

Learning Python Through Music: JythonMusic & Pyknon



Ria Baldevia | 2016 US PyCon | Portland, Oregon

(?aimsx#) \A, \d, \D.

(?aimsx#) \A, \d, \D,

(?P<name>), (?P=

?<name>), (?P=names)

(?=) etc. ., *, +, ?, *?,

etc. ., *, +, ?, *?, +?, ??

[], \b, \B, \w, \W

\b, \B, \w, \W, \1, \2,

(?aimsx#)

(?aimsx#) \A, \d, \D,

), (?P=names)

, +?, *?, +?, ??

, \B, \w, \W \1, \2,

(?P<name>), (?P=names)

(?aimsx#) \A, \d, \D,

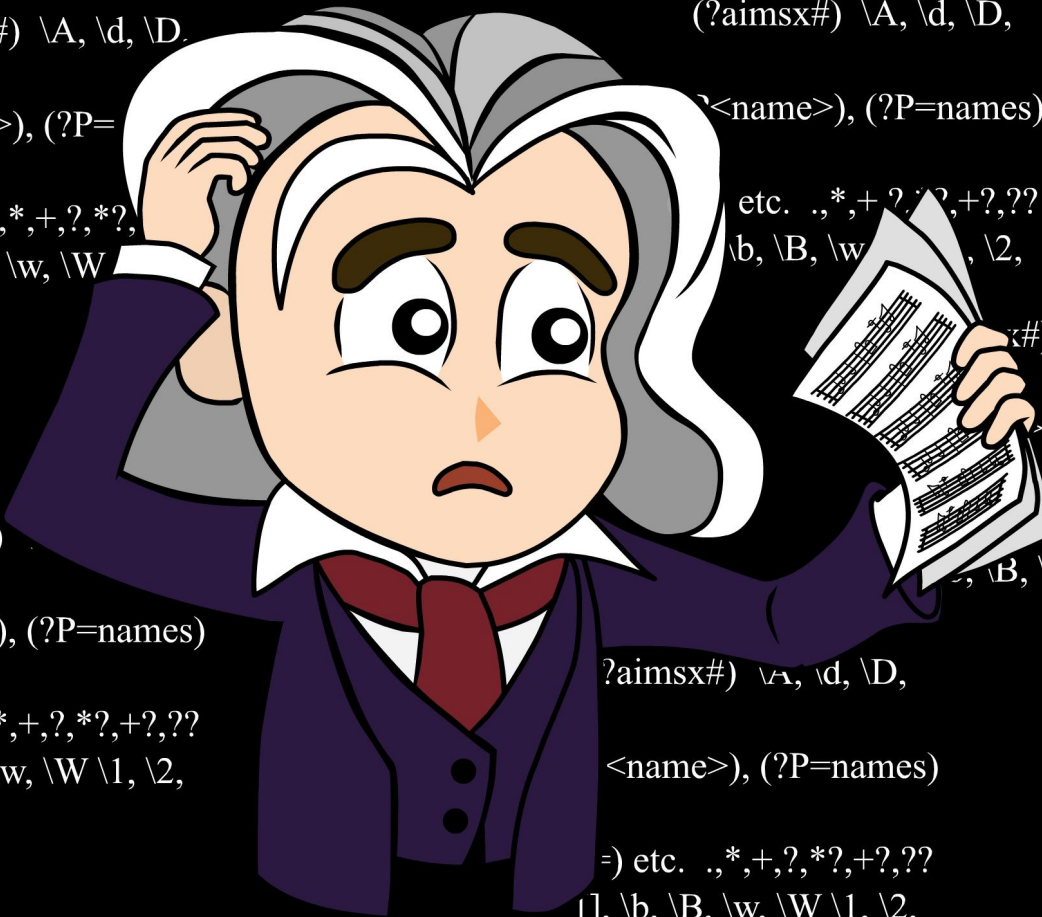
(?=) etc. ., *, +, ?, *?, +?, ??

<name>), (?P=names)

[], \b, \B, \w, \W \1, \2,

=) etc. ., *, +, ?, *?, +?, ??

