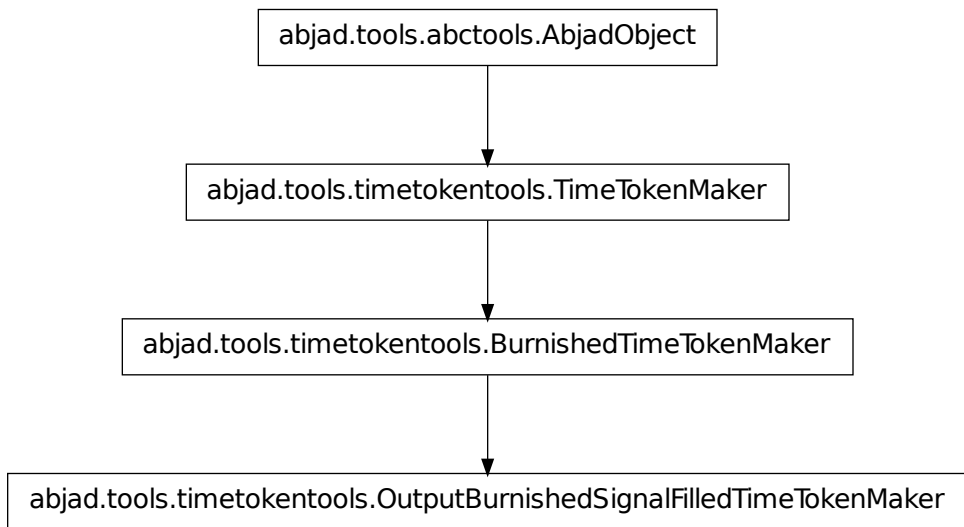
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NoteFilledTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`timetokentools.OutputBurnishedSignalFilledTimeTokenMaker`



class `abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker`.`OutputBurnishedSig`

New in version 2.8. Output-burnished signal-filled time-token maker.

Configure the time-token maker at initialization:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> pattern, denominator, prolation_addenda = [1, 2, 3], 16, [0, 2]
abjad> lefts, middles, rights = [-1], [0], [-1]
abjad> left_lengths, right_lengths = [1], [1]
abjad> secondary_divisions = [9]
abjad> maker = timetokentools.OutputBurnishedSignalFilledTimeTokenMaker(pattern, denominator, pr
```

Then call the time-token maker on arbitrary duration tokens:

```
abjad> duration_tokens = [(3, 8), (4, 8)]
abjad> music = maker(duration_tokens)
```

The resulting Abjad objects can be included in any score:

```
abjad> music = sequencetools.flatten_sequence(music)
abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)

abjad> f(staff)
\new Staff {
    {
```

```
\time 3/8
{
    r16
    c'8
    c'8.
}
}
{
    \time 4/8
    \fraction \times 3/5 {
        c'16
        c'8
        c'8
    }
    {
        c'16
        c'16
        c'8
        r16
    }
}
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`OutputBurnishedSignalFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`OutputBurnishedSignalFilledTimeTokenMaker.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OutputBurnishedSignalFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`OutputBurnishedSignalFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`OutputBurnishedSignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputBurnishedSignalFilledTimeTokenMaker.__ne__(other)`

Note: Inherited from `timetokenools.BurnishedTimeTokenMaker`

`OutputBurnishedSignalFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

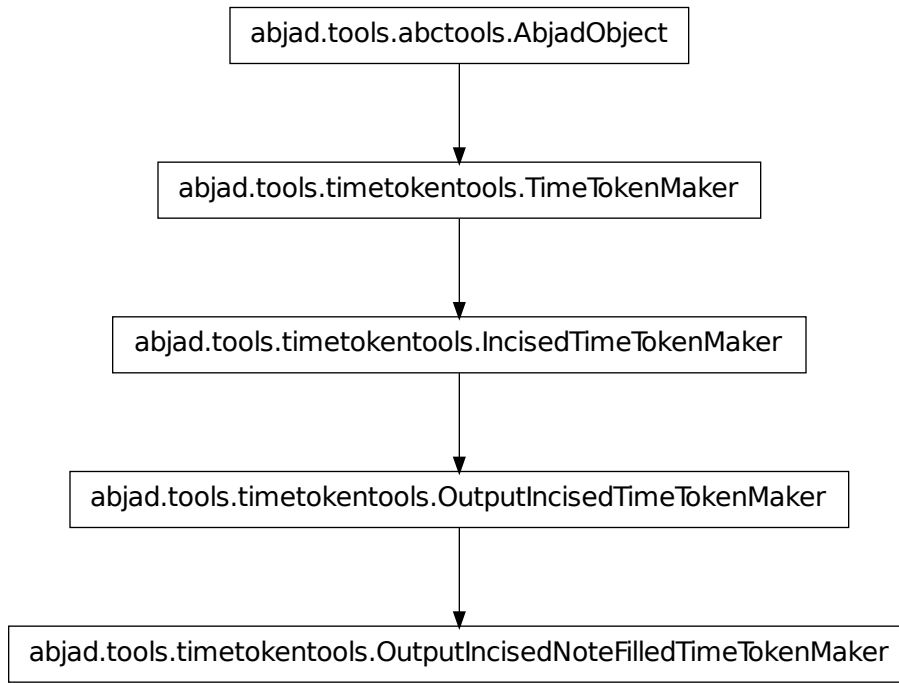
`OutputBurnishedSignalFilledTimeTokenMaker.__setattr__()`

`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`OutputBurnishedSignalFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

timetokentools.OutputIncisedNoteFilledTimeTokenMaker

class abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFille

New in version 2.8. Output-incised note-filled time-token maker.

Configure the time-token maker on initialization:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> prefix_signal, prefix_lengths = [-8], [2]
abjad> suffix_signal, suffix_lengths = [-3], [4]
abjad> denominator = 32
abjad> maker = timetokentools.OutputIncisedNoteFilledTimeTokenMaker(prefix_signal, prefix_lengths)
```

Then call the time-token maker on arbitrary duration tokens:

```
abjad> duration_tokens = [(5, 8), (5, 8), (5, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)
```

The resulting Abjad objects can be included in any score and the time-token maker can be reused:

```
abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

abjad> f(staff)
\new Staff {
    {
        \time 5/8
```

```
        r4
        r4
        c'8
    }
    {
        \time 5/8
        c'2
        c'8
    }
    {
        \time 5/8
        c'4
        r16.
        r16.
        r16.
        r16.
    }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`OutputIncisedNoteFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedNoteFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OutputIncisedNoteFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedNoteFilledTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`OutputIncisedNoteFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedNoteFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

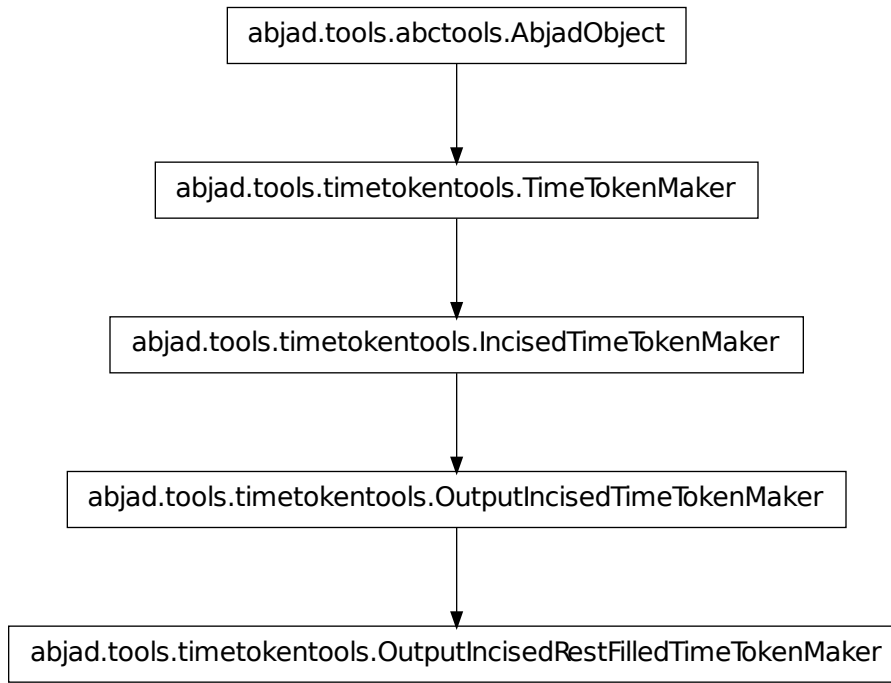
`OutputIncisedNoteFilledTimeTokenMaker.__setattr__()`

`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`OutputIncisedNoteFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

timetokentools.OutputIncisedRestFilledTimeTokenMaker

class abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFille

New in version 2.8. Output-incised rest-filled time-token maker.

Configure the time-token maker on initialization:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> prefix_signal, prefix_lengths = [8], [2]
abjad> suffix_signal, suffix_lengths = [3], [4]
abjad> denominator = 32
abjad> maker = timetokentools.OutputIncisedRestFilledTimeTokenMaker(prefix_signal, prefix_lengths,
```

Then call the time-token maker on arbitrary duration tokens:

```
abjad> duration_tokens = [(5, 8), (5, 8), (5, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)
```

The resulting Abjad objects can be included in any score and the time-token maker can be reused:

```
abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

abjad> f(staff)
\new Staff {
    {
        \time 5/8
```

```

        c'4
        c'4
        r8
    }
    {
        \time 5/8
        r2
        r8
    }
    {
        \time 5/8
        r4
        c'16.
        c'16.
        c'16.
        c'16.
    }
}

```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`OutputIncisedRestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedRestFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OutputIncisedRestFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`OutputIncisedRestFilledTimeTokenMaker.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`OutputIncisedRestFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OutputIncisedRestFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

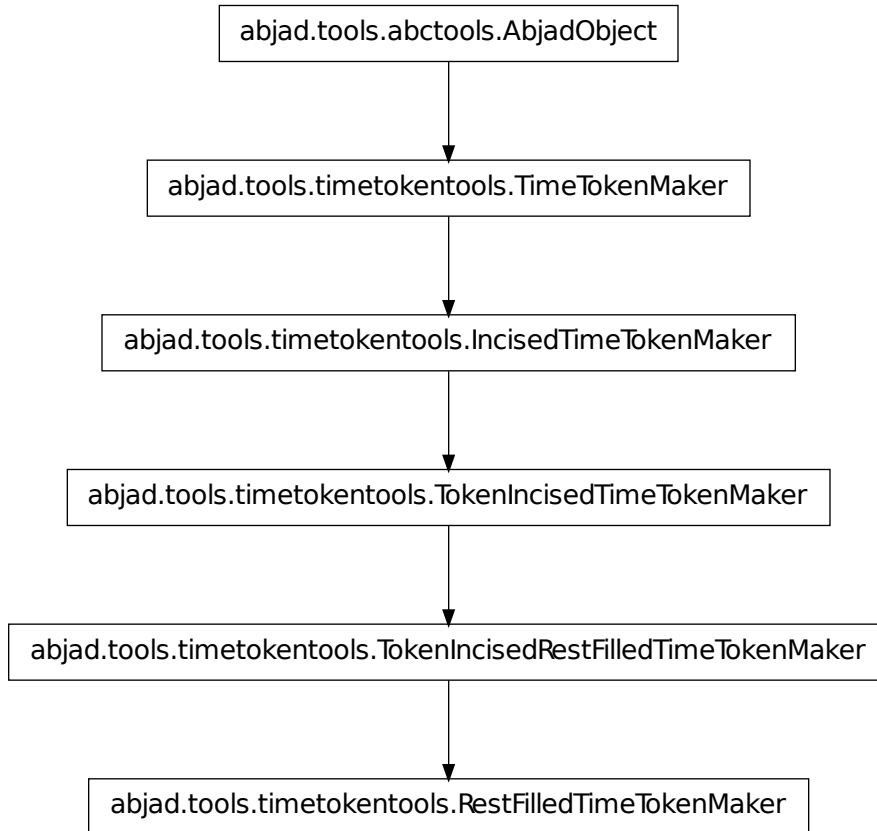
`OutputIncisedRestFilledTimeTokenMaker.__setattr__()`

`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`OutputIncisedRestFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

timetokentools.RestFilledTimeTokenMaker

class `abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker`. **RestFilledTimeTokenMaker**. New in version 2.8. Rest-filled time-token maker:

```

abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> maker = timetokentools.RestFilledTimeTokenMaker()

abjad> duration_tokens = [(5, 16), (3, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

abjad> f(staff)
\new Staff {
    {
        \time 5/16
    }
}

```

```

        r4
        r16
    }
    {
        \time 3/8
        r4.
    }
}

```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`RestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`RestFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`RestFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`RestFilledTimeTokenMaker.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`RestFilledTimeTokenMaker.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RestFilledTimeTokenMaker.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

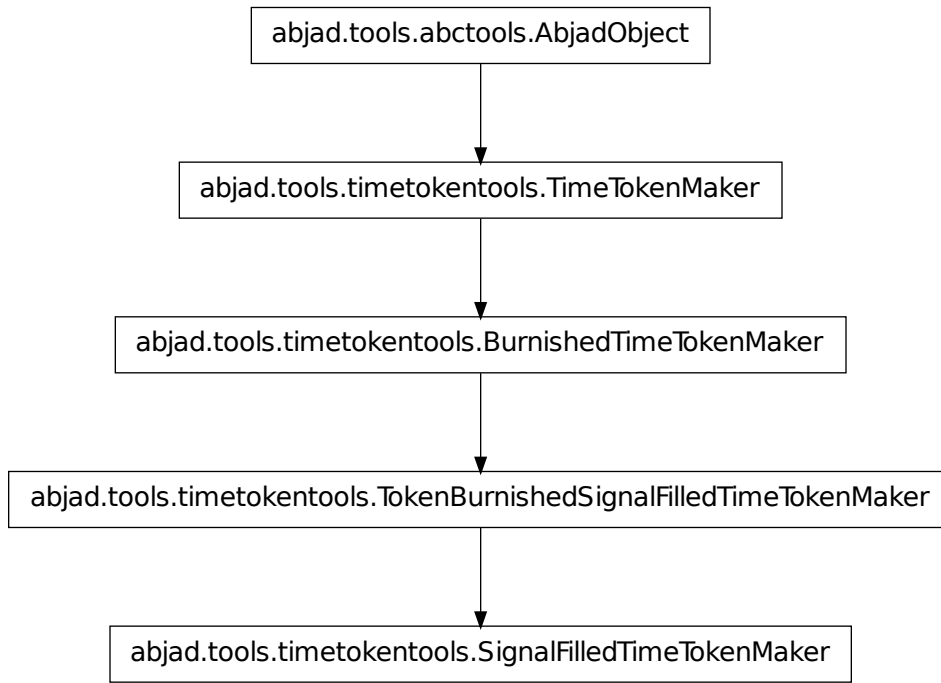
`RestFilledTimeTokenMaker.__repr__()`
`RestFilledTimeTokenMaker.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`RestFilledTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

timetokentools.SignalFilledTimeTokenMaker



class `abjad.tools.timetokentools.SignalFilledTimeTokenMaker`, `SignalFilledTimeTokenMaker`. **Signal**

New in version 2.8. Signal-affixed time-token maker.

Configure the time-token maker at initialization:

```

abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

```

```
abjad> pattern, denominator, prolation_addenda = [-1, 4, -2, 3], 16, [3, 4]
abjad> maker = timetokentools.SignalFilledTimeTokenMaker(pattern, denominator, prolation_addenda)
```

Then call the time-token maker on arbitrary duration tokens:

```
abjad> duration_tokens = [(2, 8), (5, 8)]
abjad> music = maker(duration_tokens)
```

The resulting Abjad objects can be included in any score and the time-token maker can be called indefinitely on other arbitrary sequences of duration tokens:

```
abjad> music = sequencetools.flatten_sequence(music)
abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)
```

```
abjad> f(staff)
\new Staff {
  {
    \time 2/8
    \times 4/7 {
      r16
      c'4
      r8
    }
  }
  {
    \time 5/8
    \fraction \times 5/7 {
      c'8.
      r16
      c'4
      r8
      c'8.
      r16
    }
  }
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`SignalFilledTimeTokenMaker.__call__` (*duration_tokens*, *seeds=None*)

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`SignalFilledTimeTokenMaker.__delattr__` ()
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SignalFilledTimeTokenMaker.__eq__` (*other*)

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`SignalFilledTimeTokenMaker.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`SignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__ne__(other)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`SignalFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`SignalFilledTimeTokenMaker.__setattr__()`

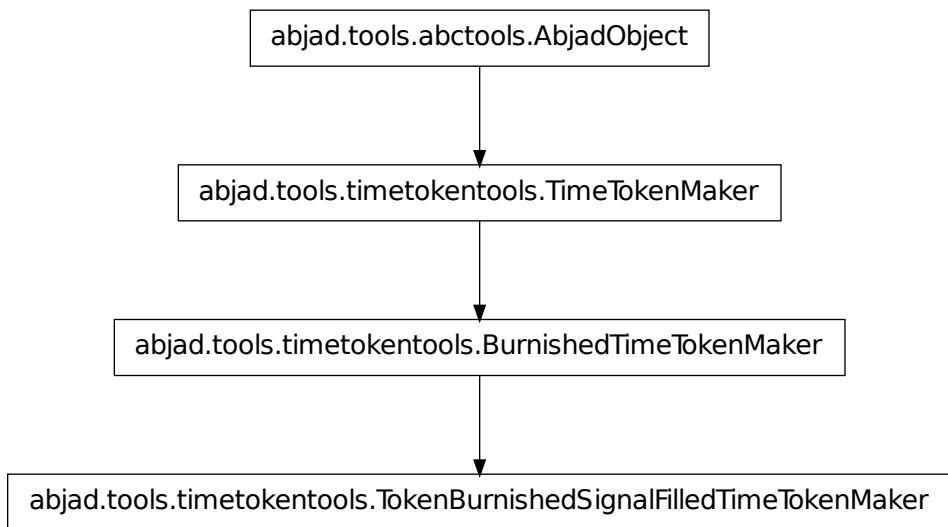
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`SignalFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

timetoken.tools.TokenBurnishedSignalFilledTimeTokenMaker



class abjad.tools.timetokentools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker

New in version 2.8. Token-burnished signal-filled time-token maker.

Configure the time-token maker at instantiation:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> pattern, denominator, prolation_addenda = [1, 1, 2, 4], 32, [0, 3]
abjad> lefts, middles, rights = [-1], [0], [-1]
abjad> left_lengths, right_lengths = [1], [1]
abjad> secondary_divisions = [14]
abjad> maker = timetokentools.TokenBurnishedSignalFilledTimeTokenMaker(pattern, denominator, prolation_addenda, lefts, middles, rights, left_lengths, right_lengths, secondary_divisions)
```

Then call the time-token maker on any sequence of duration tokens:

```
abjad> duration_tokens = [(5, 16), (6, 16)]
abjad> music = maker(duration_tokens)
```

The resulting Abjad objects can then be included in any score and the time-token maker can be called again and again on different duration tokens:

```
abjad> music = sequencetools.flatten_sequence(music)
abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, music)

abjad> f(staff)
\new Staff {
```



```
{
  \time 5/16
  {
    r32
    c'32
    c'16
    c'8
    c'32
    r32
  }
}
{
  \time 6/16
  \times 4/7 {
    r16
    c'8
    r32
  }
  {
    r32
    c'16
    c'8
    r32
  }
}
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`TokenBurnishedSignalFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`TokenBurnishedSignalFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TokenBurnishedSignalFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`TokenBurnishedSignalFilledTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TokenBurnishedSignalFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__ne__(other)`

Note: Inherited from `timetokentools.BurnishedTimeTokenMaker`

`TokenBurnishedSignalFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`TokenBurnishedSignalFilledTimeTokenMaker.__setattr__()`

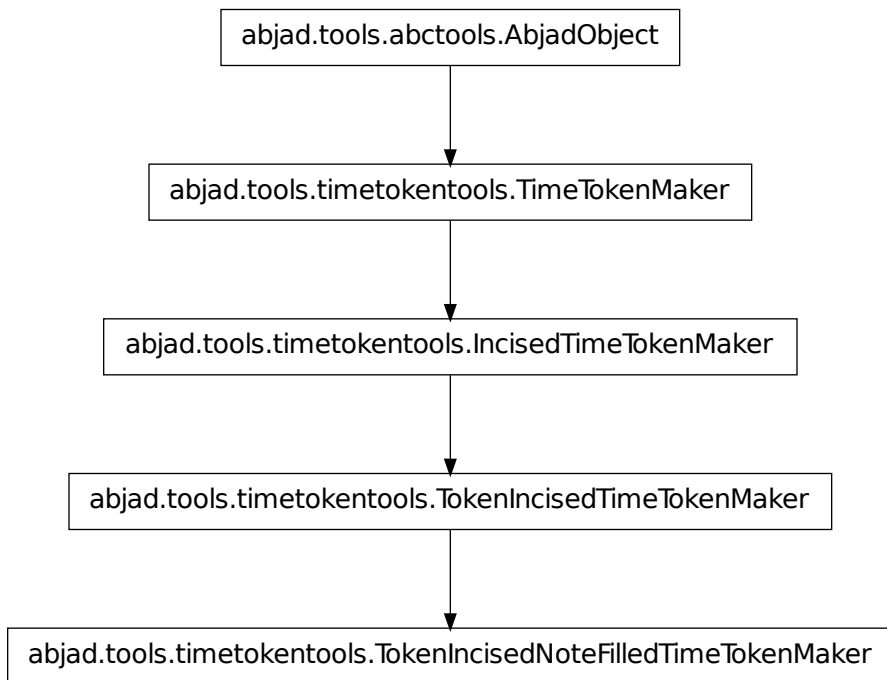
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TokenBurnishedSignalFilledTimeTokenMaker.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`timetokentools.TokenIncisedNoteFilledTimeTokenMaker`



class abjad.tools.timetokentools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledT

New in version 2.8. Token-incised note-filled time-token maker:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> prefix_signal, prefix_lengths = [-8], [0, 1]
abjad> suffix_signal, suffix_lengths = [-1], [1]
abjad> denominator = 32
abjad> maker = timetokentools.TokenIncisedNoteFilledTimeTokenMaker(prefix_signal, prefix_lengths)

abjad> duration_tokens = [(5, 8), (5, 8), (5, 8), (5, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

abjad> f(staff)
\new Staff {
  {
    \time 5/8
    c'2
    c'16.
    r32
  }
}
```

```
\time 5/8
r4
c'4
c'16.
r32
}
{
\time 5/8
c'2
c'16.
r32
}
{
\time 5/8
r4
c'4
c'16.
r32
}
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`TokenIncisedNoteFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedNoteFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TokenIncisedNoteFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedNoteFilledTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`TokenIncisedNoteFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedNoteFilledTimeTokenMaker.__setattr__()`

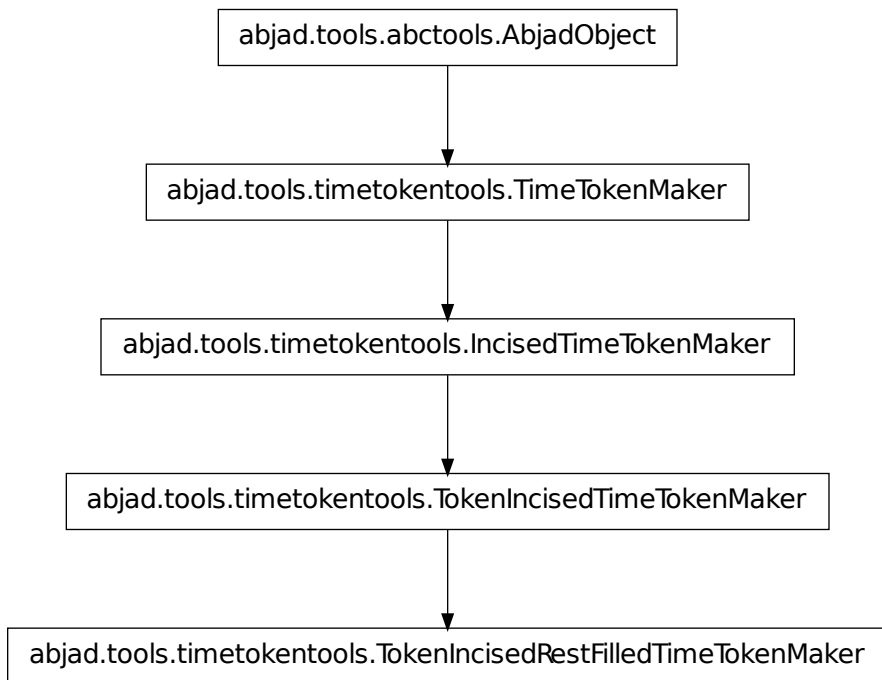
`x.__setattr__('name', value)` \Leftrightarrow `x.name = value`

Note: Inherited from `__builtin__.object`

`TokenIncisedNoteFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`timetokentools.TokenIncisedRestFilledTimeTokenMaker`



class abjad.tools.timetokentools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledT

New in version 2.8. Token-incised rest-filled time-token maker:

```
abjad> from abjad.tools import sequencetools
abjad> from abjad.tools import timetokentools

abjad> prefix_signal, prefix_lengths = [8], [1, 2, 3, 4]
abjad> suffix_signal, suffix_lengths = [1], [1]
abjad> denominator = 32
abjad> maker = timetokentools.TokenIncisedRestFilledTimeTokenMaker(prefix_signal, prefix_lengths)

abjad> duration_tokens = [(5, 8), (5, 8), (5, 8), (5, 8)]
abjad> leaf_lists = maker(duration_tokens)
abjad> leaves = sequencetools.flatten_sequence(leaf_lists)

abjad> staff = Staff(measuretools.make_measures_with_full_measure_spacer_skips(duration_tokens))
abjad> measures = measuretools.replace_contents_of_measures_in_expr(staff, leaves)

abjad> f(staff)
\new Staff {
    {
        \time 5/8
        c'4
        r4
        r16.
        c'32
    }
}
```



```
{
    \time 5/8
    c'4
    c'4
    r16.
    c'32
}
{
    \time 5/8
    c'4
    c'4
    c'8
}
{
    \time 5/8
    c'4
    c'4
    c'8
}
}
```

Usage follows the two-step instantiate-then-call pattern shown here.

Return time-token maker.

Special Methods

`TokenIncisedRestFilledTimeTokenMaker.__call__(duration_tokens, seeds=None)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedRestFilledTimeTokenMaker.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TokenIncisedRestFilledTimeTokenMaker.__eq__(other)`

Note: Inherited from `timetokentools.IncisedTimeTokenMaker`

`TokenIncisedRestFilledTimeTokenMaker.__ge__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__gt__(arg)`
Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`TokenIncisedRestFilledTimeTokenMaker.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`TokenIncisedRestFilledTimeTokenMaker.__setattr__()`

`x.__setattr__('name', value)` \Leftrightarrow `x.name = value`

Note: Inherited from `__builtin__.object`

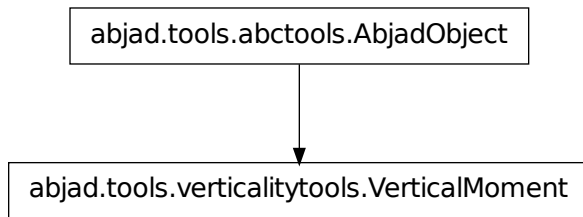
`TokenIncisedRestFilledTimeTokenMaker.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

verticalitytools

verticalitytools concrete classes

verticalitytools.VerticalMoment



class `abjad.tools.verticalitytools.VerticalMoment.VerticalMoment` (*prolated_offset*, *gov-*, *er-*, *nors*, *com-*, *po-*, *nents*)

Everything happening at a single moment in musical time:

```

abjad> from abjad.tools import verticalitytools

abjad> score = Score([scoretools.PianoStaff([Staff("c'4 e'4 d'4 f'4"), Staff('g2 f2')])])
abjad> contexttools.ClefMark('bass')(score[0][1])
ClefMark('bass')(Staff{2})

f(score)
\new Score <<
  \new PianoStaff <<
    \new Staff {
      c'4
      e'4
      d'4
      f'4
    }
    \new Staff {
      \clef "bass"
      g2
      f2
    }
  >>
>>

abjad> for vertical_moment in verticalitytools.iterate_vertical_moments_forward_in_expr(score):
...     vertical_moment
...
VerticalMoment(0, <<2>>)
  
```

```
VerticalMoment(1/4, <<2>>)  
VerticalMoment(1/2, <<2>>)  
VerticalMoment(3/4, <<2>>)
```

Create vertical moments with the getters and iterators implemented in the `verticalitytools` module.

Vertical moments are immutable.

Read-only Properties

`VerticalMoment.attack_count`

Positive integer number of pitch carriers starting at vertical moment.

`VerticalMoment.components`

Read-only tuple of zero or more components happening at vertical moment.

It is always the case that `self.components = self.overlap_components + self.start_components`.

`VerticalMoment.governors`

Read-only tuple of one or more containers in which vertical moment is evaluated.

`VerticalMoment.leaves`

Read-only tuple of zero or more leaves at vertical moment.

`VerticalMoment.measures`

Read-only tuplet of zero or more measures at vertical moment.

`VerticalMoment.next_vertical_moment`

Read-only reference to next vertical moment forward in time.

`VerticalMoment.notes`

Read-only tuple of zero or more notes at vertical moment.

`VerticalMoment.overlap_components`

Read-only tuple of components in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_leaves`

Read-only tuple of leaves in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_measures`

Read-only tuple of measures in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.overlap_notes`

Read-only tuple of notes in vertical moment starting before vertical moment, ordered by score index.

`VerticalMoment.prev_vertical_moment`

Read-only reference to prev vertical moment backward in time.

`VerticalMoment.prolated_offset`

Read-only rational-valued score offset at which vertical moment is evaluated.

`VerticalMoment.start_components`

Read-only tuple of components in vertical moment starting with at vertical moment, ordered by score index.

`VerticalMoment.start_leaves`

Read-only tuple of leaves in vertical moment starting with vertical moment, ordered by score index.

`VerticalMoment.start_notes`

Read-only tuple of notes in vertical moment starting with vertical moment, ordered by score index.

Special Methods

`VerticalMoment.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`VerticalMoment.__eq__(expr)`

`VerticalMoment.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`VerticalMoment.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`VerticalMoment.__hash__()`

`VerticalMoment.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`VerticalMoment.__len__()`

`VerticalMoment.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`VerticalMoment.__ne__(expr)`

`VerticalMoment.__repr__()`

`VerticalMoment.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`VerticalMoment.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

verticalitytools functions

verticalitytools.get_vertical_moment_at_prolated_offset_in_expr

abjad.tools.verticalitytools.get_vertical_moment_at_prolated_offset_in_expr.get_vertical_moment_at_prolated_offset_in_expr

New in version 2.0. Get vertical moment at *prolated_offset* in *governor*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score([])
abjad> score.append(Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeated_notes(2, Duration(1, 4)))])
abjad> piano_staff = scoretools.PianoStaff([])
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(2, Duration(1, 4))))
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(4)))
abjad> contexttools.ClefMark('bass') (piano_staff[1])
ClefMark('bass') (Staff{4})
abjad> score.append(piano_staff)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(list(score.leaves))
abjad> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  >>
>>
abjad> vertical_moment = verticalitytools.get_vertical_moment_at_prolated_offset_in_expr(piano_staff[1])
abjad> vertical_moment.leaves
(Note("a'4"), Note("e'8"))
```

Todo

optimize without full-component traversal.

Changed in version 2.0: renamed `iterate.get_vertical_moment_at_prolated_offset_in()` to `verticalitytools.get_vertical_moment_at_prolated_offset_in_expr()`.

verticalitytools.get_vertical_moment_starting_with_component

abjad.tools.verticalitytools.get_vertical_moment_starting_with_component.get_vertical_moment_starting_with_component

New in version 2.0. When *governor* is none, get vertical moment at `expr._offset.start` in score root of *expr*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score([])
abjad> score.append(Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeated_notes(2, Duration(1, 4)))])
abjad> piano_staff = scoretools.PianoStaff([])
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(2, Duration(1, 4))))
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(4)))
abjad> contexttools.ClefMark('bass') (piano_staff[1])
ClefMark('bass') (Staff{4})
abjad> score.append(piano_staff)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(list(score))
abjad> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  >>
>>
abjad> verticalitytools.get_vertical_moment_starting_with_component(piano_staff[1][1])
VerticalMoment(1/8, <<3>>)
```

When *governor* is not none, get vertical moment at `expr._offset.start` in *governor*.

```
abjad> verticalitytools.get_vertical_moment_starting_with_component(piano_staff[1][1], piano_staff[0][1])
VerticalMoment(1/8, <<2>>)
```

Todo

optimize without full-component traversal.

Changed in version 2.0: renamed `iterate.get_vertical_moment_starting_with()` to `verticalitytools.get_vertical_moment_starting_with_component()`. Changed `iterate.get_vertical_moment_starting_with()` to `verticalitytools.get_vertical_moment_starting_with_component()`.

version 2.0: renamed `iterate.get_vertical_moment_starting_with_component()` to `verticalitytools.get_vertical_moment_starting_with_component()`.

`verticalitytools.iterate_vertical_moments_backward_in_expr`

`abjad.tools.verticalitytools.iterate_vertical_moments_backward_in_expr.iterate_vertical_moments_backward_in_expr`

New in version 2.0. Yield vertical moments forward in *governor*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score([])
abjad> score.append(Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeat
abjad> piano_staff = scoretools.PianoStaff([])
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(2, Duration(1, 4))))
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(4)))
abjad> contexttools.ClefMark('bass')(piano_staff[1])
ClefMark('bass')(Staff{4})
abjad> score.append(piano_staff)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(list
abjad> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  }
>>
>>
abjad> for vertical_moment in verticalitytools.iterate_vertical_moments_backward_in_expr(score):
...     vertical_moment.leaves
...
(Note("b'8"), Note("g'4"), Note("c'8"))
(Note("b'8"), Note("g'4"), Note("d'8"))
(Note("c''8"), Note("g'4"), Note("d'8"))
(Note("c''8"), Note("a'4"), Note("e'8"))
(Note("d''8"), Note("a'4"), Note("e'8"))
(Note("d''8"), Note("a'4"), Note("f'8"))
abjad> for vertical_moment in verticalitytools.iterate_vertical_moments_backward_in_expr(piano_s
...     vertical_moment.leaves
...
(Note("g'4"), Note("c'8"))
(Note("g'4"), Note("d'8"))
(Note("a'4"), Note("e'8"))
```



```
(Note("a'4"), Note("f'8"))
```

Todo

optimize without multiple full-component traversal.

Changed in version 2.0: renamed `iterate.vertical_moments_backward_in()` to `verticalitytools.iterate_vertical_moments_backward_in_expr()`.
 Changed in version 2.0: renamed `iterate.vertical_moments_backward_in_expr()` to `verticalitytools.iterate_vertical_moments_backward_in_expr()`.

verticalitytools.iterate_vertical_moments_forward_in_expr

`abjad.tools.verticalitytools.iterate_vertical_moments_forward_in_expr.iterate_vertical_moments_forward_in_expr`

New in version 2.0. Yield vertical moments forward in *governor*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score([])
abjad> score.append(Staff([tuplettools.FixedDurationTuplet(Duration(4, 8), notetools.make_repeat_note("d'", 8))]))
abjad> piano_staff = scoretools.PianoStaff([])
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(2, Duration(1, 4))))
abjad> piano_staff.append(Staff(notetools.make_repeated_notes(4)))
abjad> contexttools.ClefMark('bass')(piano_staff[1])
ClefMark('bass')(Staff{4})
abjad> score.append(piano_staff)
abjad> pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr(list(score))
abjad> f(score)
\new Score <<
  \new Staff {
    \fraction \times 4/3 {
      d''8
      c''8
      b'8
    }
  }
  \new PianoStaff <<
    \new Staff {
      a'4
      g'4
    }
    \new Staff {
      \clef "bass"
      f'8
      e'8
      d'8
      c'8
    }
  }
>>

>>
abjad> for vertical_moment in verticalitytools.iterate_vertical_moments_forward_in_expr(score):
...     vertical_moment.leaves
...
(Note("d''8"), Note("a'4"), Note("f'8"))
(Note("d''8"), Note("a'4"), Note("e'8"))
(Note("c''8"), Note("a'4"), Note("e'8"))
```

```
(Note("c'8"), Note("g'4"), Note("d'8"))
(Note("b'8"), Note("g'4"), Note("d'8"))
(Note("b'8"), Note("g'4"), Note("c'8"))
abjad> for vertical_moment in verticalitytools.iterate_vertical_moments_forward_in_expr(piano_st
...     vertical_moment.leaves
...
(Note("a'4"), Note("f'8"))
(Note("a'4"), Note("e'8"))
(Note("g'4"), Note("d'8"))
(Note("g'4"), Note("c'8"))
```

Todo

optimize without multiple full-component traversal.

Changed in version 2.0: renamed `iterate.vertical_moments_forward_in()` to `verticalitytools.iterate_vertical_moments_forward_in_expr()`. Changed in version 2.0: renamed `iterate.vertical_moments_forward_in_expr()` to `verticalitytools.iterate_vertical_moments_forward_in_expr()`.

verticalitytools.label_vertical_moments_in_expr_with_chromatic_interval_classes

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_chromatic_interval_classes`

New in version 2.0. Label harmonic chromatic interval-classes of every vertical moment in *expr*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_chromatic_interval_classes(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 2 7 } } }
    e'8
    f'8 _ \markup { \small { \column { 5 5 } } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \small { \column { 4 5 } } }
  }
  \new Staff {
    \clef "bass"
    c,2 _ \markup { \small { \column { 12 7 } } }
  }
>>
```

Changed in version 2.0: renamed `label.vertical_moment_chromatic_interval_classes()` to `verticalitytools.label_vertical_moments_in_expr_with_chromatic_interval_classes()`.

verticalitytools.label_vertical_moments_in_expr_with_chromatic_intervals

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_chromatic_intervals`.**label**

New in version 2.0. Label harmonic chromatic intervals of every vertical moment in *expr*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_chromatic_intervals(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 26 19 } } }
    e'8
    f'8 _ \markup { \small { \column { 29 17 } } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \small { \column { 28 17 } } }
  }
  \new Staff {
    \clef "bass"
    c,2 _ \markup { \small { \column { 24 19 } } }
  }
>>
```

Changed in version 2.0: renamed `label.vertical_moment_chromatic_intervals()` to `verticalitytools.label_vertical_moments_in_expr_with_chromatic_intervals()`.

verticalitytools.label_vertical_moments_in_expr_with_counterpoint_intervals

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_counterpoint_intervals`.**label**

New in version 2.0. Label counterpoint interval of every vertical moment in *expr*:

```
abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
```

```

abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_counterpoint_intervals(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 2 5 } } }
    e'8
    f'8 _ \markup { \small { \column { 4 4 } } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \small { \column { 3 4 } } }
  }
  \new Staff {
    \clef "bass"
    c,2 _ \markup { \small { \column { 8 5 } } }
  }
>>

```

Changed in version 2.0: renamed `label.vertical_moment_counterpoint_intervals()` to `verticalitytools.label_vertical_moments_in_expr_with_counterpoint_intervals()`.

verticalitytools.label_vertical_moments_in_expr_with_diatonic_intervals

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_diatonic_intervals`.**label_v**

New in version 2.0. Label diatonic intervals of every vertical moment in *expr*:

```

abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_diatonic_intervals(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 16 12 } } }
    e'8
    f'8 _ \markup { \small { \column { 18 11 } } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \small { \column { 17 11 } } }
  }

```

```

    }
    \new Staff {
      \clef "bass"
      c,2 _ \markup { \small { \column { 15 12 } } }
    }
  >>

```

Changed in version 2.0: renamed `label.vertical_moment_diatonic_intervals()` to `verticalitytools.label_vertical_moments_in_expr_with_diatonic_intervals()`.

verticalitytools.label_vertical_moments_in_expr_with_interval_class_vectors

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_interval_class_vectors`.**label**

New in version 2.0. Label interval-class vector of every vertical moment in *expr*:

```

abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_interval_class_vectors(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \tiny { 0010020 } }
    e'8
    f'8 _ \markup { \tiny { 1000020 } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \tiny { 0100110 } }
  }
  \new Staff {
    \clef "bass"
    c,2 _ \markup { \tiny { 1000020 } }
  }
>>

```

Changed in version 2.0: renamed `label.vertical_moment_interval_class_vectors()` to `verticalitytools.label_vertical_moments_in_expr_with_interval_class_vectors()`.

verticalitytools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes`

New in version 2.0. Label pitch-classes of every vertical moment in *expr*:

```

abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 7 2 0 } } }
    e'8
    f'8 _ \markup { \small { \column { 5 0 } } }
  }
  \new Staff {
    \clef "alto"
    g4
    f4 _ \markup { \small { \column { 5 4 0 } } }
  }
  \new Staff {
    \clef "bass"
    c,2 _ \markup { \small { \column { 7 0 } } }
  }
>>

```

Changed in version 2.0: renamed `label.vertical_moment_pitch_classes()` to `verticalitytools.label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes()`

verticalitytools.label_vertical_moments_in_expr_with_pitch_numbers

`abjad.tools.verticalitytools.label_vertical_moments_in_expr_with_pitch_numbers.label_verti`

New in version 2.0. Label pitch numbers of every vertical moment in *expr*:

```

abjad> from abjad.tools import verticalitytools

abjad> score = Score(Staff([]) * 3)
abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> score[0].extend(notes)
abjad> contexttools.ClefMark('alto')(score[1])
ClefMark('alto')(Staff{})
abjad> score[1].extend([Note(-5, (1, 4)), Note(-7, (1, 4))])
abjad> contexttools.ClefMark('bass')(score[2])
ClefMark('bass')(Staff{})
abjad> score[2].append(Note(-24, (1, 2)))
abjad> verticalitytools.label_vertical_moments_in_expr_with_pitch_numbers(score)
abjad> f(score)
\new Score <<
  \new Staff {
    c'8
    d'8 _ \markup { \small { \column { 2 -5 -24 } } }
  }

```

```

        e'8
        f'8 _ \markup { \small { \column { 5 -7 -24 } } }
    }
    \new Staff {
        \clef "alto"
        g4
        f4 _ \markup { \small { \column { 4 -7 -24 } } }
    }
    \new Staff {
        \clef "bass"
        c,2 _ \markup { \small { \column { 0 -5 -24 } } }
    }
>>

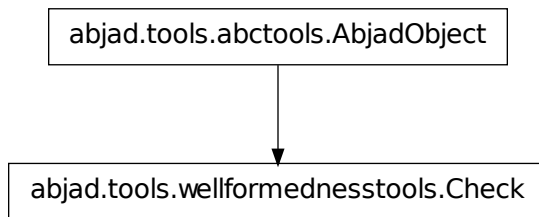
```

Changed in version 2.0: renamed `label.vertical_moment_pitch_numbers()` to `verticalitytools.label_vertical_moments_in_expr_with_pitch_numbers()`.

wellformednesstools

wellformednesstools abstract classes

wellformednesstools.Check



class `abjad.tools.wellformednesstools.Check.Check.Check`

Methods

`Check.check` (*expr*)

`Check.report` (*expr*)

`Check.violators` (*expr*)

Special Methods

`Check.__delattr__` ()

`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

Check. **__eq__** (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Check. **__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Check. **__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Check. **__hash__** () $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

Check. **__le__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Check. **__lt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Check. **__ne__** (*arg*)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

Check. **__repr__** ()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

```
Check.__setattr__ ()
x.__setattr__('name', value) <==> x.name = value
```

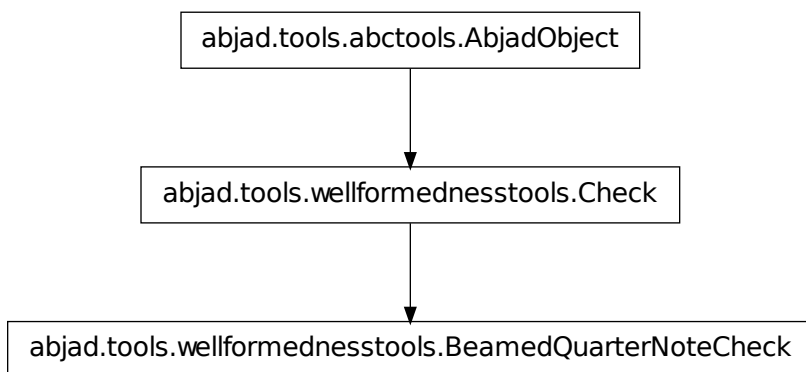
Note: Inherited from `__builtin__.object`

```
Check.__str__ () <==> str(x)
```

Note: Inherited from `__builtin__.object`

wellformednesstools concrete classes

wellformednesstools.BeamedQuarterNoteCheck



```
class abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck
```

Methods

```
BeamedQuarterNoteCheck.check (expr)
```

Note: Inherited from `wellformednesstools.Check`

```
BeamedQuarterNoteCheck.report (expr)
```

Note: Inherited from `wellformednesstools.Check`

```
BeamedQuarterNoteCheck.violators (expr)
```

Note: Inherited from `wellformednesstools.Check`

Special Methods

`BeamedQuarterNoteCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`BeamedQuarterNoteCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`BeamedQuarterNoteCheck.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`BeamedQuarterNoteCheck.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
 Return string.

Note: Inherited from `abctools.AbjadObject`

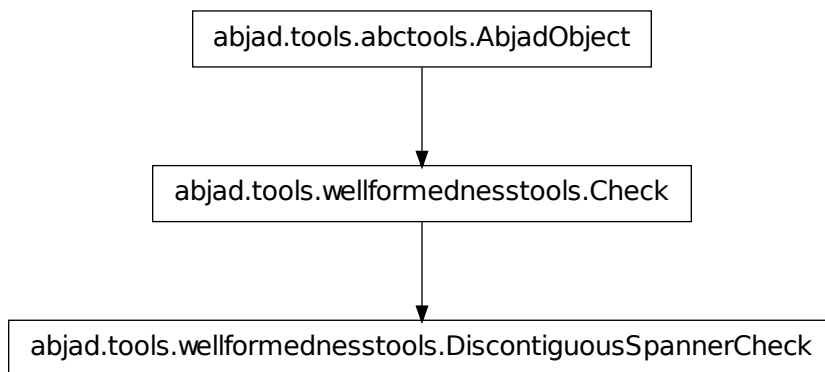
`BeamedQuarterNoteCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`BeamedQuarterNoteCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.DiscontiguousSpannerCheck



class `abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck`.
 There are now two different types of spanner. Most spanners demand that spanner components be thread-contiguous. But a few special spanners (like Tempo) do not make such a demand. The check here consults the experimental `_contiguity_constraint`.

Methods

`DiscontiguousSpannerCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`DiscontiguousSpannerCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`DiscontiguousSpannerCheck.violators(expr)`

Note: Inherited from `wellformednesstools.Check`

Special Methods

`DiscontiguousSpannerCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DiscontiguousSpannerCheck.__eq__(arg)`
True when `id(self) equals id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DiscontiguousSpannerCheck.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DiscontiguousSpannerCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

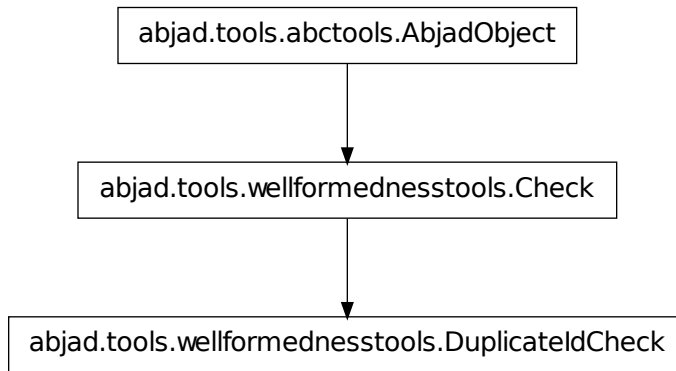
`DiscontiguousSpannerCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DiscontiguousSpannerCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.DuplicateIdCheck



class `abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck`

Methods

`DuplicateIdCheck.check` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`DuplicateIdCheck.report` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`DuplicateIdCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`DuplicateIdCheck.__delattr__` ()
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DuplicateIdCheck.__eq__` (*arg*)
 True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`DuplicateIdCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`DuplicateIdCheck.__setattr__()`

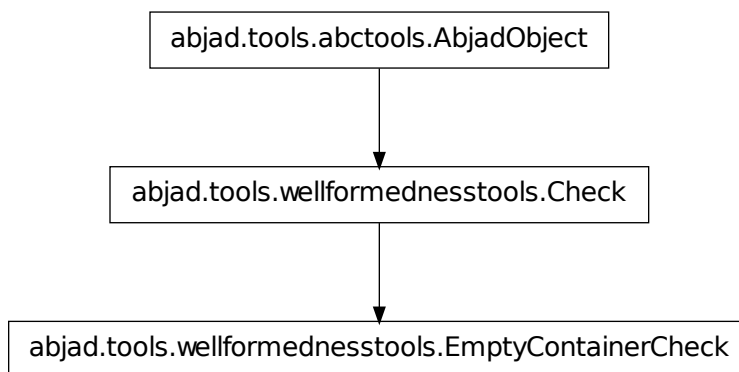
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`DuplicateIdCheck.__str__()` $\leq \Rightarrow$ `str(x)`

Note: Inherited from `__builtin__.object`

`wellformednesstools.EmptyContainerCheck`



class `abjad.tools.wellformednesstools.EmptyContainerCheck`.`EmptyContainerCheck`.**EmptyContainerCheck**

Methods

`EmptyContainerCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`EmptyContainerCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`EmptyContainerCheck.violators(expr)`

Note: Inherited from `wellformednesstools.Check`

Special Methods

```
EmptyContainerCheck.__delattr__ ()
    x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
EmptyContainerCheck.__eq__ (arg)
    True when id(self) equals id(arg).

    Return boolean.
```

Note: Inherited from `abctools.AbjadObject`

```
EmptyContainerCheck.__ge__ (arg)
    Abjad objects by default do not implement this method.

    Raise exception.
```

Note: Inherited from `abctools.AbjadObject`

```
EmptyContainerCheck.__gt__ (arg)
    Abjad objects by default do not implement this method.

    Raise exception
```

Note: Inherited from `abctools.AbjadObject`

```
EmptyContainerCheck.__hash__ () <==> hash(x)
```

Note: Inherited from `__builtin__.object`

```
EmptyContainerCheck.__le__ (arg)
    Abjad objects by default do not implement this method.

    Raise exception.
```

Note: Inherited from `abctools.AbjadObject`

```
EmptyContainerCheck.__lt__ (arg)
    Abjad objects by default do not implement this method.

    Raise exception.
```

Note: Inherited from `abctools.AbjadObject`

```
EmptyContainerCheck.__ne__ (arg)
    True when id(self) does not equal id(arg).

    Return boolean.
```

Note: Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`EmptyContainerCheck.__setattr__()`

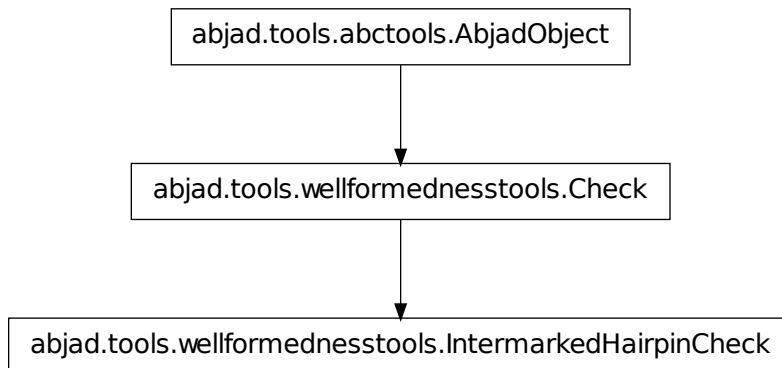
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`EmptyContainerCheck.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

`wellformednesstools.IntermarkedHairpinCheck`



class `abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck`. **Interma**

Are there any dynamic marks in the middle of a hairpin?

Methods

`IntermarkedHairpinCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`IntermarkedHairpinCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`IntermarkedHairpinCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`IntermarkedHairpinCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`IntermarkedHairpinCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__hash__()` *<==> hash(x)*

Note: Inherited from `__builtin__.object`

`IntermarkedHairpinCheck.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`IntermarkedHairpinCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

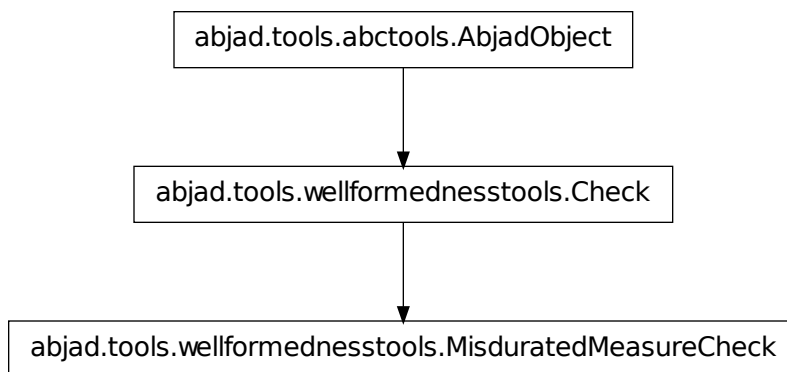
`IntermarkedHairpinCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`IntermarkedHairpinCheck.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.MisduratedMeasureCheck



class `abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck`.**MisduratedMeasureCheck**
Does the (pre)prolated duration of the measure match its meter?

Methods

MisdatedMeasureCheck.**check** (*expr*)

Note: Inherited from `wellformednesstools.Check`

MisdatedMeasureCheck.**report** (*expr*)

Note: Inherited from `wellformednesstools.Check`

MisdatedMeasureCheck.**violators** (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

MisdatedMeasureCheck.**__delattr__** ()
x.**__delattr__** ('name') <==> del x.name

Note: Inherited from `__builtin__.object`

MisdatedMeasureCheck.**__eq__** (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

MisdatedMeasureCheck.**__ge__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MisdatedMeasureCheck.**__gt__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

MisdatedMeasureCheck.**__hash__** () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

MisdatedMeasureCheck.**__le__** (*arg*)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`MisdatedMeasureCheck.__setattr__()`

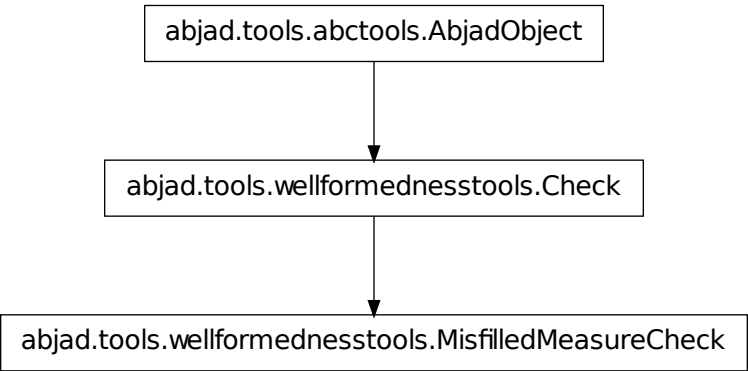
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MisdatedMeasureCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.MisfilledMeasureCheck



class `abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck`.**MisfilledMeasureCheck**
For each (rigid) measure, does effective meter duration equal preprolotted duration?

Methods

`MisfilledMeasureCheck.check` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`MisfilledMeasureCheck.report` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`MisfilledMeasureCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`MisfilledMeasureCheck.__delattr__` ()
`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`MisfilledMeasureCheck.__eq__` (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`MisfilledMeasureCheck.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`MisfilledMeasureCheck.__setattr__()`

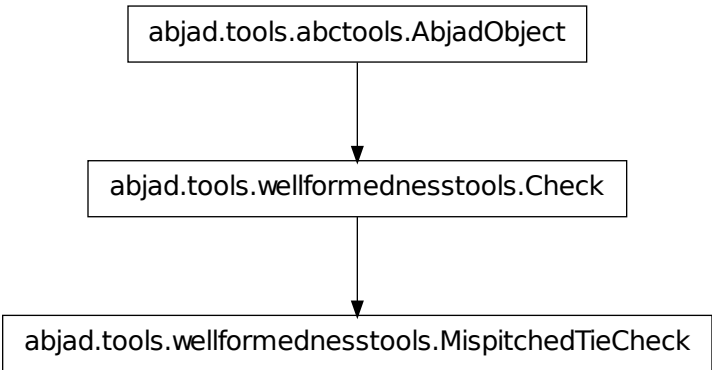
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MisfilledMeasureCheck.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.MispitchedTieCheck



class `abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck`.**MispitchedTieCheck**

Methods

`MispitchedTieCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`MispitchedTieCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`MispitchedTieCheck.violators(expr)`

Note: Inherited from `wellformednesstools.Check`

Special Methods

MispitchedTieCheck.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

MispitchedTieCheck.__eq__(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

MispitchedTieCheck.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MispitchedTieCheck.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

MispitchedTieCheck.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

MispitchedTieCheck.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MispitchedTieCheck.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MispitchedTieCheck.__ne__(arg)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`MispitchedTieCheck.__setattr__()`

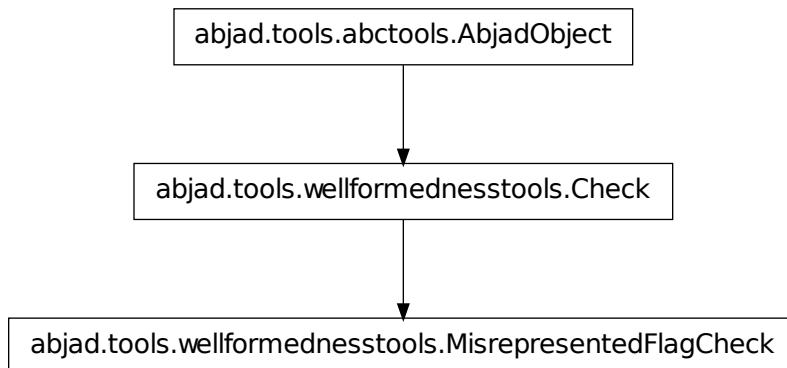
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MispitchedTieCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`wellformednesstools.MisrepresentedFlagCheck`



class `abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck`. **Misrepr**

Methods

`MisrepresentedFlagCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`MisrepresentedFlagCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

MisrepresentedFlagCheck.**violators** (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

MisrepresentedFlagCheck.**__delattr__** ()
x.**__delattr__**('name') <==> del x.name

Note: Inherited from `__builtin__.object`

MisrepresentedFlagCheck.**__eq__** (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

MisrepresentedFlagCheck.**__ge__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MisrepresentedFlagCheck.**__gt__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

MisrepresentedFlagCheck.**__hash__** () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

MisrepresentedFlagCheck.**__le__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MisrepresentedFlagCheck.**__lt__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MisrepresentedFlagCheck.__repr__()`
 Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
 Return string.

Note: Inherited from `abctools.AbjadObject`

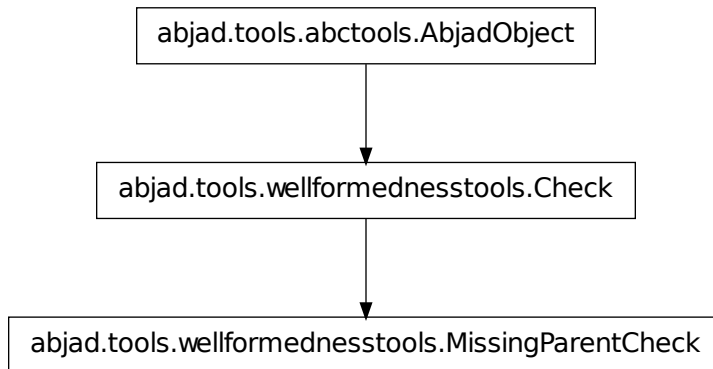
`MisrepresentedFlagCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MisrepresentedFlagCheck.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.MissingParentCheck



class `abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck`.**MissingParentCheck**
 Each node except the root needs a parent. Each node needs the correct parent.

Methods

MissingParentCheck.**check** (*expr*)

Note: Inherited from `wellformednesstools.Check`

MissingParentCheck.**report** (*expr*)

Note: Inherited from `wellformednesstools.Check`

MissingParentCheck.**violators** (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

MissingParentCheck.**__delattr__** ()
x.**__delattr__** ('name') <==> del x.name

Note: Inherited from `__builtin__.object`

MissingParentCheck.**__eq__** (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

MissingParentCheck.**__ge__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

MissingParentCheck.**__gt__** (*arg*)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

MissingParentCheck.**__hash__** () <==> *hash(x)*

Note: Inherited from `__builtin__.object`

MissingParentCheck.**__le__** (*arg*)
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MissingParentCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`MissingParentCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`MissingParentCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

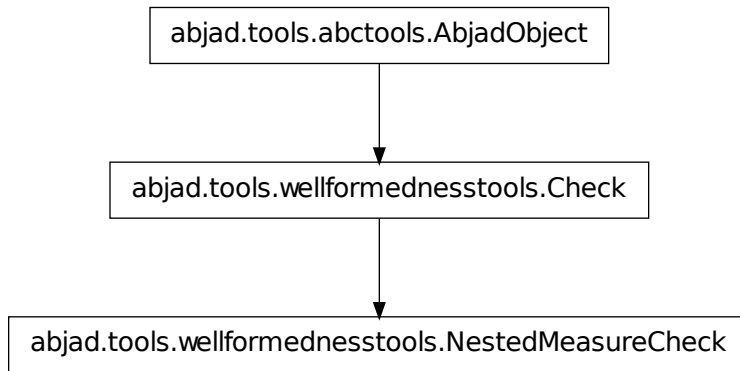
`MissingParentCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`MissingParentCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.NestedMeasureCheck



class `abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck`. **NestedMeasureCheck**
 Do we have any nested measures?

Methods

`NestedMeasureCheck.check` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`NestedMeasureCheck.report` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`NestedMeasureCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`NestedMeasureCheck.__delattr__` ()
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`NestedMeasureCheck.__eq__` (*arg*)
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`NestedMeasureCheck.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`NestedMeasureCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

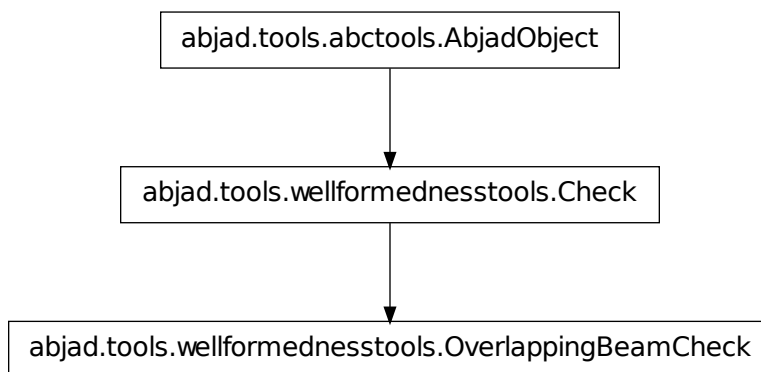
`NestedMeasureCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`NestedMeasureCheck.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

`wellformednesstools.OverlappingBeamCheck`



class `abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck`. **OverlappingBeamCheck**
Beams must not overlap.

Methods

`OverlappingBeamCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingBeamCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingBeamCheck.violators(expr)`

Note: Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingBeamCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OverlappingBeamCheck.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`OverlappingBeamCheck.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`OverlappingBeamCheck.__setattr__()`

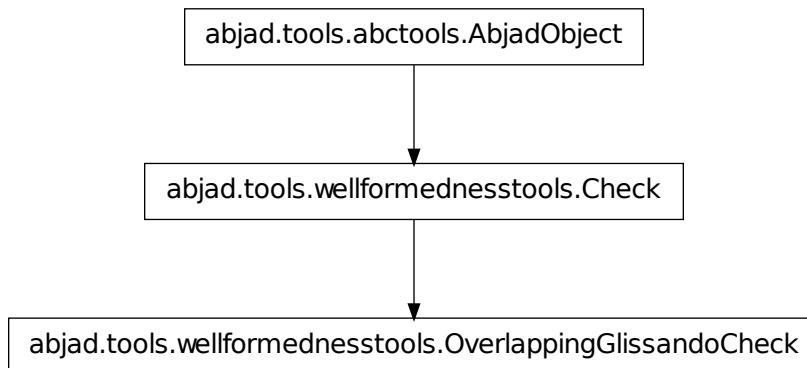
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`OverlappingBeamCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`wellformednesstools.OverlappingGlissandoCheck`



class `abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck`. **OverlappingGlissandoCheck**
 Glissandi must not overlap. Dove-tailed glissandi are OK.

Methods

`OverlappingGlissandoCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingGlissandoCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingGlissandoCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingGlissandoCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OverlappingGlissandoCheck.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`OverlappingGlissandoCheck.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingGlissandoCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

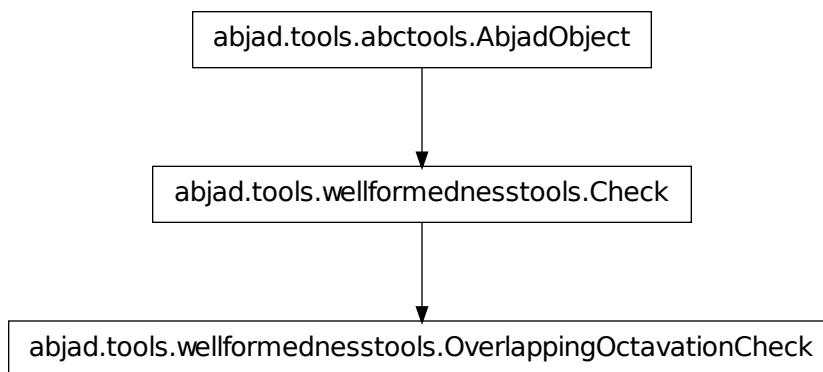
`OverlappingGlissandoCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`OverlappingGlissandoCheck.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.OverlappingOctavationCheck



class `abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.C`
Octavation spanners must not overlap.

Methods

`OverlappingOctavationCheck.check(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingOctavationCheck.report(expr)`

Note: Inherited from `wellformednesstools.Check`

`OverlappingOctavationCheck.violators(expr)`

Note: Inherited from `wellformednesstools.Check`

Special Methods

`OverlappingOctavationCheck.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OverlappingOctavationCheck.__eq__(arg)`
True when `id(self) equals id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`OverlappingOctavationCheck.__le__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__lt__(arg)`
Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OverlappingOctavationCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

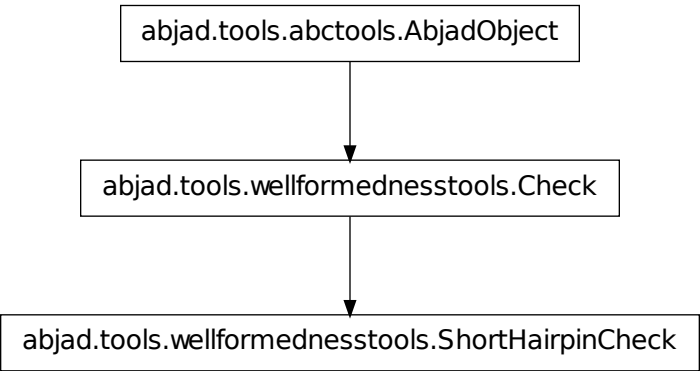
`OverlappingOctavationCheck.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`OverlappingOctavationCheck.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools.ShortHairpinCheck



class `abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck`. **ShortHairpinCheck**
Hairpins must span at least two leaves.

Methods

`ShortHairpinCheck.check` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`ShortHairpinCheck.report` (*expr*)

Note: Inherited from `wellformednesstools.Check`

`ShortHairpinCheck.violators` (*expr*)

Note: Inherited from `wellformednesstools.Check`

Special Methods

`ShortHairpinCheck.__delattr__` ()
`x.__delattr__('name')` <==> `del x.name`

Note: Inherited from `__builtin__.object`

`ShortHairpinCheck.__eq__` (*arg*)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__hash__()` $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

`ShortHairpinCheck.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__repr__()`
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

Note: Inherited from `abctools.AbjadObject`

`ShortHairpinCheck.__setattr__()`
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`ShortHairpinCheck.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

wellformednesstools functions

wellformednesstools.list_checks

`abjad.tools.wellformednesstools.list_checks.list_checks()`

New in version 2.8. List checks:

```
abjad> from abjad.tools import wellformednesstools

abjad> for check in wellformednesstools.list_checks(): check
...
BeamedQuarterNoteCheck()
DiscontiguousSpannerCheck()
DuplicateIdCheck()
EmptyContainerCheck()
IntermarkedHairpinCheck()
MisduratedMeasureCheck()
MisfilledMeasureCheck()
MispitchedTieCheck()
MisrepresentedFlagCheck()
MissingParentCheck()
NestedMeasureCheck()
OverlappingBeamCheck()
OverlappingGlissandoCheck()
OverlappingOctavationCheck()
ShortHairpinCheck()
```

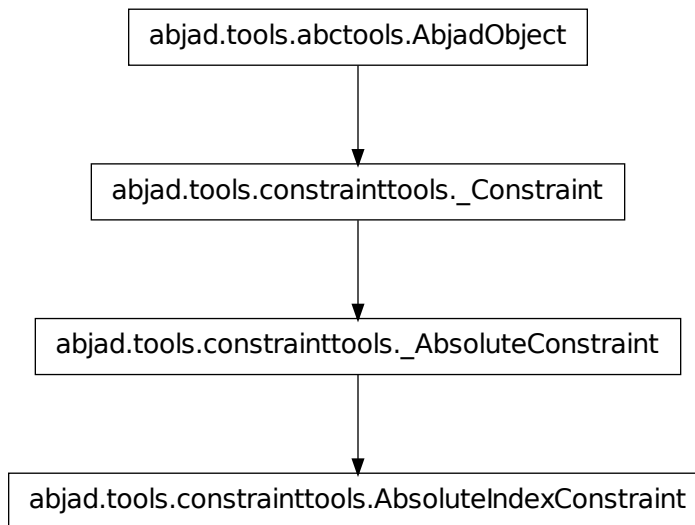
Return list of checks.

51.1.3 Unstable Abjad composition packages (load manually)

`constrainttools`

`constrainttools` concrete classes

`constrainttools.AbsoluteIndexConstraint`



```
class abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint
```

A constraint for an absolutely positioned item or group of items in a solution:

```
abjad> from abjad.tools.constrainttools import AbsoluteIndexConstraint
```

Instantiated from an index, or sequence of indices, and a function which takes as many arguments as indices were given:

```
abjad> first_is_zero = AbsoluteIndexConstraint(0, lambda x: x == 0)
abjad> third_greater_than_second = AbsoluteIndexConstraint([1, 2], lambda x, y: x < y)

abjad> first_is_zero([0, 1, 2])
True
abjad> first_is_zero([1, 12, 3, 4, 5])
False

abjad> third_greater_than_second([0, 1, 2])
True
abjad> third_greater_than_second([1, 12, 3, 4, 5])
False
```

`AbsoluteIndexConstraints` are immutable.

Returns `AbsoluteIndexConstraint` instance.

Read-only Properties

`AbsoluteIndexConstraint.indices`

Note: Inherited from `constrainttools._AbsoluteConstraint`

`AbsoluteIndexConstraint.kind`

Note: Inherited from `constrainttools._Constraint`

`AbsoluteIndexConstraint.max_index`

Note: Inherited from `constrainttools._AbsoluteConstraint`

`AbsoluteIndexConstraint.predicate`

Note: Inherited from `constrainttools._Constraint`

Special Methods

`AbsoluteIndexConstraint.__call__(solution)`

`AbsoluteIndexConstraint.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`AbsoluteIndexConstraint.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__hash__()` $\leq \Rightarrow \text{hash}(x)$

Note: Inherited from `__builtin__.object`

`AbsoluteIndexConstraint.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AbsoluteIndexConstraint.__repr__()`

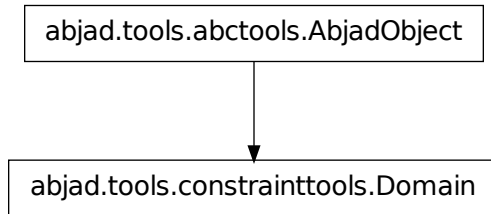
Note: Inherited from `constrainttools._Constraint`

`AbsoluteIndexConstraint.__setattr__()`
`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

`AbsoluteIndexConstraint.__str__()` $\leq \Rightarrow \text{str}(x)$

Note: Inherited from `__builtin__.object`

constrainttools.Domain

class `abjad.tools.constrainttools.Domain.Domain` *(*args)*

A two-dimensional constraints search domain:

```
abjad> from abjad.tools.constrainttools import Domain
```

May be instantiated from a non-empty sequence of non-empty sequences.

```
abjad> domain = Domain([(1, 2, 3), (4, 5, 6), (7, 8, 9)])
```

May also be instantiated from one non-empty sequence and an integer greater than zero, indicating how many “columns” to create from the first sequence:

```
abjad> domain = Domain([1, 2, 3, 4], 5)
```

Domains are immutable.

Returns `Domain` instance.

Methods

`Domain.randomized()`

Create a new `Domain` containing the same values, but with every column randomly reordered:

```
abjad> original = Domain([1, 2, 3, 4], 4)
abjad> randomized = original.randomized()
abjad> randomized[0] # doctest: +SKIP
(4, 1, 2, 3) # doctest: +SKIP
```

Returns `Domain` instance.

Special Methods

`Domain.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Domain.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Domain.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Domain.__getitem__(item)`

`Domain.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Domain.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Domain.__iter__()`

`Domain.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Domain.__len__()`

`Domain.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Domain.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Domain.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

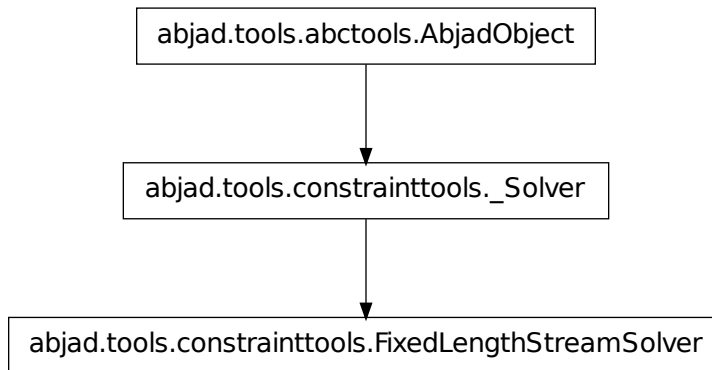
```
Domain.__setattr__ ()
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
Domain.__str__ () <==> str(x)
```

Note: Inherited from `__builtin__.object`

constrainttools.FixedLengthStreamSolver



```
class abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver
```

Recursive tree-traversal-based finite-domain constraints solver:

```
abjad> from abjad.tools.constrainttools import FixedLengthStreamSolver
abjad> from abjad.tools.constrainttools import Domain
abjad> from abjad.tools.constrainttools import GlobalCountsConstraint
abjad> from abjad.tools.constrainttools import RelativeIndexConstraint
```

Instantiates from a `Domain`, and a sequence of `Constraints`.

```
abjad> domain = Domain([1, 2, 3, 4], 4)
abjad> all_unique = GlobalCountsConstraint(lambda x: all([y == 1 for y in x.values()]))
abjad> max_interval = RelativeIndexConstraint([0, 1], lambda x, y: abs(x - y) < 3)
abjad> solver = FixedLengthStreamSolver(domain, [all_unique, max_interval])
```

Generate solutions by iterating over the `FixedLengthStreamSolver`.

```
abjad> for solution in solver: print solution
...
(1, 2, 3, 4)
(1, 2, 4, 3)
(1, 3, 2, 4)
(1, 3, 4, 2)
(2, 1, 3, 4)
(2, 4, 3, 1)
(3, 1, 2, 4)
(3, 4, 2, 1)
(4, 2, 1, 3)
(4, 2, 3, 1)
(4, 3, 1, 2)
(4, 3, 2, 1)
```

If no solutions can be found, returns none:

```
abjad> domain = Domain([1, 2, 3, 4], 100)
abjad> solver = FixedLengthStreamSolver(domain, [all_unique])
abjad> [x for x in solver]
[]
```

Can be instantiated with boolean `randomized` keyword, in order to randomize the domain on each iteration run:

```
abjad> random_solver = FixedLengthStreamSolver(domain, [all_unique, max_interval], randomized=True)
```

`FixedLengthStreamSolvers` are immutable.

Returns `FixedLengthStreamSolver` instance.

