TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển phần mềm

Play Audio trên PYTHON

GVHD: Từ Lãng Phiêu
SV: SV1 - MSSV
Nguyen Quoc Tai - 3120560085
SV2 - MSSV
Nguyen Hoai Phuc - 3120560075

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 Phần giới thiệu **2**
1.1 Giới thiệu chung . . . 2
1.2 Tổng quan phần mềm . . . 2
1.3 Mục tiêu đề tài . . . 2

2 Giới thiệu Python **2**
2.1 Ứng dụng của Python . . . 2
2.2 Đặc tính của Python . . . 3

3 Các thư viện sử dụng **3**
3.1 Thư viện os . . . 3
3.2 Thư viện Threading . . . 4
3.3 Thư viện Tkinter . . . 4
3.4 Thư viện pygame . . . 5
3.5 Thư viện PIL (Pillow) . . . 5

4 Thiết kế hệ thống **5**
4.1 Thiết kế giao diện người dùng . . . 5

5 Mã nguồn chính **7**
5.1 Import thư viện . . . 7
5.2 Phần 2: Tạo cửa sổ GUI . . . 8
5.3 Phần 3: Khởi tạo các biến toàn cục . . . 8
5.4 Phần 4: Tải các hình ảnh và khởi tạo mixer . . . 8
5.5 Phần 5: Định nghĩa các hàm xử lý . . . 9
5.5.1 Hàm AddMusic() . . . 10
5.5.2 Hàm PlayListSong() . . . 10
5.5.3 Hàm PlayRandomSong() . . . 11
5.5.4 Hàm PlayMusic() . . . 13
5.5.5 Hàm onstop() . . . 13
5.5.6 Hàm onpause() . . . 14
5.5.7 Hàm onresume() . . . 14
5.5.8 Hàm RemoveSong() . . . 14
5.5.9 Hàm ClearPlaylist() . . . 15
5.5.10 Hàm onsliderpress() . . . 15
5.5.11 Hàm onsliderrelease() . . . 15
5.5.12 Hàm updateprogess() . . . 16

1 Phần giới thiệu

1.1 Giới thiệu chung

Trong thời đại công nghệ hiện, thì nhu cầu giải trí hằng ngày của mọi người bằng âm nhạc ngày càng gia tăng. Và các thiết bị phát âm thanh và phần mềm liên quan đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Phần mềm phát âm thanh giúp người dùng dễ dàng quản lý và phát các tệp nhạc của họ, đồng thời cung cấp trải nghiệm nghe nhạc tốt hơn với các tính năng tùy chỉnh như điều khiển âm lượng, tạo danh sách phát và phát nhạc ngẫu nhiên. Với sự phát triển của công nghệ, việc xây dựng phần mềm phát âm thanh không còn là một nhiệm vụ quá phức tạp, đặc biệt là khi sử dụng Python, một ngôn ngữ lập trình mạnh mẽ và dễ học.

1.2 Tổng quan phần mềm

Chương trình này là một phần mềm phát âm thanh được phát triển bằng ngôn ngữ Python với các chức năng phát âm thanh cơ bản: phát nhạc, tạm dừng nhạc,... Bên cạnh đó, chương trình còn sử dụng các thư viện như:

- Tkinter để xây dựng giao diện đồ họa của ứng dụng
- Pygame để xử lý âm thanh (phát âm thanh,...)
- Threading để xử lí các tiến trình song song trong ứng dụng
- PIL để xử lý hình ảnh.

1.3 Mục tiêu đề tài

Mục tiêu của đề tài là phát triển một phần mềm có thể phát nhạc từ các file âm thanh, hỗ trợ các file có định dạng .wav, .mp3, .flac, giao diện của giao diện thân thiện với người dùng và các chức năng điều khiển phát nhạc cơ bản.

2 Giới thiệu Python

Python là một ngôn ngữ lập trình đa mẫu hình, nó hỗ trợ hoàn toàn mẫu lập trình hướng đối tượng và lập trình cấu trúc; ngoài ra về mặt tính năng, Python cũng hỗ trợ lập trình hàm và lập trình hướng khía cạnh. Nhờ vậy mà Python có thể làm được rất nhiều thứ, sử dụng trong nhiều lĩnh vực khác nhau.

