**QBasicMusicPlayer.py**

QBasicMusicPlayer.py is a program that is inspired heavily by Qbasic. I noticed how easy it was for someone to sit down, tell the computer what notes to play and how long, and then boom. It goes and plays it. I want to create a program in python that does something very similar to this. The user will be able to input what notes they want to play, how long it should play, and what octave it should be played on. The program will use the exact (or very similar) syntax that was used to create music in QBasic, but with the power of python behind it interpreting the code and making music with it.

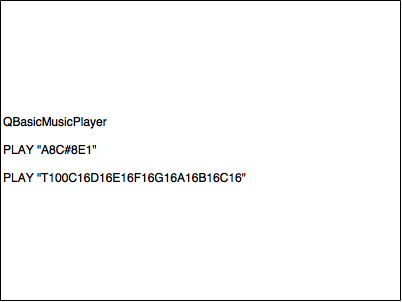
This program can be used by anyone who wants to enjoy making music in a computational way. I also want to target people who have maybe used QBasic (myself not included). I think it would be a really cool thing to see someone who is familiar with that language be able to sit down and just start making cool sounds with my project.

In my research for this project, I’ve noticed that getting sounds to play from a computer is not necessarily an easy thing to just start doing. I want this program to be a way of changing that. I think making music should always be fun, and it helps a lot when your tools work with you instead of against you.

I am going to use pygame.midi to do this by sending notes to a midi keyboard to play. The keyboard could either be a software keyboard or an externally connected keyboard. The testing I have done on my computer was done with a software keyboard called “Virtual MIDI Piano Keyboard,” and it has worked really well.

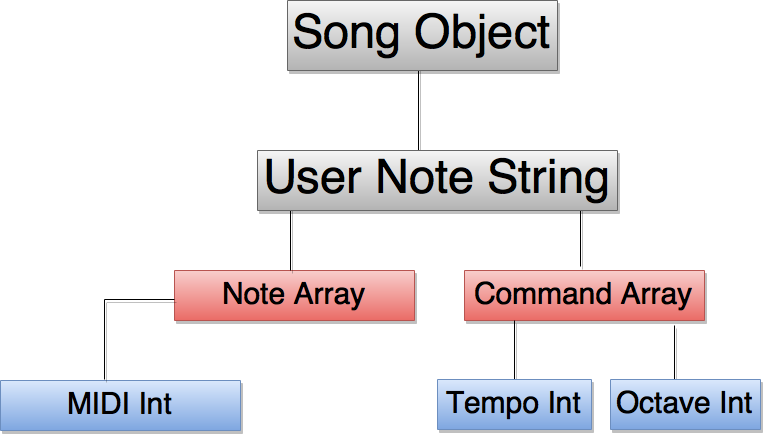
**Use Case:**

User wishes to play a simple melody or a C major scale. They will see a simple screen to type the code. They can then run it with the play button.



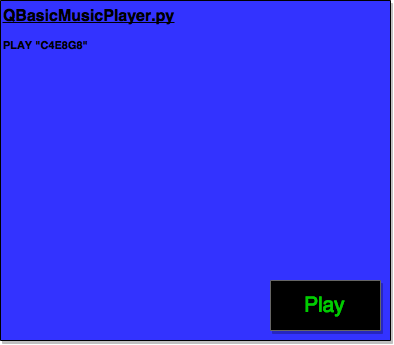
**Data Design:**

The real data behind this program, as with many midi programs, is the midi data. It will be about manipulating the user input into midi data for the synth to be able to play.



**UI Design:**

The UI is really simple because it is a place for the user to type text to be interpreted when the play button is clicked.