Read-only Properties

`FixedLengthStreamSolver.constraints`

Note: Inherited from `constrainttools._Solver`

`FixedLengthStreamSolver.domain`

Note: Inherited from `constrainttools._Solver`

`FixedLengthStreamSolver.iterator`

Note: Inherited from `constrainttools._Solver`

`FixedLengthStreamSolver.randomized`

Note: Inherited from `constrainttools._Solver`

`FixedLengthStreamSolver.solutions`

Note: Inherited from `constrainttools._Solver`

Methods

`FixedLengthStreamSolver.get_le_n_solutions(n)`

Note: Inherited from `constrainttools._Solver`

Special Methods

`FixedLengthStreamSolver.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`FixedLengthStreamSolver.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`FixedLengthStreamSolver.__iter__()`

`FixedLengthStreamSolver.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`FixedLengthStreamSolver.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

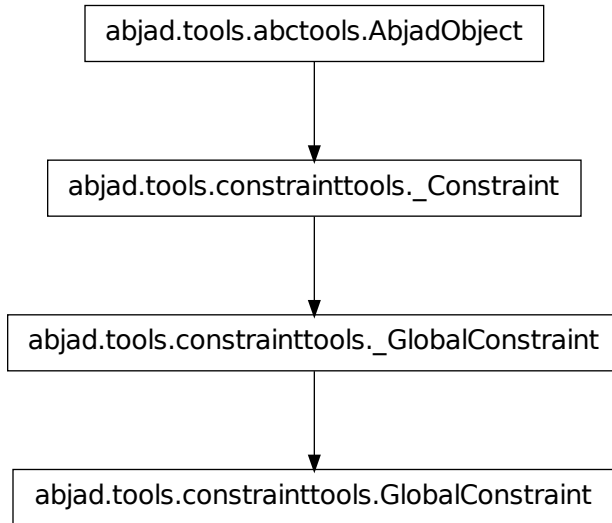
`FixedLengthStreamSolver.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`FixedLengthStreamSolver.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

constrainttools.GlobalConstraint

class `abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint` (*predicate*)
 A constraint applied against an entire solution:

```
abjad> from abjad.tools.constrainttools import GlobalConstraint
```

Instantiated from a function which takes a single argument, representing an entire solution.

```
abjad> max_total_range = GlobalConstraint(lambda seq: (max(seq) - min(seq)) < 5)

abjad> max_total_range([0, 1, 2])
True
abjad> max_total_range([0, 1, 2, 3])
True
abjad> max_total_range([0, 1, 2, 3, 4])
True
abjad> max_total_range([0, 1, 2, 3, 4, 5])
False
```

`GlobalConstraints` are immutable.

Returns `GlobalConstraint` instance.

Read-only Properties

`GlobalConstraint.kind`

Note: Inherited from `constrainttools._Constraint`

`GlobalConstraint.predicate`

Note: Inherited from `constrainttools._Constraint`

Special Methods

`GlobalConstraint.__call__(solution)`

`GlobalConstraint.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`GlobalConstraint.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`GlobalConstraint.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__ne__(arg)`
 True when `id(self)` does not equal `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlobalConstraint.__repr__()`

Note: Inherited from `constrainttools._Constraint`

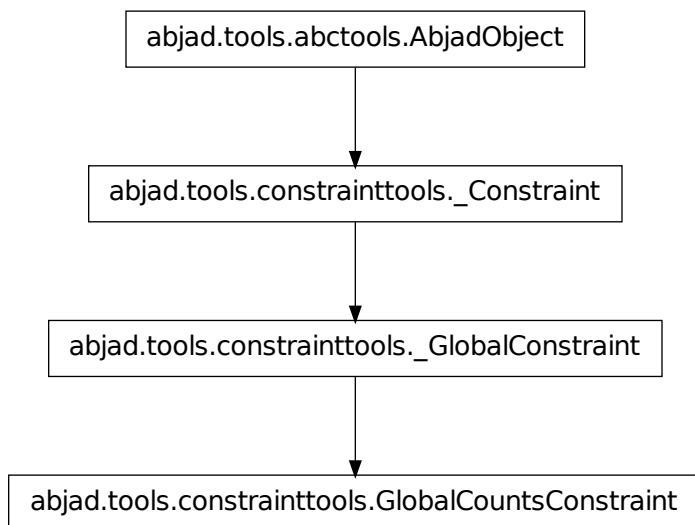
`GlobalConstraint.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`GlobalConstraint.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

`constrainttools.GlobalCountsConstraint`



class `abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint`
 A constraint which is applied against a dictionary whose keys are the values in a given solution, and whose values are the counts of those keys:

```
abjad> from abjad.tools.constrainttools import GlobalCountsConstraint
```

Instantiated from a lambda or function which takes a dictionary as its one and only argument:

```
abjad> all_unique = GlobalCountsConstraint(lambda x: all([y == 1 for y in x.values()]))
abjad> one_climax = GlobalCountsConstraint(lambda x: x[max(x.keys())] == 1)
```

```
abjad> all_unique([1, 2, 3, 4, 0, -1])
True
abjad> all_unique([1, 1, 2, 3, 4, 0, -1])
False
```

```
abjad> one_climax([1, 2, 3, 4, 0, -1])
True
abjad> one_climax([1, 5, 5, 3])
False
```

GlobalCountsConstraints are immutable.

Returns GlobalCountsConstraint instance.

Read-only Properties

GlobalCountsConstraint.**kind**

Note: Inherited from `constrainttools._Constraint`

GlobalCountsConstraint.**predicate**

Note: Inherited from `constrainttools._Constraint`

Special Methods

GlobalCountsConstraint.**__call__**(*solution*)
GlobalCountsConstraint.**__delattr__**()
 `x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

GlobalCountsConstraint.**__eq__**(*arg*)
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

GlobalCountsConstraint.**__ge__**(*arg*)
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalCountsConstraint.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`GlobalCountsConstraint.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`GlobalCountsConstraint.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalCountsConstraint.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalCountsConstraint.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlobalCountsConstraint.__repr__()`

Note: Inherited from `constrainttools._Constraint`

`GlobalCountsConstraint.__setattr__()`

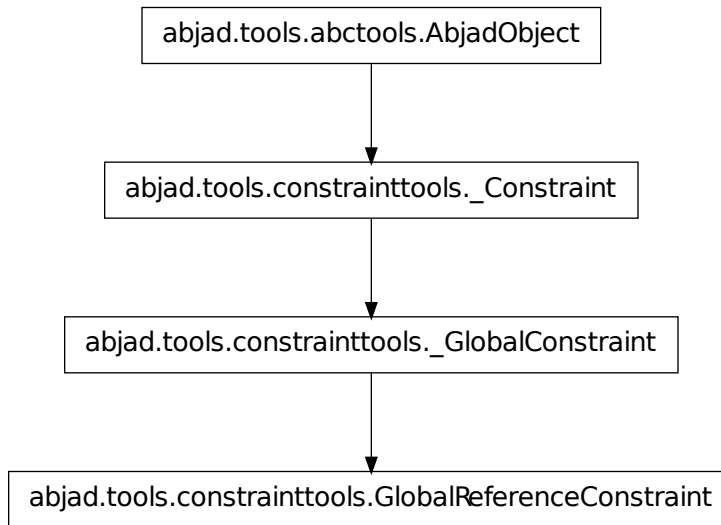
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`GlobalCountsConstraint.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

constrainttools.GlobalReferenceConstraint



```
class abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint(GlobalReferenceConstraint
```

A global constraint with an arbitrary external reference:

```
abjad> from abjad.tools.constrainttools import GlobalReferenceConstraint
```

Its predicate function should accept two arguments, the first being the current solution, and the second for the external reference provided at instantiation.

```
abjad> reference = [0, 1, 2, 3]
abjad> predicate = lambda solution, reference: all([item not in reference for item in solution])
abjad> constraint = GlobalReferenceConstraint(reference, predicate)
```

```
abjad> constraint([-1, 10, 3.5])
True
abjad> constraint([-1, 1, 2, 3, 23])
False
```

GlobalReferenceConstraints are immutable.

Returns GlobalReferenceConstraint instance.

Read-only Properties

GlobalReferenceConstraint.**kind**

Note: Inherited from constrainttools._Constraint

`GlobalReferenceConstraint.predicate`

Note: Inherited from `constrainttools._Constraint`

`GlobalReferenceConstraint.reference`

Special Methods

`GlobalReferenceConstraint.__call__(solution)`

`GlobalReferenceConstraint.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`GlobalReferenceConstraint.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`GlobalReferenceConstraint.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`GlobalReferenceConstraint.__repr__()`

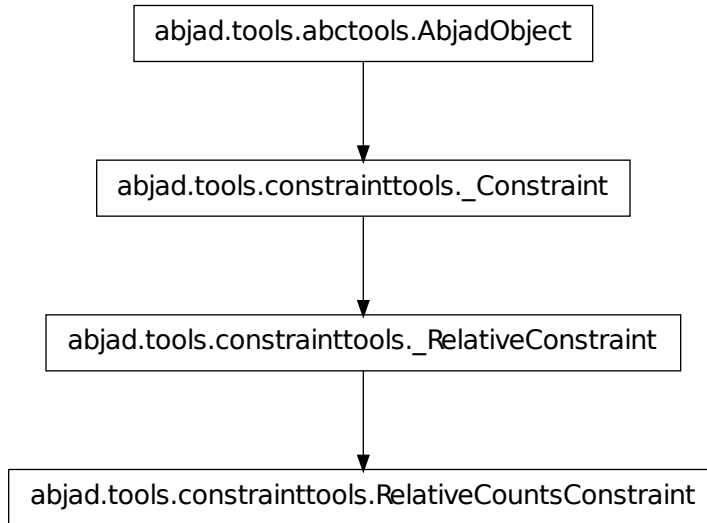
Note: Inherited from `constrainttools._Constraint`

`GlobalReferenceConstraint.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`GlobalReferenceConstraint.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

constrainttools.RelativeCountsConstraint

class `abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint`

A constraint which is applied against a dictionary whose keys are the values in a set of items from a given solution, and whose values are the counts of those keys. The constraint is always applied against the last possible group of items, and returns True automatically if the solution is of insufficient length.

```
abjad> from abjad.tools.constrainttools import RelativeCountsConstraint
```

Instantiated from an integer representing a contiguous index range, or a sequence of indices, and a function which takes as many arguments as indices were given:

```
abjad> test = lambda x: max(x.values( )) <= 2
abjad> two_repeats_max = RelativeCountsConstraint([0, 1, 2], test)
abjad> two_repeats_max([0])
True
abjad> two_repeats_max([0, 1, 2])
True
abjad> two_repeats_max([0, 0, 1])
True
abjad> two_repeats_max([0, 0, 0])
False
```

RelativeCountsConstraints are immutable.

Returns RelativeCountsConstraint instance.

Read-only Properties

RelativeCountsConstraint.**index_span**

Note: Inherited from `constrainttools._RelativeConstraint`

RelativeCountsConstraint.**indices**

Note: Inherited from `constrainttools._RelativeConstraint`

RelativeCountsConstraint.**kind**

Note: Inherited from `constrainttools._Constraint`

RelativeCountsConstraint.**predicate**

Note: Inherited from `constrainttools._RelativeConstraint`

Special Methods

RelativeCountsConstraint.**__call__** (*solution*)

RelativeCountsConstraint.**__delattr__** ()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

RelativeCountsConstraint.**__eq__** (*arg*)

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

RelativeCountsConstraint.**__ge__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

RelativeCountsConstraint.**__gt__** (*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

RelativeCountsConstraint.**__hash__** () *<==> hash(x)*

Note: Inherited from `__builtin__.object`

`RelativeCountsConstraint.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RelativeCountsConstraint.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RelativeCountsConstraint.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`RelativeCountsConstraint.__repr__()`

Note: Inherited from `constrainttools._Constraint`

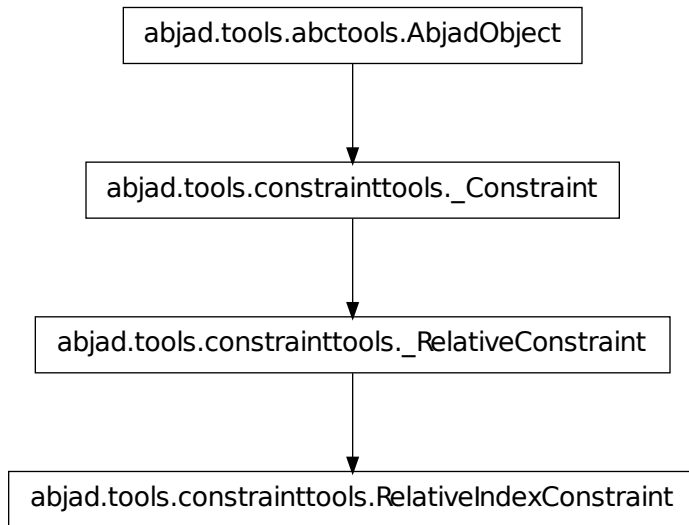
`RelativeCountsConstraint.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`RelativeCountsConstraint.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

constrainttools.RelativeIndexConstraint



```
class abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint
```

A constraint for a relatively positioned group of items, i.e. every adjacent pair of two items. The constraint is applied against the last possible grouping in a solution, with the understanding that it has been applied previously during each stage of the construction of that solution.

```
abjad> from abjad.tools.constrainttools import RelativeIndexConstraint
```

Instantiated from an integer representing a contiguous index range, or a sequence of indices, and a function which takes as many arguments as indices were given:

```
abjad> max_interval = RelativeIndexConstraint(2, lambda x, y: abs(x - y) < 3)
```

The above constraint will only be applied against the last two items of any solution.

```
abjad> max_interval([0, 2])
True
abjad> max_interval([1000, 0, 1])
True
abjad> max_interval([0, 0, 0, 0, 4])
False
```

RelativeIndexConstraints are immutable.

Returns RelativeIndexConstraint instance.

Read-only Properties

`RelativeIndexConstraint.index_span`

Note: Inherited from `constrainttools._RelativeConstraint`

`RelativeIndexConstraint.indices`

Note: Inherited from `constrainttools._RelativeConstraint`

`RelativeIndexConstraint.kind`

Note: Inherited from `constrainttools._Constraint`

`RelativeIndexConstraint.predicate`

Note: Inherited from `constrainttools._RelativeConstraint`

Special Methods

`RelativeIndexConstraint.__call__(solution)`

`RelativeIndexConstraint.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`RelativeIndexConstraint.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`RelativeIndexConstraint.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`RelativeIndexConstraint.__repr__()`

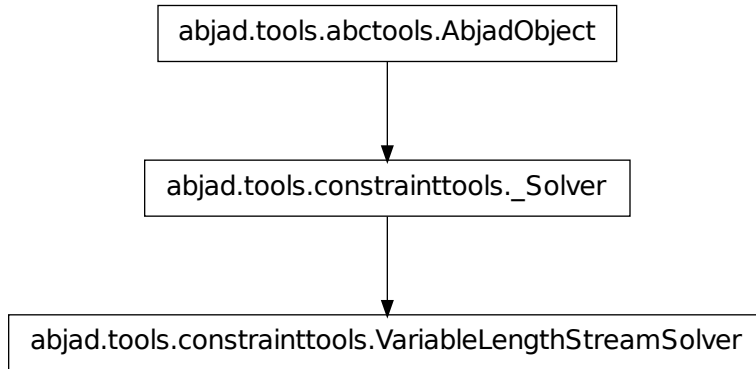
Note: Inherited from `constrainttools._Constraint`

`RelativeIndexConstraint.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`RelativeIndexConstraint.__str__()` `<==> str(x)`

Note: Inherited from `__builtin__.object`

constrainttools.VariableLengthStreamSolver

class `abjad.tools.constrainttools.VariableLengthStreamSolver`.`VariableLengthStreamSolver`.`VariableLengthStreamSolver`

A solver which behaves similarly to the `FiniteStreamSolver` except that it can produce solutions of variable rather than fixed lengths.

Instantiated from a `Domain`, a list of `Constraints`, and list of terminating constraints, which signal when to begin yielding solutions.

This solver is best suited for generating solutions which sum to a fixed value, or whose solutions fall within a range of lengths.

A poorly defined set of constraints can trigger infinite recursion, so care must be taken when constructing the problem to be explored.

```

abjad> from abjad.tools.constrainttools import *
abjad> domain = Domain([1, 2, 3, 4], 1)
abjad> target_sum = GlobalConstraint(lambda x: sum(x) == 5)
abjad> boundary_sum = GlobalConstraint(lambda x: sum(x) < 6)
abjad> solver = VariableLengthStreamSolver(domain, [boundary_sum], [target_sum])
abjad> for x in solver: x
...
(1, 1, 1, 1, 1)
(1, 1, 1, 2)
(1, 1, 2, 1)
(1, 1, 3)
(1, 2, 1, 1)
(1, 2, 2)
(1, 3, 1)

```

```
(1, 4)
(2, 1, 1, 1)
(2, 1, 2)
(2, 2, 1)
(2, 3)
(3, 1, 1)
(3, 2)
(4, 1)
```

VariableLengthStreamSolvers are immutable.

Returns VariableLengthStreamSolver instance.

Read-only Properties

VariableLengthStreamSolver.**constraints**

Note: Inherited from constrainttools._Solver

VariableLengthStreamSolver.**domain**

Note: Inherited from constrainttools._Solver

VariableLengthStreamSolver.**iterator**

Note: Inherited from constrainttools._Solver

VariableLengthStreamSolver.**randomized**

Note: Inherited from constrainttools._Solver

VariableLengthStreamSolver.**solutions**

Note: Inherited from constrainttools._Solver

VariableLengthStreamSolver.**terminators**

Methods

VariableLengthStreamSolver.**get_le_n_solutions**(*n*)

Note: Inherited from constrainttools._Solver

Special Methods

VariableLengthStreamSolver.__delattr__()
x.__delattr__('name') <==> del x.name

Note: Inherited from `__builtin__.object`

VariableLengthStreamSolver.__eq__(arg)
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

VariableLengthStreamSolver.__ge__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

VariableLengthStreamSolver.__gt__(arg)
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

VariableLengthStreamSolver.__hash__() <==> *hash(x)*

Note: Inherited from `__builtin__.object`

VariableLengthStreamSolver.__iter__()
VariableLengthStreamSolver.__le__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

VariableLengthStreamSolver.__lt__(arg)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

VariableLengthStreamSolver.__ne__(arg)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`VariableLengthStreamSolver.__repr__()`

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

`VariableLengthStreamSolver.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

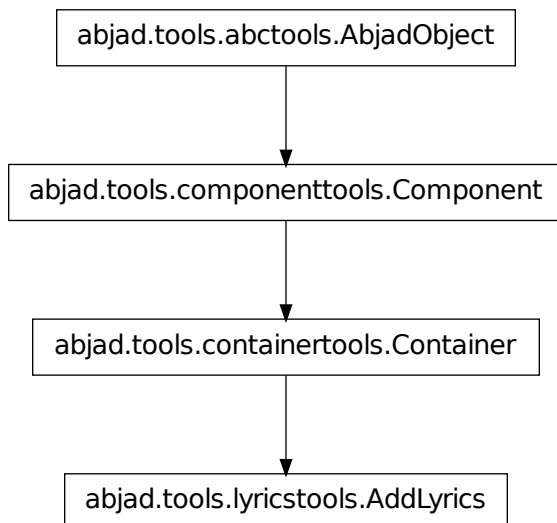
`VariableLengthStreamSolver.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

lyricstools

lyricstools concrete classes

lyricstools.AddLyrics



```
class abjad.tools.lyricstools.AddLyrics.AddLyrics(music=None,  
                                              **kwargs)
```

Read-only Properties**AddLyrics.contents_duration**

Note: Inherited from `containertools.Container`

AddLyrics.duration_in_seconds

Note: Inherited from `containertools.Container`

AddLyrics.format

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

AddLyrics.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

AddLyrics.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

AddLyrics.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.music
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

AddLyrics.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

AddLyrics.parent

Note: Inherited from `componenttools.Component`

`AddLyrics.preprolated_duration`

Note: Inherited from `containertools.Container`

`AddLyrics.prolated_duration`

Note: Inherited from `componenttools.Component`

`AddLyrics.prolation`

Note: Inherited from `componenttools.Component`

`AddLyrics.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`AddLyrics.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`AddLyrics.is_parallel`

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])
```

```
abjad> f(container)
```

```
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
abjad> container.is_parallel
```

```
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True
```

```
abjad> f(container)
```

```
<<
    \new Voice {
      c'8
      d'8
      e'8
    }
    \new Voice {
      g4.
    }
>>
```

Return none.

Note: Inherited from `containertools.Container`

Methods

`AddLyrics.append(component)`

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
```

```
{
  c'8 [
  d'8
  e'8 ]
}
```

```
abjad> container.append(Note("f'8"))
```

```
abjad> f(container)
```

```
{
  c'8 [
  d'8
  e'8 ]
  f'8
}
```

Return none.

Note: Inherited from `containertools.Container`

`AddLyrics.extend(expr)`

Extend *expr* against container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```
abjad> f(container)
```

```
{
```

```
        c'8 [
        d'8
        e'8 ]
    }

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}
```

Return none. New in version 2.3: `expr` may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`AddLyrics.index` (*component*)

Index *component* in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2
```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`AddLyrics.insert` (*i*, *component*)

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
```

```
        d'8
        e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

`AddLyrics.pop(i=-1)`

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

`AddLyrics.remove(component)`

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
```

```
        d'8 ]
    }
```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`AddLyrics.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new Container c with the content of both a and b. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`AddLyrics.__contains__(expr)`

True if expr is in container, otherwise False.

Note: Inherited from `containertools.Container`

`AddLyrics.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`AddLyrics.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`AddLyrics.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AddLyrics.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`AddLyrics.__getitem__(i)`

Return component at index i in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

AddLyrics.__gt__(arg)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

AddLyrics.__hash__() \Leftrightarrow *hash(x)*

Note: Inherited from `__builtin__.object`

AddLyrics.__iadd__(expr)

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

AddLyrics.__imul__(total)

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

AddLyrics.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AddLyrics.__len__()

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

AddLyrics.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

AddLyrics.__mul__(n)

Note: Inherited from `componenttools.Component`

AddLyrics.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`AddLyrics.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`AddLyrics.__repr__()`

String format of container for interpreter display.

Note: Inherited from `containertools.Container`

`AddLyrics.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`AddLyrics.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

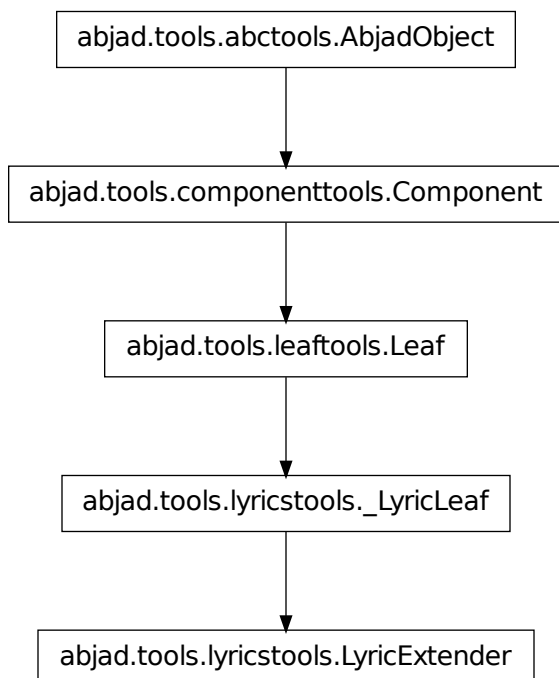
`AddLyrics.__setitem__(i, expr)`

Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

`AddLyrics.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

lyricstools.LyricExtender

class `abjad.tools.lyricstools.LyricExtender.LyricExtender`(*written_duration*,
duration_multiplier=None)

Read-only Properties

`LyricExtender.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.format`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.leaf_index`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`LyricExtender.multiplied_duration`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`LyricExtender.parent`

Note: Inherited from `componenttools.Component`

`LyricExtender.preprolated_duration`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.prolated_duration`

Note: Inherited from `componenttools.Component`

`LyricExtender.prolation`

Note: Inherited from `componenttools.Component`

`LyricExtender.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`LyricExtender.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`LyricExtender.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.written_duration`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

`LyricExtender.__and__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LyricExtender.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`LyricExtender.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__mul__(n)`

Note: Inherited from `componenttools.Component`

`LyricExtender.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricExtender.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.__repr__()`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`LyricExtender.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`LyricExtender.__str__()`

Note: Inherited from `leaftools.Leaf`

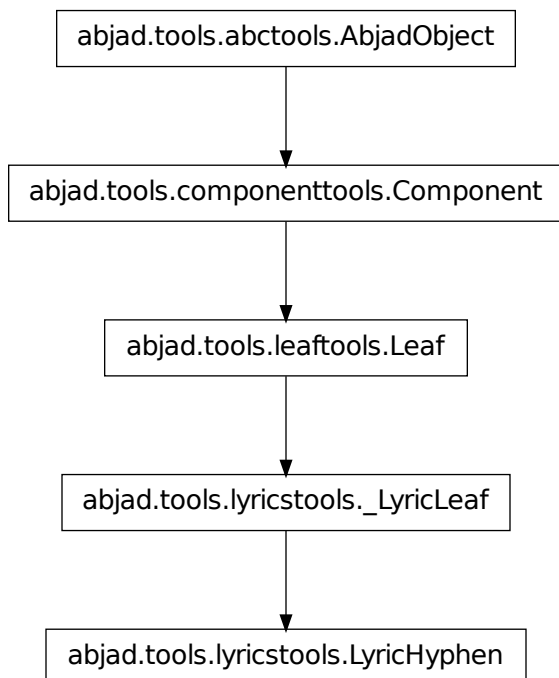
`LyricExtender.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricExtender.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

`lyricstools.LyricHyphen`



class abjad.tools.lyricstools.LyricHyphen.LyricHyphen.**LyricHyphen** (*written_duration*,
duration_multiplier=None)

Read-only Properties

LyricHyphen.**duration_in_seconds**

Note: Inherited from `leaftools.Leaf`

LyricHyphen.**format**

Note: Inherited from `leaftools.Leaf`

LyricHyphen.**leaf_index**

Note: Inherited from `leaftools.Leaf`

LyricHyphen.**marks**

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

LyricHyphen.**multiplied_duration**

Note: Inherited from `leaftools.Leaf`

LyricHyphen.**override**

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

LyricHyphen.**parent**

Note: Inherited from `componenttools.Component`

LyricHyphen.**preprolated_duration**

Note: Inherited from `leaftools.Leaf`

LyricHyphen.**prolated_duration**

Note: Inherited from `componenttools.Component`

`LyricHyphen.prolation`

Note: Inherited from `componenttools.Component`

`LyricHyphen.set`

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

`LyricHyphen.spanners`

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`LyricHyphen.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.written_duration`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

`LyricHyphen.__and__ (arg)`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.__delattr__ ()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LyricHyphen.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`LyricHyphen.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__mul__(n)`

Note: Inherited from `componenttools.Component`

`LyricHyphen.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricHyphen.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.__repr__()`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`LyricHyphen.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`LyricHyphen.__str__()`

Note: Inherited from `leaftools.Leaf`

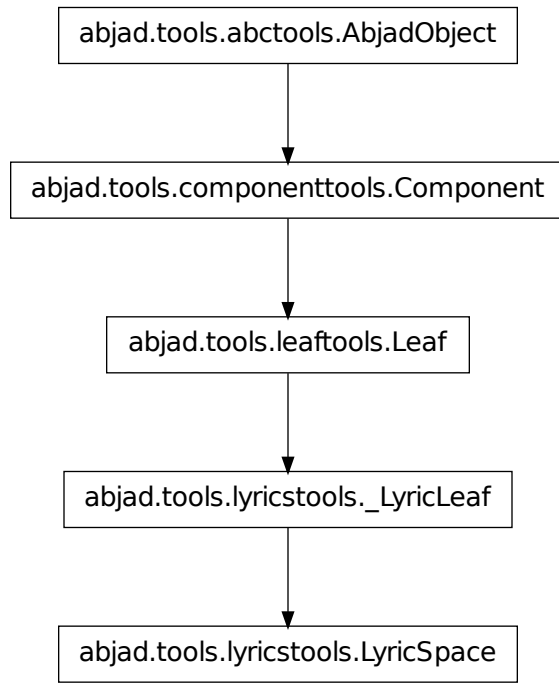
`LyricHyphen.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricHyphen.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

lyricstools.LyricSpace



```
class abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace(written_duration,
                                                                dura-
                                                                tion_multiplier=None)
```

Read-only Properties

`LyricSpace.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.format`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.leaf_index`

Note: Inherited from `leaftools.Leaf`

LyricSpace.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

LyricSpace.multiplied_duration

Note: Inherited from `leaftools.Leaf`

LyricSpace.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

LyricSpace.parent

Note: Inherited from `componenttools.Component`

LyricSpace.preprolated_duration

Note: Inherited from `leaftools.Leaf`

LyricSpace.prolated_duration

Note: Inherited from `componenttools.Component`

LyricSpace.prolation

Note: Inherited from `componenttools.Component`

LyricSpace.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

LyricSpace.spanners

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

`LyricSpace.duration_multiplier`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.written_duration`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.written_pitch_indication_is_at_sounding_pitch`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.written_pitch_indication_is_nonsemantic`

Note: Inherited from `leaftools.Leaf`

Special Methods

`LyricSpace.__and__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LyricSpace.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__hash__()` $\iff hash(x)$

Note: Inherited from `__builtin__.object`

`LyricSpace.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__mul__(n)`

Note: Inherited from `componenttools.Component`

`LyricSpace.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricSpace.__or__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.__repr__()`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`LyricSpace.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`

Note: Inherited from `__builtin__.object`

`LyricSpace.__str__()`

Note: Inherited from `leaftools.Leaf`

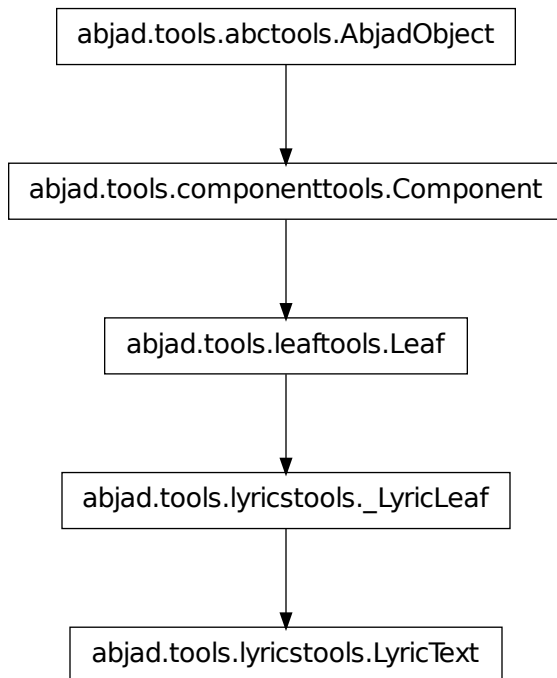
`LyricSpace.__sub__(arg)`

Note: Inherited from `leaftools.Leaf`

`LyricSpace.__xor__(arg)`

Note: Inherited from `leaftools.Leaf`

`lyricstools.LyricText`



class `abjad.tools.lyricstools.LyricText.LyricText` **LyricText** (*written_duration*, *duration_multiplier=None*)

Read-only Properties

`LyricText.duration_in_seconds`

Note: Inherited from `leaftools.Leaf`

`LyricText.format`

Note: Inherited from `leaftools.Leaf`

`LyricText.leaf_index`

Note: Inherited from `leaftools.Leaf`

`LyricText.marks`

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

`LyricText.multiplied_duration`

Note: Inherited from `leaftools.Leaf`

`LyricText.override`

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

`LyricText.parent`

Note: Inherited from `componenttools.Component`

`LyricText.preprolated_duration`

Note: Inherited from `leaftools.Leaf`

`LyricText.prolated_duration`

Note: Inherited from `componenttools.Component`

LyricText.**prolation**

Note: Inherited from `componenttools.Component`

LyricText.**set**

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

LyricText.**spanners**

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

LyricText.**duration_multiplier**

Note: Inherited from `leaftools.Leaf`

LyricText.**written_duration**

Note: Inherited from `leaftools.Leaf`

LyricText.**written_pitch_indication_is_at_sounding_pitch**

Note: Inherited from `leaftools.Leaf`

LyricText.**written_pitch_indication_is_nonsemantic**

Note: Inherited from `leaftools.Leaf`

Special Methods

LyricText.**__and__**(*arg*)

Note: Inherited from `leaftools.Leaf`

LyricText.**__delattr__**()

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`LyricText.__eq__(arg)`
True when `id(self)` equals `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

`LyricText.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricText.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`LyricText.__hash__()` $\leq \Rightarrow$ `hash(x)`

Note: Inherited from `__builtin__.object`

`LyricText.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricText.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`LyricText.__mul__(n)`

Note: Inherited from `componenttools.Component`

`LyricText.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

LyricText.__or__(arg)

Note: Inherited from `leaftools.Leaf`

LyricText.__repr__()

Note: Inherited from `leaftools.Leaf`

LyricText.__rmul__(n)

Note: Inherited from `componenttools.Component`

LyricText.__setattr__()
x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

LyricText.__str__()

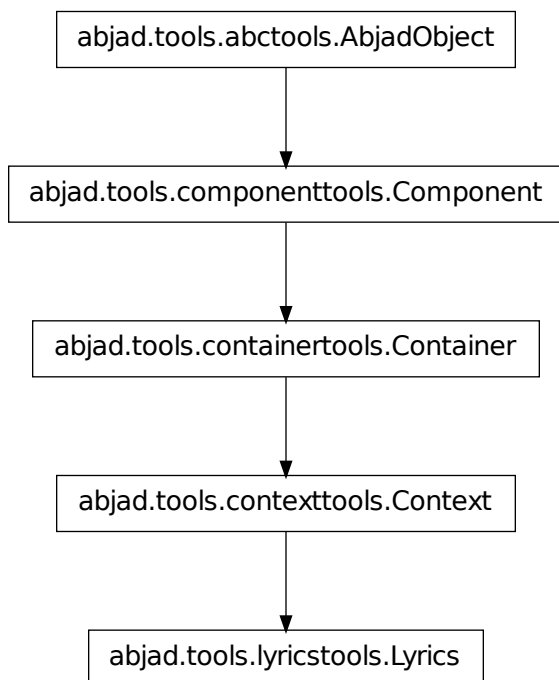
Note: Inherited from `leaftools.Leaf`

LyricText.__sub__(arg)

Note: Inherited from `leaftools.Leaf`

LyricText.__xor__(arg)

Note: Inherited from `leaftools.Leaf`

lyricstools.Lyrics

class `abjad.tools.lyricstools.Lyrics.Lyrics` (*music=None*)

Read-only Properties

`Lyrics.contents_duration`

Note: Inherited from `containertools.Container`

`Lyrics.duration_in_seconds`

Note: Inherited from `containertools.Container`

`Lyrics.engraver_consists`

New in version 2.0. Unordered set of LilyPond engravers to include in context definition.

Manage with add, update, other standard set commands.

```

abjad> staff = Staff([])
abjad> staff.engraver_consists.add('Horizontal_bracket_engraver')
abjad> f(staff)
\new Staff \with {

```

```
\consists Horizontal_bracket_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

Lyrics.engraver_removals

New in version 2.0. Unordered set of LilyPond engravers to remove from context.

Manage with add, update, other standard set commands.

```
abjad> staff = Staff([])
abjad> staff.engraver_removals.add('Time_signature_engraver')
abjad> f(staff)
\new Staff \with {
  \remove Time_signature_engraver
} {
}
```

Note: Inherited from `contexttools.Context`

Lyrics.format

Read-only LilyPond input format of component.

Note: Inherited from `componenttools.Component`

Lyrics.is_semantic

Note: Inherited from `contexttools.Context`

Lyrics.leaves

Read-only tuple of leaves in container:

```
abjad> container = Container("c'8 d'8 e'8")

abjad> container.leaves
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple of zero or more leaves.

Note: Inherited from `containertools.Container`

Lyrics.marks

Read-only tuple of marks attached to component.

Note: Inherited from `componenttools.Component`

Lyrics.music

Read-only tuple of components in container:

```
abjad> container = Container("c'8 d'8 e'8")
```

```
abjad> container.music  
(Note("c'8"), Note("d'8"), Note("e'8"))
```

Return tuple or zero or more components.

Note: Inherited from `containertools.Container`

Lyrics.override

Read-only reference to LilyPond grob override component plug-in.

Note: Inherited from `componenttools.Component`

Lyrics.parent

Note: Inherited from `componenttools.Component`

Lyrics.preprolated_duration

Note: Inherited from `containertools.Container`

Lyrics.prolated_duration

Note: Inherited from `componenttools.Component`

Lyrics.prolation

Note: Inherited from `componenttools.Component`

Lyrics.set

Read-only reference LilyPond context setting component plug-in.

Note: Inherited from `componenttools.Component`

Lyrics.spanners

Read-only reference to unordered set of spanners attached to component.

Note: Inherited from `componenttools.Component`

Read/write Properties

Lyrics.context_name

Read / write name of context as a string.

Note: Inherited from `contexttools.Context`

Lyrics.is_nonsemantic

Set indicator of nonsemantic voice:

```
abjad> measures = measuretools.make_measures_with_full_measure_spacer_skips([(1, 8), (5, 16), (5, 16)])
abjad> voice = Voice(measures)
abjad> voice.name = 'HiddenTimeSignatureVoice'

abjad> voice.is_nonsemantic = True

abjad> f(voice)
\context Voice = "HiddenTimeSignatureVoice" {
  {
    \time 1/8
    s1 * 1/8
  }
  {
    \time 5/16
    s1 * 5/16
  }
  {
    \time 5/16
    s1 * 5/16
  }
}

abjad> voice.is_nonsemantic
True
```

Get indicator of nonsemantic voice:

```
abjad> voice = Voice([])

abjad> voice.is_nonsemantic
False
```

Return boolean.

The intent of this read / write attribute is to allow composers to tag invisible voices used to house time signatures indications, bar number directives or other pieces of score-global non-musical information. Such nonsemantic voices can then be omitted from voice interaction and other functions.

Note: Inherited from `contexttools.Context`

Lyrics.is_parallel

Get parallel container:

```
abjad> container = Container([Voice("c'8 d'8 e'8"), Voice('g4.')])

abjad> f(container)
{
  \new Voice {
    c'8
    d'8
    e'8
  }
  \new Voice {
    g4.
  }
}
```

```
        g4.  
    }  
}  
  
abjad> container.is_parallel  
False
```

Return boolean.

Set parallel container:

```
abjad> container.is_parallel = True  
  
abjad> f(container)  
<<  
    \new Voice {  
        c'8  
        d'8  
        e'8  
    }  
    \new Voice {  
        g4.  
    }  
>>
```

Return none.

Note: Inherited from `containertools.Container`

Lyrics.name

Read-write name of context. Must be string or none.

Note: Inherited from `contexttools.Context`

Methods

Lyrics.append(*component*)

Append *component* to container:

```
abjad> container = Container("c'8 d'8 e'8")  
abjad> beam = spannertools.BeamSpanner(container.music)  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]  
}  
  
abjad> container.append(Note("f'8"))  
  
abjad> f(container)  
{  
    c'8 [  
    d'8  
    e'8 ]
```

```

    f'8
}

```

Return none.

Note: Inherited from `containertools.Container`

`Lyrics.extend(expr)`

Extend *expr* against container:

```

abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.extend([Note("cs'8"), Note("ds'8"), Note("es'8")])

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
    cs'8
    ds'8
    es'8
}

```

Return none. New in version 2.3: *expr* may now be a LilyPond input string.

Note: Inherited from `containertools.Container`

`Lyrics.index(component)`

Index *component* in container:

```

abjad> container = Container("c'8 d'8 e'8")

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.index(note)
2

```

Return nonnegative integer.

Note: Inherited from `containertools.Container`

`Lyrics.insert(i, component)`

Insert *component* in container at index *i*:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.insert(1, Note("cs'8"))

abjad> f(container)
{
    c'8 [
    cs'8
    d'8
    e'8 ]
}
```

Return none.

Note: Inherited from `containertools.Container`

Lyrics. `pop` (*i=-1*)

Pop component at index *i* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> container.pop(-1)
Note("e'8")

abjad> f(container)
{
    c'8 [
    d'8 ]
}
```

Return component.

Note: Inherited from `containertools.Container`

Lyrics. `remove` (*component*)

Remove *component* from container:

```
abjad> container = Container("c'8 d'8 e'8")
abjad> beam = spannertools.BeamSpanner(container.music)
```

```

abjad> f(container)
{
    c'8 [
    d'8
    e'8 ]
}

abjad> note = container[-1]
abjad> note
Note("e'8")

abjad> container.remove(note)

abjad> f(container)
{
    c'8 [
    d'8 ]
}

```

Return none.

Note: Inherited from `containertools.Container`

Special Methods

`Lyrics.__add__(expr)`

Concatenate containers self and expr. The operation `c = a + b` returns a new Container `c` with the content of both `a` and `b`. The operation is non-commutative: the content of the first operand will be placed before the content of the second operand.

Note: Inherited from `containertools.Container`

`Lyrics.__contains__(expr)`

True if `expr` is in container, otherwise False.

Note: Inherited from `containertools.Container`

`Lyrics.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Lyrics.__delitem__(i)`

Find component(s) at index or slice 'i' in container. Detach component(s) from parentage. Withdraw component(s) from crossing spanners. Preserve spanners that component(s) cover(s).

Note: Inherited from `containertools.Container`

`Lyrics.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__getitem__(i)`

Return component at index *i* in container. Shallow traversal of container for numeric indices only.

Note: Inherited from `containertools.Container`

`Lyrics.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`Lyrics.__iadd__(expr)`

`__iadd__` avoids unnecessary copying of structures.

Note: Inherited from `containertools.Container`

`Lyrics.__imul__(total)`

Multiply contents of container ‘total’ times. Return multiplied container.

Note: Inherited from `containertools.Container`

`Lyrics.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__len__()`

Return nonnegative integer number of components in container.

Note: Inherited from `containertools.Container`

`Lyrics.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__mul__(n)`

Note: Inherited from `componenttools.Component`

`Lyrics.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`Lyrics.__radd__(expr)`

Extend container by contents of `expr` to the right.

Note: Inherited from `containertools.Container`

`Lyrics.__repr__()`

Changed in version 2.0. Named contexts now print name at the interpreter.

Note: Inherited from `contexttools.Context`

`Lyrics.__rmul__(n)`

Note: Inherited from `componenttools.Component`

`Lyrics.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`Lyrics.__setitem__(i, expr)`

Set 'expr' in self at nonnegative integer index `i`. Or, set 'expr' in self at slice `i`. Find spanners that dominate `self[i]` and children of `self[i]`. Replace contents at `self[i]` with 'expr'. Reattach spanners to new contents. This operation leaves all score trees always in tact.

Note: Inherited from `containertools.Container`

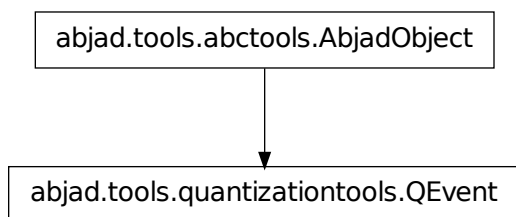
`Lyrics.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

quantizationtools

quantizationtools concrete classes

quantizationtools.QEvent



class `abjad.tools.quantizationtools.QEvent.QEvent(*args)`

A utility class for quantization comprising an offset time in milliseconds, and some pitch information: a `Number` representing a single pitch, `None` representing silence, or an `Iterable` comprised of `Numbers` representing a chord.

QEvents are immutable.

Read-only Properties

`QEvent.offset`

The offset in milliseconds of the event.

`QEvent.value`

The pitch information of the event.

Special Methods

`QEvent.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QEvent.__eq__(other)`

`QEvent.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QEvent.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`QEvent.__hash__()` $\iff hash(x)$

Note: Inherited from `__builtin__.object`

`QEvent.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QEvent.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QEvent.__ne__(arg)`
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

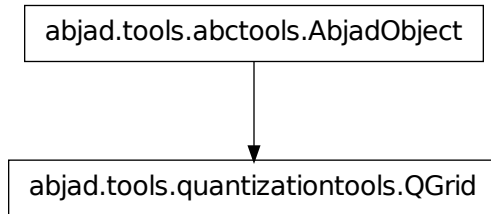
`QEvent.__repr__()`

`QEvent.__setattr__()`
`x.__setattr__('name', value) \iff x.name = value`

Note: Inherited from `__builtin__.object`

`QEvent.__str__()` $\iff str(x)$

Note: Inherited from `__builtin__.object`

quantizationtools.QGrid

class `abjad.tools.quantizationtools.QGrid.QGrid(definition, next)`
 Abjad model of a *QGrid*, a nesting division structure which assists certain quantization algorithms.

QGrids are defined by a list, which must be prime in length, whose members are either Numbers or tuples of Numbers (useful for representing timepoint or pitch information), a *QEvent* or tuple of *QEvent* objects, or None (representing silence), or other lists which must recursively obey the same rules.

QGrids also have a *next* attribute, representing the downbeat of not “this” *QGrid*, but the next *QGrid* in a list of grids. This is useful as timepoints must often be quantized not to any internal division of a the “current” beat, but to the next beat.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, 0, [0, 0]], 0)
```

The values in the grid can be access via subscript, as though the grid were a flat list.

```
abjad> q[0] = 1
abjad> q[2] = 3
abjad> q[4] = 5
abjad> q
QGrid([1, 0, [3, 0]], 5)
```

QGrids are quasi-immutable.

Read-only Properties***QGrid*.*definition***

The nested list which defines the *QGrid*’s structure.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, 0, [0, 0]], 0)
abjad> q.definition
[0, 0, [0, 0]]
```

Read-only.

QGrid*.*next

The contents of the final offset in the *QGrid*.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, 0, [0, 0]], 0)
abjad> q[-1] = 9
abjad> q
```

```
QGrid([0, 0, [0, 0]], 9)
abjad> q.next
9
```

Read-only.

QGrid.offsets

An ordered tuple of those *Offset* objects generated by the division structure of a *QGrid*.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, [0, 0], 0], 0)
abjad> q.offsets
(Offset(0, 1), Offset(1, 3), Offset(1, 2), Offset(2, 3), Offset(1, 1))
```

Read-only.

Methods

QGrid.find_divisible_indices (*points*)

Given a list of numbers $0 \leq n \leq 1$, return a list of indices in self which contain those points, as though they were segments.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, [0, 0]], 0)
abjad> q.offsets
(Offset(0, 1), Offset(1, 2), Offset(3, 4), Offset(1, 1))
abjad> points = [0.1, 0.9]
abjad> q.find_divisible_indices(points)
[0, 2]
```

Returns a list.

QGrid.find_parentage_of_index (*index*)

Return a tuple of the lengths of each container containing *index*, from the topmost to the bottommost.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, [0, [0, 0], 0], 0, 0, 0, 0], 0)
abjad> q.find_parentage_of_index(0)
(5,)
abjad> q.find_parentage_of_index(1)
(5, 3)
abjad> q.find_parentage_of_index(2)
(5, 3, 2)
abjad> q.find_parentage_of_index(7)
(5,)
```

Returns a tuple.

QGrid.format_for_beatspan (*beatspan*=*Fraction(1, 4)*)

Return an Abjad container, whose structure mirrors the division structure of the *QGrid*. The values of the items in the *QGrid* have no effect on the output.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, [0, 0], 0], 0)
abjad> q.format_for_beatspan()
Tuplet(2/3, [c'8, c'16, c'16, c'8])
```

Returns a *Tuplet* or *Container*, depending on structure.

QGrid.subdivide_indices (*pairs*)

Given a list of 2-tuples, where for each tuple *t*, *t*[0] is a valid index into self, and *t*[1] is a prime integer greater

than 1, return a new *QGrid* with those indices subdivided.

```
abjad> from abjad.tools.quantizationtools import QGrid
abjad> q = QGrid([0, 0], 0)
abjad> q.subdivide_indices([(0, 2), (1, 3)])
QGrid([[0, 0], [0, 0, 0]], 0)
```

Returns a new *QGrid*.

Special Methods

`QGrid.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QGrid.__eq__(other)`

`QGrid.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGrid.__getitem__(item)`

`QGrid.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`QGrid.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`QGrid.__iter__()`

`QGrid.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGrid.__len__()`

`QGrid.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGrid.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`QGrid.__repr__()`

`QGrid.__setattr__()`

`x.__setattr__('name', value) <==> x.name = value`

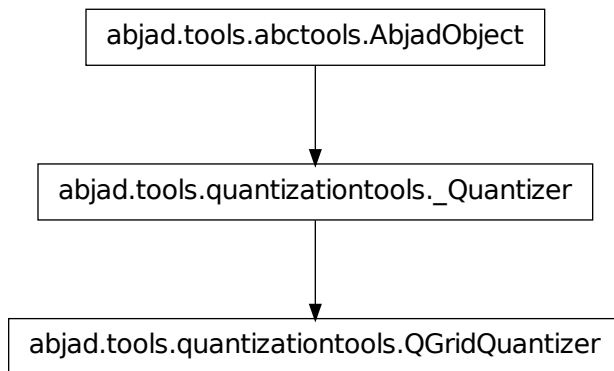
Note: Inherited from `__builtin__.object`

`QGrid.__setitem__(item, value)`

`QGrid.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

quantizationtools.QGridQuantizer



class `abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer` (*search_tree=None, beat-span=Fraction(1, 4), tempo=TempoMa*
4), 60), thresh-
old=None)

An Abjad implementation of Paul Nauert's Q-grid quantization algorithm.

Input is converted into timepoints, which are grouped according to which beat - or *beatspan* - they fall in, given a target tempo. Each beatspan is then divided into grids called Q-grids, which are based upon a nesting division structure (similar to nested tuplets). The Q-grids generated for each beatspan are then tested against the timepoints falling within that beatspan, and the grid with least deviation is chosen to represent the rhythmic skeleton for that beat.

```
abjad> from abjad.tools.quantizationtools import QGridQuantizer
abjad> q = QGridQuantizer()
```

QGridQuantizer is immutable, but cheap to instantiate. Various attributes can be defined on instantiation. Please consult the documentation for each attribute respectively, for proper usage.

```
abjad> from abjad.tools.quantizationtools import QGridSearchTree
abjad> target_tempo = contexttools.TempoMark((1, 8), 73)
abjad> beatspan = Fraction(1, 4)
abjad> search_tree = QGridSearchTree({2: {2: None, 3: None}, 5: None})
abjad> threshold = 250
abjad> q = QGridQuantizer(tempo = target_tempo, beatspan = beatspan, search_tree = search_tree,
```

QGridQuantizer can quantize lists of leaves. If the source leaves have no effective tempo, one must be provided with the *tempo* keyword.

```
abjad> q = QGridQuantizer()
abjad> source = Staff("c'4 d'4 e'4. r8 <c' e' g'>2. <d' g' b'>4")
abjad> source_tempo = contexttools.TempoMark((1, 4), 54)
abjad> result = q(source[:], tempo = source_tempo)

abjad> q = QGridQuantizer()
abjad> source = Staff("c'4 d'4 e'4. r8 <c' e' g'>2. <d' g' b'>4")
abjad> t = contexttools.TempoMark((1, 8), 34, target_context = Staff)(source)
abjad> t = contexttools.TempoMark((1, 4), 135, target_context = Staff)(source[3])
abjad> result = q(source[:])
```

QGridQuantizer can quantize lists of millisecond durations. Negative values can be used to indicate silences.

```
abjad> q = QGridQuantizer()
abjad> milliseconds = [100, 120, -133, 500, -1003, 125]
abjad> result = q(milliseconds)
```

QGridQuantizer can also quantize lists of rationals, if a tempo is provided. As with quantizing millisecond durations, negative values can be used to indicate silences.

```
abjad> q = QGridQuantizer()
abjad> rationals = [1, Fraction(1, 2), Fraction(-1, 4), 3, Fraction(-1, 3), 2]
abjad> tempo = contexttools.TempoMark((1, 4), 45)
abjad> result = q(rationals, tempo = tempo)
```

Lastly, *QGridQuantizer* can quantize lists of pairs, where the first value in each pair is a millisecond duration, and the second value is an int or float - indicating a single pitch -, None - indicating silence, or a list of ints or floats - indicating a chord. This is probably most useful for assisting in the importation of audio analyses from other tools.

```
abjad> q = QGridQuantizer()
abjad> pairs = [(130, 0), (250, 2), (500, None), (1303, [0, 1, 4])]
abjad> result = q(pairs)
```

Todo

Write a documentation chapter on quantization.

Todo

Implement multiprocessing-based QGrid comparison

Read-only Properties

`QGridQuantizer.beatspan`

The basic division of the beat for quantization.

Read-only, defaults to *Duration(1, 4)*.

`QGridQuantizer.beatspan_ms`

The duration of *beatspan* in milliseconds, as determined by *tempo*.

Read-only, defaults to *Duration(1000)*.

`QGridQuantizer.search_tree`

Reference to a `QGridSearchTree` object, which defines the permissible divisions for each `QGrid` comprising a quantization attempt.

Read-only, defaults to *QGridSearchTree()*.

Please consult the documentation for `QGrid` and `QGridSearchTree` for more information.

`QGridQuantizer.tempo`

Reference to a `TempoMark`, defining the target tempo for all quantization results.

Read-only, defaults to *TempoMark((1, 4), 60)*.

`QGridQuantizer.tempo_lookup`

Reference to a `QGridTempoLookup` object, a utility class for mapping rational divisions of a beat into milliseconds.

Read-only.

`QGridQuantizer.threshold`

Millisecond duration, which if specified at instantiation will be used to call the quantizer's `QGridSearchTree.prune()` method, in order to generate a pruned search tree for the quantizer, instead of either the user-provided or default search trees.

Read-only, defaults to `None`. See the documentation for `QGridSearchTree` for more information on pruning.

Special Methods

`QGridQuantizer.__call__(args, **kwargs)`

Note: Inherited from `quantizationtools._Quantizer`

`QGridQuantizer.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QGridQuantizer.__eq__(arg)`

True when `id(self)` equals `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`QGridQuantizer.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__lt__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__ne__(arg)`

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

`QGridQuantizer.__repr__()`

Note: Inherited from `quantizationtools._Quantizer`

`QGridQuantizer.__setattr__()`

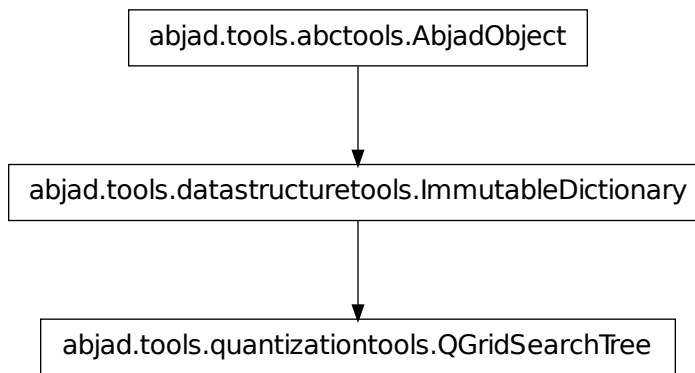
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`QGridQuantizer.__str__()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

quantizationtools.QGridSearchTree



class `abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree` (*definition=...*)
 A utility class for defining the permissible divisions of a collection of `QGrid` objects.

The search tree is defined by a nested dictionary structure, whose keys must be prime integers, and whose values must be `None` (indicating no further possible divisions) or another dictionary following the same rules.

```
abjad> from abjad.tools.quantizationtools import QGridSearchTree
```

For example, In the following tree, the beat may be divided into 2 or into 5. If divided into 2, it may be divided again into 2 or into 3.

```
abjad> search_tree = QGridSearchTree({2: {2: None, 3: None}, 5: None})
```

Return a new *QGridSearchTree*.

Read-only Properties

`QGridSearchTree.offsets`

An ordered tuple of all `Offset` objects which those `QGrid` objects governed by a specific *QGridSearchTree* can contain.

```
abjad> from abjad.tools.quantizationtools import QGridSearchTree
abjad> qst = QGridSearchTree({2: {3: None}})
abjad> qst.offsets
(Offset(0, 1), Offset(1, 6), Offset(1, 3), Offset(1, 2), Offset(2, 3), Offset(5, 6), Offset(1, 1))
```

Returns a tuple.

Methods

`QGridSearchTree.clear()` → None. Remove all items from D.

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.find_subtree_divisibility` (*parentage*)

Given a parentage signature, defining some subtree of a *QGridSearchTree*, return a tuple of permitted divisions of that subtree.

```
abjad> from abjad.tools.quantizationtools import QGridSearchTree
abjad> qst = QGridSearchTree({2: {2: None, 3: {7: None, 11: None}}, 5: None})
abjad> qst.find_subtree_divisibility((2,))
(2, 3)
abjad> qst.find_subtree_divisibility((2, 2))
()
abjad> qst.find_subtree_divisibility((2, 3))
(7, 11)
abjad> qst.find_subtree_divisibility((2, 3, 7))
()
```

Returns a tuple.

`QGridSearchTree.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.intervals()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.popitem()` → (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.prune(beatspan, tempo, threshold)`

Prune those subtrees of a *QGridSearchTree* whose divisions in milliseconds, given *beatspan* and *tempo*, would be less than *threshold*.

This allows a composer to specify the maximum speed any quantization operation will permit.

```
abjad> from abjad.tools.quantizationtools import QGridSearchTree
abjad> qst = QGridSearchTree({2: {2: {2: {2: None}}}}})
abjad> beatspan = Fraction(1, 4)
abjad> tempo = contexttools.TempoMark((1, 4), 60)
abjad> qst.prune(beatspan, tempo, 100)
{2: {2: {2: None}}}
abjad> qst.prune(beatspan, tempo, 200)
{2: {2: None}}
abjad> qst.prune(beatspan, tempo, 400)
{2: None}
```

Returns a new *QGridSearchTree*.

`QGridSearchTree.setdefault(k[, d])` → D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.update(E, **F)` → None. Update D from dict/iterable E and F.

If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.values()` → list of D's values

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.viewitems()` → a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.viewkeys()` → a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.viewvalues()` → an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`QGridSearchTree.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__contains__(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QGridSearchTree.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`QGridSearchTree.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__len__()` <==> `len(x)`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__repr__()` <==> `repr(x)`

Note: Inherited from `__builtin__.dict`

`QGridSearchTree.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

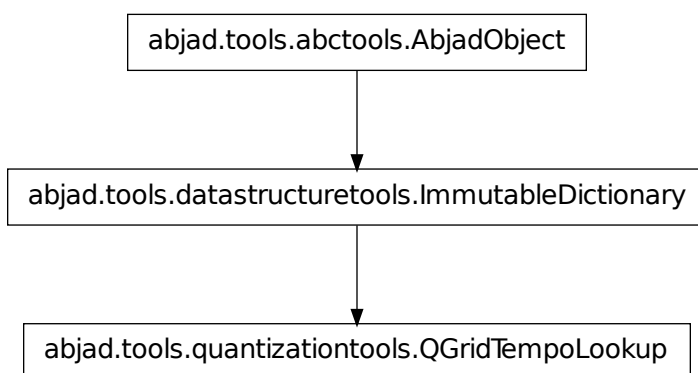
`QGridSearchTree.__setitem__ (*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`QGridSearchTree.__str__ ()` \Leftrightarrow `str(x)`

Note: Inherited from `__builtin__.object`

quantizationtools.QGridTempoLookup



class `abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup` (*offsets*, *beat-span*, *tempo*)

A utility class for matching fractional offsets within a beat to their tempo-scaled (real-time) millisecond values.

QGridTempoLookup objects are immutable.

Read-only Properties

`QGridTempoLookup.beatspan`

The duration which the `Offset` objects comprising the keys of the *QGridTempoLookup* are offsets into.

`QGridTempoLookup.tempo`

The `TempoMark` used to generate the lookup.

Methods

`QGridTempoLookup.clear ()` \rightarrow `None`. Remove all items from `D`.

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.copy()` → a shallow copy of D

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.get(k[, d])` → D[k] if k in D, else d. d defaults to None.

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.has_key(k)` → True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.items()` → list of D's (key, value) pairs, as 2-tuples

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.iteritems()` → an iterator over the (key, value) items of D

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.iterkeys()` → an iterator over the keys of D

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.itervalues()` → an iterator over the values of D

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.keys()` → list of D's keys

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.pop(k[, d])` → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise `KeyError` is raised

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.popitem()` \rightarrow (k, v), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if D is empty.

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.setdefault(k[, d])` \rightarrow D.get(k,d), also set D[k]=d if k not in D

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.update(E, **F)` \rightarrow None. Update D from dict/iterable E and F.

If E has a `.keys()` method, does: for k in E: D[k] = E[k] If E lacks `.keys()` method, does: for (k, v) in E: D[k] = v
In either case, this is followed by: for k in F: D[k] = F[k]

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.values()` \rightarrow list of D's values

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.viewitems()` \rightarrow a set-like object providing a view on D's items

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.viewkeys()` \rightarrow a set-like object providing a view on D's keys

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.viewvalues()` \rightarrow an object providing a view on D's values

Note: Inherited from `__builtin__.dict`

Special Methods

`QGridTempoLookup.__cmp__(y)` $\leq \Rightarrow \text{cmp}(x, y)$

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__contains__(k)` \rightarrow True if D has a key k, else False

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QGridTempoLookup.__delitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`QGridTempoLookup.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__iter__() <==> iter(x)`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__len__() <==> len(x)`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__ne__()`
`x.__ne__(y) <==> x!=y`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__repr__()` <==> `repr(x)`

Note: Inherited from `__builtin__.dict`

`QGridTempoLookup.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`QGridTempoLookup.__setitem__(*args)`

Note: Inherited from `datastructuretools.ImmutableDictionary`

`QGridTempoLookup.__str__()` <==> `str(x)`

Note: Inherited from `__builtin__.object`

quantizationtools functions

quantizationtools.is_valid_beatspan

`abjad.tools.quantizationtools.is_valid_beatspan.is_valid_beatspan(beatspan)`
 True if *beatspan* is a valid beatspan.

1. A beatspan must be an int or Fraction.
2. It must be a binary rational.
3. If it is greater than zero, it must be a power of two.
4. If it is less than zero, it must be Fraction, whose numerator is 1 and whose denominator is a power of two.

quantizationtools.millisecond_pitch_pairs_to_q_events

`abjad.tools.quantizationtools.millisecond_pitch_pairs_to_q_events.millisecond_pitch_pairs_to_q_events(millisecond_pitch_pairs)`
 Convert a list of pairs of millisecond durations and pitches to a list of `QEvent` instances.

Pitch values must be one of the following:

1. A single chromatic pitch number, indicating a note,

2.None, indicating a silence, or

3.An iterable of chromatic pitch numbers, indicating a chord.

```
abjad> from abjad.tools.quantizationtools import millisecond_pitch_pairs_to_q_events
abjad> durations = [1001, 503, 230, 1340]
abjad> pitches = [None, 0, (1, 2, 3), 4.5]
abjad> pairs = zip(durations, pitches)
abjad> millisecond_pitch_pairs_to_q_events(pairs)
[QEvent(Offset(0, 1), None), QEvent(Offset(1001, 1), 0), QEvent(Offset(1504, 1), (1, 2, 3)), QEvent(Offset(2007, 1), 4.5)]
```

Return a list of QEvent instances.

quantizationtools.milliseconds_to_q_events

`abjad.tools.quantizationtools.milliseconds_to_q_events.milliseconds_to_q_events(milliseconds)`
 Convert a list of millisecond durations to a list of QEvent objects.

Negative duration values can be used to indicate silence. Any resulting pitched QEvent objects will default to using middle-C.

```
abjad> from abjad.tools.quantizationtools import milliseconds_to_q_events
abjad> durations = [100, -250, 500]
abjad> milliseconds_to_q_events(durations)
[QEvent(Offset(0, 1), 0), QEvent(Offset(100, 1), None), QEvent(Offset(350, 1), 0), QEvent(Offset(500, 1), 0)]
```

Return a list of QEvent objects.

quantizationtools.tempo_scaled_leaves_to_q_events

`abjad.tools.quantizationtools.tempo_scaled_leaves_to_q_events.tempo_scaled_leaves_to_q_events(leaves, tempo)`

Convert *leaves* to a list of QEvent objects. If the leaves have no effective tempo, *tempo* must be a TempoMark.

```
abjad> from abjad.tools.quantizationtools import tempo_scaled_leaves_to_q_events
abjad> source = Staff("c'4 r4. e'8 <g' b' d'' fs''>2")
abjad> source_tempo = contexttools.TempoMark((1, 4), 55)
abjad> tempo_scaled_leaves_to_q_events(source[:], tempo = source_tempo)
[QEvent(Offset(0, 1), 0), QEvent(Offset(12000, 11), None), QEvent(Offset(30000, 11), 4), QEvent(Offset(42000, 11), 4)]
```

Return a list of QEvent objects.

quantizationtools.tempo_scaled_rational_to_milliseconds

`abjad.tools.quantizationtools.tempo_scaled_rational_to_milliseconds.tempo_scaled_rational_to_milliseconds(rational, tempo)`

Return the millisecond value of *rational* at *tempo*.

```
abjad> from abjad.tools.quantizationtools import tempo_scaled_rational_to_milliseconds
abjad> tempo = contexttools.TempoMark((1, 4), 60)
abjad> tempo_scaled_rational_to_milliseconds(Fraction(1, 4), tempo)
Duration(1000, 1)
```

Return a Duration.

quantizationtools.tempo_scaled_rationals_to_q_events

`abjad.tools.quantizationtools.tempo_scaled_rationals_to_q_events.tempo_scaled_rationals_to`

Convert a list of rational durations to a list of `QEvent` objects.

Negative duration values can be used to indicate silence. Any resulting pitched `QEvent` objects will default to using middle-C.

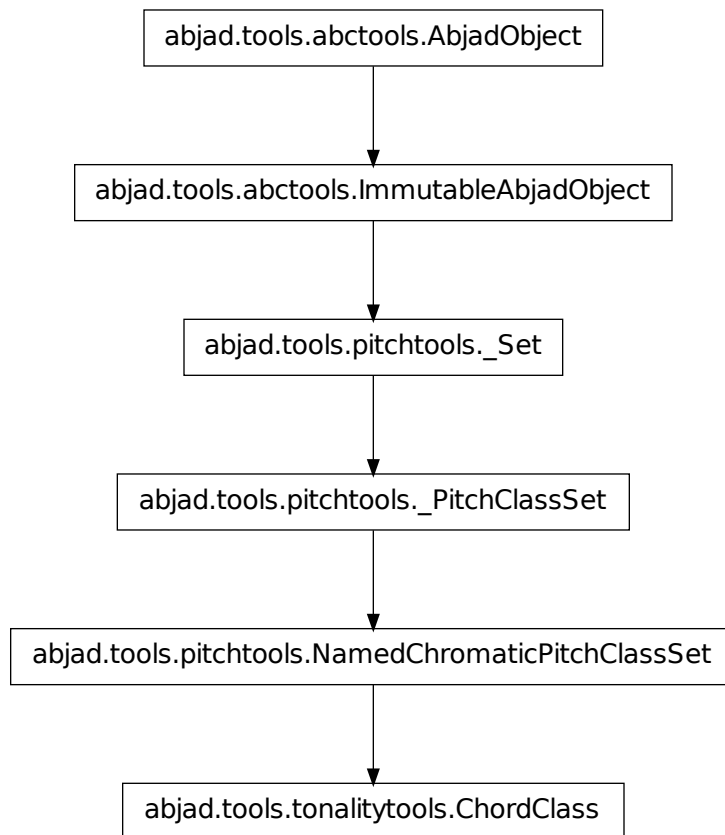
```
abjad> from abjad.tools.quantizationtools import tempo_scaled_rationals_to_q_events
abjad> durations = [Duration(-1, 2), Duration(1, 4), Duration(1, 6)]
abjad> tempo = contexttools.TempoMark((1, 4), 55)
abjad> tempo_scaled_rationals_to_q_events(durations, tempo)
[QEvent(Offset(0, 1), None), QEvent(Offset(24000, 11), 0), QEvent(Offset(36000, 11), 0), QEvent(
```

Return a list of `QEvent` objects.

`tonalitytools`

`tonalitytools` concrete classes

`tonalitytools.ChordClass`



```
class abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass (*args,
                                                                    **kwargs)
```

New in version 2.0. Abjad model of tonal chords like G 7, G 6/5, G half-diminished 6/5, etc.

Note that notions like G 7 represent an entire *class of* chords because there are many different spacings and registrations of a G 7 chord.

Read-only Properties

`ChordClass.bass`

`ChordClass.cardinality`

`ChordClass.extent`

`ChordClass.figured_bass`

`ChordClass.inversion`

`ChordClass.inversion_equivalent_diatonic_interval_class_vector`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.markup`

`ChordClass.named_chromatic_pitch_classes`

Read-only named chromatic pitch-classes:

```
abjad> named_chromatic_pitch_class_set = pitchtools.NamedChromaticPitchClassSet(['gs', 'g', 'as']
abjad> named_chromatic_pitch_class_set.named_chromatic_pitch_classes # doctest: +SKIP
(NamedChromaticPitchClass('c'), NamedChromaticPitchClass('cs'), NamedChromaticPitchClass('g'), N
```

Return tuple.

Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.numbered_chromatic_pitch_class_set`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.quality_indicator`

`ChordClass.quality_pair`

`ChordClass.root`

`ChordClass.root_string`

Methods

`ChordClass.copy()`

Return a shallow copy of a set.

Note: Inherited from `__builtin__.frozenset`

`ChordClass.difference()`

Return the difference of two or more sets as a new set.

(i.e. all elements that are in this set but not the others.)

Note: Inherited from `__builtin__.frozenset`

`ChordClass.intersection()`

Return the intersection of two or more sets as a new set.

(i.e. elements that are common to all of the sets.)

Note: Inherited from `__builtin__.frozenset`

`ChordClass.isdisjoint()`

Return True if two sets have a null intersection.

Note: Inherited from `__builtin__.frozenset`

`ChordClass.issubset()`

Report whether another set contains this set.

Note: Inherited from `__builtin__.frozenset`

`ChordClass.issuperset()`

Report whether this set contains another set.

Note: Inherited from `__builtin__.frozenset`

`ChordClass.order_by(npc_seg)`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.symmetric_difference()`

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

Note: Inherited from `__builtin__.frozenset`

`ChordClass.transpose()`

`ChordClass.union()`

Return the union of sets as a new set.

(i.e. all elements that are in either set.)

Note: Inherited from `__builtin__.frozenset`

Special Methods

`ChordClass.__and__()`

`x.__and__(y) <==> x&y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__cmp__(y) <==> cmp(x, y)`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__contains__()`

`x.__contains__(y) <==> y in x.`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ChordClass.__eq__(arg)`
`ChordClass.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__hash__()`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.__iter__()` `<==> iter(x)`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__len__()` `<==> len(x)`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__lt__()`
`x.__lt__(y) <==> x<y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__ne__(arg)`
`ChordClass.__or__()`
`x.__or__(y) <==> x|y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__rand__()`
`x.__rand__(y) <==> y&x`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__repr__()`
`ChordClass.__ror__()`
`x.__ror__(y) <==> y|x`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__rsub__()`
`x.__rsub__(y) <==> y-x`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__rxor__()`
`x.__rxor__(y) <==> y^x`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`ChordClass.__str__()`

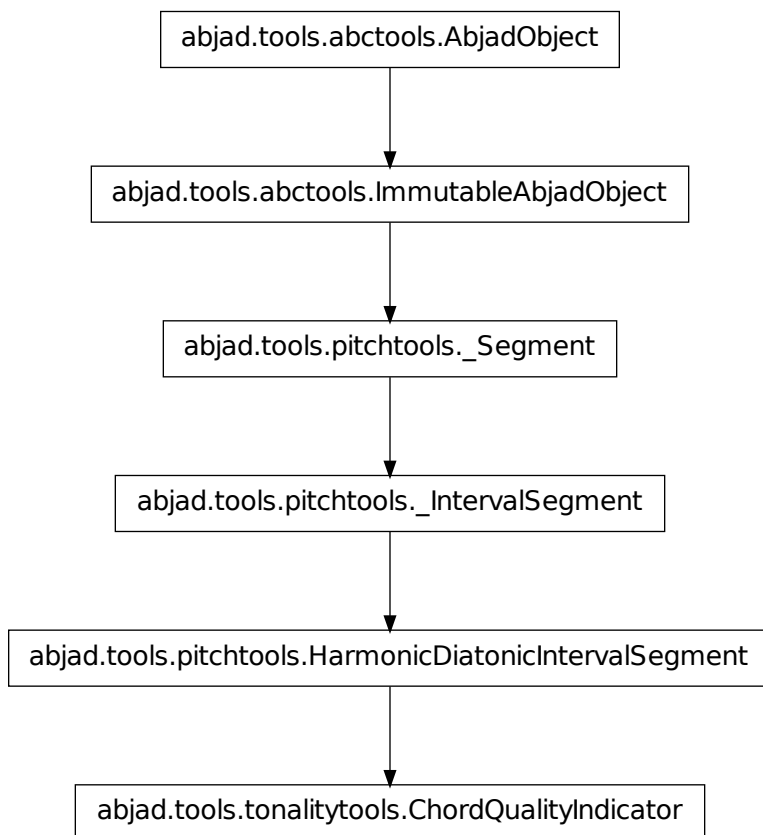
Note: Inherited from `pitchtools.NamedChromaticPitchClassSet`

`ChordClass.__sub__()`
`x.__sub__(y) <==> x-y`

Note: Inherited from `__builtin__.frozenset`

`ChordClass.__xor__()`
`x.__xor__(y) <==> x^y`

Note: Inherited from `__builtin__.frozenset`

tonalitytools.ChordQualityIndicator

class `abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator`. **ChordQualityIndicator**

New in version 2.0. Chord quality indicator.

Read-only Properties

`ChordQualityIndicator`.**cardinality**

`ChordQualityIndicator`.**extent**

`ChordQualityIndicator`.**extent_name**

`ChordQualityIndicator`.**harmonic_chromatic_interval_segment**

Note: Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator`.**interval_classes**

Note: Inherited from `pitchtools._IntervalSegment`

`ChordQualityIndicator.intervals`

Note: Inherited from `pitchtools._IntervalSegment`

`ChordQualityIndicator.inversion`

`ChordQualityIndicator.melodic_chromatic_interval_segment`

Note: Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator.melodic_diatonic_interval_segment`

Note: Inherited from `pitchtools.HarmonicDiatonicIntervalSegment`

`ChordQualityIndicator.position`

`ChordQualityIndicator.quality_string`

`ChordQualityIndicator.rotation`

Methods

`ChordQualityIndicator.count` (*value*) → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.index` (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.rotate` (*n*)

Note: Inherited from `pitchtools._IntervalSegment`

Special Methods

`ChordQualityIndicator.__add__` (*arg*)

Note: Inherited from `pitchtools._Segment`

`ChordQualityIndicator.__contains__` ()
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ChordQualityIndicator.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__getslice__(start, stop)`

Note: Inherited from `pitchtools._Segment`

`ChordQualityIndicator.__gt__()`
`x.__gt__(y) <==> x>y`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__hash__()` <==> `hash(x)`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__iter__()` <==> `iter(x)`

Note: Inherited from `__builtin__.tuple`

`ChordQualityIndicator.__le__()`
`x.__le__(y) <==> x<=y`

Note: Inherited from `__builtin__.tuple`

ChordQualityIndicator.__len__() $\iff len(x)$

Note: Inherited from `__builtin__.tuple`

ChordQualityIndicator.__lt__()
 $x.__lt__(y) \iff x < y$

Note: Inherited from `__builtin__.tuple`

ChordQualityIndicator.__mul__(n)

Note: Inherited from `pitchtools._Segment`

ChordQualityIndicator.__ne__()
 $x.__ne__(y) \iff x \neq y$

Note: Inherited from `__builtin__.tuple`

ChordQualityIndicator.__repr__()

ChordQualityIndicator.__rmul__(n)

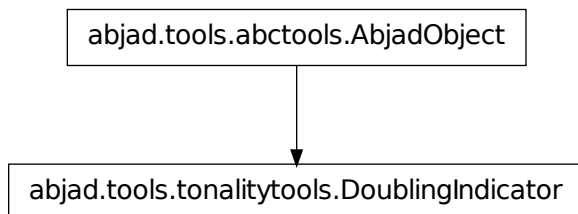
Note: Inherited from `pitchtools._Segment`

ChordQualityIndicator.__setattr__()
 $x.__setattr__('name', value) \iff x.name = value$

Note: Inherited from `__builtin__.object`

ChordQualityIndicator.__str__()

Note: Inherited from `pitchtools._IntervalSegment`

tonalitytools.DoublingIndicator

class `abjad.tools.tonalitytools.DoublingIndicator.DoublingIndicator` (*doublings*)
 New in version 2.0. Indicator of chord doubling.

Value object that can not be changed after instantiation.

