Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas.

MANUAL TÉCNICO

PRACTICA 01
LENGUAJES FORMALES Y DE PROGRAMACIÓN

202000558
Sheila Elizabeth Amaya Rodríguez

Descripción de la practica

Este programa fue desarrollado con el lenguaje de programación Python, el objetivo principal de este programa fue manejar información de los datos, enviados a través de un texto plano el cual contiene diferentes listas y valores separados por punto y coma , estos valores contienen información como nombre de una película, actores que participan en ella ,año de estreno y género.

En este programa únicamente se utilizó la librería graphviz para realizar la gráfica luego de cargar el contenido y os el cual nos ayuda a limpiar la pantalla del sistema.

¿Qué paradigma se utilizó?

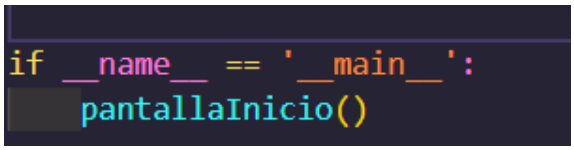
Se utilizo el paradigma imperativo o de procedimientos , este es un método que nos permite desatollar programas a través de procedimientos y además se utilizó POO (Programación Orientada a Objetos), este es un modelo o estilo de programación que proporciona unas guías acerca de cómo trabajar con él y que este está basado en el concepto de clases y objetos.

Lógica del programa

Se utilizaron varios módulos para realizar el funcionamiento del programa de estos se presenta a continuación una breve descripción de las clases, métodos y funciones que fueron implementadas para la realización del programa.

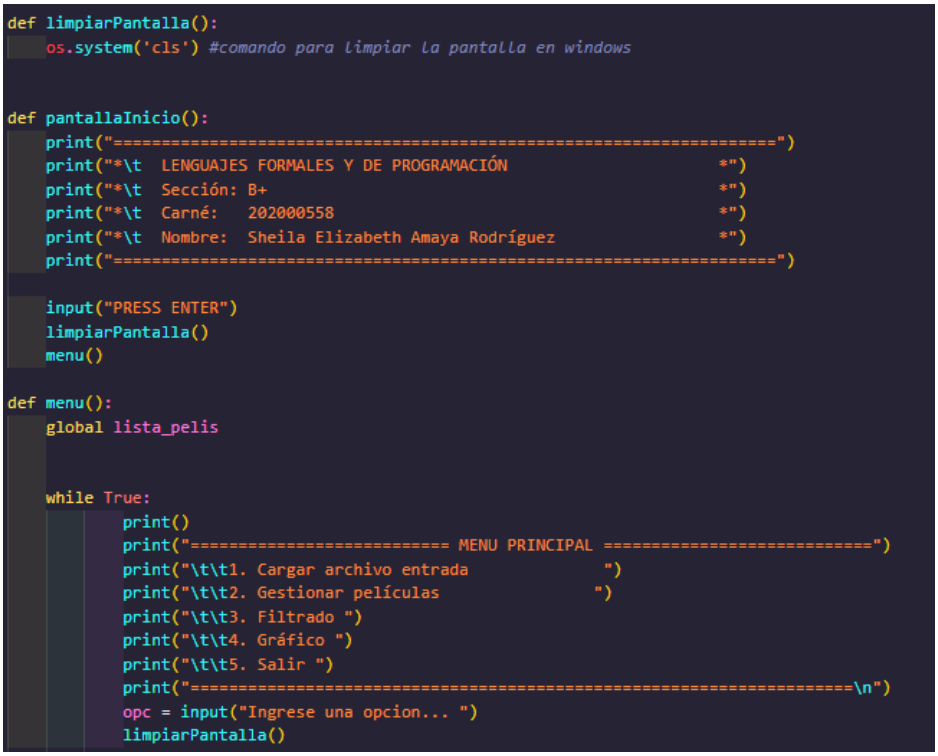
MAIN

Estas líneas definen un bloque de ejecución condicional en Python que permite verificar si el archivo actual se está ejecutando como programa principal o si ha sido importado como modulo por otro archivo.



```
if __name__ == '__main__':
    pantallaInicio()
```

