
auxjad Documentation

Release 0.2.0

Gilberto Agostinho

Aug 10, 2019

CONTENTS:

1	The <i>auxjad</i> package	1
1.1	auxjad	1
	Python Module Index	5
	Index	7

THE AUXJAD PACKAGE

1.1 auxjad

Auxiliary functions and classes for Abjad 3.0. All classes and functions have a `__doc__` attribute with usage instructions.

Documentation is available at <https://gilberthasnofb.github.io/auxjad-docs/>. A pdf version of the documentation is also available in the *docs* directory.

Bugs can be reported to <https://github.com/gilberthasnofb/auxjad/issues>.

This library is published under the MIT License.

class `auxjad.LeafDynMaker` (*decrease_monotonic=True, forbidden_duration=None, metrical_hierarchy=None, skips_instead_of_rests=False, repeat_ties=False, use_multimeasure_rests=False*)

An extension of `abjad.LeafMaker` which can also take optional lists of dynamics and articulations.

Usage is similar to `LeafMaker`:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = [(1, 32), (2, 32), (3, 32), (4, 32), (5, 32), (6, 32)]
>>> dynamics = ['pp', 'p', 'mp', 'mf', 'f', 'ff']
>>> articulations = ['.', '>', '-', '_', '^', '+']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
```

Tuple elements in `pitches` result in chords. None-valued elements in `pitches` result in rests:

```
>>> pitches = [5, None, (0, 2, 7)]
>>> durations = [(1, 4), (1, 8), (1, 16)]
>>> dynamics = ['p', None, 'f']
>>> articulations = ['staccato', None, 'tenuto']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
```

Can omit repeated dynamics with the keyword argument `no_repeat`:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = [(1, 32), (2, 32), (3, 32), (4, 32), (5, 32), (6, 32)]
>>> dynamics = ['pp', 'pp', 'mp', 'f', 'f', 'p']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
```

(continues on next page)

(continued from previous page)

```
>>> notes = leaf_dyn_maker(pitches,
...                         durations,
...                         dynamics,
...                         no_repeat=True,
...                         )
>>> staff = abjad.Staff(notes)
```

The lengths of both dynamics and articulations can be shorter than the lengths of pitches and durations (whatever is the greatest):

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = (1, 4)
>>> dynamics = ['p', 'f', 'ff']
>>> articulations = ['.', '>']
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
```

If the length of articulations is 1, it will apply to all elements. If the length of dynamics is 1, it will apply to the first element only:

```
>>> pitches = [0, 2, 4, 5, 7, 9]
>>> durations = (1, 4)
>>> dynamics = 'p'
>>> articulations = '.'
>>> leaf_dyn_maker = auxjad.LeafDynMaker()
>>> notes = leaf_dyn_maker(pitches, durations, dynamics, articulations)
>>> staff = abjad.Staff(notes)
```

`auxjad.container_comparator` (*container1*, *container2*, *include_indicators*: *bool* = *False*, *include_grace_notes*: *bool* = *False*) → *bool*

A comparator function returning True when two containers are identical and False when they are not.

When the pitches and effective durations of all leaves in both containers are identical, this function returns True:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
```

Even if all leaves of both containers are identical in pitches and in `written_duration`, the function considers the effective duration so that situations like the one below do not yield a false positive:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"\times 3/2 {c'4 d'4 e'4} f'4 <g' a'>2 r2")
```

By default, this function ignores indicators, so the containers in the example below are understood to be identical:

```
>>> container1 = abjad.Staff(r"c'4\pp d'4 e'4-. f'4 <g' a'>2-> r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
```

Setting the argument `include_indicators` to True forces the function to include indicators in its comparison. In that case, the containers in the example above are not considered identical any longer:

```
>>> container1 = abjad.Staff(r"c'4\pp d'4 e'4-. f'4 <g' a'>2-> r2")
>>> container2 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
```

By default, this function ignores grace notes, so the containers in the example below are understood to be identical:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
```

Setting the argument `include_grace_notes` to `True` forces the function to include grace notes in its comparison. In that case, the containers in the example above are not considered identical any longer:

```
>>> container1 = abjad.Staff(r"c'4 d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
```

When the argument `include_grace_notes` is set to `True`, the function will consider not only a grace note is attached to a given leaf, but also whether the contents of the grace containers are identical:

```
>>> container1 = abjad.Staff(r"c'4 \grace{c''4} d'4 e'4 f'4 <g' a'>2 r2")
>>> container2 = abjad.Staff(r"c'4 \grace{c''8} d'4 e'4 f'4 <g' a'>2 r2")
```


PYTHON MODULE INDEX

a

auxjad, [1](#)

INDEX

A

`auxjad` (module), 1

C

`container_comparator()` (in module `auxjad`), 2

L

`LeafDynMaker` (class in `auxjad`), 1