

# PROGRAMAS

MORA CABRERA DIEGO EDGAR

## **Sistema de Gestión de un Cine**

### **1. Justificación de la Necesidad del Programa**

El sistema desarrollado busca gestionar las funciones, reservas, empleados y productos de un cine. Permite a los usuarios realizar reservas de asientos para funciones, a los empleados administrar películas y promociones, y manejar la venta de productos en la zona de comida. Este sistema facilita la administración y optimización de los recursos del cine.

### **2. Clases Utilizadas**

#### **Clase Persona**

- **Atributos:**
  - nombre (str)
  - gmail (str)
- **Métodos:**
  - Constructor `__init__`: Inicializa nombre y correo.

#### **Clase Usuario (Hereda de Persona)**

- **Atributos:**
  - reservas (lista de reservas)
- **Métodos:**
  - `hacer_reserva(reserva)`: Agrega una reserva a la lista del usuario.

#### **Clase Empleado (Hereda de Persona)**

- **Atributos:**
  - rol (str)
- **Métodos:**
  - `agregar_funcion(cine, funcion)`: Agrega una función al cine.
  - `agregar_pelicula(cine, pelicula)`: Agrega una película al cine.
  - `agregar_promocion(cine, promocion)`: Agrega una promoción al cine.

#### **Clase Espacio (Superclase)**

- **Atributos:**
  - nombre (str)

### **Clase Sala (Hereda de Espacio)**

- **Atributos:**
  - tipo (str)
  - capacidad (int)
  - asientos\_ocupados (set)
- **Métodos:**
  - verificar\_disponibilidad(asientos): Verifica si los asientos están disponibles.
  - reservar\_asientos(asientos): Reserva los asientos si están disponibles.

### **Clase ZonaComida (Hereda de Espacio)**

- **Atributos:**
  - productos (dict)
- **Métodos:**
  - agregar\_producto(producto, precio, cantidad): Agrega un producto con su precio y cantidad.
  - comprar\_producto(producto, cantidad): Permite la compra de productos.

### **Clase Pelicula**

- **Atributos:**
  - titulo (str)
  - duracion (int)
  - clasificacion (str)
  - genero (str)

### **Clase Promocion**

- **Atributos:**
  - descripcion (str)
  - descuento (float)

### **Clase Funcion**

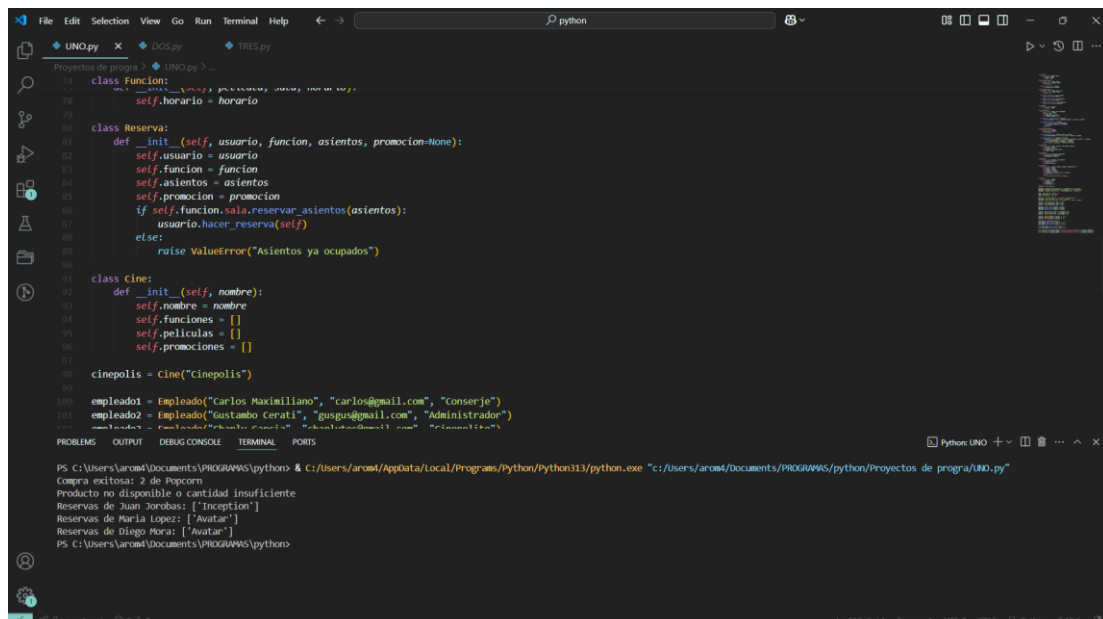
- **Atributos:**
  - pelicula (objeto Pelicula)
  - sala (objeto Sala)
  - horario (str)

