Assignment 2

AI1110: Probability and Random Variables
Indian Institute of Techonology, Hyderabad

Gunethra Bommineni*

Introduction

In this project we were asked to create a music player which can shuffle songs so that no song is repeated until all songs have been played using Numpy and other libraries in Python. I have implemented it using the libraries os, numpy, and mixer from pygame. It features audio play and pause and audio skipping using a python script.

```
import os
import numpy as np
import pygame
from pygame import mixer
```

Listing 1. Libraries used

Shuffling

Numpy library was used to create a list whose size was measured by the ‘num_songs‘ variable. The list of songs was given as input and converted array ‘song‘ and another np array of same size was created using empty-like function in numpy. The new np array with the name ‘shuffled_songs‘ would be used to store the shuffled order. The songs were indexed using ‘np.arange‘ and shuffled using ‘np.random.shuffle‘.

Python function for shuffling: [1]

```
def song_shuffler(songs):
    songs = np.array(songs)
    num_songs = len(songs)
    shuffled_songs = np.empty_like(songs)
    index_pool = np.arange(num_songs)

    np.random.shuffle(index_pool)

    for i in range(num_songs):
        shuffled_songs[i] = songs[index_pool[i
    ]]
        if i < num_songs - 1:
            np.delete(index_pool, np.where(
    index_pool == index_pool[i]))

    return shuffled_songs
```

Listing 2. Shuffling function

*The student is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: ee22btech11205@iith.ac.in.

Player

The audio playback is taken care of, by mixer from the pygame library. The library is initialized by ‘pygame.mixer.init()‘. The function ‘play_songs‘ receives a parameter ‘songs‘, which in this case, is a numpy array of shuffled song indices(shuffled_songs). The variable ‘song_path‘ is a string which stores the path of the required song. The path is a concatenation of the Folder path and the file name of the song. The ‘pygame .mixer.music.play()‘ command is responsible for playing the song. The while loop waits for user input to control the playback. If the user enters

- pause; the playback is paused and ‘pause()‘ function is invoked
- resume; it returns ”Song not paused!”
- next; it plays the next song in the list
- quit; it exits the player

Python function for playback: [2]

```
def play_songs(songs):
    pygame.mixer.init()
    for song in songs:
        print("Now playing:", song)
        song_path = os.path.join(songs_folder,
     song)
        print("From:", song_path)
        pygame.mixer.music.load(song_path)
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy():
            print('*****************') #For
    cosmetic purpose
            command = input("Enter command (
    pause/resume/next/quit): ")

            if command.lower() == "pause":
                pygame.mixer.music.pause()
                pause()
            elif command.lower() == "resume":
                print("Song not paused!")
            elif command.lower() == "next":
                pygame.mixer.music.stop()
                break
            elif command.lower() == "quit":
                quit()
                return
```

Listing 3. Play function

When the ‘pause()‘ function is invoked, the it gives the user three options for playback. The available commands are ”resume”, ”next”, and ”quit”. If the user enters

- resume; playback is resumed
- next; it plays the next song in the list
- quit; it exits the player

Python function for playback: [3]

```
def pause():
    print("Paused")
    print('******************') #For cosmetic
    purpose
    pause_command = input("Enter command (
    resume/next/quit): ")

    if pause_command.lower() == "resume":
        print("Resuming...")
        pygame.mixer.music.unpause()
    elif pause_command.lower() == "next":
        pygame.mixer.music.stop()
    elif pause_command.lower() == "quit":
        quit()
        return
```

Listing 4. Pause function

The main code

The ‘songs_folder‘ variable holds the path to the folder where the songs are located. The for loop initiates the player to play the songs in the shuffled order.

```
songs = ['IMG_0553.wav', 'IMG_0555.wav', '
    IMG_0556.wav', 'IMG_0557.wav', 'IMG_0558.
    wav', 'IMG_0559.wav', 'IMG_0560.wav', '
    IMG_0561.wav', 'IMG_0562.wav', 'IMG_0563.
    wav', 'IMG_0565.wav', 'IMG_0566.wav', '
    IMG_0567.wav', 'IMG_0568.wav', 'IMG_0569.
    wav', 'IMG_0570.wav', 'IMG_0571.wav', '
    IMG_0572.wav', 'IMG_0574.wav', 'IMG_0575.
    wav']

shuffled_songs = song_shuffler(songs)

for song in shuffled_songs:
    print('******************') #For cosmetic
    purpose
    play_songs(shuffled_songs)
```

Listing 5. Main code which invokes the functions

References

[1] https://github.com/Gunethra/AI1110-Software_Project/blob/main/shuffle.py#L8-L24.

[2] https://github.com/Gunethra/AI1110-Software_Project/blob/main/shuffle.py#L26-L51.

[3] https://github.com/Gunethra/AI1110-Software_Project/blob/main/shuffle.py#L53-L68.