En este módulo además se crea un método de tipo menú que representa una interfaz en consola conformada por un mensaje de inicio y un menú principal con varios submenús que permite que el usuario cargue archivos y maneje la información del archivo según lo que especifique cada opción de submenú.



```
def limpiarPantalla():
    os.system('cls') #comando para limpiar la pantalla en windows


def pantallaInicio():
    print("=========================================================================")
    print("*\t  LENGUAJES FORMALES Y DE PROGRAMACIÓN                          *")
    print("*\t  Sección: B+                                                   *")
    print("*\t  Carné:   202000558                                            *")
    print("*\t  Nombre:  Sheila Elizabeth Amaya Rodríguez                     *")
    print("=========================================================================")

    input("PRESS ENTER")
    limpiarPantalla()
    menu()

def menu():
    global lista_pelis


    while True:
            print()
            print("========================== MENU PRINCIPAL ===========================")
            print("\t\t1. Cargar archivo entrada             ")
            print("\t\t2. Gestionar películas               ")
            print("\t\t3. Filtrado ")
            print("\t\t4. Gráfico ")
            print("\t\t5. Salir ")
            print("=========================================================================\n")
            opc = input("Ingrese una opcion... ")
            limpiarPantalla()
```

CLASE LECTURA

Esta clase se utiliza para leer un archivo de texto que contiene información de películas y procesar los datos leídos en una lista simple enlazada de objetos “Película”.

- **cargarArchivo()**
  Este método toma como argumento la ruta del archivo que se desea leer, este método se encarga de leer todas las líneas del archivo utilizando el método readlines() y luego se llama al método procesar archivo, utiliza además un bloque de excepciones para manejar estas y cerrar el archivo después de leerlo.

```
def  cargaArchivo(self,ruta):
    try:
        with open(ruta, "r") as Archivo:
            lista_datos = Archivo.readlines()
            return self.procesar_Archivo(lista_datos)


    except FileNotFoundError:
        print("No se pudo encontrar el archivo Por favor, comprueba que el nombre y la ubicación son correctos.")
    except Exception as e:
        print("Ha ocurrido un error al cargar el archivo.")
    finally:
        if Archivo:
            Archivo.close()
```

- **procesar_Archivo()**
  Este método toma una lista de líneas que se paso como argumento y crea una lista enlazada de objetos tipo Película a partir de los datos procesados.

```
def procesar_Archivo(self,lista_datos):
    pelis = Lista_enlazada()

    for linea in lista_datos:
        lista_general = linea.strip().split(";")

        nombre = lista_general[0]
        actor= [a.strip() for a in lista_general[1].split(",")]
        anio = lista_general[2].strip()
        genero=lista_general[3].strip()

        nueva_peli = Pelicula(nombre,actor,anio,genero)

        pelis.insertar(nueva_peli)

    return pelis
```

CLASE LISTA_ENLAZADA

Esta se encarga de crear una lista que contenga objetos tipo Película, cada objeto contiene información sobre una película.

- **Insertar()**
  Este método se utiliza para insertar películas en la lista enlazada, si la película ya se encuentra en la lista se actualiza su información.

```
def insertar(self,pelicula):
    nuevo_nodo = Nodo(pelicula)
    if self.primero is None:
        self.primero = nuevo_nodo
        return
    #si la lista no esta vacia
    actual = self.primero
    while actual.siguiente:
        if actual.dato.nombre == pelicula.nombre:
            print("\nLa pelicula ",pelicula.nombre, "ya se encuentra en la lista.")
            actual.dato = pelicula
            return
        actual = actual.siguiente

    # Compara el nombre de la nueva película con el nombre de la última película en la lista
    if actual.dato.nombre == pelicula.nombre:
        print("\nLa película", pelicula.nombre, "ya está en la lista.")
        actual.dato = pelicula
        return

    actual.siguiente = nuevo_nodo
    nuevo_nodo.siguiente = None
```

- **mostrarPelicula()**
  Se encarga de mostrar en pantalla el nombre de todas las películas que se encuentran en la lista enlazada.

```
def mostrarPelicula(self):
    actual = self.primero
    i=1
    print()
    while actual:
        print(str(i)+ '. ',actual.dato.nombre)
        i+=1
        actual = actual.siguiente
    print()
```

