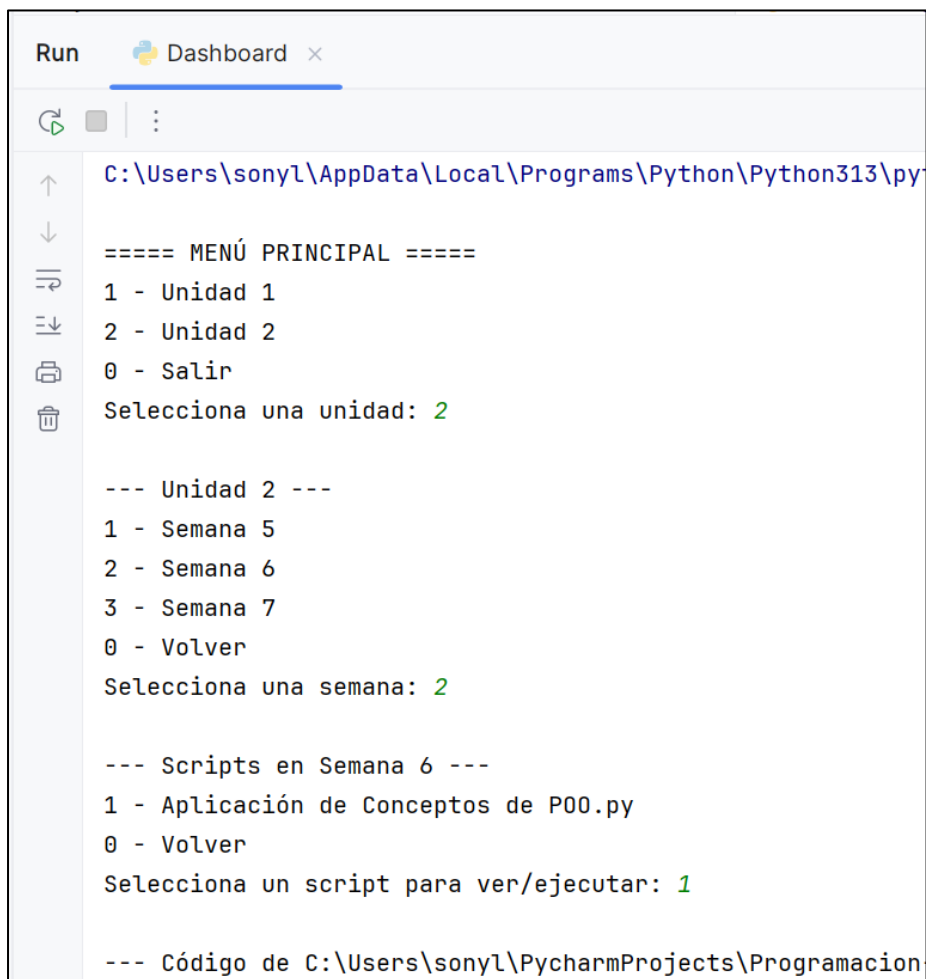


Implementación de un Dashboard

Capturas de pantalla de la ejecución de un Dashboard aplicado a el proyecto de programación orientado a objetos, el cual está dividido en unidades y estas a su vez contienen las semanas con los scripts de las tareas.

Figura 1

Ejecución del menú principal.



```
Run Dashboard x
C:\Users\sonyl\AppData\Local\Programs\Python\Python313\py
===== MENÚ PRINCIPAL =====
1 - Unidad 1
2 - Unidad 2
0 - Salir
Selecciona una unidad: 2

--- Unidad 2 ---
1 - Semana 5
2 - Semana 6
3 - Semana 7
0 - Volver
Selecciona una semana: 2

--- Scripts en Semana 6 ---
1 - Aplicación de Conceptos de P00.py
0 - Volver
Selecciona un script para ver/ejecutar: 1

--- Código de C:\Users\sonyl\PycharmProjects\Programacion
```



