

# Music Player - Report

Anirudh Saikrishnan

May 18, 2023

## 1 Introduction

In this report, we will discuss a Python program that allows users to play songs and shuffle the playlist. The program utilizes the Pygame library for audio playback and does not rely on the random library for shuffling songs.

## 2 Program Overview

The Python program consists of the following key components:

- The main function: This function serves as the entry point of the program. It initializes the Pygame mixer, sets the volume, and handles the user interaction.
- The Shuffle class: This class extends the list class and provides a custom shuffling algorithm using the numpy library. It takes a seed value to generate random numbers for shuffling the playlist.
- The play\_song function: This function is responsible for playing a song using the Pygame mixer. It also includes the functionality to pause and resume the song based on user input.

## 3 Usage

To use the program, follow these steps:

1. Ensure that you have the Pygame library installed.
2. Create a directory called "Songs" and place your MP3 files in it.
3. Run the Python program.
4. Enter "start" to start the program and begin playing songs.
5. Once the program is running, you can enter the following commands:
  - "pause": Pause the currently playing song.
  - "play": Resume playback of the paused song.
  - "next": Skip to the next song in the playlist.
  - "prev": Go back to the previous song in the playlist.
  - "shuffle": Shuffle the playlist and start playing from the beginning.
  - "quit": Exit the program.

## 4 Code Listing

Below is the complete Python code for the program:

```
1 import os
2 import numpy as np
3 import pygame
4
5 pygame.mixer.init()
6 pygame.mixer.music.set_volume(1)
7
8 class custom_random:
9     def __init__(self, seed):
10         self.seed = seed
11
12     def generate(self, max_value):
13         self.seed = (self.seed * 1103515245 + 12345) & 0x7FFFFFFF
14         return self.seed % max_value
15
16 class Shuffle(list):
17     def __init__(self, seed):
18         self.random_generator = custom_random(seed)
19         super().__init__()
20
21     def do(self, l):
22         song_order = []
23         song_list = []
24
25         while len(song_order) != 20:
26             choice = self.random_generator.generate(20)
27             if choice not in song_order:
28                 song_order.append(choice)
29                 song_list.append(l[choice])
30
31         return song_list
32
33 def main():
34     song = 0
35     start_flag = False
36     pauflag = False
37     nextflag = False
38     prevflag = False
39
40     l = [song for song in os.listdir('Songs') if song.endswith('.mp3')]
41     print(l)
42     seed = int(input("Enter a seed value: "))
43     s = Shuffle(seed)
44     l = s.do(l)
45
46     while True:
47         if not start_flag:
48             user_choice = input("Enter 'start' to enter, 'quit' to exit: ")
49             if user_choice == "start":
50                 start_flag = True
51                 play = True
52                 curr_song = l[song]
53                 pygame.mixer.music.load(os.path.join('Songs', curr_song))
54                 pygame.mixer.music.play()
55             elif user_choice == "quit":
56                 pygame.quit()
57                 break
58             else:
59                 print("Invalid choice")
60
61         print("Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'")
62         user_choice = input()
63
64         if user_choice == "next":
```

```

65         if pauflag:
66             song += 1
67             song %= 20
68             nextflag = True
69             curr_song = l[song]
70             pygame.mixer.music.load(os.path.join('Songs', curr_song))
71             print("Current song:", curr_song)
72             continue
73
74         pygame.mixer.music.stop()
75         song += 1
76         song %= 20
77         curr_song = l[song]
78         pygame.mixer.music.load(os.path.join('Songs', curr_song))
79         pygame.mixer.music.play()
80         print("Current song:", curr_song)
81
82     elif user_choice == "prev":
83         if pauflag:
84             song -= 1
85             song %= 20
86             nextflag = True
87             curr_song = l[song]
88             pygame.mixer.music.load(os.path.join('Songs', curr_song))
89             print("Current song:", curr_song)
90             continue
91
92         pygame.mixer.music.stop()
93         song -= 1
94         song %= 20
95         curr_song = l[song]
96         pygame.mixer.music.load(os.path.join('Songs', curr_song))
97         pygame.mixer.music.play()
98         print("Current song:", curr_song)
99
100    elif user_choice == "quit":
101        pygame.quit()
102        break
103
104    elif user_choice == "pause":
105        pygame.mixer.music.pause()
106        pauflag = True
107
108    elif user_choice == "play":
109        if pauflag and (nextflag or prevflag):
110            pauflag = False
111            nextflag = False
112            prevflag = False
113            pygame.mixer.music.play()
114            continue
115
116        pygame.mixer.music.unpause()
117        pauflag = False
118
119    elif user_choice == "shuffle":
120        pygame.mixer.music.stop()
121        seed = int(input("Enter a new seed value:"))
122        s = Shuffle(seed)
123        l = s.do(l)
124        play = True
125        curr_song = l[song]
126        pygame.mixer.music.load(os.path.join('Songs', curr_song))
127        pygame.mixer.music.play()
128        pauflag = False
129        nextflag = False
130        prevflag = False
131        print("Current song:", curr_song)
132

