ENGR 103 Studio 4: For and While Loops

Goals:

In this studio, we will get practice using *for* and *while* loops by generating some music. Music is rhythmic and repetitive, so our new looping tools will serve well as building blocks. Some processes that run forever, like a drum beat, can be implemented with a *while* loop. Others happen only a limited number of times (like notes in a scale) and are better suited to a *for* loop. However, as we will see, almost any repetitive process can be accomplished using either form of loop.

Additionally, we will take a look at the concept of callback functions, and what happens when you ask your code to run something *in the background*. For example, in this studio, we will have our code constantly generate both a sound and an animation, even as other parts of our code are executing. How can we control all of these separate processes at once? Callback functions.

Part 1: Getting Familiar with the Code

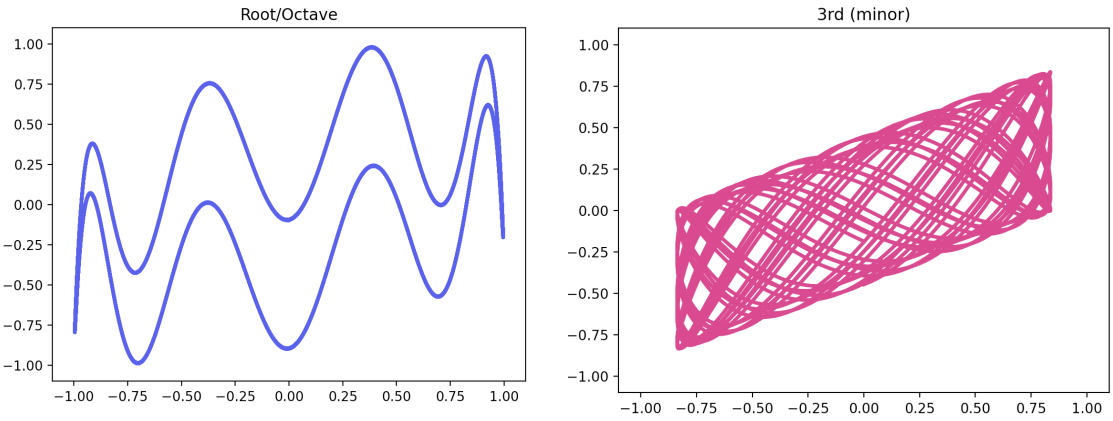
Make it look nice:

Download the Studio 4 code from Canvas and open it in PyCharm. As usual, your first step should be to make the code look readable – press the arrows next to each function definition line to collapse each function into one line. When you are done, the entire code should fit on your screen without scrolling.

Later on, as you expand the functions to observe or edit them, remember to re-collapse them when you are finished. This way, you waste less time scrolling through 100 lines looking for that one function.

Make sure it's working:

Run the code once to see a demonstration and make sure that everything can run on your machine. You will have to download the packages `numpy`, `matplotlib`, `time`, `sounddevice`, and `wavio`. Please use headphones if possible. You should see a "Harmonograph" appear that looks like this:



Side note: If your computer cannot keep up with the continuous animation and sound, we can ease the load. Within the function Harmonograph_Settings, increase the variables `Graph_Update_Interval_ms` and `Sound_Block_Size`, and decrease `NoteHold_Steps_Per_Second`.

Read through and understand:

Go through each function and try to understand what it does. It is not necessary to understand what each line does – just read the function definitions and get a general idea of what is accomplished in each. Some functions do nothing more than define a few variables for later use (like `Harmonograph_Settings`). Some functions only call other functions (like `Initialize`). Some really do perform the work to generate sound or animation data (like `Plot_Callback` or `Sound_Callback`). Some control the

variables used in the callback functions (like `Play_Note)`. Others just contain some pre-made sequences of notes and timing (like `Blues_Riff`).