## Clase Reserva

- **Atributos:**
  - usuario (objeto Usuario)
  - funcion (objeto Funcion)
  - asientos (lista de int)
  - promocion (objeto Promocion, opcional)
- **Métodos:**
  - Constructor: Verifica la disponibilidad de los asientos y registra la reserva.

## Clase Cine

- **Atributos:**
  - nombre (str)
  - funciones (lista de funciones)
  - peliculas (lista de películas)
  - promociones (lista de promociones)



```

class Funcion:
    def __init__(self, pelicula, horario):
        self.pelicula = pelicula
        self.horario = horario

class Reserva:
    def __init__(self, usuario, funcion, asientos, promocion=None):
        self.usuario = usuario
        self.funcion = funcion
        self.asientos = asientos
        self.promocion = promocion
        if self.funcion.sala.reservar_asientos(asientos):
            self.usuario.hacer_reserva(self)
        else:
            raise ValueError("Asientos ya ocupados")

class Cine:
    def __init__(self, nombre):
        self.nombre = nombre
        self.funciones = []
        self.peliculas = []
        self.promociones = []

cinopolis = Cine("Cinepolis")

emplead01 = Empleado("Carlos Maximiliano", "carlos@gmail.com", "Conserje")
emplead02 = Empleado("Gustavo Cerati", "gusgus@gmail.com", "Administrador")

# Test cases
PS C:\Users\arom\Documents\PROGRAMAS\python> python.exe "c:\Users\arom\Documents\PROGRAMAS\python\Proyectos de progra\UNO.py"
Compra exitosa: 2 de Popcorn
Producto no disponible o cantidad insuficiente
Reservas de Juan Jorobas: ['Inception']
Reservas de Maria Lopez: ['Avatar']
Reservas de Diego Rora: ['Avatar']
PS C:\Users\arom\Documents\PROGRAMAS\python>

```

## **Sistema de Gestión de una Cafetería**

### **1. Justificación del Programa**

Este sistema permite gestionar clientes, empleados, productos y pedidos en una cafetería. Permite registrar clientes, manejar inventario, realizar pedidos y aplicar promociones.

### **2. Clases Utilizadas**

#### **Clase Persona**

- **Atributos:**
  - nombre (str)

#### **Clase Cliente (Hereda de Persona)**

- **Atributos:**
  - historial\_pedidos (lista)
- **Métodos:**
  - realizar\_pedido(pedido): Registra un pedido en el historial.

#### **Clase Empleado (Hereda de Persona)**

- **Atributos:**
  - rol (str)

#### **Clase ProductoBase**

- **Atributos:**
  - nombre (str)
  - precio (float)

#### **Clase Bebida (Hereda de ProductoBase)**

- **Atributos:**
  - tamaño (str)
  - tipo (str)
  - opciones\_personalizadas (dict)

#### **Clase Postre (Hereda de ProductoBase)**

- **Atributos:**
  - es\_vegano (bool)

- sin\_gluten (bool)

## Clase Inventario

- **Atributos:**
  - ingredientes (dict)
- **Métodos:**
  - agregar\_ingredientes(nombre, cantidad): Añade ingredientes al inventario.
  - verificar\_disponibilidad(ingredientes\_requeridos): Verifica si hay ingredientes suficientes.
  - descontar\_ingredientes(ingredientes\_requeridos): Reduce ingredientes del inventario.