Read-only Properties

`DoublingIndicator.doublings`

Special Methods

`DoublingIndicator.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`DoublingIndicator.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`DoublingIndicator.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`DoublingIndicator.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

DoublingIndicator.__hash__() \Leftrightarrow hash(*x*)

Note: Inherited from `__builtin__.object`

DoublingIndicator.__le__(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

DoublingIndicator.__lt__(*arg*)
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

DoublingIndicator.__ne__(*arg*)
True when `id(self)` does not equal `id(arg)`.
Return boolean.

Note: Inherited from `abctools.AbjadObject`

DoublingIndicator.__repr__()
Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.
Return string.

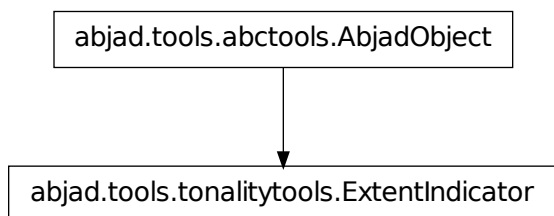
Note: Inherited from `abctools.AbjadObject`

DoublingIndicator.__setattr__()
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

DoublingIndicator.__str__() \Leftrightarrow str(*x*)

Note: Inherited from `__builtin__.object`

tonalitytools.ExtentIndicator

class `abjad.tools.tonalitytools.ExtentIndicator`.`ExtentIndicator`.**ExtentIndicator** (*arg*)
 New in version 2.0. Indicator of chord extent, such as triad, seventh chord, ninth chord, etc.

Value object that can not be changed after instantiation.

Read-only Properties

`ExtentIndicator.name`

`ExtentIndicator.number`

Special Methods

`ExtentIndicator.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ExtentIndicator.__eq__(arg)`

`ExtentIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ExtentIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`ExtentIndicator.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

ExtentIndicator.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ExtentIndicator.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

ExtentIndicator.__ne__(arg)

ExtentIndicator.__repr__()

ExtentIndicator.__setattr__()

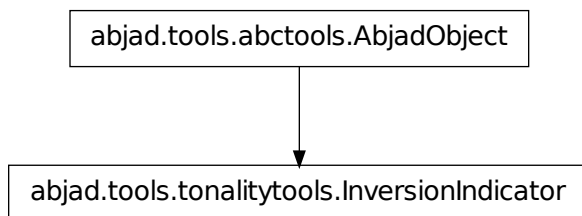
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

ExtentIndicator.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

tonalitytools.InversionIndicator



class `abjad.tools.tonalitytools.InversionIndicator.InversionIndicator` (*arg=*

New in version 2.0. Indicator of the inversion of tertian chords: 5, 63, 64 and also 7, 65, 43, 42, etc. Also root position, first, second, third inversions, etc.

Value object that can not be changed once initialized.

Read-only Properties

`InversionIndicator.name`
`InversionIndicator.number`
`InversionIndicator.title`

Methods

`InversionIndicator.extent_to_figured_bass_string`(*extent*)

Special Methods

`InversionIndicator.__delattr__()`
 `x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`InversionIndicator.__eq__(arg)`

`InversionIndicator.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionIndicator.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`InversionIndicator.__hash__()` *<==> hash(x)*

Note: Inherited from `__builtin__.object`

`InversionIndicator.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionIndicator.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`InversionIndicator.__ne__(arg)`

`InversionIndicator.__repr__()`

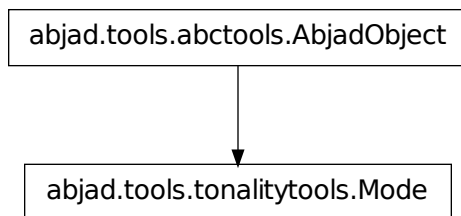
```
InversionIndicator.__setattr__()  
x.__setattr__('name', value) <==> x.name = value
```

Note: Inherited from `__builtin__.object`

```
InversionIndicator.__str__() <==> str(x)
```

Note: Inherited from `__builtin__.object`

tonalitytools.Mode



class `abjad.tools.tonalitytools.Mode.Mode.Mode(arg)`
 New in version 2.0. Diatonic mode. Can be extended for nondiatonic mode.
 Modes with different ascending and descending forms not yet implemented.

Read-only Properties

`Mode.melodic_diatonic_interval_segment`
`Mode.mode_name`

Special Methods

```
Mode.__delattr__()  
x.__delattr__('name') <==> del x.name
```

Note: Inherited from `__builtin__.object`

```
Mode.__eq__(arg)
```

```
Mode.__ge__(arg)
```

Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mode.**__gt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

Mode.**__hash__**() $\leq \Rightarrow$ *hash(x)*

Note: Inherited from `__builtin__.object`

Mode.**__le__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mode.**__len__**()

Mode.**__lt__**(*arg*)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

Mode.**__ne__**(*arg*)

Mode.**__repr__**()

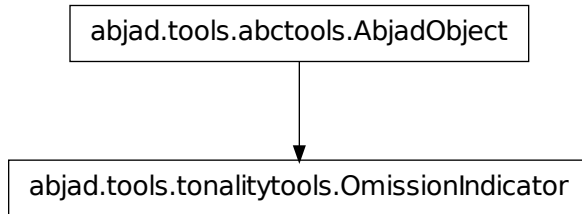
Mode.**__setattr__**()

`x.__setattr__('name', value) $\leq \Rightarrow$ x.name = value`

Note: Inherited from `__builtin__.object`

Mode.**__str__**()

tonalitytools.OmissionIndicator



class `abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator`, **OmissionIndicator**
 New in version 2.0. Indicator of missing chord tones.

Value object that can not be chnaged after instantiation.

Special Methods

`OmissionIndicator.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`OmissionIndicator.__eq__(arg)`
 True when `id(self)` equals `id(arg)`.
 Return boolean.

Note: Inherited from `abctools.AbjadObject`

`OmissionIndicator.__ge__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`OmissionIndicator.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`OmissionIndicator.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

OmissionIndicator.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

OmissionIndicator.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

OmissionIndicator.__ne__(arg)

True when `id(self)` does not equal `id(arg)`.

Return boolean.

Note: Inherited from `abctools.AbjadObject`

OmissionIndicator.__repr__()

Interpreter representation of Abjad object defaulting to class name, mandatory arguments, keyword arguments.

Return string.

Note: Inherited from `abctools.AbjadObject`

OmissionIndicator.__setattr__()

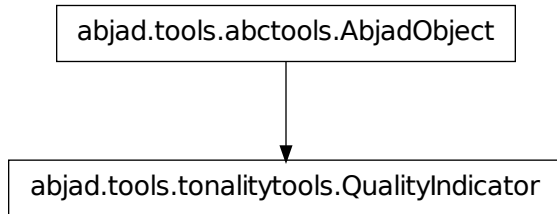
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

OmissionIndicator.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

tonalitytools.QualityIndicator



class `abjad.tools.tonalitytools.QualityIndicator.QualityIndicator` (*quality_string*)
 New in version 2.0. Indicator of chord quality, such as major, minor, dominant, diminished, etc.

Value object that can not be changed after instantiation.

Read-only Properties

`QualityIndicator.is_uppercase`

`QualityIndicator.quality_string`

Special Methods

`QualityIndicator.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`QualityIndicator.__eq__(arg)`

`QualityIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`QualityIndicator.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`QualityIndicator.__hash__()` `<==> hash(x)`

Note: Inherited from `__builtin__.object`

QualityIndicator.__le__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

QualityIndicator.__lt__(arg)

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

QualityIndicator.__ne__(arg)

QualityIndicator.__repr__()

QualityIndicator.__setattr__()

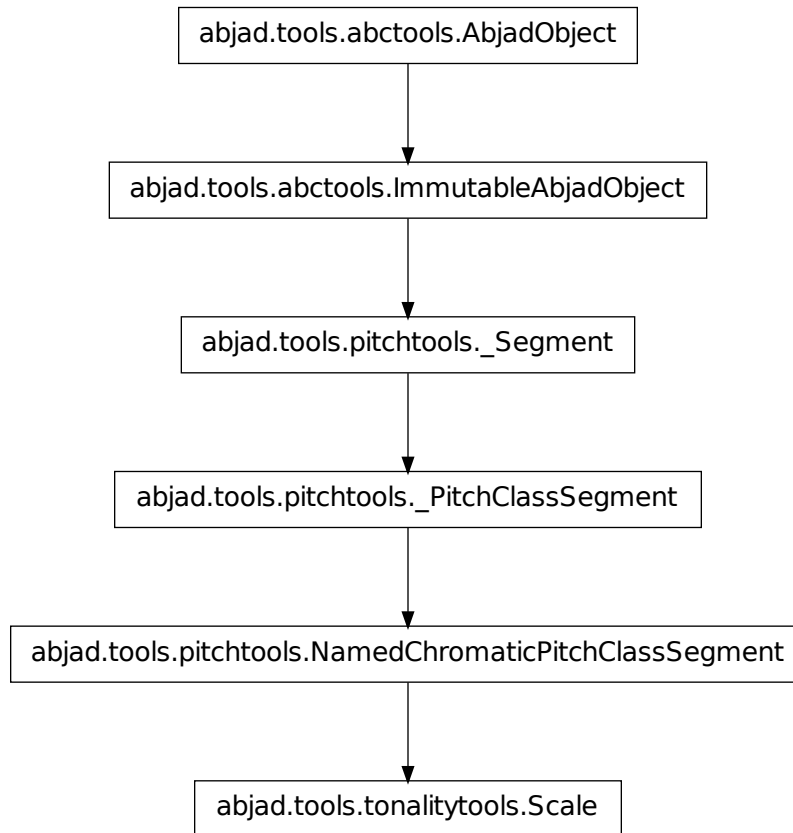
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

QualityIndicator.__str__() <==> `str(x)`

Note: Inherited from `__builtin__.object`

tonalitytools.Scale



class `abjad.tools.tonalitytools.Scale.Scale`.**Scale** (*args, **kwargs)
 New in version 2.0. Abjad model of diatonic scale.

Read-only Properties

`Scale.diatonic_interval_class_segment`

`Scale.dominant`

`Scale.inversion_equivalent_diatonic_interval_class_segment`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.key_signature`

`Scale.leading_tone`

`Scale.mediant`

`Scale.named_chromatic_pitch_class_set`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.named_chromatic_pitch_classes`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_class_segment`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_class_set`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.numbered_chromatic_pitch_classes`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.subdominant`

`Scale.submediant`

`Scale.superdominant`

`Scale.tonic`

Methods

`Scale.count` (*value*) → integer – return number of occurrences of value

Note: Inherited from `__builtin__.tuple`

`Scale.create_named_chromatic_pitch_set_in_pitch_range` (*pitch_range*)

`Scale.index` (*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises `ValueError` if the value is not present.

Note: Inherited from `__builtin__.tuple`

`Scale.is_equivalent_under_transposition` (*arg*)

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.named_chromatic_pitch_class_to_scale_degree(*args)`

`Scale.retrograde()`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.rotate(n)`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

`Scale.scale_degree_to_named_chromatic_pitch_class(*args)`

`Scale.transpose(melodic_diatonic_interval)`

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

Special Methods

`Scale.__add__(arg)`

Note: Inherited from `pitchtools._Segment`

`Scale.__contains__()`
`x.__contains__(y) <==> y in x`

Note: Inherited from `__builtin__.tuple`

`Scale.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`Scale.__eq__()`
`x.__eq__(y) <==> x==y`

Note: Inherited from `__builtin__.tuple`

`Scale.__ge__()`
`x.__ge__(y) <==> x>=y`

Note: Inherited from `__builtin__.tuple`

`Scale.__getitem__()`
`x.__getitem__(y) <==> x[y]`

Note: Inherited from `__builtin__.tuple`

Scale.__**getslice**__ (*start, stop*)

Note: Inherited from `pitchtools._Segment`

Scale.__**gt**__ ()
x.__**gt**__(y) <==> x>y

Note: Inherited from `__builtin__.tuple`

Scale.__**hash**__ () <==> *hash(x)*

Note: Inherited from `__builtin__.tuple`

Scale.__**iter**__ () <==> *iter(x)*

Note: Inherited from `__builtin__.tuple`

Scale.__**le**__ ()
x.__**le**__(y) <==> x<=y

Note: Inherited from `__builtin__.tuple`

Scale.__**len**__ () <==> *len(x)*

Note: Inherited from `__builtin__.tuple`

Scale.__**lt**__ ()
x.__**lt**__(y) <==> x<y

Note: Inherited from `__builtin__.tuple`

Scale.__**mul**__ (*n*)

Note: Inherited from `pitchtools._Segment`

Scale.__**ne**__ ()
x.__**ne**__(y) <==> x!=y

Note: Inherited from `__builtin__.tuple`

Scale.__**repr**__ ()

Scale.__rmul__(n)

Note: Inherited from `pitchtools._Segment`

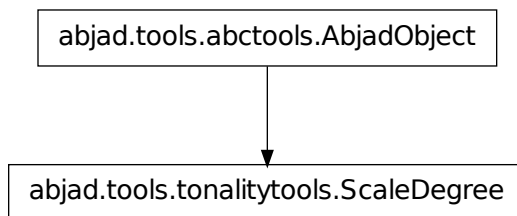
Scale.__setattr__()
 x.__setattr__('name', value) <==> x.name = value

Note: Inherited from `__builtin__.object`

Scale.__str__()

Note: Inherited from `pitchtools.NamedChromaticPitchClassSegment`

tonalitytools.ScaleDegree



class `abjad.tools.tonalitytools.ScaleDegree.ScaleDegree`(**args*)
 New in version 2.0. Abjad model of diatonic scale degrees 1, 2, 3, 4, 5, 6, 7 and also chromatic alterations including flat-2, flat-3, flat-6, etc.

Read-only Properties

`ScaleDegree.accidental`
 Read-only accidental applied to scale degree.

`ScaleDegree.name`
 Read-only name of scale degree.

`ScaleDegree.number`
 Read-only number of diatonic scale degree from 1 to 7, inclusive.

`ScaleDegree.roman_numeral_string`

`ScaleDegree.symbolic_string`

`ScaleDegree.title_string`

Methods

`ScaleDegree.apply_accidental(accidental)`
Apply accidental to self and emit new instance.

Special Methods

`ScaleDegree.__delattr__()`
`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`ScaleDegree.__eq__(arg)`

`ScaleDegree.__ge__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScaleDegree.__gt__(arg)`
Abjad objects by default do not implement this method.
Raise exception

Note: Inherited from `abctools.AbjadObject`

`ScaleDegree.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`ScaleDegree.__le__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScaleDegree.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`ScaleDegree.__ne__(arg)`

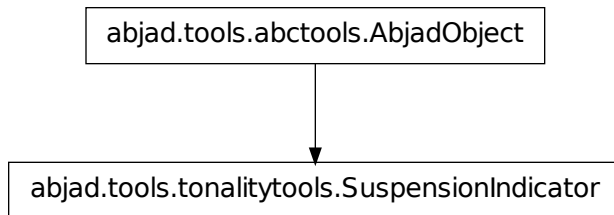
`ScaleDegree.__repr__()`

`ScaleDegree.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

ScaleDegree.__str__()

tonalitytools.SuspensionIndicator



class `abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator` (*)
 New in version 2.0. Indicator of 9-8, 7-6, 4-3, 2-1 and other types of suspension typical of, for example, the Bach chorales.

Value object that can not be changed after instantiation.

Read-only Properties

`SuspensionIndicator.chord_name`

`SuspensionIndicator.figured_bass_pair`

`SuspensionIndicator.figured_bass_string`

`SuspensionIndicator.is_empty`

`SuspensionIndicator.start`

`SuspensionIndicator.stop`

`SuspensionIndicator.title_string`

Special Methods

`SuspensionIndicator.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`SuspensionIndicator.__eq__(arg)`

`SuspensionIndicator.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__gt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception

Note: Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__hash__()` \Leftrightarrow `hash(x)`

Note: Inherited from `__builtin__.object`

`SuspensionIndicator.__le__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__lt__(arg)`
 Abjad objects by default do not implement this method.
 Raise exception.

Note: Inherited from `abctools.AbjadObject`

`SuspensionIndicator.__ne__(arg)`

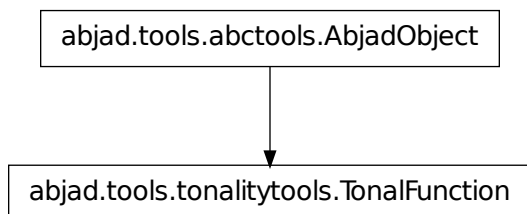
`SuspensionIndicator.__repr__()`

`SuspensionIndicator.__setattr__()`
`x.__setattr__('name', value) \Leftrightarrow x.name = value`

Note: Inherited from `__builtin__.object`

`SuspensionIndicator.__str__()`

tonalitytools.TonalFunction



class `abjad.tools.tonalitytools.TonalFunction.TonalFunction`.**TonalFunction** (*args)
New in version 2.0. Abjad model of functions in tonal harmony: I, I6, I64, V, V7, V43, V42, bII, bII6, etc., also i, i6, i64, v, v7, etc.

Value object that can not be changed after instantiation.

Read-only Properties

`TonalFunction.bass_scale_degree`

`TonalFunction.extent`

`TonalFunction.figured_bass_string`

`TonalFunction.inversion`

`TonalFunction.markup`

`TonalFunction.quality`

`TonalFunction.root_scale_degree`

`TonalFunction.scale_degree`

`TonalFunction.suspension`

`TonalFunction.symbolic_string`

Special Methods

`TonalFunction.__delattr__()`

`x.__delattr__('name') <==> del x.name`

Note: Inherited from `__builtin__.object`

`TonalFunction.__eq__(arg)`

`TonalFunction.__ge__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TonalFunction.__gt__(arg)`

Abjad objects by default do not implement this method.

Raise exception

Note: Inherited from `abctools.AbjadObject`

`TonalFunction.__hash__() <==> hash(x)`

Note: Inherited from `__builtin__.object`

`TonalFunction.__le__(arg)`

Abjad objects by default do not implement this method.

Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TonalFunction.__lt__(arg)`
Abjad objects by default do not implement this method.
Raise exception.

Note: Inherited from `abctools.AbjadObject`

`TonalFunction.__ne__(arg)`
`TonalFunction.__repr__()`
`TonalFunction.__setattr__()`
`x.__setattr__('name', value) <==> x.name = value`

Note: Inherited from `__builtin__.object`

`TonalFunction.__str__() <==> str(x)`

Note: Inherited from `__builtin__.object`

tonalitytools functions

tonalitytools.analyze_chord

`abjad.tools.tonalitytools.analyze_chord.analyze_chord(expr)`

New in version 2.0. Analyze *expr* and return chord class.

```
abjad> from abjad.tools import tonalitytools

abjad> chord = Chord([7, 10, 12, 16], (1, 4))
abjad> tonalitytools.analyze_chord(chord)
CDominantSeventhInSecondInversion
```

Return none when no tonal chord is understood.

```
abjad> chord = Chord(['c', 'cs', 'd'], (1, 4))
abjad> tonalitytools.analyze_chord(chord) is None
True
```

Raise tonal harmony error when chord can not analyze.

tonalitytools.analyze_incomplete_chord

`abjad.tools.tonalitytools.analyze_incomplete_chord.analyze_incomplete_chord(expr)`

New in version 2.0. Analyze *expr* and return chord class based on incomplete pitches.

```
abjad> from abjad.tools import tonalitytools
```

```
abjad> tonalitytools.analyze_incomplete_chord(Chord([7, 11], (1, 4)))
GMajorTriadInRootPosition

abjad> tonalitytools.analyze_incomplete_chord(Chord(['fs', 'g', 'b'], (1, 4)))
GMajorSeventhInSecondInversion
```

Return chord class.

tonalitytools.analyze_incomplete_tonal_function

abjad.tools.tonalitytools.analyze_incomplete_tonal_function.**analyze_incomplete_tonal_function**

New in version 2.0. Analyze tonal function of *expr* according to *key_signature*:

```
abjad> from abjad.tools import tonalitytools

abjad> chord = Chord("<c' e'>4")
abjad> key_signature = contexttools.KeySignatureMark('g', 'major')
abjad> tonalitytools.analyze_incomplete_tonal_function(chord, key_signature)
IVMajorTriadInRootPosition
```

Return tonal function.

tonalitytools.analyze_tonal_function

abjad.tools.tonalitytools.analyze_tonal_function.**analyze_tonal_function**(*expr*,
key_signature)

New in version 2.0. Analyze *expr* and return tonal function according to *key_signature*.

```
abjad> from abjad.tools import tonalitytools

abjad> chord = Chord(['ef', 'g', 'bf'], (1, 4))
abjad> key_signature = contexttools.KeySignatureMark('c', 'major')
abjad> tonalitytools.analyze_tonal_function(chord, key_signature)
FlatIIIMajorTriadInRootPosition
```

Return none when no tonal function is understood.

```
abjad> chord = Chord(['c', 'cs', 'd'], (1, 4))
abjad> key_signature = contexttools.KeySignatureMark('c', 'major')
abjad> tonalitytools.analyze_tonal_function(chord, key_signature) is None
True
```

Return tonal function or none.

tonalitytools.are_scalar_notes

abjad.tools.tonalitytools.are_scalar_notes.**are_scalar_notes**(**expr*)

New in version 2.0. True when notes in *expr* are scalar.

```
abjad> from abjad.tools import tonalitytools

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> tonalitytools.are_scalar_notes(t[:])
True
```

Otherwise false.

```
abjad> tonalitytools.are_scalar_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_scalar()` to `tonalitytools.are_scalar_notes()`.

tonalitytools.are_stepwise_ascending_notes

`abjad.tools.tonalitytools.are_stepwise_ascending_notes`.**are_stepwise_ascending_notes**(**expr*)
New in version 2.0. True when notes in *expr* are stepwise ascending.

```
abjad> from abjad.tools import tonalitytools

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> tonalitytools.are_stepwise_ascending_notes(t[:])
True
```

Otherwise false.

```
abjad> tonalitytools.are_stepwise_ascending_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise_ascending()` to `tonalitytools.are_stepwise_ascending_notes()`.

tonalitytools.are_stepwise_descending_notes

`abjad.tools.tonalitytools.are_stepwise_descending_notes`.**are_stepwise_descending_notes**(**expr*)
New in version 2.0. True when notes in *expr* are stepwise descending:

```
abjad> from abjad.tools import tonalitytools

abjad> notes = [Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
abjad> t = Staff(list(reversed(notes)))
abjad> tonalitytools.are_stepwise_descending_notes(t[:])
True
```

Otherwise false:

```
abjad> tonalitytools.are_stepwise_descending_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise_descending()` to `tonalitytools.are_stepwise_descending_notes()`.

tonalitytools.are_stepwise_notes

`abjad.tools.tonalitytools.are_stepwise_notes`.**are_stepwise_notes**(**expr*)
New in version 2.0. True when notes in *expr* are stepwise.

```
abjad> from abjad.tools import tonalitytools

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> tonalitytools.are_stepwise_notes(t[:])
True
```

Otherwise false.

```
abjad> tonalitytools.are_stepwise_notes(Note("c'4"), Note("c'4"))
False
```

Changed in version 2.0: renamed `tonalitytools.are_stepwise()` to `tonalitytools.are_stepwise_notes()`.

tonalitytools.chord_class_cardinality_to_extent

```
abjad.tools.tonalitytools.chord_class_cardinality_to_extent.chord_class_cardinality_to_extent
..versionadded:: 2.0
```

Change integer chord class *cardinality* to integer chord class extent:

```
abjad> from abjad.tools import tonalitytools

abjad> tonalitytools.chord_class_cardinality_to_extent(4)
7
```

The function above indicates that a tertian chord with 4 unique pitches qualifies as a seventh chord.

tonalitytools.chord_class_extent_to_cardinality

```
abjad.tools.tonalitytools.chord_class_extent_to_cardinality.chord_class_extent_to_cardinality
..versionadded:: 2.0
```

Change integer chord class *extent* to integer chord class cardinality:

```
abjad> from abjad.tools import tonalitytools

abjad> tonalitytools.chord_class_extent_to_cardinality(7)
4
```

The call above shows that a seventh chord comprises 4 unique pitch-classes.

tonalitytools.chord_class_extent_to_extent_name

```
abjad.tools.tonalitytools.chord_class_extent_to_extent_name.chord_class_extent_to_extent_name
New in version 2.0. Change integer chord class extent to extent name.
```

```
abjad> from abjad.tools import tonalitytools

abjad> tonalitytools.chord_class_extent_to_extent_name(7)
'seventh'
```

The call above shows that a tertian chord subtending 7 staff spaces qualifies as a seventh chord.

tonalitytools.diatonic_interval_class_segment_to_chord_quality_string

```
abjad.tools.tonalitytools.diatonic_interval_class_segment_to_chord_quality_string.diatonic_interval_class_segment_to_chord_quality_string
New in version 2.0. Change diatonic interval-class segment dic_seg to chord quality string:
```

```
abjad> from abjad.tools import tonalitytools
```



```
abjad> dic_seg = pitchtools.InversionEquivalentDiatonicIntervalClassSegment([
...     pitchtools.InversionEquivalentDiatonicIntervalClass('major', 3),
...     pitchtools.InversionEquivalentDiatonicIntervalClass('minor', 3),])
abjad> tonalitytools.diatonic_interval_class_segment_to_chord_quality_string(dic_seg)
'major'
```

Todo

Implement `diatonic_interval_class_set_to_chord_quality_string()`.

tonalitytools.is_neighbor_note

`abjad.tools.tonalitytools.is_neighbor_note.is_neighbor_note(note)`

New in version 2.0. True when *note* is preceded by a stepwise interval in one direction and followed by a stepwise interval in the other direction. Otherwise false.

```
abjad> from abjad.tools import tonalitytools

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> for note in t:
...     print '%s\t%s' % (note, tonalitytools.is_neighbor_note(note))
...
c'8      False
d'8      False
e'8      False
f'8      False
```

Return boolean.

tonalitytools.is_passing_tone

`abjad.tools.tonalitytools.is_passing_tone.is_passing_tone(note)`

New in version 2.0. True when *note* is both preceded and followed by scalewise sibling notes. Otherwise false.

```
abjad> from abjad.tools import tonalitytools

abjad> t = Staff("c'8 d'8 e'8 f'8")
abjad> for note in t:
...     print '%s\t%s' % (note, tonalitytools.is_passing_tone(note))
...
c'8      False
d'8      True
e'8      True
f'8      False
```

Return boolean.

tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale

`abjad.tools.tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale.is_unlikely_melodic_diatonic_interval_in_chorale(mdi)`

New in version 2.0. True when *mdi* is unlikely melodic diatonic interval in JSB chorale.

```
abjad> from abjad.tools import tonalitytools
```

```
abjad> mdi = pitchtools.MelodicDiatonicInterval('major', 7)
abjad> tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale(mdi)
True
```

Otherwise False.

```
abjad> mdi = pitchtools.MelodicDiatonicInterval('major', 2)
abjad> tonalitytools.is_unlikely_melodic_diatonic_interval_in_chorale(mdi)
False
```

Return boolean.

tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale

abjad.tools.tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale.**make_a**

New in version 2.0. Construct one up-down period of scale according to *key_signature*:

```
abjad> from abjad.tools import tonalitytools
```

```
abjad> score = tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale(contextto
```

```
abjad> f(score)
```

```
\new Score \with {
  tempoWholesPerMinute = #(ly:make-moment 30 1)
} <<
```

```
  \new Staff {
    \key e \major
    e'8
    fs'8
    gs'8
    a'8
    b'8
    cs''8
    ds''8
    e''8
    ds''8
    cs''8
    b'8
    a'8
    gs'8
    fs'8
    e'4
```

```
  }
>>
```

Changed in version 2.0: renamed `construct.scale_period()` to `tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale()`. Changed in version 2.0: renamed `leaftools.make_all_notes_in_ascending_and_descending_diatonic_scale()` to `tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale()`.

tonalitytools.make_first_n_notes_in_ascending_diatonic_scale

abjad.tools.tonalitytools.make_first_n_notes_in_ascending_diatonic_scale.**make_first_n_notes**

Construct *count* notes with *written_duration* according to *key_signature*:

```
abjad> from abjad.tools import tonalitytools

abjad> tonalitytools.make_first_n_notes_in_ascending_diatonic_scale(4)
[Note("c'8"), Note("d'8"), Note("e'8"), Note("f'8")]
```

Allow nonassignable *written_duration*:

```
abjad> staff = Staff(tonalitytools.make_first_n_notes_in_ascending_diatonic_scale(2, (5, 16)))
abjad> f(staff)
\new Staff {
    c'4 ~
    c'16
    d'4 ~
    d'16
}
```

New in version 2.0: Optional *key_signature* keyword parameter.Changed in version 2.0: re-named `leaftools.make_first_n_notes_in_ascending_diatonic_scale()` to `tonalitytools.make_first_n_notes_in_ascending_diatonic_scale()`.

BIBLIOGRAPHY

- [Adan2006] Víctor Adán. Music <-> Geometry <-> Meta-Music. Draft February 12, 2006.
- [AgonAssayagBresson2006] Carlos Agon, Gérard Assayag, Jean Bresson. The OM Composer's Book 1. Éditions Delatour, Paris. 2006.
- [AgonHaddadAssayag2002] Carlos Agon, Karim Haddad & Gerard Assayag. Représentation et rendu de structures rythmiques. Journées d'Informatique Musicale, 9th ed., Marseille, 29 - 31 May 2002.
- [Alegant1993] Brian Alegant. The seventy-seven partitions of the aggregate: Analytical and theoretical implications. Doctoral Dissertation. The University of Rochester, Eastman School of Music. 1993.
- [Ariza2005] Christopher Ariza. An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL. Dissertation.com, Boca Raton. 2005.
- [BacaAdan2007] Trevor Bača & Víctor Adán. Cuepatlahto and Lascaux: two approaches to the formalized control of musical score. Draft June 7, 2007.
- [BressonAgonAssayag2008] Jean Bresson, Carlos Agon, Gérard Assayag. The OM Composer's Book 2. Éditions Delatour, Paris. 2008.
- [Carter2002] Eliot Carter. Harmony Book. Nicholas Hopkins and John F. Link, eds. Carl Fischer, New York. 2002.
- [Haddad] Karim Haddad. Le Temps comme Territoire: pour une géographie temporelle.
- [Kampela1998] Arthur Kampela. Uma Faca Só Lâmina. Doctoral Dissertation. Columbia University, NY, NY. 1998.
- [Malt2008] Mikhail Malt. Some Considerations on Brian Ferneyhough's Musical Language Through His Use of CAC – Part I: Time and Rhythmic Structures. In [BressonAgonAssayag2008].
- [Morris1987] Robert Morris. Composition with Pitch-Classes. Yale University Press, New Haven. 1987.
- [Nauert1997] Paul Nauert. Timespan Formation in Nonmetric, Posttonal Music. Doctoral Dissertation. Columbia University, NY, NY. 1997.
- [NienhuysNieuwenhuizen2003] Han-Wen Nienhuys & Jan Nieuwenhuizen. Lilypond: A system for automated music engraving. Proceedings of the XIV Colloquium on Musical Informatics. Firenze, Italy. May 8 - 10, 2003.
- [Ross1987] Ted Ross. Teach Yourself The Art of Music Engraving and Processing. Hansen House, Miami Beach. 1987.
- [Selfridge-Field1997] Eleanor Selfridge-Field, ed. Beyond MIDI: The Handbook of Musical Codes. The MIT Press. Cambridge, Massachusetts. 1997.
- [Valle] Andrea Valle. GeoGraphy: Notazione musicale e composizione algoritmica. Centro Interdipartimentale di Ricerca sulla Multimedialità e l'Audiovisivo. Università degli Studi di Torino.
- [WulfsonBarrettWinter] Harris Wulfson, G. Douglas Barrett & Michael Winter. Automatic Notation Generators.

INDEX

Symbols

- `__abs__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 1660
- `__abs__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 1664
- `__abs__()` (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 992
- `__abs__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 994
- `__abs__()` (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1009
- `__abs__()` (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1011
- `__abs__()` (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1014
- `__abs__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1016
- `__abs__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1030
- `__abs__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1044
- `__abs__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method), 1055
- `__abs__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1057
- `__abs__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1076
- `__abs__()` (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1078
- `__abs__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1082
- `__abs__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1084
- `__abs__()` (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch method), 1097
- `__abs__()` (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1099
- `__abs__()` (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch method), 1123
- `__abs__()` (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.NamedDiatonicPitchClass method), 1126
- `__abs__()` (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch method), 1128
- `__abs__()` (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1130
- `__abs__()` (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1149
- `__abs__()` (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass method), 1151
- `__add__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), 299
- `__add__()` (abjad.tools.containertools.Container.Container.Container method), 397
- `__add__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 345
- `__add__()` (abjad.tools.contexttools.Context.Context.Context method), 354
- `__add__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 381
- `__add__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 384
- `__add__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1629
- `__add__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 1660
- `__add__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 1664
- `__add__()` (abjad.tools.gracetools.Grace.Grace.Grace method), 422
- `__add__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 633
- `__add__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 811
- `__add__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 816
- `__add__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 829
- `__add__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 841
- `__add__()` (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2032
- `__add__()` (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2050

__and__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.Crawler.ClassCrawler.ClassCrawler method), 1072

__and__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.Crawler.ClassCrawler.ClassCrawler method), 1091

__and__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.Crawler.ClassCrawler.ClassCrawler method), 1106

__and__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.Crawler.FunctionCrawler.FunctionCrawler method), 1114

__and__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.Crawler.ClassCrawler.ClassCrawler method), 1141

__and__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1215

__and__() (abjad.tools.resttools.Rest.Rest.Rest method), __call__() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 1220

__and__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass method), 1820

__and__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1822

__and__() (abjad.tools.skiptools.Skip.Skip.Skip method), __call__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 1310

__and__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 2076

__call__() (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint method), 1991

__call__() (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint method), 2000

__call__() (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint method), 2002

__call__() (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method), 2005

__call__() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint method), 2008

__call__() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint method), 2011

__call__() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 341

__call__() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 359

__call__() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method), 362

__call__() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 367

__call__() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method), 371

__call__() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 376

__call__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 381

__call__() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method), 390

__call__() (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method), 1633

__call__() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method), 1635

__call__() (abjad.tools.instrumenttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint method), 468

__call__() (abjad.tools.instrumenttools.GlobalConstraint.GlobalConstraint.GlobalConstraint method), 475

__call__() (abjad.tools.instrumenttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint method), 482

__call__() (abjad.tools.instrumenttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method), 489

__call__() (abjad.tools.instrumenttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint method), 496

__call__() (abjad.tools.instrumenttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint method), 503

__call__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 510

__call__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 517

__call__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524

__call__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531

__call__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538

__call__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545

__call__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552

__call__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 559

__call__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566

__call__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573

__call__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet.marktools.Annotation.Annotation.Annotation method), 580

__call__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.marktools.Articulation.Articulation.Articulation method), 587

__call__() (abjad.tools.instrumenttools.Flute.Flute.Flute __call__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 594 method), 855

__call__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.marktools.BendAfter.BendAfter.BendAfter method), 601 method), 858

__call__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.marktools.LilyPondCommandMark.LilyPondComm method), 608 method), 862

__call__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar __call__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.Li method), 615 method), 866

__call__() (abjad.tools.instrumenttools.Harp.Harp.Harp __call__() (abjad.tools.marktools.Mark.Mark.Mark method), 622 method), 868

__call__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.marktools.StemTremolo.StemTremolo.StemTremolo method), 629 method), 871

__call__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba __call__() (abjad.tools.markuptools.Markup.Markup.Markup method), 641 method), 892

__call__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.QGridQuantizer.QGridQuantizer.Q method), 648 method), 2060

__call__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe __call__() (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.Sc method), 655 method), 1241

__call__() (abjad.tools.instrumenttools.Piano.Piano.Piano __call__() (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.Stri method), 662 method), 1244

__call__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo __call__() (abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.Tw method), 669 method), 1246

__call__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.BurnishedTimeTokenMaker.Burnish method), 676 method), 1896

__call__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.IncisedTimeTokenMaker.IncisedTim method), 683 method), 1899

__call__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.timetokentools.NoteFilledTimeTokenMaker.NoteFill method), 690 method), 1909

__call__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.timetokentools.OutputBurnishedSignalFilledTimeTo method), 697 method), 1913

__call__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.timetokentools.OutputIncisedNoteFilledTimeTokenM method), 704 method), 1917

__call__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice __call__() (abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenM method), 711 method), 1921

__call__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet __call__() (abjad.tools.timetokentools.OutputIncisedTimeTokenMaker.Outp method), 718 method), 1901

__call__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba __call__() (abjad.tools.timetokentools.RestFilledTimeTokenMaker.RestFille method), 725 method), 1924

__call__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.timetokentools.SignalFilledTimeTokenMaker.Signal method), 732 method), 1927

__call__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone __call__() (abjad.tools.timetokentools.TimeTokenMaker.TimeTokenMaker.T method), 739 method), 1903

__call__() (abjad.tools.instrumenttools.Viola.Viola.Viola __call__() (abjad.tools.timetokentools.TokenBurnishedSignalFilledTimeTok method), 746 method), 1931

__call__() (abjad.tools.instrumenttools.Violin.Violin.Violin __call__() (abjad.tools.timetokentools.TokenIncisedNoteFilledTimeTokenM method), 753 method), 1935

__call__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone __call__() (abjad.tools.timetokentools.TokenIncisedRestFilledTimeTokenM method), 760 method), 1939

__call__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser __call__() (abjad.tools.timetokentools.TokenIncisedTimeTokenMaker.Token method), 1705 method), 1906

__cmp__()	(abjad.tools.datastructretools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1626	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 355
__cmp__()	(abjad.tools.documentationtools.InheritanceGraphInheritanceGraphInheritanceGraph.method), 1651	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 384
__cmp__()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector()HarmonicChromaticIntervalClassVector()HarmonicChromaticIntervalClassVector().method), 998	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1626
__cmp__()	(abjad.tools.pitchtools.HarmonicChromaticIntervalSetHarmonicChromaticIntervalSetHarmonicChromaticIntervalSet.method), 1006	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1629
__cmp__()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSetHarmonicDiatonicIntervalClassSetHarmonicDiatonicIntervalClassSet.method), 1019	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1651
__cmp__()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalSetHarmonicDiatonicIntervalSetHarmonicDiatonicIntervalSet.method), 1027	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 422
__cmp__()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSetInversionEquivalentChromaticIntervalClassSetInversionEquivalentChromaticIntervalClassSet.method), 1036	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 632
__cmp__()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSetInversionEquivalentChromaticIntervalClassSetInversionEquivalentChromaticIntervalClassSet.method), 1041	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 812
__cmp__()	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSetInversionEquivalentDiatonicIntervalClassSetInversionEquivalentDiatonicIntervalClassSet.method), 1052	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 816
__cmp__()	(abjad.tools.pitchtools.MelodicChromaticIntervalClassVectorMelodicChromaticIntervalClassVectorMelodicChromaticIntervalClassVector.method), 1065	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 829
__cmp__()	(abjad.tools.pitchtools.MelodicChromaticIntervalSetMelodicChromaticIntervalSetMelodicChromaticIntervalSet.method), 1072	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 841
__cmp__()	(abjad.tools.pitchtools.MelodicDiatonicIntervalSetMelodicDiatonicIntervalSetMelodicDiatonicIntervalSet.method), 1091	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 2022
__cmp__()	(abjad.tools.pitchtools.NamedChromaticPitchClassSetNamedChromaticPitchClassSetNamedChromaticPitchClassSet.method), 1106	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 2050
__cmp__()	(abjad.tools.pitchtools.NamedChromaticPitchSetNamedChromaticPitchSetNamedChromaticPitchSet.method), 1114	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 897
__cmp__()	(abjad.tools.pitchtools.NamedChromaticPitchVectorNamedChromaticPitchVectorNamedChromaticPitchVector.method), 1118	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 912
__cmp__()	(abjad.tools.pitchtools.NumberedChromaticPitchClassSetNumberedChromaticPitchClassSetNumberedChromaticPitchClassSet.method), 1141	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 923
__cmp__()	(abjad.tools.pitchtools.NumberedChromaticPitchClassVectorNumberedChromaticPitchClassVectorNumberedChromaticPitchClassVector.method), 1146	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 932
__cmp__()	(abjad.tools.quantizationtools.QGridSearchTreeQGridSearchTreeQGridSearchTree.method), 2065	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1730
__cmp__()	(abjad.tools.quantizationtools.QGridTempoLookupQGridTempoLookupQGridTempoLookup.method), 2069	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 999
__cmp__()	(abjad.tools.timeintervaltools.TimeIntervalTreeDictionaryTimeIntervalTreeDictionaryTimeIntervalTree.method), 1880	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1002
__cmp__()	(abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass() method), 2076	__cmp__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1006
__complex__()	(abjad.tools.durationtools.Duration.Duration.Durations() method), 1660	__complex__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1019
__complex__()	(abjad.tools.durationtools.Offset.Offset.Offset_contains() method), 1664	__complex__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1023
__contains__()	(abjad.tools.chordtools.Chord.Chord.Chord __contains__() method), 228	__contains__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1027
__contains__()	(abjad.tools.containertools.Cluster.Cluster.Cluster_contains() method), 299	__contains__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1032
__contains__()	(abjad.tools.containertools.Container.Container.Container_contains() method), 307	__contains__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1037
__contains__()	(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventoryClefMarkInventory.method), 345	__contains__()	(abjad.tools.immutabletools.ImmutableDictionaryImmutableDictionaryContext.ContextContext.method), 1042

[__contains__\(\) \(abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method\), 1503](#)
[__contains__\(\) \(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method\), 1527](#)
[__contains__\(\) \(abjad.tools.stafftools.Staff.Staff.Staff method\), 1538](#)
[__contains__\(\) \(abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method\), 1548](#)
[__contains__\(\) \(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method\), 1843](#)
[__contains__\(\) \(abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method\), 1856](#)
[__contains__\(\) \(abjad.tools.timeintervaltools.TimeIntervalTreeDeletion\(TimeIntervalTreeDeletion method\), 1880](#)
[__contains__\(\) \(abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method\), 2076](#)
[__contains__\(\) \(abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method\), 2080](#)
[__contains__\(\) \(abjad.tools.tonalitytools.Scale.Scale.Scale method\), 2096](#)
[__contains__\(\) \(abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method\), 1573](#)
[__contains__\(\) \(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method\), 1584](#)
[__contains__\(\) \(abjad.tools.voicetools.Voice.Voice.Voice method\), 1605](#)
[__del__\(\) \(abjad.tools.documentationtools.Pipe.Pipe.Pipe method\), 1657](#)
[__delattr__\(\) \(abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject method\), 1612](#)
[__delattr__\(\) \(abjad.tools.abctools.AttributeEqualityAbjadObject\(AttributeEqualityAbjadObject method\), 1614](#)
[__delattr__\(\) \(abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject method\), 1615](#)
[__delattr__\(\) \(abjad.tools.abctools.SortableAttributeEqualityAbjadObject\(SortableAttributeEqualityAbjadObject method\), 1617](#)
[__delattr__\(\) \(abjad.tools.chordtools.Chord.Chord.Chord method\), 228](#)
[__delattr__\(\) \(abjad.tools.componenttools.Component.Component.Component method\), 237](#)
[__delattr__\(\) \(abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint method\), 1991](#)
[__delattr__\(\) \(abjad.tools.constrainttools.Domain.Domain.Domain method\), 1993](#)
[__delattr__\(\) \(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver method\), 1997](#)
[__delattr__\(\) \(abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint method\), 2000](#)
[__delattr__\(\) \(abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint method\), 2002](#)
[__delattr__\(\) \(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method\), 2005](#)
[__delattr__\(\) \(abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint method\), 2008](#)
[__delattr__\(\) \(abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint method\), 2011](#)
[__delattr__\(\) \(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver method\), 2014](#)
[__delattr__\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster method\), 299](#)
[__delattr__\(\) \(abjad.tools.containertools.Container.Container.Container.Container method\), 307](#)
[__delattr__\(\) \(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method\), 342](#)
[__delattr__\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method\), 345](#)
[__delattr__\(\) \(abjad.tools.contexttools.Dictionary\(Dictionary method\), 355](#)
[__delattr__\(\) \(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method\), 359](#)
[__delattr__\(\) \(abjad.tools.contexttools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method\), 362](#)
[__delattr__\(\) \(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method\), 367](#)
[__delattr__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method\), 372](#)
[__delattr__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method\), 376](#)
[__delattr__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method\), 381](#)
[__delattr__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method\), 384](#)
[__delattr__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method\), 390](#)
[__delattr__\(\) \(abjad.tools.abctools.AttributeEqualityAbjadObject\(AttributeEqualityAbjadObject method\), 1622](#)
[__delattr__\(\) \(abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject method\), 1626](#)
[__delattr__\(\) \(abjad.tools.abctools.SortableAttributeEqualityAbjadObject\(SortableAttributeEqualityAbjadObject method\), 1629](#)
[__delattr__\(\) \(abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method\), 1633](#)
[__delattr__\(\) \(abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method\), 1635](#)
[__delattr__\(\) \(abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method\), 1637](#)
[__delattr__\(\) \(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter method\), 1640](#)
[__delattr__\(\) \(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method\), 1644](#)
[__delattr__\(\) \(abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter method\), 1647](#)
[__delattr__\(\) \(abjad.tools.documentationtools.GlobalReferenceCrawler.GlobalReferenceCrawler.GlobalReferenceCrawler method\), 1651](#)
[__delattr__\(\) \(abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method\), 1654](#)

__delattr__() (abjad.tools.documentationtools.Pipe.Pipe.Pipe __delattr__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 1657 method), 594

__delattr__() (abjad.tools.durationtools.Duration.Duration.Duration __delattr__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchH method), 1660 method), 601

__delattr__() (abjad.tools.durationtools.Offset.Offset.Offset __delattr__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glock method), 1665 method), 608

__delattr__() (abjad.tools.gracetools.Grace.Grace.Grace __delattr__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 422 method), 615

__delattr__() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion __delattr__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 433 method), 622

__delattr__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute __delattr__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsich method), 440 method), 629

__delattr__() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone __delattr__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentIn method), 447 method), 632

__delattr__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone __delattr__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 454 method), 641

__delattr__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet __delattr__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopra method), 461 method), 648

__delattr__() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone __delattr__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 468 method), 655

__delattr__() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice __delattr__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 475 method), 662

__delattr__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet __delattr__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 482 method), 669

__delattr__() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute __delattr__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoS method), 489 method), 676

__delattr__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone __delattr__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxo method), 496 method), 683

__delattr__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone __delattr__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.Sopr method), 503 method), 690

__delattr__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice __delattr__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophon method), 510 method), 697

__delattr__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon __delattr__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone. method), 517 method), 704

__delattr__() (abjad.tools.instrumenttools.Cello.Cello.Cello __delattr__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVo method), 524 method), 711

__delattr__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA __delattr__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 531 method), 718

__delattr__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass __delattr__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 538 method), 725

__delattr__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet __delattr__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPerc method), 545 method), 732

__delattr__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute __delattr__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibrapho method), 552 method), 739

__delattr__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone __delattr__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 559 method), 746

__delattr__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon __delattr__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 566 method), 753

__delattr__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice __delattr__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophon method), 573 method), 760

__delattr__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet __delattr__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication. method), 580 method), 1698

__delattr__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn __delattr__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 587 method), 769

__delattr__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken, method), 808

__delattr__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock, method), 812

__delattr__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock, method), 816

__delattr__() (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock, method), 820

__delattr__() (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken, method), 821

__delattr__() (abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock, method), 824

__delattr__() (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock, method), 826

__delattr__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile, method), 829

__delattr__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken, method), 832

__delattr__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken, method), 834

__delattr__() (abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock, method), 837

__delattr__() (abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock, method), 838

__delattr__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock, method), 841

__delattr__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser, method), 1705

__delattr__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics, method), 2022

__delattr__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender, method), 2027

__delattr__() (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen, method), 2031

__delattr__() (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace, method), 2036

__delattr__() (abjad.tools.lyricstools.LyricText.LyricText.LyricText, method), 2040

__delattr__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics, method), 2050

__delattr__() (abjad.tools.marktools.Annotation.Annotation.Annotation, method), 847

__delattr__() (abjad.tools.marktools.Articulation.Articulation.Articulation, method), 851

__delattr__() (abjad.tools.marktools.BarLine.BarLine.BarLine, method), 855

__delattr__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter, method), 859

__delattr__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark, method), 862

__delattr__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment, method), 866

__delattr__() (abjad.tools.marktools.Mark.Mark.Mark, method), 868

__delattr__() (abjad.tools.marktools.AbjadRevisonToken.AbjadRevisonToken.AbjadRevisonToken, method), 872

__delattr__() (abjad.tools.marktools.Markup.Markup.Markup, method), 892

__delattr__() (abjad.tools.marktools.MarkupCommand.MarkupCommand.MarkupCommand, method), 894

__delattr__() (abjad.tools.marktools.MarkupInventory.MarkupInventory.MarkupInventory, method), 897

__delattr__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure, method), 923

__delattr__() (abjad.tools.measuretools.Measure.Measure.Measure, method), 932

__delattr__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic, method), 969

__delattr__() (abjad.tools.notetools.LilyPondNoteToken.LilyPondNoteToken.LilyPondNoteToken, method), 974

__delattr__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead, method), 978

__delattr__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray, method), 1730

__delattr__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell, method), 1733

__delattr__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn, method), 1735

__delattr__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow, method), 1738

__delattr__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental, method), 990

__delattr__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval, method), 992

__delattr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass, method), 994

__delattr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector, method), 999

__delattr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment, method), 1002

__delattr__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet, method), 1006

__delattr__() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval, method), 1009

__delattr__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass, method), 1011

__delattr__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval, method), 1014

__delattr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass, method), 1016

__delattr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet, method), 1020

__delattr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment, method), 1023

__delattr__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet, method), 1027

__delattr__(abjad.tools.schemetools.SchemeVector.SchemeVectorSchemeVectorDynamicTextSpanner.DynamicTextSpanner method),	1237
__delattr__(abjad.tools.schemetools.SchemeVectorConstantSchemeVectorConstantSchemeVectorGlissandoSpanner.GlissandoSpanner method),	1239
__delattr__(abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplateScoreTemplateHairpinSpanner.HairpinSpanner.HairpinSpanner method),	1241
__delattr__(abjad.tools.scoretemplatetools.StringQuartetScoreTemplateStringQuartetScoreTemplateHidStrStaffSpanner.StaffSpanner method),	1244
__delattr__(abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplateTwoStaffPianoScoreTemplateTernHornBassStaffSpanner.StaffSpanner method),	1246
__delattr__(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method),	1256
__delattr__(abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifierInstrumentationMetricGridSpanner.MetricGridSpanner method),	1259
__delattr__(abjad.tools.scoretools.Performer.Performer.Performer() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner method),	1262
__delattr__(abjad.tools.scoretools.PerformerInventory.PerformerInventoryPerformerInventoryOctavationSpanner.OctavationSpanner method),	1265
__delattr__(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method),	1277
__delattr__(abjad.tools.scoretools.Score.Score.Score() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner method),	1287
__delattr__(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method),	1298
__delattr__(abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList() (abjad.tools.spannertools.Spanner.Spanner.Spanner method),	1745
__delattr__(abjad.tools.sequencetools.CyclicMatrix.CyclicMatrixCyclicMatrix() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner method),	1749
__delattr__(abjad.tools.sequencetools.CyclicTree.CyclicTreeCyclicTree() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method),	1759
__delattr__(abjad.tools.sequencetools.CyclicTuple.CyclicTupleCyclicTuple() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method),	1761
__delattr__(abjad.tools.sequencetools.Matrix.Matrix.Matrix() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method),	1764
__delattr__(abjad.tools.sequencetools.Tree.Tree.Tree() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method),	1773
__delattr__(abjad.tools.sievetools.ResidueClass.ResidueClassResidueClass (abjad.tools.stafftools.Staff.Staff.Staff method),	1820
__delattr__(abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpressionResidueClassExpression (abjad.tools.stafftools.TieSpanner.TieSpanner method),	1823
__delattr__(abjad.tools.skiptools.Skip.Skip.Skip() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method),	1310
__delattr__(abjad.tools.spannertools.BeamSpanner.BeamSpannerBeamSpanner() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method),	1322
__delattr__(abjad.tools.spannertools.BracketSpanner.BracketSpannerBracketSpanner() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method),	1330
__delattr__(abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpannerComplexBeamSpanner() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method),	1339
__delattr__(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpannerCrescendoSpanner() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method),	1349
__delattr__(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpannerDecrescendoSpanner() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method),	1359
__delattr__(abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpannerDuratedComplexBeamSpanner() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method),	1369

__delitem__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.CyclicList.CyclicList.CyclicList method), 633

__delitem__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock.abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 812

__delitem__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock.abjad.tools.stafftools.Staff.Staff.Staff method), 817

__delitem__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 829

__delitem__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock.abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 841

__delitem__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics) (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 2022

__delitem__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics) __delitem__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 2050

__delitem__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory.abjad.tools.voice.Voice.Voice.Voice method), 898

__delitem__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure.abjad.tools.measuretools.Measure.Measure.Measure method), 912

__delitem__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.abjad.tools.measuretools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 923

__delitem__() (abjad.tools.measuretools.Measure.Measure.Measure) __delitem__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 932

__delitem__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 999

__delitem__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1042

__delitem__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1052

__delitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1065

__delitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1119

__delitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 1146

__delitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1155

__delitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1162

__delitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1169

__delitem__() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree method), 2065

__delitem__() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup.QGridTempoLookup method), 2070

__delitem__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff) (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 1256

__delitem__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory) (abjad.tools.durationtools.Duration.Duration.Duration method), 1265

__delitem__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff) (abjad.tools.durationtools.Offset.Offset.Offset method), 1277

__delitem__() (abjad.tools.scoretools.Score.Score.Score) __divmod__() (abjad.tools.durationtools.Duration.Duration.Duration method), 1287

__delitem__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup) __divmod__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1298

__eq__() (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject.__eq__() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark method), 1612 method), 390

__eq__() (abjad.tools.abctools.AttributeEquality.AbjadObject.AttributeEquality.AbjadObject.AttributeEquality.AbjadObject method), 1614 method), 1622

__eq__() (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableDictionary.ImmutableDictionary method), 1615 method), 1626

__eq__() (abjad.tools.abctools.SortableAttributeEquality.AbjadObject.SortableAttributeEquality.AbjadObject.SortableAttributeEquality.AbjadObject method), 1617 method), 1630

__eq__() (abjad.tools.chordtools.Chord.Chord.Chord __eq__() (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method), 228 method), 1633

__eq__() (abjad.tools.componenttools.Component.Component.Component __eq__() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator method), 237 method), 1635

__eq__() (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint method), 1991 method), 1637

__eq__() (abjad.tools.constrainttools.Domain.Domain.Domain __eq__() (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter method), 1993 method), 1640

__eq__() (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver method), 1997 method), 1642

__eq__() (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint.GlobalConstraint method), 2000 method), 1644

__eq__() (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint method), 2002 method), 1647

__eq__() (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method), 2005 method), 1652

__eq__() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint method), 2008 method), 1654

__eq__() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint method), 2011 method), 1657

__eq__() (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver method), 2015 method), 1661

__eq__() (abjad.tools.containertools.Cluster.Cluster.Cluster __eq__() (abjad.tools.durationtools.Offset.Offset.Offset method), 299 method), 1665

__eq__() (abjad.tools.containertools.Container.Container.Container __eq__() (abjad.tools.gracetools.Grace.Grace.Grace method), 307 method), 423

__eq__() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark __eq__() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 342 method), 433

__eq__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 345 method), 440

__eq__() (abjad.tools.contexttools.Context.Context.Context __eq__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 355 method), 447

__eq__() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark __eq__() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 359 method), 454

__eq__() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark __eq__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 363 method), 461

__eq__() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark __eq__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 367 method), 468

__eq__() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark __eq__() (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone method), 372 method), 475

__eq__() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark __eq__() (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice method), 376 method), 482

__eq__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark __eq__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 381 method), 489

__eq__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 384 method), 496

__eq__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 503

__eq__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 510

__eq__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 517

__eq__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524

__eq__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531

__eq__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538

__eq__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545

__eq__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552

__eq__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 559

__eq__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566

__eq__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573

__eq__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 580

__eq__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 587

__eq__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594

__eq__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601

__eq__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608

__eq__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615

__eq__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 622

__eq__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 629

__eq__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 633

__eq__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641

__eq__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648

__eq__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 655

__eq__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 662

__eq__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 669

__eq__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 676

__eq__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 683

__eq__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690

__eq__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 697

__eq__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 704

__eq__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711

__eq__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718

__eq__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725

__eq__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732

__eq__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739

__eq__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 746

__eq__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 753

__eq__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760

__eq__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__eq__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 769

__eq__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 809

__eq__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 812

__eq__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 817

__eq__() (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 821

__eq__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 829

__eq__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 832

__eq__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 834

__eq__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 841

__eq__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 1706

__eq__() (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics method), 2022

__eq__() (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender method), 2027

__eq__() (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen method), 2032

__eq__() (abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace method), 2036

__eq__() (abjad.tools.lyrictools.LyricText.LyricText.LyricText method), 2040

__eq__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2050

__eq__() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 847

__eq__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 851

__eq__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 855

__eq__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 859

__eq__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 862

__eq__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 866

__eq__() (abjad.tools.marktools.Mark.Mark.Mark method), 868

__eq__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 872

__eq__() (abjad.tools.markuptools.Markup.Markup.Markup method), 892

__eq__() (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 894

__eq__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 898

__eq__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 913

__eq__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 923

__eq__() (abjad.tools.measuretools.Measure.Measure.Measure method), 932

__eq__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 969

__eq__() (abjad.tools.notetools.Note.Note.Note method), 974

__eq__() (abjad.tools.notetools.NoteHead.NoteHead.NoteHead method), 978

__eq__() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1730

__eq__() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1733

__eq__() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1735

__eq__() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1738

__eq__() (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 990

__eq__() (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 992

__eq__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 994

__eq__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 999

__eq__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1002

__eq__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1006

__eq__() (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1009

__eq__() (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1011

__eq__() (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1014

__eq__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1016

__eq__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1020

__eq__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1023

__eq__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1027

__eq__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1030

__eq__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1032

__eq__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1037

__eq__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVectorSet.InversionEquivalentChromaticIntervalClassVectorSet.InversionEquivalentChromaticIntervalClassVectorSet method), 1042

__eq__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1045

__eq__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1048

__eq__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1053

__eq__() (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method), 1056

__eq__() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1058

__eq__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1060

__eq__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1065

__eq__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1069

__eq__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1073

__eq__() (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1076

__eq__() (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1079

__eq__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1082

__eq__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1084

__eq__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClassSegment.MelodicDiatonicIntervalClassSegment.MelodicDiatonicIntervalClassSegment method), 1087

__eq__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1091

__eq__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch method), 1097

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1099

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1102

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1106

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchSegmentNamedChromaticPitchSegment method), 1110

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1114

__eq__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1119

__eq__() (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch method), 1123

__eq__() (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass method), 1126

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch method), 1128

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1131

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitchClassColor.NumberedChromaticPitchClassColor method), 1133

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1136

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1141

__eq__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1146

__eq__() (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1149

__eq__() (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass method), 1152

__eq__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1155

__eq__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent method), 1158

__eq__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1162

__eq__() (abjad.tools.pitchtools.PitchRange.PitchRange method), 1166

__eq__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1169

__eq__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1175

__eq__() (abjad.tools.quantizationtools.QEvent.QEvent method), 2053

__eq__() (abjad.tools.quantizationtools.QGrid.QGrid method), 2057

__eq__() (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer method), 2060

__eq__() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree method), 2065

__eq__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1745

__eq__() (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1749

__eq__() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1759

__eq__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1761

__eq__() (abjad.tools.sequencetools.Matrix.Matrix.Matrix method), 1764

__eq__() (abjad.tools.sequencetools.Tree.Tree.Tree method), 1773

__eq__() (abjad.tools.sequencetools.ResidueClass.ResidueClass.ResidueClass method), 1820

- `__eq__()` (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator.pitchtools.HarmonicChromaticInterval.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 2092
- `__eq__()` (abjad.tools.tonalitytools.Scale.Scale.Scale.__float__()) (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 2096
- `__eq__()` (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree.abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 2099
- `__eq__()` (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator.abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 2100
- `__eq__()` (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction.abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 2102
- `__eq__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1573
- `__eq__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet.__float__()) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1584
- `__eq__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment.abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1943
- `__eq__()` (abjad.tools.voicetools.Voice.Voice.Voice.__float__()) (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1606
- `__eq__()` (abjad.tools.wellformednesstools.BeamedQuarterNoteCheck(BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.MelodicChromaticInterval.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1956
- `__eq__()` (abjad.tools.wellformednesstools.Check.Check.Check.__float__()) (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1954
- `__eq__()` (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck(DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.MelodicCounterpointInterval.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1958
- `__eq__()` (abjad.tools.wellformednesstools.DuplicateIdCheck(DuplicateIdCheck.DuplicateIdCheck.MelodicDiatonicInterval.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1960
- `__eq__()` (abjad.tools.wellformednesstools.EmptyContainerCheck(EmptyContainerCheck.EmptyContainerCheck.MelodicDiatonicInterval.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1963
- `__eq__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck(IntermarkedHairpinCheck.IntermarkedHairpinCheck.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1965
- `__eq__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck(MisduratedMeasureCheck.MisduratedMeasureCheck.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1967
- `__eq__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck(MisfilledMeasureCheck.MisfilledMeasureCheck.NamedDiatonicPitchClass.NamedDiatonicPitchClass method), 1969
- `__eq__()` (abjad.tools.wellformednesstools.MispitchedTieCheck(MispitchedTieCheck.MispitchedTieCheck.NamedDiatonicPitchClass.NamedDiatonicPitchClass method), 1972
- `__eq__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck(MisrepresentedFlagCheck.MisrepresentedFlagCheck.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1974
- `__eq__()` (abjad.tools.wellformednesstools.MissingParentCheck(MissingParentCheck.MissingParentCheck.ChromaticPitchClass.ChromaticPitchClass method), 1976
- `__eq__()` (abjad.tools.wellformednesstools.NestedMeasureCheck(NestedMeasureCheck.NestedMeasureCheck.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1978
- `__eq__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck(OverlappingBeamCheck.OverlappingBeamCheck.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1981
- `__eq__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck(OverlappingGlissandoCheck.OverlappingGlissandoCheck.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1983
- `__eq__()` (abjad.tools.wellformednesstools.OverlappingOctaveCheck(OverlappingOctaveCheck.OverlappingOctaveCheck.NumberedDiatonicPitch.NumberedDiatonicPitch method), 1985
- `__eq__()` (abjad.tools.wellformednesstools.ShortHairpinCheck(ShortHairpinCheck.ShortHairpinCheck.AbjadObject.AbjadObject method), 1987
- `__float__()` (abjad.tools.durationtools.Duration.Duration.Duration.__int__()) (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject method), 1661
- `__float__()` (abjad.tools.durationtools.Offset.Offset.Offset.__ge__()) (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject method), 1665

- [__ge__\(\) \(abjad.tools.abctools.SortableAttributeEqualityAbjadObject\(SortableAttributeEqualityAbjadObjectSortableObjectInEqualityQ method\), 1617](#)
- [__ge__\(\) \(abjad.tools.abctools.SortableAttributeEqualityAbjadObject\(SortableAttributeEqualityAbjadObjectSortableObjectInEqualityQ method\), 1630](#)
- [__ge__\(\) \(abjad.tools.chordtools.Chord.Chord.Chord __ge__\(\) \(abjad.tools.documentationtools.APICrawler.APICrawler.APICraw method\), 228](#)
- [__ge__\(\) \(abjad.tools.chordtools.Chord.Chord.Chord __ge__\(\) \(abjad.tools.documentationtools.APICrawler.APICrawler.APICraw method\), 1634](#)
- [__ge__\(\) \(abjad.tools.componenttools.Component.ComponentsComponent\(abjad tools.documentationtools.AbjadAPIGenerator.AbjadAPIGe method\), 237](#)
- [__ge__\(\) \(abjad.tools.componenttools.Component.ComponentsComponent\(abjad tools.documentationtools.AbjadAPIGenerator.AbjadAPIGe method\), 1636](#)
- [__ge__\(\) \(abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexClassCrawler.ClassCraw method\), 1991](#)
- [__ge__\(\) \(abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexClassCrawler.ClassCraw method\), 1638](#)
- [__ge__\(\) \(abjad.tools.constrainttools.Domain.Domain.Domain.__ge__\(\) \(abjad.tools.documentationtools.ClassDocumenter.ClassDocumen method\), 1994](#)
- [__ge__\(\) \(abjad.tools.constrainttools.Domain.Domain.Domain.__ge__\(\) \(abjad.tools.documentationtools.ClassDocumenter.ClassDocumen method\), 1640](#)
- [__ge__\(\) \(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.Documen method\), 1997](#)
- [__ge__\(\) \(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.Documen method\), 1642](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawle method\), 2000](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawle method\), 1644](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 2002](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 1647](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 2005](#)
- [__ge__\(\) \(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 1652](#)
- [__ge__\(\) \(abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 2008](#)
- [__ge__\(\) \(abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 1654](#)
- [__ge__\(\) \(abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 2011](#)
- [__ge__\(\) \(abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint\(abjad.tools.documentationtools.FunctionCrawler.FunctionC method\), 1657](#)
- [__ge__\(\) \(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver method\), 2015](#)
- [__ge__\(\) \(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver method\), 1661](#)
- [__ge__\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster __ge__\(\) \(abjad.tools.durationtools.Offset.Offset.Offset method\), 299](#)
- [__ge__\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster __ge__\(\) \(abjad.tools.durationtools.Offset.Offset.Offset method\), 1665](#)
- [__ge__\(\) \(abjad.tools.containertools.Container.Container.Container.__ge__\(\) \(abjad.tools.gracetools.Grace.Grace.Grace method\), 307](#)
- [__ge__\(\) \(abjad.tools.containertools.Container.Container.Container.__ge__\(\) \(abjad.tools.gracetools.Grace.Grace.Grace method\), 423](#)
- [__ge__\(\) \(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark __ge__\(\) \(abjad.tools.instrumenttools.Accordion.Accordion.Accordion method\), 342](#)
- [__ge__\(\) \(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark __ge__\(\) \(abjad.tools.instrumenttools.Accordion.Accordion.Accordion method\), 433](#)
- [__ge__\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory\(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method\), 345](#)
- [__ge__\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory\(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method\), 440](#)
- [__ge__\(\) \(abjad.tools.contexttools.Context.Context.Context __ge__\(\) \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoS method\), 355](#)
- [__ge__\(\) \(abjad.tools.contexttools.Context.Context.Context __ge__\(\) \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoS method\), 447](#)
- [__ge__\(\) \(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark __ge__\(\) \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTr method\), 359](#)
- [__ge__\(\) \(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark __ge__\(\) \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTr method\), 454](#)
- [__ge__\(\) \(abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark\(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatCla method\), 363](#)
- [__ge__\(\) \(abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark\(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatCla method\), 461](#)
- [__ge__\(\) \(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark\(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarim method\), 367](#)
- [__ge__\(\) \(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark\(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarim method\), 468](#)
- [__ge__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark\(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method\), 372](#)
- [__ge__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark\(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method\), 475](#)
- [__ge__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark\(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.Bass method\), 376](#)
- [__ge__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark\(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.Bass method\), 482](#)
- [__ge__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark\(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassT method\), 381](#)
- [__ge__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark\(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassT method\), 489](#)
- [__ge__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 384](#)
- [__ge__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 496](#)
- [__ge__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 390](#)
- [__ge__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 503](#)
- [__ge__\(\) \(abjad.tools.datastructuretools.Digraph.Digraph.Digraph __ge__\(\) \(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method\), 1622](#)
- [__ge__\(\) \(abjad.tools.datastructuretools.Digraph.Digraph.Digraph __ge__\(\) \(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method\), 510](#)
- [__ge__\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 1626](#)
- [__ge__\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary\(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo method\), 517](#)

- `__ge__()` (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524
- `__ge__()` (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531
- `__ge__()` (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538
- `__ge__()` (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545
- `__ge__()` (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552
- `__ge__()` (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 559
- `__ge__()` (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566
- `__ge__()` (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573
- `__ge__()` (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 580
- `__ge__()` (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 587
- `__ge__()` (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594
- `__ge__()` (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601
- `__ge__()` (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608
- `__ge__()` (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615
- `__ge__()` (abjad.tools.instrumenttools.Harp.Harp.Harp method), 622
- `__ge__()` (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 629
- `__ge__()` (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 633
- `__ge__()` (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641
- `__ge__()` (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648
- `__ge__()` (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 655
- `__ge__()` (abjad.tools.instrumenttools.Piano.Piano.Piano method), 662
- `__ge__()` (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 669
- `__ge__()` (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 676
- `__ge__()` (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 683
- `__ge__()` (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690
- `__ge__()` (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 697
- `__ge__()` (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 704
- `__ge__()` (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711
- `__ge__()` (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718
- `__ge__()` (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725
- `__ge__()` (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732
- `__ge__()` (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739
- `__ge__()` (abjad.tools.instrumenttools.Viola.Viola.Viola method), 746
- `__ge__()` (abjad.tools.instrumenttools.Violin.Violin.Violin method), 753
- `__ge__()` (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760
- `__ge__()` (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699
- `__ge__()` (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 769
- `__ge__()` (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 809
- `__ge__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 812
- `__ge__()` (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 817
- `__ge__()` (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 822
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 829
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 832
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 834
- `__ge__()` (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 841
- `__ge__()` (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 1706
- `__ge__()` (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics method), 2022
- `__ge__()` (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender method), 2027
- `__ge__()` (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen method), 2032
- `__ge__()` (abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace method), 2036
- `__ge__()` (abjad.tools.lyrictools.LyricText.LyricText.LyricText method), 2041
- `__ge__()` (abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics method), 2051
- `__ge__()` (abjad.tools.lyrictools.Annotation.Annotation.Annotation method), 847
- `__ge__()` (abjad.tools.lyrictools.Articulation.Articulation.Articulation method), 851

- [__ge__\(\) \(abjad.tools.marktools.BarLine.BarLine.BarLine method\), 855](#)
- [__ge__\(\) \(abjad.tools.marktools.BendAfter.BendAfter.BendAfter method\), 859](#)
- [__ge__\(\) \(abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method\), 862](#)
- [__ge__\(\) \(abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method\), 866](#)
- [__ge__\(\) \(abjad.tools.marktools.Mark.Mark.Mark method\), 869](#)
- [__ge__\(\) \(abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method\), 872](#)
- [__ge__\(\) \(abjad.tools.markuptools.Markup.Markup.Markup method\), 892](#)
- [__ge__\(\) \(abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method\), 894](#)
- [__ge__\(\) \(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method\), 898](#)
- [__ge__\(\) \(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method\), 913](#)
- [__ge__\(\) \(abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method\), 923](#)
- [__ge__\(\) \(abjad.tools.measuretools.Measure.Measure.Measure method\), 932](#)
- [__ge__\(\) \(abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method\), 969](#)
- [__ge__\(\) \(abjad.tools.notetools.Note.Note.Note method\), 974](#)
- [__ge__\(\) \(abjad.tools.notetools.NoteHead.NoteHead.NoteHead method\), 978](#)
- [__ge__\(\) \(abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method\), 1730](#)
- [__ge__\(\) \(abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method\), 1733](#)
- [__ge__\(\) \(abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method\), 1735](#)
- [__ge__\(\) \(abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method\), 1738](#)
- [__ge__\(\) \(abjad.tools.pitchtools.Accidental.Accidental.Accidental method\), 990](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method\), 993](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method\), 994](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method\), 999](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method\), 1002](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method\), 1006](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method\), 1009](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method\), 1011](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method\), 1014](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method\), 1016](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method\), 1020](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method\), 1023](#)
- [__ge__\(\) \(abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method\), 1027](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method\), 1030](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method\), 1033](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment method\), 1037](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method\), 1042](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method\), 1045](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method\), 1048](#)
- [__ge__\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method\), 1053](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method\), 1056](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method\), 1058](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method\), 1061](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method\), 1066](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method\), 1069](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method\), 1073](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method\), 1076](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method\), 1079](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method\), 1082](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method\), 1084](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet method\), 1087](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method\), 1091](#)
- [__ge__\(\) \(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method\), 1097](#)
- [__ge__\(\) \(abjad.tools.pitchtools.NamedChromaticInterval.NamedChromaticInterval.NamedChromaticInterval method\), 1099](#)
- [__ge__\(\) \(abjad.tools.pitchtools.NamedChromaticIntervalClass.NamedChromaticIntervalClass.NamedChromaticIntervalClass method\), 1102](#)

__ge__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegmentName) (method),	1106	__ge__() (abjad.tools.pitchtools.NamedChromaticPitchClassSetName) (method),	1227
__ge__() (abjad.tools.pitchtools.NamedChromaticPitchSegmentName) (method),	1110	__ge__() (abjad.tools.pitchtools.NamedChromaticPitchSegmentName) (method),	1229
__ge__() (abjad.tools.pitchtools.NamedChromaticPitchSetName) (method),	1114	__ge__() (abjad.tools.pitchtools.NamedChromaticPitchSchemeColor) (method),	1231
__ge__() (abjad.tools.pitchtools.NamedChromaticPitchVectorName) (method),	1119	__ge__() (abjad.tools.pitchtools.NamedChromaticPitchVectorName) (method),	1234
__ge__() (abjad.tools.pitchtools.NamedDiatonicPitchName) (method),	1124	__ge__() (abjad.tools.pitchtools.NamedDiatonicPitchSchemePair) (method),	1235
__ge__() (abjad.tools.pitchtools.NamedDiatonicPitchClassName) (method),	1126	__ge__() (abjad.tools.pitchtools.NamedDiatonicPitchSchemeVector) (method),	1238
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchNumberedChromaticPitch) (method),	1128	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchNumberedChromaticPitch) (method),	1240
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassName) (method),	1131	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchPlate) (method),	1241
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassColor) (method),	1133	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchStringQuantMap) (method),	1244
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegmentName) (method),	1136	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassStaffPin) (method),	1246
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSetName) (method),	1141	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassStaffPin) (method),	1256
__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVectorName) (method),	1146	__ge__() (abjad.tools.pitchtools.NumberedChromaticPitchClassStaffPin) (method),	1259
__ge__() (abjad.tools.pitchtools.NumberedDiatonicPitchNumberedDiatonicPitch) (method),	1149	__ge__() (abjad.tools.pitchtools.NumberedDiatonicPitchPerformer) (method),	1263
__ge__() (abjad.tools.pitchtools.NumberedDiatonicPitchClassName) (method),	1152	__ge__() (abjad.tools.pitchtools.NumberedDiatonicPitchPerformer) (method),	1266
__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingOctaveTranspositionMapping) (method),	1155	__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingOctaveTranspositionMapping) (method),	1277
__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent) (method),	1158	__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent) (method),	1287
__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory) (method),	1162	__ge__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory) (method),	1298
__ge__() (abjad.tools.pitchtools.PitchRange.PitchRange) (method),	1166	__ge__() (abjad.tools.sequencetools.CyclicList.CyclicList) (method),	1745
__ge__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory) (method),	1169	__ge__() (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix) (method),	1749
__ge__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow) (method),	1175	__ge__() (abjad.tools.sequencetools.CyclicTree.CyclicTree) (method),	1759
__ge__() (abjad.tools.quantizationtools.QEvent.QEvent) (method),	2053	__ge__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple) (method),	1761
__ge__() (abjad.tools.quantizationtools.QGrid.QGrid) (method),	2057	__ge__() (abjad.tools.sequencetools.Matrix.Matrix) (method),	1764
__ge__() (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer) (method),	2061	__ge__() (abjad.tools.sequencetools.Tree.Tree) (method),	1773
__ge__() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree) (method),	2065	__ge__() (abjad.tools.sequencetools.ResidueClass.ResidueClass) (method),	1820
__ge__() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup) (method),	2070	__ge__() (abjad.tools.sequencetools.ResidueClassExpression.ResidueClassExpression) (method),	1823
__ge__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest) (method),	1216	__ge__() (abjad.tools.skip.Skip.Skip) (method),	1310
__ge__() (abjad.tools.resttools.Rest.Rest) (method),	1220	__ge__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner) (method),	1322

`__ge__()` (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator.Context.Context.Context method), 2100
`__ge__()` (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction.abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory method), 2102
`__ge__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.abjad.tools.immutabletools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1574
`__ge__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet `__getitem__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1584
`__ge__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment.abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1943
`__ge__()` (abjad.tools.voicetools.Voice.Voice.Voice `__getitem__()` (abjad.tools.gracetools.Grace.Grace.Grace method), 1606
`__ge__()` (abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.abjad.tools.instrumenttools.InstrumentTools.InstrumentTools.InstrumentTools method), 1956
`__ge__()` (abjad.tools.wellformednesstools.Check.Check.Check `__getitem__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 1954
`__ge__()` (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.abjad.tools.lilypondfiletools.BookFigureBlock.BookFigureBlock.BookFigureBlock method), 1958
`__ge__()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck.abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 1961
`__ge__()` (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck.abjad.tools.scoretools.ScoreBlock.ScoreBlock.ScoreBlock method), 1963
`__ge__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck.abjad.tools.addlyrics.AddLyrics.AddLyrics method), 1965
`__ge__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.MisduratedMeasureCheck.abjad.tools.lyrics.Lyrics.Lyrics method), 1967
`__ge__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck.abjad.tools.markup.MarkupInventory.MarkupInventory.MarkupInventory method), 1970
`__ge__()` (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck.abjad.tools.anonymousmeasure.AnonymousMeasure.AnonymousMeasure method), 1972
`__ge__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck.abjad.tools.dynamicmeasure.DynamicMeasure.DynamicMeasure method), 1974
`__ge__()` (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck.abjad.tools.measure.Measure.Measure method), 1976
`__ge__()` (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck.abjad.tools.pitchtools.PitchArray.PitchArray.PitchArray method), 1979
`__ge__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck.abjad.tools.pitchtools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1981
`__ge__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck.abjad.tools.pitchtools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1983
`__ge__()` (abjad.tools.wellformednesstools.OverlappingOctaveCheck.OverlappingOctaveCheck.OverlappingOctaveCheck.abjad.tools.pitchtools.PitchArrayRow.PitchArrayRow method), 1985
`__ge__()` (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck.abjad.tools.interval.IntervalSegment.IntervalSegment.IntervalSegment method), 1988
`__getitem__()` (abjad.tools.chordtools.Chord.Chord.Chord `__getitem__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 228
`__getitem__()` (abjad.tools.constrainttools.Domain.Domain.Domain `__getitem__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment method), 1994
`__getitem__()` (abjad.tools.containertools.Cluster.Cluster.Cluster `__getitem__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment.InversionEquivalentChromaticIntervalSegment method), 300
`__getitem__()` (abjad.tools.containertools.Container.Container.Container `__getitem__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment method), 307
`__getitem__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.abjad.tools.inversionequivalent.InversionEquivalentDiatonicIntervalSegment.InversionEquivalentDiatonicIntervalSegment method), 345

__getitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment, abjad.tools.pitchtools.MelodicChromaticIntervalSegment, method), 1061

__getitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment, abjad.tools.pitchtools.MelodicChromaticIntervalSegment, method), 1066

__getitem__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment, abjad.tools.pitchtools.MelodicChromaticIntervalSegment, method), 1069

__getitem__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment, abjad.tools.pitchtools.MelodicDiatonicIntervalSegment, method), 1087

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment, abjad.tools.pitchtools.NamedChromaticPitchClassSegment, method), 1102

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchSegment, abjad.tools.pitchtools.NamedChromaticPitchSegment, method), 1110

__getitem__() (abjad.tools.pitchtools.NamedChromaticPitchVector, abjad.tools.pitchtools.NamedChromaticPitchVector, method), 1119

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClass, abjad.tools.pitchtools.NumberedChromaticPitchClass, method), 1133

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment, abjad.tools.pitchtools.NumberedChromaticPitchClassSegment, method), 1136

__getitem__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector, abjad.tools.pitchtools.NumberedChromaticPitchClassVector, method), 1147

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping, abjad.tools.pitchtools.OctaveTranspositionMapping, method), 1155

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping, abjad.tools.pitchtools.OctaveTranspositionMapping, method), 1162

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory, abjad.tools.pitchtools.PitchRangeInventory, method), 1169

__getitem__() (abjad.tools.pitchtools.TwelveToneRow, abjad.tools.pitchtools.TwelveToneRow, method), 1175

__getitem__() (abjad.tools.quantizationtools.QGrid, abjad.tools.quantizationtools.QGrid, method), 2057

__getitem__() (abjad.tools.quantizationtools.QGridSearchTree, abjad.tools.quantizationtools.QGridSearchTree, method), 2066

__getitem__() (abjad.tools.quantizationtools.QGridTempoLookup, abjad.tools.quantizationtools.QGridTempoLookup, method), 2070

__getitem__() (abjad.tools.scoretools.GrandStaff, abjad.tools.scoretools.GrandStaff, method), 1256

__getitem__() (abjad.tools.scoretools.PerformerInventory, abjad.tools.scoretools.PerformerInventory, method), 1266

__getitem__() (abjad.tools.scoretools.PianoStaff, abjad.tools.scoretools.PianoStaff, method), 1277

__getitem__() (abjad.tools.scoretools.Score, abjad.tools.scoretools.Score, method), 1287

__getitem__() (abjad.tools.scoretools.StaffGroup, abjad.tools.scoretools.StaffGroup, method), 1298

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1745

__getitem__() (abjad.tools.sequencetools.CyclicMatrix, abjad.tools.sequencetools.CyclicMatrix, method), 1749

__getitem__() (abjad.tools.sequencetools.CyclicTree, abjad.tools.sequencetools.CyclicTree, method), 1759

__getitem__() (abjad.tools.sequencetools.CyclicTuple, abjad.tools.sequencetools.CyclicTuple, method), 1762

__getitem__() (abjad.tools.sequencetools.Matrix, abjad.tools.sequencetools.Matrix, method), 1764

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1773

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1322

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1330

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1339

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1349

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1359

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1369

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1377

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1385

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1394

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1402

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1409

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1419

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1427

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1435

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1443

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1451

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1459

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1466

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1473

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1480

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1488

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1495

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1503

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1528

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1538

__getitem__() (abjad.tools.sequencetools.CyclicList, abjad.tools.sequencetools.CyclicList, method), 1549

__getitem__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1843

__getitem__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1856

__getitem__() (abjad.tools.timeintervaltools.TimeIntervalTree.Dictionary(TimeIntervalTree.PitchRangeInventor method), 1880

__getitem__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 2081

__getitem__() (abjad.tools.tonalitytools.Scale.Scale.Scale __getslice__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory method), 2096

__getitem__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1574

__getitem__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet __getslice__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicT method), 1584

__getitem__() (abjad.tools.voicetools.Voice.Voice.Voice __getslice__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeInterval method), 1606

__getslice__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 346

__getslice__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 385

__getslice__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1630

__getslice__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 633

__getslice__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 812

__getslice__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 817

__getslice__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 829

__getslice__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 841

__getslice__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 898

__getslice__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1002

__getslice__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1024

__getslice__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSegment.HarmonicDiatonicIntervalSegment method), 1033

__getslice__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1048

__getslice__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.HarmonicDiatonicIntervalSegment method), 1061

__getslice__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.HarmonicDiatonicIntervalSegment method), 1069

__getslice__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1088

__getslice__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.HarmonicDiatonicIntervalSegment method), 1102

__getslice__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.HarmonicDiatonicIntervalSegment method), 1110

__getslice__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.HarmonicDiatonicIntervalSegment method), 1137

__getitem__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1155

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1162

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1169

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1175

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1266

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1745

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1762

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1856

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2081

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2097

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1612

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1614

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1615

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1617

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 228

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 237

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1991

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1994

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1997

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2000

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2002

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2005

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2008

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2011

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 2015

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 300

__getitem__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 307

[__gt__\(\)](#) (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 342
[__gt__\(\)](#) (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 346
[__gt__\(\)](#) (abjad.tools.contexttools.Context.Context.Context method), 355
[__gt__\(\)](#) (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 359
[__gt__\(\)](#) (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method), 363
[__gt__\(\)](#) (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method), 367
[__gt__\(\)](#) (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method), 372
[__gt__\(\)](#) (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 376
[__gt__\(\)](#) (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 381
[__gt__\(\)](#) (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 385
[__gt__\(\)](#) (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method), 390
[__gt__\(\)](#) (abjad.tools.datastructuretools.Digraph.Digraph.Digraph method), 1622
[__gt__\(\)](#) (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1627
[__gt__\(\)](#) (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1630
[__gt__\(\)](#) (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method), 1634
[__gt__\(\)](#) (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method), 1636
[__gt__\(\)](#) (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method), 1638
[__gt__\(\)](#) (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter method), 1640
[__gt__\(\)](#) (abjad.tools.documentationtools.Documenter.Documenter.Documenter method), 1642
[__gt__\(\)](#) (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method), 1644
[__gt__\(\)](#) (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter method), 1647
[__gt__\(\)](#) (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1652
[__gt__\(\)](#) (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method), 1654
[__gt__\(\)](#) (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1657
[__gt__\(\)](#) (abjad.tools.durationtools.Duration.Duration.Duration method), 1661
[__gt__\(\)](#) (abjad.tools.durationtools.Offset.Offset.Offset method), 1665
[__gt__\(\)](#) (abjad.tools.gracetools.Grace.Grace.Grace method), 423
[__gt__\(\)](#) (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 433
[__gt__\(\)](#) (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 440
[__gt__\(\)](#) (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 447
[__gt__\(\)](#) (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 454
[__gt__\(\)](#) (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 461
[__gt__\(\)](#) (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone method), 468
[__gt__\(\)](#) (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice method), 475
[__gt__\(\)](#) (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 482
[__gt__\(\)](#) (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 489
[__gt__\(\)](#) (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 496
[__gt__\(\)](#) (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 503
[__gt__\(\)](#) (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 510
[__gt__\(\)](#) (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 517
[__gt__\(\)](#) (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524
[__gt__\(\)](#) (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531
[__gt__\(\)](#) (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538
[__gt__\(\)](#) (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545
[__gt__\(\)](#) (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552
[__gt__\(\)](#) (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 559
[__gt__\(\)](#) (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566
[__gt__\(\)](#) (abjad.tools.instrumenttools.ContraVoice.ContraVoice.ContraVoice method), 573
[__gt__\(\)](#) (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 580
[__gt__\(\)](#) (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 587
[__gt__\(\)](#) (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594
[__gt__\(\)](#) (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601
[__gt__\(\)](#) (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608
[__gt__\(\)](#) (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615

__gt__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 622

__gt__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 629

__gt__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory method), 633

__gt__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641

__gt__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648

__gt__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 655

__gt__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 662

__gt__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 669

__gt__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 676

__gt__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 683

__gt__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690

__gt__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 697

__gt__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 704

__gt__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711

__gt__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718

__gt__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725

__gt__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732

__gt__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739

__gt__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 746

__gt__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 753

__gt__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760

__gt__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__gt__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 770

__gt__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 809

__gt__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 812

__gt__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 817

__gt__() (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 822

__gt__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 830

__gt__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 832

__gt__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 834

__gt__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 842

__gt__() (abjad.tools.lilypondfiletools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 1706

__gt__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2022

__gt__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender method), 2027

__gt__() (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen method), 2032

__gt__() (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace method), 2036

__gt__() (abjad.tools.lyricstools.LyricText.LyricText.LyricText method), 2041

__gt__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2051

__gt__() (abjad.tools.lyricstools.Annotation.Annotation.Annotation method), 847

__gt__() (abjad.tools.lyricstools.Articulation.Articulation.Articulation method), 851

__gt__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 856

__gt__() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 859

__gt__() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 862

__gt__() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 866

__gt__() (abjad.tools.marktools.Mark.Mark.Mark method), 869

__gt__() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 872

__gt__() (abjad.tools.markuptools.Markup.Markup.Markup method), 892

__gt__() (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 894

__gt__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 898

__gt__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 913

__gt__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 923

__gt__() (abjad.tools.measuretools.Measure.Measure.Measure method), 932

__gt__() (abjad.tools.measuretools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 969

__gt__() (abjad.tools.notetools.Note.Note.Note method), 975

__gt__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1156	__gt__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent method), 1277
__gt__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent method), 1159	__gt__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent method), 1288
__gt__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1162	__gt__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1298
__gt__() (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange method), 1166	__gt__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1745
__gt__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1169	__gt__() (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1749
__gt__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1175	__gt__() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1759
__gt__() (abjad.tools.quantizationtools.QEvent.QEvent.QEvent method), 2053	__gt__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1762
__gt__() (abjad.tools.quantizationtools.QGrid.QGrid.QGrid method), 2057	__gt__() (abjad.tools.sequencetools.Matrix.Matrix.Matrix method), 1765
__gt__() (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer.QGridQuantizer method), 2061	__gt__() (abjad.tools.sequencetools.Tree.Tree.Tree method), 1773
__gt__() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree method), 2066	__gt__() (abjad.tools.sequencetools.ResidueClass.ResidueClass.ResidueClass method), 1820
__gt__() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup.QGridTempoLookup method), 2070	__gt__() (abjad.tools.sequencetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1823
__gt__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1216	__gt__() (abjad.tools.resttools.Skip.Skip.Skip method), 1310
__gt__() (abjad.tools.resttools.Rest.Rest.Rest method), 1220	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1322
__gt__() (abjad.tools.schemetools.Scheme.Scheme.Scheme method), 1227	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1330
__gt__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList method), 1229	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1339
__gt__() (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor method), 1232	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1349
__gt__() (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment method), 1234	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1359
__gt__() (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair method), 1235	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1369
__gt__() (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector method), 1238	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1377
__gt__() (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant method), 1240	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1385
__gt__() (abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate method), 1242	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1395
__gt__() (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate.StringQuartetScoreTemplate method), 1244	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1402
__gt__() (abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate.TwoStaffPianoScoreTemplate method), 1246	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1409
__gt__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1256	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1419
__gt__() (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier.InstrumentationSpecifier method), 1260	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1427
__gt__() (abjad.tools.scoretools.Performer.Performer.Performer method), 1263	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1435
__gt__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1266	__gt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1443

- __gt__() (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck.Mark.ContextMark.ContextMark.
method), 1972
- __gt__() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck.Mark.DynamicMark.DynamicMark.
method), 1974
- __gt__() (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck.Mark.InstrumentMark.InstrumentMark.
method), 1976
- __gt__() (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck.Mark.KeySignatureMark.KeySignatureMark.
method), 1979
- __gt__() (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck.Mark.StaffChangeMark.StaffChangeMark.
method), 1981
- __gt__() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck.Mark.MusicalMark.MusicalMark.
method), 1983
- __gt__() (abjad.tools.wellformednesstools.OverlappingOctaveCheck.OverlappingOctaveCheck.OverlappingOctaveCheck.Mark.OctaveSignatureMark.OctaveSignatureMark.
method), 1985
- __gt__() (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck.Digraph.Digraph.Digraph.
method), 1988
- __hash__() (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject.abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.
method), 1612
- __hash__() (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.abjad.tools.documentationtools.APICrawler.APICrawler.
method), 1614
- __hash__() (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.abjad.tools.documentationtools.APICrawler.APICrawler.
method), 1616
- __hash__() (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.abjad.tools.documentationtools.
method), 1617
- __hash__() (abjad.tools.chordtools.Chord.Chord.Chord __hash__() (abjad.tools.documentationtools.Documenter.Documenter.Documenter.
method), 228
- __hash__() (abjad.tools.componenttools.Component.Component.Component __hash__() (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler.
method), 237
- __hash__() (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint.abjad.tools.documentationtools.Documenter.Documenter.
method), 1992
- __hash__() (abjad.tools.constrainttools.Domain.Domain.Domain __hash__() (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler.
method), 1994
- __hash__() (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.abjad.tools.constrainttools.
method), 1997
- __hash__() (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint.abjad.tools.constrainttools.
method), 2000
- __hash__() (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint.abjad.tools.constrainttools.
method), 2003
- __hash__() (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.abjad.tools.constrainttools.
method), 2005
- __hash__() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.abjad.tools.constrainttools.
method), 2008
- __hash__() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.abjad.tools.constrainttools.
method), 2011
- __hash__() (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.abjad.tools.constrainttools.
method), 2015
- __hash__() (abjad.tools.containertools.Cluster.Cluster.Cluster __hash__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.
method), 300
- __hash__() (abjad.tools.containertools.Container.Container.Container __hash__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet.
method), 307
- __hash__() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark __hash__() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone.
method), 342
- __hash__() (abjad.tools.contexttools.Context.Context.Context __hash__() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice.
method), 355

__hash__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 482

__hash__() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 489

__hash__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 496

__hash__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 503

__hash__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 510

__hash__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 517

__hash__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524

__hash__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531

__hash__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538

__hash__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545

__hash__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552

__hash__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 559

__hash__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566

__hash__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573

__hash__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 580

__hash__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 587

__hash__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594

__hash__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601

__hash__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608

__hash__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615

__hash__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 622

__hash__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 630

__hash__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641

__hash__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648

__hash__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 655

__hash__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 662

__hash__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 669

__hash__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 676

__hash__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 683

__hash__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690

__hash__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 697

__hash__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 704

__hash__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711

__hash__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718

__hash__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725

__hash__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732

__hash__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739

__hash__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 746

__hash__() (abjad.tools.instrumenttools.Violoncello.Violoncello.Violoncello method), 753

__hash__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760

__hash__() (abjad.tools.leafstools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__hash__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 809

__hash__() (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock method), 820

__hash__() (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 822

__hash__() (abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock method), 824

__hash__() (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock method), 826

__hash__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 833

__hash__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 835

__hash__() (abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock method), 837

__hash__() (abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock method), 838

__hash__() (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser method), 1706

__hash__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2023

__hash__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender method), 2028

2147

__hash__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.VerboseNameScheme.PitchClassSegment), 1103

__hash__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.VerboseNameScheme.PitchClassSegment), 1106

__hash__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.VerboseNameScheme.PitchSegment), 1110

__hash__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.VerboseNameScheme.PitchSet), 1114

__hash__() (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.VerboseNameScheme.Pitch), 1124

__hash__() (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.VerboseNameScheme.PitchClass), 1126

__hash__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.VerboseNameScheme.Pitch), 1129

__hash__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.VerboseNameScheme.PitchClass), 1131

__hash__() (abjad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap.VerboseNameScheme.PitchClassColorMap), 1133

__hash__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.VerboseNameScheme.PitchClassSegment), 1137

__hash__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.VerboseNameScheme.PitchClassSet), 1142

__hash__() (abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.VerboseNameScheme.Pitch), 1150

__hash__() (abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass.VerboseNameScheme.PitchClass), 1152

__hash__() (abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent.VerboseNameScheme.PitchClass), 1159

__hash__() (abjad.tools.pitchtools.PitchRange.PitchRange.VerboseNameScheme.PitchRange), 1166

__hash__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.VerboseNameScheme.PitchClass), 1175

__hash__() (abjad.tools.quantizationtools.QEvent.QEvent.VerboseNameScheme.PitchClass), 2054

__hash__() (abjad.tools.quantizationtools.QGrid.QGrid.VerboseNameScheme.PitchClass), 2057

__hash__() (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer.VerboseNameScheme.PitchClass), 2061

__hash__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.VerboseNameScheme.PitchClass), 1216

__hash__() (abjad.tools.resttools.Rest.Rest.Rest method), 1220

__hash__() (abjad.tools.schemetools.Scheme.Scheme.Scheme method), 1227

__hash__() (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.VerboseNameScheme.PitchClass), 1229

__hash__() (abjad.tools.schemetools.SchemeColor.SchemeColor.VerboseNameScheme.PitchClass), 1232

__hash__() (abjad.tools.schemetools.SchemeMoment.SchemeMoment.VerboseNameScheme.PitchClass), 1234

__hash__() (abjad.tools.schemetools.SchemePair.SchemePair.VerboseNameScheme.PitchClass), 1236

__hash__() (abjad.tools.schemetools.SchemeVector.SchemeVector.VerboseNameScheme.PitchClass), 1238

__hash__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass), 1820

__hash__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.VerboseNameScheme.PitchClass), 1823

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1311

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1323

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1330

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1340

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1349

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1359

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1369

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1377

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1385

__hash__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1395

- __hash__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker.OutputIncisedTimeTokenMaker method), 1402
- __hash__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker method), 1410
- __hash__() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker method), 1419
- __hash__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker method), 1428
- __hash__() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.BurnishedSignalFilledTimeTokenMaker.BurnishedSignalFilledTimeTokenMaker.BurnishedSignalFilledTimeTokenMaker method), 1435
- __hash__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker method), 1444
- __hash__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker method), 1451
- __hash__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker method), 1459
- __hash__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1467
- __hash__() (abjad.tools.spannertools.Spanner.Spanner.Spanner.__hash__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1473
- __hash__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner.__hash__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1481
- __hash__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner.__hash__() (abjad.tools.tonalitytools.DoublingIndicator.DoublingIndicator.DoublingIndicator method), 1488
- __hash__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.__hash__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator method), 1496
- __hash__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.__hash__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1503
- __hash__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.__hash__() (abjad.tools.tonalitytools.Mode.Mode.Mode method), 1528
- __hash__() (abjad.tools.stafftools.Staff.Staff.Staff.__hash__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1538
- __hash__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.__hash__() (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator method), 1549
- __hash__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.__hash__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1843
- __hash__() (abjad.tools.timeintervaltools.TimeIntervalAggregator.TimeIntervalAggregator.TimeIntervalAggregator.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1835
- __hash__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1838
- __hash__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1856
- __hash__() (abjad.tools.timetokentools.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1897
- __hash__() (abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker.IncisedTimeTokenMaker.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1899
- __hash__() (abjad.tools.timetokentools.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.__hash__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree method), 1909
- __hash__() (abjad.tools.timetokentools.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1914
- __hash__() (abjad.tools.timetokentools.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker method), 1917
- __hash__() (abjad.tools.timetokentools.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker method), 1921

__hash__() (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck.MarkupInventory.MarkupInventory.MarkupInventory method), 1961
 __hash__() (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck.Measure.AnonymousMeasure.AnonymousMeasure method), 1963
 __hash__() (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck.DynamicMarkHairpinCheck.Measure.D method), 1965
 __hash__() (abjad.tools.wellformednesstools.MisdurationMeasureCheck.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1967
 __hash__() (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck.PitchArray.PitchArray.PitchArray method), 1970
 __hash__() (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1972
 __hash__() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck.OctaveTranspositionMappingInventory.O method), 1974
 __hash__() (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck method), 1976
 __hash__() (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1979
 __hash__() (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck.Staff.Staff.Staff method), 1981
 __hash__() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 1983
 __hash__() (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck method), 1985
 __hash__() (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck.Score.Score.Score method), 1988
 __iadd__() (abjad.tools.containertools.Cluster.Cluster.Cluster.__iadd__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 300
 __iadd__() (abjad.tools.containertools.Container.Container.Container.__iadd__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 307
 __iadd__() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 346
 __iadd__() (abjad.tools.contexttools.Context.Context.Context.__iadd__() (abjad.tools.stafftools.Staff.Staff.Staff method), 356
 __iadd__() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 385
 __iadd__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.Tuplet.Tuplet.Tuplet method), 1630
 __iadd__() (abjad.tools.gracetools.Grace.Grace.Grace.__iadd__() (abjad.tools.voicetools.Voice.Voice.Voice method), 423
 __iadd__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.Cluster.Cluster.Cluster method), 633
 __iadd__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 813
 __iadd__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 817
 __iadd__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 830
 __iadd__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 842
 __iadd__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2023
 __iadd__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics.__imul__() (abjad.tools.gracetools.Grace.Grace.Grace method), 2051

__imul__	(abjad.tools.instrumenttools.InstrumentInventory.Inventory.Inventory)	(abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval)
method),	633	method), 1010
__imul__	(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock)	(abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass)
method),	813	method), 1012
__imul__	(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock)	(abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval)
method),	817	method), 1015
__imul__	(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile)	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass)
method),	830	method), 1017
__imul__	(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock)	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass)
method),	842	method), 1030
__imul__	(abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics)	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass)
method),	2023	method), 1045
__imul__	(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics)	__int__(abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval)
method),	2051	method), 1056
__imul__	(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory)	(abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass)
method),	898	method), 1058
__imul__	(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure)	(abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval)
method),	913	method), 1077
__imul__	(abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure)	(abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass)
method),	924	method), 1079
__imul__	(abjad.tools.measuretools.Measure.Measure.Measure)	__int__(abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval)
method),	932	method), 1082
__imul__	(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping)	(abjad.tools.pitchtools.OctaveTranspositionMappingClass.OctaveTranspositionMappingClass)
method),	1156	method), 1084
__imul__	(abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory)	(abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch)
method),	1163	method), 1097
__imul__	(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory)	(abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass)
method),	1170	method), 1099
__imul__	(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff)	(abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch)
method),	1257	method), 1124
__imul__	(abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory)	(abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass)
method),	1266	method), 1126
__imul__	(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff)	(abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch)
method),	1278	method), 1129
__imul__	(abjad.tools.scoretools.Score.Score.Score)	__int__(abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass)
method),	1288	method), 1131
__imul__	(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup)	(abjad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch)
method),	1298	method), 1150
__imul__	(abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList)	(abjad.tools.pitchtools.NumberedDiatonicPitchClass.NumberedDiatonicPitchClass)
method),	1746	method), 1152
__imul__	(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff)	(abjad.tools.constrainttools.Domain.Domain.Domain)
method),	1528	method), 1994
__imul__	(abjad.tools.stafftools.Staff.Staff.Staff)	__iter__(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver)
method),	1539	method), 1997
__imul__	(abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet)	(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver)
method),	1574	method), 2015
__imul__	(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet)	__iter__(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory)
method),	1585	method), 346
__imul__	(abjad.tools.voicetools.Voice.Voice.Voice)	__iter__(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory)
method),	1606	method), 385
__int__	(abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval)	(abjad.tools.idestools.ChromaticInterval.ChromaticInterval)
method),	993	method), 1627
__int__	(abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass)	(abjad.tools.idestools.ChromaticIntervalClass.ChromaticIntervalClass)
method),	995	method), 1630

__iter__() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.toInheritanceGraph method), 1652

__iter__() (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.toModuleCrawler method), 1654

__iter__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.toInstrumentInventory method), 634

__iter__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.toBookBlock method), 813

__iter__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.toBookpartBlock method), 818

__iter__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.toLilyPondFile method), 830

__iter__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.toScoreBlock method), 842

__iter__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.toMarkupInventory method), 899

__iter__() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.toHarmonicChromaticIntervalClassVector method), 999

__iter__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.toHarmonicChromaticIntervalSegment method), 1003

__iter__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.toHarmonicChromaticIntervalSet method), 1006

__iter__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.toHarmonicDiatonicIntervalClassSet method), 1020

__iter__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.toHarmonicDiatonicIntervalSegment method), 1024

__iter__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1028

__iter__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1033

__iter__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1037

__iter__() (abjad.tools.pitchtools.InversionEquivalentChromaticInterval.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1042

__iter__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1048

__iter__() (abjad.tools.pitchtools.InversionEquivalentDiatonicInterval.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1053

__iter__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1061

__iter__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1066

__iter__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1069

__iter__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1073

__iter__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1088

__iter__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1092

__iter__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1103

__iter__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.HarmonicDiatonicIntervalSet.toHarmonicDiatonicIntervalSet method), 1106

- [__le__\(\) \(abjad.tools.chordtools.Chord.Chord.Chord method\), 228](#)
- [__le__\(\) \(abjad.tools.componenttools.Component.Component.Component method\), 237](#)
- [__le__\(\) \(abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint method\), 1992](#)
- [__le__\(\) \(abjad.tools.constrainttools.Domain.Domain.Domain method\), 1994](#)
- [__le__\(\) \(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver method\), 1997](#)
- [__le__\(\) \(abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint method\), 2000](#)
- [__le__\(\) \(abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint method\), 2003](#)
- [__le__\(\) \(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method\), 2005](#)
- [__le__\(\) \(abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint method\), 2009](#)
- [__le__\(\) \(abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint method\), 2012](#)
- [__le__\(\) \(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver method\), 2015](#)
- [__le__\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster method\), 300](#)
- [__le__\(\) \(abjad.tools.containertools.Container.Container.Container method\), 308](#)
- [__le__\(\) \(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method\), 342](#)
- [__le__\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method\), 346](#)
- [__le__\(\) \(abjad.tools.contexttools.Context.Context.Context method\), 356](#)
- [__le__\(\) \(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method\), 359](#)
- [__le__\(\) \(abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark method\), 363](#)
- [__le__\(\) \(abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark method\), 368](#)
- [__le__\(\) \(abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark method\), 372](#)
- [__le__\(\) \(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark method\), 376](#)
- [__le__\(\) \(abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method\), 381](#)
- [__le__\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method\), 385](#)
- [__le__\(\) \(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark method\), 391](#)
- [__le__\(\) \(abjad.tools.datastructuretools.Digraph.Digraph.Digraph method\), 1623](#)
- [__le__\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method\), 1627](#)
- [__le__\(\) \(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method\), 1630](#)
- [__le__\(\) \(abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler method\), 1634](#)
- [__le__\(\) \(abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator method\), 1636](#)
- [__le__\(\) \(abjad.tools.documentationtools.AbjadIndex.AbjadIndex.AbjadIndex method\), 1638](#)
- [__le__\(\) \(abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler method\), 1641](#)
- [__le__\(\) \(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter method\), 1643](#)
- [__le__\(\) \(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler method\), 1645](#)
- [__le__\(\) \(abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter method\), 1647](#)
- [__le__\(\) \(abjad.tools.documentationtools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint method\), 1652](#)
- [__le__\(\) \(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method\), 1654](#)
- [__le__\(\) \(abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler method\), 1657](#)
- [__le__\(\) \(abjad.tools.durationtools.Offset.Offset.Offset method\), 1666](#)
- [__le__\(\) \(abjad.tools.gracetools.Grace.Grace.Grace method\), 423](#)
- [__le__\(\) \(abjad.tools.instrumenttools.Accordion.Accordion.Accordion method\), 434](#)
- [__le__\(\) \(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method\), 441](#)
- [__le__\(\) \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method\), 448](#)
- [__le__\(\) \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method\), 455](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method\), 462](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method\), 469](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method\), 475](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method\), 483](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method\), 490](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method\), 497](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method\), 504](#)
- [__le__\(\) \(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method\), 510](#)
- [__le__\(\) \(abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method\), 518](#)
- [__le__\(\) \(abjad.tools.instrumenttools.Cello.Cello.Cello method\), 524](#)

__le__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 532

__le__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538

__le__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 546

__le__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 553

__le__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 560

__le__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 567

__le__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573

__le__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 581

__le__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 588

__le__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594

__le__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601

__le__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608

__le__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615

__le__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 623

__le__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 630

__le__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 634

__le__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641

__le__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648

__le__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 656

__le__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 663

__le__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 670

__le__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 677

__le__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 684

__le__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690

__le__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 698

__le__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 705

__le__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711

__le__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718

__le__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725

__le__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732

__le__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739

__le__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 746

__le__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 753

__le__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760

__le__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__le__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 770

__le__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 809

__le__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 813

__le__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 818

__le__() (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken method), 822

__le__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 830

__le__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 833

__le__() (abjad.tools.lilypondfiletools.LilyPondHeaderToken.LilyPondHeaderToken.LilyPondHeaderToken method), 835

__le__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 842

__le__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 1706

__le__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2023

__le__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender method), 2028

__le__() (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen method), 2032

__le__() (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace method), 2037

__le__() (abjad.tools.lyricstools.LyricText.LyricText.LyricText method), 2041

__le__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2051

__le__() (abjad.tools.marktools.Annotation.Annotation.Annotation method), 848

__le__() (abjad.tools.marktools.Articulation.Articulation.Articulation method), 851

__le__() (abjad.tools.marktools.BarLine.BarLine.BarLine method), 856

- `__le__()` (abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 859
- `__le__()` (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 863
- `__le__()` (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment method), 866
- `__le__()` (abjad.tools.marktools.Mark.Mark.Mark method), 869
- `__le__()` (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo method), 872
- `__le__()` (abjad.tools.markuptools.Markup.Markup.Markup method), 892
- `__le__()` (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand method), 895
- `__le__()` (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 899
- `__le__()` (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 913
- `__le__()` (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 924
- `__le__()` (abjad.tools.measuretools.Measure.Measure.Measure method), 933
- `__le__()` (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 969
- `__le__()` (abjad.tools.notetools.Note.Note.Note method), 975
- `__le__()` (abjad.tools.notetools.NoteHead.NoteHead.NoteHead method), 979
- `__le__()` (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1731
- `__le__()` (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 1733
- `__le__()` (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1736
- `__le__()` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1739
- `__le__()` (abjad.tools.pitchtools.Accidental.Accidental.Accidental method), 991
- `__le__()` (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval method), 993
- `__le__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass method), 995
- `__le__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 999
- `__le__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1003
- `__le__()` (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1007
- `__le__()` (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval.HarmonicCounterpointInterval method), 1010
- `__le__()` (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass method), 1012
- `__le__()` (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval method), 1015
- `__le__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1017
- `__le__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1020
- `__le__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1024
- `__le__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1028
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1030
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1033
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1037
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1042
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1045
- `__le__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method), 1048
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method), 1056
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass method), 1058
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1061
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1066
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1069
- `__le__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1073
- `__le__()` (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.MelodicCounterpointInterval method), 1077
- `__le__()` (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass method), 1079
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1082
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass method), 1085
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet method), 1088
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1092
- `__le__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1097
- `__le__()` (abjad.tools.pitchtools.NamedChromaticIntervalClass.NamedChromaticIntervalClass.NamedChromaticIntervalClass method), 1099
- `__le__()` (abjad.tools.pitchtools.NamedChromaticIntervalClassSet.NamedChromaticIntervalClassSet.NamedChromaticIntervalClassSet method), 1103
- `__le__()` (abjad.tools.pitchtools.NamedChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment.NamedChromaticIntervalClassSegment method), 1107

__le__() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1340

__le__() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1350

__le__() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker method), 1360

__le__() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DisabledTimeTokenMaker.DisabledTimeTokenMaker.DisabledTimeTokenMaker method), 1370

__le__() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker.NoteFilledTimeTokenMaker method), 1377

__le__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker.OutputBurnishedSignalFilledTimeTokenMaker method), 1385

__le__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker.OutputIncisedNoteFilledTimeTokenMaker method), 1395

__le__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker.OutputIncisedRestFilledTimeTokenMaker method), 1402

__le__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner.TokenMaker.TokenMaker.TokenMaker method), 1410

__le__() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.RestMeasuredTokenMaker.RestMeasuredTokenMaker.RestMeasuredTokenMaker method), 1419

__le__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker method), 1428

__le__() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.TimeTokenMaker.TimeTokenMaker.TimeTokenMaker method), 1436

__le__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker method), 1444

__le__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker method), 1451

__le__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker method), 1459

__le__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner.abjad.tools.timetoken tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker method), 1467

__le__() (abjad.tools.spannertools.Spanner.Spanner.Spanner__le__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1473

__le__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator method), 1481

__le__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner.DoublingIndicator.DoublingIndicator.DoublingIndicator method), 1488

__le__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator method), 1496

__le__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1503

__le__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.abjad.tools.tonalitytools.Mode.Mode.Mode method), 1529

__le__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1539

__le__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner__le__() (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator method), 1549

__le__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.abjad.tools.tonalitytools.Scale.Scale.Scale method), 1843

__le__() (abjad.tools.timeintervaltools.TimeIntervalAggregate.Mixin.Mixin.Mixin.abjad.tools.tonalitytools.ScaleAggregate.Mixin.Mixin.Mixin method), 1835

__le__() (abjad.tools.timeintervaltools.TimeIntervalMixin.Mixin.Mixin.abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator method), 1838

- `__le__()` (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction method), 2102
- `__le__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1574
- `__le__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1585
- `__le__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment method), 1943
- `__le__()` (abjad.tools.voicetools.Voice.Voice.Voice method), 1606
- `__le__()` (abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck method), 1956
- `__le__()` (abjad.tools.wellformednesstools.Check.Check.Check method), 1954
- `__le__()` (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck method), 1958
- `__le__()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck method), 1961
- `__le__()` (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck method), 1963
- `__le__()` (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck method), 1965
- `__le__()` (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.MisduratedMeasureCheck method), 1967
- `__le__()` (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck method), 1970
- `__le__()` (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck method), 1972
- `__le__()` (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck method), 1974
- `__le__()` (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck method), 1976
- `__le__()` (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck method), 1979
- `__le__()` (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck method), 1981
- `__le__()` (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 1983
- `__le__()` (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck method), 1985
- `__le__()` (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck method), 1988
- `__len__()` (abjad.tools.chordtools.Chord.Chord.Chord method), 229
- `__len__()` (abjad.tools.constrainttools.Domain.Domain.Domain method), 1994
- `__len__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), 300
- `__len__()` (abjad.tools.containertools.Container.Container.Container method), 308
- `__len__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 346
- `__len__()` (abjad.tools.contexttools.Context.Context.Context method), 356
- `__len__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1631
- `__len__()` (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1652
- `__len__()` (abjad.tools.gracetools.Grace.Grace.Grace method), 424
- `__len__()` (abjad.tools.olasinstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck method), 634
- `__len__()` (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 813
- `__len__()` (abjad.tools.lilypondfiletools.SpannerCheck.SpannerCheck.SpannerCheck method), 818
- `__len__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 830
- `__len__()` (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 842
- `__len__()` (abjad.tools.lyrics.AcadHymnicAcadHymnicAcadHymnic method), 2023
- `__len__()` (abjad.tools.lyrics.MisdurationMeasureCheck.MisdurationMeasureCheck.MisdurationMeasureCheck method), 2051
- `__len__()` (abjad.tools.lyrics.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck method), 899
- `__len__()` (abjad.tools.lyrics.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck method), 913
- `__len__()` (abjad.tools.lyrics.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck method), 924
- `__len__()` (abjad.tools.lyrics.MissingParentCheck.MissingParentCheck.MissingParentCheck method), 933
- `__len__()` (abjad.tools.lyrics.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck method), 1739
- `__len__()` (abjad.tools.lyrics.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck method), 1000
- `__len__()` (abjad.tools.lyrics.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 1003
- `__len__()` (abjad.tools.lyrics.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck method), 1007
- `__len__()` (abjad.tools.lyrics.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck method), 1020
- `__len__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1024
- `__len__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1028
- `__len__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1033
- `__len__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1037
- `__len__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass method), 1042
- `__len__()` (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass method), 1049

- __len__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVectorInversionTools.CyclicDiatonicIntervalClassVectorInversionTools method), 1053
- __len__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegmentAbjadTools.MelodicChromaticIntervalClassSegmentCyclicMelodicChromaticIntervalClassSegment method), 1061
- __len__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVectorAbjadTools.MelodicChromaticIntervalClassVectorCyclicMelodicChromaticIntervalClassVector method), 1066
- __len__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegmentAbjadTools.MelodicChromaticIntervalSegment method), 1070
- __len__() (abjad.tools.pitchtools.MelodicChromaticIntervalSetAbjadTools.MelodicChromaticIntervalSet method), 1073
- __len__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegmentAbjadTools.MelodicDiatonicIntervalSegment method), 1088
- __len__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSetAbjadTools.MelodicDiatonicIntervalSet method), 1092
- __len__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegmentAbjadTools.NamedChromaticPitchClassSegment method), 1103
- __len__() (abjad.tools.pitchtools.NamedChromaticPitchClassSetAbjadTools.NamedChromaticPitchClassSet method), 1107
- __len__() (abjad.tools.pitchtools.NamedChromaticPitchSegmentAbjadTools.NamedChromaticPitchSegment method), 1111
- __len__() (abjad.tools.pitchtools.NamedChromaticPitchSetAbjadTools.NamedChromaticPitchSet method), 1114
- __len__() (abjad.tools.pitchtools.NamedChromaticPitchVectorAbjadTools.NamedChromaticPitchVector method), 1119
- __len__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegmentAbjadTools.NumberedChromaticPitchClassSegment method), 1137
- __len__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSetAbjadTools.NumberedChromaticPitchClassSet method), 1142
- __len__() (abjad.tools.pitchtools.NumberedChromaticPitchClassVectorAbjadTools.NumberedChromaticPitchClassVector method), 1147
- __len__() (abjad.tools.pitchtools.OctaveTranspositionMappingAbjadTools.OctaveTranspositionMapping method), 1156
- __len__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventoryAbjadTools.OctaveTranspositionMappingInventory method), 1163
- __len__() (abjad.tools.pitchtools.PitchRangeInventoryAbjadTools.PitchRangeInventory method), 1170
- __len__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1175
- __len__() (abjad.tools.quantizationtools.QGrid.QGrid.QGrid__len__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method), 2057
- __len__() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTreeAbjadTools.QGridSearchTree method), 2066
- __len__() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookupAbjadTools.QGridTempoLookup method), 2070
- __len__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff__len__() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1257
- __len__() (abjad.tools.scoretools.PerformerInventoryAbjadTools.PerformerInventory method), 1267
- __len__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff__len__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method), 1278
- __len__() (abjad.tools.scoretools.Score.Score.Score__len__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1288
- __len__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup__len__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1299

- `__len__()` (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.method), 1529
- `__len__()` (abjad.tools.stafftools.Staff.Staff.Staff.method), 1539
- `__len__()` (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.method), 1549
- `__len__()` (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.method), 1843
- `__len__()` (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree.method), 1857
- `__len__()` (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.method), 1881
- `__len__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.method), 2077
- `__len__()` (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator.method), 2081
- `__len__()` (abjad.tools.tonalitytools.Mode.Mode.Mode.method), 2089
- `__len__()` (abjad.tools.tonalitytools.Scale.Scale.Scale.method), 2097
- `__len__()` (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.method), 1574
- `__len__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet.method), 1585
- `__len__()` (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment.method), 1943
- `__len__()` (abjad.tools.voicetools.Voice.Voice.Voice.method), 1606
- `__lt__()` (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject.method), 1613
- `__lt__()` (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.method), 1614
- `__lt__()` (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.method), 1616
- `__lt__()` (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.method), 1618
- `__lt__()` (abjad.tools.chordtools.Chord.Chord.Chord.method), 229
- `__lt__()` (abjad.tools.componenttools.Component.Component.Component.method), 237
- `__lt__()` (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint.method), 1992
- `__lt__()` (abjad.tools.constrainttools.Domain.Domain.Domain.method), 1994
- `__lt__()` (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.method), 1997
- `__lt__()` (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint.method), 2000
- `__lt__()` (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint.method), 2003
- `__lt__()` (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.method), 2005
- `__lt__()` (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.method), 2009
- `__lt__()` (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.method), 2012
- `__lt__()` (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.method), 2015
- `__lt__()` (abjad.tools.containertools.Cluster.Cluster.Cluster.method), 300
- `__lt__()` (abjad.tools.containertools.Container.Container.Container.method), 308
- `__lt__()` (abjad.tools.cleftools.ClefMark.ClefMark.ClefMark.method), 342
- `__lt__()` (abjad.tools.cleftools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.method), 346
- `__lt__()` (abjad.tools.contexttools.Context.Context.Context.method), 356
- `__lt__()` (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark.method), 359
- `__lt__()` (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark.method), 363
- `__lt__()` (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.method), 368
- `__lt__()` (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark.method), 372
- `__lt__()` (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.method), 376
- `__lt__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.method), 381
- `__lt__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.method), 385
- `__lt__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.method), 391
- `__lt__()` (abjad.tools.abctools.AbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.method), 1623
- `__lt__()` (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject.method), 1627
- `__lt__()` (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.method), 1631
- `__lt__()` (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.method), 1634
- `__lt__()` (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator.method), 1636
- `__lt__()` (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler.method), 1638
- `__lt__()` (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter.method), 1641
- `__lt__()` (abjad.tools.documentationtools.Documenter.Documenter.Documenter.method), 1643
- `__lt__()` (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler.method), 1645
- `__lt__()` (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter.method), 1647
- `__lt__()` (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph.method), 1652
- `__lt__()` (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler.method), 1654

__lt__() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1657

__lt__() (abjad.tools.durationtools.Duration.Duration.Duration method), 1661

__lt__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1666

__lt__() (abjad.tools.gracetools.Grace.Grace.Grace method), 424

__lt__() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 434

__lt__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 441

__lt__() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 448

__lt__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 455

__lt__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 462

__lt__() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method), 469

__lt__() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method), 476

__lt__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 483

__lt__() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 490

__lt__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 497

__lt__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 504

__lt__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 511

__lt__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 518

__lt__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 525

__lt__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 532

__lt__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 539

__lt__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 546

__lt__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 553

__lt__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 560

__lt__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 567

__lt__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 574

__lt__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 581

__lt__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 588

__lt__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 595

__lt__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 601

__lt__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 609

__lt__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 616

__lt__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 623

__lt__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 630

__lt__() (abjad.tools.instrumenttools.Horn.Horn.Horn method), 634

__lt__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 642

__lt__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 649

__lt__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 656

__lt__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 663

__lt__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 670

__lt__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 677

__lt__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 684

__lt__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 691

__lt__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 698

__lt__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 705

__lt__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 712

__lt__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 719

__lt__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 726

__lt__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 733

__lt__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 740

__lt__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 747

__lt__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 754

__lt__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 761

__lt__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__lt__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 770

__lt__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegmentNumberedDiatonicPitchClassSet method), 1049	__lt__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVectorInversionEquivalentDiatonicPitchClassVectorNumberedDiatonicPitchClassVector method), 1053	__lt__() (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.NaturalDiatonicPitch.NumberedDiatonicPitch method), 1056	__lt__() (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass.NaturalDiatonicPitchClass.NaturalDiatonicPitchClass method), 1058	__lt__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.NaturalDiatonicPitchClassSegment method), 1061	__lt__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector.NaturalDiatonicPitchClassVector method), 1066	__lt__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.NaturalDiatonicPitchClassSegment method), 1070	__lt__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.NaturalDiatonicPitchClassRange method), 1073	__lt__() (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval.PitchRangeInventory.PitchRange method), 1077	__lt__() (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass.PitchRangeInventory.PitchRange method), 1079	__lt__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.QEvent.QEvent method), 1082	__lt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass.QEvent.QEvent method), 1085	__lt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.QEvent.QEvent method), 1088	__lt__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.QEvent.QEvent method), 1092	__lt__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.QGridTempoLookup.QGridTempoLookup method), 1097	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.QGridMeasureRest.MultiMeasureRest method), 1100	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.QGridMeasureRest.MultiMeasureRest method), 1103	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchClassSetName.NamedChromaticPitchClassSetName.QGridMeasureRest.MultiMeasureRest method), 1107	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchSegmentName.NamedChromaticPitchSegmentName.QGridMeasureRest.MultiMeasureRest method), 1111	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.QGridMeasureRest.MultiMeasureRest method), 1115	__lt__() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.QGridMeasureRest.MultiMeasureRest method), 1119	__lt__() (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.SchemePair.SchemePair.SchemePair method), 1124	__lt__() (abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass.QSchemeVector.SchemeVector.SchemeVector method), 1126	__lt__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.SchemeVectorConstant.SchemeVectorConstant method), 1129	__lt__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.SchemeVectorConstant.SchemeVectorConstant method), 1131	__lt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassColor.NamedChromaticPitchClassColor.QSchemeVectorConstant.SchemeVectorConstant method), 1133	__lt__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.QSchemeVectorConstant.SchemeVectorConstant method), 1137
---	--	---	--	---	--	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	---	--	--	--	--	--

__lt__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1257

__lt__() (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier.InstrumentationSpecifier method), 1260

__lt__() (abjad.tools.scoretools.Performer.Performer.Performer method), 1263

__lt__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1267

__lt__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1278

__lt__() (abjad.tools.scoretools.Score.Score.Score method), 1288

__lt__() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1299

__lt__() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1746

__lt__() (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1749

__lt__() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1759

__lt__() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1762

__lt__() (abjad.tools.sequencetools.Matrix.Matrix.Matrix method), 1765

__lt__() (abjad.tools.sequencetools.Tree.Tree.Tree method), 1774

__lt__() (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass method), 1821

__lt__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1823

__lt__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1311

__lt__() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1323

__lt__() (abjad.tools.spannertools.BraceSpanner.BraceSpanner.BraceSpanner method), 1331

__lt__() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 1340

__lt__() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1350

__lt__() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1360

__lt__() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1370

__lt__() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1378

__lt__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1385

__lt__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1395

__lt__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1402

__lt__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1410

__lt__() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1419

__lt__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1428

__lt__() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1436

__lt__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1444

__lt__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1452

__lt__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1459

__lt__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1467

__lt__() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1473

__lt__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1481

__lt__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1488

__lt__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1496

__lt__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1504

__lt__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1529

__lt__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1539

__lt__() (abjad.tools.spanspanners.TieSpanner.TieSpanner.TieSpanner method), 1550

__lt__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1843

__lt__() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1835

__lt__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin method), 1839

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1857

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1881

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1897

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1899

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1910

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1914

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1918

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1922

__lt__() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1902

__lt__() (abjad.tools.timetoken.tools.RestFilledTimeTokenMaker.RestFilledTimeTokenMaker.check.EmptyCheck.EmptyCheck method), 1925

__lt__() (abjad.tools.timetoken.tools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker.check.IntermediateCheck.IntermediateCheck method), 1928

__lt__() (abjad.tools.timetoken.tools.TimeTokenMaker.TimeTokenMaker.check.BadnessTools.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1904

__lt__() (abjad.tools.timetoken.tools.TokenBurnishedSignalFilledTimeTokenMaker.TokenBurnishedSignalFilledTimeTokenMaker.check.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1932

__lt__() (abjad.tools.timetoken.tools.TokenIncisedNoteFilledTimeTokenMaker.TokenIncisedNoteFilledTimeTokenMaker.check.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1936

__lt__() (abjad.tools.timetoken.tools.TokenIncisedRestFilledTimeTokenMaker.TokenIncisedRestFilledTimeTokenMaker.check.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1940

__lt__() (abjad.tools.timetoken.tools.TokenIncisedTimeTokenMaker.TokenIncisedTimeTokenMaker.check.MisdurationMeasureCheck.MisdurationMeasureCheck method), 1907

__lt__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck method), 2077

__lt__() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck method), 2082

__lt__() (abjad.tools.tonalitytools.DoublingIndicator.DoublingIndicator.abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck method), 2084

__lt__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck method), 2086

__lt__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck method), 2087

__lt__() (abjad.tools.tonalitytools.Mode.Mode.Mode __mod__() (abjad.tools.durationtools.Duration.Duration.Duration method), 2089

__lt__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.abjad.tools.wellformednesstools.Offset.Offset.Offset method), 2091

__lt__() (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.abjad.tools.wellformednesstools.Offset.Offset.Offset method), 2093

__lt__() (abjad.tools.tonalitytools.Scale.Scale.Scale __mul__() (abjad.tools.componenttools.Component.Component.Component method), 2097

__lt__() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree.abjad.tools.containertools.Cluster.Cluster.Cluster method), 2099

__lt__() (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.abjad.tools.containertools.Container.Container.Container method), 2101

__lt__() (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction.abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 2103

__lt__() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.abjad.tools.tuplettools.Context.Context.Context method), 1574

__lt__() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet __mul__() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark method), 1585

__lt__() (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.abjad.tools.verticalitytools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 1943

__lt__() (abjad.tools.voicetools.Voice.Voice.Voice __mul__() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1607

__lt__() (abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck method), 1956

__lt__() (abjad.tools.wellformednesstools.Check.Check.Check __mul__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1954

__lt__() (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck method), 1959

__lt__() (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck.InstrumentInventory.InstrumentInventory method), 1961

__mul__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 770

__mul__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 813

__mul__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 818

__mul__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 830

__mul__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 842

__mul__() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2023

__mul__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender method), 2028

__mul__() (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen method), 2032

__mul__() (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace method), 2037

__mul__() (abjad.tools.lyricstools.LyricText.LyricText.LyricText method), 2041

__mul__() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2052

__mul__() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 899

__mul__() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 914

__mul__() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 924

__mul__() (abjad.tools.measuretools.Measure.Measure.Measure method), 933

__mul__() (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 970

__mul__() (abjad.tools.notetools.Note.Note.Note method), 975

__mul__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1003

__mul__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1024

__mul__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment method), 1034

__mul__() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment method), 1049

__mul__() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1062

__mul__() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1070

__mul__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1083

__mul__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1088

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1103

__mul__() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1111

__mul__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1137

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1156

__mul__() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1163

__mul__() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory method), 1170

__mul__() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1176

__mul__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1216

__mul__() (abjad.tools.resttools.Rest.Rest.Rest method), 1221

__mul__() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1257

__mul__() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1267

__mul__() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1278

__mul__() (abjad.tools.scoretools.Score.Score.Score method), 1288

__mul__() (abjad.tools.stafftools.StaffGroup.StaffGroup.StaffGroup method), 1299

__mul__() (abjad.tools.stafftools.CyclicList.CyclicList.CyclicList method), 1746

__mul__() (abjad.tools.stafftools.CyclicTuple.CyclicTuple.CyclicTuple method), 1762

__mul__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1311

__mul__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1529

__mul__() (abjad.tools.stafftools.Staff.Staff.Staff method), 1539

__mul__() (abjad.tools.chordtools.ChordQualityIndicator.ChordQualityIndicator method), 2082

__mul__() (abjad.tools.chordtools.HarmonicChordQualityIndicator.HarmonicChordQualityIndicator method), 2097

__mul__() (abjad.tools.chordtools.FixedDurationTonalClassSegment.FixedDurationTonalClassSegment method), 1575

__mul__() (abjad.tools.chordtools.EquivalentTupletTupletIntervalClassSegment.EquivalentTupletTupletIntervalClassSegment method), 1585

__mul__() (abjad.tools.chordtools.VoiceClassSegment.VoiceClassSegment method), 1607

__mul__() (abjad.tools.chordtools.MelodicChordObject.MelodicChordObject method), 1613

__mul__() (abjad.tools.chordtools.AttributeEqualMelodicChordObject.AttributeEqualMelodicChordObject method), 1614

__mul__() (abjad.tools.chordtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1616

__mul__() (abjad.tools.chordtools.NamedChordClassSegment.NamedChordClassSegment method), 1618

__mul__() (abjad.tools.chordtools.NamedChordPitchSegment.NamedChordPitchSegment method), 229

- `__ne__()` (abjad.tools.componenttools.Component.Component.__ne__()
method), 237
- `__ne__()` (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint.
method), 1992
- `__ne__()` (abjad.tools.constrainttools.Domain.Domain.Domain.__ne__()
method), 1994
- `__ne__()` (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.
method), 1998
- `__ne__()` (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint.__ne__()
method), 2000
- `__ne__()` (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint.
method), 2003
- `__ne__()` (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.
method), 2006
- `__ne__()` (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.
method), 2009
- `__ne__()` (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.
method), 2012
- `__ne__()` (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.
method), 2015
- `__ne__()` (abjad.tools.containertools.Cluster.Cluster.Cluster.__ne__()
method), 301
- `__ne__()` (abjad.tools.containertools.Container.Container.Container.__ne__()
method), 308
- `__ne__()` (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark.__ne__()
method), 342
- `__ne__()` (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.__ne__()
method), 347
- `__ne__()` (abjad.tools.contexttools.Context.Context.Context.__ne__()
method), 356
- `__ne__()` (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark.__ne__()
method), 359
- `__ne__()` (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark.__ne__()
method), 363
- `__ne__()` (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.__ne__()
method), 368
- `__ne__()` (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark.__ne__()
method), 372
- `__ne__()` (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.__ne__()
method), 377
- `__ne__()` (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.__ne__()
method), 382
- `__ne__()` (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.__ne__()
method), 386
- `__ne__()` (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.__ne__()
method), 391
- `__ne__()` (abjad.tools.datastructuretools.Digraph.Digraph.Digraph.__ne__()
method), 1623
- `__ne__()` (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.__ne__()
method), 1627
- `__ne__()` (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.__ne__()
method), 1631
- `__ne__()` (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.__ne__()
method), 1634
- `__ne__()` (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator.__ne__()
method), 1636
- `__ne__()` (abjad.tools.documentationtools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint.
method), 1638
- `__ne__()` (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter.__ne__()
method), 1641
- `__ne__()` (abjad.tools.documentationtools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver.
method), 1643
- `__ne__()` (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler.__ne__()
method), 1645
- `__ne__()` (abjad.tools.documentationtools.GlobalConstraint.GlobalConstraint.GlobalConstraint.__ne__()
method), 1648
- `__ne__()` (abjad.tools.documentationtools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.
method), 1652
- `__ne__()` (abjad.tools.documentationtools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.
method), 1654
- `__ne__()` (abjad.tools.documentationtools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.
method), 1657
- `__ne__()` (abjad.tools.documentationtools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.
method), 1661
- `__ne__()` (abjad.tools.durationtools.Offset.Offset.Offset.__ne__()
method), 1666
- `__ne__()` (abjad.tools.gracetools.Grace.Grace.Grace.__ne__()
method), 424
- `__ne__()` (abjad.tools.instrumenttools.Accordion.Accordion.Accordion.__ne__()
method), 434
- `__ne__()` (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute.__ne__()
method), 441
- `__ne__()` (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone.__ne__()
method), 448
- `__ne__()` (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.__ne__()
method), 455
- `__ne__()` (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet.__ne__()
method), 462
- `__ne__()` (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone.__ne__()
method), 469
- `__ne__()` (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice.__ne__()
method), 476
- `__ne__()` (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.__ne__()
method), 483
- `__ne__()` (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.__ne__()
method), 490
- `__ne__()` (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone.__ne__()
method), 497
- `__ne__()` (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone.__ne__()
method), 504
- `__ne__()` (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice.__ne__()
method), 511
- `__ne__()` (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon.__ne__()
method), 518
- `__ne__()` (abjad.tools.instrumenttools.Cello.Cello.Cello.__ne__()
method), 525
- `__ne__()` (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA.__ne__()
method), 532

[__ne__\(\)](#) (abjad.tools.instrumenttools.Contrabass.Contrabass.[ContraBass](#)(abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 539 [method](#)), 726
[__ne__\(\)](#) (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet([Clarinet](#)(abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion method), 546 [method](#)), 733
[__ne__\(\)](#) (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute([Flute](#)(abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 553 [method](#)), 740
[__ne__\(\)](#) (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone([Saxophone](#)(abjad.tools.instrumenttools.Viola.Viola.Viola method), 560 [method](#)), 747
[__ne__\(\)](#) (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon([Bassoon](#)(abjad.tools.instrumenttools.Violin.Violin.Violin method), 567 [method](#)), 754
[__ne__\(\)](#) (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice([Voice](#)(abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 574 [method](#)), 761
[__ne__\(\)](#) (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet([Clarinet](#)(abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 581 [method](#)), 1699
[__ne__\(\)](#) (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn([EnglishHorn](#)(abjad.tools.leaftools.Leaf.Leaf.Leaf method), 588 [method](#)), 770
[__ne__\(\)](#) (abjad.tools.instrumenttools.Flute.Flute.Flute [__ne__\(\)](#) (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken method), 595 [method](#)), 809
[__ne__\(\)](#) (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn([FrenchHorn](#)(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 602 [method](#)), 813
[__ne__\(\)](#) (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel([Glockenspiel](#)(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 609 [method](#)), 818
[__ne__\(\)](#) (abjad.tools.instrumenttools.Guitar.Guitar.Guitar [__ne__\(\)](#) (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 616 [method](#)), 822
[__ne__\(\)](#) (abjad.tools.instrumenttools.Harp.Harp.Harp [__ne__\(\)](#) (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 623 [method](#)), 830
[__ne__\(\)](#) (abjad.tools.instrumenttools.Harpsichord.Harpsichord([Harpsichord](#)(abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken method), 630 [method](#)), 833
[__ne__\(\)](#) (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory([InstrumentInventory](#)(abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken method), 634 [method](#)), 835
[__ne__\(\)](#) (abjad.tools.instrumenttools.Marimba.Marimba.Marimba([Marimba](#)(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 642 [method](#)), 842
[__ne__\(\)](#) (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice([MezzoSopranoVoice](#)(abjad.tools.LilyPondParser.LilyPondParser.LilyPondParser method), 649 [method](#)), 1706
[__ne__\(\)](#) (abjad.tools.instrumenttools.Oboe.Oboe.Oboe [__ne__\(\)](#) (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics method), 656 [method](#)), 2023
[__ne__\(\)](#) (abjad.tools.instrumenttools.Piano.Piano.Piano [__ne__\(\)](#) (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender method), 663 [method](#)), 2028
[__ne__\(\)](#) (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo [__ne__\(\)](#) (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen method), 670 [method](#)), 2032
[__ne__\(\)](#) (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone([Saxophone](#)(abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace method), 677 [method](#)), 2037
[__ne__\(\)](#) (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone([SopranoSaxophone](#)(abjad.tools.lyrictools.LyricText.LyricText.LyricText method), 684 [method](#)), 2041
[__ne__\(\)](#) (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice([SopranoVoice](#)(abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics method), 691 [method](#)), 2052
[__ne__\(\)](#) (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone([TenorSaxophone](#)(abjad.tools.Annotation.Annotation.Annotation method), 698 [method](#)), 848
[__ne__\(\)](#) (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone([Trombone](#)(abjad.tools.Annotation.Annotation.Annotation method), 705 [method](#)), 852
[__ne__\(\)](#) (abjad.tools.instrumenttools.TenorVoice.TenorVoice([TenorVoice](#)(abjad.tools.marktools.BarLine.BarLine.BarLine method), 712 [method](#)), 856
[__ne__\(\)](#) (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet([Trumpet](#)(abjad.tools.marktools.BendAfter.BendAfter.BendAfter method), 719 [method](#)), 859

__ne__() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.__ne__() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner method), 1350 method), 1881

__ne__() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.__ne__() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner method), 1360 method), 1897

__ne__() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.__ne__() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1370 method), 1899

__ne__() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.__ne__() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner method), 1378 method), 1910

__ne__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.__ne__() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner method), 1385 method), 1914

__ne__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.__ne__() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner method), 1395 method), 1918

__ne__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.__ne__() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner method), 1403 method), 1922

__ne__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.__ne__() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1410 method), 1902

__ne__() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.__ne__() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1419 method), 1925

__ne__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.__ne__() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner method), 1428 method), 1928

__ne__() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.__ne__() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner method), 1436 method), 1904

__ne__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.__ne__() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner method), 1444 method), 1932

__ne__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.__ne__() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1452 method), 1936

__ne__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.__ne__() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner method), 1460 method), 1940

__ne__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.__ne__() (abjad.tools.spannertools.SlurSpanner.SlurSpanner method), 1467 method), 1907

__ne__() (abjad.tools.spannertools.Spanner.Spanner.Spanner__ne__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 1473 method), 2077

__ne__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.__ne__() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner method), 1481 method), 2082

__ne__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.__ne__() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner method), 1488 method), 2084

__ne__() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner__ne__() (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator method), 1496 method), 2086

__ne__() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner__ne__() (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator method), 1504 method), 2087

__ne__() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff__ne__() (abjad.tools.tonalitytools.Mode.Mode.Mode method), 1529 method), 2089

__ne__() (abjad.tools.stafftools.Staff.Staff.Staff method), __ne__() (abjad.tools.tonalitytools.OmissionIndicator.OmissionIndicator.OmissionIndicator method), 1539 method), 2091

__ne__() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner__ne__() (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator method), 1550 method), 2093

__ne__() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval__ne__() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 1843 method), 2097

__ne__() (abjad.tools.timeintervaltools.TimeIntervalAggregatorMixin.(abjad.tools.timeintervaltools.TimeIntervalAggregatorMixin.__ne__() (abjad.tools.timeintervaltools.TimeIntervalAggregatorMixin.__ne__() (abjad.tools.timeintervaltools.TimeIntervalAggregatorMixin.__ne__() (abjad.tools.timeintervaltools.TimeIntervalAggregatorMixin method), 1835 method), 2099

__ne__() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.__ne__() (abjad.tools.tonalitytools.MisSuspensionIndicator.MisSuspensionIndicator.MisSuspensionIndicator method), 1839 method), 2101

__ne__() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.__ne__() (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction method), 1857 method), 2103

`__or__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.Staff.Staff.Staff method), 1142
`__or__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.Staff.Staff.Staff method), 1539
`__or__()` (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest.TupletTools.FixedDurationTuplet.FixedDurationTuplet method), 1216
`__or__()` (abjad.tools.resttools.Rest.Rest.Rest method), `__radd__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1575
`__or__()` (abjad.tools.resttools.Rest.Rest.Rest method), `__radd__()` (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1585
`__or__()` (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass (abjad.tools.voicetools.Voice.Voice.Voice method), 1821
`__or__()` (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass (abjad.tools.voicetools.Voice.Voice.Voice method), 1607
`__or__()` (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression.ChromaticIntervalSet.HarmonicIntervalSet method), 1824
`__or__()` (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression.ChromaticIntervalSet.HarmonicIntervalSet method), 1007
`__or__()` (abjad.tools.skiptools.Skip.Skip.Skip method), `__rand__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1311
`__or__()` (abjad.tools.skiptools.Skip.Skip.Skip method), `__rand__()` (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1021
`__or__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 2077
`__or__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1028
`__pos__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rand__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1661
`__pos__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rand__()` (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1038
`__pos__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rand__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1666
`__pos__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rand__()` (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1074
`__pow__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rand__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1661
`__pow__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rand__()` (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1092
`__pow__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rand__()` (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1666
`__pow__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rand__()` (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1107
`__radd__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), `__rand__()` (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 301
`__radd__()` (abjad.tools.containertools.Cluster.Cluster.Cluster method), `__rand__()` (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1115
`__radd__()` (abjad.tools.containertools.Container.Container.Container method), `__rand__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 308
`__radd__()` (abjad.tools.containertools.Container.Container.Container method), `__rand__()` (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1142
`__radd__()` (abjad.tools.contexttools.Context.Context.Context method), `__rand__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 356
`__radd__()` (abjad.tools.contexttools.Context.Context.Context method), `__rand__()` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 2078
`__radd__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rdiv__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 1661
`__radd__()` (abjad.tools.durationtools.Duration.Duration.Duration method), `__rdiv__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 1661
`__radd__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rdiv__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 1667
`__radd__()` (abjad.tools.durationtools.Offset.Offset.Offset method), `__rdiv__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 1667
`__radd__()` (abjad.tools.gracetools.Grace.Grace.Grace method), `__rdivmod__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 424
`__radd__()` (abjad.tools.gracetools.Grace.Grace.Grace method), `__rdivmod__()` (abjad.tools.durationtools.Duration.Duration.Duration method), 1661
`__radd__()` (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), `__rdivmod__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 2024
`__radd__()` (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), `__rdivmod__()` (abjad.tools.durationtools.Offset.Offset.Offset method), 1667
`__radd__()` (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), `__repr__()` (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject method), 2052
`__radd__()` (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), `__repr__()` (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject method), 1613
`__radd__()` (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure.MeasureEqualityAbjadObject.AttributeEqualityAbjadObject method), 914
`__radd__()` (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure.MeasureEqualityAbjadObject.AttributeEqualityAbjadObject method), 1614
`__radd__()` (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.MeasureEqualityAbjadObject.AttributeEqualityAbjadObject method), 924
`__radd__()` (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.MeasureEqualityAbjadObject.AttributeEqualityAbjadObject method), 1616
`__radd__()` (abjad.tools.measuretools.Measure.Measure.Measure method), `__repr__()` (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 933
`__radd__()` (abjad.tools.measuretools.Measure.Measure.Measure method), `__repr__()` (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 1618
`__radd__()` (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), `__repr__()` (abjad.tools.chordtools.Chord.Chord.Chord method), 1257
`__radd__()` (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), `__repr__()` (abjad.tools.chordtools.Chord.Chord.Chord method), 229
`__radd__()` (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), `__repr__()` (abjad.tools.componenttools.Component.Component.Component method), 1278
`__radd__()` (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), `__repr__()` (abjad.tools.componenttools.Component.Component.Component method), 238
`__radd__()` (abjad.tools.scoretools.Score.Score.Score method), `__repr__()` (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint method), 1289
`__radd__()` (abjad.tools.scoretools.Score.Score.Score method), `__repr__()` (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint method), 1992
`__radd__()` (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), `__repr__()` (abjad.tools.constrainttools.Domain.Domain.Domain method), 1299
`__radd__()` (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), `__repr__()` (abjad.tools.constrainttools.Domain.Domain.Domain method), 1994
`__radd__()` (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), `__repr__()` (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver method), 1529
`__radd__()` (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), `__repr__()` (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver method), 1998

__repr__	() (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint.method), 2001	__repr__	() (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint.method), 1645
__repr__	() (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint.method), 2003	__repr__	() (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.method), 1648
__repr__	() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint.method), 2006	__repr__	() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.method), 1652
__repr__	() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint.method), 2009	__repr__	() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.method), 1655
__repr__	() (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver.method), 2012	__repr__	() (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.method), 1658
__repr__	() (abjad.tools.containertools.Cluster.Cluster.Cluster_repr.method), 301	__repr__	() (abjad.tools.durationtools.Offset.Offset.Offset.method), 1661
__repr__	() (abjad.tools.containertools.Container.Container.Container_repr.method), 308	__repr__	() (abjad.tools.gracetools.Grace.Grace.Grace.method), 424
__repr__	() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark_repr.method), 342	__repr__	() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion.method), 434
__repr__	() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory_repr.method), 347	__repr__	() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute.method), 441
__repr__	() (abjad.tools.contexttools.Context.Context.Context_repr.method), 356	__repr__	() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone_repr.method), 448
__repr__	() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark_repr.method), 360	__repr__	() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone_repr.method), 455
__repr__	() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark_repr.method), 363	__repr__	() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet_repr.method), 462
__repr__	() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark_repr.method), 368	__repr__	() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone_repr.method), 469
__repr__	() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark_repr.method), 372	__repr__	() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice_repr.method), 476
__repr__	() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark_repr.method), 377	__repr__	() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet_repr.method), 483
__repr__	() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark_repr.method), 382	__repr__	() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute_repr.method), 490
__repr__	() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory_repr.method), 386	__repr__	() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone_repr.method), 497
__repr__	() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark_repr.method), 391	__repr__	() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone_repr.method), 504
__repr__	() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph_repr.method), 1623	__repr__	() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice_repr.method), 511
__repr__	() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary_repr.method), 1627	__repr__	() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon_repr.method), 518
__repr__	() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory_repr.method), 1631	__repr__	() (abjad.tools.instrumenttools.Cello.Cello.Cello_repr.method), 525
__repr__	() (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler_repr.method), 1634	__repr__	() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA_repr.method), 532
__repr__	() (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator_repr.method), 1636	__repr__	() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass_repr.method), 539
__repr__	() (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler_repr.method), 1638	__repr__	() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet_repr.method), 546
__repr__	() (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter_repr.method), 1641	__repr__	() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute_repr.method), 553
__repr__	() (abjad.tools.documentationtools.Documenter.Documenter.Documenter_repr.method), 1643	__repr__	() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone_repr.method), 560

__repr__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon.instrumenttools.Violin.Violin.Violin method), 567

__repr__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice.instrumenttools.Xylophone.Xylophone.Xylophone method), 574

__repr__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 581

__repr__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.leaftools.Leaf.Leaf.Leaf method), 588

__repr__() (abjad.tools.instrumenttools.Flute.Flute.Flute __repr__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 595

__repr__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.__repr__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 602

__repr__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.__repr__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 609

__repr__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar __repr__() (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock method), 616

__repr__() (abjad.tools.instrumenttools.Harp.Harp.Harp __repr__() (abjad.tools.lilypondfiletools.DateTimeToken.DateTimeToken.DateTimeToken method), 623

__repr__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord __repr__() (abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock method), 630

__repr__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.__repr__() (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock method), 634

__repr__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba __repr__() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 642

__repr__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.__repr__() (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken method), 649

__repr__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe __repr__() (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken method), 656

__repr__() (abjad.tools.instrumenttools.Piano.Piano.Piano __repr__() (abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock method), 663

__repr__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo __repr__() (abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock method), 670

__repr__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.__repr__() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 677

__repr__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.__repr__() (abjad.tools.lilypondfiletools.SyllableBlock.SyllableBlock.SyllableBlock method), 684

__repr__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 691

__repr__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 698

__repr__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 705

__repr__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 712

__repr__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet __repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 719

__repr__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba __repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 726

__repr__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 733

__repr__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.__repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 740

__repr__() (abjad.tools.instrumenttools.Viola.Viola.Viola __repr__() (abjad.tools.lilypondfiletools.LilyPondParser.LilyPondParser.LilyPondParser method), 747

2176 Index

[illegible]

<code>_repr__()</code>	(abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction)	(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile)
<code>method</code>	, 2103	method), 831
<code>_repr__()</code>	(abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet)	(abjad.tools.scoretools.ScoreBlock.ScoreBlock.ScoreBlock)
<code>method</code>	, 1575	method), 843
<code>_repr__()</code>	(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet)	<code>_reversed__()</code> (abjad.tools.markuptools.MarkupInventory.MarkupInventory)
<code>method</code>	, 1586	method), 899
<code>_repr__()</code>	(abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment)	(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping)
<code>method</code>	, 1943	method), 1157
<code>_repr__()</code>	(abjad.tools.voicetools.Voice.Voice.Voice)	<code>_reversed__()</code> (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory)
<code>method</code>	, 1607	method), 1163
<code>_repr__()</code>	(abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck)	(abjad.tools.rangeinventorytools.RangeInventory.RangeInventory.RangeInventory)
<code>method</code>	, 1957	method), 1170
<code>_repr__()</code>	(abjad.tools.wellformednesstools.Check.Check.Check)	<code>_reversed__()</code> (abjad.tools.scoretools.PerformerInventory.PerformerInventory)
<code>method</code>	, 1954	method), 1267
<code>_repr__()</code>	(abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck)	(abjad.tools.cylindrictools.CylindricalList.CylindricalList.CylindricalList)
<code>method</code>	, 1959	method), 1746
<code>_repr__()</code>	(abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck)	(abjad.tools.durationtools.Duration.Duration.Duration)
<code>method</code>	, 1961	method), 1661
<code>_repr__()</code>	(abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck)	<code>_offset__()</code> (abjad.tools.offsettools.Offset.Offset.Offset)
<code>method</code>	, 1963	method), 1667
<code>_repr__()</code>	(abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck)	(abjad.tools.durationtools.Hairpin.Hairpin.Hairpin)
<code>method</code>	, 1966	method), 1662
<code>_repr__()</code>	(abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.MisduratedMeasureCheck)	(abjad.tools.offsettools.OffsetCheck.OffsetCheck.OffsetCheck)
<code>method</code>	, 1968	method), 1667
<code>_repr__()</code>	(abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck)	(abjad.tools.checktools.MisfilledMeasureCheck.MisfilledMeasureCheck)
<code>method</code>	, 1970	method), 229
<code>_repr__()</code>	(abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck)	(abjad.tools.componenttools.Component.Component.Component)
<code>method</code>	, 1972	method), 238
<code>_repr__()</code>	(abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck)	(abjad.tools.flagtools.FlagCheck.FlagCheck.FlagCheck)
<code>method</code>	, 1975	method), 301
<code>_repr__()</code>	(abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck)	(abjad.tools.containertools.Container.Container.Container)
<code>method</code>	, 1977	method), 308
<code>_repr__()</code>	(abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck)	(abjad.tools.clefmarktools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory)
<code>method</code>	, 1979	method), 347
<code>_repr__()</code>	(abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck)	(abjad.tools.contexttools.Context.Context.Context)
<code>method</code>	, 1981	method), 356
<code>_repr__()</code>	(abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck)	(abjad.tools.tempooverlappingsystems.TempoOverlappingSystem.TempoOverlappingSystem)
<code>method</code>	, 1984	method), 386
<code>_repr__()</code>	(abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck)	(abjad.tools.objectiontools.ObjectionCheck.ObjectionCheck.ObjectionCheck)
<code>method</code>	, 1986	method), 1631
<code>_repr__()</code>	(abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck)	(abjad.tools.durationtools.ShortHairpinCheck.ShortHairpinCheck)
<code>method</code>	, 1988	method), 1662
<code>_reversed__()</code>	(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory)	<code>_offset__()</code> (abjad.tools.offsettools.Offset.Offset.Offset)
<code>method</code>	, 347	method), 1667
<code>_reversed__()</code>	(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory)	(abjad.tools.gracegrace.Grace.Grace.Grace)
<code>method</code>	, 386	method), 424
<code>_reversed__()</code>	(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory)	(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory)
<code>method</code>	, 1631	method), 634
<code>_reversed__()</code>	(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory)	(abjad.tools.leafftools.Leaf.Leaf.Leaf)
<code>method</code>	, 634	method), 770
<code>_reversed__()</code>	(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock)	(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock)
<code>method</code>	, 814	method), 814
<code>_reversed__()</code>	(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock)	(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock)
<code>method</code>	, 818	method), 818

[__ror__\(\)](#) (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.ChordClass.ChordClass.ChordClass method), 1142
[__ror__\(\)](#) (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass) (abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject method), 2078
[__rpow__\(\)](#) (abjad.tools.durationtools.Duration.Duration.Duration) (abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject method), 1662
[__rpow__\(\)](#) (abjad.tools.durationtools.Offset.Offset.Offset) (abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject method), 1667
[__rsub__\(\)](#) (abjad.tools.durationtools.Duration.Duration.Duration) (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 1662
[__rsub__\(\)](#) (abjad.tools.durationtools.Offset.Offset.Offset) (abjad.tools.chordtools.Chord.Chord.Chord method), 1667
[__rsub__\(\)](#) (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1007
[__rsub__\(\)](#) (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1021
[__rsub__\(\)](#) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1028
[__rsub__\(\)](#) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1038
[__rsub__\(\)](#) (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1074
[__rsub__\(\)](#) (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1092
[__rsub__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1107
[__rsub__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1115
[__rsub__\(\)](#) (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1142
[__rsub__\(\)](#) (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass) (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver method), 2078
[__rtruediv__\(\)](#) (abjad.tools.durationtools.Duration.Duration.Duration) (abjad.tools.containertools.Cluster.Cluster.Cluster method), 1662
[__rtruediv__\(\)](#) (abjad.tools.durationtools.Offset.Offset.Offset) (abjad.tools.containertools.Container.Container.Container method), 1667
[__rxor__\(\)](#) (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1007
[__rxor__\(\)](#) (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1021
[__rxor__\(\)](#) (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1029
[__rxor__\(\)](#) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1038
[__rxor__\(\)](#) (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1074
[__rxor__\(\)](#) (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1093
[__rxor__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1107
[__rxor__\(\)](#) (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1115
[__rxor__\(\)](#) (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1143

__setattr__(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.BassSaxophone.BassSaxophone.BassSaxophone.F
method), 386

__setattr__(abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.BassTrombone.BassTrombone.BassTrombone.Ba
method), 391

__setattr__(abjad.tools.datastructuretools.Digraph.Digraph.Digraph.__setattr__(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice
method), 1623

__setattr__(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.Bassoon.Bassoon.Bassoon
method), 1627

__setattr__(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.__setattr__(abjad.tools.instrumenttools.Cello.Cello.Cello
method), 1631

__setattr__(abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler.__setattr__(abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA
method), 1634

__setattr__(abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator.__setattr__(abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabas
method), 1637

__setattr__(abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler.__setattr__(abjad.tools.instrumenttools.ContrabassClarinet.ContrabassCla
method), 1638

__setattr__(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter.__setattr__(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.
method), 1641

__setattr__(abjad.tools.documentationtools.Documenter.Documenter.Documenter.__setattr__(abjad.tools.instrumenttools.ContrabassSaxophone.Contrabass
method), 1643

__setattr__(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler.__setattr__(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Co
method), 1645

__setattr__(abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter.__setattr__(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.Co
method), 1648

__setattr__(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph.__setattr__(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlat
method), 1653

__setattr__(abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler.__setattr__(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.English
method), 1655

__setattr__(abjad.tools.documentationtools.Pipe.Pipe.Pipe.__setattr__(abjad.tools.instrumenttools.Flute.Flute.Flute
method), 1658

__setattr__(abjad.tools.durationtools.Duration.Duration.Duration.__setattr__(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchH
method), 1662

__setattr__(abjad.tools.durationtools.Offset.Offset.Offset.__setattr__(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glock
method), 1668

__setattr__(abjad.tools.gracetools.Grace.Grace.Grace.__setattr__(abjad.tools.instrumenttools.Guitar.Guitar.Guitar
method), 424

__setattr__(abjad.tools.instrumenttools.Accordion.Accordion.Accordion.__setattr__(abjad.tools.instrumenttools.Harp.Harp.Harp
method), 434

__setattr__(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute.__setattr__(abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsic
method), 441

__setattr__(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone.__setattr__(abjad.tools.instrumenttools.InstrumentInventory.InstrumentIn
method), 448

__setattr__(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.__setattr__(abjad.tools.instrumenttools.Marimba.Marimba.Marimba
method), 455

__setattr__(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet.__setattr__(abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopra
method), 462

__setattr__(abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone.__setattr__(abjad.tools.instrumenttools.Oboe.Oboe.Oboe
method), 469

__setattr__(abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice.__setattr__(abjad.tools.instrumenttools.Piano.Piano.Piano
method), 476

__setattr__(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.__setattr__(abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo
method), 483

__setattr__(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.__setattr__(abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoS
method), 490

__setattr__(abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.LilyPondParser.LilyPondParser method), 684

__setattr__(abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.lyrictools.AddLyrics.AddLyrics.AddLyrics method), 691

__setattr__(abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.LyricExtender.LyricExtender.LyricExtender method), 698

__setattr__(abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.LyricHyphen.LyricHyphen.LyricHyphen method), 705

__setattr__(abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.LyricSpace.LyricSpace.LyricSpace method), 712

__setattr__(abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.LyricText.LyricText.LyricText method), 719

__setattr__(abjad.tools.instrumenttools.Tuba.Tuba.Tuba.Lyrics.Lyrics.Lyrics method), 726

__setattr__(abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.Annotation.Annotation.Annotation method), 733

__setattr__(abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.Articulation.Articulation.Articulation method), 740

__setattr__(abjad.tools.instrumenttools.Viola.Viola.Viola.BarLine.BarLine.BarLine method), 747

__setattr__(abjad.tools.instrumenttools.Violin.Violin.Violin.BendAfter.BendAfter.BendAfter method), 754

__setattr__(abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark method), 761

__setattr__(abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication.LilyPondComment.LilyPondComment.LilyPondComment method), 1699

__setattr__(abjad.tools.leaftools.Leaf.Leaf.Leaf.Mark.Mark.Mark method), 771

__setattr__(abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken.StemTremolo.StemTremolo.StemTremolo method), 809

__setattr__(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock.Markup.Markup.Markup method), 814

__setattr__(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock.MarkupCommand.MarkupCommand.MarkupCommand method), 819

__setattr__(abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock.MarkupInventory.MarkupInventory.MarkupInventory method), 820

__setattr__(abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 822

__setattr__(abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 824

__setattr__(abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock.Measure.Measure.Measure method), 826

__setattr__(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic method), 831

__setattr__(abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken.NoteHead.NoteHead.NoteHead method), 833

__setattr__(abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken.NoteHead.NoteHead.NoteHead method), 835

__setattr__(abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock.PitchArray.PitchArray.PitchArray method), 837

__setattr__(abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock.PitchArrayCell.PitchArrayCell.PitchArrayCell method), 838

__setattr__(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 843

[illegible]

__setattr__(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.CyclicMatrix.CyclicMatrix.CyclicMatrix method), 1171

__setattr__(abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow.CyclicTree.CyclicTree.CyclicTree method), 1176

__setattr__(abjad.tools.quantizationtools.QEvent.QEvent.QEvent.CyclicTuple.CyclicTuple.CyclicTuple method), 2054

__setattr__(abjad.tools.quantizationtools.QGrid.QGrid.QGrid.CyclicTuple.CyclicTuple.CyclicTuple method), 2058

__setattr__(abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer.QGridQuantizer.CyclicTuple.CyclicTuple.CyclicTuple method), 2061

__setattr__(abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree.ResidueClass.ResidueClass.ResidueClass method), 2066

__setattr__(abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup.QGridTempoLookup.ResidueClass.ResidueClass.ResidueClass method), 2071

__setattr__(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest.Skip.Skip.Skip method), 1217

__setattr__(abjad.tools.resttools.Rest.Rest.Rest.Skip.Skip.Skip method), 1221

__setattr__(abjad.tools.schemetools.Scheme.Scheme.Scheme.BracketSpanner.BracketSpanner.BracketSpanner method), 1228

__setattr__(abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 1230

__setattr__(abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1232

__setattr__(abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1234

__setattr__(abjad.tools.schemetools.SchemePair.SchemePair.SchemePair.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1236

__setattr__(abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1238

__setattr__(abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1241

__setattr__(abjad.tools.scoretemplatetools.ScoreTemplate.ScoreTemplate.ScoreTemplate.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1242

__setattr__(abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.ScoreTemplate.ScoreTemplate.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1245

__setattr__(abjad.tools.scoretemplatetools.TwoStaffPianoScoreTemplate.ScoreTemplate.ScoreTemplate.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1247

__setattr__(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1258

__setattr__(abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier.InstrumentationSpecifier.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1260

__setattr__(abjad.tools.scoretools.Performer.Performer.Performer.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1263

__setattr__(abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1267

__setattr__(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1279

__setattr__(abjad.tools.scoretools.Score.Score.Score.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1289

__setattr__(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup.SlurSpanner.SlurSpanner.SlurSpanner method), 1299

__setattr__(abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList.Spanner.Spanner.Spanner method), 1747

__setattr__(abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck.
method), 1982

__setattr__(abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck.
method), 1984

__setattr__(abjad.tools.wellformednesstools.OverlappingOctaveCheck.OverlappingOctaveCheck.OverlappingOctaveCheck.
method), 1986

__setattr__(abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck.
method), 1988

__setitem__(abjad.tools.chordtools.Chord.Chord.Chord __setitem__(abjad.tools.pitchtools.NumberedChromaticPitchClass Vector.
method), 229

__setitem__(abjad.tools.containertools.Cluster.Cluster.Cluster __setitem__(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveT
method), 301

__setitem__(abjad.tools.containertools.Container.Container.Container __setitem__(abjad.tools.pitchtools.OctaveTranspositionMappingInventory.
method), 308

__setitem__(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory.PitchRangeInventory.PitchRangeInven
method), 347

__setitem__(abjad.tools.contexttools.Context.Context.Context __setitem__(abjad.tools.quantizationtools.QGrid.QGrid.QGrid
method), 357

__setitem__(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.QGridSearchTree.QGridSearch
method), 386

__setitem__(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.QGridTempoLookup.QGridTem
method), 1627

__setitem__(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.GrandStaff.GrandStaff.GrandStaff
method), 1631

__setitem__(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph.PerformerInventory.PerformerInventor
method), 1653

__setitem__(abjad.tools.gracetools.Grace.Grace.Grace __setitem__(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff
method), 424

__setitem__(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.Score.Score.Score
method), 635

__setitem__(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock __setitem__(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup
method), 814

__setitem__(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock __setitem__(abjad.tools.scoretools.CyclicList.CyclicList.CyclicList
method), 819

__setitem__(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile __setitem__(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.Rhythmi
method), 831

__setitem__(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock __setitem__(abjad.tools.stafftools.Staff.Staff.Staff
method), 843

__setitem__(abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics __setitem__(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.Tim
method), 2024

__setitem__(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics __setitem__(abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.Tir
method), 2052

__setitem__(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory __setitem__(abjad.tools.timeintervaltools.FixedDurationTuplet.FixedDurationT
method), 900

__setitem__(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure __setitem__(abjad.tools.timeintervaltools.Tuplet.Tuplet.Tuplet
method), 914

__setitem__(abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure __setitem__(abjad.tools.timeintervaltools.Voice.Voice.Voice
method), 925

__setitem__(abjad.tools.measuretools.Measure.Measure.Measure __setitem__(abjad.tools.timeintervaltools.Voice.Voice.Voice
method), 934

__setitem__(abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray __setitem__(abjad.tools.timeintervaltools.Voice.Voice.Voice
method), 1731

__setitem__(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector.
method), 1000

__setslice__(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.Container.Container.Container method), 635

__setslice__(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock(abjad.tools.contexttools.ClefMark.ClefMark.ClefMark method), 814

__setslice__(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock(abjad.tools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 819

__setslice__(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile(abjad.tools.contexttools.Context.Context.Context method), 831

__setslice__(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock(abjad.tools.contexttools.ContextMark.ContextMark.ContextMark method), 843

__setslice__(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory(abjad.tools.markuptools.DynamicMark.DynamicMark.DynamicMark method), 900

__setslice__(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping(abjad.tools.pitchtools.OctaveTranspositionMapping.Instrument method), 1157

__setslice__(abjad.tools.pitchtools.OctaveTranspositionMapping.Instrument.OctaveTranspositionMapping.Instrument.OctaveTranspositionMapping method), 1164

__setslice__(abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory(abjad.tools.pitchtools.StaffChangeMark.StaffChangeMark.StaffChangeMark method), 1171

__setslice__(abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory(abjad.tools.scoretools.TempoMark.TempoMark.TempoMark method), 1268

__setslice__(abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList(abjad.tools.sequencetools.CyclicList.CyclicList method), 1747

__str__(abjad.tools.abctools.AbjadObject.AbjadObject.AbjadObject(abjad.tools.abctools.AbjadObject.AbjadObject method), 1613

__str__(abjad.tools.abctools.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject.AttributeEqualityAbjadObject method), 1615

__str__(abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject.ImmutableAbjadObject(abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject method), 1616

__str__(abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject method), 1618

__str__(abjad.tools.abctools.Chord.Chord.Chord __str__(abjad.tools.abctools.Chord.Chord.Chord method), 229

__str__(abjad.tools.componenttools.Component.Components.Components(abjad.tools.componenttools.Component.Components method), 238

__str__(abjad.tools.constrainttools.AbsoluteIndexConstraints.AbsoluteIndexConstraints.AbsoluteIndexConstraints(abjad.tools.constrainttools.AbsoluteIndexConstraints method), 1992

__str__(abjad.tools.constrainttools.Domain.Domain.Domain __str__(abjad.tools.constrainttools.Domain.Domain.Domain method), 1995

__str__(abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver(abjad.tools.constrainttools.FixedLengthStreamSolver method), 1998

__str__(abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint.GlobalConstraint(abjad.tools.constrainttools.GlobalConstraint method), 2001

__str__(abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint.GlobalCountsConstraint(abjad.tools.constrainttools.GlobalCountsConstraint method), 2003

__str__(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint.GlobalReferenceConstraint(abjad.tools.constrainttools.GlobalReferenceConstraint method), 2006

__str__(abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint.RelativeCountsConstraint(abjad.tools.constrainttools.RelativeCountsConstraint method), 2009

__str__(abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint.RelativeIndexConstraint(abjad.tools.constrainttools.RelativeIndexConstraint method), 2012

__str__(abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver(abjad.tools.constrainttools.VariableLengthStreamSolver method), 2016

__str__(abjad.tools.containertools.Cluster.Cluster.Cluster __str__(abjad.tools.containertools.Cluster.Cluster.Cluster method), 301

__str__(abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler(abjad.tools.documentationtools.APICrawler method), 1635

__str__(abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator(abjad.tools.documentationtools.AbjadAPIGenerator method), 1637

__str__(abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler(abjad.tools.documentationtools.ClassCrawler method), 1639

__str__(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter.ClassDocumenter(abjad.tools.documentationtools.ClassDocumenter method), 1641

__str__(abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler(abjad.tools.documentationtools.FunctionCrawler method), 1643

__str__(abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter.FunctionDocumenter(abjad.tools.documentationtools.FunctionDocumenter method), 1645

__str__(abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler(abjad.tools.documentationtools.ModuleCrawler method), 1648

__str__(abjad.tools.documentationtools.ModuleDocumenter.ModuleDocumenter.ModuleDocumenter(abjad.tools.documentationtools.ModuleDocumenter method), 1653

__str__(abjad.tools.durationtools.Offset.Offset.Offset __str__(abjad.tools.durationtools.Offset.Offset.Offset method), 1668

__str__() (abjad.tools.gracetools.Grace.Grace.Grace method), 425

__str__() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 434

__str__() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 441

__str__() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 448

__str__() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 455

__str__() (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 462

__str__() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone method), 469

__str__() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice method), 476

__str__() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 483

__str__() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 490

__str__() (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 497

__str__() (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 504

__str__() (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 511

__str__() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 518

__str__() (abjad.tools.instrumenttools.Cello.Cello.Cello method), 525

__str__() (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 532

__str__() (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 539

__str__() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 546

__str__() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 553

__str__() (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone method), 560

__str__() (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 567

__str__() (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 574

__str__() (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 581

__str__() (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 588

__str__() (abjad.tools.instrumenttools.Flute.Flute.Flute method), 595

__str__() (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 602

__str__() (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 609

__str__() (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 616

__str__() (abjad.tools.instrumenttools.Harp.Harp.Harp method), 623

__str__() (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 630

__str__() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 635

__str__() (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 642

__str__() (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 649

__str__() (abjad.tools.instrumenttools.Oboe.Oboe.Oboe method), 656

__str__() (abjad.tools.instrumenttools.Piano.Piano.Piano method), 663

__str__() (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 670

__str__() (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 677

__str__() (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 684

__str__() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 691

__str__() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 698

__str__() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 705

__str__() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 712

__str__() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 719

__str__() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba method), 726

__str__() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 733

__str__() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 740

__str__() (abjad.tools.instrumenttools.Viola.Viola.Viola method), 747

__str__() (abjad.tools.instrumenttools.Violin.Violin.Violin method), 754

__str__() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 761

__str__() (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication method), 1699

__str__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 771

__str__() (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken method), 810

__str__() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 814

__str__() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 819

__str__	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass) (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass method), 1043
__str__	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass method), 1046
__str__	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment method), 1049
__str__	(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector) (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector method), 1054
__str__	(abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval) (abjad.tools.pitchtools.MelodicChromaticInterval method), 1056
__str__	(abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass) (abjad.tools.pitchtools.MelodicChromaticIntervalClass method), 1059
__str__	(abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment) (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment method), 1062
__str__	(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector) (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector method), 1067
__str__	(abjad.tools.pitchtools.MelodicChromaticIntervalSegment) (abjad.tools.pitchtools.MelodicChromaticIntervalSegment method), 1070
__str__	(abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet) (abjad.tools.pitchtools.MelodicChromaticIntervalSet method), 1074
__str__	(abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval) (abjad.tools.pitchtools.MelodicCounterpointInterval method), 1077
__str__	(abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass) (abjad.tools.pitchtools.MelodicCounterpointIntervalClass method), 1080
__str__	(abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval) (abjad.tools.pitchtools.MelodicDiatonicInterval method), 1083
__str__	(abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass) (abjad.tools.pitchtools.MelodicDiatonicIntervalClass method), 1085
__str__	(abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment) (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment method), 1089
__str__	(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet) (abjad.tools.pitchtools.MelodicDiatonicIntervalSet method), 1093
__str__	(abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch) (abjad.tools.pitchtools.NamedChromaticPitch method), 1097
__str__	(abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass) (abjad.tools.pitchtools.NamedChromaticPitchClass method), 1100
__str__	(abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment) (abjad.tools.pitchtools.NamedChromaticPitchClassSegment method), 1104
__str__	(abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet) (abjad.tools.pitchtools.NamedChromaticPitchClassSet method), 1108
__str__	(abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment) (abjad.tools.pitchtools.NamedChromaticPitchSegment method), 1111
__str__	(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet) (abjad.tools.pitchtools.NamedChromaticPitchSet method), 1115
__str__	(abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector) (abjad.tools.pitchtools.NamedChromaticPitchVector method), 1120
__str__	(abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch) (abjad.tools.pitchtools.NamedDiatonicPitch method), 1125
__str__	(abjad.tools.pitchtools.NamedDiatonicPitchClass.NamedDiatonicPitchClass) (abjad.tools.pitchtools.NamedDiatonicPitchClass method), 1127
__str__	(abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch) (abjad.tools.pitchtools.NumberedChromaticPitch method), 1129
__str__	(abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass) (abjad.tools.pitchtools.NumberedChromaticPitchClass method), 1132

__sub__() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1008
 __sub__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1021
 __sub__() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1029
 __sub__() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1038
 __sub__() (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval.MelodicChromaticInterval method), 1056
 __sub__() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1074
 __sub__() (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.MelodicDiatonicInterval method), 1083
 __sub__() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1093
 __sub__() (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch method), 1098
 __sub__() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass method), 1100
 __sub__() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1108
 __sub__() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1115
 __sub__() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.NumberedChromaticPitch method), 1129
 __sub__() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1132
 __sub__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1143
 __sub__() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1222
 __sub__() (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest method), 1217
 __sub__() (abjad.tools.resttools.Rest.Rest.Rest method), 1222
 __sub__() (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1824
 __sub__() (abjad.tools.skiptools.Skip.Skip.Skip method), 1312
 __sub__() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 2078
 __truediv__() (abjad.tools.durationtools.Duration.Duration.Duration method), 1662
 __truediv__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1668
 __trunc__() (abjad.tools.durationtools.Duration.Duration.Duration method), 1662
 __trunc__() (abjad.tools.durationtools.Offset.Offset.Offset method), 1668
 __xor__() (abjad.tools.chordtools.Chord.Chord.Chord method), 230
 __xor__() (abjad.tools.leaftools.Leaf.Leaf.Leaf method), 771
 __xor__() (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender method), 2029
 __xor__() (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen method), 2033

A

AbjadAPIGenerator (class in abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator), 1635
 AbjadObject (class in abjad.tools.abctools.AbjadObject.AbjadObject), 1612
 AbjadRevisionToken (class in abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken), 808
 AbsoluteIndexConstraint (class in abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint), 1990
 accidentalAbjad (attribute), 2098

Accidental (class in abjad.tools.pitchtools.Accidental.Accidental),
989

all_are_contiguous_components() (in module abjad.tools.componenttools.all_are_contiguous_components),
240

Accordion (class in abjad.tools.instrumenttools.Accordion.Accordion),
428

all_are_contiguous_components_in_same_parent() (in module abjad.tools.componenttools.all_are_contiguous_components_in_same_parent),
240

add_artificial_harmonic_to_note() (in module abjad.tools.notetools.add_artificial_harmonic_to_note),
979

all_are_contiguous_components_in_same_score() (in module abjad.tools.componenttools.all_are_contiguous_components_in_same_score),
241

add_double_bar_to_end_of_score() (in module abjad.tools.scoretools.add_double_bar_to_end_of_score),
1300

all_are_contiguous_components_in_same_thread() (in module abjad.tools.componenttools.all_are_contiguous_components_in_same_thread),
241

add_markup_to_end_of_score() (in module abjad.tools.scoretools.add_markup_to_end_of_score),
1300

add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration() (in module abjad.tools.durationtools.all_are_duration_tokens),
1550

add_or_remove_tie_chain_notes_to_achieve_scaled_written_duration() (in module abjad.tools.durationtools.all_are_durations),
1669

add_or_remove_tie_chain_notes_to_achieve_written_duration() (in module abjad.tools.durationtools.all_are_equal),
1550

all_are_equal() (in module abjad.tools.sequencetools.all_are_equal),
1775

AddLyrics (class in abjad.tools.lyricstools.AddLyrics.AddLyrics),
2016

all_are_grace_containers() (in module abjad.tools.gracetools.all_are_grace_containers),
425

all_are_assignable_integers() (in module abjad.tools.sequencetools.all_are_assignable_integers),
1774

all_are_integer_equivalent_numbers() (in module abjad.tools.sequencetools.all_are_integer_equivalent_numbers),
1775

all_are_chords() (in module abjad.tools.chordtools.all_are_chords),
230

all_are_intervals_or_trees_or_empty() (in module abjad.tools.timeintervaltools.all_are_intervals_or_trees_or_empty),
1881

all_are_chromatic_pitch_class_name_octave_number_pairs() (in module abjad.tools.pitchtools.all_are_chromatic_pitch_class_name_octave_number_pairs),
1176

all_are_leaves() (in module abjad.tools.markuptools.all_are_leaves),
771

all_are_markup() (in module abjad.tools.markuptools.all_are_markup),
900

all_are_measures() (in module abjad.tools.measuretools.all_are_measures),
934

all_are_components() (in module abjad.tools.componenttools.all_are_components),
238

all_are_components_in_same_parent() (in module abjad.tools.componenttools.all_are_components_in_same_parent),
239

all_are_named_chromatic_pitch_tokens() (in module abjad.tools.pitchtools.all_are_named_chromatic_pitch_tokens),
1176

all_are_components_in_same_score() (in module abjad.tools.componenttools.all_are_components_in_same_score),
239

all_are_nonnegative_integer_equivalent_numbers() (in module abjad.tools.sequencetools.all_are_nonnegative_integer_equivalent_numbers),
1775

all_are_components_in_same_thread() (in module abjad.tools.componenttools.all_are_components_in_same_thread),
239

all_are_nonnegative_integer_powers_of_two() (in module abjad.tools.sequencetools.all_are_nonnegative_integer_powers_of_two),
1776

all_are_components_scalable_by_multiplier() (in module abjad.tools.componenttools.all_are_components_scalable_by_multiplier),
240

all_are_nonnegative_integers() (in module abjad.tools.sequencetools.all_are_nonnegative_integers),
1776

all_are_containers() (in module abjad.tools.containertools.all_are_containers),
309

all_are_notes() (in module jad.tools.notetools.all_are_notes), 980	ab-	all_clefs (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 452
all_are_numbers() (in module jad.tools.sequencetools.all_are_numbers), 1776	ab-	all_clefs (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone attribute), 466
all_are_orphan_components() (in module jad.tools.componenttools.all_are_orphan_components), 241	ab-	all_clefs (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice attribute), 472
all_are_pairs() (in module jad.tools.sequencetools.all_are_pairs), 1777	ab-	all_clefs (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 480
all_are_pairs_of_types() (in module jad.tools.sequencetools.all_are_pairs_of_types), 1777	ab-	all_clefs (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 487
all_are_pitch_arrays() (in module jad.tools.pitcharraytools.all_are_pitch_arrays), 1739	ab-	all_clefs (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 515
all_are_positive_integer_equivalent_numbers() (in module jad.tools.sequencetools.all_are_positive_integer_equivalent_numbers), 1778	ab-	all_clefs (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 494
all_are_positive_integers() (in module jad.tools.sequencetools.all_are_positive_integers), 1778	ab-	all_clefs (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 501
all_are_residue_class_expressions() (in module jad.tools.sievetools.all_are_residue_class_expressions), 1824	ab-	all_clefs (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 507
all_are_rests() (in module jad.tools.resttools.all_are_rests), 1222	ab-	all_clefs (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 459
all_are_scores() (in module jad.tools.scoretools.all_are_scores), 1301	ab-	all_clefs (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 521
all_are_skips() (in module jad.tools.skiptools.all_are_skips), 1312	ab-	all_clefs (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 529
all_are_spanners() (in module jad.tools.spannertools.all_are_spanners), 1504	ab-	all_clefs (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 535
all_are_staves() (in module jad.tools.stafftools.all_are_staves), 1540	ab-	all_clefs (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet attribute), 543
all_are_thread_contiguous_components() (in module jad.tools.componenttools.all_are_thread_contiguous_components), 242	ab-	all_clefs (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 550
all_are_tuplets() (in module jad.tools.tuplettools.all_are_tuplets), 1586	ab-	all_clefs (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 564
all_are_unequal() (in module jad.tools.sequencetools.all_are_unequal), 1778	ab-	all_clefs (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone attribute), 557
all_are_voices() (in module jad.tools.voicetools.all_are_voices), 1608	ab-	all_clefs (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 570
all_clefs (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 431	ab-	all_clefs (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 578
all_clefs (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 438	ab-	all_clefs (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 585
all_clefs (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 445	ab-	all_clefs (abjad.tools.instrumenttools.Flute.Flute.Flute attribute), 591
		all_clefs (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 598
		all_clefs (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 605
		all_clefs (abjad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 612
		all_clefs (abjad.tools.instrumenttools.Harp.Harp.Harp attribute), 620
		all_clefs (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 627
		all_clefs (abjad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 638

all_clefs (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice in abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone), 645

all_clefs (abjad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 653

all_clefs (abjad.tools.instrumenttools.Piano.Piano.Piano attribute), 660

all_clefs (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 667

all_clefs (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 674

all_clefs (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 681

all_clefs (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 687

all_clefs (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 695

all_clefs (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 702

all_clefs (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 708

all_clefs (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 715

all_clefs (abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 722

all_clefs (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 729

all_clefs (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 736

all_clefs (abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 743

all_clefs (abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 750

all_clefs (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 757

all_intervals_are_contiguous() (in module abjad.tools.timeintervaltools.all_intervals_are_contiguous), 1882

all_intervals_are_nonoverlapping() (in module abjad.tools.timeintervaltools.all_intervals_are_nonoverlapping), 1882

alpha() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1135

alpha() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1173

alphabetic_accidental_abbreviation (abjad.tools.pitchtools.Accidental.Accidental.Accidental attribute), 989

alphabetic_accidental_abbreviation_to_symbolic_accidental_string() (in module abjad.tools.pitchtools.alphabetic_accidental_abbreviation_to_symbolic_accidental_string), 1177

AltoFlute (class in abjad.tools.instrumenttools.AltoFlute.AltoFlute), 435

AltoSaxophone (class in abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone), 442

AltoTrombone (class in abjad.tools.instrumenttools.AltoTrombone.AltoTrombone), 449

analyze_chord() (in module abjad.tools.tonalitytools.analyze_chord), 2103

analyze_incomplete_chord() (in module abjad.tools.tonalitytools.analyze_incomplete_chord), 2103

analyze_incomplete_tonal_function() (in module abjad.tools.tonalitytools.analyze_incomplete_tonal_function), 2104

analyze_tonal_function() (in module abjad.tools.tonalitytools.analyze_tonal_function), 2104

Annotation (class in abjad.tools.marktools.Annotation.Annotation), 845

AnonymousMeasure (class in abjad.tools.measuretools AnonymousMeasure AnonymousMeasure), 904

APICrawler (class in abjad.tools.downloadtools APICrawler APICrawler), 1633

apply() (abjad.tools.chordtools.Chord.Chord.Chord method), 226

append() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 296

append() (abjad.tools.containertools.Container.Container.Container method), 304

append() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 344

append() (abjad.tools.contexttools.Context.Context.Context method), 352

append() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 383

append() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1628

append() (abjad.tools.downloadtools APICrawler APICrawler method), 1633

append() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 631

append() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 811

append() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 815

append() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 828

append() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 840

append() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2019

append() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2047
 append() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 896
 append() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 909
 append() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 920
 append() (abjad.tools.measuretools.Measure.Measure.Measure method), 929
 append() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn method), 1735
 append() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1738
 append() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1154
 append() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1160
 append() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1167
 append() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1253
 append() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1264
 append() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1274
 append() (abjad.tools.scoretools.Score.Score.Score method), 1284
 append() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1295
 append() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1744
 append() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1318
 append() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1326
 append() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1335
 append() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1345
 append() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1355
 append() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1365
 append() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1373
 append() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1381
 append() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1391
 append() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1398
 append() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1405
 append() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1415
 append() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1423
 append() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1431
 append() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1439
 append() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1447
 append() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1455
 append() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1463
 append() (abjad.tools.spannertools.StaffLineSpanner.StaffLineSpanner.StaffLineSpanner method), 1469
 append() (abjad.tools.spannertools.StaffLineSpanner.StaffLineSpanner.StaffLineSpanner method), 1476
 append() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1484
 append() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner method), 1491
 append() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1499
 append() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1524
 append() (abjad.tools.stafftools.Staff.Staff.Staff method), 1535
 append() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1545
 append() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1570
 append() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1581
 append() (abjad.tools.voicetools.Voice.Voice.Voice method), 1602
 append() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray method), 1730
 append() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1319
 append() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1327
 append() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1336
 append() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1345
 append() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1355
 append() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1366
 append() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1373
 append() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1381
 append() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1391
 append() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1398
 append() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1381