[], \b, \B, \w, \W \1, \2,



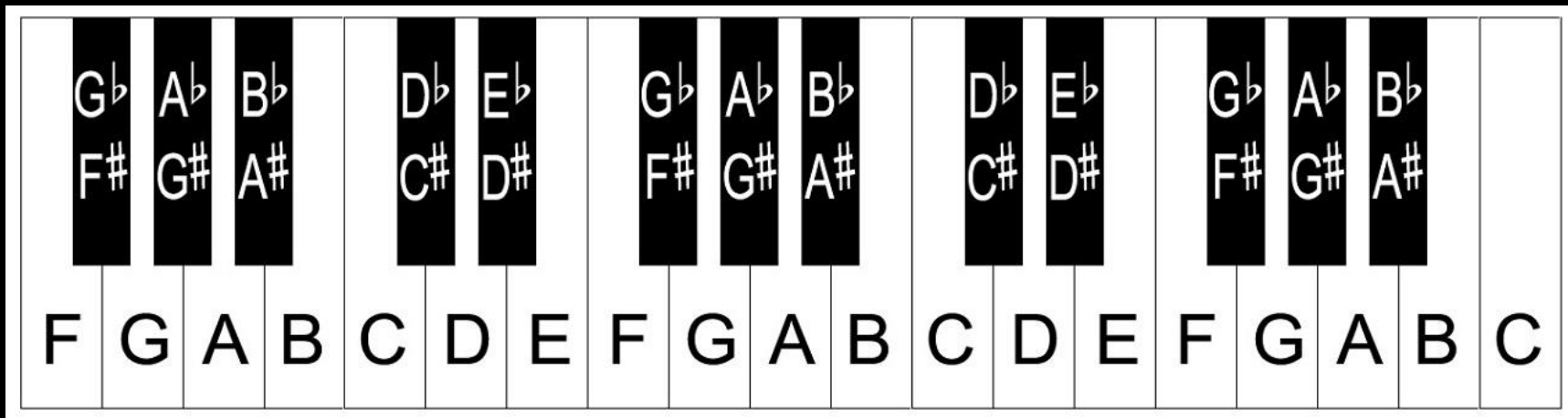


JythonMusic

JythonMusic: Jython Environment for Music

An open source environment conducive to creative programming. It is for programmers and musicians who want to explore music and coding.

Easy to download JEM via JythonMusic.org

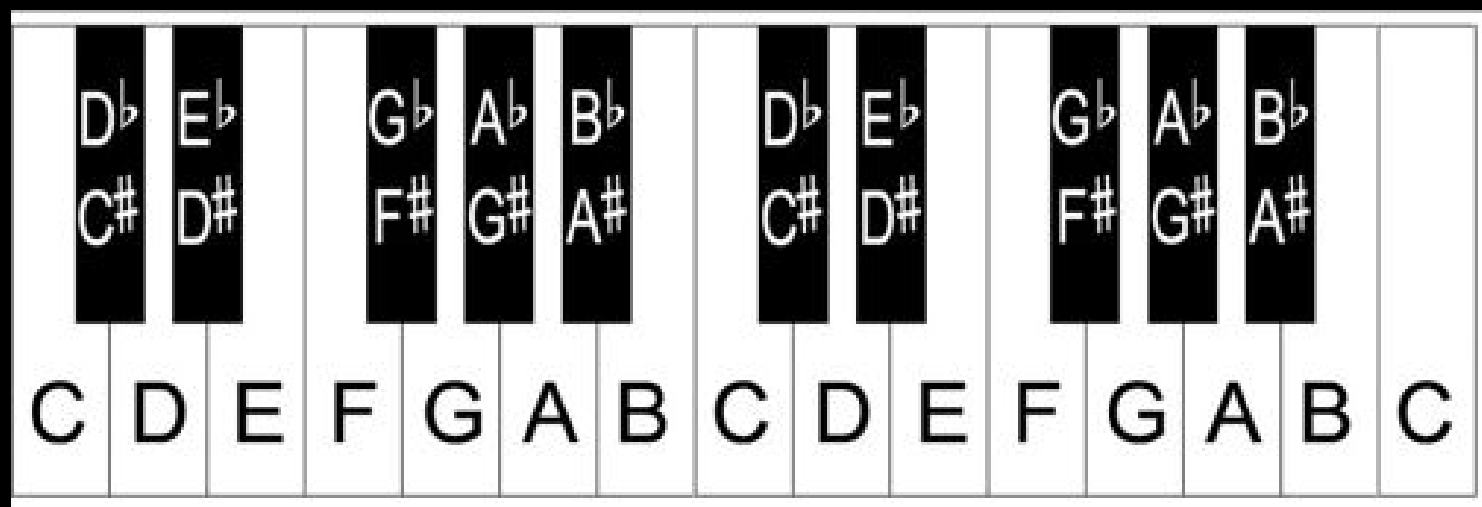


Symphony No.5 in C Minor, Op. 67

Ludwig Van Beethoven



G G G Eb F F F D



Octave 4

Octave 5

fifth_jython.py

```
1 from music import *
2
3 pitches1 = [G4, G4, G4, DS4, F4, F4, F4, D4]
4 durations1 = [EN, EN, EN, WN, EN, EN, EN, WN]
5
6
7 # create an empty phrase, and construct theme from the above motifs
8 theme = Phrase()
9 theme.addNoteList(pitches1, durations1)
10
11 # play it
12 Play.midi(theme)
```

