Probability Software Assignment

Name -: Arjit Jain
Roll no -: AI22BTECH11002

1 Introduction

The purpose of this report is to explain the implementation of a music playlist using threading in Python. The program utilizes the pygame library for audio playback and employs threads to enable concurrent execution of multiple songs. The user is prompted to proceed to the next song after each playback, and the playlist can be replayed if desired.

2 Implementation

The program consists of the following components:

- **pygame**: Used for audio playback.
- **numpy**: Utilized for generating random numbers
- **threading**: Enables the creation and management of threads.
- **time**: Imported for adding delays.
- **os**: Used for file path operations.

3 Explanation of code

- Defining the play song Function The play song function is responsible for playing an individual song in a separate thread. It takes two arguments: song number and stop event.
  The function loads and plays an audio file corresponding to the song number using pygame.mixer.music. It enters a loop that continues until the song finishes playing or the stop event is set.
- Defining the my playlist Function The my playlist function handles the overall playlist functionality.
  It initializes an empty list, my list, to hold the song numbers. An instance of threading.Event() is created as stop event to control the playback. The pygame.mixer is initialized for audio playback. The main loop runs until my list contains 20 songs. In each iteration, a random number between 1 and 20 (inclusive) is generated and checked if it already exists in my list. If the number is new, it is added to my list. A new thread is created for playing the song using the play song function, passing the song number and stop event. The main thread prompts the user for input to proceed to the next song. If the input is not 'n', the stop event is set to stop the current song, and the loop is broken. If the user input is 'n', the stop event is set to stop the current song, a short delay is added to ensure the song stops completely, and the stop event is cleared for the next song.
- Starting the Playlist The my playlist function is initially called to start the playlist. The program enters a loop that prompts the user whether they want to listen to the playlist again. If the user inputs 'y', the my playlist function is called again to replay the playlist. If the user inputs 'n', the program breaks out of the loop.
- Closing Message After the user decides not to listen to the playlist again, a closing message is printed.

4 Usage

To use the program, follow these steps:

1) Run the program in a Python environment (Python 3 or above).
2) Provide the path to the folder containing the audio files.
3) The program will play the audio files in a random order each time it is run.

5 Conclusion

In conclusion, this report has outlined the implementation of a music playlist using threading in Python. The program leverages the pygame library for audio playback and employs threads to enable concurrent song playback. The user is prompted after each song to proceed to the next one, and the playlist can be replayed if desired. This implementation allows for an interactive and dynamic music experience.

In summary, the random audio player project demonstrates the effective utilization of Python libraries to create an enjoyable audio playback experience.

OUTPUT

Random numbers are generated on the display.

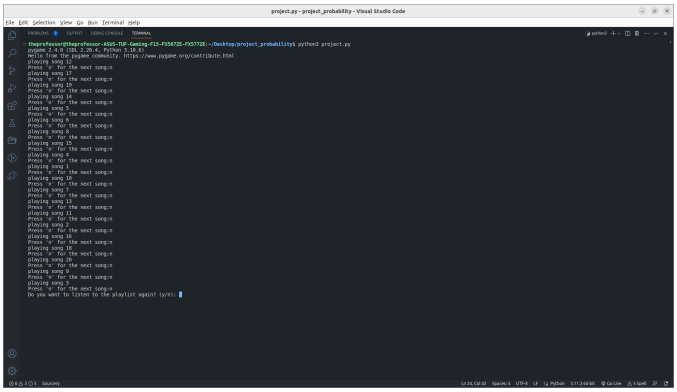

Fig. 1. output