software assignment

E.Naveen Naik
CS22BTECH11021

1 Command-Line Audio Player Report

Q:The purpose of this report is to provide an overview of a command-line audio player implemented in Python using the Pygame and PyDub libraries. The audio player allows users to manage and play a collection of audio files from a specified directory. This report will outline the code's structure, functionality, and usage.

article

2 Introduction

The purpose of this report is to provide an overview of a command-line audio player implemented in Python using the Pygame and PyDub libraries. The audio player allows users to manage and play a collection of audio files from a specified directory. This report will outline the code's structure, functionality, and usage.

3 Code Overview

The provided code consists of several functions and global variables responsible for managing the audio player's behavior. Here's a breakdown of the code's components:

3.1 Global Variables

- `audio_files`: A list containing the names of audio files in the specified directory.
- `current_index`: An index indicating the current position in the playlist.
- `is_playing`: A boolean flag indicating if audio is currently playing.
- `is_paused`: A boolean flag indicating if audio is currently paused.
- `shuffled_indexes`: A list storing shuffled indexes of audio files.

3.2 Loading Audio Files

The `load_audio_files(directory_path)` function loads the names of audio files with the ".mp3" extension from the specified directory. If no audio files are found, an appropriate message is displayed, and the function returns `False`.

3.3 Generating Shuffled Indexes

The `generate_shuffled_indexes()` function creates a shuffled list of indexes corresponding to the audio files in the playlist. It uses the `random.shuffle()` function to randomize the indexes.

3.4 Playback Functions

- `play_audio()`: Sets the playing and paused flags, loads and plays the current audio file.
- `pause_audio()`: Pauses the audio playback if it's currently playing and not paused.
- `resume_audio()`: Resumes the audio playback if it's currently playing and paused.
- `stop_audio()`: Stops the audio playback, resetting playing and paused flags.
- `play_next()`: Stops the current audio playback, moves to the next track in the playlist, and starts playing it.
- `play_previous()`: Stops the current audio playback, moves to the previous track in the playlist, and starts playing it.
- `shuffle_songs()`: Generates a new shuffled list of indexes for the audio files in the playlist.

3.5 User Interaction

- `print_options()`: Prints the available options for the user to interact with the audio player.
- `handle_user_choice(choice)`: Handles the user's input and performs the corresponding action based on the choice selected.

4 Usage

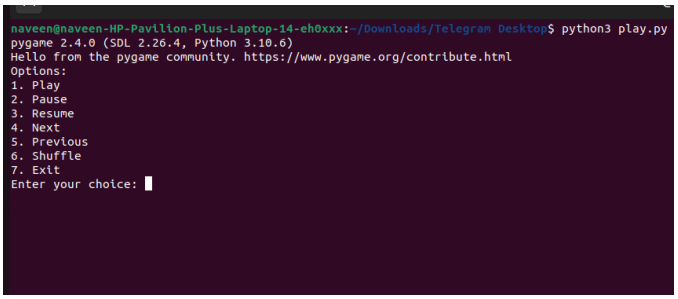
4.1 Directory Path

Before running the program, ensure that the correct directory path containing the audio files is provided. Modify the `directory_path` variable to point to the desired directory.

4.2 Functionality

The audio player offers the following options:

- Play: Starts playing the audio files.
- Pause: Pauses the playback if it's currently playing



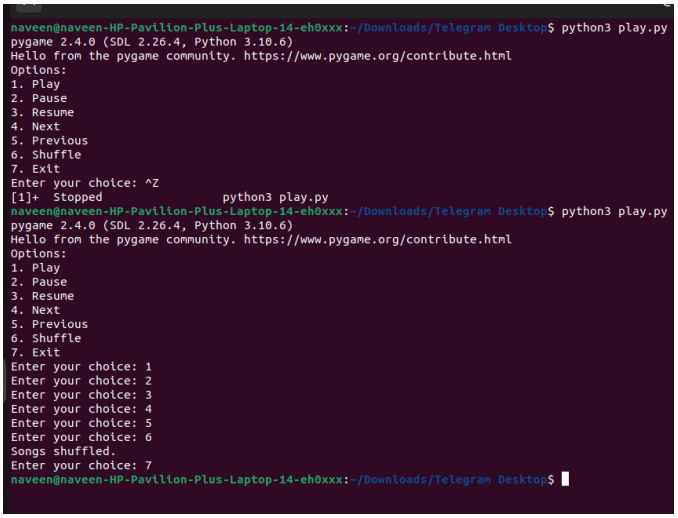


Fig. 0. Two Pictures Side by Side