## Clase Pedido

- **Atributos:**
  - cliente (Cliente)
  - productos (lista de productos)
  - estado (str)
- **Métodos:**
  - calcular\_total(): Calcula el costo total del pedido.

## Clase Promocion

- **Atributos:**
  - descuento (float)
  - criterio (función)

```

101 bebida = Bebida("Café Americano", 3.0, "Grande", "Caliente", {"café": 1, "leche de almendra": 1})
102 bebida2 = Bebida("Té Chai", 4.0, "Mediano", "Caliente", {"leche de almendra": 1})
103 bebida3 = Bebida("Latte", 5.0, "Grande", "Caliente", {"café": 2, "leche de almendra": 1})
104 postre = Postre("Tiramisu", 2.5, False, False)
105
106 pedido1 = Pedido(cliente1)
107 pedido2 = Pedido(cliente2)
108 pedido3 = Pedido(cliente3)
109
110 if inventario.verificar_disponibilidad(bebida1.opciones_personalizadas):
111     inventario.descontar_ingredientes(bebida1.opciones_personalizadas)
112     pedido1.agregar_producto(bebida1)
113     cliente1.realizar_pedido(pedido1)
114     print(f"Pedido de {cliente1.nombre}: {pedido1.calcular_total()}$")
115 else:
116     print(f"No hay suficientes ingredientes para preparar el pedido de {cliente1.nombre}.")
117
118 if inventario.verificar_disponibilidad(bebida2.opciones_personalizadas):
119     inventario.descontar_ingredientes(bebida2.opciones_personalizadas)
120     pedido2.agregar_producto(bebida2)
121     cliente2.realizar_pedido(pedido2)
122     print(f"Pedido de {cliente2.nombre}: {pedido2.calcular_total()}$")
123 else:
124     print(f"No hay suficientes ingredientes para preparar el pedido de {cliente2.nombre}.")
125
126 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
127 PS C:\Users\arom\Documents\PROGRAMAS\python> & c:\Users\arom\AppData\Local\Programs\Python\Python113\python.exe "c:\Users\arom\Documents\PROGRAMAS\python\Proyectos de progra\005.py"
128 Pedido de Carlos: 3.0$
129 Pedido de Ana: 6.5$
130 No hay suficientes ingredientes para preparar el pedido de Luis.
131 PS C:\Users\arom\Documents\PROGRAMAS\python>
  
```

## **Sistema de Gestión de una Biblioteca**

### **1. Justificación del Programa**

Este sistema permite gestionar préstamos, devoluciones y catálogo de materiales en una biblioteca. Facilita la administración de libros, revistas y materiales digitales.

### **2. Clases Utilizadas**

#### **Clase Material**

- **Atributos:**
  - titulo (str)
  - estado (str)

#### **Clases Derivadas de Material**

- **Libro:**
  - autor (str)
  - género (str)
- **Revista:**
  - edición (str)
  - periodicidad (str)
- **Material Digital:**
  - tipo\_archivo (str)
  - enlace\_descarga (str)

#### **Clase Persona**

- **Atributos:**
  - nombre (str)

#### **Clase Usuario (Hereda de Persona)**

- **Atributos:**
  - materiales\_prestados (lista)
  - multa (float)

#### **Clase Bibliotecario (Hereda de Persona)**

- **Métodos:**

- gestionar\_prestamo(usuario, material, sucursal): Maneja préstamos.
- transferir\_material(material, sucursal\_origen, sucursal\_destino): Transfiere materiales.