```

append_left() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.append_left(), 1391
method), 1391
append_left() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.append_left(), 1177
method), 1177
append_left() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.append_left(), 1406
method), 1406
append_left() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.append_left(), 1415
method), 1415
append_left() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.append_left(), 1423
method), 1423
append_left() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.append_left(), 1432
method), 1432
append_left() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.append_left(), 1440
method), 1440
append_left() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.append_left(), 1447
method), 1447
append_left() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.append_left(), 1455
method), 1455
append_left() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.append_left(), 1463
method), 1463
append_left() (abjad.tools.spannertools.Spanner.Spanner.Spanner.append_left(), 1470
method), 1470
append_left() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.append_left(), 1477
method), 1477
append_left() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.append_left(), 1484
method), 1484
append_left() (abjad.tools.spannertools.TextSpanner.TextSpanner.append_left(), 1492
method), 1492
append_left() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.append_left(), 1499
method), 1499
append_left() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.append_left(), 1545
method), 1545
append_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.append_row(), 1738
method), 1738
append_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.append_row(), 1730
method), 1730
append_spacer_skip_to_underfull_measure() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.append_spacer_skip_to_underfull_measure(), 1730
method), 1730
append_spacer_skip_to_underfull_measure() (in module abjad.tools.measuretools.append_spacer_skip_to_underfull_measure(), 934
method), 934
append_spacer_skips_to_underfull_measures_in_expr() (in module abjad.tools.tietools.apply_tie_spanner_to_leaf_pair(), 1551
method), 1551
append_spacer_skips_to_underfull_measures_in_expr() (in module abjad.tools.tietools.apply_tie_spanner_to_leaf_pair(), 1551
method), 1551
apply_accidental() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.apply_accidental(), 1098
method), 1098
apply_accidental() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.apply_accidental(), 1128
method), 1128
apply_accidental() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.apply_accidental(), 1130
method), 1130
apply_accidental() (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.apply_accidental(), 2099
method), 2099

```


[are_stepwise_notes\(\)](#) (in module `abjad.tools.tonalitytools`), [2105](#)
[args](#) (`abjad.tools.markuptools.MarkupCommand.MarkupCommand` attribute), [894](#)
[arguments](#) (`abjad.tools.documentationtools.Pipe.Pipe.Pipe` attribute), [1655](#)
[arithmetic_mean\(\)](#) (in module `abjad.tools.mathtools`), [1707](#)
[arpeggiate_chord\(\)](#) (in module `abjad.tools.chordtools`), [230](#)
[Articulation](#) (class in `abjad.tools.marktools`), [849](#)
[assignable_rational_to_dot_count\(\)](#) (in module `abjad.tools.durationtools`), [1669](#)
[assignable_rational_to_lilypond_duration_string\(\)](#) (in module `abjad.tools.durationtools`), [1670](#)
[attach\(\)](#) (`abjad.tools.contexttools.ClefMark.ClefMark.ClefMark` method), [341](#)
[attach\(\)](#) (`abjad.tools.contexttools.ContextMark.ContextMark.ContextMark` method), [358](#)
[attach\(\)](#) (`abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark` method), [362](#)
[attach\(\)](#) (`abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark` method), [367](#)
[attach\(\)](#) (`abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark` method), [371](#)
[attach\(\)](#) (`abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark` method), [375](#)
[attach\(\)](#) (`abjad.tools.contexttools.TempoMark.TempoMark.TempoMark` method), [380](#)
[attach\(\)](#) (`abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark` method), [390](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Accordion.Accordion.Accordion` method), [432](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute` method), [439](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone` method), [446](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone` method), [453](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone` method), [467](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice` method), [474](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet` method), [481](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute` method), [488](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon` method), [516](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone` method), [516](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone` method), [502](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice` method), [509](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet` method), [460](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Cello.Cello.Cello` method), [523](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA` method), [530](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass` method), [537](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet` method), [544](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute` method), [551](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon` method), [565](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone` method), [558](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice` method), [572](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet` method), [572](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn` method), [586](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Flute.Flute.Flute` method), [592](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn` method), [600](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel` method), [612](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Guitar.Guitar.Guitar` method), [614](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Harp.Harp.Harp` method), [621](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord` method), [621](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Marimba.Marimba.Marimba` method), [640](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice` method), [654](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Oboe.Oboe.Oboe` method), [654](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Piano.Piano.Piano` method), [661](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo` method), [668](#)
[attach\(\)](#) (`abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone` method), [675](#)

attach()	(abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.attach_lilypond_comments_to_components_in_expr() method),	682
attach()	(abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.attach_lilypond_comments_to_components_in_expr() method),	689
attach()	(abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	696
attach()	(abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	703
attach()	(abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	710
attach()	(abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	717
attach()	(abjad.tools.instrumenttools.Tuba.Tuba.Tuba.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	724
attach()	(abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	731
attach()	(abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	738
attach()	(abjad.tools.instrumenttools.Viola.Viola.Viola.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	745
attach()	(abjad.tools.instrumenttools.Violin.Violin.Violin.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	752
attach()	(abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	759
attach()	(abjad.tools.marktools.Annotation.Annotation.Annotation.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	846
attach()	(abjad.tools.marktools.Articulation.Articulation.Articulation.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	850
attach()	(abjad.tools.marktools.BarLine.BarLine.BarLine.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	854
attach()	(abjad.tools.marktools.BendAfter.BendAfter.BendAfter.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	858
attach()	(abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	861
attach()	(abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	865
attach()	(abjad.tools.marktools.Mark.Mark.Mark.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	868
attach()	(abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	871
attach()	(abjad.tools.markuptools Markup.Markup.Markup.attach_stem_tremolos_to_notes_and_chords_in_expr() method),	891
attach_annotations_to_components_in_expr()	(in module abjad.tools.marktools.attach_annotations_to_components_in_expr)	873
attach_articulations_to_notes_and_chords_in_expr()	(in module abjad.tools.marktools.attach_articulations_to_notes_and_chords_in_expr)	873
attach_lilypond_command_marks_to_components_in_expr()	(in module abjad.tools.marktools.attach_lilypond_command_marks_to_components_in_expr)	874

BeamedQuarterNoteCheck (class in abjad.tools.wellformednesstools.BeamedQuarterNoteCheck.BeamedQuarterNoteCheck), 1955

calculate_depth_centroid_of_intervals() (in module abjad.tools.timeintervaltools.calculate_depth_centroid_of_intervals), 1882

BeamSpanner (class in abjad.tools.spannertools.BeamSpanner.BeamSpanner), 1316

calculate_depth_centroid_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.calculate_depth_centroid_of_intervals_in_interval), 1882

beatspan (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer attribute), 2060

calculate_depth_density_of_intervals() (in module abjad.tools.timeintervaltools.calculate_depth_density_of_intervals), 1882

beatspan (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup attribute), 2067

beatspan_ms (abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer attribute), 2060

calculate_depth_density_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.calculate_depth_density_of_intervals_in_interval), 1883

bend_amount (abjad.tools.marktools.BendAfter.BendAfter attribute), 858

BendAfter (class in abjad.tools.marktools.BendAfter.BendAfter), 857

calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_harmonic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier), 1178

BFlatClarinet (class in abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet), 456

calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_harmonic_chromatic_interval_from_pitch_carrier_to_pitch_carrier), 1178

binomial_coefficient() (in module abjad.tools.mathtools.binomial_coefficient), 1708

calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch), 1178

BookBlock (class in abjad.tools.lilypondfiletools.BookBlock.BookBlock), 810

calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch), 1178

BookpartBlock (class in abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock), 815

calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_from_named_chromatic_pitch_to_named_chromatic_pitch), 1178

bounds (abjad.tools.timeintervaltools.TimeInterval.TimeInterval attribute), 1840

calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch), 1179

bounds (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin attribute), 1831

calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch), 1179

bounds (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin attribute), 1836

calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch), 1179

bounds (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1845

calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_diatonic_interval_from_named_chromatic_pitch_to_named_chromatic_pitch), 1179

bounds (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1859

calculate_mean_attack_of_intervals() (in module abjad.tools.timeintervaltools.calculate_mean_attack_of_intervals), 1883

BracketSpanner (class in abjad.tools.spannertools.BracketSpanner.BracketSpanner), 1324

calculate_mean_release_of_intervals() (in module abjad.tools.timeintervaltools.calculate_mean_release_of_intervals), 1883

BurnishedTimeTokenMaker (class in abjad.tools.timetokenmaker.BurnishedTimeTokenMaker.BurnishedTimeTokenMaker), 1895

calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier), 1179

C

calculate_density_of_attacks_in_interval() (in module abjad.tools.timeintervaltools.calculate_density_of_attacks_in_interval), 1882

calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier), 1179

calculate_density_of_releases_in_interval() (in module abjad.tools.timeintervaltools.calculate_density_of_releases_in_interval), 1882

calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_melodic_chromatic_interval_from_pitch_carrier_to_pitch_carrier), 1179

calculate_mean_attack_of_intervals() (in module abjad.tools.timeintervaltools.calculate_mean_attack_of_intervals), 1883

calculate_mean_release_of_intervals() (in module abjad.tools.timeintervaltools.calculate_mean_release_of_intervals), 1883

calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier() (in module abjad.tools.pitchtools.calculate_melodic_chromatic_interval_class_from_pitch_carrier_to_pitch_carrier), 1179

calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch() (in module abjad.tools.pitchtools.calculate_harmonic_counterpoint_interval_class_from_named_chromatic_pitch_to_named_chromatic_pitch), 1178

`jad.tools.pitchtools.calculate_melodic_counterpoint_interval` (abjad.tools.pitchtools.IntervalAggregateMixin.TimeInterval), 1840
 1179 attribute), 1840
`calculate_melodic_counterpoint_interval_from_named_chromatic` (abjad.tools.pitchtools.IntervalAggregateMixin.TimeInterval), 1832
 (in module ab- attribute), 1832
`jad.tools.pitchtools.calculate_melodic_counterpoint_interval_from_named_chromatic` (abjad.tools.pitchtools.IntervalAggregateMixin.TimeInterval), 1837
 1180 attribute), 1837
`calculate_melodic_diatonic_interval_class_from_named_chromatic` (abjad.tools.pitchtools.IntervalTree.TimeIntervalTree.TimeIntervalTree), 1845
 (in module ab- attribute), 1845
`jad.tools.pitchtools.calculate_melodic_diatonic_interval` (abjad.tools.pitchtools.IntervalTree.DichromaticIntervalTree.DichromaticIntervalTree), 1859
 1180 attribute), 1859
`calculate_melodic_diatonic_interval_from_named_chromatic` (abjad.tools.pitchtools.IntervalTree.DichromaticIntervalTree.DichromaticIntervalTree), 991
 (in module ab- attribute), 991
`jad.tools.pitchtools.calculate_melodic_diatonic_interval` (abjad.tools.pitchtools.IntervalTree.DichromaticIntervalTree.DichromaticIntervalTree), 1008
 1180 attribute), 1008
`calculate_min_mean_and_max_depth_of_intervals()` (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval), 1013
 (in module ab- attribute), 1013
`jad.tools.timeintervaltools.calculate_min_mean_and_max_depth_of_intervals()` (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval), 1054
 1883 attribute), 1054
`calculate_min_mean_and_max_durations_of_intervals()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval), 1075
 (in module ab- attribute), 1075
`jad.tools.timeintervaltools.calculate_min_mean_and_max_durations_of_intervals()` (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval), 1081
 1883 attribute), 1081
`calculate_sustain_centroid_of_intervals()` (in module ab- `change_augmented_tuplets_in_expr_to_diminished()`), 1587
`jad.tools.timeintervaltools.calculate_sustain_centroid_of_intervals()` (in module ab- `jad.tools.tuplettools.change_augmented_tuplets_in_expr_to_diminished()`), 1587
 1884 `jad.tools.tuplettools.change_augmented_tuplets_in_expr_to_diminished()`, 1587
`capitalize_string_start()` (in module ab- `change_defective_chord_to_note_or_rest()`), 231
`jad.tools.iotools.capitalize_string_start()`, (in module ab- `change_defective_chord_to_note_or_rest()`), 231
 1686 (in module ab- `change_defective_chord_to_note_or_rest()`), 231
`cardinality` (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass), 2074
 attribute), 2074
`cardinality` (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator), 2079
 attribute), 2079
`cell_tokens` (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn), 1735
 attribute), 1735
`cell_tokens` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1737
 attribute), 1737
`cell_tokens_by_row` (ab- `change_written_leaf_duration_and_preserve_preprolated_leaf_duration()`), 771
`jad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray` (in module ab- `change_written_leaf_duration_and_preserve_preprolated_leaf_duration()`), 771
 attribute), 1729 `Check` (class in abjad.tools.wellformednesstools.Check.Check), 1958
`cell_widths` (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn), 1735
 attribute), 1735
`cell_widths` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1737
 attribute), 1737
`cell_widths_by_row` (ab- `check()` (abjad.tools.wellformednesstools.Check.Check.Check), 1953
`jad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray` (in module ab- `check()` (abjad.tools.wellformednesstools.Check.Check.Check), 1953
 attribute), 1729 `check()` (abjad.tools.wellformednesstools.DiscontiguousSpannerCheck.DiscontiguousSpannerCheck), 1958
 method), 1958
`Cello` (class in abjad.tools.instrumenttools.Cello.Cello), `check()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck), 1960
 519 `check()` (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck), 1960
 method), 1960
`cells` (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray), 1729
 attribute), 1729
`cells` (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn), 1735
 attribute), 1735
`cells` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1737
 attribute), 1737
`cells` (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow), 1737
 attribute), 1737

[illegible]

clear() (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner.clef_and_staff_position_number_to_named_chromatic_pitch method), 1327

clear() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner.ClefMark.ClefMark.ClefMark attribute), 1336

clear() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner in abjad.tools.contexttools.ClefMark.ClefMark), 1346

clear() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner in abjad.tools.contexttools.ClefMark.ClefMark), 1356

clear() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.ClefMarkInventory (class in abjad.tools.spannertools.ClefMarkInventory)), 1366

clear() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner in module abjad.tools.timeintervaltools.clip_interval_durations_to_range), 1374

clear() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner close() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1381

clear() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner Cluster (class in abjad.tools.containertools.Cluster.Cluster), 1391

clear() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner code_root (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler attribute), 1399

clear() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner code_root (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler attribute), 1406

clear() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner code_root (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler attribute), 1415

clear() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner code_root (abjad.tools.documentationtools.ModuleCrawler.ModuleCrawler.ModuleCrawler attribute), 1423

clear() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner code_tools_path (abjad.tools.documentationtools.AbjadAPIGenerator.AbjadAPIGenerator.AbjadAPIGenerator attribute), 1432

clear() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner color_chord_note_heads_by_pitch_class_color_map() (in module abjad.tools.chordtools.color_chord_note_heads_by_pitch_class_color_map), 1440

clear() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner color_contents_of_container() (in module abjad.tools.containertools.color_contents_of_container), 1448

clear() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner color_note_head_by_numbered_chromatic_pitch_class_color_map() (in module abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1456

clear() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner color_note_head_by_numbered_chromatic_pitch_class_color_map() (in module abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1463

clear() (abjad.tools.spannertools.Spanner.Spanner.Spanner color_leaf() (in module abjad.tools.leaftools.color_leaf), 1470

clear() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner color_leaves_in_expr() (in module abjad.tools.leaftools.color_leaves_in_expr), 1477

clear() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner color_measure() (in module abjad.tools.measuretools.color_measure), 1484

clear() (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner color_nonbinary_measures_in_expr() (in module abjad.tools.measuretools.color_nonbinary_measures_in_expr), 1492

clear() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner color_note_head_by_numbered_chromatic_pitch_class_color_map() (in module abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1500

clear() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner color_note_head_by_numbered_chromatic_pitch_class_color_map() (in module abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1546

clear() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval in module abjad.tools.timeintervaltools.TimeIntervalTreeDictionary (class in abjad.tools.timeintervaltools.TimeIntervalTreeDictionary)), 1841

clear() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1861

clear_terminal() (in module abjad.tools.iotools.clear_terminal), 1687

clef_and_staff_position_number_to_named_chromatic_pitch() (in module abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn attribute), 1732

clef_and_staff_position_number_to_named_chromatic_pitch() (in module abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1732

columns (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1729

columns (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix attribute), 1748

columns (abjad.tools.sequencetools.Matrix.Matrix.Matrix attribute), 1764

combine_markup_commands() (in module abjad.tools.markuptools.combine_markup_commands), 900

command (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand attribute), 894

command_name (abjad.tools.marktools.BarLine.BarLine.BarLine attribute), 854

command_name (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark attribute), 861

comment_measures_in_container_with_measure_numbers() (in module abjad.tools.measuretools.comment_measures_in_container_with_measure_numbers), 942

communicate() (abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1656

ComplexBeamSpanner (class in abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner), 1332

Component (class in abjad.tools.componenttools.Component.Component), 236

component_to_parentage_signature() (in module abjad.tools.componenttools.component_to_parentage_signature), 242

component_to_pitch_and_rhythm_skeleton() (in module abjad.tools.componenttools.component_to_pitch_and_rhythm_skeleton), 243

component_to_score_depth() (in module abjad.tools.componenttools.component_to_score_depth), 244

component_to_score_index() (in module abjad.tools.componenttools.component_to_score_index), 244

component_to_score_root() (in module abjad.tools.componenttools.component_to_score_root), 245

component_to_thread_signature() (in module abjad.tools.threadtools.component_to_thread_signature), 1826

component_to_tuplet_depth() (in module abjad.tools.componenttools.component_to_tuplet_depth), 245

components (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 1317

components (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 1325

components (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner attribute), 1333

components (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute), 1342

components (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute), 1352

components (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 1362

components (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner attribute), 1379

components (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner attribute), 1379

components (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1388

components (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1396

components (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1404

components (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 1412

components (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner attribute), 1421

components (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner attribute), 1430

components (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner attribute), 1437

components (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner attribute), 1445

components (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner attribute), 1453

components (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner attribute), 1461

components (abjad.tools.spannertools.Spanner.Spanner.Spanner attribute), 1468

components (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner attribute), 1474

components (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner attribute), 1482

components (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner attribute), 1490

components (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute), 1497

components (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute), 1543

components (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

composite_tree (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1859

compute_depth_of_intervals() (in module abjad.tools.timeintervaltools.compute_depth_of_intervals), 1884

compute_depth_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.compute_depth_of_intervals_in_interval), 1884

compute_logical_and_of_intervals() (in module abjad.tools.timeintervaltools.compute_logical_and_of_intervals), 1885

compute_logical_and_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.compute_logical_and_of_intervals_in_interval), 1885

compute_logical_not_of_intervals() (in module abjad.tools.timeintervaltools.compute_logical_not_of_intervals), 1885

compute_logical_not_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.compute_logical_not_of_intervals_in_interval), 1885

compute_logical_or_of_intervals() (in module abjad.tools.timeintervaltools.compute_logical_or_of_intervals), 1885

compute_logical_or_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.compute_logical_or_of_intervals_in_interval), 1885

compute_logical_xor_of_intervals() (in module abjad.tools.timeintervaltools.compute_logical_xor_of_intervals), 1886

compute_logical_xor_of_intervals_in_interval() (in module abjad.tools.timeintervaltools.compute_logical_xor_of_intervals_in_interval), 1886

concatenate_pitch_arrays() (in module abjad.tools.pitcharraytools.concatenate_pitch_arrays), 1740

concatenate_trees() (in module abjad.tools.timeintervaltools.concatenate_trees), 1886

conjugate() (abjad.tools.durationtools.Duration.Duration.Duration method), 1660

conjugate() (abjad.tools.durationtools.Offset.Offset.Offset method), 1664

constraints (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver.FixedLengthStreamSolver), 1996

constraints (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver.VariableLengthStreamSolver), 2014

Container (class in abjad.tools.containertools.Container.Container), 302

contents_duration (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 294

contents_duration (abjad.tools.containertools.Container.Container.Container attribute), 302

contents_duration (abjad.tools.contexttools.Context.Context.Context attribute), 348

contents_duration (abjad.tools.gracetools.Grace.Grace.Grace attribute), 416

contents_duration (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics attribute), 2017

contents_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2043

contents_duration (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 905

contents_duration (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 916

contents_duration (abjad.tools.measuretools.Measure.Measure.Measure attribute), 926

contents_duration (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1249

contents_duration (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1270

contents_duration (abjad.tools.scoretools.Score.Score.Score attribute), 1280

contents_duration (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1291

contents_duration (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1521

contents_duration (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1531

contents_duration (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1564

contents_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1576

contents_duration (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1598

contents_string (abjad.tools.marktools.LilyPondComment.LilyPondComment attribute), 865

contents_string (abjad.tools.markuptools.Markup.Markup.Markup attribute), 890

Context (class in abjad.tools.contexttools.Context.Context), 348

context_blocks (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock attribute), 825

context_name (abjad.tools.contexttools.Context.Context.Context attribute), 350

context_name (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2043

context_name (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1251

context_name (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1272

context_name (abjad.tools.scoretools.Score.Score.Score attribute), 1282

context_name (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1293

context_name (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1523

context_name (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1533

context_name (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1601

ContextBlock (class in abjad.tools.lilypondfiletools.ContextBlock.ContextBlock), 819

ContextMark (class in abjad.tools.contexttools.ContextMark.ContextMark), 357

contexts (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock attribute), 825

Contrabass (class in abjad.tools.instrumenttools.Contrabass.Contrabass), 533

ContrabassClarinet (class in abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet), 540

ContrabassFlute (class in abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute), 547

Contrabassoon (class in abjad.tools.instrumenttools.Contrabassoon.Contrabassoon), 561

ContrabassSaxophone (class in abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone), 554

ContraltoVoice (class in abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice), 568

copy() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary method), 1624

copy() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph method), 1649

copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 997

copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1005

copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1018

copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1026

copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1035

copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1040

copy() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1050

copy() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1063

copy() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1071

copy() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1090

copy() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1105

copy() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1112

copy() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1117

copy() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1139

copy() (abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method), 1144

copy() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree method), 2063

copy() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup method), 2068

copy() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1861

copy() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 2075

copy_and_partition_governed_component_subtree_by_leaf_counts() (in module abjad.tools.componenttools.copy_and_partition_governed_component_subtree_by_leaf_counts), 245

copy_components_and_covered_spanners() (in module abjad.tools.componenttools.copy_components_and_covered_spanners), 245

copy_components_and_fracture_crossing_spanners() (in module abjad.tools.componenttools.copy_components_and_fracture_crossing_spanners), 248

copy_of_first_component() (in module abjad.tools.componenttools.copy_of_first_component), 249

copy_components_and_immediate_parents() (in module abjad.tools.componenttools.copy_components_and_immediate_parents), 249

copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method), 997

copy() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1005

copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1018

copy() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1026

copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1035

copy() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method), 1040

copy() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet.InversionEquivalentDiatonicIntervalClassSet method), 1050

copy() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method), 1063

copy() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1071

copy() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1090

copy() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1105

count() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory (in module abjad.tools.sequencetools.count_length_two_runs_in_sequence), method), 383

count() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory (in module abjad.tools.sequencetools.count_length_two_runs_in_sequence), method), 1628

count() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory (in module abjad.tools.sequencetools.count_length_two_runs_in_sequence), method), 631

count() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock (class in abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner), method), 811

count() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock (class in abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner), method), 815

count() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile (class in abjad.tools.mathtools.cumulative_products), method), 828

count() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock (class in abjad.tools.mathtools.cumulative_products), method), 840

count() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory (in module abjad.tools.mathtools.cumulative_signed_weights), method), 896

count() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1001

count() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1023

count() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1032

count() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1047

count() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1060

count() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1068

count() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1087

count() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1101

count() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1109

count() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1135

count() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1154

count() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1160

count() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1168

count() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1173

count() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1264

count() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1744

count() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple (in module abjad.tools.mathtools.cumulative_signed_weights), method), 1761

count() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator.ChordQualityIndicator (in module abjad.tools.mathtools.cumulative_signed_weights), method), 2080

count() (abjad.tools.tonalitytools.Scale.Scale.Scale (in module abjad.tools.mathtools.cumulative_signed_weights), method), 2095

default_instrument_name jad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark. attribute), 365	(ab- default_instrument_name jad.tools.contexttools.ContrabassFlute.ContrabassFlute.Contra attribute), 548	(ab-
default_instrument_name jad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 429	default_instrument_name jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contraba attribute), 562	(ab-
default_instrument_name jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 436	default_instrument_name jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxop attribute), 555	(ab-
default_instrument_name jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 443	default_instrument_name jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.Contralto attribute), 569	(ab-
default_instrument_name jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 450	default_instrument_name jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 576	(ab-
default_instrument_name jad.tools.instrumenttools.BartoneSaxophone.BartoneSaxophone.BartoneSaxophone attribute), 464	default_instrument_name jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 583	(ab-
default_instrument_name jad.tools.instrumenttools.BartoneVoice.BartoneVoice.BartoneVoice attribute), 471	default_instrument_name jad.tools.instrumenttools.Flute.Flute.Flute attribute), 590	(ab-
default_instrument_name jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 478	default_instrument_name jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 596	(ab-
default_instrument_name jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 485	default_instrument_name jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 604	(ab-
default_instrument_name jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 513	default_instrument_name jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 611	(ab-
default_instrument_name jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 492	default_instrument_name jad.tools.instrumenttools.Harp.Harp.Harp attribute), 618	(ab-
default_instrument_name jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 499	default_instrument_name jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 625	(ab-
default_instrument_name jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 506	default_instrument_name jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 637	(ab-
default_instrument_name jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 457	default_instrument_name jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoic attribute), 644	(ab-
default_instrument_name jad.tools.instrumenttools.Cello.Cello.Cello attribute), 520	default_instrument_name jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 651	(ab-
default_instrument_name jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 527	default_instrument_name jad.tools.instrumenttools.Piano.Piano.Piano attribute), 658	(ab-
default_instrument_name jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 534	default_instrument_name jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 665	(ab-
default_instrument_name jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 541	default_instrument_name jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxoph attribute), 672	(ab-

default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.attribute), 679		jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone.attribute), 443	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.attribute), 686		jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.attribute), 450	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.attribute), 693		jad.tools.instrumenttools.BartoneSaxophone.BartoneSaxophone.BartoneSaxophone.attribute), 464	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.attribute), 700		jad.tools.instrumenttools.BartoneVoice.BartoneVoice.BartoneVoice.attribute), 471	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.attribute), 707		jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.attribute), 478	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.attribute), 714		jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.attribute), 485	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Tuba.Tuba.Tuba.attribute), 721		jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon.attribute), 513	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.attribute), 728		jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone.attribute), 492	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.attribute), 735		jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone.attribute), 499	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Viola.Viola.Viola.attribute), 742		jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice.attribute), 506	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Violin.Violin.Violin.attribute), 749		jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet.attribute), 457	
default_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.attribute), 756		jad.tools.instrumenttools.Cello.Cello.Cello.attribute), 520	
default_instrument_name_to_instrument_class()	default_short_instrument_name	(ab-	
(in module ab-	jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA.attribute), 507		
jad.tools.instrumenttools.default_instrument_name_to_instrument_class()	default_short_instrument_name	(ab-	
761			
default_language (abjad.tools.lilypondparsertools.LilyPondParser.LilyPondParser.LilyPondParser.attribute), 1705		jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass.attribute), 534	
default_paper_size	(ab-	default_short_instrument_name	(ab-
jad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile.attribute), 827		jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.C attribute), 541	
default_short_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.attribute), 365		jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.Contra attribute), 548	
default_short_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.Accordion.Accordion.Accordion.attribute), 429		jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contraba attribute), 562	
default_short_instrument_name	(ab-	default_short_instrument_name	(ab-
jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute.attribute), 436		jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxop attribute), 555	

denominator (abjad.tools.measuretools.AnonymousMeasuredAnchors (abjad.tools.measuretools.BassSaxophone.BassSaxophone.BassSaxophone), 908
 attribute), 908
 method), 495
 denominator (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure (abjad.tools.measuretools.BassTrombone.BassTrombone.BassTrombone), 918
 attribute), 918
 method), 502
 depth (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice), 1729
 attribute), 1729
 method), 509
 depth (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet), 1735
 attribute), 1735
 method), 460
 depth (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow (abjad.tools.instrumenttools.Cello.Cello.Cello), 1737
 attribute), 1737
 method), 523
 depth (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA), 1751
 attribute), 1751
 method), 530
 depth (abjad.tools.sequencetools.Tree.Tree.Tree (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 1767
 attribute), 1767
 method), 537
 destroy_all_spanners_attached_to_component() detach() (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet), 544
 (in module abjad.tools.instrumenttools), 544
 method), 544
 jadt.tools.spannertools.destroy_all_spanners_attached_to_component() detach() (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 1505
 method), 551
 detach() (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 341
 method), 341
 detach() (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 358
 method), 358
 detach() (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice), 362
 method), 362
 detach() (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet), 367
 method), 367
 detach() (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn), 371
 method), 371
 detach() (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark (abjad.tools.instrumenttools.Flute.Flute.Flute), 375
 method), 375
 detach() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn), 380
 method), 380
 detach() (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel), 390
 method), 390
 detach() (abjad.tools.gracetools.Grace.Grace.Grace (abjad.tools.instrumenttools.Guitar.Guitar.Guitar), 419
 method), 419
 detach() (abjad.tools.instrumenttools.Accordion.Accordion.Accordion (abjad.tools.instrumenttools.Harp.Harp.Harp), 432
 method), 432
 detach() (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord), 439
 method), 439
 detach() (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone (abjad.tools.instrumenttools.Marimba.Marimba.Marimba), 446
 method), 446
 detach() (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice), 453
 method), 453
 detach() (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone (abjad.tools.instrumenttools.Oboe.Oboe.Oboe), 467
 method), 467
 detach() (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice (abjad.tools.instrumenttools.Piano.Piano.Piano), 474
 method), 474
 detach() (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo), 481
 method), 481
 detach() (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone), 488
 method), 488
 detach() (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone), 516
 method), 516

detach() (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.detach_context_marks_attached_to_component()
method), 689 392

detach() (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.detach_to_component()
method), 696 (in module ab-

detach() (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.detach_dynamic_marks_attached_to_component()
method), 703 392

detach() (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.detach_voice_containers_attached_to_leaf()
method), 710 (in module ab-

detach() (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.detach_grace_containers_attached_to_leaf(),
method), 717 426

detach() (abjad.tools.instrumenttools.Tuba.Tuba.Tuba.detach_instrument_marks_attached_to_component()
method), 724 (in module ab-

detach() (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.detach_instrument_marks_attached_to_component()
method), 731 393

detach() (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.detach_key_signature_marks_attached_to_component()
method), 738 (in module ab-

detach() (abjad.tools.instrumenttools.Viola.Viola.Viola.detach_key_signature_marks_attached_to_component(),
method), 745 393

detach() (abjad.tools.instrumenttools.Violin.Violin.Violin.detach_lilypond_command_marks_attached_to_component()
method), 752 (in module ab-

detach() (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.detach_lilypond_command_marks_attached_to_component(),
method), 759 876

detach() (abjad.tools.marktools.Annotation.Annotation.Annotation.detach_lilypond_comments_attached_to_component()
method), 847 (in module ab-

detach() (abjad.tools.marktools.Articulation.Articulation.Articulation.detach_lilypond_comments_attached_to_component(),
method), 850 877

detach() (abjad.tools.marktools.BarLine.BarLine.BarLine.detach_marks_attached_to_component() (in module ab-
method), 855 jad.tools.marktools.detach_marks_attached_to_component()),

detach() (abjad.tools.marktools.BendAfter.BendAfter.BendAfter.detach_noncontext_marks_attached_to_component()
method), 858 877

detach() (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark.detach_noncontext_marks_attached_to_component(),
method), 862 877

detach() (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment.detach_staff_change_marks_attached_to_component(),
method), 865 877

detach() (abjad.tools.marktools.Mark.Mark.Mark.detach_staff_change_marks_attached_to_component(),
method), 868 877

detach() (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo.detach_stem_tremolos_attached_to_component(),
method), 871 877

detach() (abjad.tools.markuptools.Markup.Markup.Markup.detach_stem_tremolos_attached_to_component(),
method), 892 878

detach_annotations_attached_to_component()
(in module ab- detach_tempo_marks_attached_to_component()
jad.tools.marktools.detach_annotations_attached_to_component(), module ab-
875 jad.tools.contexttools.detach_tempo_marks_attached_to_component()

detach_articulations_attached_to_component()
(in module ab- detach_time_signature_marks_attached_to_component()
jad.tools.marktools.detach_articulations_attached_to_component(), module ab-
875 jad.tools.contexttools.detach_time_signature_marks_attached_to_component()

detach_clef_marks_attached_to_component()
(in module ab- deviation_in_cents (ab-
jad.tools.contexttools.detach_clef_marks_attached_to_component(), module ab-
391 jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.detach_clef_marks_attached_to_component(), 1095

detach_context_marks_attached_to_component()
(in module ab- diatonic_interval_class (ab-
jad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.detach_context_marks_attached_to_component(), 1095

attribute), 1013

diatonic_interval_class (ab- 1187
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval.diatonic_interval_class_number()
attribute), 1081 (in module ab-
diatonic_interval_class_segment (ab- jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_class
jad.tools.tonalitytools.Scale.Scale.Scale at- 1188
tribute), 2094 diatonic_pitch_name_to_chromatic_pitch_name()
diatonic_interval_class_segment_to_chord_quality_string() (in module ab-
(in module ab- jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_name
jad.tools.tonalitytools.diatonic_interval_class_segment_to_chord_quality_string),
2106 diatonic_pitch_name_to_chromatic_pitch_number()
diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_interval() (in module ab-
(in module ab- jad.tools.pitchtools.diatonic_pitch_name_to_chromatic_pitch_number
jad.tools.pitchtools.diatonic_interval_number_and_chromatic_interval_number_to_melodic_diatonic_interval),
1186 diatonic_pitch_name_to_diatonic_pitch_class_name()
diatonic_pitch_class_name (ab- (in module ab-
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.diatonic_pitch_name_to_diatonic_pitch_class
attribute), 1095 1188
diatonic_pitch_class_name (ab- diatonic_pitch_name_to_diatonic_pitch_class_number()
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.in module ab-
attribute), 1122 jad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_class
diatonic_pitch_class_name_to_chromatic_pitch_class_number() 1188
(in module ab- diatonic_pitch_name_to_diatonic_pitch_number()
jad.tools.pitchtools.diatonic_pitch_class_name_to_chromatic_pitch_class_number() (in module ab-
1187 jad.tools.pitchtools.diatonic_pitch_name_to_diatonic_pitch_number
diatonic_pitch_class_name_to_diatonic_pitch_class_number() 1189
(in module ab- diatonic_pitch_number (ab-
jad.tools.pitchtools.diatonic_pitch_class_name_to_diatonic_pitch_number) jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch
1187 attribute), 1095
diatonic_pitch_class_number (ab- diatonic_pitch_number (ab-
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.diatonic_pitch_number diatonic_pitch_number
attribute), 1095 attribute), 1122
diatonic_pitch_class_number (ab- diatonic_pitch_number (ab-
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.in module ab-
attribute), 1122 attribute), 1128
diatonic_pitch_class_number (ab- diatonic_pitch_number (ab-
jad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.in module ab-
attribute), 1128 attribute), 1148
diatonic_pitch_class_number_to_chromatic_pitch_class_number() diatonic_pitch_number_to_chromatic_pitch_number()
(in module ab- (in module ab-
jad.tools.pitchtools.diatonic_pitch_class_number_to_chromatic_pitch_class_number) jad.tools.pitchtools.diatonic_pitch_number_to_chromatic_pitch_r
1187 1189
diatonic_pitch_class_number_to_diatonic_pitch_class_name() diatonic_pitch_number_to_diatonic_pitch_class_name()
(in module ab- (in module ab-
jad.tools.pitchtools.diatonic_pitch_class_number_to_diatonic_pitch_class_name) jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_cla
1187 1189
diatonic_pitch_name (ab- diatonic_pitch_number_to_diatonic_pitch_class_number()
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.in module ab-
attribute), 1095 jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_class
diatonic_pitch_name (ab- 1189
jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.diatonic_pitch_name() (in module ab-
attribute), 1122
diatonic_pitch_name_to_chromatic_pitch_class_name() jad.tools.pitchtools.diatonic_pitch_number_to_diatonic_pitch_name
(in module ab- 1189

difference()	(abjad.tools.pitchtools.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet.method), 1005
difference()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.method), 1018
difference()	(abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.method), 1026
difference()	(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet.method), 1035
difference()	(abjad.tools.pitchtools.MelodicChromaticIntervalClassSet.MelodicChromaticIntervalClassSet.method), 1071
difference()	(abjad.tools.pitchtools.MelodicDiatonicIntervalClassSet.MelodicDiatonicIntervalClassSet.method), 1090
difference()	(abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.method), 1105
difference()	(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.method), 1113
difference()	(abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.method), 1139
difference()	(abjad.tools.tonalitytools.ChordClass.ChordClass.method), 2075
difference_series()	(in module abjad.tools.mathtools.difference_series), 1710
Digraph	(class in abjad.tools.datastructuretools.Digraph.Digraph), 1621
dimensions	(abjad.tools.pitcharraytools.PitchArray.PitchArray.attribute), 1729
dimensions	(abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.attribute), 1735
dimensions	(abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.attribute), 1737
direction	(abjad.tools.marktools.Articulation.Articulation.attribute), 850
direction	(abjad.tools.markuptools.Markup.Markup.attribute), 891
direction	(abjad.tools.spannertools.BeamSpanner.BeamSpanner.attribute), 1318
direction	(abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.attribute), 1334
direction	(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.attribute), 1344
direction	(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.attribute), 1354
direction	(abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.attribute), 1363
direction	(abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.attribute), 1389
direction	(abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.attribute), 1413
direction	(abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.attribute), 1431
direction	(abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.attribute), 1447

[divide_sequence_elements_by_greatest_common_divisor\(\)](#) [duration](#) ([abjad.tools.timeintervaltools.TimeIntervalTreeDictionary](#).[TimeIntervalTreeDictionary](#).[TimeIntervalTreeDictionary](#) attribute), 1859
[\(in module abjad.tools.sequencetools.divide_sequence_elements_by_greatest_common_divisor\)](#), in [abjad.tools.durationtools.Duration](#).[Duration](#).[Duration](#) attribute), 1779
[divisors\(\)](#) ([in module abjad.tools.mathtools.divisors](#)), 1659
[docs_api_index_path](#) ([abjad.tools.documentationtools.AbjadAPIGenerator](#).[AbjadAPIGenerator](#).[AbjadAPIGenerator](#) attribute), 1635
[docs_root](#) ([abjad.tools.documentationtools.APICrawler](#).[APICrawler](#).[APICrawler](#) attribute), 1633
[docs_tools_path](#) ([abjad.tools.documentationtools.AbjadAPIGenerator](#).[AbjadAPIGenerator](#).[AbjadAPIGenerator](#) attribute), 1635
[Documenter](#) ([class in abjad.tools.documentationtools.Documenter](#).[Documenter](#) attribute), 294
[domain](#) ([abjad.tools.constrainttools.FixedLengthStreamSolver](#).[FixedLengthStreamSolver](#).[FixedLengthStreamSolver](#) attribute), 1996
[domain](#) ([abjad.tools.constrainttools.VariableLengthStreamSolver](#).[VariableLengthStreamSolver](#).[VariableLengthStreamSolver](#) attribute), 2014
[Domain](#) ([class in abjad.tools.constrainttools.Domain](#).[Domain](#) attribute), 1993
[dominant](#) ([abjad.tools.tonalitytools.Scale](#).[Scale](#).[Scale](#) attribute), 2094
[DoublingIndicator](#) ([class in abjad.tools.tonalitytools.DoublingIndicator](#).[DoublingIndicator](#) attribute), 2083
[doublings](#) ([abjad.tools.tonalitytools.DoublingIndicator](#).[DoublingIndicator](#).[DoublingIndicator](#) attribute), 2083
[duplicate_pitch_classes](#) ([abjad.tools.pitchtools.NamedChromaticPitchSet](#).[NamedChromaticPitchSet](#).[NamedChromaticPitchSet](#) attribute), 1112
[DuplicateIdCheck](#) ([class in abjad.tools.wellformednesstools.DuplicateIdCheck](#).[DuplicateIdCheck](#) attribute), 1960
[DulatedComplexBeamSpanner](#) ([class in abjad.tools.spannertools.DulatedComplexBeamSpanner](#).[DulatedComplexBeamSpanner](#) attribute), 1361
[duration](#) ([abjad.tools.contexttools.TempoMark](#).[TempoMark](#).[TempoMark](#) attribute), 379
[duration](#) ([abjad.tools.contexttools.TimeSignatureMark](#).[TimeSignatureMark](#).[TimeSignatureMark](#) attribute), 387
[duration](#) ([abjad.tools.schemetools.SchemeMoment](#).[SchemeMoment](#).[SchemeMoment](#) attribute), 1233
[duration](#) ([abjad.tools.timeintervaltools.TimeInterval](#).[TimeInterval](#).[TimeInterval](#) attribute), 1840
[duration](#) ([abjad.tools.timeintervaltools.TimeIntervalAggregateMixin](#).[TimeIntervalAggregateMixin](#).[TimeIntervalAggregateMixin](#) attribute), 1832
[duration](#) ([abjad.tools.timeintervaltools.TimeIntervalMixin](#).[TimeIntervalMixin](#).[TimeIntervalMixin](#) attribute), 1837
[duration](#) ([abjad.tools.timeintervaltools.TimeIntervalTree](#).[TimeIntervalTree](#).[TimeIntervalTree](#) attribute), 1845

Index	2219
--------------	-------------

[duration_in_seconds](#) (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1576
[duration_in_seconds](#) (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1599
[duration_multiplier](#) (abjad.tools.chordtools.Chord.Chord.Chord attribute), 226
[duration_multiplier](#) (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 769
[duration_multiplier](#) (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender attribute), 2026
[duration_multiplier](#) (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2031
[duration_multiplier](#) (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace attribute), 2035
[duration_multiplier](#) (abjad.tools.lyricstools.LyricText.LyricText.LyricText attribute), 2040
[duration_multiplier](#) (abjad.tools.note tools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 968
[duration_multiplier](#) (abjad.tools.note tools.Note.Note.Note attribute), 973
[duration_multiplier](#) (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1215
[duration_multiplier](#) (abjad.tools.resttools.Rest.Rest.Rest attribute), 1219
[duration_multiplier](#) (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1310
[duration_pair_to_prolation_string\(\)](#) (in module abjad.tools.durationtools.duration_pair_to_prolation_string), 1670
[duration_token_to_big_endian_list_of_assignable_duration_pairs\(\)](#) (in module abjad.tools.durationtools.duration_token_to_big_endian_list_of_assignable_duration_pairs), 1670
[duration_token_to_duration_pair\(\)](#) (in module abjad.tools.durationtools.duration_token_to_duration_pair), 1670
[duration_token_to_rational\(\)](#) (in module abjad.tools.durationtools.duration_token_to_rational), 1671
[duration_tokens_to_duration_pairs\(\)](#) (in module abjad.tools.durationtools.duration_tokens_to_duration_pairs), 1671
[duration_tokens_to_duration_pairs_with_least_common_denominator\(\)](#) (in module abjad.tools.durationtools.duration_tokens_to_duration_pairs_with_least_common_denominator), 1671
[duration_tokens_to_least_common_denominator\(\)](#) (in module abjad.tools.durationtools.duration_tokens_to_least_common_denominator), 1672
[duration_tokens_to_rationals\(\)](#) (in module abjad.tools.durationtools.duration_tokens_to_rationals), 1672
[durations](#) (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 1364
[dynamic_name](#) (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark attribute), 362
[DynamicMark](#) (class in abjad.tools.contexttools.DynamicMark.DynamicMark), 360
[DynamicMeasure](#) (class in abjad.tools.measuretools.DynamicMeasure.DynamicMeasure), 915
[DynamicTextSpanner](#) (class in abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner), 1371

E

[edges](#) (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1621
[effective_context](#) (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark attribute), 340
[effective_context](#) (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark attribute), 361
[effective_context](#) (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark attribute), 361
[effective_context](#) (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark attribute), 365
[effective_context](#) (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 369
[effective_context](#) (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark attribute), 374
[effective_context](#) (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark attribute), 378
[effective_context](#) (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), 388

`effective_context` (`abjad.tools.instrumenttools.Accordion.Accordion`), 607
`effective_context` (`abjad.tools.instrumenttools.Harp.Harp.Harp`), 618
`effective_context` (`abjad.tools.instrumenttools.AltoFlute.AltoFlute`), 625
`effective_context` (`abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord`), 625
`effective_context` (`abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone`), 637
`effective_context` (`abjad.tools.instrumenttools.Marimba.Marimba.Marimba`), 637
`effective_context` (`abjad.tools.instrumenttools.AltoTrombone.AltoTrombone`), 644
`effective_context` (`abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice`), 644
`effective_context` (`abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone`), 651
`effective_context` (`abjad.tools.instrumenttools.Oboe.Oboe.Oboe`), 651
`effective_context` (`abjad.tools.instrumenttools.BariotoneSaxophone.BariotoneSaxophone`), 651
`effective_context` (`abjad.tools.instrumenttools.Piano.Piano.Piano`), 658
`effective_context` (`abjad.tools.instrumenttools.BassClarinet.BassClarinet`), 665
`effective_context` (`abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo`), 665
`effective_context` (`abjad.tools.instrumenttools.BassFlute.BassFlute`), 672
`effective_context` (`abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone`), 672
`effective_context` (`abjad.tools.instrumenttools.Bassoon.Bassoon`), 679
`effective_context` (`abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone`), 679
`effective_context` (`abjad.tools.instrumenttools.BassSaxophone.BassSaxophone`), 686
`effective_context` (`abjad.tools.instrumenttools.SopranoVoice.SopranoVoice`), 686
`effective_context` (`abjad.tools.instrumenttools.BassTrombone.BassTrombone`), 693
`effective_context` (`abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone`), 693
`effective_context` (`abjad.tools.instrumenttools.BassVoice.BassVoice`), 700
`effective_context` (`abjad.tools.instrumenttools.TenorTrombone.TenorTrombone`), 700
`effective_context` (`abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet`), 707
`effective_context` (`abjad.tools.instrumenttools.TenorVoice.TenorVoice`), 707
`effective_context` (`abjad.tools.instrumenttools.Cello.Cello.Cello`), 714
`effective_context` (`abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet`), 714
`effective_context` (`abjad.tools.instrumenttools.ClarineteInA.ClarineteInA`), 721
`effective_context` (`abjad.tools.instrumenttools.Tuba.Tuba.Tuba`), 721
`effective_context` (`abjad.tools.instrumenttools.Contrabass.Contrabass`), 728
`effective_context` (`abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion`), 728
`effective_context` (`abjad.tools.instrumenttools.ContrabassClarineteInB.ContrabassClarineteInB`), 735
`effective_context` (`abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone`), 735
`effective_context` (`abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute`), 742
`effective_context` (`abjad.tools.instrumenttools.Viola.Viola.Viola`), 742
`effective_context` (`abjad.tools.instrumenttools.Contrabassoon.Contrabassoon`), 749
`effective_context` (`abjad.tools.instrumenttools.Violin.Violin.Violin`), 749
`effective_context` (`abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone`), 756
`effective_context` (`abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone`), 756
`effective_context` (`abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice`), 769
`effective_context` (`abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet`), 776
`effective_context` (`abjad.tools.instrumenttools.EjectContentsOfContainer()`), 312
`effective_context` (`abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow`), 1738
`effective_context` (`abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck`), 582
`effective_context` (`abjad.tools.Glockenspiel.Glockenspiel`), 604
`effective_context` (`abjad.tools.EnglishHorn.EnglishHorn`), 582
`effective_context` (`abjad.tools.Guitar.Guitar`), 611

[engraver_consists \(abjad.tools.contexttools.Context.Context.Context attribute\), 348](#)
[engraver_consists \(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute\), 2043](#)
[engraver_consists \(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute\), 1249](#)
[engraver_consists \(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute\), 1270](#)
[engraver_consists \(abjad.tools.scoretools.Score.Score.Score \(in module abjad.tools.pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class\(\)\), 1280](#)
[engraver_consists \(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute\), 1291](#)
[engraver_consists \(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff module abjad.tools.leaftools.expr_has_leaf_with_dotted_written_duration\(\)\), 1521](#)
[engraver_consists \(abjad.tools.stafftools.Staff.Staff.Staff attribute\), 1531](#)
[engraver_consists \(abjad.tools.voicetools.Voice.Voice.Voice \(in module abjad.tools.pitchtools.expr_to_melodic_chromatic_interval_segment\(\)\), 1599](#)
[engraver_removals \(abjad.tools.contexttools.Context.Context.Context attribute\), 348](#)
[engraver_removals \(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute\), 2044](#)
[engraver_removals \(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute\), 1249](#)
[engraver_removals \(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute\), 1270](#)
[engraver_removals \(abjad.tools.scoretools.Score.Score.Score attribute\), 1280](#)
[engraver_removals \(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute\), 1291](#)
[engraver_removals \(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute\), 1521](#)
[engraver_removals \(abjad.tools.stafftools.Staff.Staff.Staff attribute\), 1531](#)
[engraver_removals \(abjad.tools.voicetools.Voice.Voice.Voice attribute\), 1599](#)
[executable \(abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute\), 1655](#)
[explode_intervals_compactly\(\) \(in module abjad.tools.timeintervaltools.explode_intervals_compactly\(\)\), 1886](#)
[explode_intervals_into_n_trees_heuristically\(\) \(in module abjad.tools.timeintervaltools.explode_intervals_into_n_trees_heuristically\(\)\), 1886](#)
[explode_intervals_uncompactly\(\) \(in module abjad.tools.timeintervaltools.explode_intervals_uncompactly\(\)\), 1886](#)
[expr_has_duplicate_named_chromatic_pitch\(\) \(in module abjad.tools.pitchtools.expr_has_duplicate_named_chromatic_pitch\(\)\), 1886](#)
[expr_has_duplicate_numbered_chromatic_pitch_class\(\) \(in module abjad.tools.pitchtools.expr_has_duplicate_numbered_chromatic_pitch_class\(\)\), 1886](#)
[expr_has_leaf_with_dotted_written_duration\(\) \(in module abjad.tools.leaftools.expr_has_leaf_with_dotted_written_duration\(\)\), 1886](#)
[expr_to_melodic_chromatic_interval_segment\(\) \(in module abjad.tools.pitchtools.expr_to_melodic_chromatic_interval_segment\(\)\), 1886](#)
[extend\(\) \(abjad.tools.chordtools.Chord.Chord.Chord method\), 227](#)
[extend\(\) \(abjad.tools.containertools.Cluster.Cluster.Cluster method\), 297](#)
[extend\(\) \(abjad.tools.containertools.Container.Container.Container method\), 305](#)
[extend\(\) \(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method\), 344](#)
[extend\(\) \(abjad.tools.contexttools.Context.Context.Context method\), 352](#)
[extend\(\) \(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method\), 383](#)
[extend\(\) \(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method\), 1628](#)
[extend\(\) \(abjad.tools.gracetools.Grace.Grace.Grace method\), 420](#)
[extend\(\) \(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method\), 631](#)
[extend\(\) \(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method\), 811](#)
[extend\(\) \(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method\), 816](#)
[extend\(\) \(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method\), 828](#)
[extend\(\) \(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method\), 840](#)
[extend\(\) \(abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method\), 2019](#)
[extend\(\) \(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method\), 2048](#)
[extend\(\) \(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method\), 896](#)
[extend\(\) \(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method\), 910](#)

