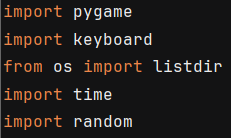
|  |  |
| --- | --- |
| **Assignment 5 Coding** | **Due 7 May** |
| **Name:Tanzil Chowdhury** |
| **Instructions**  This assignment involves implementing code for an element of your project idea or complete the provided coding worksheets.  In either this Word document or a separate document you upload you are expected to write a description of the code you implemented with screenshots of relevant code and outputs. You should also write a reflection offering a detailed analysis of the successes, challenges encountered, key learnings, and strategies for future improvement. A mark scheme can be found at the bottom of this document.  We recommend using the Gibbs Reflective Cycle to structure your reflection: <https://reflection.ed.ac.uk/reflectors-toolkit/reflecting-on-experience/gibbs-reflective-cycle>  You can access the worksheets here: <https://drive.google.com/drive/folders/1EuAYQ9y4KQmWwXMk7h6dG4ILsCfIhz-9?usp=sharing> | |

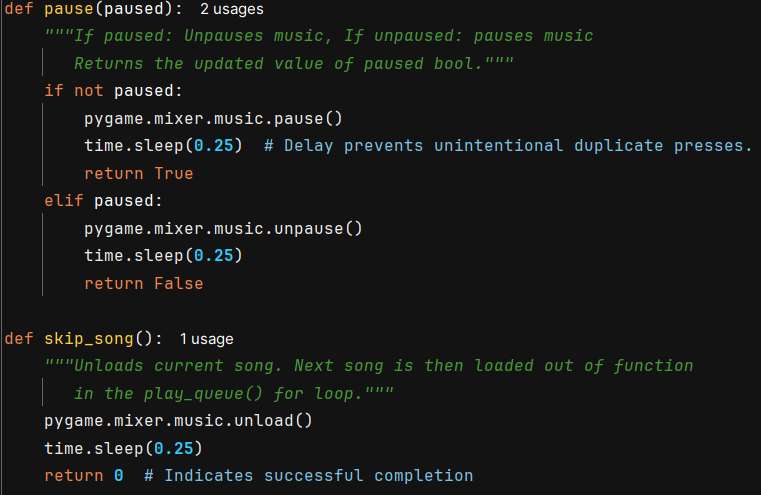
Music Player App

For this project I have developed a music player app to aid students with their learning and letting them focus on their work. Below is my analysis of the development of the actual music player section of the app.

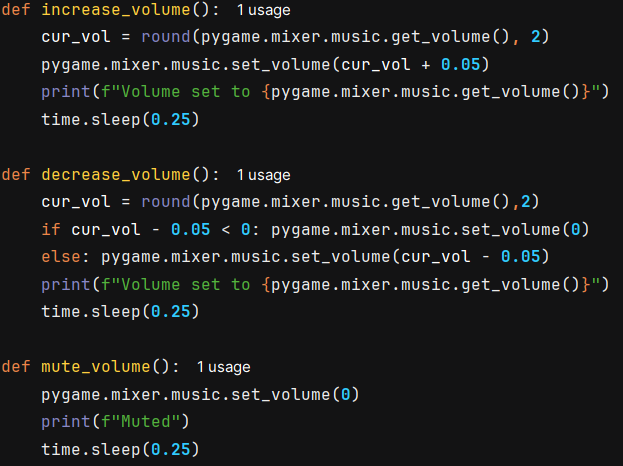
Language

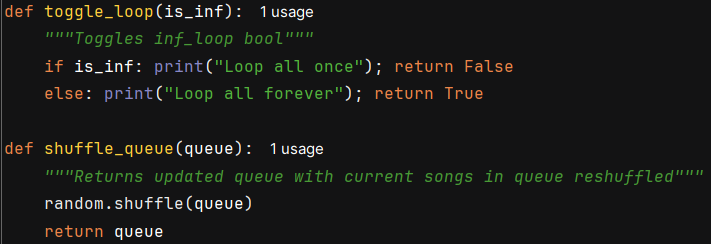
For this project I have chosen to write this program in [Python](https://www.python.org/) due to its extensive number of libraries as well as my familiarity with the language making development of the project easier as less new code would have to be written as the libraires would already provide the code for me.

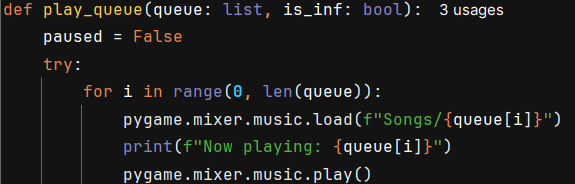
Dependencies  
[Pygame](https://www.pygame.org/news) – I chose to use Pygame for playing the audio files like mp3, and wav as it can handle most of features, I want to implement with its pre-written functions like volume adjustment and pausing whereas other external modules like [playsound](https://github.com/TaylorSMarks/playsound) do not offer such functionality. Furthermore, it can also be used to handle the GUI of the project for a desktop version of the app.   
  