2.1 Ứng dụng của Python

- Làm Web với các Framework của Python: Django và Flask là 2 framework phổ biến hiện nay dành cho các lập trình viên Python để tạo ra các website.
- Tool tự động hóa: các ứng dụng như từ điển, crawl dữ liệu từ website, tool giúp tự động hóa công việc được các lập trình viên ưu tiên lựa chọn Python để viết nhờ tốc độ code nhanh của nó.
- Khoa học máy tính: Trong Python có rất nhiều thư viện quan trọng phục vụ cho ngành khoa học máy tính như: OpenCV cho xử lý ảnh và machine learning, Scipy và Numpys cho lĩnh vực toán học, đại số tuyến tính, Pandas cho việc phân tích dữ liệu, ...

- Lĩnh vực IoT: Python có thể viết được các ứng dụng cho nền tảng nhúng, đồng thời cũng được lựa chọn cho việc xử lý dữ liệu lớn. Vì thế Python là một ngôn ngữ quen thuộc trong lĩnh vực Internet kết nối vạn vật
- Làm game: Pygame là một bộ module Python cross-platform được thiết kế để viết game cho cả máy tính và các thiết bị di động

2.2 Đặc tính của Python

Python đang trở nên phổ biến trong cộng đồng lập trình nhờ có các đặc tính sau:

- Ngôn ngữ thông dịch: Python được xử lý trong thời gian chạy bởi Trình thông dịch Python.
- Ngôn ngữ hướng đối tượng: Nó hỗ trợ các tính năng và kỹ thuật lập trình hướng đối tượng.
- Ngôn ngữ lập trình tương tác: Người dùng có thể tương tác trực tiếp với trình thông dịch python để viết chương trình.
- Ngôn ngữ dễ học: Python rất dễ học, đặc biệt là cho người mới bắt đầu.
- Cú pháp đơn giản: Việc hình thành cú pháp Python rất đơn giản và dễ hiểu, điều này cũng làm cho nó trở nên phổ biến.
- Dễ đọc: Mã nguồn Python được xác định rõ ràng và có thể nhìn thấy bằng mắt.
- Di động: Mã Python có thể chạy trên nhiều nền tảng phần cứng có cùng giao diện.
- Có thể mở rộng: Người dùng có thể thêm các mô-đun cấp thấp vào trình thông dịch Python.
- Có thể cải tiến: Python cung cấp một cấu trúc cải tiến để hỗ trợ các chương trình lớn sau đó là shell-script.

3 Các thư viện sử dụng

3.1 Thư viện os

Module os trong Python cung cấp các chức năng được sử dụng để tương tác với hệ điều hành và cũng có được thông tin liên quan về nó. OS đi theo các Module tiện ích tiêu chuẩn của Python. Module này cung cấp một cách linh động sử dụng chức năng phụ thuộc vào hệ điều hành.

Module os trong python cho phép chúng ta làm việc với các tập tin và thư mục.

Các chức năng chính:

- Quản lý tập tin và thư mục
  - Tạo, xóa, đổi tên, và di chuyển tập tin và thư mục.
  - Liệt kê các tập tin và thư mục trong một thư mục cụ thể.
  - Kiểm tra sự tồn tại của tập tin hoặc thư mục.
- Thông tin hệ thống và môi trường
  - Lấy thông tin về người dùng hiện tại.
  - Lấy các biến môi trường.
  - Thay đổi thư mục làm việc hiện tại.

- Thực hiện các lệnh hệ thống
  - Thực thi các lệnh hệ thống và chương trình con từ bên trong Python

Một số hàm thông dụng:

- `os.chdir(path)`: Thay đổi thư mục làm việc hiện tại.
- `os.listdir(path)`: Liệt kê các tập tin và thư mục trong một thư mục cụ thể.

3.2 Thư viện Threading

Cung cấp các công cụ để tạo và quản lý các luồng (threads) trong Python. Một luồng là một dòng thực thi độc lập, cho phép các tác vụ được thực hiện song song, cải thiện hiệu suất và tương tác đồng thời trong ứng dụng.

Các hàm chính trong thư viện threading:

- `threading.Thread(target, args)`: Tạo một luồng mới với một hàm mục tiêu và các đối số tương ứng.
- `start()`: Bắt đầu thực thi luồng.
- `join()`: Chờ đến khi luồng kết thúc.
- `is_alive()`: Kiểm tra xem luồng đang hoạt động hay không.
- `Lock()`: Tạo một đối tượng khóa mới để đồng bộ hóa truy cập vào các tài nguyên chia sẻ.

