Joyful Noise: Making Music with Python

Login to the PC in front of you and ***click*** the "Jython Music" icon on the desktop.

Note that this software is a free and easy download from jythonmusic.org; feel free to install it on your own computer (no, not now!).

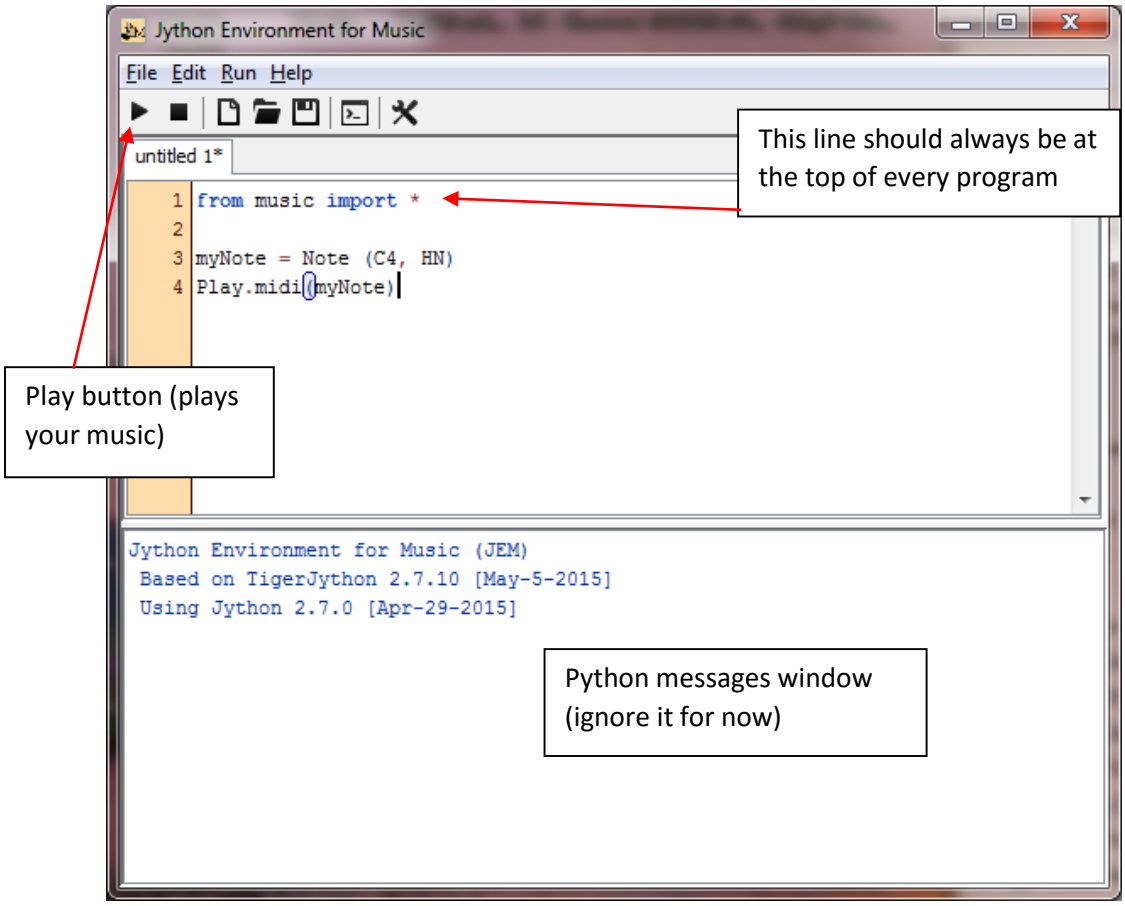
During this workshop you will follow along with me as we play and create some music. At various points we will pause and you will create some music of your own.

The sound will come out of the computer's (not very good) speaker. Please ***feel free to plug in your own earbuds/headphones*** or borrow some from us if you would like a better listening experience! Whatever you use to hear the sounds, your computer's ***volume should be turned ALL THE WAY UP.***

Jython Music allows you to create **notes**, **phrases**, **parts**, and **scores**. It does a whole bunch of other neat things as well, but you'll have to explore those on your own! One important note (pun intended): the programming language we are using today (Python) is case-sensitive: that means "N" and "n" are two different things, so watch out for that.

I. Notes

First we will work on creating different individual notes, then we will put them together into larger units. ***Type*** the lines of text into your JEM window as shown in the following picture:



Once you have typed in the program code (4 lines, one of them blank), ***click*** the "Play" button to hear your note. It will take a few seconds for the playing to occur. Remember to have your volume turned up so you can actually hear it!

In that example, you created and played a note with a specific pitch (C4) and duration (HN). In addition to pitch and duration, notes can also have a "dynamic" (loudness) and "panning" (how much of the sound goes to the left and right stereo speakers).

Change your note (the one named "myNote") as follows:

```
myNote = Note (C4, HN, P, 1.0)
```

When you ***play*** that note (do it now, if you didn't already), you should hear the same note, but softer, and coming entirely out of the right speaker only. What do all these symbols (C4, HN, P, etc.) mean? For all the options, please see the first page of the additional handout.