Deliverable:

1. Describe what the function `Initialize` does and when it is called.
   (Hint: the call to the function *does* appear as a line of code somewhere in the file)
2. Describe what the function `Sound_Callback` does and when it is called.
   (Hint: the call to the function *does not* appear as a line of code anywhere in the file).
3. Describe what `Jeeves` is in this code. (Hint: it is defined in the first line of the `main` function).

Part 2: Writing a Song without Loops

Find which notes you like:

Using the pre-written code as a template, play a sequence of notes that you invent! No previous musical knowledge is required. You can play a note with the function `Play_Note`. It takes two inputs: `Note` and `Duration`. The `Note` is a whole number (-2,-1,0,1,2,3,4…) describing how many keys up or down you want to play from the middle of the piano. The `Duration` is a time in seconds, for how long that note will last.

For example, `Play_Note(Note=0, Duration=1)` will play the note in the middle of the piano for one second. `Play_Note(Note=24, Duration=0.37)` will play 24 notes above the middle of the piano, for 0.37 seconds. The numbers from 0 to 12 correspond to the following notes:

0	1	2	3	4	5	6	7	8	9	10	11	12
Root	Step	2nd	3rd minor	3rd Major	4th	Tri-tone	5th	6th minor	6th Major	7th minor	7th Major	Root/ Octave

See if you can replicate any songs you know, or make your own! In general, a scale is made from the root, 2nd, 3rd, 4th, 5th, 6th, and 7th, and major notes get along with other major notes while minor notes get along with other minor notes. Beware of the tritone and single step; they tend to make disharmonious sounds.

Side note: the `Duration` controls the total time spent playing the note, but the `Note_Hold_Time` controls how long it takes the note's volume to fade. You can make a small blip followed by a pause if you set the `Note_Hold_Time` to be much less than the `Duration`. Conversely if the Note_Hold_Time is much longer than the duration, there will be no noticeable decrease in volume throughout the note.

Deliverable:

4. A .wav file containing your song written without loops. The .wav file should be automatically saved as FirstName_LastName_Part2.wav. Change the filename manually once it's saved, or change the line of code so it saves with your name instead of Carson's. Submit this alongside your .pdf in Canvas, but not within the .pdf.
5. Did you know what you wanted to play in advance, or did you put in some random numbers and listen for what it sounds like?
6. What do you notice about the correlation between the sound and the graph? For instance, do "pretty" sounds look any different than "ugly" sounds?

Part 3: Writing a Song with Loops

This exercise is open-ended; you do not have to adhere to these structures, but you are encouraged to practice using for loops, while loops, and nested loops to make music.

Use a for loop (non-generative):

Use a for loop to simplify your work from the previous part. Rather than separately calling Play_Note() to specify each note you want to play, write your notes in a **list** and loop through them! For example, the following notes will play the first few notes of *Fur Elise*: [7,6,7,6,7,2,5,3,0]

Find the proper syntax for writing loops either by looking around in the code, or by googling "Python for loop." It should look something like this –

```
For each note:
    Play the note
```

You can even wrap this loop into another loop to repeat the sequence, like this –

```
For a few times in a row:
    For each note:
        Play the note
```

A lot of western music plays the same sequence three times in a row, and then a new sequence the fourth time around, like this –

```
For three times in a row:
    For each note in the main sequence:
        Play the note
For each note in the final sequence:
    Play the note
```

Use a for loop (generative):

Use a for loop to generate music that you haven't even written out yet. You could simply move from the bottom note to the top note on the piano:

```
For each note from -36 to 36:
    Play the note
```

It can also be fun to move in patterns. For example, try to play the root note (0), move up 4 notes, then up 3, then up 5, then back 5, and repeat.

Use a while loop:

Use a while loop to extend your musical creation. You could generate random notes until a minor 3rd appears. Or you could use this approach to play forever –

```
While True:
    Keep on playing
```

Deliverable:

7. What was your approach in creating this music? Did you know in advance what you wanted or were you surprised?
8. A screenshot of the loop that created the music
9. A .wav file of your favorite musical creation. Submit alongside your .pdf, not within.