```

extend() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.Measure
method), 920
extend() (abjad.tools.measuretools.Measure.Measure.Measure
method), 929
extend() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn
method), 1735
extend() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow
method), 1738
extend() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping
method), 1154
extend() (abjad.tools.pitchtools.OctaveTranspositionMappingInterval.OctaveTranspositionMappingInterval
method), 1161
extend() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory
method), 1168
extend() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff
method), 1253
extend() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory
method), 1264
extend() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff
method), 1274
extend() (abjad.tools.scoretools.Score.Score.Score
method), 1284
extend() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup
method), 1295
extend() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList
method), 1744
extend() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner
method), 1319
extend() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner
method), 1327
extend() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner
method), 1336
extend() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner
method), 1346
extend() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner
method), 1356
extend() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner
method), 1366
extend() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner
method), 1374
extend() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner
method), 1382
extend() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner
method), 1391
extend() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner
method), 1399
extend() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner
method), 1406
extend() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner
method), 1416
extend() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner
method), 1424
extend() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner
method), 1432

```


[extend_left\(\) \(abjad.tools.spannertools.HairpinSpanner.HairpinSpanner class method\), 1392](#)
[extend_left\(\) \(abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner class method\), 1399](#)
[extend_left\(\) \(abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner class method\), 1406](#)
[extend_left\(\) \(abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner class method\), 1416](#)
[extend_left\(\) \(abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner class method\), 1424](#)
[extend_left\(\) \(abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner class method\), 1432](#)
[extend_left\(\) \(abjad.tools.spannertools.OctavationSpanner.OctavationSpanner class method\), 1441](#)
[extend_left\(\) \(abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner class method\), 1448](#)
[extend_left\(\) \(abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner class method\), 1456](#)
[extend_left\(\) \(abjad.tools.spannertools.SlurSpanner.SlurSpanner class method\), 1464](#)
[extend_left\(\) \(abjad.tools.spannertools.Spanner.Spanner class method\), 1470](#)
[extend_left\(\) \(abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner class method\), 1477](#)
[extend_left\(\) \(abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner class method\), 1485](#)
[extend_left\(\) \(abjad.tools.spannertools.TextSpanner.TextSpanner class method\), 1493](#)
[extend_left\(\) \(abjad.tools.spannertools.TrillSpanner.TrillSpanner class method\), 1500](#)
[extend_left\(\) \(abjad.tools.tietools.TieSpanner.TieSpanner class method\), 1546](#)
[extend_left_in_parent_of_component_and_do_not_grow_spanners\(\) \(in module abjad.tools.measuretools\), 943](#)
[extend_left_in_parent_of_component_and_grow_spanners\(\) \(in module abjad.tools.measuretools\), 944](#)
[extend_measures_in_expr_and_apply_full_measure_tuplets_to_measured_contents\(\) \(in module abjad.tools.measuretools\), 944](#)
[extent \(abjad.tools.tonalitytools.ChordClass.ChordClass class attribute\), 2074](#)
[extent \(abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator class attribute\), 2079](#)
[extent \(abjad.tools.tonalitytools.TonalFunction.TonalFunction class attribute\), 2102](#)
[extent_name \(abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator class attribute\), 2079](#)
[extent_to_figured_bass_string\(\) \(abjad.tools.tonalitytools.InversionIndicator.InversionIndicator class method\), 2056](#)

`find_index_of_spanner_component_at_score_offset()` (in module `abjad.tools.spannertools`), 1849
`find_index_of_spanner_component_at_score_offset()` (`abjad.tools.spannertools.find_index_of_spanner_component_at_score_offset()` method), 1866
`find_intervals_intersecting_or_tangent_to_interval()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_intersecting_or_tangent_to_interval()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1847
`find_intervals_intersecting_or_tangent_to_interval()` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` method), 1862
`find_intervals_intersecting_or_tangent_to_offset()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_intersecting_or_tangent_to_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1847
`find_intervals_intersecting_or_tangent_to_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` method), 1862
`find_intervals_starting_after_offset()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_starting_after_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1848
`find_intervals_starting_after_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` method), 1863
`find_intervals_starting_and_stopping_within_interval()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_starting_and_stopping_within_interval()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1848
`find_intervals_starting_and_stopping_within_interval()` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` method), 1864
`find_intervals_starting_at_offset()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_starting_at_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1848
`find_intervals_starting_at_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary` method), 1865
`find_intervals_starting_before_offset()` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin` method), 1833
`find_intervals_starting_before_offset()` (`abjad.tools.timeintervaltools.TimeIntervalTree` method), 1848

method), 1851

find_intervals_stopping_within_interval() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary method), 1871

find_parentage_of_index() (abjad.tools.quantizationtools.QGrid.QGrid.QGrid method), 2056

find_spanner_component_starting_at_exactly_score_offset() (in module abjad.tools.spannertools.find_spanner_component_starting_at_exactly_score_offset), 1506

find_subtree_divisibility() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree method), 2063

fingered_pitch (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 966

fingered_pitch (abjad.tools.notetools.Note.Note.Note attribute), 971

fingered_pitches (abjad.tools.chordtools.Chord.Chord.Chord attribute), 224

fix_contents_of_tuplets_in_expr() (in module abjad.tools.tuplettools.fix_contents_of_tuplets_in_expr), 1588

FixedDurationTuplet (class in abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet), 1564

FixedLengthStreamSolver (class in abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver), 1995

flatten_sequence() (in module abjad.tools.sequencetools.flatten_sequence), 1779

flatten_sequence_at_indices() (in module abjad.tools.sequencetools.flatten_sequence_at_indices), 1780

Flute (class in abjad.tools.instrumenttools.Flute.Flute), 589

force_fraction (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1567

force_fraction (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1579

format (abjad.tools.chordtools.Chord.Chord.Chord attribute), 224

format (abjad.tools.componenttools.Component.Component.Component attribute), 236

format (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 294

format (abjad.tools.containertools.Container.Container.Container attribute), 302

format (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark attribute), 340

format (abjad.tools.contexttools.Context.Context.Context attribute), 349

format (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark attribute), 361

format (abjad.tools.contexttools.FingeringMark.FingeringMark.FingeringMark attribute), 365

format (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 370

format (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark attribute), 374

format (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark attribute), 378

format (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), 388

format (abjad.tools.contexttools.TrillMark.TrillMark.TrillMark attribute), 416

format (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 429

format (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 436

format (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 443

format (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 450

format (abjad.tools.instrumenttools.BaronSaxophone.BaronSaxophone.BaronSaxophone attribute), 464

format (abjad.tools.instrumenttools.BaronVoice.BaronVoice.BaronVoice attribute), 471

format (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 485

format (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 485

format (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 513

format (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 492

format (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 499

format (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 507

format (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 457

format (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 520

format (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 527

format (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 534

format (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 541

format (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 548

format (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 562

format (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 555

format (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 569

format (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 576

format (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 583

format (abjad.tools.instrumenttools.Flute.Flute.Flute attribute), 590

format (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 597

format (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 604

format (abjad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 611

format (abjad.tools.instrumenttools.Harp.Harp.Harp attribute), 618

format (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 625

format (abjad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 637

format (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 644

format (abjad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 651

format (abjad.tools.instrumenttools.Piano.Piano.Piano attribute), 658

format (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 665

format (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 672

format (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 679

format (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 686

format (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 693

format (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 700

format (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 707

format (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 714

format (abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 721

format (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 728

format (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 735

format (abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 742

format (abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 749

format (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 756

format (abjad.tools.lilypondfiletools.AbjadRevisionToken.AbjadRevisionToken.AbjadRevisionToken attribute), 808

format (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock attribute), 810

format (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock attribute), 815

format (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock attribute), 820

format (abjad.tools.lilypondfiletools.DateToken.DateToken.DateToken attribute), 821

format (abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock attribute), 823

format (abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock attribute), 825

format (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile attribute), 827

format (abjad.tools.lilypondfiletools.LilyPondLanguageToken.LilyPondLanguageToken.LilyPondLanguageToken attribute), 832

format (abjad.tools.lilypondfiletools.LilyPondVersionToken.LilyPondVersionToken.LilyPondVersionToken attribute), 834

format (abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock attribute), 836

format (abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock attribute), 838

format (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock attribute), 839

format (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics attribute), 2017

format (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender attribute), 2025

format (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2030

format (abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics attribute), 2044

format (abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace attribute), 2034

format (abjad.tools.lyrictools.LyricText.LyricText.LyricText attribute), 2039

format (abjad.tools.marktools.Articulation.Articulation.Articulation attribute), 849

format (abjad.tools.marktools.BarLine.BarLine.BarLine attribute), 853

format (abjad.tools.marktools.BendAfter.BendAfter.BendAfter attribute), 857

format (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark attribute), 860

format (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment attribute), 864

format (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo attribute), 870

format (abjad.tools.markuptools.Markup.Markup.Markup attribute), 891

format (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand attribute), 894	format (abjad.tools.markuptools.MarkupCommand.MarkupCommand.MarkupCommand attribute), 1576
format (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 905	format (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 1599
format (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 916	format (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 1600
format (abjad.tools.measuretools.Measure.Measure.Measure attribute), 926	format (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1606
format (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 966	format (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 1687
format (abjad.tools.notetools.Note.Note.Note attribute), 972	format (abjad.tools.notetools.Note.Note.Note attribute), 1687
format (abjad.tools.notetools.NoteHead.NoteHead.NoteHead attribute), 977	format (abjad.tools.notetools.NoteHead.NoteHead.NoteHead attribute), 1687
format (abjad.tools.pitchtools.Accidental.Accidental.Accidental attribute), 989	format (abjad.tools.pitchtools.Accidental.Accidental.Accidental attribute), 1687
format (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch attribute), 1096	format (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch attribute), 1687
format (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1122	format (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1687
format (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1214	format (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1687
format (abjad.tools.resttools.Rest.Rest.Rest attribute), 1218	format (abjad.tools.resttools.Rest.Rest.Rest attribute), 1687
format (abjad.tools.schemetools.Scheme.Scheme.Scheme attribute), 1227	format (abjad.tools.schemetools.Scheme.Scheme.Scheme attribute), 1687
format (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList attribute), 1229	format (abjad.tools.schemetools.SchemeAssociativeList.SchemeAssociativeList.SchemeAssociativeList attribute), 1687
format (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor attribute), 1231	format (abjad.tools.schemetools.SchemeColor.SchemeColor.SchemeColor attribute), 1687
format (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment attribute), 1233	format (abjad.tools.schemetools.SchemeMoment.SchemeMoment.SchemeMoment attribute), 1687
format (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair attribute), 1235	format (abjad.tools.schemetools.SchemePair.SchemePair.SchemePair attribute), 1687
format (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector attribute), 1237	format (abjad.tools.schemetools.SchemeVector.SchemeVector.SchemeVector attribute), 1687
format (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant attribute), 1239	format (abjad.tools.schemetools.SchemeVectorConstant.SchemeVectorConstant.SchemeVectorConstant attribute), 1687
format (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1249	format (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1687
format (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1270	format (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1687
format (abjad.tools.scoretools.Score.Score.Score attribute), 1281	format (abjad.tools.scoretools.Score.Score.Score attribute), 1687
format (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1291	format (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1687
format (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1308	format (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1687
format (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1521	format (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1687
format (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1531	format (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1687
format (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1565	format (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1687

[fuse_like_named_contiguous_containers_in_expr\(\)](#) (in module `abjad.tools.containertools`), 313
[fuse_measures\(\)](#) (in module `abjad.tools.measuretools`), 946
[fuse_overlapping_intervals\(\)](#) (in module `abjad.tools.timeintervaltools`), 1886
[fuse_tangent_or_overlapping_intervals\(\)](#) (in module `abjad.tools.timeintervaltools`), 1887
[fuse_tied_leaves_in_components_once_by_prolated_durations_within_a_duration\(\)](#) (in module `abjad.tools.leafertools`), 777
[fuse_tuplets\(\)](#) (in module `abjad.tools.tuplettools`), 1588

G

[get\(\)](#) (`abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary` method), 1624
[get\(\)](#) (`abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph` method), 1650
[get\(\)](#) (`abjad.tools.pitchtools.HarmonicChromaticIntervalClass.Vector.HarmonicChromaticIntervalClass.Vector` method), 997
[get\(\)](#) (`abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.Vector.InversionEquivalentChromaticIntervalClass.Vector` method), 1040
[get\(\)](#) (`abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.Vector.InversionEquivalentDiatonicIntervalClass.Vector` method), 1050
[get\(\)](#) (`abjad.tools.pitchtools.MelodicChromaticIntervalClass.Vector.MelodicChromaticIntervalClass.Vector` method), 1063
[get\(\)](#) (`abjad.tools.pitchtools.NamedChromaticPitch.Vector.NamedChromaticPitch.Vector` method), 1117
[get\(\)](#) (`abjad.tools.pitchtools.NumberedChromaticPitchClass.ColorMap.NumberedChromaticPitchClass.ColorMap` method), 1132
[get\(\)](#) (`abjad.tools.pitchtools.NumberedChromaticPitchClass.Vector.NumberedChromaticPitchClass.Vector` method), 1144
[get\(\)](#) (`abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree` method), 2063
[get\(\)](#) (`abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup` method), 2068
[get\(\)](#) (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval` method), 1841
[get\(\)](#) (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary` method), 1871
[get_abjad_revision_string\(\)](#) (in module `abjad.tools.configurationtools`), 1618
[get_abjad_version_string\(\)](#) (in module `abjad.tools.configurationtools`), 1619
[get_all_unique_bounds_in_intervals\(\)](#) (in module `abjad.tools.timeintervaltools`), 778
[get_annotation_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 879
[get_annotations_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 879
[get_arithmetic_mean_of_chord\(\)](#) (in module `abjad.tools.chordtools`), 879
[get_articulation_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 880
[get_articulations_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 880
[get_beam_spanner_attached_to_component\(\)](#) (in module `abjad.tools.spannertools`), 1508
[get_boolean_train\(\)](#) (`abjad.tools.sievetools.ResidueClass.ResidueClass` method), 1820
[get_clef_mark_attached_to_component\(\)](#) (in module `abjad.tools.contexttools`), 396
[get_clef_marks_attached_to_component\(\)](#) (in module `abjad.tools.contexttools`), 396
[get_component_start_offset\(\)](#) (in module `abjad.tools.componenttools`), 257
[get_component_start_offset_in_seconds\(\)](#) (in module `abjad.tools.componenttools`), 257
[get_component_stop_offset\(\)](#) (in module `abjad.tools.componenttools`), 258
[get_component_stop_offset_in_seconds\(\)](#) (in module `abjad.tools.componenttools`), 258
[get_composite_offset_difference_series_from_leaves_in_expr\(\)](#) (in module `abjad.tools.leafertools`), 777
[get_composite_offset_series_from_leaves_in_expr\(\)](#) (in module `abjad.tools.leafertools`), 778

[get_congruent_bases\(\)](#) (abjad.tools.sievetools.ResidueClass.ResidueClass.ResidueClass method), 1820
[get_congruent_bases\(\)](#) (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression.ResidueClassExpression method), 1822
[get_context_mark_attached_to_component\(\)](#) (in module abjad.tools.contexttools.get_context_mark_attached_to_component), 397
[get_context_marks_attached_to_any_improper_parent_of_component\(\)](#) (in module abjad.tools.contexttools.get_context_marks_attached_to_any_improper_parent_of_component), 397
[get_context_marks_attached_to_component\(\)](#) (in module abjad.tools.contexttools.get_context_marks_attached_to_component), 398
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Accordion.Accordion.Accordion method), 433
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute method), 440
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone method), 447
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone method), 454
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BartoneSaxophone.BartoneSaxophone.BartoneSaxophone method), 468
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BartoneVoice.BartoneVoice.BartoneVoice method), 475
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet method), 482
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute method), 489
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon method), 517
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone method), 496
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone method), 503
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice method), 510
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet method), 461
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Cello.Cello.Cello method), 524
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA method), 531
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass method), 538
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet method), 545
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute method), 552
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon method), 566
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone method), 559
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice method), 573
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet method), 580
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn method), 587
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Flute.Flute.Flute method), 594
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn method), 600
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel method), 608
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Guitar.Guitar.Guitar method), 615
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Harp.Harp.Harp method), 622
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 629
[get_default_performer_name\(\)](#) (abjad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641

`get_default_performer_name()` (ab- 901
`jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.component()`
`method`), 648 (in module ab-
`get_default_performer_name()` (ab- jad.tools.contexttools.get_dynamic_mark_attached_to_component
`jad.tools.instrumenttools.Oboe.Oboe.Oboe` 398
`method`), 655 `get_dynamic_marks_attached_to_component()`
`get_default_performer_name()` (ab- (in module ab-
`jad.tools.instrumenttools.Piano.Piano.Piano` jad.tools.contexttools.get_dynamic_marks_attached_to_compone
`method`), 662 399
`get_default_performer_name()` (ab- `get_effective_clef()` (in module ab-
`jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo` jad.tools.contexttools.get_effective_clef),
`method`), 669 399
`get_default_performer_name()` (ab- `get_effective_context_mark()` (in module ab-
`jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone` jad.tools.contexttools.get_effective_context_mark),
`method`), 676 399
`get_default_performer_name()` (ab- `get_effective_dynamic()` (in module ab-
`jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone` jad.tools.contexttools.get_effective_dynamic),
`method`), 683 400
`get_default_performer_name()` (ab- `get_effective_instrument()` (in module ab-
`jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice` jad.tools.contexttools.get_effective_instrument),
`method`), 690 400
`get_default_performer_name()` (ab- `get_effective_key_signature()` (in module ab-
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone` jad.tools.contexttools.get_effective_key_signature),
`method`), 697 401
`get_default_performer_name()` (ab- `get_effective_staff()` (in module ab-
`jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone` jad.tools.contexttools.get_effective_staff),
`method`), 704 401
`get_default_performer_name()` (ab- `get_effective_tempo()` (in module ab-
`jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice` jad.tools.contexttools.get_effective_tempo),
`method`), 711 402
`get_default_performer_name()` (ab- `get_effective_time_signature()` (in module ab-
`jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet` jad.tools.contexttools.get_effective_time_signature),
`method`), 718 402
`get_default_performer_name()` (ab- `get_element_starting_at_exactly_prolated_offset()`
`jad.tools.instrumenttools.Tuba.Tuba.Tuba` (in module ab-
`method`), 725 jad.tools.containertools.get_element_starting_at_exactly_prolated
`get_default_performer_name()` (ab- 314
`jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion` `get_first_component_in_expr_with_name()`
`method`), 732 (in module ab-
`get_default_performer_name()` (ab- jad.tools.componenttools.get_first_component_in_expr_with_name
`jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone` 259
`method`), 739 `get_first_component_with_name_in_improper_parentage_of_component()`
`get_default_performer_name()` (ab- (in module ab-
`jad.tools.instrumenttools.Viola.Viola.Viola` jad.tools.componenttools.get_first_component_with_name_in_im
`method`), 746 259
`get_default_performer_name()` (ab- `get_first_component_with_name_in_proper_parentage_of_component()`
`jad.tools.instrumenttools.Violin.Violin.Violin` (in module ab-
`method`), 753 jad.tools.componenttools.get_first_component_with_name_in_pro
`get_default_performer_name()` (ab- 259
`jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone` `get_first_container_in_improper_parentage_of_component()`
`method`), 760 (in module ab-
`get_down_markup_attached_to_component()` jad.tools.containertools.get_first_container_in_improper_parentag
(in module ab- 314
`jad.tools.markuptools.get_down_markup_attached_to_component()`,

get_first_container_in_proper_parentage_of_component() (in module ab-	jad.tools.tuplettools.get_first_tuplet_in_improper_parentage_of_c 1589
jad.tools.containertools.get_first_container_in_proper_parentage_of_component() 314	(in module ab-
get_first_element_starting_at_or_after_prolated_offset() (in module ab-	jad.tools.tuplettools.get_first_tuplet_in_proper_parentage_of_com 1590
jad.tools.containertools.get_first_element_startingget_first_voice_in_improper_parentage_of_component() 315	(in module ab-
get_first_element_starting_before_or_at_prolated_offset() (in module ab-	jad.tools.voicetools.get_first_voice_in_improper_parentage_of_co 1608
jad.tools.containertools.get_first_element_startingbefore first voice prolated offset parentage_of_component() 315	(in module ab-
get_first_element_starting_strictly_after_prolated_offset() (in module ab-	jad.tools.voicetools.get_first_voice_in_proper_parentage_of_com 1609
jad.tools.containertools.get_first_element_startingstrictly after prolated offset to leaf() (in module ab- 315	jad.tools.gracetools.get_grace_containers_attached_to_leaf),
get_first_element_starting_strictly_before_prolated_offset() (in module ab-	426
jad.tools.containertools.get_first_element_starting_strictly before prolated offset to leaf() (in module ab- 316	get_improper_parentage_of_component() (in module ab- 260
get_first_instance_of_class_in_improper_parentage_of_component() (in module ab-	get_instrument_marks_attached_to_component() (in module ab- 260
jad.tools.componenttools.get_first_instance_of_class_in_improper_parentage_of_component() 1780	get_instrument_marks_attached_to_component() (in module ab- 403
get_first_instance_of_class_in_proper_parentage_of_component() (in module ab-	get_instrument_marks_attached_to_component() (in module ab- 403
jad.tools.componenttools.get_first_instance_of_class_in_proper_parentage_of_component() 403	get_instrument_marks_attached_to_component() (in module ab- 403
get_first_measure_in_improper_parentage_of_component() (in module ab-	get_instrument_marks_attached_to_component() (in module ab- 403
jad.tools.measuretools.get_first_measure_in_improper_parentage_of_component() 947	get_instrument_marks_attached_to_component() (in module ab- 403
get_first_measure_in_proper_parentage_of_component() (in module ab-	get_key_signature_marks_attached_to_component() (in module ab- 404
jad.tools.measuretools.get_first_measure_in_proper_parentage_of_component() 947	get_key_signature_marks_attached_to_component() (in module ab- 404
get_first_score_in_improper_parentage_of_component() (in module ab-	get_key_signature_marks_attached_to_component() (in module ab- 404
jad.tools.scoretools.get_first_score_in_improper_parentage_of_component() 1301	get_key_signature_marks_attached_to_component() (in module ab- 404
get_first_score_in_proper_parentage_of_component() (in module ab-	get_last_output_file_name() (in module ab- 1688
jad.tools.scoretools.get_first_score_in_proper_parentage_of_component() 1302	get_last_output_file_name() (in module ab- 1688
get_first_staff_in_improper_parentage_of_component() (in module ab-	get_le_n_solutions() (ab- 1541
jad.tools.stafftools.get_first_staff_in_improper_parentage_of_component() 1541	get_le_n_solutions() (ab- 1541
get_first_staff_in_proper_parentage_of_component() (in module ab-	get_le_n_solutions() (ab- 1541
jad.tools.stafftools.get_first_staff_in_proper_parentage_of_component() 1541	get_le_n_solutions() (ab- 1541
get_first_tuplet_in_improper_parentage_of_component() (in module ab-	778

[get_leaf_in_expr_with_maximum_prolated_duration\(\)](#) (in module `abjad.tools.leaftools`), 1769
[get_leaf_in_expr_with_maximum_prolated_duration\(\)](#) (in module `abjad.tools.leaftools`), 1779
[get_leaf_in_expr_with_minimum_prolated_duration\(\)](#) (in module `abjad.tools.leaftools`), 1779
[get_leaves_in_tie_chain\(\)](#) (in module `abjad.tools.tietools`), 1551
[get_likely_multiplier_of_components\(\)](#) (in module `abjad.tools.componenttools`), 261
[get_lilypond_command_mark_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 881
[get_lilypond_command_marks_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 881
[get_lilypond_comment_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 882
[get_lilypond_comments_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 884
[get_lilypond_version_string\(\)](#) (in module `abjad.tools.configurationtools`), 1619
[get_mark_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 883
[get_marks_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 883
[get_markup_attached_to_component\(\)](#) (in module `abjad.tools.markuptools`), 901
[get_named_chromatic_pitch_from_pitch_carrier\(\)](#) (in module `abjad.tools.pitchtools`), 1190
[get_next_measure_from_component\(\)](#) (in module `abjad.tools.measuretools`), 948
[get_next_n_complete_nodes_at_level\(\)](#) (in module `abjad.tools.sequencetools`), 1754
[get_next_n_complete_nodes_at_level\(\)](#) (in module `abjad.tools.sequencetools`), 1191
[get_next_n_nodes_at_level\(\)](#) (in module `abjad.tools.sequencetools`), 1754
[get_next_output_file_name\(\)](#) (in module `abjad.tools.iotools`), 1688
[get_node_at_position\(\)](#) (in module `abjad.tools.sequencetools`), 1755
[get_node_at_position\(\)](#) (in module `abjad.tools.sequencetools`), 1770
[get_nonbinary_factor_from_time_signature_denominator\(\)](#) (in module `abjad.tools.timesignaturetools`), 1894
[get_noncontext_marks_attached_to_component\(\)](#) (in module `abjad.tools.marktools`), 884
[get_note_head_from_chord_by_pitch\(\)](#) (in module `abjad.tools.chordtools`), 233
[get_nth_component_in_expr\(\)](#) (in module `abjad.tools.componenttools`), 261
[get_nth_leaf_in_expr\(\)](#) (in module `abjad.tools.leaftools`), 780
[get_nth_leaf_in_spanner\(\)](#) (in module `abjad.tools.spannertools`), 1508
[get_nth_leaf_in_thread_from_leaf\(\)](#) (in module `abjad.tools.leaftools`), 781
[get_nth_measure_in_expr\(\)](#) (in module `abjad.tools.measuretools`), 949
[get_nth_namesake_from_component\(\)](#) (in module `abjad.tools.componenttools`), 262
[get_numbered_chromatic_pitch_class_from_pitch_carrier\(\)](#) (in module `abjad.tools.pitchtools`), 1191

get_one_indexed_measure_number_in_expr()
(in module abjad.tools.measuretools.get_one_indexed_measure_number_in_expr(), 503
949

get_performer_names()
(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone) method, 503

get_overlap_with_interval()
(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice) method, 510
jad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method, 1841

get_overlap_with_interval()
(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet) method, 510
jad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method, 1833

get_overlap_with_interval()
(abjad.tools.instrumenttools.Cello.Cello.Cello) method, 510
jad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin method, 1837

get_overlap_with_interval()
(abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA) method, 510
jad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method, 1851

get_overlap_with_interval()
(abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass) method, 510
jad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method, 1871

get_parent_and_start_stop_indices_of_components()
(in module abjad.tools.componenttools.get_parent_and_start_stop_indices_of_components(), 263

get_performer_names()
(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute) method, 552

get_performer_names()
(abjad.tools.instrumenttools.Accordion.Accordion.Accordion) method, 433

get_performer_names()
(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon) method, 566

get_performer_names()
(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute) method, 440

get_performer_names()
(abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone) method, 559

get_performer_names()
(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone) method, 447

get_performer_names()
(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice) method, 573

get_performer_names()
(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone) method, 454

get_performer_names()
(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet) method, 580

get_performer_names()
(abjad.tools.instrumenttools.BartitoneSaxophone.BartitoneSaxophone.BartitoneSaxophone) method, 468

get_performer_names()
(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn) method, 587

get_performer_names()
(abjad.tools.instrumenttools.BartitoneVoice.BartitoneVoice.BartitoneVoice) method, 475

get_performer_names()
(abjad.tools.instrumenttools.Flute.Flute.Flute) method, 594

get_performer_names()
(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet) method, 482

get_performer_names()
(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn) method, 600

get_performer_names()
(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute) method, 489

get_performer_names()
(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel) method, 608

get_performer_names()
(abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon) method, 517

get_performer_names()
(abjad.tools.instrumenttools.Guitar.Guitar.Guitar) method, 615

get_performer_names()
(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone) method, 496

get_performer_names()
(abjad.tools.instrumenttools.Harp.Harp.Harp) method, 622

get_performer_names() jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord method), 629	(ab- get_performer_names() jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone method), 760	(ab-
get_performer_names() jad.tools.instrumenttools.Marimba.Marimba.Marimba method), 641	(ab- get_position_of_descendant() jad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1755	(ab-
get_performer_names() jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice method), 648	(ab- get_position_of_descendant() jad.tools.sequencetools.Tree.Tree method), 1770	(ab-
get_performer_names() jad.tools.instrumenttools.Oboe.Oboe.Oboe method), 655	(ab- get_preprolated_tie_chain_duration() (in module ab- jad.tools.tietools.get_preprolated_tie_chain_duration), 1551	
get_performer_names() jad.tools.instrumenttools.Piano.Piano.Piano method), 662	(ab- get_prev_measure_from_component() (in module ab- jad.tools.measuretools.get_prev_measure_from_component), 950	
get_performer_names() jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo method), 669	(ab- get_prolated_tie_chain_duration() (in module ab- jad.tools.tietools.get_prolated_tie_chain_duration), 1552	
get_performer_names() jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone method), 676	(ab- get_proper_parentage_of_component() (in module ab- jad.tools.sequencetools.get_proper_parentage_of_component), 263	
get_performer_names() jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone method), 683	(ab- get_python_version_string() (in module ab- jad.tools.sequencetools.get_python_version_string), 1619	
get_performer_names() jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice method), 690	(ab- get_sequence_degree_of_rotational_symmetry() (in module ab- jad.tools.sequencetools.get_sequence_degree_of_rotational_symmetry), 1780	
get_performer_names() jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone method), 697	(ab- get_sequence_element_at_cyclic_index() (in module ab- jad.tools.sequencetools.get_sequence_element_at_cyclic_index), 1781	
get_performer_names() jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone method), 704	(ab- get_sequence_elements_at_indices() (in module ab- jad.tools.sequencetools.get_sequence_elements_at_indices), 1781	
get_performer_names() jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice method), 711	(ab- get_sequence_elements_frequency_distribution() (in module ab- jad.tools.sequencetools.get_sequence_elements_frequency_distribution), 1782	
get_performer_names() jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet method), 718	(ab- get_sequence_period_of_rotation() (in module ab- jad.tools.sequencetools.get_sequence_period_of_rotation), 1782	
get_performer_names() jad.tools.instrumenttools.Tuba.Tuba.Tuba method), 725	(ab- get_shared_numeric_sign() (in module ab- jad.tools.mathtools.get_shared_numeric_sign), 1701	
get_performer_names() jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion method), 732	(ab- get_spanners_attached_to_any_improper_child_of_component() (in module ab- jad.tools.spannertools.get_spanners_attached_to_any_improper_child_of_component), 1508	
get_performer_names() jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone method), 739	(ab- get_spanners_attached_to_any_improper_parent_of_component() (in module ab- jad.tools.spannertools.get_spanners_attached_to_any_improper_parent_of_component), 1509	
get_performer_names() jad.tools.instrumenttools.Viola.Viola.Viola method), 746	(ab- get_spanners_attached_to_any_proper_child_of_component() (in module ab- jad.tools.spannertools.get_spanners_attached_to_any_proper_child_of_component), 1509	
get_performer_names() jad.tools.instrumenttools.Violin.Violin.Violin method), 753	(ab-	

jad.tools.spannertools.get_spanners_attached_to_any_proper_child_of_component(), 1509	get_tempo_marks_attached_to_component()
get_spanners_attached_to_any_proper_parent_of_component() (in module ab- (in module ab- jad.tools.spannertools.get_spanners_attached_to_any_proper_parent_of_component()), 1510	get_the_only_spanner_attached_to_any_improper_parent_of_component()
get_spanners_attached_to_component() (in module ab- jad.tools.spannertools.get_spanners_attached_to_component()), 1511	get_the_only_spanner_attached_to_component() (in module ab- jad.tools.spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component()), 1513
get_spanners_contained_by_components() (in module ab- jad.tools.spannertools.get_spanners_contained_by_components()), 1511	get_the_only_spanner_attached_to_component() (in module ab- jad.tools.spannertools.get_the_only_spanner_attached_to_any_improper_parent_of_component()), 1513
get_spanners_covered_by_components() (in module ab- jad.tools.spannertools.get_spanners_covered_by_components()), 1511	get_tie_chain() (in module ab- jad.tools.tietools.get_tie_chain()), 1552
get_spanners_on_components_or_component_children() (in module ab- jad.tools.spannertools.get_spanners_on_components_or_component_children()), 1512	get_tie_chain_duration_in_seconds() (in module ab- jad.tools.tietools.get_tie_chain_duration_in_seconds()), 1552
get_spanners_that_cross_components() (in module ab- jad.tools.spannertools.get_spanners_that_cross_components()), 1512	get_tie_chains_in_expr() (in module ab- jad.tools.tietools.get_tie_chains_in_expr()), 1552
get_spanners_that_dominates_component_pair() (in module ab- jad.tools.spannertools.get_spanners_that_dominates_component_pair()), 1512	signature_mark_attached_to_component() (in module ab- jad.tools.contexttools.get_time_signature_mark_attached_to_component()), 407
get_spanners_that_dominates_components() (in module ab- jad.tools.spannertools.get_spanners_that_dominates_components()), 1512	signature_marks_attached_to_component() (in module ab- jad.tools.contexttools.get_time_signature_marks_attached_to_components()), 407
get_spanners_that_dominates_container_components_from_to() (in module ab- jad.tools.spannertools.get_spanners_that_dominates_container_components_from_to()), 1513	signature_marks_attached_to_component() (in module ab- jad.tools.markuptools.get_up_markup_attached_to_component()), 902
get_staff_change_mark_attached_to_component() (in module ab- jad.tools.contexttools.get_staff_change_mark_attached_to_component()), 405	signature_marks_attached_to_component() (in module ab- jad.tools.marktools.get_value_of_annotation_attached_to_component()), 885
get_staff_change_marks_attached_to_component() (in module ab- jad.tools.contexttools.get_staff_change_marks_attached_to_component()), 405	signature_marks_attached_to_component() (in module ab- jad.tools.verticalitytools.get_vertical_moment_at_prolated_offset_in_expr()), 1944
get_stem_tremolo_attached_to_component() (in module ab- jad.tools.marktools.get_stem_tremolo_attached_to_component()), 885	signature_marks_attached_to_component() (in module ab- jad.tools.verticalitytools.get_vertical_moment_starting_with_component()), 1945
get_stem_tremolos_attached_to_component() (in module ab- jad.tools.marktools.get_stem_tremolos_attached_to_component()), 885	signature_marks_attached_to_component() (in module ab- jad.tools.tietools.get_written_tie_chain_duration()), 1552
get_tempo_mark_attached_to_component() (in module ab- jad.tools.contexttools.get_tempo_mark_attached_to_component()), 828	signature_marks_attached_to_component() (in module ab- jad.tools.constrainttools.GlobalConstraint.GlobalConstraint), 828

1999
GlobalCountsConstraint (class in abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint),
jad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint), 602

2001
GlobalReferenceConstraint (class in abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint),
jad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint), 1009

2004
Glockenspiel (class in abjad.tools.instrumenttools.Glockenspiel.Glockenspiel),
jad.tools.instrumenttools.Glockenspiel.Glockenspiel), attribute), 1005

603
governors (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment),
attribute), 1942

Grace (class in abjad.tools.gracetools.Grace.Grace), 415

GrandStaff (class in abjad.tools.scoretools.GrandStaff.GrandStaff),
jad.tools.scoretools.GrandStaff.GrandStaff), attribute), 1067

1248
greatest_common_divisor() (in module abjad.tools.mathtools.greatest_common_divisor),
attribute), 1086

1712
greatest_multiple_less_equal() (in module abjad.tools.mathtools.greatest_multiple_less_equal),
attribute), 1109

1712
greatest_power_of_two_less_equal() (in module abjad.tools.mathtools.greatest_power_of_two_less_equal),
attribute), 2079

1713
group_duration_tokens_by_implied_prolation() (in module abjad.tools.durationtools.group_duration_tokens_by_implied_prolation),
attribute), 1026

1672
group_leaves_in_tie_chain_by_immediate_parents() (in module abjad.tools.tietools.group_leaves_in_tie_chain_by_immediate_parents),
attribute), 1071

1553
group_overlapping_intervals_and_yield_groups() (in module abjad.tools.timeintervaltools.group_overlapping_intervals_and_yield_groups),
attribute), 1090

1887
group_tangent_or_overlapping_intervals_and_yield_groups() (in module abjad.tools.timeintervaltools.group_tangent_or_overlapping_intervals_and_yield_groups),
attribute), 1013

1887
Guitar (class in abjad.tools.instrumenttools.Guitar.Guitar),
attribute), 1081

610
harmonic_chromatic_interval_class (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval), 1002

harmonic_chromatic_interval_class_segment (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1009

harmonic_chromatic_interval_numbers (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet), 1005

harmonic_chromatic_interval_segment (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1067

harmonic_chromatic_interval_segment (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment), 1086

harmonic_chromatic_interval_segment (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment), 1109

harmonic_chromatic_interval_segment (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator), 2079

harmonic_chromatic_interval_set (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet), 1026

harmonic_chromatic_interval_set (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet), 1071

harmonic_chromatic_interval_set (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet), 1090

harmonic_chromatic_intervals (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet), 1005

harmonic_counterpoint_interval (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval), 1013

harmonic_counterpoint_interval_class (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval), 1009

H
HairpinSpanner (class in abjad.tools.spannertools.HairpinSpanner.HairpinSpanner),
jad.tools.spannertools.HairpinSpanner.HairpinSpanner), attribute), 1081

1386
harmonic_chromatic_interval (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval),
attribute), 1055

harmonic_chromatic_interval (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval),
attribute), 1081

harmonic_chromatic_interval (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval),
attribute), 1013

harmonic_chromatic_interval (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment),
attribute), 1109

harmonic_diatonic_interval_classes (ab- Harp (class in abjad.tools.instrumenttools.Harp.Harp),
jad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet
attribute), 1018 Harpsichord (class in ab-

harmonic_diatonic_interval_numbers (ab- jad.tools.instrumenttools.Harpsichord.Harpsichord),
jad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet
attribute), 1026 has_key() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableD

harmonic_diatonic_interval_segment (ab- method), 1624
jad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment
attribute), 1086 method), 1650

harmonic_diatonic_interval_segment (ab- has_key() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.H
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment
attribute), 1109 has_key() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClas

harmonic_diatonic_interval_set (ab- method), 1040
jad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet
attribute), 1090 method), 1050

harmonic_diatonic_intervals (ab- has_key() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.Me
jad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet
attribute), 1026 has_key() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChrom

HarmonicChromaticInterval (class in ab- method), 1117
jad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval.HarmonicChromaticInterval
991 method), 1144

HarmonicChromaticIntervalClass (class in ab- has_key() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree
jad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass),
994 has_key() (abjad.tools.quantizationtools.QGridTempoLookup.QGridTempo

HarmonicChromaticIntervalClassVector (class in ab- method), 2068
jad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector
996 method), 1872

HarmonicChromaticIntervalSegment (class in ab- has_none_of() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVect
jad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment),
1001 method), 1907

HarmonicChromaticIntervalSet (class in ab- has_spanning_cell_over_index() (ab-
jad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet),
1004 has_spanning_cell_over_index() (ab-

HarmonicCounterpointInterval (class in ab- jad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray
jad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval),
1008 has_voice_crossing (ab-

HarmonicCounterpointIntervalClass (class in ab- jad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray
jad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass),
1011 has_voice_crossing (ab-

HarmonicDiatonicInterval (class in ab- jad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.P
jad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval),
1013 HeaderBlock (class in ab-

HarmonicDiatonicIntervalClass (class in ab- jad.tools.lilypondfiletools.HeaderBlock.HeaderBlock),
jad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass),
1015 HiddenStaffSpanner (class in ab-

HarmonicDiatonicIntervalClassSet (class in ab- jad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner),
jad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet),
1018 HorizontalBracketSpanner (class in ab-

HarmonicDiatonicIntervalSegment (class in ab- jad.tools.spannertools.HorizontalBracketSpanner.HorizontalBrack
jad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment),
1022

HarmonicDiatonicIntervalSet (class in ab- |
jad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet), (ab-
1025 jad.tools.documentationtools.ModuleCrawler.ModuleCrawler.Mo

- attribute), 1653
- imag (abjad.tools.durationtools.Duration.Duration.Duration attribute), 1660
- imag (abjad.tools.durationtools.Offset.Offset.Offset attribute), 1664
- ImmutableAbjadObject (class in abjad.tools.abctools.ImmutableAbjadObject.ImmutableAbjadObject), 1615
- ImmutableDictionary (class in abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary), 1624
- improper_parentage (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1752
- improper_parentage (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1767
- IncisedTimeTokenMaker (class in abjad.tools.timetokentools.IncisedTimeTokenMaker.IncisedTimeTokenMaker), 1898
- include_private_objects (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler attribute), 1637
- include_private_objects (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler attribute), 1644
- include_rests (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1344
- include_rests (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1354
- include_rests (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1389
- increase_sequence_elements_at_indices_by_addenda() (in module abjad.tools.sequencetools.increase_sequence_elements_at_indices_by_addenda), 1782
- increase_sequence_elements_cyclically_by_addenda() (in module abjad.tools.sequencetools.increase_sequence_elements_cyclically_by_addenda), 1782
- index() (abjad.tools.containertools.Cluster.Cluster.Cluster method), 297
- index() (abjad.tools.containertools.Container.Container.Container method), 305
- index() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 344
- index() (abjad.tools.contexttools.Context.Context.Context method), 353
- index() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 383
- index() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1628
- index() (abjad.tools.gracetools.Grace.Grace.Grace method), 420
- index() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 632
- index() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 811
- index() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 816
- index() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 828
- index() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 840
- index() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2020
- index() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2048
- index() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 897
- index() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 910
- index() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 921
- index() (abjad.tools.measuretools.Measure.Measure.Measure method), 930
- index() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1738
- index() (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment method), 1002
- index() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment method), 1023
- index() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment method), 1032
- index() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment method), 1047
- index() (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment method), 1060
- index() (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment method), 1068
- index() (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment method), 1087
- index() (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment method), 1102
- index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment method), 1109
- index() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1135
- index() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1154
- index() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1161
- index() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1168
- index() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow method), 1173
- index() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff method), 1254

index() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1264	index() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory method), 1494
index() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff method), 1275	index() (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner method), 1502
index() (abjad.tools.scoretools.Score.Score.Score method), 1285	index() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff method), 1525
index() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1295	index() (abjad.tools.stafftools.Staff.Staff.Staff method), 1536
index() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1744	index() (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner method), 1548
index() (abjad.tools.sequencetools.CyclicTuple.CyclicTuple.CyclicTuple method), 1761	index() (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator method), 2080
index() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1321	index() (abjad.tools.tonalitytools.Scale.Scale.Scale method), 2095
index() (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner method), 1329	index() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet method), 1571
index() (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner method), 1338	index() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet method), 1582
index() (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner method), 1348	index() (abjad.tools.voicetools.Voice.Voice.Voice method), 1603
index() (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner method), 1358	index() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1752
index() (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner method), 1368	index() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1768
index() (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner method), 1376	index() (abjad.tools.scoretools.RelativeCountsConstraint.RelativeCountsConstraint attribute), 2007
index() (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner method), 1383	index() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint attribute), 2010
index() (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner method), 1393	index() (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint attribute), 1991
index() (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner method), 1400	index() (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint attribute), 2008
index() (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner method), 1408	index() (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint attribute), 2011
index() (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner method), 1417	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1425	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner method), 1434	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner method), 1442	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner method), 1450	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner method), 1457	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner method), 1465	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.Spanner.Spanner.Spanner method), 1471	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner method), 1479	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732
index() (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner method), 1486	index() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1732

insert() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.Voice.Voice.Voice
 method), 383
 insert() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.insert_subruns_in_chromatic_pitch_class_number_1
 method), 1629
 insert() (abjad.tools.gracetools.Grace.Grace.Grace abjad.tools.pitchtools.insert_and_transpose_nested_subruns_in_chro
 method), 421
 insert() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.insert_and_transpose_nested_subruns_in_chro
 method), 632
 insert() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock abjad.tools.containertools.insert_component_and_do_not_fracture_
 method), 811
 insert() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock abjad.tools.containertools.insert_component_and_fracture_crossing_spanners()
 method), 816
 insert() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile abjad.tools.containertools.insert_component_and_fracture_crossing_
 method), 828
 insert() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock abjad.tools.containertools.insert_component_and_fracture_crossing_
 method), 840
 insert() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics abjad.tools.pitchtools.instantiate_pitch_and_interval_test_collection
 method), 2020
 insert() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics instrument_count (abjad.tools.scoretools.InstrumentationSpecifier.Instrumen
 method), 2048
 insert() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory abjad.tools.scoretools.Performer.Performer.Performer
 method), 897
 insert() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure abjad.tools.measuretools.InstrumentMark.InstrumentMark
 method), 911
 insert() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure abjad.tools.measuretools.InstrumentMark.InstrumentMark
 method), 921
 insert() (abjad.tools.measuretools.Measure.Measure.Measure instrument_name (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute
 method), 930
 insert() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping
 method), 1154
 insert() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping
 method), 1161
 insert() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone
 method), 1168
 insert() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff instrument_name (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice
 method), 1254
 insert() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory abjad.tools.instrumenttools.BassClarinet.BassClarinet.Ba
 method), 1265
 insert() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff instrument_name (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlu
 method), 1275
 insert() (abjad.tools.scoretools.Score.Score.Score instrument_name (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon
 method), 1285
 insert() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup instrument_name (abjad.tools.instrumenttools.BassSaxophone.BassSaxopho
 method), 1296
 insert() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList instrument_name (abjad.tools.instrumenttools.BassTrombone.BassTrombon
 method), 1744
 insert() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff abjad.tools.instrumenttools.BassVoice.BassVoice.BassVo
 method), 1526
 insert() (abjad.tools.stafftools.Staff.Staff.Staff instrument_name (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.I
 method), 1536
 insert() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet
 method), 1571
 insert() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet instrument_name (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.Clar
 method), 1582

attribute), 522	attribute), 653
instrument_name_markup jad.tools.instrumenttools.ClarineteInA.ClarineteInA.ClarineteInA attribute), 529	instrument_name_markup jad.tools.instrumenttools.Piano.Piano.Piano attribute), 660
instrument_name_markup jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 536	instrument_name_markup jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 667
instrument_name_markup jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 543	instrument_name_markup jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 674
instrument_name_markup jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 550	instrument_name_markup jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 681
instrument_name_markup jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 564	instrument_name_markup jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 688
instrument_name_markup jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 557	instrument_name_markup jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 695
instrument_name_markup jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 571	instrument_name_markup jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 702
instrument_name_markup jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 578	instrument_name_markup jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 709
instrument_name_markup jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 585	instrument_name_markup jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 716
instrument_name_markup jad.tools.instrumenttools.Flute.Flute.Flute attribute), 592	instrument_name_markup jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 723
instrument_name_markup jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 599	instrument_name_markup jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 730
instrument_name_markup jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 606	instrument_name_markup jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 737
instrument_name_markup jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 613	instrument_name_markup jad.tools.instrumenttools.Viola.Viola.Viola attribute), 744
instrument_name_markup jad.tools.instrumenttools.Harp.Harp.Harp attribute), 620	instrument_name_markup jad.tools.instrumenttools.Violin.Violin.Violin attribute), 751
instrument_name_markup jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 627	instrument_name_markup jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 758
instrument_name_markup jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 639	InstrumentationSpecifier (class in ab- jad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpec 1258
instrument_name_markup jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 646	InstrumentInventory (class in ab- jad.tools.scoretools.InstrumentInventory.InstrumentInventory 631
instrument_name_markup jad.tools.instrumenttools.Oboe.Oboe.Oboe	InstrumentMark (class in ab- jad.tools.contexttools.InstrumentMark.InstrumentMark),

364	intersection() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1105
instruments (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1259	intersection() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1139
instruments (abjad.tools.scoretools.Performer.Performer.Performer attribute), 1262	intersection() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass method), 2075
integer_equivalent_number_to_integer() (in module abjad.tools.mathtools.integer_equivalent_number_to_integer), 1714	interval_class (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 992
integer_tempo_to_multiplier_tempo_pairs() (in module abjad.tools.tempotools.integer_tempo_to_multiplier_tempo_pairs), 1825	interval_class (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1009
integer_tempo_to_multiplier_tempo_pairs_report() (in module abjad.tools.tempotools.integer_tempo_to_multiplier_tempo_pairs_report), 1825	interval_class (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1013
integer_to_base_k_tuple() (in module abjad.tools.mathtools.integer_to_base_k_tuple), 1714	interval_class (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval attribute), 1055
integer_to_binary_string() (in module abjad.tools.mathtools.integer_to_binary_string), 1714	interval_class (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval attribute), 1075
interlace_sequences() (in module abjad.tools.sequencetools.interlace_sequences), 1783	interval_class (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1081
IntermarkedHairpinCheck (class in abjad.tools.wellformednesstools.IntermarkedHairpinCheck), 1964	interval_class_numbers (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment attribute), 1032
interpolate_cosine() (in module abjad.tools.mathtools.interpolate_cosine), 1715	interval_class_numbers (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment attribute), 1060
interpolate_divide() (in module abjad.tools.mathtools.interpolate_divide), 1715	interval_classes (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment attribute), 1001
interpolate_divide_multiple() (in module abjad.tools.mathtools.interpolate_divide_multiple), 1716	interval_classes (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1022
interpolate_exponential() (in module abjad.tools.mathtools.interpolate_exponential), 1716	interval_classes (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSegment.InversionEquivalentChromaticIntervalClassSegment attribute), 1032
interpolate_linear() (in module abjad.tools.mathtools.interpolate_linear), 1717	interval_classes (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment attribute), 1047
intersection() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1005	interval_classes (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment attribute), 1060
intersection() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1019	interval_classes (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment attribute), 1067
intersection() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1026	interval_classes (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment attribute), 1086
intersection() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 1036	interval_classes (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 2079
intersection() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1072	interval_set (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 436
intersection() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1090	interval_of_transposition (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 450
intersection() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1105	interval_of_transposition (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 450
	interval_of_transposition (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone attribute), 450

attribute), 464
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BartitoneVoice.BartitoneVoice.BartitoneVoice
attribute), 471
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet
attribute), 478
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel
attribute), 485
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon jad.tools.instrumenttools.Guitar.Guitar.Guitar
attribute), 513
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone jad.tools.instrumenttools.Harp.Harp.Harp
attribute), 492
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord
attribute), 499
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice jad.tools.instrumenttools.Marimba.Marimba.Marimba
attribute), 506
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice
attribute), 457
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.Cello.Cello.Cello jad.tools.instrumenttools.Oboe.Oboe.Oboe
attribute), 520
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA jad.tools.instrumenttools.Piano.Piano.Piano
attribute), 527
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo
attribute), 534
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone
attribute), 541
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.
attribute), 548
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice
attribute), 562
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone
attribute), 555
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone
attribute), 569
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice
attribute), 576
interval_of_transposition (ab- interval_of_transposition (ab-
jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet

attribute), 714

interval_of_transposition (abjad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 721

interval_of_transposition (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion attribute), 728

interval_of_transposition (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 735

interval_of_transposition (abjad.tools.instrumenttools.Viola.Viola.Viola attribute), 742

interval_of_transposition (abjad.tools.instrumenttools.Violin.Violin.Violin attribute), 749

interval_of_transposition (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 756

interval_string (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1014

interval_string (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1081

intervals (abjad.tools.pitchtools.HarmonicChromaticIntervalSegment.HarmonicChromaticIntervalSegment attribute), 1001

intervals (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1022

intervals (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment.InversionEquivalentDiatonicIntervalClassSegment attribute), 1047

intervals (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment attribute), 1068

intervals (abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment attribute), 1086

intervals (abjad.tools.timeintervaltools.TimeIntervalAggregate.Mixin.TimeIntervalAggregateMixin attribute), 1832

intervals (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree attribute), 1846

intervals (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1860

intervals (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 2080

inventory_aggregate_subsets() (in module abjad.tools.pitchtools.inventory_aggregate_subsets), 1193

inventory_inversion_equivalent_diatonic_interval_classes() (in module abjad.tools.pitchtools.inventory_inversion_equivalent_diatonic_interval_classes), 1193

inventory_name (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory attribute), 344

inventory_name (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory attribute), 383

inventory_name (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory attribute), 1628

inventory_name (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory attribute), 631

inventory_name (abjad.tools.markuptools.MarkupInventory.MarkupInventory attribute), 896

inventory_name (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping attribute), 1160

inventory_name (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory attribute), 1167

inventory_name (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory attribute), 1264

inversion (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 2074

inversion (abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 2080

inversion (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction attribute), 2102

inversion_equivalent_chromatic_interval_class (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass attribute), 1035

inversion_equivalent_chromatic_interval_class_segment (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment attribute), 1109

inversion_equivalent_chromatic_interval_class_set (abjad.tools.pitchtools.NumberedChromaticPitchClassSet attribute), 1139

inversion_equivalent_chromatic_interval_class_vector (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1109

inversion_equivalent_chromatic_interval_classes (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet attribute), 1035

inversion_equivalent_diatonic_interval_class_segment (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NamedChromaticPitchClassSegment attribute), 1101

inversion_equivalent_diatonic_interval_class_segment (abjad.tools.tonalitytools.Scale.Scale.Scale attribute), 2094	is_assignable_integer() (in module abjad.tools.mathtools.is_assignable_integer), 1717
inversion_equivalent_diatonic_interval_class_vector (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet attribute), 1104	is_assignable_rational() (in module abjad.tools.durationtools.is_assignable_rational), 1672
inversion_equivalent_diatonic_interval_class_vector (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 2075	is_at_level() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1755
InversionEquivalentChromaticIntervalClass (class in abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet), 1029	is_at_level() (abjad.tools.sequencetools.Tree.Tree.Tree method), 1774
InversionEquivalentChromaticIntervalClassSegment (class in abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet), 1031	is_augmentation (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1565
InversionEquivalentChromaticIntervalClassSet (class in abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet), 1035	is_augmentation (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1576
InversionEquivalentChromaticIntervalClassVector (class in abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet), 1039	is_bar_line_crossing_leaf() (in module abjad.tools.leaftools.is_bar_line_crossing_leaf), 781
InversionEquivalentDiatonicIntervalClass (class in abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet), 1044	is_beamable_component() (in module abjad.tools.componenttools.is_beamable_component), 264
InversionEquivalentDiatonicIntervalClassSegment (class in abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet), 1046	is_binary (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 905
InversionEquivalentDiatonicIntervalClassVector (class in abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSet), 1050	is_binary (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 916
InversionIndicator (class in abjad.tools.tonalitytools.InversionIndicator), 2086	is_binary (abjad.tools.measuretools.Measure.Measure attribute), 926
invert() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass method), 1016	is_binary (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1565
invert() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass method), 1130	is_binary (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1577
invert() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment method), 1135	is_binary_rational() (in module abjad.tools.durationtools.is_binary_rational), 1723
invert() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140	is_chromatic_pitch_class_name() (in module abjad.tools.pitchtools.is_chromatic_pitch_class_name), 1194
invert() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow method), 1173	is_chromatic_pitch_class_name_octave_number_pair() (in module abjad.tools.pitchtools.is_chromatic_pitch_class_name_octave_number_pair), 1194
is_abstract (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1639	is_chromatic_pitch_name() (in module abjad.tools.pitchtools.is_chromatic_pitch_name), 1195
is_adjusted (abjad.tools.pitchtools.Accidental.Accidental attribute), 990	is_chromatic_pitch_number() (in module abjad.tools.pitchtools.is_chromatic_pitch_number), 1195
is_alphabetic_accidental_abbreviation() (in module abjad.tools.pitchtools.is_alphabetic_accidental_abbreviation), 886	is_chromatic_pitch_number() (in module abjad.tools.pitchtools.is_chromatic_pitch_number), 1195

[is_component_with_articulation_attached\(\)](#)
 (in module `abjad.tools.marktools`), 887

[is_component_with_beam_spanner_attached\(\)](#)
 (in module `abjad.tools.spannertools`), 1514

[is_component_with_clef_mark_attached\(\)](#) (in module `abjad.tools.contexttools`), 408

[is_component_with_context_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 408

[is_component_with_dynamic_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 409

[is_component_with_instrument_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 409

[is_component_with_key_signature_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 410

[is_component_with_lilypond_command_mark_attached\(\)](#)
 (in module `abjad.tools.marktools`), 887

[is_component_with_lilypond_comment_attached\(\)](#)
 (in module `abjad.tools.marktools`), 887

[is_component_with_mark_attached\(\)](#) (in module `abjad.tools.marktools`), 888

[is_component_with_noncontext_mark_attached\(\)](#)
 (in module `abjad.tools.marktools`), 888

[is_component_with_spanner_attached\(\)](#) (in module `abjad.tools.spannertools`), 1514

[is_component_with_staff_change_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 410

[is_component_with_stem_tremolo_attached\(\)](#)
 (in module `abjad.tools.marktools`), 889

[is_component_with_tempo_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 411

[is_component_with_tie_spanner_attached\(\)](#)
 (in module `abjad.tools.tietools`), 1553

[is_component_with_time_signature_mark_attached\(\)](#)
 (in module `abjad.tools.contexttools`), 412

[is_congruent_base\(\)](#)
 (`abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression` method), 1822

[is_contained_by_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval` method), 1841

[is_contained_by_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin` method), 1833

[is_contained_by_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin` method), 1838

[is_contained_by_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree` method), 1851

[is_contained_by_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary` method), 1872

[is_container_of_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval` method), 1841

[is_container_of_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin` method), 1833

[is_container_of_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin` method), 1838

[is_container_of_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree` method), 1851

[is_container_of_interval\(\)](#)
 (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary` method), 1872

[is_cyclic\(\)](#) (`abjad.tools.datastructuretools.Digraph.Digraph.Digraph` attribute), 1621

[is_defective\(\)](#) (`abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn` attribute), 1735

[is_defective\(\)](#) (`abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow` attribute), 1737

[is_diatonic_pitch_class_name\(\)](#) (in module `abjad.tools.pitchtools`), 1195

[is_diatonic_pitch_class_number\(\)](#) (in module `abjad.tools.pitchtools`.[is_diatonic_pitch_class_number\(\)](#),
[1195](#)
[is_diatonic_pitch_name\(\)](#) (in module `abjad.tools.pitchtools`.[is_diatonic_pitch_name\(\)](#),
[1195](#)
[is_diatonic_pitch_number\(\)](#) (in module `abjad.tools.pitchtools`.[is_diatonic_pitch_number\(\)](#),
[1196](#)
[is_diatonic_quality_abbreviation\(\)](#) (in module `abjad.tools.pitchtools`.[is_diatonic_quality_abbreviation\(\)](#),
[1196](#)
[is_diminution\(\)](#) (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet`
[attribute](#)), [1565](#)
[is_diminution\(\)](#) (`abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet`
[attribute](#)), [1577](#)
[is_dotted_integer\(\)](#) (in module `abjad.tools.mathtools`.[is_dotted_integer\(\)](#), [1717](#)
[is_doubling\(\)](#) (`abjad.tools.scoretools.Performer.Performer.Performer`
[attribute](#)), [1261](#)
[is_duration_pair\(\)](#) (in module `abjad.tools.durationtools`.[is_duration_pair\(\)](#),
[1673](#)
[is_duration_token\(\)](#) (in module `abjad.tools.durationtools`.[is_duration_token\(\)](#),
[1674](#)
[is_empty\(\)](#) (`abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator`
[attribute](#)), [2100](#)
[is_equivalent_under_transposition\(\)](#) (`abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment`
[method](#)), [1102](#)
[is_equivalent_under_transposition\(\)](#) (`abjad.tools.tonalitytools.Scale.Scale.Scale`
[method](#)), [2095](#)
[is_first_in_row\(\)](#) (`abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell`
[attribute](#)), [1732](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock`
[attribute](#)), [810](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock`
[attribute](#)), [815](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock`
[attribute](#)), [820](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.HeaderBlock.HeaderBlock.HeaderBlock`
[attribute](#)), [823](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock.LayoutBlock`
[attribute](#)), [825](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.MIDIBlock.MIDIBlock.MIDIBlock`
[attribute](#)), [837](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock`
[attribute](#)), [838](#)
[is_formatted_when_empty\(\)](#) (`abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock`
[attribute](#)), [840](#)
[is_full\(\)](#) (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure`
[attribute](#)), [905](#)
[is_full\(\)](#) (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure`
[attribute](#)), [916](#)
[is_full\(\)](#) (`abjad.tools.measuretools.Measure.Measure.Measure`
[attribute](#)), [926](#)
[is_harmonic_diatonic_interval_abbreviation\(\)](#) (in module `abjad.tools.pitchtools`.[is_harmonic_diatonic_interval_abbreviation\(\)](#),
[1196](#)
[is_imprecise\(\)](#) (`abjad.tools.contexttools.TempoMark.TempoMark.TempoMark`
[attribute](#)), [378](#)
[is_in_range\(\)](#) (`abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow`
[attribute](#)), [1737](#)
[is_integer_equivalent_expr\(\)](#) (in module `abjad.tools.mathtools`.[is_integer_equivalent_expr\(\)](#),
[1718](#)
[is_integer_equivalent_number\(\)](#) (in module `abjad.tools.mathtools`.[is_integer_equivalent_number\(\)](#),
[1718](#)
[is_invisible\(\)](#) (`abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet`
[attribute](#)), [1570](#)
[is_last_in_row\(\)](#) (`abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell`
[attribute](#)), [1732](#)
[is_lilypond_duration_name\(\)](#) (in module `abjad.tools.durationtools`.[is_lilypond_duration_name\(\)](#),
[1674](#)
[is_lilypond_duration_string\(\)](#) (in module `abjad.tools.durationtools`.[is_lilypond_duration_string\(\)](#),
[1674](#)
[is_lilypond_rest_string\(\)](#) (in module `abjad.tools.resttools`.[is_lilypond_rest_string\(\)](#),
[1674](#)
[is_lowercase_string\(\)](#) (in module `abjad.tools.iotools`.[is_lowercase_string\(\)](#),
[1674](#)
[is_melodic_diatonic_interval_abbreviation\(\)](#) (in module `abjad.tools.pitchtools`.[is_melodic_diatonic_interval_abbreviation\(\)](#),
[1196](#)
[is_monotonically_decreasing_sequence\(\)](#) (in module `abjad.tools.sequencetools`.[is_monotonically_decreasing_sequence\(\)](#),
[1783](#)
[is_monotonically_increasing_sequence\(\)](#) (in module `abjad.tools.sequencetools`.[is_monotonically_increasing_sequence\(\)](#),
[1784](#)

[is_named_chromatic_pitch_token\(\)](#) (in module `abjad.tools.pitchtools`), 1197
[is_named_chromatic_pitch_token\(\)](#) (in module `abjad.tools.pitchtools`), 1197
[is_negative_integer\(\)](#) (in module `abjad.tools.mathtools`), 1719
[is_neighbor_note\(\)](#) (in module `abjad.tools.tonalitytools`), 2107
[is_nonbinary](#) (`abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark` attribute), 388
[is_nonbinary](#) (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure` attribute), 906
[is_nonbinary](#) (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` attribute), 916
[is_nonbinary](#) (`abjad.tools.measuretools.Measure.Measure` attribute), 926
[is_nonbinary](#) (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet` attribute), 1565
[is_nonbinary](#) (`abjad.tools.tuplettools.Tuplet.Tuplet` attribute), 1577
[is_nonnegative_integer\(\)](#) (in module `abjad.tools.mathtools`), 1719
[is_nonnegative_integer_equivalent_number\(\)](#) (in module `abjad.tools.mathtools`), 1719
[is_nonnegative_integer_power_of_two\(\)](#) (in module `abjad.tools.mathtools`), 1720
[is_nonsemantic](#) (`abjad.tools.contexttools.Context.Context` attribute), 350
[is_nonsemantic](#) (`abjad.tools.lyricstools.Lyrics.Lyrics` attribute), 2046
[is_nonsemantic](#) (`abjad.tools.scoretools.GrandStaff.GrandStaff` attribute), 1251
[is_nonsemantic](#) (`abjad.tools.scoretools.PianoStaff.PianoStaff` attribute), 1272
[is_nonsemantic](#) (`abjad.tools.scoretools.Score.Score` attribute), 1282
[is_nonsemantic](#) (`abjad.tools.scoretools.StaffGroup.StaffGroup` attribute), 1293
[is_nonsemantic](#) (`abjad.tools.stafftools.RhythmicStaff.RhythmicStaff` attribute), 1523
[is_nonsemantic](#) (`abjad.tools.stafftools.Staff.Staff` attribute), 1533
[is_nonsemantic](#) (`abjad.tools.voicetools.Voice.Voice` attribute), 1601
[is_octave_tick_string\(\)](#) (in module `abjad.tools.pitchtools`), 1197
[is_orphan_component\(\)](#) (in module `abjad.tools.componenttools`), 1568
[is_overfull](#) (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure` attribute), 906
[is_overfull](#) (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` attribute), 916
[is_overfull](#) (`abjad.tools.measuretools.Measure.Measure` attribute), 926
[is_overlapped_by_interval\(\)](#) (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval` attribute), 1834
[is_overlapped_by_interval\(\)](#) (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin` attribute), 1838
[is_overlapped_by_interval\(\)](#) (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin` attribute), 1872
[is_parallel](#) (`abjad.tools.containertools.Cluster.Cluster` attribute), 295
[is_parallel](#) (`abjad.tools.containertools.Container.Container` attribute), 303
[is_parallel](#) (`abjad.tools.contexttools.Context.Context` attribute), 351
[is_parallel](#) (`abjad.tools.gracetools.Grace.Grace` attribute), 418
[is_parallel](#) (`abjad.tools.lyricstools.AddLyrics.AddLyrics` attribute), 2018
[is_parallel](#) (`abjad.tools.lyricstools.Lyrics.Lyrics` attribute), 2046
[is_parallel](#) (`abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure` attribute), 908
[is_parallel](#) (`abjad.tools.measuretools.DynamicMeasure.DynamicMeasure` attribute), 918
[is_parallel](#) (`abjad.tools.measuretools.Measure.Measure` attribute), 928
[is_parallel](#) (`abjad.tools.scoretools.GrandStaff.GrandStaff` attribute), 1252
[is_parallel](#) (`abjad.tools.scoretools.PianoStaff.PianoStaff` attribute), 1273
[is_parallel](#) (`abjad.tools.scoretools.Score.Score` attribute), 1283
[is_parallel](#) (`abjad.tools.scoretools.StaffGroup.StaffGroup` attribute), 1294
[is_parallel](#) (`abjad.tools.stafftools.RhythmicStaff.RhythmicStaff` attribute), 1524
[is_parallel](#) (`abjad.tools.stafftools.Staff.Staff` attribute), 1534
[is_parallel](#) (`abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet` attribute), 1568

[is_parallel \(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute\), 1580](#)
[is_parallel \(abjad.tools.voicetools.Voice.Voice.Voice attribute\), 1601](#)
[is_passing_tone\(\) \(in module abjad.tools.tonalitytools.is_passing_tone\), 2107](#)
[is_permutation\(\) \(in module abjad.tools.sequencetools.is_permutation\), 1785](#)
[is_pitch_carrier\(\) \(in module abjad.tools.pitchtools.is_pitch_carrier\), 1197](#)
[is_pitch_class_octave_number_string\(\) \(in module abjad.tools.pitchtools.is_pitch_class_octave_number_string\), 1197](#)
[is_pitch_class_unique \(abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet attribute\), 1112](#)
[is_positive_integer\(\) \(in module abjad.tools.mathtools.is_positive_integer\), 1720](#)
[is_positive_integer_equivalent_number\(\) \(in module abjad.tools.mathtools.is_positive_integer_equivalent_number\), 1720](#)
[is_positive_integer_power_of_two\(\) \(in module abjad.tools.mathtools.is_positive_integer_power_of_two\), 1721](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute\), 430](#)
[is_primary_instrument \(abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute\), 437](#)
[is_primary_instrument \(abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute\), 444](#)
[is_primary_instrument \(abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute\), 451](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BartoneSaxophone.BartoneSaxophone.BartoneSaxophone attribute\), 465](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BartoneVoice.BartoneVoice.BartoneVoice attribute\), 471](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute\), 479](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute\), 486](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute\), 514](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute\), 493](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute\), 500](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute\), 506](#)
[is_primary_instrument \(abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute\), 458](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Cello.Cello.Cello attribute\), 520](#)
[is_primary_instrument \(abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute\), 528](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute\), 534](#)
[is_primary_instrument \(abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute\), 542](#)
[is_primary_instrument \(abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute\), 549](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute\), 563](#)
[is_primary_instrument \(abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute\), 556](#)
[is_primary_instrument \(abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute\), 569](#)
[is_primary_instrument \(abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute\), 577](#)
[is_primary_instrument \(abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute\), 584](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Flute.Flute.Flute attribute\), 590](#)
[is_primary_instrument \(abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute\), 597](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute\), 604](#)
[is_primary_instrument \(abjad.tools.instrumenttools.Guitar.Guitar.Guitar attribute\), 611](#)

is_primary_instrument jad.tools.instrumenttools.Harp.Harp.Harp attribute), 619	(ab- is_primary_instrument jad.tools.instrumenttools.Violin.Violin.Violin attribute), 749	(ab-
is_primary_instrument jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 626	(ab- is_primary_instrument jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 756	(ab-
is_primary_instrument jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 637	(ab- is_proper_tuplet_multiplier() (in module ab- jad.tools.tuplettools.is_proper_tuplet_multiplier), 1590	
is_primary_instrument jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice attribute), 644	(ab- is_rectangular (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 726	
is_primary_instrument jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 652	(ab- is_repetition_free_sequence() (in module ab- jad.tools.sequencetools.is_repetition_free_sequence), 1785	
is_primary_instrument jad.tools.instrumenttools.Piano.Piano.Piano attribute), 659	(ab- is_restricted_growth_function() (in module ab- jad.tools.sequencetools.is_restricted_growth_function), 1785	
is_primary_instrument jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 666	is_secondary_instrument (ab- jad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 430	
is_primary_instrument jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone attribute), 673	is_secondary_instrument (ab- jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 367	
is_primary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 680	is_secondary_instrument (ab- jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 446	
is_primary_instrument jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 686	is_secondary_instrument (ab- jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 451	
is_primary_instrument jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 694	is_secondary_instrument (ab- jad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone attribute), 465	
is_primary_instrument jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 701	is_secondary_instrument (ab- jad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice attribute), 479	
is_primary_instrument jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 714	is_secondary_instrument (ab- jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 486	
is_primary_instrument jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 721	is_secondary_instrument (ab- jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 514	
is_primary_instrument jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion attribute), 728	is_secondary_instrument (ab- jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 493	
is_primary_instrument jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 735	is_secondary_instrument (ab- jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 500	
is_primary_instrument jad.tools.instrumenttools.Viola.Viola.Viola attribute), 742	is_secondary_instrument (ab- jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 507	

is_secondary_instrument jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet. attribute), 458	(ab- is_secondary_instrument jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 645	(ab- is_secondary_instrument jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice. attribute), 645
is_secondary_instrument jad.tools.instrumenttools.Cello.Cello.Cello attribute), 521	(ab- is_secondary_instrument jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 652	(ab- is_secondary_instrument jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 652
is_secondary_instrument jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA. attribute), 528	(ab- is_secondary_instrument jad.tools.instrumenttools.Piano.Piano.Piano attribute), 659	(ab- is_secondary_instrument jad.tools.instrumenttools.Piano.Piano.Piano attribute), 659
is_secondary_instrument jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass. attribute), 535	(ab- is_secondary_instrument jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 666	(ab- is_secondary_instrument jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 666
is_secondary_instrument jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet. attribute), 542	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 673	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 673
is_secondary_instrument jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute. attribute), 549	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 680	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone. attribute), 680
is_secondary_instrument jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon. attribute), 563	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 687	(ab- is_secondary_instrument jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice. attribute), 687
is_secondary_instrument jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone. attribute), 556	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 694	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone. attribute), 694
is_secondary_instrument jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice. attribute), 570	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone. attribute), 701	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone. attribute), 701
is_secondary_instrument jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet. attribute), 577	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice. attribute), 708	(ab- is_secondary_instrument jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice. attribute), 708
is_secondary_instrument jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn. attribute), 584	(ab- is_secondary_instrument jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 715	(ab- is_secondary_instrument jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 715
is_secondary_instrument jad.tools.instrumenttools.Flute.Flute.Flute attribute), 591	(ab- is_secondary_instrument jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 722	(ab- is_secondary_instrument jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 722
is_secondary_instrument jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn. attribute), 597	(ab- is_secondary_instrument jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 729	(ab- is_secondary_instrument jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion. attribute), 729
is_secondary_instrument jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel. attribute), 605	(ab- is_secondary_instrument jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 736	(ab- is_secondary_instrument jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 736
is_secondary_instrument jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 612	(ab- is_secondary_instrument jad.tools.instrumenttools.Viola.Viola.Viola attribute), 743	(ab- is_secondary_instrument jad.tools.instrumenttools.Viola.Viola.Viola attribute), 743
is_secondary_instrument jad.tools.instrumenttools.Harp.Harp.Harp attribute), 619	(ab- is_secondary_instrument jad.tools.instrumenttools.Violin.Violin.Violin attribute), 750	(ab- is_secondary_instrument jad.tools.instrumenttools.Violin.Violin.Violin attribute), 750
is_secondary_instrument jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord. attribute), 626	(ab- is_secondary_instrument jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 757	(ab- is_secondary_instrument jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 757
is_secondary_instrument jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 638	(ab- is_semantic (abjad.tools.contexttools.Context.Context.Context attribute), 349	(ab- is_semantic (abjad.tools.contexttools.Context.Context.Context attribute), 349

is_semantic (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2044
 is_semantic (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1250
 is_semantic (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1270
 is_semantic (abjad.tools.scoretools.Score.Score.Score attribute), 1281
 is_semantic (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1291
 is_semantic (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1521
 is_semantic (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1532
 is_semantic (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1599
 is_space_delimited_lowercase_string() (in module abjad.tools.iotools.is_space_delimited_lowercase_string), 1689
 is_strictly_decreasing_sequence() (in module abjad.tools.sequencetools.is_strictly_decreasing_sequence), 1786
 is_strictly_increasing_sequence() (in module abjad.tools.sequencetools.is_strictly_increasing_sequence), 1787
 is_symbolic_accidental_string() (in module abjad.tools.pitchtools.is_symbolic_accidental_string), 1197
 is_symbolic_pitch_range_string() (in module abjad.tools.pitchtools.is_symbolic_pitch_range_string), 1198
 is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval method), 1841
 is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1834
 is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1838
 is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree method), 1852
 is_tangent_to_interval() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1872
 is_tempo_mark_token() (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark attribute), 380
 is_tertian (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassSegment attribute), 1047
 is_tie_chain() (in module abjad.tools.tietools.is_tie_chain), 1554
 is_tie_chain_with_all_leaves_in_same_parent() (in module abjad.tools.tietools.is_tie_chain_with_all_leaves_in_same_parent), 1554
 is_time_signature_with_equivalent_binary_representation() (in module abjad.tools.timesignaturetools.is_time_signature_with_equivalent_binary_representation), 1894
 is_transposed_subset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140
 is_transposed_superset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140
 is_transposing (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 430
 is_transposing (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 437
 is_transposing (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 444
 is_transposing (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 451
 is_transposing (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone attribute), 465
 is_transposing (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice attribute), 472
 is_transposing (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 479
 is_transposing (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 486
 is_transposing (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 514
 is_transposing (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 493
 is_transposing (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 509
 is_transposing (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 507
