
pychord Documentation

Author

Sep 19, 2021

Contents:

1	pychord package	1
1.1	Subpackages	1
1.1.1	pychord.constants package	1
1.1.1.1	Submodules	1
1.1.1.2	pychord.constants.qualities module	1
1.1.1.3	pychord.constants.scales module	1
1.1.1.4	Module contents	1
1.2	Submodules	1
1.3	pychord.analyzer module	1
1.4	pychord.chord module	2
1.5	pychord.parser module	3
1.6	pychord.progression module	4
1.7	pychord.quality module	4
1.8	pychord.utils module	5
1.9	Module contents	6
2	Indices and tables	7
	Python Module Index	9
	Index	11

1.1 Subpackages

1.1.1 pychord.constants package

1.1.1.1 Submodules

1.1.1.2 pychord.constants.qualities module

1.1.1.3 pychord.constants.scales module

1.1.1.4 Module contents

1.2 Submodules

1.3 pychord.analyzer module

`pychord.analyzer.get_all_rotated_notes` (*notes*)

Get all rotated notes

`get_all_rotated_notes([1,3,5]) -> [[1,3,5],[3,5,1],[5,1,3]]`

Return type `list[list[str]]`

`pychord.analyzer.note_to_chord` (*notes*)

Convert note list to chord list

Parameters *notes* (`list[str]`) – list of note arranged from lower note. ex) ["C", "Eb", "G"]

Return type `list[pychord.Chord]`

Returns list of chord

`pychord.analyzer.notes_to_positions` (*notes*, *root*)

Get notes positions.

ex) `notes_to_positions(["C", "E", "G"], "C")` -> [0, 4, 7]

Parameters

- **notes** (*list[str]*) – list of notes
- **root** (*str*) – the root note

Return type `list[int]`

Returns list of note positions

1.4 pychord.chord module

class `pychord.chord.Chord` (*chord*)

Bases: `object`

Class to handle a chord.

Parameters

- **_chord** (*str*) – Name of the chord. (e.g. C, Am7, F#m7-5/A)
- **_root** (*str*) – The root note of chord.
- **_quality** (*pychord.Quality*) – The quality of chord. (e.g. m7, 6, M9, ...)
- **_appended** (*list[str]*) – The appended notes on chord.
- **_on** (*str*) – The base note of slash chord.

appended

The appended notes on chord

chord

The name of chord

components (*visible=True*)

Return the component notes of chord

Parameters **visible** (*bool*) – returns the name of notes if True else list of int

Return type `list[(str or int)]`

Returns component notes of chord

components_with_pitch (*root_pitch*)

Return the component notes of chord formatted like ["C4", "E4", "G4"]

Parameters **root_pitch** (*int*) – the pitch of the root note

Return type `list[str]`

Returns component notes of chord

classmethod **from_note_index** (*note*, *quality*, *scale*, *diatonic=False*)

Create a Chord from note index in a scale

`Chord.from_note_index(1, "", "Cmaj")` returns I of C major => `Chord("C")` `Chord.from_note_index(3, "m7", "Fmaj")` returns IIImin of F major => `Chord("Am7")` `Chord.from_note_index(5, "7", "Amin")` returns Vmin of A minor => `Chord("E7")`

Parameters

- **note** (*int*) – Note index in a Scale I, II, ..., VIII
- **quality** (*str*) – Quality of a chord (m7, sus4, ...)
- **scale** (*str*) – Base scale (Cmaj, Amin, F#maj, Ebmin, ...)

Return type *Chord***info** ()

Return information of chord to display

on

The base note of slash chord

quality

The quality of chord

root

The root note of chord

transpose (*trans*, *scale*='C')

Transpose the chord

Parameters

- **trans** (*int*) – Transpose key
- **scale** (*str*) – key scale

Returns`pychord.chord.as_chord(chord)`

convert from str to Chord instance if input is str

Parameters **chord** (*str/pychord.Chord*) – Chord name or Chord instance**Return type** `pychord.Chord`**Returns** Chord instance

1.5 pychord.parser module

`pychord.parser.check_note(note, chord)`

Return True if the note is valid.

Parameters

- **note** (*str*) – note to check its validity
- **chord** (*str*) – the chord which includes the note

Return type `bool``pychord.parser.parse(chord)`

Parse a string to get chord component

Parameters **chord** (*str*) – str expression of a chord**Return type** (`str`, `pychord.Quality`, `str`, `str`)**Returns** (root, quality, appended, on)

1.6 pychord.progression module

class `pychord.progression.ChordProgression` (*initial_chords=None*)

Bases: `object`

Class to handle chord progressions.

Parameters `_chords` (*list* [`pychord.Chord`]) – component chords of chord progression.

append (*chord*)

Append a chord to chord progressions

Parameters `chord` (*str* | `pychord.Chord`) – A chord to append

Returns

chords

Get component chords of chord progression

Return type `list`[`pychord.Chord`]

insert (*index, chord*)

Insert a chord to chord progressions

Parameters

- **index** (*int*) – Index to insert a chord
- **chord** (*str* | `pychord.Chord`) – A chord to insert

Returns

pop (*index=-1*)

Pop a chord from chord progressions

Parameters **index** (*int*) – Index of the chord to pop (default: -1)

Returns `pychord.Chord`

transpose (*trans*)

Transpose whole chord progressions

Parameters **trans** (*int*) – Transpose key

Returns

1.7 pychord.quality module

class `pychord.quality.Quality` (*name, components*)