[Keyboard](https://github.com/boppreh/keyboard) – Whilst pygame does offer support for reading keyboard inputs whilst in development I found it to be quite finicky often not picking up on inputs the user made past the first second of booting into the program. I decided that this module would be better to use for inputs due to extended development time caused by trying to find a solution to the problem.  
  
[Os](https://docs.python.org/3/library/os.html) – I use the listdir() function to be able to create a list with all the audio files the user put in the specially created “Songs” directory which is the only directory the program looks for songs from.  
  
[Time](https://docs.python.org/3/library/time.html) – Creates small delays to prevent accidental double clicks from users holding a button.  
  
[Random](https://docs.python.org/3/library/random.html) – Used for the shuffle functionality.  


Code Screenshots & Analysis  


For skip\_song() I chose to only unload the current song being played as this will mean the player is no longer busy and will then move onto the next song in the queue automatically.

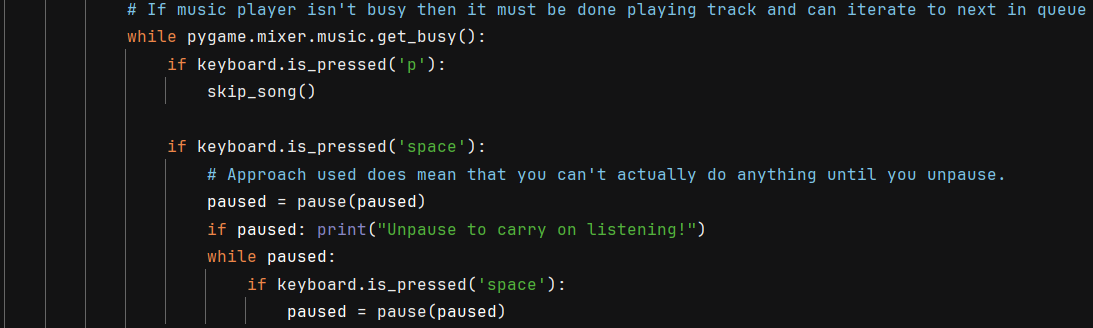
  
The volume in Pygame is measured as a float from 0 to 1. 0 being muted and 1 being the loudest. I wanted my program to have in app volume controls so the user could easily lower the volume of the music whilst not lowering their whole system volume. I wanted to have the volume be in increments of 0.05 (eg: 0.5,0,55,1,0.95) but when manipulating the volume the number is not changed as expected. The round() function is applied to cur\_vol to help mitigate this to some degree but ultimately does not solve the problem.

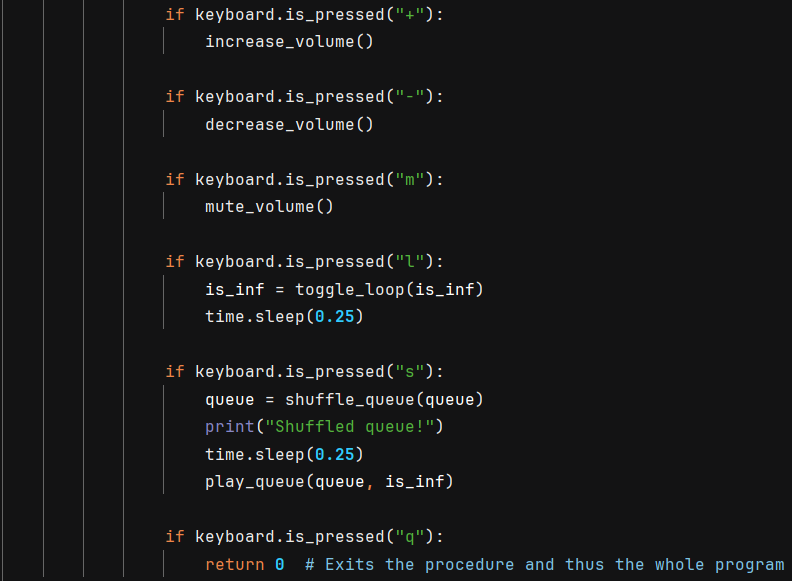




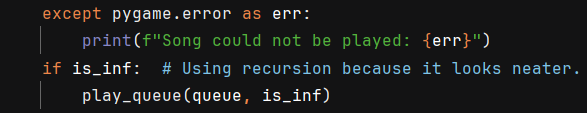
Variable paused needs to be declared so is declared at the beginning of the procedure.

