MANUAL TÉCNICO

[PRACTICA 2]

DAVED ABSHALON EJCALON CHONAY - 202105668

INGENIERÍA EN CIENCIAS Y SISTEMAS

LENGUAJES FORMALES Y DE POGRAMACIÓN, B+

REQUISITOS

- Sistema operativo
  Windows 10

- Procesador
  Intel Core 2 Duo E8200, AMD Phenom II X6 1075 a 3,0 GHz

- Memoria
  1 GB de RAM

- Almacenamiento
  1 GB de espacio disponible

- Tarjeta gráfica
  NVIDIA GT 630 / AMD HD 7750

- Internet
  Acceso a internet, con cualquier explorador web

- Visual Studio
  Con python instalado

DESARROLLO DEL PROGRAMA

- Carpeta principal

Archivos de código fuente: Aquí se encuentran todos los archivos Python que componen el programa.

Documentación: Carpeta opcional que contiene documentos como este manual técnico.

Paradigma de programación utilizado

Principalmente programación orientada a objetos (POO) donde los conceptos claves del programa son las clases y objetos.

Clases y objetos: Este programa está organizado por clases y objetos en los cuales están encapsulados el conjunto de datos y funciones relacionadas que representan el código unificado del programa. (Carga, Gestion, Filtrado, Grafica y la clase Main).

Encapsulación: Cada clase tiene su propia funcionalidad en la cual todas funcionan en conjunto y cada detalle interno se necesita uno del otro.

Abstracción: Todas las clases proporcionan una abstracción al mundo real como lo son las películas y actores, esto permite una mayor compresión por parte del programador.

CLASES

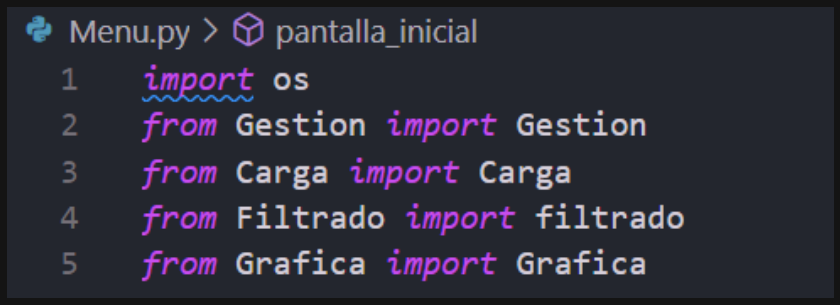
- Meny.py

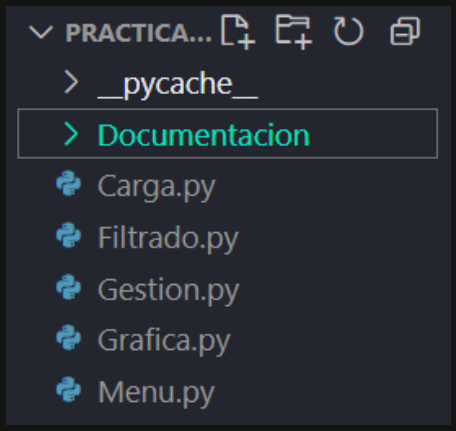
Aquí principalmente se encuentra el código para la gestión de todas la funcionalidad del programa, este script utiliza clases y funciones para la organización lógica del programa.

- **Importaciones de módulos**:
  - El código importa los módulos necesarios para el funcionamiento del programa, que incluyen:
  - importacion os: este módulo proporciona las funciones que permiten intereactuar con el sistema operativo.
  - Gestion.py: aca se hubica todo el codigo necesario para la gestion de las peliculas incluye las funciones de "mostrar peliculas y mostrar actores" posteriormente se da una explicacion sobre el funcionamiento y como se realizo la logica de este apartado.
  - Carga.py: esta el codigo necesario para la funcion de carga de peliculas, cabe resaltar que es necesaria la sintaxis adecuada.
  - Filtrado.py: se hubica las funciones de filtrados, por nombre de actor, año o genero de pelicula.

- **Resumen**:
  - En resumen, este script Python crea un programa de línea de comandos simple que permite al usuario cargar datos sobre películas desde un archivo, gestionar esas películas, realizar filtrados y generar un gráfico de relaciones entre actores y películas. Utiliza clases y funciones para organizar la lógica del programa de manera modular y legible.

    Este aparato es la raíz principal del programa, aquí se encuentra todo el esqueleto que hace posible el funcionamiento correcto de nuestro programa de gestión de películas, continuación explicaré las subclases.