3.3 Thư viện Tkinter

Tkinter là thư viện GUI tiêu chuẩn cho Python. Tkinter trong Python cung cấp một cách nhanh chóng và dễ dàng để tạo ra các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI.

Đặc điểm của tkinter:

- Khả năng tạo ra các cửa sổ ứng dụng: Tkinter cho phép tạo và quản lý các cửa sổ ứng dụng với các yếu tố giao diện như nút bấm, nhãn, ô nhập liệu, hộp thoại và nhiều thành phần khác.
- Thiết kế giao diện theo mô hình sự kiện: Tkinter sử dụng mô hình lập trình theo sự kiện, trong đó các hành động của người dùng như nhấn nút hoặc nhập liệu sẽ kích hoạt các hàm xử lý sự kiện tương ứng.
- Tính đa nền tảng: Ứng dụng Tkinter có thể chạy trên nhiều hệ điều hành khác nhau như Windows, macOS, và Linux mà không cần thay đổi mã nguồn.

Các bước để tạo một ứng dụng Tkinter:

1. Import Tkinter module. (`from tkinter import *`)
2. Tạo cửa sổ chính của ứng dụng GUI. (`top = Tk()`)
3. Thêm một hoặc nhiều widget nói trên vào ứng dụng GUI.
4. Gọi vòng lặp sự kiện chính để các hành động có thể diễn ra trên màn hình máy tính của người dùng. (`top.mainloop()`)

3.4 Thư viện pygame

Pygame là một thư viện mạnh mẽ để phát triển các trò chơi video và các ứng dụng đa phương tiện trong Python. Trong đề tài này, Pygame được sử dụng để xử lý âm thanh. Một số tính năng nổi bật của Pygame bao gồm:

- Hỗ trợ phát và quản lý âm thanh: Pygame cung cấp các công cụ để tải, phát và điều khiển các tệp âm thanh với nhiều định dạng khác nhau như MP3, WAV, và Ogg.
- Tính năng trộn âm thanh: Thư viện `pygame.mixer` cho phép phát đồng thời nhiều tệp âm thanh, điều chỉnh âm lượng và quản lý các kênh âm thanh.
- Khả năng tích hợp cao: Pygame có thể dễ dàng tích hợp với các thư viện khác của Python để tạo ra các ứng dụng đa phương tiện phức tạp.

3.5 Thư viện PIL (Pillow)

Pillow, trước đây được gọi là PIL (Python Imaging Library), là một thư viện xử lý hình ảnh trong Python, cung cấp nhiều công cụ để thao tác và xử lý các file hình ảnh. Các tính năng chính của Pillow bao gồm:

- Hỗ trợ nhiều định dạng hình ảnh: Pillow có thể mở, lưu và chuyển đổi giữa các định dạng hình ảnh phổ biến như JPEG, PNG, GIF, BMP, và TIFF.
- Xử lý và biến đổi hình ảnh: Thư viện này cung cấp các hàm để thay đổi kích thước, cắt, xoay, lọc, và áp dụng các hiệu ứng đặc biệt lên hình ảnh.
- Tích hợp với Tkinter: Pillow dễ dàng tích hợp với Tkinter để hiển thị hình ảnh trong các ứng dụng GUI.

4 Thiết kế hệ thống

4.1 Thiết kế giao diện người dùng

Giao diện người dùng bao gồm các thành phần chính:

- Nút Play, Pause, Stop, Resume
- Thanh trượt âm lượng
- Thanh trượt tiến trình phát nhạc
- Danh sách phát nhạc
- Nút thêm file và thư mục nhạc
- Nút phát bài hát ngẫu nhiên
- Nút phát lại bài hát ban đầu

Các chức năng chính:

- **Thêm Thư Mục/Nhạc (AddMusicFolder, AddMusic)**: Cho phép người dùng thêm các tập tin nhạc từ thư mục hoặc tập tin cụ thể vào danh sách phát. Các chức năng này được kích hoạt khi nhấp vào các nút "Open File" hoặc "Open Folder".