```

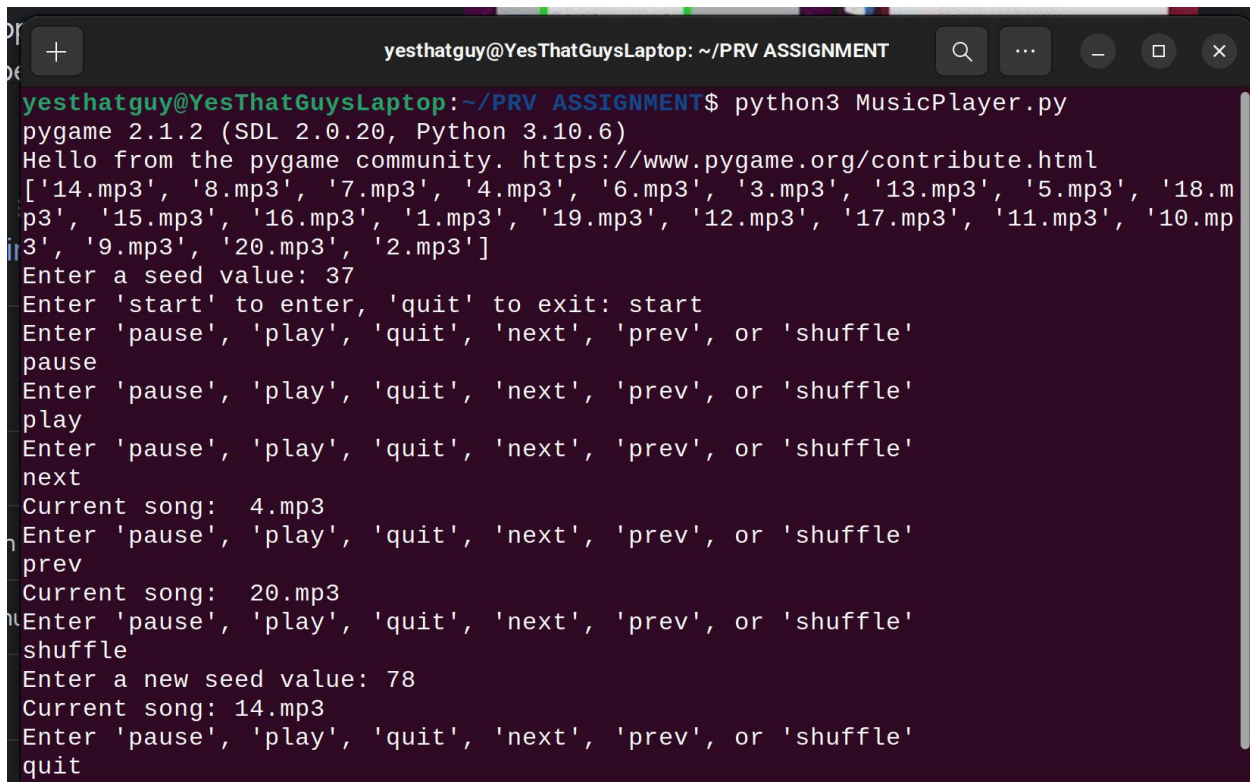
```

133         else:
134             print("Invalid input")
135
136 if __name__ == '__main__':
137     main()

```

## 5 Conclusion

In this report, we have discussed a Python program that enables users to play songs with shuffle functionality. The program utilizes the Pygame library for audio playback and incorporates a custom shuffling algorithm without relying on the random library. With this program, users can enjoy playing their favorite songs in a shuffled order and control the playback using simple commands.



```

+ yesthatguy@YesThatGuysLaptop: ~/PRV ASSIGNMENT
yesthatguy@YesThatGuysLaptop:~/PRV ASSIGNMENT$ python3 MusicPlayer.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
['14.mp3', '8.mp3', '7.mp3', '4.mp3', '6.mp3', '3.mp3', '13.mp3', '5.mp3', '18.m
p3', '15.mp3', '16.mp3', '1.mp3', '19.mp3', '12.mp3', '17.mp3', '11.mp3', '10.mp
3', '9.mp3', '20.mp3', '2.mp3']
Enter a seed value: 37
Enter 'start' to enter, 'quit' to exit: start
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
pause
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
play
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
next
Current song: 4.mp3
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
prev
Current song: 20.mp3
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
shuffle
Enter a new seed value: 78
Current song: 14.mp3
Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'
quit

```

Figure 1: Music Player Functionality