Deriving Music Theory with Python

Western music theory utilizes two main conventions:

1. Doubling or halving the frequency of a note does not fundamentally change its musical function. This is called “octave equivalence.”
2. The octave is divided into 12 equally-spaced parts on a logarithmic scale. This is called “12 equal temperament.” The distance between two neighboring parts is called a semitone.

Let’s divide the octave into 12 intervals, representing distances between the pitch of two notes. “Min” means Minor, “Maj” means Major, “Per” means Perfect and “Dim” means Diminished.

```
from enum import IntEnum
Interval = IntEnum('Interval', 'Unison \
Min2nd Maj2nd Min3rd Maj3rd Per4th Dim5th \
Per5th Min6th Maj6th Min7th Maj7th Octave',
start=0)
>>> print(Interval.Octave.value)
12
```

We also name the twelve notes according to convention – the reason for this convention becomes obvious soon. “sh” stands for sharp (♯) and “b” stands for flat (♭). Sharps raise the pitch of the named note by one semitone, while flats lower it by the same amount. Hence, C ♯ sounds exactly like D ♭ .

```
Note = IntEnum('Note','C Csh_Db D Dsh_Eb E F \
 Fsh_Gb G Gsh_Ab A Ash_Bb B', start=0)
```

Notes can be transposed up or down in pitch by any given interval. Thanks to octave equivalence, notes an octave apart can use the same name. Therefore, the interval addition is performed modulo 12.

```
def transpose(note, interval) -> Note:
    return Note(  (note + interval)
                % Interval.Octave )
def transpose_loop(note, interval, repeat):
    for i in range(repeat):
        note = transpose(note, interval)
    return note
```

We now have enough to derive the C major scale from first principles.

```
C_major_derived = [transpose_loop(
         Note.F, Interval.Per5th, index
         ) for index in range(7)]
C_major = sorted(C_major_derived)
>>> print(C_major)
[<Note.C: 0>, <Note.D: 2>, <Note.E: 4>,
<Note.F: 5>, <Note.G: 7>, <Note.A: 9>,
<Note.B: 11>]
```

By repeatedly transposing the note F up by a perfect fifth, we can see that the C major scale is revealed.

The perfect fifth is a special interval because it sounds very consonant, and it describes a simple frequency ratio of 3/2.

The difference, or distance between two notes is also an interval. The subtraction is performed modulo 12 thanks to octave equivalence.

```
def note_diff(n1, n2):
    return Interval((n1 - n2) % Interval.Octave)
```

The basic triads are the building blocks of western harmony. These are three-note chords obtained by skipping every other note in the major scale. Let’s derive these from the major scale.

```
triads_in_C_major = []
for index in range(len(C_major)):
    # skip every other note
    chord = [ C_major[(index + triad_index)
             % len(C_major)]
           for triad_index in range(0, 5, 2) ]
    triads_in_C_major.append(chord)
>>> print(triads_in_C_major)
[[<Note.C: 0>, <Note.E: 4>, <Note.G: 7>] . . .
```

To really understand these chords, we must take a look at the intervals they are composed of. We define a chord as consisting of a root note followed by a series of intervals representing the distance from the root.

```
chords_in_C_major = {}
for triad in triads_in_C_major:
    root = triad[0]
    qual = [note_diff(note, root)
            for note in triad]
    print(f"{root.name}: {qual[0].name} \
          {qual[1].name} {qual[2].name}")
    chords_in_C_major[root] = qual
```

Let's print these chords in the order the notes were originally derived:

```
>>> for root in C_major_derived:
>>>     qual = chords_in_C_major[root]
>>>     print(f"{root.name}: {qual[0].name} \
>>>           {qual[1].name} {qual[2].name}")
F: Unison Maj3rd Per5th
C: Unison Maj3rd Per5th
G: Unison Maj3rd Per5th
D: Unison Min3rd Per5th
A: Unison Min3rd Per5th
E: Unison Min3rd Per5th
B: Unison Min3rd Dim5th
```

A magical pattern emerges. Triads built on F, C and G contain major thirds, and we call these “major chords.” Triads from D, A and E contain minor thirds, and we call these “minor chords”. B forms a chord with a minor third and a diminished fifth, which we call a “diminished chord.”

If you’d like to hear the notes and chords in this article, I’m afraid you have some work to do, as this is just a zine. However, all you need to know is that you can obtain the MIDI Note Number of any Note in this article by adding 60 to its value.

Full code available at:

https://github.com/tiniuclx/harmonylib