- **Chơi Danh Sách Phát (PlayListSong)**: Cho phép người dùng chơi toàn bộ danh sách nhạc. Khi một bài hát kết thúc, chương trình sẽ tự động chuyển sang bài hát tiếp theo trong danh sách.
- **Chơi Ngẫu Nhiên (PlayRandomSong)**: Chơi một bài hát ngẫu nhiên từ danh sách nhạc. Mỗi khi bài hát kết thúc, một bài hát ngẫu nhiên khác sẽ được chọn và phát.
- **Chơi Nhạc (PlayMusic)**: Chơi bài hát được chọn từ danh sách nhạc.
- **Dừng Nhạc (onstop)**: Dừng bài hát đang phát.
- **Tạm Dừng Nhạc (onpause)**: Tạm dừng bài hát đang phát.
- **Tiếp Tục Nhạc (onresume)**: Tiếp tục phát nhạc sau khi đã tạm dừng.
- **Điều Chỉnh Thanh Tiến Trình Nhạc (adjust_progess)**: Cho phép người dùng điều chỉnh thanh tiến trình để tua nhanh hoặc tua lùi bài hát.
- **Điều Chỉnh Âm Lượng (set_volume)**: Cho phép người dùng điều chỉnh âm lượng của bài hát.

Sơ đồ hoạt động:

1. **Khởi tạo Giao Diện Người Dùng (GUI)**:
   - Tạo cửa sổ giao diện người dùng sử dụng thư viện tkinter.
   - Định cấu hình kích thước, màu nền và tiêu đề cho cửa sổ.
2. **Thêm Nhạc từ Thư Mục hoặc Tập Tin**:
   - Người dùng có thể thêm nhạc từ thư mục hoặc tập tin cụ thể vào danh sách phát.
   - Sử dụng hộp thoại để chọn thư mục hoặc tập tin nhạc và thêm chúng vào danh sách phát.
3. **Phát Nhạc từ Danh Sách Phát (PlayListSong)**:
   - Cho phép người dùng chơi toàn bộ danh sách nhạc.
   - Lặp qua danh sách nhạc và phát từng bài hát một.
   - Khi một bài hát kết thúc, chuyển sang bài hát tiếp theo trong danh sách.
4. **Phát Bài Hát Ngẫu Nhiên (PlayRandomSong)**:
   - Chọn một bài hát ngẫu nhiên từ danh sách nhạc và phát nó.
   - Lặp lại quá trình này cho đến khi người dùng dừng lại hoặc thoát khỏi chương trình.
5. **Chơi, Dừng, Tạm Dừng, Tiếp Tục Phát Nhạc**:
   - Cung cấp các chức năng chơi, dừng, tạm dừng và tiếp tục phát nhạc cho người dùng.
   - Khi nhấn vào các nút tương ứng, chương trình thực hiện hành động tương ứng với chức năng đã chọn.
6. **Điều Chỉnh Thanh Tiến Trình và Âm Lượng**:
   - Cho phép người dùng điều chỉnh thanh tiến trình để tua nhanh hoặc tua lùi bài hát.

- Cho phép người dùng điều chỉnh âm lượng của bài hát.

7. **Cập Nhật Tiến Trình Phát Nhạc (update_progess)**:
   - Theo dõi tiến trình phát nhạc và cập nhật thanh tiến trình tương ứng trên giao diện người dùng.
   - Chương trình sẽ tự động cập nhật thanh tiến trình mỗi khi bài hát đang phát.
8. **Thread (Luồng)**:
   - Sử dụng luồng (thread) để cập nhật tiến trình phát nhạc mà không làm tắc nghẽn giao diện người dùng.
   - Luồng này chạy đồng thời với giao diện người dùng để cập nhật thanh tiến trình phát nhạc.
9. **Mainloop**:
   - Khởi động vòng lặp chính (mainloop) để chạy ứng dụng và đợi sự tương tác từ phía người dùng.
   - Ứng dụng sẽ tiếp tục chạy cho đến khi người dùng đóng cửa sổ hoặc thoát khỏi chương trình.

5 Mã nguồn chính

5.1 Import thư viện

```
import os
import time
import threading
import random
import tkinter as tk
from tkinter import *
from tkinter import Tk
from tkinter import filedialog
from pygame import mixer
from pygame import time
import pygame
from PIL import Image, ImageTk
```

Phần này import các thư viện cần thiết cho chương trình:

- **os**: Thư viện hệ thống để tương tác với hệ điều hành.
- **time**: Thư viện xử lý thời gian.
- **threading**: Thư viện để tạo và quản lý các luồng (threads).
- **random**: Thư viện để tạo ra các số ngẫu nhiên.
- **tkinter**: Thư viện giao diện đồ họa để tạo cửa sổ và các thành phần giao diện.
- **pygame**: Thư viện để xử lý âm thanh và đa phương tiện.
- **PIL (Python Imaging Library)**: Thư viện để xử lý hình ảnh.