Take a few minutes now to ***experiment*** with different notes: try at least one different pitch, duration, volume level, and panning setting (that's at least 4 different notes) to get a feel for how you can use those options to create individual notes (use the other handout for a guide as to what options are available for the notes). If you don't want to think of your own, here are a few you could try:

- A0 whole (WN) note, medium volume (MF), center pan (0.5)
- C9 32nd (TN) note, loud volume (FFF), extreme left (0.0)
- G5 half (HN) note, loud volume (FFF), extreme right (1.0)
- Rest (REST) whole (WN) note
- BF3 double whole (WN+WN) note, loud volume (FFF), center pan (0.5)

Listen to each note, possibly referencing the other handout, to ensure you know why you're hearing what you're hearing.

If you have any questions about individual notes, please ask! If you want to save your last individual note to a file, use the "File, Save" to save it to some place on your network drive (do not save it to the PC as you will probably never find it again!). Then use "File, New" to open up a new tab for your next masterpiece.

II. Phrases

OK, that was fun, and even Mozart began with a single note, but we can do more! We can place a sequence of notes into a phrase. ***Type*** the following into your JEM window (the first 2 lines are the same as before). Note that if you leave out the "panning", as we did below, it will be 0.5 (centered):

```
from music import *

note1 = Note (C4, QN, FFF)
note2 = Note (E4, EN, MP)

myPhrase = Phrase()
myPhrase.addNote (note1)
myPhrase.addNote (note2)

Play.midi(myPhrase)
```

Play the phrase. ***Add*** a third note to the phrase that has a different pitch, duration, and volume, and ***play*** your phrase again.

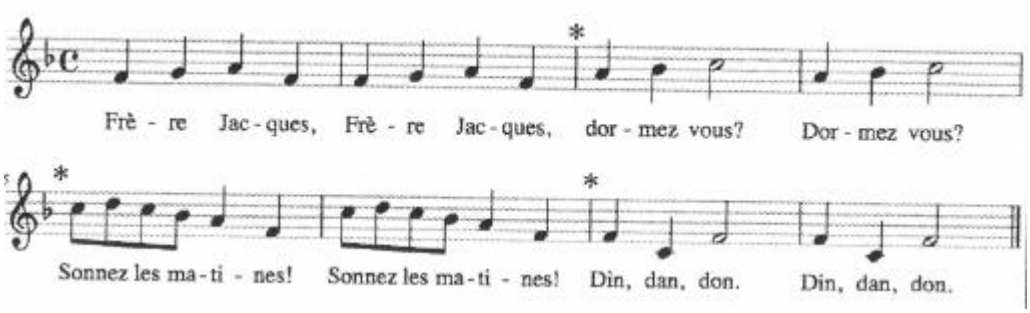
That's not the easiest way to create a phrase – it works, but now we will learn an easier way that will get us closer to something that sounds like real music. If you want to save what you just worked on, follow the instructions at the end of Section I. We can make **lists** of notes and add the whole list to the phrase at once. For now, to simplify things, we will just use the pitches and durations for the notes: we won't specify a volume or panning. ***Delete*** all but the first and last lines of your program (you should have just the "from…" and "Play…" lines remaining). In between them, ***type*** the following lines:

```
pitches =   [G3, G3, G3, DS3]
durations = [EN, EN, EN, HN]
myPhrase = Phrase()
myPhrase.addNoteList (pitches, durations)
```

Before you play it, ***look*** at the pitches and durations and see if you notice a pattern that might help you recognize the music. To help you understand the notation: the first note is a G3 with duration eighth note. So are the second and third notes. The fourth note is a D#3 with duration half note. If this does not help you recognize the tune, that's OK! ***Play*** the phrase to see how the program works.

Now let's look at a slightly bigger example of creating phrases like this. Using the "File, Open" menu item, ***open*** the file "_2_furElise.py". ***Examine*** the program, and you will notice we have several different lists of pitches and rhythms (another word for durations). We add several different lists of notes into the phrase – we even add one of those note lists twice! ***Play*** the music and try to hear the correspondence between the notes in the program and the notes you hear.

Now it's your turn to create a program to play a tune. In this example, we will give you the musical score (see below) and you ***create the program***. Use the "File, New" menu item to open a new tab; you will still have the "Fur Elise" tab open to use as a guide for how to construct your program and ***when you have your song completed, call over the instructor or a TA to listen to it***. HINTS: The first note should be F4, quarter note. Also notice the "B-flat". Take advantage of repetition whenever you can!



Well, that was pretty neat, but you might have some questions. For example, you may wonder "what if I don't want everything to sound like a piano?" or "what if I want to make it go faster?". We can do that! For example, if you have a phrase called "theme", the following line of program code:

```
theme.setInstrument (TRUMPET)
```

will make the phrase sound like a trumpet instead of a piano. Think about where you should put this line of code in the program you just wrote for the song pictured above, ***add*** it, and ***play*** the phrase again. Now change that line to use any other instrument: see pages 2 and 3 of the other handout for a list of possible instruments – pick something interesting and ***try*** it!