Output Latency = 40.0 msec---- Pure Java JSyn www.softsynth.com - rate = 44100, RT, V16.5.14 (build 448, 2012-12-10)

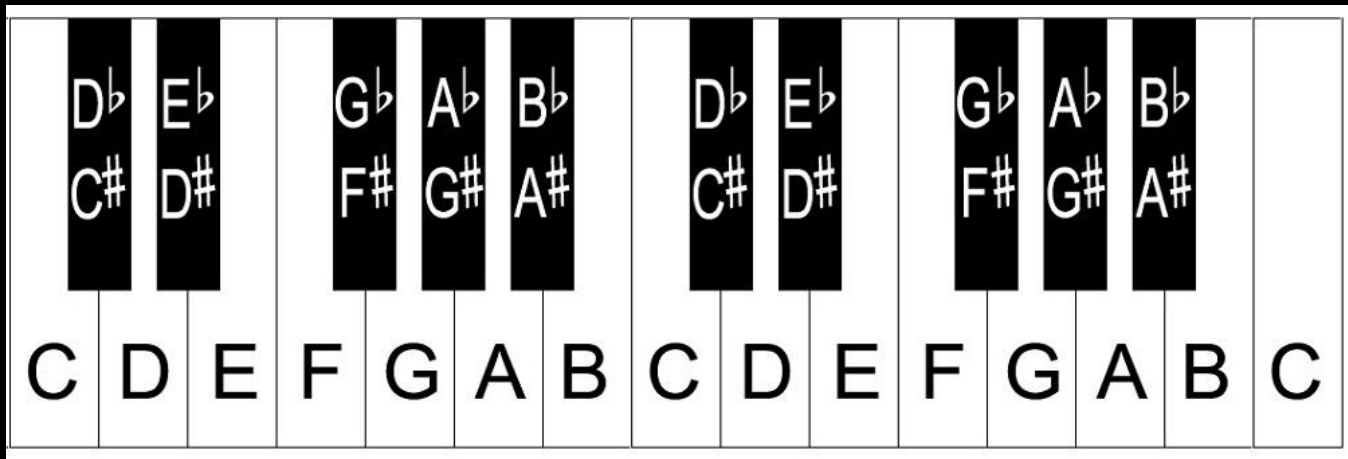
Output buffer size = 7056 bytes.