5.2 Phần 2: Tạo cửa sổ GUI

```
root = tk.Tk()
root.title("Music Player")
root.geometry("920x735+400+85")
root.configure(background='#A43B84')
root.resizable(False,False)
```

Phần này tạo ra một cửa sổ GUI với các tùy chọn sau:

- **root = tk.Tk()**: Tạo một đối tượng cửa sổ gốc của Tkinter.
- **root.title("Music Player")**: Đặt tiêu đề cho cửa sổ là "Music Player".
- **root.geometry("920x735+400+85")**: Đặt kích thước và vị trí của cửa sổ trên màn hình (920 pixel rộng, 735 pixel cao, vị trí (400, 85) từ góc trên bên trái màn hình).
- **root.configure(background='A43B84')**: Đặt màu nền của cửa sổ là mã màu hex A43B84.
- **root.resizable(False,False)**: Không cho phép thay đổi kích thước của cửa sổ theo chiều ngang và chiều dọc.

5.3 Phần 3: Khởi tạo các biến toàn cục

```
Song_selected = False
auto = True
count = 0
allstop = False
tempstop = False
```

Phần này khởi tạo một số biến toàn cục để kiểm soát trạng thái của ứng dụng:

- **Song_selected = False**: Biến boolean để theo dõi xem có bài hát nào được chọn hay không.
- **auto = True**: Biến boolean để kiểm soát chế độ phát tự động.
- **count = 0**: Biến đếm để theo dõi tiến trình phát của bài hát.
- **allstop = False**: Biến boolean để kiểm soát việc dừng phát tất cả các bài hát.
- **tempstop = False**: Biến boolean để kiểm soát việc tạm dừng phát bài hát hiện tại.

5.4 Phần 4: Tải các hình ảnh và khởi tạo mixer

```
ButtonPlay = Image.open("music/playresize(2).png")
ButtonStop = Image.open("music/stopMusic.png")
ButtonPause = Image.open("music/pauseresize.png")
ButtonResume = Image.open("music/resumeMusic.png")
ButtonRandom = Image.open("music/random.png")
ButtonPLaylist = Image.open("music/playlist.png")
```

```
ButtonOpenFile = Image.open("music/openFile.png")
ButtonOpenFolder = Image.open("music/openFolder.png")
ButtonPrevious = Image.open("music/previous.png") # Thm hnh nh cho nt
    Previous
ButtonNext = Image.open("music/next.png")
ButtonRemoveSong = Image.open("music/delete.png")
ButtonClearPlaylist = Image.open("music/delete_all.png")
mixer.init()
pygame.init()
```

Phần này:

- Tải các hình ảnh sẽ được sử dụng cho các nút trên giao diện đồ họa bằng cách sử dụng thư viện PIL.
- Khởi tạo mixer của pygame để xử lý âm thanh bằng lệnh **mixer.init()**.
- Khởi tạo pygame bằng lệnh **pygame.init()** để có thể sử dụng các tính năng khác của pygame.

5.5 Phần 5: Định nghĩa các hàm xử lý

```
def AddMusicFolder():
    path = filedialog.askdirectory()
    if path:
        os.chdir(path)
        songs = os.listdir(path)

        for song in songs:
            flag = False
            if song.endswith(".mp3") or song.endswith(".wav") or
                song.endswith(".flac") :
                for i in range(0,Playlist.size()):
                    if(song == Playlist.get(i)):
                        flag = True
                if flag == False:
                    Playlist.insert(END, song)
```

Hàm này cho phép người dùng chọn một thư mục chứa các file nhạc và thêm tất cả các file nhạc có đuôi `.mp3`, `.wav` hoặc `.flac` vào danh sách phát `Playlist`. Cụ thể:

- **filedialog.askdirectory()**: Mở hộp thoại cho người dùng chọn thư mục.
- **os.chdir(path)**: Thay đổi thư mục làm việc hiện tại thành thư mục đã chọn.
- **os.listdir(path)**: Lấy danh sách các file và thư mục con trong thư mục đã chọn.
- Với mỗi file trong danh sách, kiểm tra xem file đó có phải là file nhạc hay không bằng cách kiểm tra đuôi file.
- Nếu file đó là file nhạc, kiểm tra xem nó đã có trong danh sách phát hay chưa bằng cách duyệt qua danh sách phát.
- Nếu file chưa có trong danh sách phát, thêm nó vào cuối danh sách bằng **Playlist.insert(END, song)**.