- **mostrasP_A()**
  Este método muestra la lista de películas y luego de seleccionar alguna muestra los actores que participan en ella.

```
def mostrarP_A(self):#no se usa
    actual = self.primero #muestra la lita de peliculas
    i=1
    while actual:
            print(str(i)+". ",actual.dato.nombre,)
            actual = actual.siguiente
            i +=1
    num_peli = input("\n\tIngrese el número de la pelicula seleccionanda para ver sus actores: ")
    actual = self.primero

    i=1

    while actual and i<int(num_peli):
        actual = actual.siguiente
        i += 1
    if actual:
        for actor in actual.dato.actores: #
            print("\t-",actor)
        return
    else:
        print("\nNo se encontro la pelicula \n")
```

- **mostrarActores()**
  este método muestra una lista de actores únicos de todas las películas en la lista enlazada de películas.

```
def mostrarActores(self):
    actual = self.primero
    i = 1
    actores_ = []
    while actual:

        for actor in actual.dato.actores:
            if actor not in actores_:
                actores_.append(actor)
        actual = actual.siguiente
        i += 1

    for actor in actores_:
        print("\t-"+actor)
```

- **buscar_actor()**
  Este método se utiliza para mostrar una lista de todos los actores que aparecen en una película seleccionada.

```
def buscar_actor(self, nombre_actor):
    actual = self.primero
    peliculas_encontradas = []

    while actual:
        if nombre_actor in actual.dato.actores:
            peliculas_encontradas.append(actual.dato.nombre)
        actual = actual.siguiente

    if len(peliculas_encontradas)>0:
            print("\nPeliculas en las que participe el actor ",nombre_actor)
            for pelicula in peliculas_encontradas:
                print("- ", pelicula)
    else:
        print("\tNo se encontraron peliculas en las que participo el actor, ",nombre_actor)
```

- **buscar_anio()**
  Este método se utiliza para buscar todas las películas en la lista enlazada que se lanzaron en un año determinado.

```
def buscar_anio(self, entrada):
    actual = self.primero
    anio_peli = []

    while actual:
        if entrada == actual.dato.anio:
            anio_peli.append((actual.dato.nombre,actual.dato.genero))
        actual = actual.siguiente

    if anio_peli:
        #print("Peliculas del año: ", entrada)
        for anio in anio_peli:
            #print(anio)
            a = anio[0]
            genero = anio[1]
            print("\tPelicula: ",a, ", genero: ",genero)
    else:
        print("El año,",entrada," no se encuentra en el sistema.")
```

- **buscar_genero()**
  Este método se utiliza para buscar todas las películas en la lista enlazada que pertenecen a un género determinado.

```
def buscar_genero(self,entrada):
    actual = self.primero
    genero_peli = []

    while actual:
        if entrada == actual.dato.genero:
            genero_peli.append(actual.dato.nombre)
        actual = actual.siguiente

    if genero_peli:
        for genero in genero_peli:
            print("\t-",genero)
    else:
        print("No se encontro el genero",entrada)
```

- **Graph()**
  Este método crea un grafo de la lista enlazada, creando un nodo para cada película con la etiqueta nombre, género y año después recorre la lista de nuevo y crea las conexiones entre cada película y sus actores.

```
def graph(self):
    dot = Digraph(filename='peliculas.gv', node_attr={'shape': 'rectangle'}) # Crear una instancia de Digrap
    # recorremos la lista y creamos los nodos
    actual = self.primero
    while actual:
        etiqueta_nodo = "{}\n[{},{}]".format(actual.dato.nombre, actual.dato.genero, actual.dato.anio)
        dot.node(actual.dato.nombre, label=etiqueta_nodo)
        actual = actual.siguiente
    # recorremos la lista de nuevo y creamos las conexiones
    actual = self.primero
    while actual:
        for actor in actual.dato.actores:
            dot.edge(actual.dato.nombre,actor)
        actual = actual.siguiente
    # mostramos el grafo
    print("Grafico generado con exito..")
    dot.render("../Grafico/peliculas",format="pdf")
```

CLASE NODO

Es una estructura básica que se utiliza en muchas implementaciones de listas enlazadas, árboles y otros tipos.

```python
class Nodo:
    def __init__(self,dato):
        self.dato = dato
        self.siguiente = None
```

CLASE PELICULA

Es una clase orientada a objetos que se puede utilizar para representar una película.

```python
class Pelicula:
    def __init__(self,nombre,actores,anio,genero):
        self.nombre = nombre
        self.actores = actores
        self.anio = anio
        self.genero = genero
```