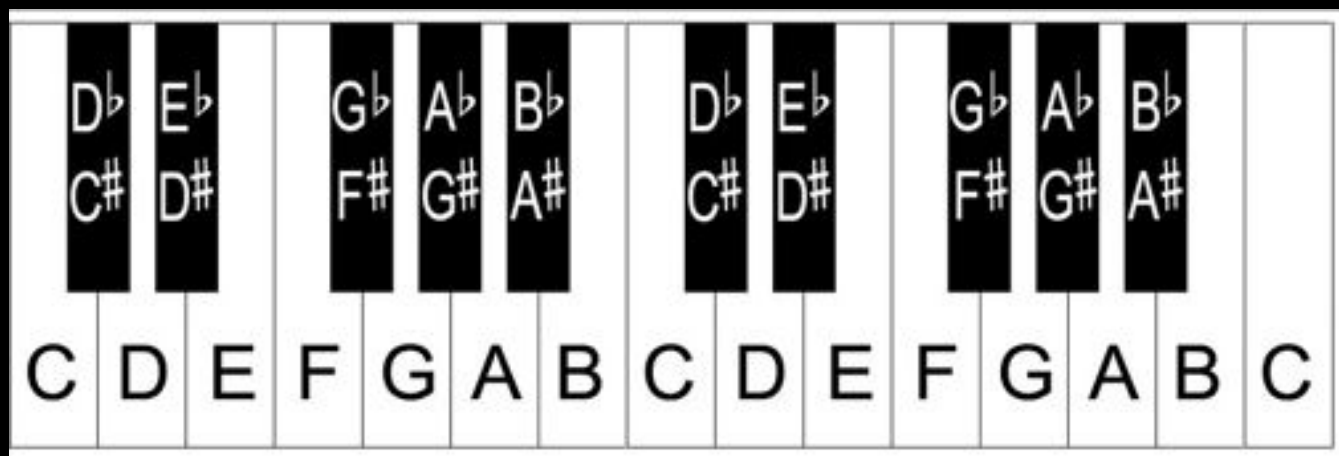


Pyknon

By Pedro Kroger

Book: Music for Geeks & Nerds



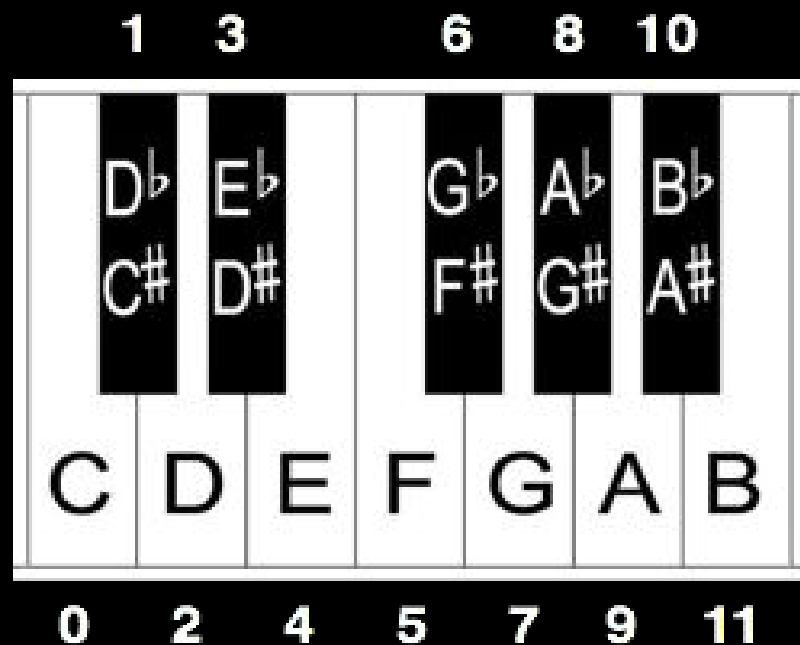


Octave 4

Octave 5

```
>>> from pyknon.music import Note
>>> def mod12(n):
...     return n % 12
...
>>> def note_name(number):
...     notes = "C C# D D# E F F# G G# A A# B".split()
...     return notes[mod12(number)]
```

```
>>> note_name(5)
'F'
>>> note_name(0)
'C'
>>> note_name(60)
'C'
```