Figura 2

Código del script - Aplicación de conceptos de POO.

```
Run Dashboard x
#Programa que realiza la compra de tickets para el cine
#-----
#Se demuestra la aplicación de conceptos de POO como:
# HERENCIA : AdultoTicket, NinoTicket y TerceraEdadTicket se derivan de Ticket.
# ENCAPSULACIÓN : El atributo __precio es privado; se expone con la propiedad precio.
# POLIMORFISMO : Cada subclase sobrescribe calcular_precio().
#
# Además, Carrito.agregar_ticket usa *tickets (argumentos variables) y acepta cualquier subclase de Ticket.

from __future__ import annotations

#Clase base para un ticket de cine
class Ticket:

    def __init__(self, pelicula: str, asiento: str, precio_base: float) -> None:
        self.pelicula = pelicula          # atributo público
        self.asiento = asiento             # atributo público
        self.__precio = precio_base        # atributo privado (encapsulado)

    # ----- Encapsulación -----
    @property #usado cada vez que el programa lee precio
    def precio(self) -> float:
        return self.__precio #Precio final del ticket (solo lectura pública).

    # Método protegido que permite a las subclases modificar el precio de forma segura.
    def _precio_fijo(self, valor: float) -> None:
        self.__precio = valor

    # ----- Polimorfismo (Sobrescritura) -----
    def calcular_precio(self) -> None:
        self._precio_fijo(self.precio) #En la clase base no hay cambio (precio base). Asegura que el precio se mantiene igual

    # ----- Utilidad -----
    def __str__(self) -> str:
        return f"{self.__class__.__name__} | '{self.pelicula}' | Asiento {self.asiento} | ${self.precio:.2f}"

#Ticket de adulto (sin descuento).
class AdultoTicket(Ticket):
    def calcular_precio(self) -> None: # sobrescribe
```

Figura 4

Ejecución del script - Aplicación de conceptos de POO.

```
Run Dashboard x
¿Deseas ejecutarlo? (1: Sí, 0: No): 1

----- RESUMEN DE COMPRA -----

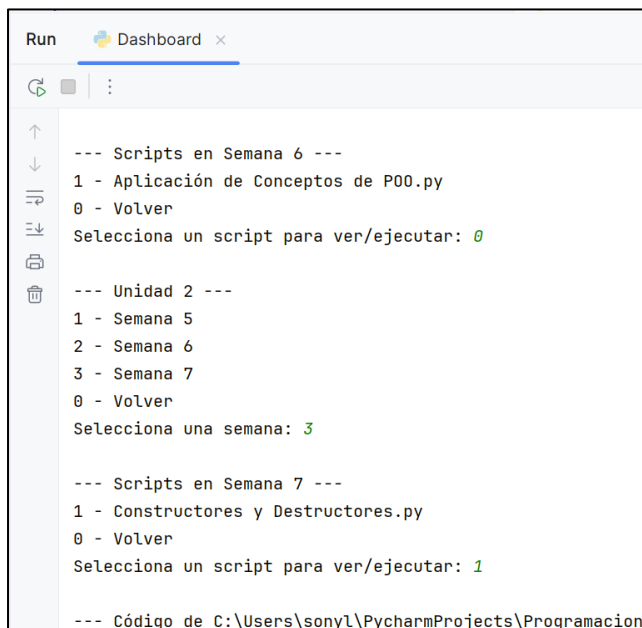
AdultoTicket | 'Superman' | Asiento B7 | $7.60
NinosTicket | 'Superman' | Asiento B8 | $3.80
TerceraEdadTicket | 'Superman' | Asiento B9 | $5.32

TOTAL A PAGAR: $16.72

Presiona Enter para continuar...
```

Figura 5

Selección de la semana 7.



```
Run Dashboard x
--- Scripts en Semana 6 ---
1 - Aplicación de Conceptos de P00.py
0 - Volver
Selecciona un script para ver/ejecutar: 0

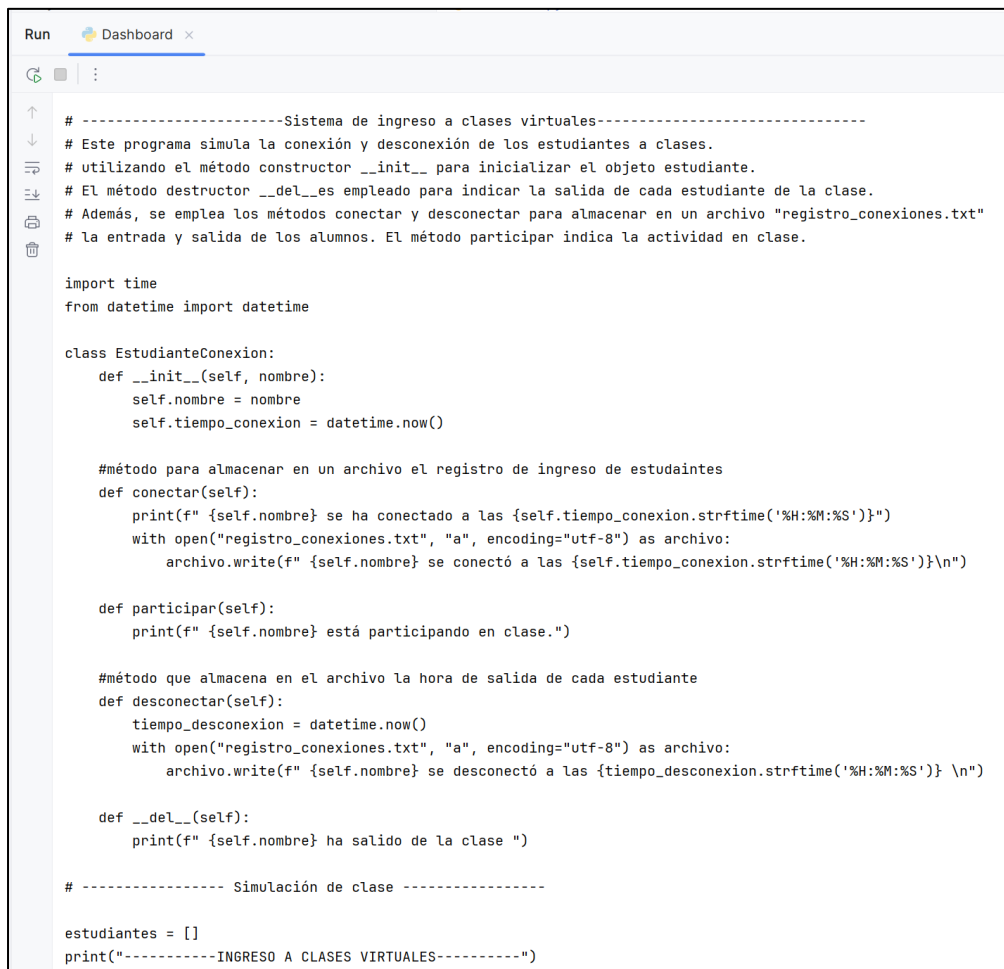
--- Unidad 2 ---
1 - Semana 5
2 - Semana 6
3 - Semana 7
0 - Volver
Selecciona una semana: 3

--- Scripts en Semana 7 ---
1 - Constructores y Destructores.py
0 - Volver
Selecciona un script para ver/ejecutar: 1

--- Código de C:\Users\sonyl\PycharmProjects\Programacion
```

