Documentación del Proyecto Playlist

Marc Aliaga Borras

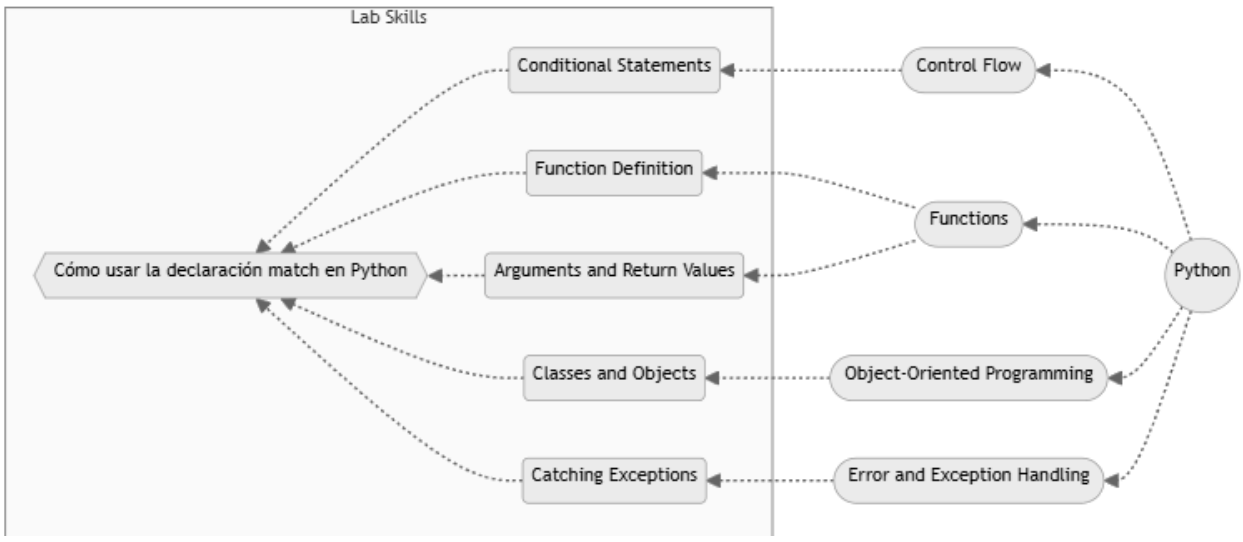
1. Introducción

En este proyecto se ha desarrollado una aplicación en Python para gestionar una lista de reproducción de música, donde se utilizan archivos XML para almacenar información sobre álbumes, artistas, canciones y géneros musicales. El código permite interactuar con los datos mediante un menú interactivo que ofrece diversas opciones para listar, buscar y filtrar los elementos en las listas.

2. Investigación

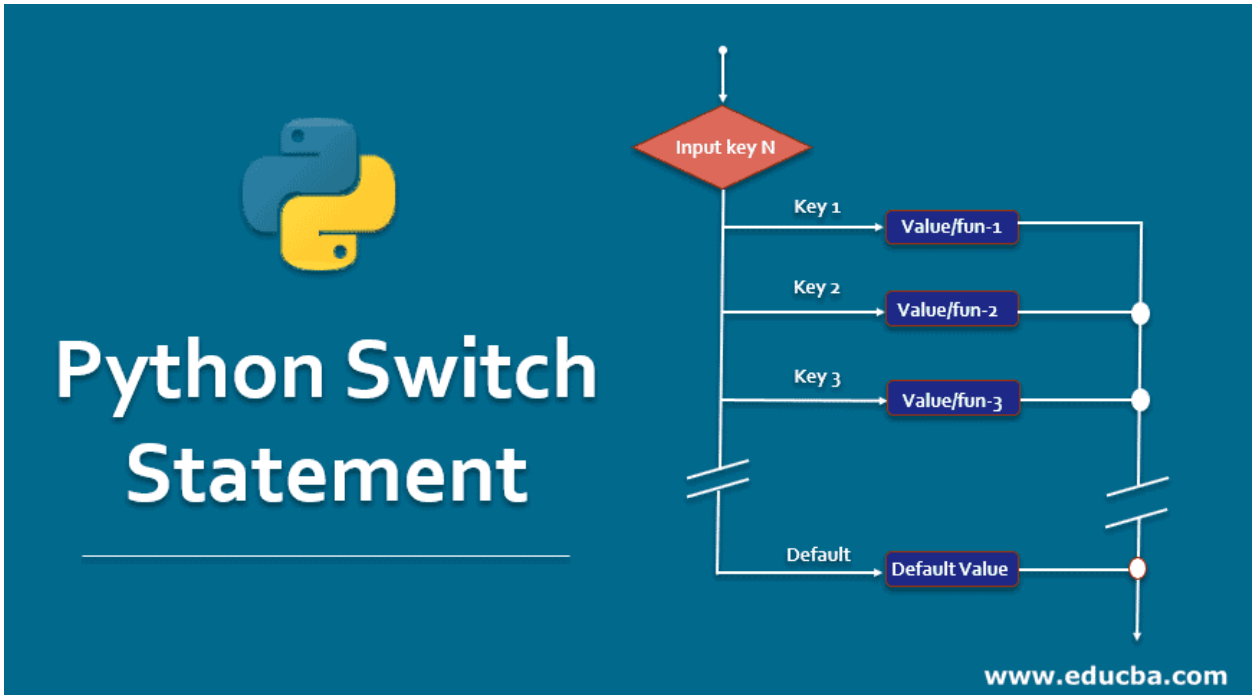
2.1. ¿Qué es y cómo funciona match en Python?