Bases: `object`

Chord quality

Parameters **_quality** (*str*) – str expression of chord quality

append_note (*note, root, scale=0*)

Append a note to quality

Parameters

- **note** (*str*) – note to append on quality
- **root** (*str*) – root note of chord

- **scale** (*int*) – key scale

append_notes (*notes*, *root*, *scale=0*)
Append notes to quality

Parameters

- **notes** (*list[str]*) – notes to append on quality
- **root** (*str*) – root note of chord
- **scale** (*int*) – key scale

append_on_chord (*on_chord*, *root*)
Append on chord

To create Am7/G q = Quality('m7') q.append_on_chord('G', root='A')

Parameters

- **on_chord** (*str*) – bass note of the chord
- **root** (*str*) – root note of the chord

get_components (*root='C'*, *visible=False*)
Get components of chord quality

Parameters

- **root** (*str*) – the root note of the chord
- **visible** (*bool*) – returns the name of notes if True

Return type list[str|int]

Returns components of chord quality

quality
Get name of quality

class pychord.quality.**QualityManager**
Bases: object

Singleton class to manage the qualities

find_quality_from_components (*components*)
Find a quality from components

Parameters **components** (*Tuple[int]*) – components of quality

get_quality (*name*)

load_default_qualities ()

set_quality (*name*, *components*)
Set a Quality

This method will not affect any existing Chord instances. :param str name: name of quality :param Tuple[int] components: components of quality

1.8 pychord.utils module

pychord.utils.**display_appended** (*appended*)

pychord.utils.**display_on** (*on_note*)

`pychord.utils.note_to_val(note)`
Convert note to int

```
>>> note_to_val("C")
0
>>> note_to_val("B")
11
```

Return type int

`pychord.utils.transpose_note(note, transpose, scale='C')`
Transpose a note

Parameters

- **note** (*str*) – note to transpose
- **scale** (*str*) – key scale

Return type str

Returns transposed note

`pychord.utils.val_to_note(val, scale='C')`
Convert int to note

```
>>> val_to_note(0)
"C"
>>> val_to_note(11, "D")
"D#"
```

Parameters **scale** (*str*) – key scale

Return type str

1.9 Module contents

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

p

- `pychord`, 6
- `pychord.analyzer`, 1
- `pychord.chord`, 2
- `pychord.constants`, 1
- `pychord.constants.qualities`, 1
- `pychord.constants.scales`, 1
- `pychord.parser`, 3
- `pychord.progression`, 4
- `pychord.quality`, 4
- `pychord.utils`, 5

A

`append()` (*pychord.progression.ChordProgression method*), 4
`append_note()` (*pychord.quality.Quality method*), 4
`append_notes()` (*pychord.quality.Quality method*), 5
`append_on_chord()` (*pychord.quality.Quality method*), 5
`appended` (*pychord.chord.Chord attribute*), 2
`as_chord()` (*in module pychord.chord*), 3

C

`check_note()` (*in module pychord.parser*), 3
`Chord` (*class in pychord.chord*), 2
`chord` (*pychord.chord.Chord attribute*), 2
`ChordProgression` (*class in pychord.progression*), 4
`chords` (*pychord.progression.ChordProgression attribute*), 4
`components()` (*pychord.chord.Chord method*), 2
`components_with_pitch()` (*pychord.chord.Chord method*), 2

D

`display_appended()` (*in module pychord.utils*), 5
`display_on()` (*in module pychord.utils*), 5

F

`find_quality_from_components()` (*pychord.quality.QualityManager method*), 5
`from_note_index()` (*pychord.chord.Chord class method*), 2

G

`get_all_rotated_notes()` (*in module pychord.analyzer*), 1
`get_components()` (*pychord.quality.Quality method*), 5
`get_quality()` (*pychord.quality.QualityManager method*), 5

I

`info()` (*pychord.chord.Chord method*), 3
`insert()` (*pychord.progression.ChordProgression method*), 4

L

`load_default_qualities()` (*pychord.quality.QualityManager method*), 5

N

`note_to_chord()` (*in module pychord.analyzer*), 1
`note_to_val()` (*in module pychord.utils*), 5
`notes_to_positions()` (*in module pychord.analyzer*), 1

O

`on` (*pychord.chord.Chord attribute*), 3

P

`parse()` (*in module pychord.parser*), 3
`pop()` (*pychord.progression.ChordProgression method*), 4
`pychord` (*module*), 6
`pychord.analyzer` (*module*), 1
`pychord.chord` (*module*), 2
`pychord.constants` (*module*), 1
`pychord.constants.qualities` (*module*), 1
`pychord.constants.scales` (*module*), 1
`pychord.parser` (*module*), 3
`pychord.progression` (*module*), 4
`pychord.quality` (*module*), 4
`pychord.utils` (*module*), 5

Q

`Quality` (*class in pychord.quality*), 4
`quality` (*pychord.chord.Chord attribute*), 3
`quality` (*pychord.quality.Quality attribute*), 5
`QualityManager` (*class in pychord.quality*), 5

R

`root` (*pychord.chord.Chord* attribute), [3](#)

S

`set_quality()` (*pychord.quality.QualityManager* method), [5](#)

T

`transpose()` (*pychord.chord.Chord* method), [3](#)

`transpose()` (*pychord.progression.ChordProgression* method), [4](#)

`transpose_note()` (*in module pychord.utils*), [6](#)

V

`val_to_note()` (*in module pychord.utils*), [6](#)