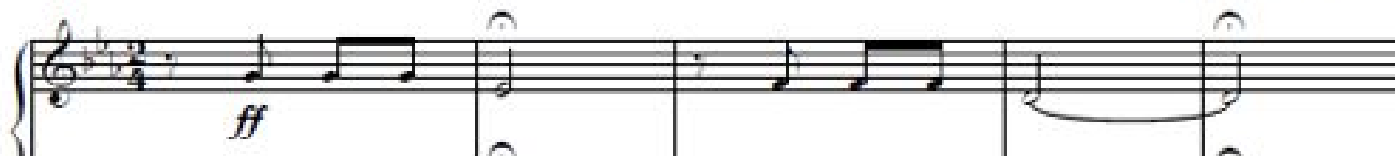
```
>>> from pyknon.music import Note
>>> from pyknon.music import NoteSeq
>>> a = Note()
>>> a
<C>
>>> a.octave
5
>>> a.value
0
>>> a.volume
100
>>> a.dur
0.25
>>> a.midi_number
60
>>> a = Note(3)
>>> a
<D#>
>>> a.octave
5
>>> a.volume
100
>>> a.value
3
>>> a.dur
0.25
>>> a.midi_number
63
```

```
>>> from pyknon.music import NoteSeq
>>> from pyknon.music import Note
>>> NoteSeq([Note(0), Rest(1), Note("C#8")])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Rest' is not defined
>>> from pyknon.music import Rest
>>> NoteSeq([Note(0), Rest(1), Note("C#8")])
<Seq: [<C>, <R: 1>, <C#>]>
>>> NoteSeq("C R C#8")
<Seq: [<C>, <R: 0.25>, <C#>]>
>>> NoteSeq("C#2 D#")
<Seq: [<C#>, <D#>]>
>>> NoteSeq("C# D#")
<Seq: [<C#>, <D#>]>
>>> NoteSeq([Note(0), Note(2)])
<Seq: [<C>, <D>]>
>>> NoteSeq("G G Eb")
<Seq: [<G>, <G>, <D#>]>
>>> NoteSeq("F F F D")
<Seq: [<F>, <F>, <F>, <D>]>
```

```
>>> from pyknon.music import Note
>>> from pyknon.music import NoteSeq
>>> c = Note("C")
>>> c
<C>
>>> c_major_scale = NoteSeq("C D E F G A B")
>>> c_major_scale
<Seq: [<C>, <D>, <E>, <F>, <G>, <A>, <B>]>
>>> c.harmonize(c_major_scale)
[<C>, <E>, <G>]
>>> d = Note("D")
>>> d
<D>
>>> d_major_scale = NoteSeq("D E F# G A B C#)
File "<stdin>", line 1
    d_major_scale = NoteSeq("D E F# G A B C#)
                          ^
SyntaxError: EOL while scanning string literal
>>> d_major_scale = NoteSeq("D E F# G A B C#")
>>> d_major_scale
<Seq: [<D>, <E>, <F#>, <G>, <A>, <B>, <C#>]>
>>> d.harmonize(d_major_scale)
[<D>, <F#>, <A>]
>>> d_major_scale.harmonize(size=4)
[<Seq: [<D>, <F#>, <A>, <C#>]>, <Seq: [<E>, <G>, <B>, <D>]>, <Seq: [<F#>, <A>, <C#>, <E>]>, <Seq: [<G>, <B>, <D>, <F#>]>, <Seq:
[<A>, <C#>, <E>, <G>]>, <Seq: [<B>, <D>, <F#>, <A>]>, <Seq: [<C#>, <E>, <G>, <B>]>]
```


Symphony No.5 in C Minor, Op. 67

Ludwig Van Beethoven



G G G Eb F F F D

Jythn G4 G4 G4 EF4/DS4 F4 F4 F4 D4

Pyknon 7 7 7 3 5 5 5 2
G5 G5 G5 D#5 F5 F5 F5 D5

```
>>> from pyknon.genmidi import Midi
>>> from pyknon.music import NoteSeq
>>> from pyknon.music import Note
>>> notes1=NoteSeq([Note(7, dur=.25), Note(7, dur=.25), Note(7, dur=.25), Note(3, dur=1),
Note(5, dur=.25), Note(5, dur=.25), Note(5, dur=.25), Note(2, dur=1)])
>>> midi = Midi(number_tracks=1, tempo=108)
>>> midi.seq_notes(notes1, track=0)
14.0
>>> midi.write("fifth10.mid")
```



Ria Baldevia
Fifth



2



Add to playlist



Add to group



Share



Download



1


lilybin.com

Apps Gloria Marie Baraqu... cataracthi.com/podc... 26176870701_6017c... TweetDeck m 0 Notifications JavaScript for Cats bash - brew install p... How_to_read_Venetia...

LilyBin Preview Stable Reset Save to Undo Redo Open

```
1 % LilyBin
2 \score{
3   {
4     c'd'e'f'g'a'b'
5   }
6
7   \layout{}
8   \midi{
9     \tempo 4 = 108}
10
11 }
12
```

Page: 1 of 1 Automatic Zoom



The image shows a screenshot of the LilyBin web application. The browser's address bar shows the URL 'lilybin.com'. The top navigation bar includes links for 'Preview', 'Stable', 'Reset', 'Save to', 'Undo', 'Redo', and 'Open'. The main content area is split into two panels. The left panel is a code editor showing LilyPond source code for a musical score. The code defines a score with a single staff containing the notes 'c'd'e'f'g'a'b', a layout, and a MIDI tempo of 4 = 108. The right panel is a preview window showing the rendered musical notation on a treble clef staff. The notation consists of a sequence of eight eighth notes: C, D, E, F, G, A, B, and a final C. The preview window also includes a search bar, navigation arrows, and a zoom control set to 'Automatic Zoom'.

...

	Jython	Pyknon	Others
Center of the Keyboard	Octave 4	Octave 5	.ly ‘
Environment	JEM	Can code from Command Line	EarSketch: application
Difficulty	Beginners	Beginners-Advanced	