La declaración match de Python, introducida en Python 3.10, revoluciona la coincidencia de patrones y ofrece a los desarrolladores una forma poderosa y expresiva de manejar la lógica condicional compleja., similar a las estructuras switch en otros lenguajes de programación. El bloque match-case permite hacer coincidencias de patrones de manera más eficiente y clara.



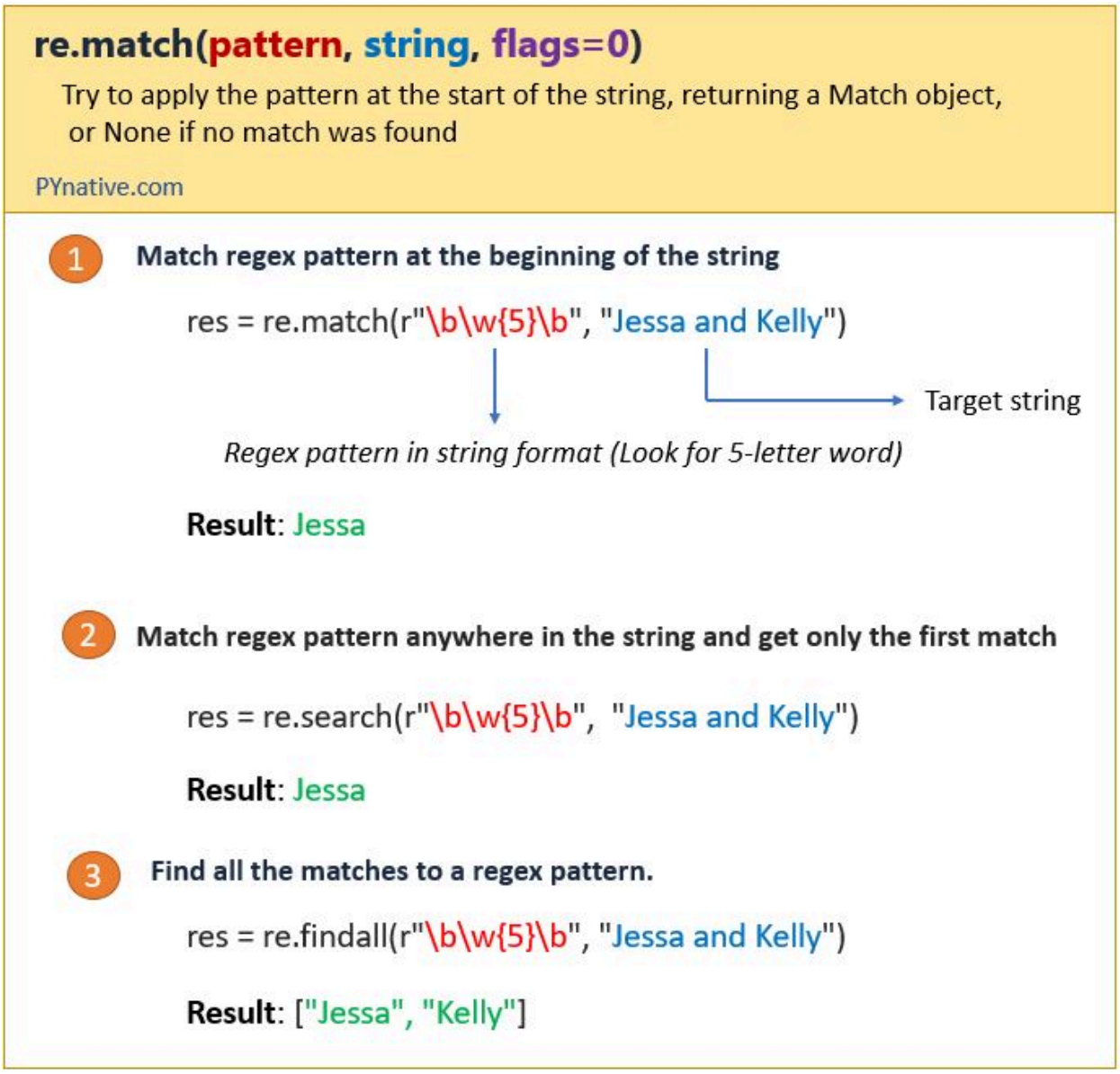
Característica	Descripción

Coincidencia de patrones	Permite la coincidencia compleja contra diferentes patrones
Verificación de tipos	Puede coincidir con tipos y estructuras específicos
Patrón comodín	Utiliza _ para coincidir con cualquier valor
Coincidencia condicional	Admite condiciones adicionales con guardas if



2.2. ¿Cómo utilizar el módulo re para encontrar coincidencias en una cadena de texto?

El módulo re se utiliza para trabajar con expresiones regulares en Python, lo que permite buscar patrones dentro de cadenas de texto. Uno de los métodos más comunes es re.search(), que permite verificar si un patrón existe dentro de una cadena, devolviendo un objeto de coincidencia si se encuentra o None si no se encuentra.



Ejemplo de re.search():

—----------

```
import re

text = "Scary Monsters and Nice Sprites"
pattern = "monsters"

if re.search(pattern, text, re.IGNORECASE):
    print("¡Coincidencia encontrada!")
else:
    print("No se encontró ninguna coincidencia.")
```

En este caso, re.search() busca el patrón "monsters" dentro de la cadena "Scary Monsters and Nice Sprites", ignorando las mayúsculas o minúsculas debido al argumento re.IGNORECASE.

---

3. Explicación del código del proyecto

3.1. Estructura general del código

El código está estructurado en varias funciones principales que gestionan las opciones del menú y cargan los datos desde los archivos XML. La función main() es la encargada de mostrar el menú principal y gestionar las selecciones del usuario.

3.2. Función load_data()

La función load_data() se encarga de cargar los datos desde los archivos XML correspondientes a los álbumes, artistas, canciones y géneros. Utiliza la librería xml.etree.ElementTree para analizar los archivos XML y extraer la información de cada sección.

```
def load_data():
    # Cargar los archivos XML