 is_transposing (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 458
 is_transposing (abjad.tools.instrumenttools.Cello.Cello.Cello attribute), 529
 is_transposing (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 528
 is_transposing (abjad.tools.instrumenttools.Contrabass.Contrabass attribute), 535
 is_transposing (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet attribute), 542
 is_transposing (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute attribute), 549
 is_transposing (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon attribute), 563
 is_transposing (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone attribute), 556

is_transposing (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 570
 is_transposing (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 577
 is_transposing (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 584
 is_transposing (abjad.tools.instrumenttools.Flute.Flute.Flute.is_underfull (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 591
 is_transposing (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.abjad.tools.measuretools.Measure.Measure.Measure attribute), 598
 is_transposing (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.is_underfull (abjad.tools.measuretools.Measure.Measure.Measure attribute), 605
 is_transposing (abjad.tools.instrumenttools.Guitar.Guitar.Guitar.is_underscore_delimited_lowercase_file_name() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_file_name), 612
 is_transposing (abjad.tools.instrumenttools.Harp.Harp.Harp.is_underscore_delimited_lowercase_file_name_with_extension() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_file_name), 619
 is_transposing (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.is_underscore_delimited_lowercase_file_name_with_extension() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_file_name), 626
 is_transposing (abjad.tools.instrumenttools.Marimba.Marimba.Marimba.is_underscore_delimited_lowercase_package_name() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_package_name), 638
 is_transposing (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.is_underscore_delimited_lowercase_package_name() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_package_name), 645
 is_transposing (abjad.tools.instrumenttools.Oboe.Oboe.Oboe.is_underscore_delimited_lowercase_string() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_string), 652
 is_transposing (abjad.tools.instrumenttools.Piano.Piano.Piano.is_underscore_delimited_lowercase_string() (in module abjad.tools.iotools.is_underscore_delimited_lowercase_string), 659
 is_transposing (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo.is_likely_melodic_diatonic_interval_in_chorale() (in module abjad.tools.iotools.is_likely_melodic_diatonic_interval_in_chorale), 666
 is_transposing (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.is_likely_melodic_diatonic_interval_in_chorale() (in module abjad.tools.iotools.is_likely_melodic_diatonic_interval_in_chorale), 673
 is_transposing (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.is_likely_melodic_diatonic_interval_in_chorale() (in module abjad.tools.iotools.is_likely_melodic_diatonic_interval_in_chorale), 680
 is_transposing (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.is_uppercase (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator attribute), 687
 is_transposing (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.is_uppercase (abjad.tools.tonalitytools.QualityIndicator.QualityIndicator.QualityIndicator attribute), 694
 is_transposing (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.is_valid_beatspan() (in module abjad.tools.componenttools.is_valid_beatspan), 701
 is_transposing (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice.is_valid_beatspan() (in module abjad.tools.componenttools.is_valid_beatspan), 708
 is_transposing (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet.is_valid_beatspan() (in module abjad.tools.componenttools.is_valid_beatspan), 715
 is_transposing (abjad.tools.instrumenttools.Tuba.Tuba.Tuba.is_disjoint() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 722
 is_transposing (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion.is_disjoint() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 729
 is_transposing (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone.is_disjoint() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 736
 is_transposing (abjad.tools.instrumenttools.Viola.Viola.Viola.is_disjoint() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.InversionEquivalentChromaticIntervalClassSet method), 743
 is_transposing (abjad.tools.instrumenttools.Violin.Violin.Violin.is_disjoint() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 750
 is_transposing (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone.is_disjoint() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 757

isdisjoint() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NaturalChromaticPitchClassSet method), 1105
 isdisjoint() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet.NaturalChromaticPitchSet method), 1113
 isdisjoint() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140
 isdisjoint() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.ChordClass method), 2075
 issubset() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1005
 issubset() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1019
 issubset() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1026
 issubset() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet method), 1036
 issubset() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1072
 issubset() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1090
 issubset() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1105
 issubset() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1113
 issubset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140
 issubset() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.ChordClass method), 2076
 issuperset() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet method), 1005
 issuperset() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet method), 1019
 issuperset() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet method), 1026
 issuperset() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet.InversionEquivalentChromaticIntervalSet method), 1036
 issuperset() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet method), 1072
 issuperset() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet method), 1090
 issuperset() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet method), 1105
 issuperset() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet method), 1113
 issuperset() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet method), 1140
 issuperset() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.ChordClass method), 2076
 items() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary method), 1625
 items() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph method), 1650
 items() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet.HarmonicChromaticIntervalClassSet method), 997

[318](#)
[412](#)
[413](#)
[782](#)
[783](#)
[784](#)
[951](#)
[952](#)
[1198](#)
[267](#)
[268](#)
[785](#)
[785](#)
[762](#)

[981](#)
[982](#)
[1772](#)
[1223](#)
[1223](#)
[1302](#)
[1302](#)
[609](#)
[1610](#)
[787](#)
[788](#)
[789](#)
[1789](#)
[1789](#)

`iterate_contexts_backward_in_expr()` (in module `abjad.tools.contexttools.iterate_contexts_backward_in_expr`),
`iterate_contexts_forward_in_expr()` (in module `abjad.tools.contexttools.iterate_contexts_forward_in_expr`),
`iterate_depth_first()` (`abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree` method),
`iterate_depth_first()` (`abjad.tools.sequencetools.Tree.Tree.Tree` method),
`iterate_leaf_pairs_forward_in_expr()` (in module `abjad.tools.leaftools.iterate_leaf_pairs_forward_in_expr`),
`iterate_leaves_backward_in_expr()` (in module `abjad.tools.leaftools.iterate_leaves_backward_in_expr`),
`iterate_leaves_forward_in_expr()` (in module `abjad.tools.leaftools.iterate_leaves_forward_in_expr`),
`iterate_measures_backward_in_expr()` (in module `abjad.tools.measuretools.iterate_measures_backward_in_expr`),
`iterate_measures_forward_in_expr()` (in module `abjad.tools.measuretools.iterate_measures_forward_in_expr`),
`iterate_named_chromatic_pitch_pairs_forward_in_expr()` (in module `abjad.tools.pitchtools.iterate_named_chromatic_pitch_pairs_forward_in_expr`),
`iterate_namesakes_backward_from_component()` (in module `abjad.tools.componenttools.iterate_namesakes_backward_from_component`),
`iterate_namesakes_forward_from_component()` (in module `abjad.tools.componenttools.iterate_namesakes_forward_from_component`),
`iterate_notes_and_chords_backward_in_expr()` (in module `abjad.tools.leaftools.iterate_notes_and_chords_backward_in_expr`),
`iterate_notes_and_chords_forward_in_expr()` (in module `abjad.tools.leaftools.iterate_notes_and_chords_forward_in_expr`),
`iterate_notes_and_chords_in_expr_outside_traditional_instrument_range()` (in module `abjad.tools.instrumenttools.iterate_notes_and_chords_in_expr_outside_traditional_instrument_range`),
`iterate_notes_backward_in_expr()` (in module `abjad.tools.notetools.iterate_notes_backward_in_expr`),

`iterate_notes_forward_in_expr()` (in module `abjad.tools.notetools.iterate_notes_forward_in_expr`),
`iterate_payload()` (`abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree` method),
`iterate_payload()` (`abjad.tools.sequencetools.Tree.Tree.Tree` method),
`iterate_rests_backward_in_expr()` (in module `abjad.tools.resttools.iterate_rests_backward_in_expr`),
`iterate_rests_forward_in_expr()` (in module `abjad.tools.resttools.iterate_rests_forward_in_expr`),
`iterate_scores_backward_in_expr()` (in module `abjad.tools.scoretools.iterate_scores_backward_in_expr`),
`iterate_scores_forward_in_expr()` (in module `abjad.tools.scoretools.iterate_scores_forward_in_expr`),
`iterate_semantic_voices_backward_in_expr()` (in module `abjad.tools.voicetools.iterate_semantic_voices_backward_in_expr`),
`iterate_semantic_voices_forward_in_expr()` (in module `abjad.tools.voicetools.iterate_semantic_voices_forward_in_expr`),
`iterate_sequence_cyclically()` (in module `abjad.tools.sequencetools.iterate_sequence_cyclically`),
`iterate_sequence_cyclically_from_start_to_stop()` (in module `abjad.tools.sequencetools.iterate_sequence_cyclically_from_start_to_stop`),
`iterate_sequence_forward_and_backward_nonoverlapping()` (in module `abjad.tools.sequencetools.iterate_sequence_forward_and_backward_nonoverlapping`),
`iterate_sequence_forward_and_backward_overlapping()` (in module `abjad.tools.sequencetools.iterate_sequence_forward_and_backward_overlapping`),
`iterate_sequence_nwise_cyclic()` (in module `abjad.tools.sequencetools.iterate_sequence_nwise_cyclic`),
`iterate_sequence_nwise_strict()` (in module `abjad.tools.sequencetools.iterate_sequence_nwise_strict`),
`iterate_sequence_nwise_wrapped()` (in module `abjad.tools.sequencetools.iterate_sequence_nwise_wrapped`),
`iterate_sequence_pairwise_cyclic()` (in module `abjad.tools.sequencetools.iterate_sequence_pairwise_cyclic`),

[1790](#)
[iterate_sequence_pairwise_strict\(\)](#) (in module [abjad.tools.sequencetools](#).[iterate_sequence_pairwise_strict\(\)](#)), [1790](#)
[iterate_sequence_pairwise_wrapped\(\)](#) (in module [abjad.tools.sequencetools](#).[iterate_sequence_pairwise_wrapped\(\)](#)), [1790](#)
[iterate_skips_backward_in_expr\(\)](#) (in module [abjad.tools.skiptools](#).[iterate_skips_backward_in_expr\(\)](#)), [1313](#)
[iterate_skips_forward_in_expr\(\)](#) (in module [abjad.tools.skiptools](#).[iterate_skips_forward_in_expr\(\)](#)), [1313](#)
[iterate_staves_backward_in_expr\(\)](#) (in module [abjad.tools.stafftools](#).[iterate_staves_backward_in_expr\(\)](#)), [1541](#)
[iterate_staves_forward_in_expr\(\)](#) (in module [abjad.tools.stafftools](#).[iterate_staves_forward_in_expr\(\)](#)), [1542](#)
[iterate_thread_backward_from_component\(\)](#) (in module [abjad.tools.threadtools](#).[iterate_thread_backward_from_component\(\)](#)), [1826](#)
[iterate_thread_backward_in_expr\(\)](#) (in module [abjad.tools.threadtools](#).[iterate_thread_backward_in_expr\(\)](#)), [1828](#)
[iterate_thread_forward_from_component\(\)](#) (in module [abjad.tools.threadtools](#).[iterate_thread_forward_from_component\(\)](#)), [1828](#)
[iterate_thread_forward_in_expr\(\)](#) (in module [abjad.tools.threadtools](#).[iterate_thread_forward_in_expr\(\)](#)), [1830](#)
[iterate_tie_chains_backward_in_expr\(\)](#) (in module [abjad.tools.tietools](#).[iterate_tie_chains_backward_in_expr\(\)](#)), [1554](#)
[iterate_tie_chains_forward_in_expr\(\)](#) (in module [abjad.tools.tietools](#).[iterate_tie_chains_forward_in_expr\(\)](#)), [1555](#)
[iterate_timeline_backward_from_component\(\)](#) (in module [abjad.tools.componenttools](#).[iterate_timeline_backward_from_component\(\)](#)), [269](#)
[iterate_timeline_backward_in_expr\(\)](#) (in module [abjad.tools.componenttools](#).[iterate_timeline_backward_in_expr\(\)](#)), [270](#)
[iterate_timeline_forward_from_component\(\)](#) (in module [abjad.tools.componenttools](#).[iterate_timeline_forward_from_component\(\)](#)), [271](#)
[iterate_timeline_forward_in_expr\(\)](#) (in module [abjad.tools.componenttools](#).[iterate_timeline_forward_in_expr\(\)](#)), [271](#)
[iterate_topmost_tie_chains_and_components_forward_in_expr\(\)](#) (in module [abjad.tools.tietools](#).[iterate_topmost_tie_chains_and_components_forward_in_expr\(\)](#)), [1556](#)
[iterate_tuplets_backward_in_expr\(\)](#) (in module [abjad.tools.tuplettools](#).[iterate_tuplets_backward_in_expr\(\)](#)), [1591](#)
[iterate_tuplets_forward_in_expr\(\)](#) (in module [abjad.tools.tuplettools](#).[iterate_tuplets_forward_in_expr\(\)](#)), [1591](#)
[iterate_vertical_moments_backward_in_expr\(\)](#) (in module [abjad.tools.verticalitytools](#).[iterate_vertical_moments_backward_in_expr\(\)](#)), [1946](#)
[iterate_vertical_moments_forward_in_expr\(\)](#) (in module [abjad.tools.verticalitytools](#).[iterate_vertical_moments_forward_in_expr\(\)](#)), [1947](#)
[iterate_voices_backward_in_expr\(\)](#) (in module [abjad.tools.voicetools](#).[iterate_voices_backward_in_expr\(\)](#)), [1611](#)
[iterate_voices_forward_in_expr\(\)](#) (in module [abjad.tools.voicetools](#).[iterate_voices_forward_in_expr\(\)](#)), [1611](#)
[iteritems\(\)](#) ([abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver](#) attribute), [1996](#)
[iteritems\(\)](#) ([abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver](#) attribute), [2014](#)
[iteritems\(\)](#) ([abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary](#) method), [1625](#)
[iteritems\(\)](#) ([abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph](#) method), [1650](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector](#) method), [997](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector](#) method), [1040](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector](#) method), [1051](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector](#) method), [1064](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector](#) method), [117](#)
[iteritems\(\)](#) ([abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector](#) method), [1145](#)
[iteritems\(\)](#) ([abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree](#) method), [2063](#)
[iteritems\(\)](#) ([abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup](#) method), [2068](#)
[iteritems\(\)](#) ([abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval](#) method), [1841](#)
[iteritems\(\)](#) ([abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary](#) method), [1872](#)
[iterkeys\(\)](#) ([abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary](#) method), [1625](#)

Index

[jad.tools.leaftools.label_leaves_in_expr_with_inversion_equivalent_chromatic_interval_classes\(\)](#),
[786](#) [label_leaves_in_expr_with_written_leaf_duration\(\)](#)
[label_leaves_in_expr_with_leaf_depth\(\)](#) (in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_leaf_depth\(\)](#), [jad.tools.leaftools.label_leaves_in_expr_with_written_leaf_durati](#)
[786](#) [792](#)
[label_leaves_in_expr_with_leaf_durations\(\)](#) [label_notes_in_expr_with_note_indices\(\)](#) (in module [ab-](#)
(in module [ab-](#) [jad.tools.notetools.label_notes_in_expr_with_note_indices\(\)](#),
[jad.tools.leaftools.label_leaves_in_expr_with_leaf_durations\(\)](#),
[787](#) [983](#)
[label_leaves_in_expr_with_leaf_indices\(\)](#) (in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_leaf_indices\(\)](#), [jad.tools.tietools.label_tie_chains_in_expr_with_prolated_tie_cha](#)
[787](#) [1556](#)
[label_leaves_in_expr_with_leaf_numbers\(\)](#) [label_tie_chains_in_expr_with_tie_chain_durations\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_leaf_numbers\(\)](#), [jad.tools.tietools.label_tie_chains_in_expr_with_tie_chain_durati](#)
[787](#) [1557](#)
[label_leaves_in_expr_with_melodic_chromatic_interval_classes\(\)](#) [label_tie_chains_in_expr_with_written_tie_chain_duration\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_chromatic_interval_classes\(\)](#), [jad.tools.tietools.label_tie_chains_in_expr_with_written_tie_chai](#)
[788](#) [1557](#)
[label_leaves_in_expr_with_melodic_chromatic_intervals\(\)](#) [label_vertical_moments_in_expr_with_chromatic_interval_classes\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_chromatic_intervals\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_c](#)
[788](#) [1948](#)
[label_leaves_in_expr_with_melodic_counterpoint_interval_classes\(\)](#) [label_vertical_moments_in_expr_with_chromatic_intervals\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_counterpoint_interval_classes\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_c](#)
[789](#) [1949](#)
[label_leaves_in_expr_with_melodic_counterpoint_intervals\(\)](#) [label_vertical_moments_in_expr_with_counterpoint_intervals\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_counterpoint_intervals\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_c](#)
[789](#) [1949](#)
[label_leaves_in_expr_with_melodic_diatonic_interval_classes\(\)](#) [label_vertical_moments_in_expr_with_diatonic_intervals\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_diatonic_interval_classes\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_d](#)
[790](#) [1950](#)
[label_leaves_in_expr_with_melodic_diatonic_intervals\(\)](#) [label_vertical_moments_in_expr_with_interval_class_vectors\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_melodic_diatonic_intervals\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_in](#)
[790](#) [1951](#)
[label_leaves_in_expr_with_pitch_class_numbers\(\)](#) [label_vertical_moments_in_expr_with_numbered_chromatic_pitch_classes](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_pitch_class_numbers\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_n](#)
[790](#) [1951](#)
[label_leaves_in_expr_with_pitch_numbers\(\)](#) [label_vertical_moments_in_expr_with_pitch_numbers\(\)](#)
(in module [ab-](#) (in module [ab-](#)
[jad.tools.leaftools.label_leaves_in_expr_with_pitch_numbers\(\)](#), [jad.tools.verticalitytools.label_vertical_moments_in_expr_with_p](#)
[791](#) [1952](#)
[label_leaves_in_expr_with_prolated_leaf_duration\(\)](#) [latest_start](#) ([abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.Time](#)
(in module [ab-](#) [attribute](#)), [1832](#)
[jad.tools.leaftools.label_leaves_in_expr_with_prolated_leaf_duration\(\)](#), [abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTre](#)
[791](#) [attribute](#)), [1846](#)
[label_leaves_in_expr_with_tuplet_depth\(\)](#) (in module [ab-](#) [latest_start](#) ([abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeI](#)
[jad.tools.leaftools.label_leaves_in_expr_with_tuplet_depth\(\)](#), [attribute](#)), [1860](#)

latest_stop (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.attribute), 1832

latest_stop (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.attribute), 1846

latest_stop (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.attribute), 1860

LayoutBlock (class in abjad.tools.lilypondfiletools.LayoutBlock.LayoutBlock), 824

leading_tone (abjad.tools.tonalitytools.Scale.Scale.Scale.attribute), 2094

Leaf (class in abjad.tools.leaftools.Leaf.Leaf), 768

leaf_index (abjad.tools.chordtools.Chord.Chord.Chord.attribute), 224

leaf_index (abjad.tools.leaftools.Leaf.Leaf.Leaf.attribute), 768

leaf_index (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender.attribute), 2025

leaf_index (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen.attribute), 2030

leaf_index (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace.attribute), 2034

leaf_index (abjad.tools.lyricstools.LyricText.LyricText.LyricText.attribute), 2039

leaf_index (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic.attribute), 966

leaf_index (abjad.tools.notetools.Note.Note.Note.attribute), 972

leaf_index (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.attribute), 1214

leaf_index (abjad.tools.resttools.Rest.Rest.Rest.attribute), 1218

leaf_index (abjad.tools.skiptools.Skip.Skip.Skip.attribute), 1308

leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration (in module abjad.tools.leaftools.leaf_to_augmented_tuplet_with_n_notes_of_equal_written_duration), 792

leaf_to_augmented_tuplet_with_proportions() (in module abjad.tools.leaftools.leaf_to_augmented_tuplet_with_proportions()), 793

leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration (in module abjad.tools.leaftools.leaf_to_diminished_tuplet_with_n_notes_of_equal_written_duration), 793

leaf_to_diminished_tuplet_with_proportions() (in module abjad.tools.leaftools.leaf_to_diminished_tuplet_with_proportions()), 794

least_common_multiple() (in module abjad.tools.mathtools.least_common_multiple), 1721

least_multiple_greater_equal() (in module abjad.tools.mathtools.least_multiple_greater_equal), 1722

leaves (abjad.tools.containertools.Cluster.Cluster.Cluster.attribute), 294

leaves (abjad.tools.containertools.Container.Container.Container.attribute), 302

leaves (abjad.tools.contexttools.Context.Context.Context.attribute), 349

leaves (abjad.tools.gracetools.Grace.Grace.Grace.attribute), 416

leaves (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics.attribute), 2017

leaves (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics.attribute), 2044

leaves (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.attribute), 906

leaves (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure.attribute), 917

leaves (abjad.tools.measuretools.Measure.Measure.Measure.attribute), 927

leaves (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff.attribute), 1250

leaves (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff.attribute), 1271

leaves (abjad.tools.scoretools.Score.Score.Score.attribute), 1281

leaves (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup.attribute), 1291

leaves (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.attribute), 1317

leaves (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.attribute), 1325

leaves (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.attribute), 1333

leaves (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.attribute), 1342

leaves (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.attribute), 1352

leaves (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.attribute), 1362

leaves (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.attribute), 1372

leaves (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.attribute), 1380

leaves (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.attribute), 1388

leaves (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.attribute), 1397

leaves (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.attribute), 1404

leaves (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 1412
 leaves (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner.LilyPondComment.LilyPondComment), attribute), 1421
 leaves (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.LilyPondFile.LilyPondFile), attribute), 1430
 leaves (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.LilyPondLanguageToken.LilyPondLanguageToken), attribute), 1438
 leaves (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.LilyPondLanguageToken.LilyPondLanguageToken), attribute), 1446
 leaves (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.LilyPondParser.LilyPondParser), attribute), 1453
 leaves (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1461
 leaves (abjad.tools.spannertools.Spanner.Spanner.Spanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1468
 leaves (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1475
 leaves (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1483
 leaves (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1490
 leaves (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1498
 leaves (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.LilyPondVersionToken.LilyPondVersionToken), attribute), 1521
 leaves (abjad.tools.stafftools.Staff.Staff.Staff.LilyPondVersionToken.LilyPondVersionToken), attribute), 1532
 leaves (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.LilyPondVersionToken.LilyPondVersionToken), attribute), 1544
 leaves (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.LilyPondVersionToken.LilyPondVersionToken), attribute), 1566
 leaves (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet.LilyPondVersionToken.LilyPondVersionToken), attribute), 1577
 leaves (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment.LilyPondVersionToken.LilyPondVersionToken), attribute), 1942
 leaves (abjad.tools.voicetools.Voice.Voice.Voice.LilyPondVersionToken.LilyPondVersionToken), attribute), 1599
 level (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree.LilyPondVersionToken.LilyPondVersionToken), attribute), 1752
 level (abjad.tools.sequencetools.Tree.Tree.Tree.LilyPondVersionToken.LilyPondVersionToken), attribute), 1768
 likely_instruments_based_on_performer_name (abjad.tools.scoretools.Performer.Performer.Performer.LilyPondVersionToken.LilyPondVersionToken), attribute), 1261
 lilypond_duration_string_to_rational() (in module abjad.tools.durationtools.lilypond_duration_string_to_rational), 1675
 lilypond_duration_string_to_rational_list() (in module abjad.tools.durationtools.lilypond_duration_string_to_rational_list), 1675
 LilyPondCommandMark (class in abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark), 273

list_instrument_names()	(in module abjad.tools.instrumenttools.list_instrument_names),	list_prolated_durations_of_leaves_in_expr()	(in module abjad.tools.leaftools.list_prolated_durations_of_leaves_in_expr),
762		794	
list_instruments()	(in module abjad.tools.instrumenttools.list_instruments),	list_secondary_instruments()	(in module abjad.tools.instrumenttools.list_secondary_instruments),
763			
list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers()	(in module abjad.tools.pitchtools.list_inversion_equivalent_chromatic_interval_classes_pairwise_between_pitch_carriers),		
1200		list_time_signatures_of_measures_in_expr()	(in module abjad.tools.measuretools.list_time_signatures_of_measures_in_expr),
		953	
list_leftmost_components_with_prolated_duration_at_most()	(in module abjad.tools.componenttools.list_leftmost_components_with_prolated_duration_at_most),	list_unordered_named_chromatic_pitch_pairs_in_expr()	(in module abjad.tools.pitchtools.list_unordered_named_chromatic_pitch_pairs_in_expr),
273			
list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers()	(in module abjad.tools.pitchtools.list_melodic_chromatic_interval_numbers_pairwise_between_pitch_carriers),		
1201		list_written_durations_of_leaves_in_expr()	(in module abjad.tools.leaftools.list_written_durations_of_leaves_in_expr),
		594	
list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class()	(in module abjad.tools.pitchtools.list_named_chromatic_pitch_carriers_in_expr_sorted_by_numbered_chromatic_pitch_class),		
1202		local_maxima (abjad.tools.pitchtools.NamedChromaticPitchSegment.Name	local_minima (abjad.tools.pitchtools.NamedChromaticPitchSegment.Name
		attribute), 1109	
list_named_chromatic_pitches_in_expr()	(in module abjad.tools.pitchtools.list_named_chromatic_pitches_in_expr),	log (in module abjad.tools.iotools.log),	1690
1202		lone (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner	attribute), 1334
list_nonspanning_subarrays_of_pitch_array()	(in module abjad.tools.pitcharraytools.list_nonspanning_subarrays_of_pitch_array),		
1740		lone (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner	attribute), 1364
		lone (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner	attribute), 1414
list_numbered_chromatic_pitch_classes_in_expr()	(in module abjad.tools.pitchtools.list_numbered_chromatic_pitch_classes_in_expr),	ly() (in module abjad.tools.iotools.ly),	1691
1202		lyric_extender (in module abjad.tools.lyrictools.LyricExtender.LyricExtender),	
		2025	
list_octave_transpositions_of_pitch_carrier_within_pitch_range()	(in module abjad.tools.pitchtools.list_octave_transpositions_of_pitch_carrier_within_pitch_range),	LyricHyphen (class in abjad.tools.lyrictools.LyricHyphen.LyricHyphen),	
1203		2029	
list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2()	(in module abjad.tools.pitchtools.list_ordered_named_chromatic_pitch_pairs_from_expr_1_to_expr_2),	LyricSpace (class in abjad.tools.lyrictools.LyricSpace.LyricSpace),	
1203		2034	
list_package_dependency_versions()	(in module abjad.tools.configurationtools.list_package_dependency_versions),	(class in abjad.tools.lyrictools.LyricText.LyricText),	
1620		2038	
list_performer_names()	(in module abjad.tools.scoretools.list_performer_names),		
1303			
list_primary_instruments()	(in module abjad.tools.instrumenttools.list_primary_instruments),	make_accelerating_notes_with_lilypond_multipliers()	(in module abjad.tools.notetools.make_accelerating_notes_with_lilypond_multipliers),
764		984	
list_primary_performer_names()	(in module abjad.tools.scoretools.list_primary_performer_names),	make_all_notes_in_ascending_and_descending_diatonic_scale()	(in module abjad.tools.tonalitytools.make_all_notes_in_ascending_and_descending_diatonic_scale),
1303			

M

2108
make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots() (in module ab-
(in module ab- jad.tools.resttools.make_multi_measure_rests),
jad.tools.tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_avoid_dots),
1592 make_n_middle_c_centered_pitches() (in module ab-
make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots() jad.tools.pitchtools.make_n_middle_c_centered_pitches),
(in module ab- 1203
jad.tools.tuplettools.make_augmented_tuplet_from_duration_and_proportions_and_encourage_dots), ab-
1593 jad.tools.notetools.make_notes), 984
make_basic_lilypond_file() (in module ab- make_notes_with_multiplied_durations() (in module ab-
jad.tools.lilypondfiletools.make_basic_lilypond_file), jad.tools.notetools.make_notes_with_multiplied_durations),
843 985
make_big_centered_page_number_markup() make_percussion_note() (in module ab-
(in module ab- jad.tools.notetools.make_percussion_note),
jad.tools.markuptools.make_big_centered_page_number_markup),
902 make_performer_name_instrument_dictionary() (ab-
make_covered_spanner_schema() (in module ab- jad.tools.scoretools.Performer.Performer.Performer
jad.tools.spannertools.make_covered_spanner_schema), method), 1262
1515 make_piano_score_from_leaves() (in module ab-
make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots() jad.tools.scoretools.make_piano_score_from_leaves),
(in module ab- 1305
jad.tools.tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_avoid_dots) (in module ab-
1593 jad.tools.scoretools.make_piano_sketch_score_from_leaves),
make_diminished_tuplet_from_duration_and_proportions_and_encourage_dots() 1305
(in module ab- make_pitch_array_score_from_pitch_arrays()
jad.tools.tuplettools.make_diminished_tuplet_from_duration_and_proportions_and_encourage_dots), ab-
1594 jad.tools.scoretools.make_pitch_array_score_from_pitch_arrays),
make_dynamic_spanner_below_with_nib_at_right() 1306
(in module ab- make_polyphonic_percussion_score_from_nonoverlapping_trees()
jad.tools.spannertools.make_dynamic_spanner_below_with_nib_at_right), module ab-
1516 jad.tools.timeintervaltools.make_polyphonic_percussion_score_fr
make_empty_piano_score() (in module ab- 1888
jad.tools.scoretools.make_empty_piano_score), make_populated_pitch_array_from_list_of_pitch_lists()
1304 (in module ab-
make_empty_pitch_array_from_list_of_pitch_lists() jad.tools.pitcharraytools.make_populated_pitch_array_from_list_
(in module ab- 1742
jad.tools.pitcharraytools.make_empty_pitch_array_from_list_of_pitch_lists), lilypond_multipliers()
1741 (in module ab-
make_first_n_notes_in_ascending_diatonic_scale() jad.tools.notetools.make_quarter_notes_with_lilypond_multiplier
(in module ab- 986
jad.tools.tonalitytools.make_first_n_notes_in_ascending_diatonic_scale), (in module ab-
2109 jad.tools.notetools.make_repeated_notes),
make_leaves() (in module ab- 986
jad.tools.leaftools.make_leaves), 795 make_repeated_notes_from_time_signature()
make_leaves_from_note_value_signal() (in module ab- (in module ab-
jad.tools.leaftools.make_leaves_from_note_value_signal), jad.tools.notetools.make_repeated_notes_from_time_signature),
796 987
make_measures_with_full_measure_spacer_skips() make_repeated_notes_from_time_signatures()
(in module ab- (in module ab-
jad.tools.measuretools.make_measures_with_full_measure_spacer_skips), jad.tools.timeintervaltools.make_repeated_notes_from_time_signatures),
953 987
make_monophonic_percussion_score_from_nonoverlapping_intervals() make_repeated_notes_with_shorter_notes_at_end()
(in module ab- (in module ab-
jad.tools.timeintervaltools.make_monophonic_percussion_score_from_nonoverlapping_intervals), jad.tools.timeintervaltools.make_repeated_notes_with_shorter_notes_at_

987
make_repeated_rests_from_time_signature()
(in module abjad.tools.resttools.make_repeated_rests_from_time_signature), 1224
make_repeated_rests_from_time_signatures()
(in module abjad.tools.resttools.make_repeated_rests_from_time_signatures), 1224
make_repeated_skips_from_time_signature()
(in module abjad.tools.skiptools.make_repeated_skips_from_time_signature), 1314
make_repeated_skips_from_time_signatures()
(in module abjad.tools.skiptools.make_repeated_skips_from_time_signatures), 1314
make_rests()
(in module abjad.tools.resttools.make_rests), 1224
make_rhythmic_sketch_staff()
(in module abjad.tools.stafftools.make_rhythmic_sketch_staff), 1542
make_skips_with_multiplied_durations()
(in module abjad.tools.skiptools.make_skips_with_multiplied_durations), 1314
make_solid_text_spanner_above_with_nib_at_right()
(in module abjad.tools.spannertools.make_solid_text_spanner_above_with_nib_at_right), 1517
make_solid_text_spanner_below_with_nib_at_right()
(in module abjad.tools.spannertools.make_solid_text_spanner_below_with_nib_at_right), 1517
make_spacing_vector()
(in module abjad.tools.layouttools.make_spacing_vector), 1700
make_spanner_schema()
(in module abjad.tools.spannertools.make_spanner_schema), 1518
make_tuplet_from_proportions_and_pair()
(in module abjad.tools.tuplettools.make_tuplet_from_proportions_and_pair), 1594
map_sequence_elements_to_canonic_tuples()
(in module abjad.tools.sequencetools.map_sequence_elements_to_canonic_tuples), 1791
map_sequence_elements_to_numbered_sublists()
(in module abjad.tools.sequencetools.map_sequence_elements_to_numbered_sublists), 1792
Marimba
(class in abjad.tools.instrumenttools.Marimba.Marimba), 636
mark
(abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner attribute), 1373
Mark
(class in abjad.tools.marktools.Mark.Mark), 867
marks
(abjad.tools.chordtools.Chord.Chord.Chord attribute), 224
marks
(abjad.tools.componenttools.Component.Component.Component attribute), 236
marks
(abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 294
marks
(abjad.tools.containertools.Container.Container.Container attribute), 302
marks
(abjad.tools.contexttools.Context.Context.Context attribute), 349
marks
(abjad.tools.gracetools.Grace.Grace.Grace attribute), 417
marks
(abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 768
marks
(abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics attribute), 2017
marks
(abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender attribute), 2025
marks
(abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2030
marks
(abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics attribute), 2044
marks
(abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace attribute), 2034
marks
(abjad.tools.lyrictools.LyricText.LyricText.LyricText attribute), 2039
marks
(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 906
marks
(abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 917
marks
(abjad.tools.measuretools.Measure.Measure.Measure attribute), 927
marks
(abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 966
marks
(abjad.tools.notetools.Note.Note.Note attribute), 972
marks
(abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1214
marks
(abjad.tools.resttools.Rest.Rest.Rest attribute), 1218
marks
(abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1250
marks
(abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1271
marks
(abjad.tools.scoretools.Score.Score.Score attribute), 1281
marks
(abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1292
marks
(abjad.tools.skiptools.Skip.Skip.Skip attribute), 1309

marks (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.melodic_interval (ab-
attribute), 1522
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval

marks (abjad.tools.stafftools.Staff.Staff.Staff attribute), attribute), 1081
1532
melodic_chromatic_interval_class (ab-

marks (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.melodic_chromatic_interval (ab-
attribute), 1566
jad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval

marks (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet melodic_chromatic_interval_class_segment (ab-
attribute), 1577
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment

marks (abjad.tools.voicetools.Voice.Voice.Voice at- attribute), 1068
tribute), 1600
melodic_chromatic_interval_class_segment (ab-

markup (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClassjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChrom
attribute), 2075
attribute), 1109

markup (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunctionmelodic_interval_class_vector (ab-
attribute), 2102
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment

Markup (class in ab- attribute), 1068
jad.tools.markuptools.Markup.Markup),
889
melodic_chromatic_interval_numbers (ab-
jad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment

markup_name (abjad.tools.markuptools.Markup.Markup.Markup attribute), 1068
attribute), 891
melodic_chromatic_interval_numbers (ab-

MarkupCommand (class in ab- jad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChrom
jad.tools.markuptools.MarkupCommand.MarkupCommand)attribute), 1071
893
melodic_chromatic_interval_segment (ab-

MarkupInventory (class in ab- jad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicD
jad.tools.markuptools.MarkupInventory.MarkupInventory), attribute), 1023
896
melodic_chromatic_interval_segment (ab-

mask_intervals_with_intervals() (in module ab- jad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDia
jad.tools.timeintervaltools.mask_intervals_with_intervals), attribute), 1086
1888
melodic_chromatic_interval_segment (ab-

matches_cell() (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCellNamedChromaticPitchSegment.NamedChrom
method), 1733
attribute), 1109

Matrix (class in abjad.tools.sequencetools.Matrix.Matrix), melodic_chromatic_interval_segment (ab-
1763
jad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator

max_index (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint.AbsoluteIndexConstraint
attribute), 1991
melodic_chromatic_interval_set (ab-

Measure (class in ab- jad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicI
jad.tools.measuretools.Measure.Measure),
925
attribute), 1090
melodic_chromatic_intervals (ab-

measure_number (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasuremelodic_chromatic_interval_set (ab-
attribute), 906
attribute), 1071
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval

measure_number (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 917
jad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval

measure_number (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1081
attribute), 927
melodic_counterpoint_interval_class (ab-

measure_to_one_line_input_string() (in module ab- jad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounter
jad.tools.measuretools.measure_to_one_line_input_string), attribute), 1076
954
melodic_diatonic_interval_ascending (ab-

MeasuredComplexBeamSpanner (class in ab- jad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicI
jad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner),
1411
attribute), 1081
melodic_diatonic_interval_class (ab-

measures (abjad.tools.verticalitytools.VerticalMoment.VerticalMomentVerticalMomentjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicI
attribute), 1942
attribute), 1081

mediant (abjad.tools.tonalitytools.Scale.Scale.Scale at- melodic_diatonic_interval_class_segment (ab-
tribute), 2094
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment

melodic_diatonic_interval_descending	(abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet attribute), 1014	MelodicDiatonicIntervalSet (class in abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.MelodicDiatonicIntervalSet), 1089
melodic_diatonic_interval_numbers	(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet attribute), 1090	merge() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1090 meters (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1422
melodic_diatonic_interval_segment	(abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1023	method (abjad.tools.pitchtools.HarmonicDiatonicIntervalSegment.HarmonicDiatonicIntervalSegment attribute), 1639
melodic_diatonic_interval_segment	(abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment attribute), 1109	MetricGridSpanner (class in abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner attribute), 1420
melodic_diatonic_interval_segment	(abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 2080	MezzoSopranoVoice (class in abjad.tools.tonalitytools.ChordQualityIndicator.ChordQualityIndicator attribute), 643
melodic_diatonic_interval_segment	(abjad.tools.tonalitytools.Mode.Mode.Mode attribute), 2088	middle_c_position (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark attribute), 340
melodic_diatonic_intervals	(abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet attribute), 1090	MIDIBlock (class in abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet), 836
MelodicChromaticInterval	(class in abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval), 1054	millisecond_pitch_pairs_to_q_events() (in module abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval), 2071
MelodicChromaticIntervalClass	(class in abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass), 1057	milliseconds_to_q_events() (in module abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass), 2072
MelodicChromaticIntervalClassSegment	(class in abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment), 1059	minimal_page_breaking (abjad.tools.pitchtools.MelodicChromaticIntervalClassSegment.MelodicChromaticIntervalClassSegment attribute), 838
MelodicChromaticIntervalClassVector	(class in abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector), 1063	MisduratedMeasureCheck (class in abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector), 1966
MelodicChromaticIntervalSegment	(class in abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1067	MisfilledMeasureCheck (class in abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1969
MelodicChromaticIntervalSet	(class in abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet), 1071	MispitchedTieCheck (class in abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet), 1971
MelodicCounterpointInterval	(class in abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval), 1075	MisrepresentedFlagCheck (class in abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval), 1973
MelodicCounterpointIntervalClass	(class in abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass), 1078	MissingParentCheck (class in abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass), 1975
MelodicDiatonicInterval	(class in abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval), 1080	mode (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 370 Mode (class in abjad.tools.tonalitytools.Mode.Mode), 2088
MelodicDiatonicIntervalClass	(class in abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass), 1083	MelodicDiatonicIntervalClass (class in abjad.tools.tonalitytools.Mode.Mode.Mode attribute), 2088
MelodicDiatonicIntervalSegment	(class in abjad.tools.pitchtools.MelodicDiatonicIntervalSegment.MelodicDiatonicIntervalSegment), 1086	module_crawler (abjad.tools.documentationtools.APICrawler.APICrawler.APICrawler attribute), 1067

module_crawler (abjad.tools.documentationtools.ClassCrawler.ClassCrawler attribute), 1637
 module_crawler (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler attribute), 1644
 module_name (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1639
 module_name (abjad.tools.documentationtools.Documenter.Documenter attribute), 1642
 module_name (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter attribute), 1646
 ModuleCrawler (class in abjad.tools.lyrictools.LyricSpace.LyricSpace attribute), 2035
 modulo (abjad.tools.sievetools.ResidueClass.ResidueClass attribute), 1819
 most_likely_instrument_based_on_performer_name (abjad.tools.scoretools.Performer.Performer attribute), 1262
 move_component_subtree_to_right_in_immediate_parent_of_component (in module abjad.tools.componenttools.move_component_subtree_to_right_in_immediate_parent_of_component), 274
 move_measure_prolation_to_full_measure_tuplet (in module abjad.tools.measuretools.move_measure_prolation_to_full_measure_tuplet), 954
 move_parentage_and_spanners_from_components_to_components (in module abjad.tools.componenttools.move_parentage_and_spanners_from_components_to_components), 275
 move_parentage_children_and_spanners_from_components_to_empty_tuplet (in module abjad.tools.containertools.move_parentage_children_and_spanners_from_components_to_empty_tuplet), 319
 move_prolation_of_full_measure_tuplet_to_meter_of_measure_tuplet (in module abjad.tools.measuretools.move_prolation_of_full_measure_tuplet_to_meter_of_measure_tuplet), 955
 move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet (in module abjad.tools.tuplettools.move_prolation_of_tuplet_to_contents_of_tuplet_and_remove_tuplet), 1595
 move_spanners_from_component_to_children_of_component (in module abjad.tools.spannertools.move_spanners_from_component_to_children_of_component), 1519
 MultiMeasureRest (class in abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest), 1213
 MultipartBeamSpanner (class in abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner), 1429
 multiplied_duration (abjad.tools.chordtools.Chord.Chord attribute), 1173

multiply_contents_of_measures_in_expr() (in module abjad.tools.measuretools.multiply_contents_of_measures_in_expr), 955

multiply_contents_of_measures_in_expr_and_scale_meter_denominators() (in module abjad.tools.measuretools.multiply_contents_of_measures_in_expr_and_scale_meter_denominators), 956

multiply_duration_pair() (in module abjad.tools.durationtools.multiply_duration_pair), 1675

multiply_duration_pair_and_reduce_factors() (in module abjad.tools.durationtools.multiply_duration_pair_and_reduce_factors), 1675

multiply_duration_pair_and_try_to_preserve_numerator() (in module abjad.tools.durationtools.multiply_duration_pair_and_try_to_preserve_numerator), 1676

music (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 294

music (abjad.tools.containertools.Container.Container.Container attribute), 303

music (abjad.tools.contexttools.Context.Context.Context attribute), 349

music (abjad.tools.gracetools.Grace.Grace.Grace attribute), 417

music (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics attribute), 2017

music (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2044

music (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 906

music (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 917

music (abjad.tools.measuretools.Measure.Measure.Measure attribute), 927

music (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1250

music (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1271

music (abjad.tools.scoretools.Score.Score.Score attribute), 1281

music (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1292

music (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1522

music (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1532

music (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1566

music (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1578

music (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1600

name (abjad.tools.contexttools.Context.Context.Context attribute), 352

name (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark attribute), 370

name (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2047

name (abjad.tools.marktools.Annotation.Annotation.Annotation attribute), 846

name (abjad.tools.marktools.Articulation.Articulation.Articulation attribute), 850

name (abjad.tools.pitchtools.Accidental.Accidental.Accidental attribute), 990

name (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1253

name (abjad.tools.scoretools.Performer.Performer.Performer attribute), 1262

name (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1273

name (abjad.tools.scoretools.Score.Score.Score attribute), 1284

name (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1294

name (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1524

name (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1535

name (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator attribute), 2085

name (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator attribute), 2089

name (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 2098

name (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1602

named_chromatic_pitch (abjad.tools.noteheadtools.NoteHead.NoteHead.NoteHead attribute), 977

named_chromatic_pitch (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1122

named_chromatic_pitch_and_clef_to_staff_position_number() (in module abjad.tools.pitchtools.named_chromatic_pitch_and_clef_to_staff_position_number), 1204

named_chromatic_pitch_class (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch attribute), 1096

named_chromatic_pitch_class (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1123

named_chromatic_pitch_class_set (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment attribute), 1101

named_chromatic_pitch_class_set	(abjad.tools.tonalitytools.Scale.Scale.Scale	named_diatonic_pitch_class	(ab-
tribute), 2094	at-	jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch	tribute), 1096
named_chromatic_pitch_class_to_scale_degree()	(abjad.tools.tonalitytools.Scale.Scale.Scale	named_diatonic_pitch_class	(ab-
method), 2095	method), 2095	jad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch.Na	tribute), 1123
named_chromatic_pitch_class_vector	(ab-	named_diatonic_pitch_class	(ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedDiatonicPitchSegment.NamedDiatonicPitchSegment	attribute), 1109	attribute), 1149	attribute), 1149
named_chromatic_pitch_classes	(ab-	named_diatonic_pitch_class	(ab-
jad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment.NamedDiatonicPitchClassSegment.NamedDiatonicPitchClassSegment	attribute), 1101	attribute), 1151	attribute), 1151
named_chromatic_pitch_classes	(ab-	NamedChromaticPitch	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedDiatonicPitchClassSet.NamedDiatonicPitchClassSet	attribute), 1105	1094	1094
named_chromatic_pitch_classes	(ab-	NamedChromaticPitchClass	(class in ab-
jad.tools.tonalitytools.ChordClass.ChordClass.ChordClass	attribute), 2075	1098	1098
named_chromatic_pitch_classes	(ab-	NamedChromaticPitchClassSegment	(class in ab-
jad.tools.tonalitytools.Scale.Scale.Scale	tribute), 2095	1101	1101
named_chromatic_pitch_set	(ab-	NamedChromaticPitchClassSet	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedDiatonicPitchSegment.NamedDiatonicPitchSegment	attribute), 1109	1104	1104
named_chromatic_pitch_tokens_to_named_chromatic_pitches	(in module ab-	NamedChromaticPitchSegment	(class in ab-
jad.tools.pitchtools.named_chromatic_pitch_tokens_to_named_chromatic_pitches),	1204	1208	1208
named_chromatic_pitch_vector	(ab-	NamedChromaticPitchSet	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment	attribute), 1109	1112	1112
named_chromatic_pitches	(ab-	NamedChromaticPitchVector	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment	attribute), 1109	1116	1116
named_chromatic_pitches	(ab-	NamedDiatonicPitch	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet	attribute), 1112	1122	1122
named_chromatic_pitches	(ab-	NamedDiatonicPitchClass	(class in ab-
jad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector	attribute), 1116	1125	1125
named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary()	(in module ab-	NaturalHarmonic	(class in ab-
jad.tools.pitchtools.named_chromatic_pitches_to_harmonic_chromatic_interval_class_number_dictionary()	1204	965	965
named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary()	(in module ab-	1792	1792
named_chromatic_pitches_to_inversion_equivalent_chromatic_interval_class_number_dictionary()	(in module ab-	1205	1205
named_diatonic_pitch	(ab-	negate_absolute_value_of_sequence_elements_at_indices()	(in module ab-
jad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch.NamedChromaticPitch	attribute), 1096	1122	1122
named_diatonic_pitch	(ab-	negate_sequence_elements_at_indices()	(in module ab-
jad.tools.pitchtools.NumberedDiatonicPitch.NumberedDiatonicPitch.NumberedDiatonicPitch	attribute), 1149	1149	1149

negate_sequence_elements_cyclically() (in module abjad.tools.sequencetools.negative_sequence_elements_cyclically), 1793

negative_level (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1752

negative_level (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1768

NestedMeasureCheck (class in abjad.tools.wellformednesstools.NestedMeasureCheck), 1978

next (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell attribute), 1732

next (abjad.tools.quantizationtools.QGrid.QGrid.QGrid attribute), 2055

next_integer_partition() (in module abjad.tools.mathtools.next_integer_partition), 1723

next_vertical_moment (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1942

nodes (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1621

normalized_spacing_duration (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication attribute), 1698

Note (class in abjad.tools.notetools.Note.Note), 971

note_head (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic attribute), 968

note_head (abjad.tools.notetools.Note.Note.Note attribute), 973

note_heads (abjad.tools.chordtools.Chord.Chord.Chord attribute), 226

NoteFilledTimeTokenMaker (class in abjad.tools.timetokentools.NoteFilledTimeTokenMaker), 1908

NoteHead (class in abjad.tools.notetools.NoteHead.NoteHead), 977

notes (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

notes_and_chords_in_expr_are_on_expected_clefs() (in module abjad.tools.instrumenttools.notes_and_chords_in_expr_are_on_expected_clefs), 765

notes_and_chords_in_expr_are_within_traditional_instrument_ranges() (in module abjad.tools.instrumenttools.notes_and_chords_in_expr_are_within_traditional_instrument_ranges), 766

number (abjad.tools.pitchtools.HarmonicChromaticInterval.HarmonicChromaticInterval attribute), 992

number (abjad.tools.pitchtools.HarmonicChromaticIntervalClass.HarmonicChromaticIntervalClass attribute), 994

number (abjad.tools.pitchtools.HarmonicCounterpointInterval.HarmonicCounterpointInterval attribute), 1009

number (abjad.tools.pitchtools.HarmonicCounterpointIntervalClass.HarmonicCounterpointIntervalClass attribute), 1011

number (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval attribute), 1014

number (abjad.tools.pitchtools.HarmonicDiatonicIntervalClass.HarmonicDiatonicIntervalClass attribute), 1016

number (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClass.InversionEquivalentChromaticIntervalClass attribute), 1030

number (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClass.InversionEquivalentDiatonicIntervalClass attribute), 1044

number (abjad.tools.pitchtools.MelodicChromaticInterval.MelodicChromaticInterval attribute), 1055

number (abjad.tools.pitchtools.MelodicChromaticIntervalClass.MelodicChromaticIntervalClass attribute), 1057

number (abjad.tools.pitchtools.MelodicCounterpointInterval.MelodicCounterpointInterval attribute), 1076

number (abjad.tools.pitchtools.MelodicCounterpointIntervalClass.MelodicCounterpointIntervalClass attribute), 1078

number (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1081

number (abjad.tools.pitchtools.MelodicDiatonicIntervalClass.MelodicDiatonicIntervalClass attribute), 1084

number (abjad.tools.tonalitytools.ExtentIndicator.ExtentIndicator.ExtentIndicator attribute), 2085

number (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator attribute), 2087

number (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 2098

number_is_between_prolated_start_and_stop_offsets_of_component() (in module abjad.tools.componenttools.number_is_between_prolated_start_and_stop_offsets_of_component), 275

number_is_between_start_and_stop_offsets_of_component_in_seconds() (in module abjad.tools.componenttools.number_is_between_start_and_stop_offsets_of_component_in_seconds), 276

numbered_chromatic_pitch (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1096

numbered_chromatic_pitch (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitch attribute), 1123

numbered_chromatic_pitch_class (abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitchClass attribute), 1096

numbered_chromatic_pitch_class (abjad.tools.pitchtools.NamedDiatonicPitch.NamedDiatonicPitchClass attribute), 1098

numbered_chromatic_pitch_class_segment (abjad.tools.pitchtools.NamedChromaticPitchClassSegment.NamedChromaticPitchClassSegment attribute), 1101

ObjectInventory (class in abjad.tools.datastructuretools.ObjectInventory.ObjectInventory attribute), 1430
1628
Oboe (class in abjad.tools.instrumenttools.Oboe.Oboe), attribute), 1438
650
OctavationSpanner (class in abjad.tools.spannertools.OctavationSpanner.OctavationSpanner attribute), 1446
1437
octave_number (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet attribute), 1097
octave_number_to_octave_tick_string() (in module abjad.tools.pitchtools.octave_number_to_octave_tick_string), attribute), 1469
1205
octave_tick_string_to_octave_number() (in module abjad.tools.pitchtools.octave_tick_string_to_octave_number), attribute), 1475
1206
OctaveTranspositionMapping (class in abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping attribute), 1498
1153
OctaveTranspositionMappingComponent (class in abjad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent attribute), 1544
1158
OctaveTranspositionMappingInventory (class in abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory attribute), 1544
1160
offset (abjad.tools.quantizationtools.QEvent.QEvent.QEvent attribute), 2053
offset (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 2062
1317
offset (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner attribute), 1325
2090
offset (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner attribute), 1333
offset (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1343
offset (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1353
1165
offset (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 1362
1822
offset (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1372
1105
offset (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1380
2076
offset (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1388
offset (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1397
1206
offset (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1404
offset (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 1413
1911
offset (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1422
1915

OutputIncisedRestFilledTimeTokenMaker (class in abjad.tools.timetokenmaker.tools.OutputIncisedRestFilledTimeTokenMaker), 1919

OutputIncisedTimeTokenMaker (class in abjad.tools.timetokenmaker.tools.OutputIncisedTimeTokenMaker), 1900

overlap_components (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1942

overlap_leaves (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1942

overlap_measures (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1942

overlap_notes (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment attribute), 1942

OverlappingBeamCheck (class in abjad.tools.wellformednesstools.OverlappingBeamCheck), 1980

OverlappingGlissandoCheck (class in abjad.tools.wellformednesstools.OverlappingGlissandoCheck), 1982

OverlappingOctavationCheck (class in abjad.tools.wellformednesstools.OverlappingOctavationCheck), 1984

override (abjad.tools.chordtools.Chord.Chord.Chord attribute), 224

override (abjad.tools.componenttools.Component.Component attribute), 236

override (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 295

override (abjad.tools.containertools.Container.Container.Container attribute), 303

override (abjad.tools.contexttools.Context.Context.Context attribute), 349

override (abjad.tools.gracetools.Grace.Grace.Grace attribute), 417

override (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 768

override (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock attribute), 820

override (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics attribute), 2017

override (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender attribute), 2026

override (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2030

override (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2045

override (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace attribute), 2035

override (abjad.tools.lyricstools.LyricText.LyricText.LyricText attribute), 2039

override (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 907

override (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 917

override (abjad.tools.measuretools.Measure.Measure.Measure attribute), 927

override (abjad.tools.measuretools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 966

override (abjad.tools.notetools.Note.Note.Note attribute), 972

override (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1244

override (abjad.tools.resttools.Rest.Rest.Rest attribute), 1244

override (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1244

override (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1271

override (abjad.tools.scoretools.Score.Score.Score attribute), 1281

override (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1281

override (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1309

override (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 1318

override (abjad.tools.spannertools.BraceSpanner.BraceSpanner.BraceSpanner attribute), 1326

override (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner.ComplexBeamSpanner attribute), 1334

override (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1343

override (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1353

override (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner attribute), 1363

override (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1372

override (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1380

override (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1389

override (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1397

override (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1405

override (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 1413

override (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1422

override (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner attribute), 1431

override (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner attribute), 1438

override (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner attribute), 1446

override (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.Component.Component.Component attribute), 1454
 override (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner(abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 1462
 override (abjad.tools.spannertools.Spanner.Spanner.Spannerparent (abjad.tools.containertools.Container.Container.Container attribute), 1469
 override (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner(abjad.tools.containertools.Context.Context.Context attribute), 1475
 override (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner(abjad.tools.containertools.Grace.Grace.Grace attribute), 1483
 override (abjad.tools.spannertools.TextSpanner.TextSpannerparent (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 1491
 override (abjad.tools.spannertools.TrillSpanner.TrillSpannerparent (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics attribute), 1498
 override (abjad.tools.stafftools.RhythmicStaff.RhythmicStaffRhythmicStaff(abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender attribute), 1522
 override (abjad.tools.stafftools.Staff.Staff.Staff attribute), parent (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 1532
 override (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpannerparent (abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics attribute), 1544
 override (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTupletFixedDurationTuplet(abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace attribute), 1566
 override (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), parent (abjad.tools.lyrictools.LyricText.LyricText.LyricText attribute), 1578
 override (abjad.tools.voicetools.Voice.Voice.Voice attribute), parent (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 1600
 overwrite_sequence_elements_at_indices() parent (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 1793
 (in module abjad.tools.sequencetools.overwrite_sequence_elements_at_indices) parent (abjad.tools.measuretools.Measure.Measure.Measure attribute), 1793

P

p() (in module abjad.tools.iotools.p), 1691
 pad_measures_in_expr_with_rests() (in module abjad.tools.measuretools.pad_measures_in_expr_with_rests) parent (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 956
 pad_measures_in_expr_with_skips() (in module abjad.tools.measuretools.pad_measures_in_expr_with_skips) parent (abjad.tools.resttools.Rest.Rest.Rest attribute), 958
 pad_to_depth() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1250
 method), 1730 parent (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1271
 pad_to_width() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1271
 method), 1730 parent (abjad.tools.scoretools.Score.Score.Score attribute), 1282
 pad_to_width() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1282
 method), 1738 parent (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1292
 pair (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark attribute), 388
 parent (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1309
 pairs (abjad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap attribute), 1132
 parent (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1522
 PaperBlock (class in abjad.tools.lilypondfiletools.PaperBlock.PaperBlock.PaperBlock attribute), 837
 parent (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1532
 parent (abjad.tools.chordtools.Chord.Chord.Chord attribute), 225
 parent (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1566

parent (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1578

parent (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1600

parent_array (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1732

parent_array (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1735

parent_array (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1737

parent_column (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1732

parent_graph (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1622

parent_row (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1732

parse_lilypond_input_string() (in module abjad.tools.iotools.parse_lilypond_input_string), 1692

parse_rtm_syntax() (in module abjad.tools.rhythmtreetools.parse_rtm_syntax), 1743

partial (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark attribute), 389

partition() (abjad.tools.datastructuretools.Digraph.Digraph.Digraph method), 1622

partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang()), 276

partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_exactly_with_overhang()), 276

partition_components_cyclically_by_durations_in_seconds_ge_with_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_ge_with_overhang()), 277

partition_components_cyclically_by_durations_in_seconds_ge_without_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_ge_without_overhang()), 280

partition_components_cyclically_by_durations_in_seconds_le_with_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_le_with_overhang()), 277

partition_components_cyclically_by_durations_in_seconds_le_without_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_durations_in_seconds_le_without_overhang()), 280

partition_components_cyclically_by_prolated_durations_exactly_with_overhang() (in module abjad.tools.componenttools.partition_components_cyclically_by_prolated_durations_exactly_with_overhang()), 277

[jad.tools.componenttools.partition_components_once_by_prolated_durations_ge_with_overhang\(\)](#),
[280](#) [partition_sequence_cyclically_by_weights_at_least_with_overhang\(\)](#)
[partition_components_once_by_prolated_durations_ge_without_overhang\(\)](#) [\(in module ab-](#)
[\(in module ab-](#) [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[jad.tools.componenttools.partition_components_once_by_prolated_durations_ge_without_overhang\(\)](#),
[280](#) [partition_sequence_cyclically_by_weights_at_least_without_overhang\(\)](#)
[partition_components_once_by_prolated_durations_le_with_overhang\(\)](#) [\(in module ab-](#)
[\(in module ab-](#) [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[jad.tools.componenttools.partition_components_once_by_prolated_durations_le_with_overhang\(\)](#),
[281](#) [partition_sequence_cyclically_by_weights_at_most_with_overhang\(\)](#)
[partition_components_once_by_prolated_durations_le_without_overhang\(\)](#) [\(in module ab-](#)
[\(in module ab-](#) [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[jad.tools.componenttools.partition_components_once_by_prolated_durations_le_without_overhang\(\)](#),
[281](#) [partition_sequence_cyclically_by_weights_at_most_without_overhang\(\)](#)
[partition_integer_by_ratio\(\)](#) [\(in module ab-](#) [\(in module ab-](#)
[jad.tools.mathtools.partition_integer_by_ratio\)](#), [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[1723](#) [1797](#)
[partition_integer_into_canonic_parts\(\)](#) [\(in module ab-](#) [partition_sequence_cyclically_by_weights_exactly_with_overhang\(\)](#)
[jad.tools.mathtools.partition_integer_into_canonic_parts\)](#), [\(in module ab-](#)
[1724](#) [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[partition_integer_into_halves\(\)](#) [\(in module ab-](#) [1797](#)
[jad.tools.mathtools.partition_integer_into_halves\)](#) [partition_sequence_cyclically_by_weights_exactly_without_overhang\(\)](#)
[1725](#) [\(in module ab-](#)
[partition_integer_into thirds\(\)](#) [\(in module ab-](#) [jad.tools.sequencetools.partition_sequence_cyclically_by_weight](#)
[jad.tools.mathtools.partition_integer_into thirds\)](#), [1798](#)
[1726](#) [partition_sequence_extended_to_counts_with_overhang\(\)](#)
[partition_integer_into_units\(\)](#) [\(in module ab-](#) [\(in module ab-](#)
[jad.tools.mathtools.partition_integer_into_units\)](#), [jad.tools.sequencetools.partition_sequence_extended_to_counts_](#)
[1726](#) [1798](#)
[partition_sequence_by_ratio_of_lengths\(\)](#) [\(in module ab-](#) [partition_sequence_extended_to_counts_without_overhang\(\)](#)
[jad.tools.sequencetools.partition_sequence_by_ratio_of_lengths\)](#), [\(in module ab-](#)
[1793](#) [jad.tools.sequencetools.partition_sequence_extended_to_counts_](#)
[partition_sequence_by_ratio_of_weights\(\)](#) [\(in module ab-](#) [1798](#)
[jad.tools.sequencetools.partition_sequence_by_ratio_of_weights\)](#) [partition_sequence_once_by_counts_with_overhang\(\)](#)
[1794](#) [\(in module ab-](#)
[partition_sequence_by_restricted_growth_function\(\)](#) [jad.tools.sequencetools.partition_sequence_once_by_counts_with](#)
[\(in module ab-](#) [1799](#)
[jad.tools.sequencetools.partition_sequence_by_restricted_growth_function\)](#) [partition_sequence_once_by_counts_without_overhang\(\)](#)
[1794](#) [\(in module ab-](#)
[partition_sequence_by_sign_of_elements\(\)](#) [jad.tools.sequencetools.partition_sequence_once_by_counts_with](#)
[\(in module ab-](#) [1799](#)
[jad.tools.sequencetools.partition_sequence_by_sign_of_elements\)](#) [partition_sequence_once_by_weights_at_least_with_overhang\(\)](#)
[1795](#) [\(in module ab-](#)
[partition_sequence_by_value_of_elements\(\)](#) [jad.tools.sequencetools.partition_sequence_once_by_weights_at_](#)
[\(in module ab-](#) [1799](#)
[jad.tools.sequencetools.partition_sequence_by_value_of_elements\)](#) [partition_sequence_once_by_weights_at_least_without_overhang\(\)](#)
[1795](#) [\(in module ab-](#)
[partition_sequence_cyclically_by_counts_with_overhang\(\)](#) [jad.tools.sequencetools.partition_sequence_once_by_weights_at_](#)
[\(in module ab-](#) [1799](#)
[jad.tools.sequencetools.partition_sequence_cyclically_by_counts_with_overhang\)](#) [partition_sequence_once_by_weights_at_most_with_overhang\(\)](#)
[1796](#) [\(in module ab-](#)
[partition_sequence_cyclically_by_counts_without_overhang\(\)](#) [jad.tools.sequencetools.partition_sequence_once_by_weights_at_](#)
[\(in module ab-](#) [1800](#)
[jad.tools.sequencetools.partition_sequence_cyclically_by_counts_without_overhang\)](#),

partition_sequence_once_by_weights_at_most_without_overhang() (in module abjad.tools.sequencetools.partition_sequence_once_by_weights_at_most_without_overhang), 1800

partition_sequence_once_by_weights_exactly_with_overhang() (in module abjad.tools.sequencetools.partition_sequence_once_by_weights_exactly_with_overhang), 1800

partition_sequence_once_by_weights_exactly_without_overhang() (in module abjad.tools.sequencetools.partition_sequence_once_by_weights_exactly_without_overhang), 1801

pdf() (in module abjad.tools.iotools.pdf), 1692

pentatonic_pitch_number_to_chromatic_pitch_number() (in module abjad.tools.pitchtools.pentatonic_pitch_number_to_chromatic_pitch_number), 1206

Performer (class in abjad.tools.scoretools.Performer.Performer), 1261

performer_count (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1259

performer_name_string (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1259

PerformerInventory (class in abjad.tools.scoretools.PerformerInventory.PerformerInventory), 1264

performers (abjad.tools.scoretools.InstrumentationSpecifier.InstrumentationSpecifier attribute), 1259

period (abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression attribute), 1822

permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row() (in module abjad.tools.pitchtools.permute_named_chromatic_pitch_carrier_list_by_twelve_tone_row), 1207

permute_sequence() (in module abjad.tools.sequencetools.permute_sequence), 1801

PhrasingSlurSpanner (class in abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner), 1445

Piano (class in abjad.tools.instrumenttools.Piano.Piano), 657

PianoPedalSpanner (class in abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner), 1452

PianoStaff (class in abjad.tools.scoretools.PianoStaff.PianoStaff), 1269

Piccolo (class in abjad.tools.instrumenttools.Piccolo.Piccolo), 664

Piano (class in abjad.tools.documentatointools.Piano.Piano), 1655

pitch_abjad_to_midi_without_trill_spanner (abjad.tools.trillspanner.TrillSpanner attribute), 1499

pitch_array_row_to_measure() (in module abjad.tools.measuretools.pitch_array_row_to_measure), 966

pitch_array_to_measures() (in module abjad.tools.measuretools.pitch_array_to_measures), 960

pitch_class_octave_number_string_to_chromatic_pitch_name() (in module abjad.tools.pitchtools.NamedChromaticPitch.NamedChromaticPitch attribute), 1097

pitch_class_octave_number_string_to_chromatic_pitch_number() (in module abjad.tools.pitchtools.pitch_class_octave_number_string_to_chromatic_pitch_number), 1207

pitch_iterables (abjad.tools.pitchtools.NumberedChromaticPitchClassColor attribute), 1132

pitch_range (abjad.tools.instrumenttools.Accordion.Accordion attribute), 1132

pitch_range (abjad.tools.instrumenttools.AltoFlute.AltoFlute attribute), 438

pitch_range (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone attribute), 445

pitch_range (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone attribute), 452

pitch_range (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone attribute), 452

pitch_range (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice attribute), 452

pitch_range (abjad.tools.instrumenttools.BassClarinet.BassClarinet attribute), 480

pitch_range (abjad.tools.instrumenttools.BassFlute.BassFlute attribute), 487

pitch_range (abjad.tools.instrumenttools.Bassoon.Bassoon attribute), 515

pitch_range (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone attribute), 494

pitch_range (abjad.tools.instrumenttools.BassTrombone.BassTrombone attribute), 501

pitch_range (abjad.tools.instrumenttools.BassVoice.BassVoice attribute), 508

pitch_range (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet attribute), 459

pitch_range (abjad.tools.instrumenttools.Cello.Cello attribute), 522

pitch_range (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA attribute), 529

pitch_range (abjad.tools.instrumenttools.Contrabass.Contrabass attribute), 536

pitch_range (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet attribute), 543

pop() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory.QGridTempoLookup.QGridTempoLookup method), 383

pop() (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary.ImmutableDictionary.CredStaff.CredStaff.CredStaff method), 1625

pop() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory.PerformerInventory.PerformerInventory.PerformerInventory method), 1629

pop() (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph.PianoStaff.PianoStaff.PianoStaff method), 1650

pop() (abjad.tools.gracetools.Grace.Grace.Grace method), 421

pop() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory.StaffGroup.StaffGroup.StaffGroup method), 632

pop() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 811

pop() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 816

pop() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 828

pop() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 840

pop() (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics method), 2021

pop() (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2049

pop() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 897

pop() (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 911

pop() (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure method), 921

pop() (abjad.tools.measuretools.Measure.Measure.Measure method), 930

pop() (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1738

pop() (abjad.tools.pitchtools.HarmonicChromaticIntervalClassVec.HarmonicChromaticIntervalClassVec.HarmonicChromaticIntervalClassVec method), 998

pop() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVec.InversionEquivalentChromaticIntervalClassVec.InversionEquivalentChromaticIntervalClassVec method), 1040

pop() (abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVec.InversionEquivalentDiatonicIntervalClassVec.InversionEquivalentDiatonicIntervalClassVec method), 1051

pop() (abjad.tools.pitchtools.MelodicChromaticIntervalClassVec.MelodicChromaticIntervalClassVec.MelodicChromaticIntervalClassVec method), 1064

pop() (abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector.NamedChromaticPitchVector method), 1117

pop() (abjad.tools.pitchtools.NumberedChromaticPitchClassVec.NumberedChromaticPitchClassVec.NumberedChromaticPitchClassVec method), 1145

pop() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1154

pop() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory method), 1161

pop() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory method), 1168

pop() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree method), 2064

pop() (abjad.tools.requiemtools.RequiemInventory.RequiemInventory.RequiemInventory.QGridRequiemLookup.QGridRequiemLookup method), 2068

pop() (abjad.tools.scoretools.Score.Score.Score method), 1254

pop() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1265

pop() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1275

pop() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup method), 1286

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1296

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1744

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1321

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1329

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1338

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1348

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1358

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1368

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1376

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1384

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1393

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1401

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1408

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1418

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1426

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1434

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1442

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1450

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1458

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1465

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1472

pop() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList method), 1479

[pop\(\)](#) (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.
[method](#)), [1487](#)

[pop\(\)](#) (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.
[method](#)), [1494](#)

[pop\(\)](#) (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.
[method](#)), [1502](#)

[pop\(\)](#) (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.
[method](#)), [1526](#)

[pop\(\)](#) (abjad.tools.stafftools.Staff.Staff.Staff [method](#)), [pop_left\(\)](#) (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.S
[1536](#) [method](#)), [1479](#)

[pop\(\)](#) (abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner.
[method](#)), [1548](#)

[pop\(\)](#) (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval.
[method](#)), [1842](#)

[pop\(\)](#) (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.
[method](#)), [1873](#)

[pop\(\)](#) (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.
[method](#)), [1572](#)

[pop\(\)](#) (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet [method](#)), [pop_row\(\)](#) (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray
[method](#)), [1583](#) [method](#)), [1730](#)

[pop\(\)](#) (abjad.tools.voicetools.Voice.Voice.Voice [method](#)), [popitem\(\)](#) (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableD
[1604](#) [method](#)), [1625](#)

[pop_column\(\)](#) (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.
[method](#)), [1730](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [1321](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [1329](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [1339](#)

[pop_left\(\)](#) (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.
[method](#)), [1348](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.
[method](#)), [1358](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.
[method](#)), [1368](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.
[method](#)), [1376](#)

[pop_left\(\)](#) (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.
[method](#)), [1384](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.
[method](#)), [1394](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.
[method](#)), [1401](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner.
[method](#)), [1408](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.
[method](#)), [1418](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner.
[method](#)), [1426](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.
[method](#)), [1434](#)

[pop_left\(\)](#) (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.
[method](#)), [1443](#)

[pop_left\(\)](#) (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner.
[method](#)), [1450](#)

[pop_left\(\)](#) (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner.
[method](#)), [1458](#)

[pop_left\(\)](#) (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner.
[method](#)), [1466](#)

[pop_left\(\)](#) (abjad.tools.spannertools.Spanner.Spanner.Spanner.
[method](#)), [1472](#)

[pop_left\(\)](#) (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner.
[method](#)), [1479](#)

[pop_left\(\)](#) (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner.
[method](#)), [1487](#)

[pop_left\(\)](#) (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner.
[method](#)), [1495](#)

[pop_left\(\)](#) (abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner.
[method](#)), [1502](#)

[pop_left\(\)](#) (abjad.tools.spannertools.TieSpanner.TieSpanner.TieSpanner.
[method](#)), [1548](#)

[pop_row\(\)](#) (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray
[method](#)), [1730](#)

[popitem\(\)](#) (abjad.tools.datastructuretools.ImmutableDictionary.ImmutableD
[method](#)), [1625](#)

[pop_column\(\)](#) (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.
[method](#)), [1650](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [998](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [1040](#)

[pop_left\(\)](#) (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner.
[method](#)), [1051](#)

[pop_left\(\)](#) (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner.
[method](#)), [1064](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner.
[method](#)), [1118](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner.
[method](#)), [1145](#)

[pop_left\(\)](#) (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner.
[method](#)), [2064](#)

[pop_left\(\)](#) (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner.
[method](#)), [2068](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner.
[method](#)), [1842](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner.
[method](#)), [1873](#)

[pop_left\(\)](#) (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner.
[method](#)), [1753](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.
[method](#)), [1768](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner.
[method](#)), [2080](#)

[pop_left\(\)](#) (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner.
[method](#)), [1434](#)

[pop_left\(\)](#) (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner.
[method](#)), [1677](#)

predicate (abjad.tools.constrainttools.AbsoluteIndexConstraint.AbsoluteIndexConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1991	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2045
predicate (abjad.tools.constrainttools.GlobalConstraint.GlobalConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1999	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2045
predicate (abjad.tools.constrainttools.GlobalCountsConstraint.GlobalCountsConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2002	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2035
predicate (abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2005	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2035
predicate (abjad.tools.constrainttools.RelativeCountsConstraint.RelativeCountsConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2008	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2035
predicate (abjad.tools.constrainttools.RelativeIndexConstraint.RelativeIndexConstraint (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2011	preprolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 907
preferred_denominator (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1569	preprolated_duration (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 918
preferred_denominator (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1581	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
prefix (abjad.tools.documentationtools.APICrawler.APICrawler (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1633	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
prefix (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1640	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
prefix (abjad.tools.documentationtools.Documenter.Documenter (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1642	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
prefix (abjad.tools.documentationtools.FunctionDocumenter.FunctionDocumenter (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 1646	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.chordtools.Chord.Chord.Chord (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 225	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.containertools.Cluster.Cluster.Cluster (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 295	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.containertools.Container.Container.Container (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 303	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.contexttools.Context.Context.Context (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 350	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.gracetools.Grace.Grace.Grace (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 417	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.leafstools.Leaf.Leaf.Leaf (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 769	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2018	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2026	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927
preprolated_duration (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2030	preprolated_duration (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 927

primary_clefs (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 459

primary_clefs (abjad.tools.instrumenttools.Cello.Cello.Celloprimary_clefs (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 522

primary_clefs (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA.instrumenttools.Tuba.Tuba.Tuba attribute), 529

primary_clefs (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass.instrumenttools.UntunedPercussion.UntunedPercussion attribute), 536

primary_clefs (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 543

primary_clefs (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute.instrumenttools.Viola.Viola.Viola attribute), 550

primary_clefs (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon.instrumenttools.Violin.Violin.Violin attribute), 564

primary_clefs (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 557

primary_clefs (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice.instrumenttools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet attribute), 571

primary_clefs (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet(in module abjad.tools.iotools.profile_expr), 578

primary_clefs (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.instrumenttools.Chord.Chord.Chord attribute), 585

primary_clefs (abjad.tools.instrumenttools.Flute.Flute.Flute prolated_duration (abjad.tools.componenttools.Component.Component.Component attribute), 592

primary_clefs (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.instrumenttools.Cluster.Cluster.Cluster attribute), 599

primary_clefs (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.instrumenttools.Container.Container.Container attribute), 606

primary_clefs (abjad.tools.instrumenttools.Guitar.Guitar.Guitar prolated_duration (abjad.tools.contexttools.Context.Context.Context attribute), 613

primary_clefs (abjad.tools.instrumenttools.Harp.Harp.Harp prolated_duration (abjad.tools.gracetools.Grace.Grace.Grace attribute), 620

primary_clefs (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.instrumenttools.Leaf.Leaf.Leaf attribute), 627

primary_clefs (abjad.tools.instrumenttools.Marimba.Marimba.Marimba.instrumenttools.AddLyrics.AddLyrics.AddLyrics attribute), 639

primary_clefs (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice.instrumenttools.LyricExtender.LyricExtender.LyricExtender attribute), 646

primary_clefs (abjad.tools.instrumenttools.Oboe.Oboe.Oboe prolated_duration (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 653

primary_clefs (abjad.tools.instrumenttools.Piano.Piano.Piano prolated_duration (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 660

primary_clefs (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo prolated_duration (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace attribute), 667

primary_clefs (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone.instrumenttools.LyricText.LyricText.LyricText attribute), 674

primary_clefs (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone.instrumenttools.LyricText.LyricText.LyricText attribute), 681

primary_clefs (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice.instrumenttools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 688

primary_clefs (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone.instrumenttools.Measure.Measure.Measure attribute), 695

primary_clefs (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone.instrumenttools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 702

prolation (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 967
 prolation (abjad.tools.notetools.Note.Note.Note attribute), 973
 prolation (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1214
 prolation (abjad.tools.resttools.Rest.Rest.Rest attribute), 1219
 prolation (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1251
 prolation (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1272
 prolation (abjad.tools.scoretools.Score.Score.Score attribute), 1282
 prolation (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1292
 prolation (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1309
 prolation (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1522
 prolation (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1533
 prolation (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1567
 prolation (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1578
 prolation (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1600
 proper_parentage (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1753
 proper_parentage (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1768
 proportional_notation_duration (abjad.tools.layouttools.SpacingIndication.SpacingIndication.SpacingIndication attribute), 1698
 prune() (abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree.QGridSearchTree method), 2064

Q

QEvent (class in abjad.tools.quantizationtools.QEvent.QEvent), 2053
 QGrid (class in abjad.tools.quantizationtools.QGrid.QGrid), 2055
 QGridQuantizer (class in abjad.tools.quantizationtools.QGridQuantizer.QGridQuantizer), 2058
 QGridSearchTree (class in abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree), 2062
 QGridTempoLookup (class in abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup), 2067

R

randomized (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver attribute), 1996
 randomized (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver attribute), 2014
 randomized() (abjad.tools.constrainttools.Domain.Domain.Domain method), 1993
 ratio_string (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet attribute), 1567
 ratio_string (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1578

rational_to_duration_pair_with_specified_integer_denominator	(in module abjad.tools.durationtools.rational_to_duration_pair_with_specified_integer_denominator)	1677	remove()	(abjad.tools.durationtools.rational_to_duration_pair_with_specified_integer_denominator)	1207
rational_to_duration_pair_with_specified_integer_denominator	(in module abjad.tools.durationtools.rational_to_duration_pair_with_specified_integer_denominator)	1678	RelativeIndexConstraint	(class in abjad.tools.constrainttools.RelativeCountsConstraint.RelativeIndexConstraint)	
rational_to_equal_or_greater_assignable_rational()	(in module abjad.tools.durationtools.rational_to_equal_or_greater_assignable_rational)	1679	remove()	(abjad.tools.chordtools.Chord.Chord.Chord method), 227	
rational_to_equal_or_greater_binary_rational()	(in module abjad.tools.durationtools.rational_to_equal_or_greater_binary_rational)	1679	remove()	(abjad.tools.containertools.Cluster.Cluster.Cluster method), 298	
rational_to_equal_or_greater_binary_rational()	(in module abjad.tools.durationtools.rational_to_equal_or_greater_binary_rational)	1679	remove()	(abjad.tools.containertools.Container.Container.Container method), 306	
rational_to_equal_or_lesser_assignable_rational()	(in module abjad.tools.durationtools.rational_to_equal_or_lesser_assignable_rational)	1680	remove()	(abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory method), 344	
rational_to_equal_or_lesser_binary_rational()	(in module abjad.tools.durationtools.rational_to_equal_or_lesser_binary_rational)	1681	remove()	(abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory method), 383	
rational_to_flag_count()	(in module abjad.tools.durationtools.rational_to_flag_count)	1681	remove()	(abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory method), 1629	
rational_to_fraction_string()	(in module abjad.tools.durationtools.rational_to_fraction_string)	1681	remove()	(abjad.tools.gracetools.Grace.Grace.Grace method), 422	
rational_to_prolation_string()	(in module abjad.tools.durationtools.rational_to_prolation_string)	1682	remove()	(abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory method), 632	
rational_to_proportion_string()	(in module abjad.tools.durationtools.rational_to_proportion_string)	1682	remove()	(abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock method), 811	
rational_to_proportion_string()	(in module abjad.tools.durationtools.rational_to_proportion_string)	1682	remove()	(abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock method), 816	
rational_to_proportion_string()	(in module abjad.tools.durationtools.rational_to_proportion_string)	1682	remove()	(abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile method), 828	
rational_to_proportion_string()	(in module abjad.tools.durationtools.rational_to_proportion_string)	1682	remove()	(abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock method), 840	
rcs	(abjad.tools.sievetools.ResidueClassExpression.ResidueClassExpression attribute), 1822		remove()	(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2021	
read()	(abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1656		remove()	(abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics method), 2049	
readonly_properties	(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1640		remove()	(abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory method), 880	
readwrite_properties	(abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1640		remove()	(abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure method), 912	
real	(abjad.tools.durationtools.Duration.Duration.Duration attribute), 1660		remove()	(abjad.tools.measuretools.Measure.Measure.Measure method), 931	
real	(abjad.tools.durationtools.Offset.Offset.Offset attribute), 1664		remove()	(abjad.tools.measuretools.Measure.Measure.Measure method), 931	
redo()	(in module abjad.tools.iotools.redo), 1693		remove()	(abjad.tools.measuretools.Measure.Measure.Measure method), 931	
reference	(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint attribute), 2005		remove()	(abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow method), 1738	
reference	(abjad.tools.constrainttools.GlobalReferenceConstraint.GlobalReferenceConstraint attribute), 2005		remove()	(abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping method), 1161	

remove() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory.remove_markup_attached_to_component(), method), 1168

remove() (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff.remove_markup_from_leaves_in_expr() (in module abjad.tools.markuptools.remove_markup_from_leaves_in_expr(), method), 1255

remove() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory.remove_outer_rests_from_sequence() (in module abjad.tools.leafstools.remove_outer_rests_from_sequence(), method), 1265

remove() (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff.remove_pitches() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.remove_pitches(), method), 1276

remove() (abjad.tools.scoretools.Score.Score.Score.remove_pitches() (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.remove_pitches(), method), 1286

remove() (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup.remove_powers_of_two() (in module abjad.tools.mathtools.remove_powers_of_two(), method), 1297

remove() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1744

remove() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1757

remove() (abjad.tools.sequencetools.Tree.Tree.Tree.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1772

remove() (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1527

remove() (abjad.tools.stafftools.Staff.Staff.Staff.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1537

remove() (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1572

remove() (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1583

remove() (abjad.tools.voicetools.Voice.Voice.Voice.remove_row() (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray.remove_row(), method), 1605

remove_abjad__pycache__directories() (in module abjad.tools.iotools.remove_abjad__pycache__directories(), method), 1694

remove_abjad_pyc_files() (in module abjad.tools.iotools.remove_abjad_pyc_files(), method), 1694

remove_all_leaves_in_tie_chain_except_first() (in module abjad.tools.tietools.remove_all_leaves_in_tie_chain_except_first(), method), 1558

remove_component_subtree_from_score_and_spanners() (in module abjad.tools.componenttools.remove_component_subtree_from_score_and_spanners(), method), 281

remove_empty_containers_in_expr() (in module abjad.tools.containertools.remove_empty_containers_in_expr(), method), 319

remove_initial_rests_from_sequence() (in module abjad.tools.leafstools.remove_initial_rests_from_sequence(), method), 796

remove_leaf_and_shrink_durated_parent_containers() (in module abjad.tools.leafstools.remove_leaf_and_shrink_durated_parent_containers(), method), 797

remove_markup_attached_to_component() (in module abjad.tools.markuptools.remove_markup_attached_to_component(), method), 799