5.5.1 Hàm AddMusic()

```
def AddMusic():
    flag = False
    path = filedialog.askopenfilename()
    print(path)
    songpath = os.path.dirname(path)
    os.chdir(songpath)
    song = os.path.basename(path)
    if song.endswith(".mp3") or song.endswith(".wav") or song.endswith(".flac")
        :
        for i in range(0,Playlist.size()):
            if(song == Playlist.get(i)):
                flag = True
        if flag == False:
            Playlist.insert(END,song)
```

Hàm này cho phép người dùng chọn một file nhạc cụ thể và thêm nó vào danh sách phát `Playlist`. Cụ thể:

- **filedialog.askopenfilename()**: Mở hộp thoại cho người dùng chọn file.
- **os.path.dirname(path)**: Lấy đường dẫn thư mục chứa file đã chọn.
- **os.chdir(songpath)**: Thay đổi thư mục làm việc hiện tại thành thư mục chứa file đã chọn.
- **os.path.basename(path)**: Lấy tên file đã chọn.
- Kiểm tra xem file đã chọn có phải là file nhạc hay không bằng cách kiểm tra đuôi file.
- Nếu file đó là file nhạc, kiểm tra xem nó đã có trong danh sách phát hay chưa bằng cách duyệt qua danh sách phát.
- Nếu file chưa có trong danh sách phát, thêm nó vào cuối danh sách bằng **Playlist.insert(END, song)**.

5.5.2 Hàm PlayListSong()

```
def PlayListSong():
    global allstop
    global count
    global tempstop
    allstop = False
    global Song_selected
    x = 0
    num_items = Playlist.size()
    flag = False
    if num_items > 0:
        while not allstop:
            if allstop:
                break
            i = 0
            while(i <= num_items):
```

```
            if allstop:
                break
            if(i == num_items):
                i = 0
            if(flag == True):
                flag = False
                i = i - 1
                item = Playlist.get(i)
                count = MusicProgess_slider.get()
                mixer.music.play(0,period*count)
            else:
                item = Playlist.get(i)
                Song_selected = True
                mixer.music.load(item)
                mixer.music.play()
                count = 0
    # Wait for the song to finish playing
            while mixer.music.get_busy():
                root.update()
            if(tempstop):
                music = mixer.Sound(Playlist.get(ACTIVE))
                music_length_in_seconds = music.get_length()
                period = (music_length_in_seconds/100)
                while tempstop:
                    flag = True
                    root.update()
                    if not tempstop:
                        break
            i += 1
```

Hàm này phát các bài hát trong danh sách phát `Playlist` theo thứ tự. Cụ thể:

- Khởi tạo các biến toàn cục `allstop`, `count`, `tempstop` và `Song_selected`.
- Lấy số lượng bài hát trong danh sách phát bằng `Playlist.size()`.
- Nếu danh sách phát không rỗng:
  - Chạy vòng lặp vô tận cho đến khi biến `allstop` được đặt thành `True`.
  - Trong vòng lặp:
    - Nếu biến `flag` đúng, nghĩa là bài hát đang phát bị tạm dừng, phát tiếp bài hát đó từ vị trí đã tạm dừng.
    - Nếu biến `flag` sai, phát bài hát tiếp theo trong danh sách.
    - Đợi cho đến khi bài hát kết thúc bằng `mixer.music.get_busy()`.
    - Nếu biến `tempstop` đúng, nghĩa là bài hát đang phát bị tạm dừng, thì đặt biến `flag` thành `True` và tiếp tục chờ.
  - Tăng biến đếm `i` để chuyển sang bài hát tiếp theo.

5.5.3 Hàm PlayRandomSong()

```
def PlayRandomSong():
    global count
    global Song_selected
    global allstop
    global tempstop
    mark = -1
    flag = False
    allstop = False
    x = 0
    num_items = Playlist.size()
    if num_items > 0:
        while not allstop:
            if allstop:
                break
            random_index = random.randint(0, num_items - 1)
            if (random_index != x or num_items == 1):
                if allstop:
                 break
                if(flag == True):
                    flag = False
                    random_index = mark
                    item = Playlist.get(random_index)
                    count = MusicProgess_slider.get()
                    mixer.music.play(0,period*count)
                else:
                    item = Playlist.get(random_index)
                    Song_selected = True
                    mixer.music.load(item)
                    mixer.music.play()
                    count = 0
                print(random_index)
                print(item)
                x = random_index
        # Wait for the song to finish playing
                while mixer.music.get_busy():
                    root.update()
                if(tempstop):
                    music = mixer.Sound(Playlist.get(ACTIVE))
                    music_length_in_seconds = music.get_length()
                    period = (music_length_in_seconds/100)
                    mark = random_index
                    while tempstop:
                        flag = True
                        root.update()
                        if not tempstop:
                            break
```