The tempo (speed) of a phrase can also be varied. The following line of program code:

```
theme.setTempo(220)
```

will make your song play very fast. ***Try*** it, then ***try*** a slower tempo so you get a feel for what the numbers in parentheses mean. Experiment with different combinations of instrument and tempo. Make sure you ***SAVE*** the final version you try so we can use it later.

III. Parts

If you're ever participated in a musical group (band, chorus, ensemble, etc.) you know that different instruments/voices are producing different sounds at the same time. Jython Music allows us to create these separate parts then combine them into a complete score. Here we will create one simple part, and in the next activity we will create a second part and put them into a score.

Start by ***typing*** in the following program (use "File, New") to create a short trumpet melody part:

```
from music import *

trumpetPart = Part(TRUMPET, 0)

trumpetPhrase = Phrase()
trumpetPitches =   [E4, FS4, G4, C5]
trumpetDurations = [QN, QN,  QN, QN]

trumpetPhrase.addNoteList(trumpetPitches, trumpetDurations)

trumpetPart.addPhrase(trumpetPhrase)

Play.midi(trumpetPart)
```

Notice that a Part consists of Phrases (in this case, to keep it simple, just one phrase) and a Phrase, as we learned earlier, consists of a sequence of Notes (in this case, 4 notes). ***Play*** your part.

IV. Scores

Now you will add a new part to this composition and combine the two parts into a score. Add a second part using VIBES as the instrument:

```
vibesPart = Part (VIBES, 1)
```

The "0" and "1" used when we create the parts tell the computer which "channel" to put the part on: every part gets its own channel. ***Add*** the line above to your program then ***add*** additional lines to create notes and a phrase for the part. *HINT*: You can use any notes you want, as long as the length is correct (4 quarter notes, or 2 quarter notes and a half note, etc.). The pitches are also up to you. Don't worry about how it sounds later along with the trumpet. To hear just the VIBES part, ***change*** the last line of your program to play the "vibesPart" instead of the "trumpetPart" and ***play*** it.

Now that we have both parts created, let's combine them into a score so we will have a complete (but brief) piece of music. ***Add*** the following line just below the "from..." line (make up a better title if you like) to create the score:

```
myScore = Score("My Title", 140)
```

The "140" sets the tempo of the score to 140 beats per minute (a "beat" is one quarter note). Near the end of the program (just above the "Play …" line), ***add*** the two parts to the score. Here is the code to add the trumpet part; you also need to add the vibes part:

```
myScore.addPart (trumpetPart)
```

Finally, ***change*** the "Play" line to play your score instead of just one of the parts, then ***play*** your program to hear what you have created! Feel free to experiment with the tempo or other aspects of the piece. When you are done, ***save*** your composition.

V. Playing Around with Phrases

There are many fun modifications we can make to phrases (these also work with parts and scores). ***Re-open*** your creation from Part II (Frere Jacques) using "File, Open". To speed things up a bit, ***set*** the tempo to at least 200. ***Add*** the following line to the program just before the "Play":

```
Mod.retrograde(theme)
```

and then ***play*** the piece again. What does that do to the music? Now we will ***disable*** that (using "#") and ***add*** a different modification of the phrase:

```
#Mod.retrograde(theme)
Mod.shake(theme)
```

After changing your program as in the two lines above, ***play*** it again to hear what "shake" does. Now ***experiment*** with some other modifications, either using them alone or in combination (e.g., you can shake and shuffle, or ….). Here is a partial list of interesting modifications (try them one at a time at first and try to understand what each one does):

```
Mod.palindrome (theme)

Mod.shuffle (theme)

Mod.invert (theme, A4)

Mod.crescendo (theme, 0, 16, PP, FF)  # try other numbers, volumes

Mod.elongate (theme, 1.5)  #  also try 0.5
```

These and the other available modifications give you a lot of options for being creative with your music. You can apply them to phrases, parts, or entire scores. For example, you could reverse the violin part but not the other parts of a piece and see what that does, etc.

VI. Creation

Now you know enough to create a masterpiece! ***Create*** a new piece of music using a score. The score should have at least 2 parts, and each part should include at least 2 different phrases. Each phrase must consist of at least 4 notes. One of the parts should be modified by a crescendo and the other one by some other modification. Your pieced does not need to sound musical at all (though if it does, that's great!). When you have something you'd like us to hear, ***call over the instructor or a TA to listen to it.***

Save your composition to your network drive.

VII. Other Features …

There is much, much more to the JEM editor and Jython music; we encourage you to check out the web site and see what else is possible! Here are a couple neat things you could try right now:

- Viewing your music using standard notation: If you have a part, phrase, or score named "theme", the following line will display the standard sheet music notation for your composition:

  ```
  View.notation(theme)
  ```

  Of course, it doesn't have to be called "theme", that's just an example!

- You can also create interactive instruments using graphical displays. Opening and playing the file _8_simpleButtonInstrument.py will show you a really simple example, but there is much more you can do, including creating sliding controls (try the program sliderControl.py).