```

repeat_runs_in_sequence_to_count() (in module abjad.tools.sequencetools.repeat_runs_in_sequence_to_count), 325
1802
replace_leaves_in_expr_with_named_parallel_voices()
repeat_sequence_elements_at_indices() (in module abjad.tools.sequencetools.repeat_sequence_elements_at_indices), 325
1803
800
replace_leaves_in_expr_with_parallel_voices()
repeat_sequence_elements_at_indices_cyclically() (in module abjad.tools.sequencetools.repeat_sequence_elements_at_indices_cyclically), 325
1804
802
replace_leaves_in_expr_with_rests() (in module abjad.tools.resttools.replace_leaves_in_expr_with_rests), 1224
repeat_sequence_elements_n_times_each() (in module abjad.tools.sequencetools.repeat_sequence_elements_n_times_each), 1804
replace_leaves_in_expr_with_skips() (in module abjad.tools.skiptools.replace_leaves_in_expr_with_skips), 1314
repeat_sequence_n_times() (in module abjad.tools.sequencetools.repeat_sequence_n_times), 1804
replace_n_edge_elements_in_container_with_big_endian_rests()
repeat_sequence_to_length() (in module abjad.tools.sequencetools.repeat_sequence_to_length), 325
1805
325
replace_n_edge_elements_in_container_with_big_endian_rests()
repeat_sequence_to_weight_at_least() (in module abjad.tools.sequencetools.repeat_sequence_to_weight_at_least), 1805
326
326
replace_n_edge_elements_in_container_with_rests()
repeat_sequence_to_weight_at_most() (in module abjad.tools.sequencetools.repeat_sequence_to_weight_at_most), 1805
327
327
replace_n_edge_elements_in_container_with_rests()
repeat_sequence_to_weight_exactly() (in module abjad.tools.sequencetools.repeat_sequence_to_weight_exactly), 1805
328
328
replace_sequence_elements_cyclically_with_new_material()
replace_components_with_children_of_components() (in module abjad.tools.sequencetools.replace_sequence_elements_cyclically_with_new_material), 806
282
replace_smaller_left_half_of_elements_in_container_with_big_endian_rests()
replace_contents_of_measures_in_expr() (in module abjad.tools.measuretools.replace_contents_of_measures_in_expr), 961
328
328
replace_smaller_left_half_of_elements_in_container_with_big_endian_rests()
replace_contents_of_target_container_with_contents_of_source_container() (in module abjad.tools.containertools.replace_contents_of_target_container_with_contents_of_source_container), 322
328
328
replace_smaller_right_half_of_elements_in_container_with_big_endian_rests()
replace_larger_left_half_of_elements_in_container_with_big_endian_rests() (in module abjad.tools.containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests), 323
329
329
replace_smaller_right_half_of_elements_in_container_with_big_endian_rests()
replace_larger_left_half_of_elements_in_container_with_big_endian_rests() (in module abjad.tools.containertools.replace_larger_left_half_of_elements_in_container_with_big_endian_rests), 323
330
330
replace_smaller_right_half_of_elements_in_container_with_big_endian_rests()
replace_larger_right_half_of_elements_in_container_with_big_endian_rests() (in module abjad.tools.containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests), 1955
324
324
replace_smaller_right_half_of_elements_in_container_with_big_endian_rests()
replace_larger_right_half_of_elements_in_container_with_big_endian_rests() (in module abjad.tools.containertools.replace_larger_right_half_of_elements_in_container_with_big_endian_rests), 1958
324
324

```


report() (abjad.tools.wellformednesstools.DuplicateIdCheck.DuplicateIdCheck.DuplicateIdCheck, 1820
method), 1960
report() (abjad.tools.wellformednesstools.EmptyContainerCheck.EmptyContainerCheck.EmptyContainerCheck ab-
method), 1962
report() (abjad.tools.wellformednesstools.IntermarkedHairpinCheck.IntermarkedHairpinCheck.IntermarkedHairpinCheck
method), 1964
report() (abjad.tools.wellformednesstools.MisduratedMeasureCheck.MisduratedMeasureCheck.MisduratedMeasureCheck
method), 1967
report() (abjad.tools.wellformednesstools.MisfilledMeasureCheck.MisfilledMeasureCheck.MisfilledMeasureCheck
method), 1969
report() (abjad.tools.wellformednesstools.MispitchedTieCheck.MispitchedTieCheck.MispitchedTieCheck overlaps_between_nonoverlap
method), 1971
report() (abjad.tools.wellformednesstools.MisrepresentedFlagCheck.MisrepresentedFlagCheck.MisrepresentedFlagCheck remainders_le
method), 1973
report() (abjad.tools.wellformednesstools.MissingParentCheck.MissingParentCheck.MissingParentCheck overlaps_between_nonoverlap
method), 1976
report() (abjad.tools.wellformednesstools.NestedMeasureCheck.NestedMeasureCheck.NestedMeasureCheck flats()
method), 1978
report() (abjad.tools.wellformednesstools.OverlappingBeamCheck.OverlappingBeamCheck.OverlappingBeamCheck pitches_in_expr_w
method), 1980
report() (abjad.tools.wellformednesstools.OverlappingGlissandoCheck.OverlappingGlissandoCheck.OverlappingGlissandoCheck
method), 1982
report() (abjad.tools.wellformednesstools.OverlappingOctavationCheck.OverlappingOctavationCheck.OverlappingOctavationCheck
method), 1985
report() (abjad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck.ShortHairpinCheck), 1218
method), 1987
report_as_string_format_contributions_of_all_spanners_attached_to_(in module ab-
jad.tools.sequencetools.report_as_string_format_contributions_of_all_spanners_attached_to_(in module ab-
1519
report_as_string_format_contributions_of_all_spanners_attached_to_(in module ab-
1519
report_component_format_contributions_as_string() (in module ab-
1807
report_container_modifications_as_string() (in module ab-
283
report_meter_distribution_as_string() (in module ab-
962
representative_boolean_train (ab-
jad.tools.sievetools.ResidueClassExpression.ResidueClassExpression
attribute), 1822
representative_congruent_bases (ab-
jad.tools.sievetools.ResidueClassExpression.ResidueClassExpression
attribute), 1822
requires() (in module abjad.tools.decoratortools.requires),
1632

reverse() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock attribute), 811

reverse() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock attribute), 816

reverse() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile attribute), 2102

reverse() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock attribute), 2075

reverse() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory attribute), 897

reverse() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping attribute), 1154

reverse() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory attribute), 1161

reverse() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory attribute), 1168

reverse() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory attribute), 1265

reverse() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList attribute), 1744

reverse_contents_of_container() (in module abjad.tools.containertools.reverse_contents_of_container), 331

reverse_sequence() (in module abjad.tools.sequencetools.reverse_sequence), 1807

reverse_sequence_elements() (in module abjad.tools.sequencetools.reverse_sequence_elements), 1807

rewrite_rational_under_new_tempo() (in module abjad.tools.durationtools.rewrite_rational_under_new_tempo), 1683

RhythmicStaff (class in abjad.tools.stafftools.RhythmicStaff.RhythmicStaff), 1520

roman_numeral_string (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 2098

root (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree attribute), 1753

root (abjad.tools.sequencetools.Tree.Tree.Tree attribute), 1768

root (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass attribute), 2075

root_class (abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph.InheritanceGraph attribute), 1649

root_nodes (abjad.tools.datastructuretools.Digraph.Digraph.Digraph attribute), 1622

root_package_name (abjad.tools.documentationtools.ClassCrawler.ClassCrawler.ClassCrawler attribute), 1637

root_package_name (abjad.tools.documentationtools.FunctionCrawler.FunctionCrawler.FunctionCrawler attribute), 1644

round_interval_bounds_to_nearest_multiple_of_rational() (in module abjad.tools.timeintervaltools.round_interval_bounds_to_nearest_multiple_of_rational), 1889

row_index (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell attribute), 1733

row_index (abjad.tools.pitcharraytools.PitchArrayRow.PitchArrayRow.PitchArrayRow attribute), 1738

rows (abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray attribute), 1729

rows (abjad.tools.sequencetools.CyclicMatrix.CyclicMatrix.CyclicMatrix attribute), 1748

rows (abjad.tools.sequencetools.Matrix.Matrix.Matrix attribute), 1764

save_last_ly_as() (in module abjad.tools.io.save_last_ly_as), 1694

save_last_pdf_as() (in module abjad.tools.io.save_last_pdf_as), 1694

scale (abjad.tools.tonalitytools.Scale.Scale.Scale attribute), 2094

Index 2293

- semitones (abjad.tools.pitchtools.HarmonicChromaticInterval set (abjad.tools.pitchtools.HarmonicChromaticInterval attribute), 992
- semitones (abjad.tools.pitchtools.HarmonicCounterpointInterval set (abjad.tools.pitchtools.HarmonicCounterpointInterval attribute), 1009
- semitones (abjad.tools.pitchtools.HarmonicDiatonicInterval set (abjad.tools.pitchtools.HarmonicDiatonicInterval attribute), 1014
- semitones (abjad.tools.pitchtools.MelodicChromaticInterval set (abjad.tools.pitchtools.MelodicChromaticInterval attribute), 1055
- semitones (abjad.tools.pitchtools.MelodicCounterpointInterval set (abjad.tools.pitchtools.MelodicCounterpointInterval attribute), 1076
- semitones (abjad.tools.pitchtools.MelodicDiatonicInterval.MelodicDiatonicInterval attribute), 1081
- send_signal() (abjad.tools.documentationtools.Pipe.Pipe.Pipe method), 1656
- set (abjad.tools.chordtools.Chord.Chord.Chord attribute), 225
- set (abjad.tools.componenttools.Component.Component.Component attribute), 236
- set (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 295
- set (abjad.tools.containertools.Container.Container.Container attribute), 303
- set (abjad.tools.contexttools.Context.Context.Context attribute), 350
- set (abjad.tools.gracetools.Grace.Grace.Grace attribute), 417
- set (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 769
- set (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock attribute), 820
- set (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics attribute), 2018
- set (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender attribute), 2026
- set (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2031
- set (abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics attribute), 2045
- set (abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace attribute), 2035
- set (abjad.tools.lyrictools.LyricText.LyricText.LyricText attribute), 2040
- set (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 907
- set (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 918
- set (abjad.tools.measuretools.Measure.Measure.Measure attribute), 928
- set (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 967
- set (abjad.tools.notetools.Note.Note.Note attribute), 973
- set (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1214
- set (abjad.tools.resttools.Rest.Rest.Rest attribute), 1219
- set (abjad.tools.chordtools.Chord.Chord.Chord attribute), 225
- set (abjad.tools.componenttools.Component.Component.Component attribute), 236
- set (abjad.tools.containertools.Cluster.Cluster.Cluster attribute), 295
- set (abjad.tools.containertools.Container.Container.Container attribute), 303
- set (abjad.tools.contexttools.Context.Context.Context attribute), 350
- set (abjad.tools.gracetools.Grace.Grace.Grace attribute), 417
- set (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 769
- set (abjad.tools.lilypondfiletools.ContextBlock.ContextBlock.ContextBlock attribute), 820
- set (abjad.tools.lyrictools.AddLyrics.AddLyrics.AddLyrics attribute), 2018
- set (abjad.tools.lyrictools.LyricExtender.LyricExtender.LyricExtender attribute), 2026
- set (abjad.tools.lyrictools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2031
- set (abjad.tools.lyrictools.Lyrics.Lyrics.Lyrics attribute), 2045
- set (abjad.tools.lyrictools.LyricSpace.LyricSpace.LyricSpace attribute), 2035
- set (abjad.tools.lyrictools.LyricText.LyricText.LyricText attribute), 2040
- set (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure attribute), 907
- set (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 918
- set (abjad.tools.measuretools.Measure.Measure.Measure attribute), 928
- set (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 967
- set (abjad.tools.notetools.Note.Note.Note attribute), 973
- set (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest attribute), 1214
- set (abjad.tools.resttools.Rest.Rest.Rest attribute), 1219
- set (abjad.tools.pitchtools.HarmonicChromaticInterval attribute), 1251
- set (abjad.tools.pitchtools.HarmonicCounterpointInterval attribute), 1272
- set (abjad.tools.pitchtools.HarmonicDiatonicInterval attribute), 1282
- set (abjad.tools.pitchtools.MelodicChromaticInterval attribute), 1293
- set (abjad.tools.pitchtools.MelodicCounterpointInterval attribute), 1309
- set (abjad.tools.pitchtools.MelodicDiatonicInterval attribute), 1326
- set (abjad.tools.spannertools.BeamSpanner.BeamSpanner.BeamSpanner attribute), 1334
- set (abjad.tools.spannertools.BracketSpanner.BracketSpanner.BracketSpanner attribute), 1334
- set (abjad.tools.spannertools.ComplexBeamSpanner.ComplexBeamSpanner attribute), 1343
- set (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1353
- set (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1363
- set (abjad.tools.spannertools.DynamicTextSpanner.DynamicTextSpanner.DynamicTextSpanner attribute), 1373
- set (abjad.tools.spannertools.GlissandoSpanner.GlissandoSpanner.GlissandoSpanner attribute), 1381
- set (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1389
- set (abjad.tools.spannertools.HiddenStaffSpanner.HiddenStaffSpanner.HiddenStaffSpanner attribute), 1398
- set (abjad.tools.spannertools.HorizontalBracketSpanner.HorizontalBracketSpanner.HorizontalBracketSpanner attribute), 1405
- set (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner attribute), 1413
- set (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner attribute), 1422
- set (abjad.tools.spannertools.MultipartBeamSpanner.MultipartBeamSpanner.MultipartBeamSpanner attribute), 1431
- set (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner attribute), 1439
- set (abjad.tools.spannertools.PhrasingSlurSpanner.PhrasingSlurSpanner.PhrasingSlurSpanner attribute), 1447
- set (abjad.tools.spannertools.PianoPedalSpanner.PianoPedalSpanner.PianoPedalSpanner attribute), 1454
- set (abjad.tools.spannertools.SlurSpanner.SlurSpanner.SlurSpanner attribute), 1462
- set (abjad.tools.spannertools.Spanner.Spanner.Spanner attribute), 1469
- set (abjad.tools.spannertools.StaffLinesSpanner.StaffLinesSpanner.StaffLinesSpanner attribute), 1476
- set (abjad.tools.spannertools.TextScriptSpanner.TextScriptSpanner.TextScriptSpanner attribute), 1484
- set (abjad.tools.spannertools.TextSpanner.TextSpanner.TextSpanner attribute), 1491

[set \(abjad.tools.spannertools.TrillSpanner.TrillSpanner.TrillSpanner attribute\), 1499](#)
[set \(abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute\), 1523](#)
[set \(abjad.tools.stafftools.Staff.Staff.Staff attribute\), 1533](#)
[set \(abjad.tools.tietools.TieSpanner.TieSpanner.TieSpanner attribute\), 1545](#)
[set \(abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute\), 1567](#)
[set \(abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute\), 1579](#)
[set \(abjad.tools.voicetools.Voice.Voice.Voice attribute\), 1600](#)
[set_accidental_style_on_sequential_contexts_in_expr\(\) \(in module abjad.tools.contexttools.set_accidental_style_on_sequential_contexts_in_expr\), 414](#)
[set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr\(\) \(in module abjad.tools.pitchtools.set_ascending_named_chromatic_pitches_on_nontied_pitched_components_in_expr\), 1209](#)
[set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr\(\) \(in module abjad.tools.pitchtools.set_ascending_named_diatonic_pitches_on_nontied_pitched_components_in_expr\), 1209](#)
[set_container_multiplier\(\) \(in module abjad.tools.containertools.set_container_multiplier\), 333](#)
[set_default_accidental_spelling\(\) \(in module abjad.tools.configurationtools.set_default_accidental_spelling\), 1620](#)
[set_denominator_of_tuplets_in_expr_to_at_least\(\) \(in module abjad.tools.tuplettools.set_denominator_of_tuplets_in_expr_to_at_least\), 1596](#)
[set_line_breaks_cyclically_by_line_duration_ge\(\) \(in module abjad.tools.layouttools.set_line_breaks_cyclically_by_line_duration_ge\), 1700](#)
[set_line_breaks_cyclically_by_line_duration_in_seconds_ge\(\) \(in module abjad.tools.layouttools.set_line_breaks_cyclically_by_line_duration_in_seconds_ge\), 1702](#)
[set_measure_denominator_and_adjust_numerator\(\) \(in module abjad.tools.measuretools.set_measure_denominator_and_adjust_numerator\), 963](#)
[set_preprolated_leaf_duration\(\) \(in module abjad.tools.leafertools.set_preprolated_leaf_duration\), 805](#)
[set_vertical_positioning_pitch_on_rest\(\) \(in module abjad.tools.resttools.set_vertical_positioning_pitch_on_rest\), 1225](#)
[setdefault\(\) \(abjad.tools.datastructuretools.ImmutableDictionary.ImmutableDictionary method\), 1625](#)
[setdefault\(\) \(abjad.tools.documentationtools.InheritanceGraph.InheritanceGraph method\), 1650](#)
[setdefault\(\) \(abjad.tools.pitchtools.HarmonicChromaticIntervalClassVector.HarmonicChromaticIntervalClassVector method\), 998](#)
[setdefault\(\) \(abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassVector.InversionEquivalentChromaticIntervalClassVector method\), 1044](#)
[setdefault\(\) \(abjad.tools.pitchtools.InversionEquivalentDiatonicIntervalClassVector.InversionEquivalentDiatonicIntervalClassVector method\), 1051](#)
[setdefault\(\) \(abjad.tools.pitchtools.MelodicChromaticIntervalClassVector.MelodicChromaticIntervalClassVector method\), 1064](#)
[setdefault\(\) \(abjad.tools.pitchtools.NamedChromaticPitchVector.NamedChromaticPitchVector method\), 1118](#)
[setdefault\(\) \(abjad.tools.pitchtools.NumberedChromaticPitchClassVector.NumberedChromaticPitchClassVector method\), 1145](#)
[setdefault\(\) \(abjad.tools.quantizationtools.QGridSearchTree.QGridSearchTree method\), 2064](#)
[setdefault\(\) \(abjad.tools.quantizationtools.QGridTempoLookup.QGridTempoLookup method\), 2149](#)
[setdefault\(\) \(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method\), 1842](#)
[setdefault\(\) \(abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method\), 1834](#)
[shape_string \(abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner attribute\), 1344](#)
[shape_string \(abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner attribute\), 1354](#)
[shape_string \(abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute\), 1390](#)
[shift_aggregate_offset_by_rational\(\) \(in module abjad.tools.timeintervaltools.shift_aggregate_offset_by_rational\), 1892](#)
[shift_aggregate_offset_to_rational\(\) \(in module abjad.tools.timeintervaltools.shift_aggregate_offset_to_rational\), 1892](#)
[shift_by_rational\(\) \(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method\), 1842](#)
[shift_by_rational\(\) \(abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method\), 1834](#)
[shift_by_rational\(\) \(abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method\), 1838](#)
[shift_by_rational\(\) \(abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method\), 1854](#)
[shift_by_rational\(\) \(abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method\), 1876](#)
[shift_to_rational\(\) \(abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method\), 1842](#)

[shift_to_rational\(\)](#) (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.[shift_to_rational\(\)](#) method), [1834](#)
[shift_to_rational\(\)](#) (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.[shift_to_rational\(\)](#) method), [1838](#)
[short_instrument_name](#) (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.[short_instrument_name](#) attribute), [536](#)
[short_instrument_name](#) (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.[short_instrument_name](#) attribute), [1877](#)
[short_instrument_name](#) (abjad.tools.contexttools.InstrumentMark.InstrumentMark.[short_instrument_name](#) attribute), [366](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.[short_instrument_name](#) attribute), [550](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Accordion.Accordion.[short_instrument_name](#) attribute), [432](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.[short_instrument_name](#) attribute), [564](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.AltoFlute.AltoFlute.[short_instrument_name](#) attribute), [438](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.[short_instrument_name](#) attribute), [557](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.[short_instrument_name](#) attribute), [445](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.[short_instrument_name](#) attribute), [571](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.[short_instrument_name](#) attribute), [452](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.[short_instrument_name](#) attribute), [578](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BariSaxophone.BariSaxophone.[short_instrument_name](#) attribute), [466](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.[short_instrument_name](#) attribute), [585](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BassoonVoice.BassoonVoice.[short_instrument_name](#) attribute), [473](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Flute.Flute.Flute.[short_instrument_name](#) attribute), [592](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BassClarinet.BassClarinet.[short_instrument_name](#) attribute), [480](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.[short_instrument_name](#) attribute), [599](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.[short_instrument_name](#) attribute), [487](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute.[short_instrument_name](#) attribute), [606](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon.[short_instrument_name](#) attribute), [515](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Guitar.Guitar.Guitar.[short_instrument_name](#) attribute), [613](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.[short_instrument_name](#) attribute), [494](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Harp.Harp.Harp.[short_instrument_name](#) attribute), [620](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BassTrombone.BassTrombone.[short_instrument_name](#) attribute), [501](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.[short_instrument_name](#) attribute), [627](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Marimba.Marimba.Marimba.[short_instrument_name](#) attribute), [508](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.[short_instrument_name](#) attribute), [639](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.[short_instrument_name](#) attribute), [459](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Cello.Cello.Cello.[short_instrument_name](#) attribute), [646](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Oboe.Oboe.Oboe.[short_instrument_name](#) attribute), [522](#)
[short_instrument_name](#) (abjad.tools.instrumenttools.Oboe.Oboe.Oboe.[short_instrument_name](#) attribute), [653](#)

short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Piano.Piano.Piano		jad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone), 660	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo		jad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone), 667	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone), 674		jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone), 674	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone), 681		jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice), 681	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice), 688		jad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet), 688	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone), 695		jad.tools.instrumenttools.BassFlute.BassFlute.BassFlute), 695	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone), 702		jad.tools.instrumenttools.Bassoon.Bassoon.Bassoon), 702	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice), 709		jad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone), 709	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet), 716		jad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone), 716	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Tuba.Tuba.Tuba), 723		jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice), 723	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion), 730		jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet), 730	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone), 737		jad.tools.instrumenttools.Cello.Cello.Cello), 737	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Viola.Viola.Viola), 744		jad.tools.instrumenttools.ClarineteInA.ClarineteInA.ClarineteInA), 744	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Violin.Violin.Violin), 751		jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 751	
short_instrument_name	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone), 758		jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet), 758	
short_instrument_name_markup	(ab-	short_instrument_name_markup	(ab-
jad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark), 366		jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 366	
short_instrument_name_markup	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.Accordion.Accordion.Accordion), 432		jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 432	
short_instrument_name_markup	(ab-	short_instrument_name_markup	(ab-
jad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute), 439		jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 439	

`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.CornettoVoice`
 attribute), 572 attribute), 703
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet`
 attribute), 579 attribute), 710
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn`
 attribute), 586 attribute), 717
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.Flute.Flute.Flute`
 attribute), 593 attribute), 724
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn`
 attribute), 599 attribute), 731
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel`
 attribute), 607 attribute), 738
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.Guitar.Guitar.Guitar`
 attribute), 614 attribute), 745
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.Harp.Harp.Harp`
 attribute), 621 attribute), 752
`short_instrument_name_markup` (ab- `short_instrument_name_markup` (ab-
`jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord`
 attribute), 628 attribute), 759
`short_instrument_name_markup` (ab- `ShortHairpinCheck` (class in ab-
`jad.tools.instrumenttools.Marimba.Marimba.Marimba`
 attribute), 640 `jad.tools.wellformednesstools.ShortHairpinCheck.ShortHairpinCheck`
 1987
`short_instrument_name_markup` (ab- `show()` (in module `abjad.tools.iotools.show`), 1694
`jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice`
 attribute), 647 `jad.tools.leafstools.show_leaves`), 806
`short_instrument_name_markup` (ab- `sign()` (in module `abjad.tools.mathtools.sign`), 1727
`jad.tools.instrumenttools.Oboe.Oboe.Oboe`
 attribute), 654 `SignalFilledTimeTokenMaker` (class in ab-
`jad.tools.instrumenttools.Piano.Piano.Piano`
 attribute), 661 `jad.tools.timetokentools.SignalFilledTimeTokenMaker.SignalFilledTimeTokenMaker`
 1926
`short_instrument_name_markup` (ab- `signature` (`abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval`
`jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo`
 attribute), 668 attribute), 1840
`short_instrument_name_markup` (ab- `signature` (`abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin`
`jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone`
 attribute), 675 attribute), 1832
`short_instrument_name_markup` (ab- `signature` (`abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin`
`jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone`
 attribute), 682 attribute), 1837
`short_instrument_name_markup` (ab- `signature` (`abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree`
`jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice`
 attribute), 689 attribute), 1846
`short_instrument_name_markup` (ab- `signature` (`abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary`
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone`
 attribute), 696 attribute), 1868
`short_instrument_name_markup` (ab- `size` (`abjad.tools.pitcharraytools.PitchArray.PitchArray.PitchArray`
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone`
 attribute), 696 attribute), 1729
`short_instrument_name_markup` (ab- `skip` (`abjad.tools.skiptools.Skip.Skip.Skip`), 1308
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone`
 attribute), 696 `slope` (`abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment`
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone`
 attribute), 696 attribute), 1068
`short_instrument_name_markup` (ab- `SlurSpanner` (class in ab-
`jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone`
 attribute), 696 `jad.tools.spannertools.SlurSpanner.SlurSpanner`),

1460	solutions (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver (attribute), 1996	sort() (abjad.tools.constrainttools.FixedLengthStreamSolver.FixedLengthStreamSolver (attribute), 439
	solutions (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver (attribute), 2014	sort() (abjad.tools.constrainttools.VariableLengthStreamSolver.VariableLengthStreamSolver (attribute), 481
	SopranoSaxophone (class in abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone), 453	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone (attribute), 1996
671	SopranoSaxophone (class in abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone), 467	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone (attribute), 453
678	SopranoVoice (class in abjad.tools.instrumenttools.SopranoVoice.SopranoVoice), 474	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone (attribute), 467
685	sort() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory (method), 345	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice (attribute), 474
	sort() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory (method), 384	sort() (abjad.tools.contexttools.ClefMarkInventory.ClefMarkInventory.ClefMarkInventory (method), 481
	sort() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory (method), 1629	sort() (abjad.tools.contexttools.TempoMarkInventory.TempoMarkInventory.TempoMarkInventory (method), 488
	sort() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory (method), 632	sort() (abjad.tools.datastructuretools.ObjectInventory.ObjectInventory.ObjectInventory (method), 1629
	sort() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock (method), 811	sort() (abjad.tools.instrumenttools.InstrumentInventory.InstrumentInventory.InstrumentInventory (method), 516
	sort() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock (method), 816	sort() (abjad.tools.lilypondfiletools.BookBlock.BookBlock.BookBlock (method), 811
	sort() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile (method), 828	sort() (abjad.tools.lilypondfiletools.BookpartBlock.BookpartBlock.BookpartBlock (method), 816
	sort() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock (method), 840	sort() (abjad.tools.lilypondfiletools.LilyPondFile.LilyPondFile.LilyPondFile (method), 828
	sort() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory (method), 897	sort() (abjad.tools.lilypondfiletools.ScoreBlock.ScoreBlock.ScoreBlock (method), 840
	sort() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping (method), 1154	sort() (abjad.tools.markuptools.MarkupInventory.MarkupInventory.MarkupInventory (method), 1009
	sort() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory (method), 1161	sort() (abjad.tools.pitchtools.OctaveTranspositionMapping.OctaveTranspositionMapping.OctaveTranspositionMapping (method), 460
	sort() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory (method), 1168	sort() (abjad.tools.pitchtools.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory.OctaveTranspositionMappingInventory (method), 1161
	sort() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory (method), 1265	sort() (abjad.tools.pitchtools.PitchRangeInventory.PitchRangeInventory.PitchRangeInventory (method), 1168
	sort() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList (method), 1744	sort() (abjad.tools.scoretools.PerformerInventory.PerformerInventory.PerformerInventory (method), 530
	SortableAttributeEqualityAbjadObject (class in abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject), 1617	sort() (abjad.tools.sequencetools.CyclicList.CyclicList.CyclicList (method), 1744
	sounding_pitch (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic (attribute), 967	sort() (abjad.tools.abctools.SortableAttributeEqualityAbjadObject.SortableAttributeEqualityAbjadObject), 1617
	sounding_pitch (abjad.tools.notetools.Note.Note.Note (attribute), 973	sounding_pitch (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic (attribute), 967
	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Accordion.Accordion.Accordion (attribute), 432	sounding_pitch_of_fingered_middle_c (abjad.tools.notetools.Note.Note.Note (attribute), 973
	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute (attribute), 432	sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass (attribute), 537
		sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet (attribute), 551
		sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute (attribute), 551
		sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon (attribute), 565
		sounding_pitch_of_fingered_middle_c (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone (attribute), 565

attribute), 558

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice.jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone), 572

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet.jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice), 579

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn.jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet), 586

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.Flute.Flute.Flute jad.tools.instrumenttools.Tuba.Tuba.Tuba), 593

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn.jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion), 600

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel.jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone), 607

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.Guitar.Guitar.Guitar jad.tools.instrumenttools.Viola.Viola.Viola), 614

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.Harp.Harp.Harp jad.tools.instrumenttools.Violin.Violin.Violin), 621

sounding_pitch_of_fingered_middle_c (ab- sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord.jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone), 628

sounding_pitch_of_fingered_middle_c (ab- sounding_pitches (abjad.tools.chordtools.Chord.Chord.Chord
jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 225
attribute), 640 source_pitch_range (ab-
jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 647

sounding_pitch_of_fingered_middle_c (ab- (in module ab-
jad.tools.instrumenttools.Oboe.Oboe.Oboe jad.tools.iotools.space_delimited_lowercase_to_uppercamelcase(),
attribute), 654 1695

sounding_pitch_of_fingered_middle_c (ab- SpacingIndication (class in ab-
jad.tools.instrumenttools.Piano.Piano.Piano jad.tools.layouttools.SpacingIndication.SpacingIndication),
attribute), 661 1698

sounding_pitch_of_fingered_middle_c (ab- span (abjad.tools.spannertools.DuratedComplexBeamSpanner.DuratedComplexBeamSpanner), 1365
jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 668

sounding_pitch_of_fingered_middle_c (ab- span (abjad.tools.spannertools.MeasuredComplexBeamSpanner.MeasuredComplexBeamSpanner), 1415
jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone jad.tools.spannertools.Spanner.Spanner),
attribute), 675 1468

sounding_pitch_of_fingered_middle_c (ab- spanned_saxophone_chordtools.Chord.Chord.Chord at-
jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone attribute), 682

sounding_pitch_of_fingered_middle_c (ab- spanners (abjad.tools.componenttools.Component.Component.Component
jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 236
attribute), 689 spanners (abjad.tools.containertools.Cluster.Cluster.Cluster
attribute), 295

sounding_pitch_of_fingered_middle_c (ab-
jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone

spanners (abjad.tools.containertools.Container.Container.Container.Container attribute), 303

spanners (abjad.tools.contexttools.Context.Context.Context.Context attribute), 350

spanners (abjad.tools.gracetools.Grace.Grace.Grace attribute), 418

spanners (abjad.tools.leaftools.Leaf.Leaf.Leaf attribute), 769

spanners (abjad.tools.lyricstools.AddLyrics.AddLyrics.AddLyrics attribute), 2018

spanners (abjad.tools.lyricstools.LyricExtender.LyricExtender.LyricExtender attribute), 2026

spanners (abjad.tools.lyricstools.LyricHyphen.LyricHyphen.LyricHyphen attribute), 2031

spanners (abjad.tools.lyricstools.Lyrics.Lyrics.Lyrics attribute), 2045

spanners (abjad.tools.lyricstools.LyricSpace.LyricSpace.LyricSpace attribute), 2035

spanners (abjad.tools.lyricstools.LyricText.LyricText.LyricText attribute), 2040

spanners (abjad.tools.measuretools.AnonymousMeasure.AnonymousMeasure.AnonymousMeasure attribute), 907

spanners (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure.DynamicMeasure attribute), 918

spanners (abjad.tools.measuretools.Measure.Measure.Measure attribute), 928

spanners (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 967

spanners (abjad.tools.notetools.Note.Note.Note attribute), 973

spanners (abjad.tools.resttools.MultiMeasureRest.MultiMeasureRest.MultiMeasureRest attribute), 1215

spanners (abjad.tools.resttools.Rest.Rest.Rest attribute), 1219

spanners (abjad.tools.scoretools.GrandStaff.GrandStaff.GrandStaff attribute), 1251

spanners (abjad.tools.scoretools.PianoStaff.PianoStaff.PianoStaff attribute), 1272

spanners (abjad.tools.scoretools.Score.Score.Score attribute), 1282

spanners (abjad.tools.scoretools.StaffGroup.StaffGroup.StaffGroup attribute), 1293

spanners (abjad.tools.skiptools.Skip.Skip.Skip attribute), 1309

spanners (abjad.tools.stafftools.RhythmicStaff.RhythmicStaff.RhythmicStaff attribute), 1523

spanners (abjad.tools.stafftools.Staff.Staff.Staff attribute), 1533

spanners (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet attribute), 1567

spanners (abjad.tools.tuplettools.Tuplet.Tuplet.Tuplet attribute), 1579

spanners (abjad.tools.voicetools.Voice.Voice.Voice attribute), 1601

special_methods (abjad.tools.documentationtools.ClassDocumenter.ClassDocumenter attribute), 1640

splice_new_elements_between_sequence_elements() (in module abjad.tools.sequencetools.splice_new_elements_between_sequence_elements), 1808

split_at_rationals() (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval method), 1842

split_at_rationals() (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin method), 1834

split_at_rationals() (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin method), 1838

split_at_rationals() (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree method), 1855

split_at_rationals() (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary method), 1857

split_component_at_prolated_duration_and_do_not_fracture_crossing_spanners() (in module abjad.tools.componenttools.split_component_at_prolated_duration_and_do_not_fracture_crossing_spanners), 283

split_component_at_prolated_duration_and_fracture_crossing_spanners() (in module abjad.tools.componenttools.split_component_at_prolated_duration_and_fracture_crossing_spanners), 284

split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossing_spanners() (in module abjad.tools.componenttools.split_components_cyclically_by_prolated_durations_and_do_not_fracture_crossing_spanners), 285

split_components_cyclically_by_prolated_durations_and_fracture_crossing_spanners() (in module abjad.tools.componenttools.split_components_cyclically_by_prolated_durations_and_fracture_crossing_spanners), 286

split_components_once_by_prolated_durations_and_do_not_fracture_crossing_spanners() (in module abjad.tools.componenttools.split_components_once_by_prolated_durations_and_do_not_fracture_crossing_spanners), 287

split_container_at_index_and_do_not_fracture_crossing_spanners() (in module abjad.tools.containertools.split_container_at_index_and_do_not_fracture_crossing_spanners), 334

[split_container_cyclically_by_counts_and_do_not_fracture_staves\(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark\)](#), 335
 (in module ab- attribute), 375
[jad.tools.containertools.split_container_cyclically_by_counts_and_do_not_fracture_staves\(abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark\)](#), 335
[split_container_cyclically_by_counts_and_fracture_crossing_spanners\(\)](#), 1014
 (in module ab- staff_spaces (abjad.tools.pitchtools.HarmonicDiatonicInterval.HarmonicDiatonicInterval.HarmonicDiatonicInterval)), 1084
[jad.tools.containertools.split_container_cyclically_by_counts_and_fracture_crossing_spanners\(\)](#), 336
 StaffChangeMark (class in ab-
[split_container_once_by_counts_and_do_not_fracture_crossing_spanners\(\)](#), 373
 (in module ab-
[jad.tools.containertools.split_container_once_by_counts_and_do_not_fracture_crossing_spanners\(\)](#), 337
 StaffGroup (class in ab-
[split_container_once_by_counts_and_fracture_crossing_spanners\(\)](#), 1290
 (in module ab- StaffLinesSpanner (class in ab-
[jad.tools.containertools.split_container_once_by_counts_and_fracture_crossing_spanners\(\)](#), 338
 StaffLinesSpanner), 1474
[split_intervals_at_rationals\(\)](#) (in module ab- start (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner)), 1439
[jad.tools.timeintervaltools.split_intervals_at_rationals\(\)](#), 1893
 start (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval)), 1840
[split_leaf_at_prolated_duration_and_rest_right_half\(\)](#), 832
 (in module ab- start (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin)), 807
[jad.tools.leaftools.split_leaf_at_prolated_duration_and_rest_right_half\(\)](#), 807
 start (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin)), 1426
[split_on_bar\(\)](#) (abjad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method), 1426
 start (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree)), 1847
[split_sequence_cyclically_by_weights_with_overhang\(\)](#), 1808
 (in module ab- start (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary)), 1808
[jad.tools.sequencetools.split_sequence_cyclically_by_weights_with_overhang\(\)](#), 1808
 start (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator)), 2100
[split_sequence_cyclically_by_weights_without_overhang\(\)](#), 1809
 (in module ab- start_cells (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn)), 1809
[jad.tools.sequencetools.split_sequence_cyclically_by_weights_without_overhang\(\)](#), 1809
 start_component (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark)), 340
[split_sequence_extended_to_weights_with_overhang\(\)](#), 1809
 (in module ab- start_component (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark.ContextMark)), 361
[jad.tools.sequencetools.split_sequence_extended_to_weights_with_overhang\(\)](#), 1809
 start_component (abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark.DynamicMark)), 361
[split_sequence_extended_to_weights_without_overhang\(\)](#), 1809
 (in module ab- start_component (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark.InstrumentMark)), 361
[jad.tools.sequencetools.split_sequence_extended_to_weights_without_overhang\(\)](#), 1809
 start_component (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark.KeySignatureMark)), 370
[split_sequence_once_by_weights_with_overhang\(\)](#), 1809
 (in module ab- start_component (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark.StaffChangeMark)), 379
[jad.tools.sequencetools.split_sequence_once_by_weights_with_overhang\(\)](#), 1809
 start_component (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark.TempoMark)), 379
[split_sequence_once_by_weights_without_overhang\(\)](#), 1810
 (in module ab- start_component (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark)), 430
[jad.tools.sequencetools.split_sequence_once_by_weights_without_overhang\(\)](#), 1810
 start_component (abjad.tools.instrumenttools.Accordion.Accordion.Accordion.Accordion)), 437
[splitting_condition\(\)](#) (ab- attribute), 437
[jad.tools.spannertools.MetricGridSpanner.MetricGridSpanner.MetricGridSpanner method\), 1426](#)
[spread](#) (abjad.tools.pitchtools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 1068
 start (abjad.tools.segmenttools.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment.MelodicChromaticIntervalSegment), 444
 attribute), 444

start_component (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice), 451

start_component (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone), 465

start_component (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice.instrumenttools.Piano.Piano.Piano), 472

start_component (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet.instrumenttools.Piccolo.Piccolo.Piccolo), 479

start_component (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute), 486

start_component (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon), 514

start_component (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone), 493

start_component (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone), 500

start_component (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice), 507

start_component (abjad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet), 458

start_component (abjad.tools.instrumenttools.Cello.Cello.Cello), 521

start_component (abjad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA), 528

start_component (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass), 535

start_component (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet), 542

start_component (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute), 549

start_component (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon), 563

start_component (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone), 556

start_component (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice), 570

start_component (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet), 577

start_component (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn), 584

start_component (abjad.tools.instrumenttools.Flute.Flute.Flute), 591

start_component (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn), 598

start_component (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel), 605

start_component (abjad.tools.instrumenttools.Guitar.Guitar.Guitar), 612

start_component (abjad.tools.instrumenttools.Harp.Harp.Harp), 619

start_component (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord), 626

start_component (abjad.tools.instrumenttools.Marimba.Marimba.Marimba), 638

start_component (abjad.tools.marktools.Annotation.Annotation.Annotation), 845

start_component (abjad.tools.marktools.Articulation.Articulation.Articulation), 849

start_component (abjad.tools.marktools.BarLine.BarLine.BarLine), 853

start_component (abjad.tools.marktools.BendAfter.BendAfter.BendAfter), 857

start_component (abjad.tools.marktools.LilyPondCommandMark.LilyPondCommandMark.LilyPondCommandMark), 861

start_component (abjad.tools.marktools.LilyPondComment.LilyPondComment.LilyPondComment), 864

start_component (abjad.tools.marktools.Mark.Mark.Mark), 868

start_component (abjad.tools.marktools.StemTremolo.StemTremolo.StemTremolo), 870

start_component (abjad.tools.markuptools.Markup.Markup.Markup), 891

start_component (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment), 1942

start_dynamic_string (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1344

start_dynamic_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1354

start_dynamic_string (abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate attribute), 1390

start_leaves (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

start_notes (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

start_pitch (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1165

start_pitch_is_included_in_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1165

start_pitches (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1735

StemTremolo (class in abjad.tools.marktools.StemTremolo.StemTremolo), 870

stop (abjad.tools.spannertools.OctavationSpanner.OctavationSpanner.OctavationSpanner attribute), 1439

stop (abjad.tools.timeintervaltools.TimeInterval.TimeInterval.TimeInterval attribute), 1841

stop (abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin attribute), 1833

stop (abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin.TimeIntervalMixin attribute), 1837

stop (abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree.TimeIntervalTree attribute), 1847

stop (abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary attribute), 1861

stop (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator attribute), 2100

stop_cells (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1735

stop_dynamic_string (abjad.tools.spannertools.CrescendoSpanner.CrescendoSpanner.CrescendoSpanner attribute), 1345

stop_dynamic_string (abjad.tools.spannertools.DecrescendoSpanner.DecrescendoSpanner.DecrescendoSpanner attribute), 1355

stop_dynamic_string (abjad.tools.spannertools.HairpinSpanner.HairpinSpanner.HairpinSpanner attribute), 1390

stop_pitch (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1166

stop_pitch_is_included_in_range (abjad.tools.pitchtools.PitchRange.PitchRange.PitchRange attribute), 1166

stop_pitches (abjad.tools.pitcharraytools.PitchArrayColumn.PitchArrayColumn.PitchArrayColumn attribute), 1735

string_to_strict_directory_name() (in module abjad.tools.iotools.string_to_strict_directory_name), 1695

StringQuartetScoreTemplate (class in abjad.tools.scoretemplatetools.StringQuartetScoreTemplate.StringQuartetScoreTemplate), 1390

strip_diacritics_from_binary_string() (in module abjad.tools.marktools.StripDiacriticsFromBinaryString.StripDiacriticsFromBinaryString attribute), 1695

style (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

style (abjad.tools.verticalitytools.VerticalMoment.VerticalMoment.VerticalMoment attribute), 1942

subdivide_indices() (abjad.tools.quantizationtools.QGrid.QGrid.QGrid method), 2056

submediant (abjad.tools.tonalitytools.Scale.Scale.Scale attribute), 2095

suggest_clef_for_named_chromatic_pitches() (abjad.tools.pitchtools.suggest_clef_for_named_chromatic_pitches module), 1740

sum_consecutive_sequence_elements_by_sign() (abjad.tools.sequencetools.sum_consecutive_sequence_elements_by_sign module), 1841

sum_duration_of_components_in_seconds() (abjad.tools.componenttools.sum_duration_of_components_in_seconds module), 1847

sum_preprolated_duration_of_components() (abjad.tools.componenttools.sum_preprolated_duration_of_components module), 2100

sum_prolated_duration_of_components() (in module abjad.tools.componenttools.sum_prolated_duration_of_components module), 2095

sum_sequence_elements_at_indices() (in module abjad.tools.sequencetools.sum_sequence_elements_at_indices module), 1833

suono_reale (abjad.tools.notetools.NaturalHarmonic.NaturalHarmonic.NaturalHarmonic attribute), 968

super_HairpinSpanner (abjad.tools.tonalitytools.Scale.Scale.Scale attribute), 2095

suppress_meter (abjad.tools.measuretools.DynamicMeasure.DynamicMeasure attribute), 919

suspension (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction attribute), 2102

SuspensionIndicator (class in abjad.tools.contexttools.DynamicMark.DynamicMark.DynamicMark) (attribute), 361
 jad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator (attribute), 361
 2100 target_context (abjad.tools.contexttools.InstrumentMark.InstrumentMark.InstrumentMark) (attribute), 365
 symbolic_accidental_string (abjad.tools.pitchtools.Accidental.Accidental.Accidental) (attribute), 990
 target_context (abjad.tools.contexttools.KeySignatureMark.KeySignatureMark.KeySignatureMark) (attribute), 370
 symbolic_accidental_string_to_alphabetic_accidental_abbreviation (abjad.tools.contexttools.StaffChangeMark.StaffChangeMark.StaffChangeMark) (in module abjad.tools.pitchtools.symbolic_accidental_string_to_alphabetic_accidental_abbreviation) (attribute), 375
 1210 target_context (abjad.tools.contexttools.TempoMark.TempoMark.TempoMark) (attribute), 379
 symbolic_string (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree) (attribute), 2098
 target_context (abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark.TimeSignatureMark) (attribute), 389
 symbolic_string (abjad.tools.tonalitytools.TonalFunction.TonalFunction.TonalFunction) (attribute), 2102
 target_context (abjad.tools.instrumenttools.Accordion.Accordion.Accordion) (attribute), 430
 symmetric_difference() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet) (method), 1005
 target_context (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone) (attribute), 444
 symmetric_difference() (abjad.tools.pitchtools.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet) (method), 1019
 target_context (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute) (attribute), 451
 symmetric_difference() (abjad.tools.pitchtools.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet.HarmonicDiatonicIntervalSet) (method), 1026
 target_context (abjad.tools.instrumenttools.BaritoneSaxophone.BaritoneSaxophone.BaritoneSaxophone) (attribute), 472
 symmetric_difference() (abjad.tools.pitchtools.InversionEquivalentChromaticIntervalClassSet.HarmonicDiatonicIntervalClassSet.HarmonicDiatonicIntervalClassSet) (method), 1036
 target_context (abjad.tools.instrumenttools.BaritoneVoice.BaritoneVoice.BaritoneVoice) (attribute), 479
 symmetric_difference() (abjad.tools.pitchtools.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet.MelodicChromaticIntervalSet) (method), 1072
 target_context (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute) (attribute), 514
 symmetric_difference() (abjad.tools.pitchtools.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet.MelodicDiatonicIntervalSet) (method), 1090
 target_context (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon) (attribute), 514
 symmetric_difference() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet) (method), 1105
 target_context (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone) (attribute), 507
 symmetric_difference() (abjad.tools.pitchtools.NamedChromaticPitchSet.NamedChromaticPitchSet.NamedChromaticPitchSet) (method), 1113
 target_context (abjad.tools.instrumenttools.BassVoice.BassVoice.BassVoice) (attribute), 507
 symmetric_difference() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet) (method), 1141
 target_context (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet) (attribute), 528
 symmetric_difference() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass) (method), 2076
 target_context (abjad.tools.instrumenttools.Contrabass.Contrabass.Contrabass) (attribute), 535
 T target_context (abjad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet) (attribute), 542
 tabulate_well_formedness_violations_in_expr() (in module abjad.tools.componenttools.tabulate_well_formedness_violations_in_expr) (attribute), 549
 291 target_context (abjad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute) (attribute), 563
 target_context (abjad.tools.contexttools.ClefMark.ClefMark.ClefMark) (attribute), 340
 target_context (abjad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon) (attribute), 556
 target_context (abjad.tools.contexttools.ContextMark.ContextMark.ContextMark) (attribute), 358
 target_context (abjad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone) (attribute), 570
 target_context (abjad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice) (attribute), 570

target_context (abjad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet (ab-
attribute), 577
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 577

target_context (abjad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn (ab-
attribute), 584
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 584

target_context (abjad.tools.instrumenttools.Flute.Flute.Flute (ab-
attribute), 591
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 591

target_context (abjad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn (ab-
attribute), 598
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 598

target_context (abjad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel (ab-
attribute), 605
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 605

target_context (abjad.tools.instrumenttools.Guitar.Guitar.Guitar (ab-
attribute), 612
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 612

target_context (abjad.tools.instrumenttools.Harp.Harp.Harp (ab-
attribute), 619
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 619

target_context (abjad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord (ab-
attribute), 626
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 626

target_context (abjad.tools.instrumenttools.Marimba.Marimba.Marimba (ab-
attribute), 638
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 638

target_context (abjad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice.MezzoSopranoVoice (ab-
attribute), 645
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 645

target_context (abjad.tools.instrumenttools.Oboe.Oboe.Oboe (ab-
attribute), 652
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 652

target_context (abjad.tools.instrumenttools.Piano.Piano.Piano (ab-
attribute), 659
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 659

target_context (abjad.tools.instrumenttools.Piccolo.Piccolo.Piccolo (ab-
attribute), 666
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 666

target_context (abjad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone.SopraninoSaxophone (ab-
attribute), 673
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 673

target_context (abjad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone.SopranoSaxophone (ab-
attribute), 680
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 680

target_context (abjad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice (ab-
attribute), 687
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 687

target_context (abjad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone (ab-
attribute), 694
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 694

target_context (abjad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone (ab-
attribute), 701
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 701

target_context (abjad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice (ab-
attribute), 708
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 708

target_context (abjad.tools.instrumenttools.Trumpet.Trumpet.Trumpet (ab-
attribute), 715
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 715

target_context (abjad.tools.instrumenttools.Tuba.Tuba.Tuba (ab-
attribute), 722
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 722

target_context (abjad.tools.instrumenttools.UntunedPercussion.UntunedPercussion.UntunedPercussion (ab-
attribute), 729
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 729

target_context (abjad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone (ab-
attribute), 736
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 736

target_context (abjad.tools.instrumenttools.Viola.Viola.Viola (ab-
attribute), 743
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 743

target_context (abjad.tools.instrumenttools.Violin.Violin.Violin (ab-
attribute), 750
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 750

target_context (abjad.tools.instrumenttools.Xylophone.Xylophone.Xylophone (ab-
attribute), 757
jad.tools.pitchtools.OctaveTranspositionMappingComponent.OctaveTranspositionMappingComponent), 757

target_duration (abjad.tools.tuplettools.FixedDurationTuplet.FixedDurationTuplet.FixedDurationTuplet (ab-
attribute), 1570
jad.tools.tietools.tie_chain_to_augmented_tuplet_with_proportions_and_avoid_dots()), 1570

1558 token (abjad.tools.pitcharraytools.PitchArrayCell.PitchArrayCell.PitchArrayCell), 1733

tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots() (in module abjad.tools.tietools.tie_chain_to_augmented_tuplet_with_proportions_and_encourage_dots), 1560

tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots() (in module abjad.tools.tietools.tie_chain_to_diminished_tuplet_with_proportions_and_avoid_dots), 1561

tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots() (in module abjad.tools.tietools.tie_chain_to_diminished_tuplet_with_proportions_and_encourage_dots), 1562

TieSpanner (class in abjad.tools.timetokenmaker.TieSpanner), 1543

time_signature_to_binary_time_signature() (in module abjad.tools.timesignaturetools.time_signature_to_binary_time_signature), 1895

TimeInterval (class in abjad.tools.timeintervaltools.TimeInterval.TimeInterval), 1840

TimeIntervalAggregateMixin (class in abjad.tools.timeintervaltools.TimeIntervalAggregateMixin.TimeIntervalAggregateMixin), 1831

TimeIntervalMixin (class in abjad.tools.timeintervaltools.TimeIntervalMixin.TimeIntervalMixin), 1836

TimeIntervalTree (class in abjad.tools.timeintervaltools.TimeIntervalTree.TimeIntervalTree), 1844

TimeIntervalTreeDictionary (class in abjad.tools.timeintervaltools.TimeIntervalTreeDictionary.TimeIntervalTreeDictionary), 1858

timeout (abjad.tools.documentationtools.Pipe.Pipe.Pipe attribute), 1656

TimeSignatureMark (class in abjad.tools.contexttools.TimeSignatureMark.TimeSignatureMark), 387

TimeTokenMaker (class in abjad.tools.timetokenmaker.TimeTokenMaker.TimeTokenMaker), 1903

title (abjad.tools.tonalitytools.InversionIndicator.InversionIndicator.InversionIndicator attribute), 2087

title_string (abjad.tools.tonalitytools.ScaleDegree.ScaleDegree.ScaleDegree attribute), 2098

title_string (abjad.tools.tonalitytools.SuspensionIndicator.SuspensionIndicator.SuspensionIndicator attribute), 2100

to_nested_lists() (abjad.tools.sequencetools.CyclicTree.CyclicTree.CyclicTree method), 1758

to_nested_lists() (abjad.tools.sequencetools.Tree.Tree.Tree method), 1773

TokenBurnishedSignalFilledTimeTokenMaker (class in abjad.tools.timetokenmaker.TokenBurnishedSignalFilledTimeTokenMaker), 1903

TokenIncisedNoteFilledTimeTokenMaker (class in abjad.tools.timetokenmaker.TokenIncisedNoteFilledTimeTokenMaker), 1903

TokenIncisedRestFilledTimeTokenMaker (class in abjad.tools.timetokenmaker.TokenIncisedRestFilledTimeTokenMaker), 1903

TokenIncisedTimeTokenMaker (class in abjad.tools.timetokenmaker.TokenIncisedTimeTokenMaker), 1905

TonalFunction (class in abjad.tools.tonalitytools.TonalFunction.TonalFunction), 2101

tonic (abjad.tools.tonalitytools.Scale.Scale.Scale attribute), 2095

traditional_pitch_range (abjad.tools.instrumenttools.Accordion.Accordion.Accordion attribute), 439

traditional_pitch_range (abjad.tools.instrumenttools.AltoFlute.AltoFlute.AltoFlute attribute), 437

traditional_pitch_range (abjad.tools.instrumenttools.AltoSaxophone.AltoSaxophone.AltoSaxophone attribute), 444

traditional_pitch_range (abjad.tools.instrumenttools.AltoTrombone.AltoTrombone.AltoTrombone attribute), 445

traditional_pitch_range (abjad.tools.instrumenttools.BaronetSaxophone.BaronetSaxophone.BaronetSaxophone attribute), 465

traditional_pitch_range (abjad.tools.instrumenttools.BaronetVoice.BaronetVoice.BaronetVoice attribute), 472

traditional_pitch_range (abjad.tools.instrumenttools.BassClarinet.BassClarinet.BassClarinet attribute), 479

traditional_pitch_range (abjad.tools.instrumenttools.BassFlute.BassFlute.BassFlute attribute), 486

traditional_pitch_range (abjad.tools.instrumenttools.Bassoon.Bassoon.Bassoon attribute), 514

traditional_pitch_range (abjad.tools.instrumenttools.BassSaxophone.BassSaxophone.BassSaxophone attribute), 493

traditional_pitch_range (abjad.tools.instrumenttools.BassTrombone.BassTrombone.BassTrombone attribute), 493

attribute), 500	attribute), 626
traditional_pitch_range jad.tools.instrumenttools.BassVoice.BassVoice.BassVoice attribute), 507	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Marimba.Marimba.Marimba attribute), 638
traditional_pitch_range jad.tools.instrumenttools.BFlatClarinet.BFlatClarinet.BFlatClarinet attribute), 458	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.MezzoSopranoVoice.MezzoSopranoVoice attribute), 645
traditional_pitch_range jad.tools.instrumenttools.Cello.Cello.Cello attribute), 521	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Oboe.Oboe.Oboe attribute), 652
traditional_pitch_range jad.tools.instrumenttools.ClarinetInA.ClarinetInA.ClarinetInA attribute), 528	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Piano.Piano.Piano attribute), 659
traditional_pitch_range jad.tools.instrumenttools.Contrabass.Contrabass.Contrabass attribute), 535	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Piccolo.Piccolo.Piccolo attribute), 666
traditional_pitch_range jad.tools.instrumenttools.ContrabassClarinet.ContrabassClarinet.ContrabassClarinet attribute), 542	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.SopraninoSaxophone.SopraninoSaxophone attribute), 673
traditional_pitch_range jad.tools.instrumenttools.ContrabassFlute.ContrabassFlute.ContrabassFlute attribute), 549	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.SopranoSaxophone.SopranoSaxophone attribute), 680
traditional_pitch_range jad.tools.instrumenttools.Contrabassoon.Contrabassoon.Contrabassoon attribute), 563	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.SopranoVoice.SopranoVoice.SopranoVoice attribute), 687
traditional_pitch_range jad.tools.instrumenttools.ContrabassSaxophone.ContrabassSaxophone.ContrabassSaxophone attribute), 556	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.TenorSaxophone.TenorSaxophone.TenorSaxophone attribute), 694
traditional_pitch_range jad.tools.instrumenttools.ContraltoVoice.ContraltoVoice.ContraltoVoice attribute), 570	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.TenorTrombone.TenorTrombone.TenorTrombone attribute), 701
traditional_pitch_range jad.tools.instrumenttools.EFlatClarinet.EFlatClarinet.EFlatClarinet attribute), 577	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.TenorVoice.TenorVoice.TenorVoice attribute), 708
traditional_pitch_range jad.tools.instrumenttools.EnglishHorn.EnglishHorn.EnglishHorn attribute), 584	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Trumpet.Trumpet.Trumpet attribute), 715
traditional_pitch_range jad.tools.instrumenttools.Flute.Flute.Flute attribute), 591	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Tuba.Tuba.Tuba attribute), 722
traditional_pitch_range jad.tools.instrumenttools.FrenchHorn.FrenchHorn.FrenchHorn attribute), 598	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.UntunedPercussion.UntunedPercussion attribute), 729
traditional_pitch_range jad.tools.instrumenttools.Glockenspiel.Glockenspiel.Glockenspiel attribute), 605	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Vibraphone.Vibraphone.Vibraphone attribute), 736
traditional_pitch_range jad.tools.instrumenttools.Guitar.Guitar.Guitar attribute), 612	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Viola.Viola.Viola attribute), 743
traditional_pitch_range jad.tools.instrumenttools.Harp.Harp.Harp attribute), 619	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Violin.Violin.Violin attribute), 750
traditional_pitch_range jad.tools.instrumenttools.Harpsichord.Harpsichord.Harpsichord attribute), 626	(ab- traditional_pitch_range (ab- jad.tools.instrumenttools.Xylophone.Xylophone.Xylophone attribute), 757

attribute), 757
transpose() (abjad.tools.pitchtools.NamedChromaticPitchClass.NamedChromaticPitchClass.NamedChromaticPitchClass.range), method), 1099
transpose() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.range), method), 1102
transpose() (abjad.tools.pitchtools.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.NamedChromaticPitchClassSet.range), method), 1106
transpose() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment.range), method), 1109
transpose() (abjad.tools.pitchtools.NamedChromaticPitchSegment.NamedChromaticPitchSegment.NamedChromaticPitchSegment.range), method), 1113
transpose() (abjad.tools.pitchtools.NumberedChromaticPitch.NumberedChromaticPitch.NumberedChromaticPitch.range), method), 1128
transpose() (abjad.tools.pitchtools.NumberedChromaticPitchClass.NumberedChromaticPitchClass.NumberedChromaticPitchClass.range), method), 1130
transpose() (abjad.tools.pitchtools.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.NumberedChromaticPitchClassSegment.range), method), 1136
transpose() (abjad.tools.pitchtools.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.NumberedChromaticPitchClassSet.range), method), 1141
transpose() (abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow.TwelveToneRow.range), method), 1174
transpose() (abjad.tools.tonalitytools.ChordClass.ChordClass.ChordClass.range), method), 2076
transpose() (abjad.tools.tonalitytools.Scale.Scale.Scale.Tuba (class in abjad.tools.instrumenttools.Tuba.Tuba), method), 2096
transpose_chromatic_pitch_by_melodic_chromatic_interval_tuple() (class in abjad.tools.tuplettools.Tuplet.Tuplet), (in module abjad.tools.pitchtools.transpose_chromatic_pitch_by_melodic_chromatic_interval_tuple), 1210
transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chromatic_pitch_number() (abjad.tools.pitchtools.transpose_chromatic_pitch_class_number_by_octaves_to_nearest_neighbor_of_chromatic_pitch_number), 1210
transpose_chromatic_pitch_number_by_octave_transposition_mapping() (abjad.tools.pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping), (in module abjad.tools.pitchtools.TwelveToneRow.TwelveToneRow), 1172
transpose_chromatic_pitch_number_by_octave_transposition_mapping() (abjad.tools.pitchtools.transpose_chromatic_pitch_number_by_octave_transposition_mapping), (abjad.tools.pitchtools.NumberedChromaticPitchClassColorMap.NumberedChromaticPitchClassColorMap), 1211
transpose_named_chromatic_pitch_by_melodic_chromatic_interval_tuple() (class in abjad.tools.tuplettools.Tuplet.Tuplet), (in module abjad.tools.pitchtools.transpose_named_chromatic_pitch_by_melodic_chromatic_interval_tuple), 1212
transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch() (abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch), (in module abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_fingered_pitch_to_sounding_pitch), 766
transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch() (abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch), (in module abjad.tools.instrumenttools.transpose_notes_and_chords_in_expr_from_sounding_pitch_to_fingered_pitch), 767
transpose_pitch_carrier_by_melodic_interval() (abjad.tools.iotools.underscore_delimited_lowercase_to_uppercamel), (in module abjad.tools.iotools.underscore_delimited_lowercase_to_uppercamel), 1696
transpose_pitch_carrier_by_melodic_interval() (abjad.tools.pitchtools.HarmonicChromaticIntervalSet.HarmonicChromaticIntervalSet), method), 1005

[yield_all_positive_rationals_in_cantor_diagonalized_order\(yield_groups_of_rests_in_sequence\(\)\)](#) (in module `abjad.tools.resttools`),
[yield_all_positive_rationals_in_cantor_diagonalized_order\(yield_groups_of_rests_in_sequence\(\)\)](#) (in module `abjad.tools.durationtools`),
[yield_all_positive_rationals_in_cantor_diagonalized_order\(yield_groups_of_rests_in_sequence\(\)\)](#) (in module `abjad.tools.skiptools`),
[yield_all_positive_rationals_in_cantor_diagonalized_order\(yield_groups_of_rests_in_sequence\(\)\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalized_order\(\)](#) (in module `abjad.tools.componenttools`),
[yield_all_prolation_rewrite_pairs_of_rational_in_cantor_diagonalized_order\(\)](#) (in module `abjad.tools.componenttools`),
[yield_all_restricted_growth_functions_of_length\(\)](#) (in module `abjad.tools.componenttools`),
[yield_all_rotations_of_sequence\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_set_partitions_of_sequence\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_set_partitions_of_sequence\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_subchords_of_chord\(\)](#) (in module `abjad.tools.chordtools`),
[yield_all_subchords_of_chord\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_subsequences_of_sequence\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_all_unordered_pairs_of_sequence\(\)](#) (in module `abjad.tools.sequencetools`),
[yield_components_grouped_by_preprolated_duration\(\)](#) (in module `abjad.tools.componenttools`),
[yield_components_grouped_by_prolated_duration\(\)](#) (in module `abjad.tools.componenttools`),
[yield_groups_of_chords_in_sequence\(\)](#) (in module `abjad.tools.chordtools`),
[yield_groups_of_mixed_klasses_in_sequence\(\)](#) (in module `abjad.tools.componenttools`),
[yield_groups_of_mixed_notes_and_chords_in_sequence\(\)](#) (in module `abjad.tools.leaftools`),
[yield_groups_of_notes_in_sequence\(\)](#) (in module `abjad.tools.notetools`),