Figura 6

Código del script Constructores y Destructores.



```
Run Dashboard x
# -----Sistema de ingreso a clases virtuales-----
# Este programa simula la conexión y desconexión de los estudiantes a clases.
# utilizando el método constructor __init__ para inicializar el objeto estudiante.
# El método destructor __del__ es empleado para indicar la salida de cada estudiante de la clase.
# Además, se emplea los métodos conectar y desconectar para almacenar en un archivo "registro_conexiones.txt"
# la entrada y salida de los alumnos. El método participar indica la actividad en clase.

import time
from datetime import datetime

class EstudianteConexion:
    def __init__(self, nombre):
        self.nombre = nombre
        self.tiempo_conexion = datetime.now()

    #método para almacenar en un archivo el registro de ingreso de estudaintes
    def conectar(self):
        print(f" {self.nombre} se ha conectado a las {self.tiempo_conexion.strftime('%H:%M:%S')}")
        with open("registro_conexiones.txt", "a", encoding="utf-8") as archivo:
            archivo.write(f" {self.nombre} se conectó a las {self.tiempo_conexion.strftime('%H:%M:%S')}\n")

    def participar(self):
        print(f" {self.nombre} está participando en clase.")

    #método que almacena en el archivo la hora de salida de cada estudiante
    def desconectar(self):
        tiempo_desconexion = datetime.now()
        with open("registro_conexiones.txt", "a", encoding="utf-8") as archivo:
            archivo.write(f" {self.nombre} se desconectó a las {tiempo_desconexion.strftime('%H:%M:%S')} \n")

    def __del__(self):
        print(f" {self.nombre} ha salido de la clase ")

# ----- Simulación de clase -----

estudiantes = []
print("-----INGRESO A CLASES VIRTUALES-----")
```



Figura 7

Ejecución del script Constructores y Destructores.

```
Run Dashboard x
¿Deseas ejecutarlo? (1: Sí, 0: No): 1
-----INGRESO A CLASES VIRTUALES-----
Ana Torres se ha conectado a las 16:22:59
Luis Pérez se ha conectado a las 16:23:01
Carla Gómez se ha conectado a las 16:23:03
-----
Ana Torres está participando en clase.
Luis Pérez está participando en clase.
Carla Gómez está participando en clase.
-----
Carla Gómez ha salido de la clase
Luis Pérez ha salido de la clase
Ana Torres ha salido de la clase

Presiona Enter para continuar...
```

Figura 8

Finalización del programa.

```
Run Dashboard x
--- Scripts en Semana 7 ---
1 - Constructores y Destructores.py
0 - Volver
Selecciona un script para ver/ejecutar: 0

--- Unidad 2 ---
1 - Semana 5
2 - Semana 6
3 - Semana 7
0 - Volver
Selecciona una semana: 0

===== MENÚ PRINCIPAL =====
1 - Unidad 1
2 - Unidad 2
0 - Salir
Selecciona una unidad: 0
Saliendo del programa.

Process finished with exit code 0
```