musictheory

Release 0.0.1

Fabian C. Moss

Apr 22, 2020

CONTENTS

CHAPTER

ONE

INTRODUCTION

This is not a pedagogical resource for basic music theory concepts but an in-depth introduction into the structures of Western music, built axiomatically from tones and their relations. The logo, a hemidemisemiquaver (or a sixty-fourth note), symbolically reflects this level of difficulty.

This project has been inspired by this great book:

- Lewin, D. (2007). *Generalized Intervals and Transformations*. Oxford: Oxford University Press.

I recently also discovered Music for Geeks and Nerds by Pedro Kroger which looks very interesting.

Since this is ongoing work, I can give no guarantee for completeness or accuracy. Feel free to contact me with your questions and suggestions!

CHAPTER

TWO

CONTENT

2.1 Installation

Warning: These instructions do not work yet.

To install `musictheory` type the following in your terminal:

```
pip install musictheory
```

2.2 API

The entire `musictheory` API.

2.2.1 Tone

class `main.`**`Tone`**(*octave=None*, *fifth=None*, *third=None*, *name=None*)

Class for tones.

`get_accidentals`()

Gets the accidentals of the tone (flats (*b*) or sharps (*#*))..

Parameters **`None`** –

Returns The accidentals of the tone.

Return type str

Example

```
>>> t = Tone(0,7,0) # C sharp
>>> t.get_accidentals()
`#`
```

`get_frequency`(*chamber_tone=440.0*, *precision=2*)

Get the frequency of the tone.

Parameters

- **chamber_tone** (*float*) – The frequency in Hz of the chamber tone. Default: 440.0 (A)
- **precision** (*int*) – Rounding precision.

Returns The frequency of the tone in Hertz (Hz).

Return type float

Example

```
>>> t = Tone(0,0,0)
>>> t.get_frequency(precision=3)
261.626
```

get_label()

Gets the complete label of the tone, consisting of its note name, syntonic position, and octave.

Parameters **None** –

Returns The accidentals of the tone.

Return type str

Example

```
>>> c = Tone(0,0,0)
>>> ab = Tone(0,1,-1)
>>> c.get_label(), ab.get_label()
`C_0` `Ab,1`
```

get_midi_pitch()

Get the MIDI pitch of the tone.

Parameters **None** –

Returns The MIDI pitch of the tone if it is in MIDI pitch range (0–128)

Return type int

Example

```
>>> t = Tone(0,0,0)
>>> t.get_midi_pitch()
60
```

get_pitch_class(*start=0*, *order='chromatic'*)

Get the pitch-class number on the circle of fifths or the chromatic circle.

Parameters

- **start** (*int*) – Pitch-class number that gets mapped to C (default: 0).
- **order** (*str*) – Return pitch-class number on the chromatic circle (default) or the circle of fifths.

Returns The pitch class of the tone on the circle of fifths or the chromatic circle.

Return type int

Example

```
>>> t = Tone(0,7,0) # C sharp
>>> t.get_pitch_class(order="chromatic")
1
```

```
>>> t = Tone(0,7,0) # C sharp
>>> t.get_pitch_class(order="fifths")
7
```

get_step()

Gets the diatonic letter name (C, D, E, F, G, A, or B) of the tone *without* accidentals.

Parameters **None** –

Returns The diatonic step of the tone.

Return type str

Example

```
>>> t = Tone(0,7,0) # C sharp
>>> t.get_step()
`C`
```

get_syntonic()

Gets the value of the syntonic level in Euler space. Tones on the same syntonic line as central C are marked with _, and those above or below this line with ' or ,, respectively.

Parameters **None** –

Returns The number of thirds above or below the central C.

Return type int

Example

```
>>> e1 = Tone(0,4,0) # Pythagorean major third above C
>>> e2 = Tone(0,0,1) # Just major third above C
>>> e3 = Tone(0,8,-1) # Just major third below G sharp
>>> e1.get_syntonic(), e2.get_syntonic(), e3.get_syntonic()
`'` `_` `,`
```

2.2.2 Interval

class main.**Interval**(*source*, *target*)

Class for an interval between two tones *s* (source) and *t* (target).

get_euclidean_distance(*precision=2*)

Calculates the Euclidean distance between two tones with coordinates in Euler space.

Parameters **precision** (*int*) – Rounding precision.

Returns The Euclidean distance between two tones *s* (source) and *t* (target).

Return type float

Example

```
>>> s = Tone(0,0,0) # C_0
>>> t = Tone(1,2,1) # D'1
>>> i = Interval(s,t)
>>> i.get_euclidean_distance()
2.45
```

get_generic_interval(*directed=True*)
Generic interval (directed) between two tones.

Parameters **directed** (*bool*) – Affects whether the returned interval is directed or not.

Returns (Directed) generic interval from *s* to *t*.

Return type int

Example

```
>>> db = Tone(0,-1,-1) # Db,0
>>> b = Tone(0,1,1) # B'0
>>> i1 = Interval(db, b) # the interval between Db0 and B1 is an ascending␣
↪thirteenth
>>> i1.generic_interval()
13
```

```
>>> i2 = Interval(b, db) # the interval between B1 and Db0 is a descending␣
↪thirteenth
>>> i2.generic_interval()
-13
```

```
>>> i3 = Interval(b, db) # the interval between B1 and Db0 is a descending␣
↪thirteenth
>>> i3.generic_interval(directed=False)
13
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search