Hàm này phát các bài hát trong danh sách phát `Playlist` theo thứ tự ngẫu nhiên. Cụ thể:

- Khởi tạo các biến toàn cục `count`, `Song_selected`, `allstop`, `tempstop`.
- Khởi tạo biến `mark` để đánh dấu bài hát đang phát, và biến `flag` để đánh dấu trạng

thái tạm dừng.

- Lấy số lượng bài hát trong danh sách phát bằng `Playlist.size()`.
- Nếu danh sách phát không rỗng:
  - Chạy vòng lặp vô tận cho đến khi biến `allstop` được đặt thành `True`.
  - Trong vòng lặp:
    - Chọn một chỉ số ngẫu nhiên bằng `random.randint(0, num_items - 1)`.
    - Nếu chỉ số ngẫu nhiên khác với chỉ số bài hát hiện tại (trừ khi chỉ có một bài hát trong danh sách):
      - Nếu biến `flag` đúng, nghĩa là bài hát đang phát bị tạm dừng, phát tiếp bài hát đó từ vị trí đã tạm dừng.
      - Nếu biến `flag` sai, phát bài hát tại chỉ số ngẫu nhiên.
      - Đợi cho đến khi bài hát kết thúc bằng `mixer.music.get_busy()`.
      - Nếu biến `tempstop` đúng, nghĩa là bài hát đang phát bị tạm dừng, đánh dấu chỉ số hiện tại vào biến `mark`, đặt biến `flag` thành `True` và tiếp tục chờ.

5.5.4 Hàm PlayMusic()

```
def PlayMusic():
    global allstop
    allstop = False
    global count
    global Song_selected
    Music_Name = Playlist.get(ACTIVE)
    print(Music_Name)
    Song_selected = True
    mixer.music.load(Playlist.get(ACTIVE))
    mixer.music.play()
    count = 0
```

Hàm này phát bài hát hiện đang được chọn trong danh sách phát `Playlist`. Cụ thể:

- Đặt biến toàn cục `allstop` thành `False` để cho phép phát nhạc.
- Lấy tên bài hát đang được chọn bằng `Playlist.get(ACTIVE)`.
- Đặt biến `Song_selected` thành `True` để đánh dấu rằng có bài hát đang được chọn.
- Tải bài hát đang được chọn vào mixer bằng `mixer.music.load(Playlist.get(ACTIVE))`.
- Phát bài hát bằng `mixer.music.play()`.
- Đặt biến `count` (theo dõi tiến trình phát của bài hát) về 0.

5.5.5 Hàm onstop()

```
def onstop():
    global allstop
    mixer.music.stop()
    allstop = True
```

Hàm này dừng phát bài hát hiện tại. Cụ thể:

- Gọi `mixer.music.stop()` để dừng phát bài hát.
- Đặt biến toàn cục `allstop` thành `True` để ngăn không cho phát bài hát khác.

5.5.6 Hàm onpause()

```
def onpause():
    global tempstop
    mixer.music.pause()
    tempstop = True
```

Hàm này tạm dừng phát bài hát hiện tại. Cụ thể:

- Gọi `mixer.music.pause()` để tạm dừng phát bài hát.
- Đặt biến toàn cục `tempstop` thành `True` để đánh dấu rằng bài hát đang bị tạm dừng.

5.5.7 Hàm onresume()

```
def onresume():
    global tempstop
    mixer.music.unpause()
    tempstop = False
```

Hàm này tiếp tục phát bài hát đã bị tạm dừng. Cụ thể:

- Gọi `mixer.music.unpause()` để tiếp tục phát bài hát.
- Đặt biến toàn cục `tempstop` thành `False` để đánh dấu rằng bài hát không còn bị tạm dừng nữa.