    albums_tree = ET.parse('albums.xml')
    artists_tree = ET.parse('artists.xml')
    songs_tree = ET.parse('songs.xml')
    genres_tree = ET.parse('genres.xml')

    albums = albums_tree.getroot()
    artists = artists_tree.getroot()
    songs = songs_tree.getroot()
    genres = genres_tree.getroot()

    return albums, artists, songs, genres
```

3.3. Función show_menu()

La función show_menu() es la que presenta al usuario las opciones disponibles en el menú principal. Permite navegar entre las diferentes secciones (álbumes, artistas, canciones, géneros) y salir de la aplicación. Si el usuario ingresa una opción no válida, el programa muestra un mensaje de error.

```
def show_menu():
    print("\nMenú Principal:")
    print("1. Álbumes")
    print("2. Artistas")
    print("3. Canciones")
    print("4. Géneros")
    print("0. Salir")
```

```
if choice == '0':
    print("Saliendo del programa.")
    return None
elif choice == '1':
    list_albums()
elif choice == '2':
    list_artists()
elif choice == '3':
    list_songs()
elif choice == '4':
    list_genres()
else:
    print("Opción no válida. Intente de nuevo.")
```

3.4. Funciones de listado: list_albums(), list_artists(), list_songs(), list_genres()

Cada una de estas funciones se encarga de listar los elementos de cada categoría. Por ejemplo, la función list_albums() muestra todos los álbumes cargados desde el archivo XML correspondiente.

```
def list_albums(albums):
    # Aquí listamos todos los álbumes
    print("\nÁlbumes disponibles:")
    for album in albums.findall('album'):
        title = album.find('title').text
        artist = album.find('artist').text
        year = album.find('year').text
        print(f"Título: {title}, Artista: {artist}, Año: {year}")
```

Estas funciones funcionan de manera similar para cada tipo de dato (artistas, canciones, géneros).

3.5. Función show_menu_songs()

La función show_menu_songs() es un submenú que se presenta cuando el usuario selecciona la opción de canciones. Ofrece opciones para listar todas las canciones o buscar una canción por título.

—----------

```python
def show_menu_songs():
    print("\nMenú de Canciones:")
    print("1. Listar todas las canciones")
    print("2. Buscar canción por título")
    print("0. Volver")
```