The main for loop is within a try except statement to allow for the program to not crash if a file ends up missing or unable to be played for various reasons. The for loop uses an index to traverse the queue to make skipping and going back to songs easier as you only need to update an index variable. The pygame mixer is initialised outside of the procedure.

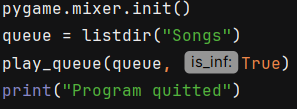
When programming the pause functionality I ran into an issue where pausing the song would automatically flag the mixer as no longer busy therefore automatically skip to the next song. I solved this issue by putting the user in an infinite loop waiting for them to resume the song before being able to carry on using the program as normal.



When choosing the key bindings for the functionalities I tried to go off of what felt right and somewhat intuitive such as making “+” increase volume or “l” being toggling loop as loop begins with l.

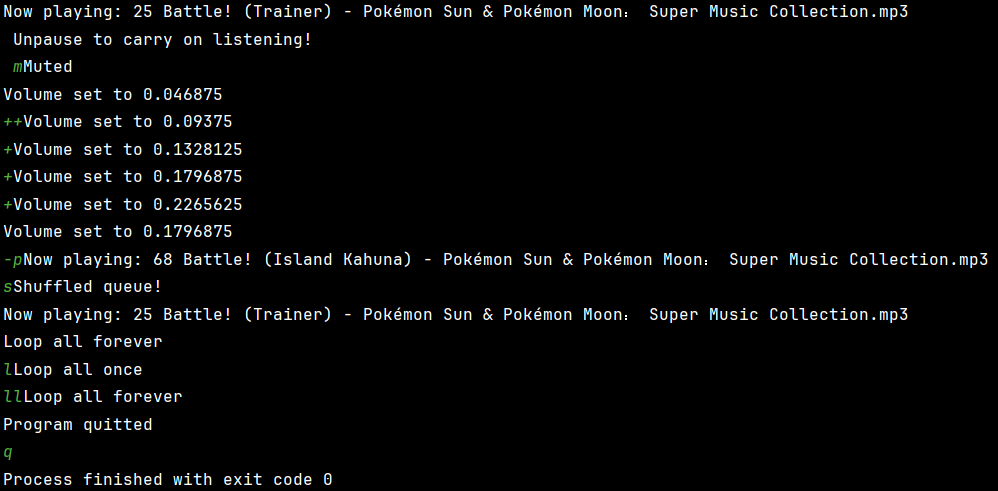


This except statement catches all of the errors the mixer faces. However if a corrupted/invalid file is attempted to be played currently it will stop playing the queue and act as if the queue had finished playing.



listdir() obtains all the files in the “/Songs” directory. (The python file should be kept one above it.)

Example run showing all the features:



Evaluation

This program achieves the basic premise of my goals for the app. A simple non distracting app which lets users listen to music to help them study. It has the basic functionality of the app ideated previously. However, there are still many different things to implement for it to achieve its goal properly.

Future improvements for this app are to fix some of the quirks and bugs this app currently experiences such as the volume control, pausing, error handling, etc. Furthermore, I would like to add a feature to go back in the queue to the previous song. Development was laid out with the use of indexing to traverse the queue. Another development which is a priority is a functional GUI (Graphical User Interface) to make the app easier to use and more pretty to look at then a large monochrome command line.

Other improvements for the future could include integration with [Spotify’s API](https://developer.spotify.com/documentation/web-api) and a way to search more directories for audio files which the user may have to remove the need to move the files into the “Songs” directory.

|  |  |
| --- | --- |
| **Feedback** | |
| **Academic Skills** |  |
| **Description of Code** | Your code and code-style are good. When you’re writing Python, remember to use PyLint to ensure your code is up to style standards (PEP 8) and to catch potential errors early. |
| **Reflection** | Your reflection is good, well done. Could be improved by using the Gibbs’ reflective cycle. |
| **Total Marks 5/6** |  |

|  |  |  |
| --- | --- | --- |
| **Assignment 5**  **Academic Skills**  **Coding**  **Reflecting on Code**  **Total Marks: 6** | * Demonstrates an in-depth understanding of the code with clear descriptions including screenshots of either code or outputs. * The student provides a comprehensive reflection, offering a detailed analysis of the successes, challenges encountered, key learnings, and strategies for future improvement. | 5-6 |
| * Demonstrates an understanding of the code with descriptions including screenshots of either code or outputs. * The student provides a reflection, offering an analysis of the successes, challenges encountered, key learnings, and strategies for future improvement. | 3-4 |
| * Demonstrates limited understanding of the code, limited descriptions or no clear evidence of code or outputs. * The student provides limited reflection on the assignment. | 1-2 |
| No relevant assignment | 0 |