pychord Documentation

Author

Sep 19, 2021

Contents:

CHAPTER 1

pychord package

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.**find_chords_from_notes**(*notes: List[str]*) → List[pychord.chord.Chord]
Find possible chords consisted from notes

Parameters **notes** – List of note arranged from lower note. ex) [“C”, “Eb”, “G”]

Returns List of chord

pychord.analyzer.**get_all_rotated_notes**(*notes: List[str]*) → List[List[str]]
Get all rotated notes

get_all_rotated_notes([A,C,E]) -> [[A,C,E],[C,E,A],[E,A,C]]

pychord.analyzer.**notes_to_positions**(*notes: List[str], root: str*) → List[int]
Get notes positions from the root note

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

Parameters

- **notes** – List of notes
- **root** – Root note

Returns List of note positions

1.4 pychord.chord module

class pychord.chord.**Chord**(*chord: str*)

Bases: object

Class to handle a chord.

Attributes: _chord: Name of the chord. (e.g. C, Am7, F#m7-5/A) _root: The root note of chord. (e.g. C, A, F#) _quality: The quality of chord. (e.g. maj, m7, m7-5) _appended: The appended notes on chord. _on: The base note of slash chord.

appended
The appended notes on chord

chord
The name of chord

components(*visible: bool = True*) → Union[List[str], List[int]]
Return the component notes of chord

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

Returns component notes of chord

components_with_pitch(*root_pitch: int*) → List[str]
Return the component notes of chord formatted like ["C4", "E4", "G4"]

Parameters **root_pitch** – the pitch of the root note

Returns component notes of chord

classmethod from_note_index(*note: int*, *quality: str*, *scale: str*, *diatonic: bool = False*) → pychord.chord.Chord
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** – Note index in a Scale I, II, . . . , VIII
- **quality** – Quality of a chord (m7, sus4, . . .)
- **scale** – Base scale (Cmaj, Amin, F#maj, Ebmin, . . .)
- **diatonic** – Adjust certain chord qualities according to the scale

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: int*, *scale: str = 'C'*) → None
Transpose the chord

Parameters

- **trans** – Transpose key
- **scale** – key scale

1.5 pychord.parser module

pychord.parser.**parse**(*chord: str*) → Tuple[str, pychord.quality.Quality, List[str], str]
Parse a string to get chord component

Parameters **chord** – str expression of a chord

Returns (root, quality, appended, on)

1.6 pychord.progression module

class pychord.progression.**ChordProgression**(*initial_chords: Union[str, pychord.chord.Chord, List[Union[str, pychord.chord.Chord]]] = None*)
Bases: object

Class to handle chord progressions.

Attributes: _chords: component chords of chord progression.

append(*chord: Union[str, pychord.chord.Chord]*) → None
Append a chord to chord progressions

Parameters **chord** – A chord to append

chords
Get component chords of chord progression

insert(*index: int, chord: Union[str, pychord.chord.Chord]*) → None
Insert a chord to chord progressions

Parameters

- **index** – Index to insert a chord
- **chord** – A chord to insert

pop(*index: int = -1*) → pychord.chord.Chord
Pop a chord from chord progressions

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

transpose(*trans: int*) → None
Transpose whole chord progressions

Parameters **trans** – Transpose key

1.7 pychord.quality module

class pychord.quality.**Quality**(*name: str, components: Tuple[int, ...]*)
Bases: object

Chord quality

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: List[int]*)
Find a quality from components

Parameters **components** – components of quality

get_quality(*name: str*) → pychord.quality.Quality

load_default_qualities()

set_quality(*name: str, components: Tuple[int, ...]*)
Set a Quality

This method will not affect any existing Chord instances. :param name: name of quality :param components: components of quality

1.8 pychord.utils module

pychord.utils.**display_appended**(*appended: List[str]*) → str

pychord.utils.**display_on**(*on_note: str*) → str

pychord.utils.**note_to_val**(*note: str*) → int

Get index value of a note

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

pychord.utils.**transpose_note**(*note: str*, *transpose: int*, *scale: str = 'C'*) → str

Transpose a note

```
>>> transpose_note("C", 1)
"Db"
>>> transpose_note("D", 4, "A")
"F#"
```

pychord.utils.**val_to_note**(*val: int*, *scale: str = 'C'*) → str

Return note by index in a scale

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

1.9 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

S

T

V