5.5.8 Hàm RemoveSong()

```
def RemoveSong():
global Song_selected
current_selection = Playlist.curselection()
if current_selection:
Playlist.delete(current_selection)
Song_selected = False
```

Hàm này xóa bài hát đang được chọn khỏi danh sách phát `Playlist`. Cụ thể:

- Lấy chỉ số của bài hát đang được chọn bằng `Playlist.curselection()`.
- Nếu có bài hát được chọn (danh sách chỉ số không rỗng):
  - Xóa bài hát khỏi danh sách phát bằng `Playlist.delete(current`$_s$*election*).Œ*tbin*

5.5.9 Hàm ClearPlaylist()

```
def ClearPlaylist():

global Song_selected
Playlist.delete(0, END)
Song_selected = False
```

Hàm này xóa tất cả các bài hát khỏi danh sách phát `Playlist`. Cụ thể:

- Xóa tất cả các phần tử trong danh sách phát bằng `Playlist.delete(0, END)`.

5.5.10 Hàm onsliderpress()

```
def on_slider_press():
global tempstop
global auto
auto = False
tempstop = True
mixer.music.stop()
```

Hàm này được gọi khi thanh trượt tiến trình bài hát (MusicProgessslider) bắt đầu được nhấn giữ. Cụ thể:

- Đặt biến toàn cục `auto` thành `False` để tắt chế độ phát tự động.
- Đặt biến toàn cục `tempstop` thành `True` để đánh dấu rằng bài hát đang bị tạm dừng.
- Gọi `mixer.music.stop()` để dừng phát bài hát hiện tại.

5.5.11 Hàm onsliderrelease()

```
def on_slider_release():
global tempstop
global auto
global count
tempstop = False
mixer.music.load(Playlist.get(ACTIVE))
music = mixer.Sound(Playlist.get(ACTIVE))
music_length_in_seconds = music.get_length()
period = (music_length_in_seconds/100)
auto = True
count = MusicProgess_slider.get()
mixer.music.play(0,periodcount)
```

Hàm này được gọi khi thanh trượt tiến trình bài hát (MusicProgessslider) bị nhả ra sau khi đã nhấn giữ. Cụ thể:

- Đặt biến toàn cục `tempstop` thành `False` để đánh dấu rằng bài hát không còn bị tạm dừng nữa.
- Tải bài hát đang được chọn vào mixer bằng `mixer.music.load(Playlist.get(ACTIVE))`.
- Tạo một đối tượng `mixer.Sound` từ bài hát đang được chọn.
- Lấy độ dài của bài hát (tính bằng giây) bằng `music.getlength()`.
- Tính khoảng thời gian tương ứng với 1
- Đặt biến toàn cục `auto` thành `True` để bật chế độ phát tự động.
- Lấy vị trí hiện tại của thanh trượt bằng `MusicProgessslider.get()`.
- Phát bài hát từ vị trí đã chọn bằng `mixer.music.play(0, periodcount)`.

5.5.12 Hàm updateprogess()

```
def update_progess():
global allstop
global auto
global tempstop
allstop = False
global Song_selected
if(auto == True):
global Song_selected
while(Song_selected == False):
pygame.time.wait(100)
if(Song_selected):
if(mixer.music.get_busy()):
music = mixer.Sound(Playlist.get(ACTIVE))
music_length_in_seconds = music.get_length()
period = (music_length_in_seconds/100)*1000
while(1):
if allstop:
MusicProgess_slider.set(0)
if not tempstop:
pygame.time.wait(int(period))
MusicProgess_slider.set(count)
count += 1
```

Hàm này cập nhật thanh trượt tiến trình bài hát (MusicProgessslider) trong khi bài hát đang phát. Cụ thể:

- Đặt biến toàn cục `allstop` thành `False` để cho phép phát nhạc.
- Nếu biến `auto` đúng (chế độ phát tự động):
  - Chờ cho đến khi có bài hát được chọn (biến `Songselected` đúng).
  - Nếu có bài hát đang phát:
    - Tạo một đối tượng `mixer.Sound` từ bài hát đang phát.
    - Lấy độ dài của bài hát (tính bằng giây) bằng `music.getlength()`.
    - Tính khoảng thời gian tương ứng với 1
    - Chạy vòng lặp vô tận:

- Nếu biến `allstop` đúng, đặt thanh trượt về 0.
- Nếu bài hát không bị tạm dừng:
- Đợi trong khoảng thời gian tương ứng với 1
- Cập nhật vị trí của thanh trượt bằng `MusicProgessslider.set(count)`.
- Tăng biến đếm `count`.