## Clase Sucursal

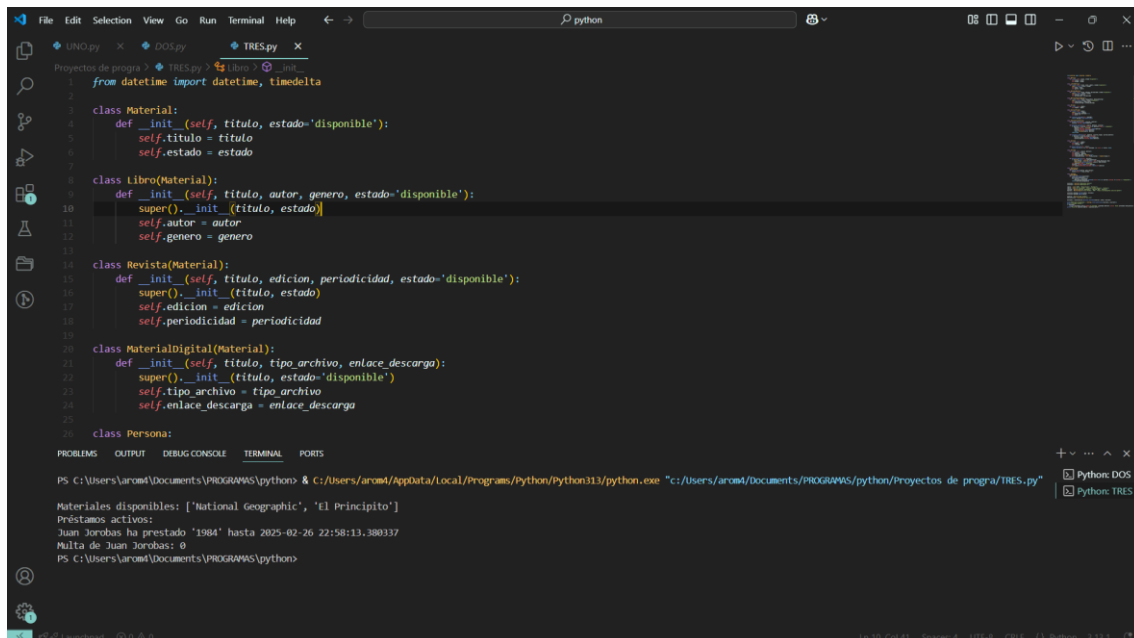
- **Atributos:**
  - nombre (str)
  - catalogo (lista de Materiales)

## Clase Prestamo

- **Atributos:**
  - usuario (Usuario)
  - material (Material)
  - fecha\_prestamo (datetime)
  - fecha\_devolucion (datetime)

## Clase Penalizacion

- **Métodos:**
  - aplicar\_multa(usuario, dias\_retraso): Aplica multas por retraso.



```

File Edit Selection View Go Run Terminal Help
python

Proyectos de progra > TRES.py > Libro > _init_
from datetime import datetime, timedelta

class Material:
    def __init__(self, titulo, estado='disponible'):
        self.titulo = titulo
        self.estado = estado

class Libro(Material):
    def __init__(self, titulo, autor, genero, estado='disponible'):
        super().__init__(titulo, estado)
        self.autor = autor
        self.genero = genero

class Revista(Material):
    def __init__(self, titulo, edicion, periodicidad, estado='disponible'):
        super().__init__(titulo, estado)
        self.edicion = edicion
        self.periodicidad = periodicidad

class MaterialDigital(Material):
    def __init__(self, titulo, tipo_archivo, enlace_descarga):
        super().__init__(titulo, estado='disponible')
        self.tipo_archivo = tipo_archivo
        self.enlace_descarga = enlace_descarga

class Persona:

PS C:\Users\arom4\Documents\PROGRAMAS\python> & C:\Users\arom4\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\arom4\Documents\PROGRAMAS\python\Proyectos de progra/TRES.py"
Python: DOS
Python: TRES

Materiales disponibles: ['National Geographic', 'El Principito']
Préstamos activos:
Juan Jorobas ha prestado '1984' hasta 2025-02-26 22:58:13.380337
Multa de Juan Jorobas: 0
PS C:\Users\arom4\Documents\PROGRAMAS\python>
  
```

<https://github.com/tekijimun/PRACTICA-U1>