- # Carga.py
  - **Resumen**:
    - Esta clase “Carga.py” proporciona funcionalidades para cargar datos de películas desde un archivo de texto y manipularlos en forma de diccionarios en una lista.
  - **Método __init__**
    - Define el método constructor que inicializa la lista de películas como una lista vacía.
  - **Método pelicula_tiene_actor**
    - Este método toma el nombre de una película y un actor como entrada y verifica si el actor aparece en el elenco de la película especificada en la lista de películas cargadas. Retorna True si el actor está en la película, de lo contrario retorna False.

```python
Carga.py > Carga > leer_archivo_peliculas
class Carga:
    def __init__(self):
        self.lista_peliculas = []

    def pelicula_tiene_actor(self, nombre_película, actor):
        for película in self.lista_peliculas:
            if película['nombre'] == nombre_película:
                return actor in película['actores']
        return False
```

  - **Métodos get_nombre_peliculas y get_nombre_actores**
    - Estos métodos devuelven una lista de nombres de películas y una lista de nombres de actores respectivamente, extrayendo la información de la lista de películas cargadas.

```python
    def get_nombre_peliculas(self):
        nombres = []
        for película in self.lista_peliculas:
            nombres.append(película['nombre'])
        return nombres

    def get_nombre_actores(self):
        nombres = []
        for película in self.lista_peliculas:
            nombres.extend(película['actores'])
        return list(set(nombres))
```

- **Método creacion_diccionario_pelicula**

  - Crea y devuelve un diccionario que representa una película, con las claves "nombre", "actores", "año", y "género".

```python
def creacion_diccionario_pelicula(self, nombre, actores, año, género):
    película = {
        "nombre": nombre,
        "actores": actores.split(","),
        "año": int(año),
        "género": género
    }
    return película
```

- **Método leer_archivo_peliculas**

  - Toma la ruta de un archivo como entrada y trata de abrirlo para leerlo.
  - Retorna una lista de líneas leídas del archivo si tiene éxito, o una lista vacía si el archivo no se encuentra.

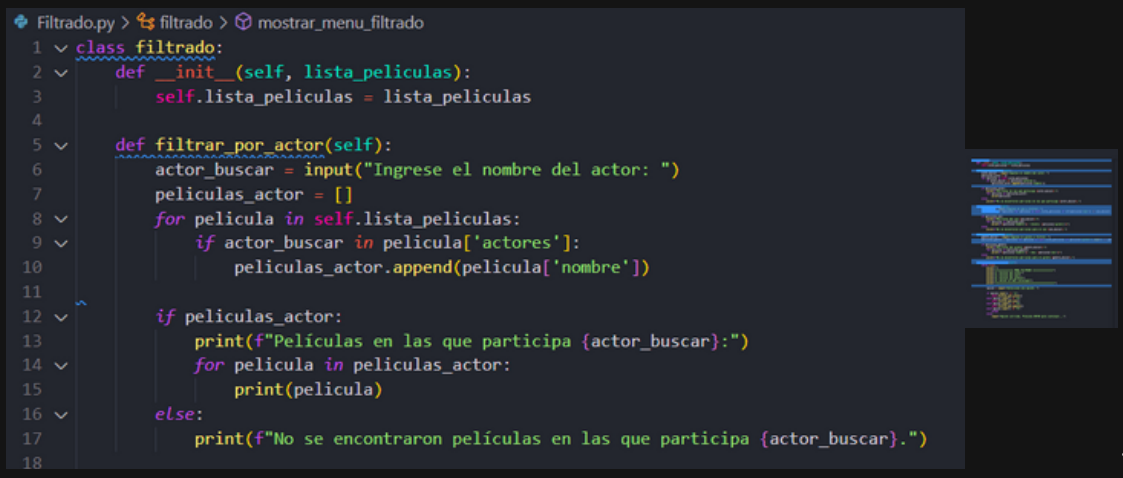
```python
    def leer_archivo_peliculas(self, ruta):

        try:
            with open(ruta, 'r', encoding='utf8') as lectura_lfp:
                lines = lectura_lfp.readlines()
            return lines
        except FileNotFoundError:
            print("El archivo no fue encontrado.")
```