****

**Algorithm:**

Class: Song (Inherits from midi.Output)

Attributes:

* \_\_notes - initialize and be static as a list of notes
* \_\_noteValues – the octave 1 midi values for the notes
* \_\_commands - list of commands that can't be changed
* octave - sets what octave the notes are in by being a multiple of 13 that is added or subtracted the midi values. To be changed, runs through setOctave function
* tempo - sets what tempo the song runs in by changing the sleep value between notes
* midiOutput – the device number of what the midi will be sent to
* music - input that is taken from the user as a string and used to play music

Methods:

getOutputDevice () – runs a function in midi.Ouput that finds the device number of a midi device that is currently running on the computer. Then it sets an attribute called midiOutput to be that number.

sortSong () - goes through userNoteText and finds all notes and commands. If it's a note, passes it to noteCreate function. If it is a command, it will pass to the commandExecute function.

noteCreate (musicCounter) – finds the midi value of the note, puts it in the correct octave. If the note is a rest, it changes the velocity to 0, so it will make no sound, just pause. It then runs through the next things in the string to get the note length, which can be 1,2, or even 3 digits in rare cases. When it has this info, it passes it to the playNote function.

commandExecute (musicCounter) - takes the command that is passed to it, finds the corresponding command and runs that code. For instance, if the command is tempo, it passes the new tempo to the changeTempo method. It then adds to the musicCounter to move the program further in the music string.

setOctave (octave) - makes sure that the value input for an octave change is possible. For instance if a user wants to play on octave 20, that is not possible and this will set it back to the highest value of 9 or 10.

changeTempo (tempo) - checks to make sure tempo value is valid. If not, sets it back to a valid number.

playNote (noteNum,length,octave,tempoBPM,velocity) - takes the note information that gets passed into its parameters and then turns that note on using the pygame.midi.noteOn function. Using the duration that comes with the note, it will control how long the system sleeps (using time library in python) before turning the note off, moving on, and playing the next note.

Simply put, the program will initialize and ask the user to play some music. What they input will be set as the music attribute and will then be ran into sortSong which finds out what the current thing in the string is by searching for it in note and command arrays. It then passes that value to the appropriate function to be executed. That function will run, and return the number of characters in the string it used as musicCounter. So for tempo, it will return musicCounter with 3 added on to it because tempo in this program is 3 characters. It will do this until it reaches the end of the string.

Sample Songs:

**Clocks – Coldplay:**

T130 O5 E-8 < B-8 G8 > E-8 < B-8 G8 > E-8 < B-8 > D-8 < B-8 F8 > D-8 < B-8 F8 > D-8 < B-8 > D-8 < B-8 F8 > D-8 < B-8 F8 > D-8 < B-8 > C8 < A-8 F8 > C8 < A-8 F8 > C8 < A-8

**Ode to Joy**

T120 e8e8f8g8g8f8e8d8c8c8d8e8e8d12d4 e8e8f8g8g8f8e8d8c8c8d8e8d8c12c4d8d8e8c8d8e12f12e8c8d8e12f12e8d8c8d8p8e8e8f8g8g8f8e8d8c8c8d8e8d8c12c4

**Regina Spektor – Us**

T176 o2 p8 f8 a-8 f8 a-8 f8 a-8 f8 p8 g-8 b-8 g-8 b-8 g-8 b-8 g-8 p8 f8 a-8 f8 a-8 f8 a-8 f8 p8 g-8 b-8 g-8 b-8 g-8 b-8 g-8 p8 f8 a-8 f8 a-8 f8 a-8 f8 p8 g-8 b-8 g-8 b-8 g-8 b-8 g-8 p8 f8 a-8 f8 a-8 f8 a-8 f8 p8 g-8 b-8 g-8 b-8 g-8 b-8 g-8

**How to Save a Life – The Fray**

T120 o5 b-8 F8 D8 F4 F8 D8 F8 > C8 < F8 D8 F4 F8 D8 F8 > D8 < F8 D8 F4 F8 D8 F8 A8 F8 D8 F4 F8 C8 D8 o5 b-8 F8 D8 F4 F8 D8 F8 > C8 < F8 D8 F4 F8 D8 F8 > D8 < F8 D8 F4 F8 D8 F8 A8 F8 D8 F4 F8 C8 D8 o5 b-8 F8 D8 F4 F8 D8 F8 > C8 < F8 D8 F4 F8 D8 F8 > D8 < F8 D8 F4 F8 D8 F8 A8 F8 D8 F4 F8 C8 D8 o5 b-8 F8 D8 F4 F8 D8 F8 > C8 < F8 D8 F4 F8 D8 F8 > D8 < F8 D8 F4 F8 D8 F8 A8 F8 D8 F4 F8 C8 D8