- **Método agregar_peliculas**

  - Solicita al usuario que ingrese la ruta del archivo de películas.
  - Lee cada línea del archivo, la divide en partes separadas por ;, y verifica si el formato es correcto.
  - Crea un diccionario de película utilizando el método creacion_diccionario_pelicula y lo agrega a la lista de películas.

```python
    def agregar_peliculas(self):

        ruta_archivo = input("Ingresa la ruta del archivo de películas: ")
        lines = self.leer_archivo_peliculas(ruta_archivo)

        for linea in lines:
            datos = linea.strip().split(';')

            if len(datos) != 4:
                print(f"La línea '{linea}' no tiene el formato correcto.")
                continue

            nombre_pelicula = datos[0]
            actores = datos[1]
            año = datos[2]
            género = datos[3]

            # Verificar si existe una película con el mismo nombre
            if nombre_pelicula in self.get_nombre_peliculas():
                print(f"La película '{nombre_pelicula}' ya fue cargada.")

                # Remover la película con el mismo nombre
                self.lista_peliculas = [p for p in self.lista_peliculas if p['nombre'] != nombre_pelicula]

            película = self.creacion_diccionario_pelicula(nombre_pelicula, actores, año, género)
            self.lista_peliculas.append(película)

        print(f"Se han cargado {len(self.lista_peliculas)} películas.")
```

- Filtrado.py
  - **Resumen**:
    - Proporciona una interfaz de usuario para filtrar la lista de películas según diferentes criterios, como actor, año y género
  - **Método de filtrado**
    - filtrar_por_actor: Solicita al usuario que ingrese el nombre de un actor. Luego, itera sobre la lista de películas y verifica si el actor está en la lista de actores de cada película. Si encuentra películas en las que participa el actor, las imprime; de lo contrario, muestra un mensaje indicando que no se encontraron películas con ese actor.
    - filtrar_por_año: Solicita al usuario que ingrese un año. Luego, utiliza una comprensión de lista para filtrar las películas cuyo año coincida con el año ingresado. Imprime las películas encontradas o un mensaje si no se encuentran películas para ese año.
    - filtrar_por_genero: Solicita al usuario que ingrese un género. Luego, utiliza una comprensión de lista para filtrar las películas cuyo género coincida con el género ingresado (ignorando mayúsculas/minúsculas). Imprime las películas encontradas o un mensaje si no se encuentran películas para ese género.
  - **Método mostrar_menu_filtrado**
    - Muestra un menú para que el usuario seleccione el tipo de filtrado que desea realizar.
    - Utiliza un bucle while para mantener el menú en pantalla hasta que el usuario elija la opción para volver al menú principal.
    - Llama a los métodos de filtrado correspondientes según la opción seleccionada por el usuario.



- Gestion.py
  - **Resumen**:
    - Permite gestionar y visualizar la lista de peliculas cargadas.
  - **Método num_peliculas**
    - Retorna el número total de películas en la lista de películas.
  - **Método mostrar_menu_gestion_peliculas**
    - Muestra un menú de gestión de películas que permite al usuario seleccionar diferentes opciones.
    - Utiliza un bucle while para mantener el menú en pantalla hasta que el usuario elija la opción para volver al menú principal.
    - Llama a los métodos mostrar_peliculas o mostrar_actores según la opción seleccionada por el usuario.
  - **Método mostrar_peliculas:**
    - Imprime la lista de películas cargadas en un formato específico que incluye el nombre, año y género de cada película.
  - **Método mostrar_actores:**
    - Muestra las películas cargadas y solicita al usuario que seleccione una película ingresando su número.
    - Si el usuario ingresa un número válido, muestra los actores de la película seleccionada.
    - Si el usuario ingresa 0, regresa al menú anterior.
    - Si el usuario ingresa un número inválido, muestra un mensaje de error.

```
def mostrar_peliculas(self):
    print("")
    print("======================  LISTA DE PELÍCULAS ====================")
    for i, pelicula in enumerate(self.lista_peliculas, 1):
        print(f"{i}. {pelicula['nombre']} ({pelicula['año']}) - Género: {pelicula['género']}")
    print("============================================================")

def mostrar_actores(self):
    while True:
        self.mostrar_peliculas()
        seleccion = input("Selecciona el número de la película para ver los actores (0 para volver): ")
        if seleccion.isdigit():
            seleccion = int(seleccion)
            if seleccion == 0: # Volver
                break
            if seleccion <= len(self.lista_peliculas):
                pelicula = self.lista_peliculas[seleccion - 1]
                print("")
                print(f"Actores de '{pelicula['nombre']}':\n{', '.join(pelicula['actores'])}")
            else: # Número fuera de rango
                print("Número de película inválido. Inténtalo de nuevo.")
        else: # No es un número
            print("Por favor, ingresa un número válido.")
```

- Grafica.py
  - **Resumen**:
    - Imprime una imagen .jpg que permite ver las relaciones entre actores y películas en forma de un diagrama gráfico generado automáticamente.
  - **Método generar_grafica:**
    - Crea un objeto Digraph del módulo graphviz.
    - Aqui se generan los nodos del grafico para cada actor utilizando su nombre como identificador, asi mismo genera los nodos de la pelicula en forma de tabla.