

# Ejercicios de Python

**Semana 6**

28.07.2025 - 03.08.2025



## Caso General

### Gestión de un Cine



Un cine necesita gestionar su cartelera, los horarios de las funciones, la venta de entradas y la disponibilidad de asientos. Un sistema informático es esencial para que los clientes puedan consultar información, comprar entradas online y para que la administración pueda analizar qué películas son más exitosas.

# Nivel Fácil: Consultar Horarios de una Película

## Caso

Un cliente quiere saber a qué horas se proyecta una película específica para poder planificar su visita al cine. Esta es la consulta más básica y frecuente.



## Objetivo

Escribe una función en Python llamada **consultar\_horarios** que reciba una cartelera (una lista de diccionarios) y un **titulo\_pelicula**. Cada diccionario en la cartelera representa una película y contiene su 'titulo' y una lista de 'horarios'. La función debe devolver la lista de horarios para la película buscada. Si la película no está en la cartelera, debe devolver una lista vacía.

## Ejemplo real

Entrar a la app del cine, seleccionar "Dune: Part Two" y ver la lista de horarios disponibles para hoy: ["17:00", "20:00", "23:00"].

## Pruebas de validación

```
def probar_consultar_horarios():
    cartelera = [
        {'titulo': 'Inception', 'horarios': ['18:00', '21:00']},
        {'titulo': 'The Matrix', 'horarios': ['17:30', '20:30']},
        {'titulo': 'Parasite', 'horarios': ['19:00', '22:00']}
    ]

    # Prueba 1: Consultar horarios de una película existente
    horarios_matrix = consultar_horarios(cartelera, 'The Matrix')
    print(f"Prueba 1: {horarios_matrix == ['17:30', '20:30']}")

    # Prueba 2: Consultar horarios de una película que no está en cartelera
    horarios_interstellar = consultar_horarios(cartelera, 'Interstellar')
    print(f"Prueba 2: {horarios_interstellar == []}")

    # Prueba 3: Consultar en una cartelera vacía
    horarios_vacio = consultar_horarios([], 'Inception')
    print(f"Prueba 3: {horarios_vacio == []}")

    # Descomenta la siguiente línea cuando tengas tu función lista
    # probar_consultar_horarios()
```

Copie el código



Pegue estas líneas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 

# Nivel Medio: Vender Entradas para una Sesión

## Caso

Un cliente ha decidido a qué película y horario quiere ir y procede a comprar las entradas. El sistema debe verificar si hay asientos disponibles y, si es así, actualizar el número de asientos para esa sesión específica.



## Objetivo

Crea una función llamada **vender\_entrada** que reciba sesiones (un diccionario), `titulo_pelicula`, `horario` y `cantidad_entradas`. La estructura de sesiones es un diccionario donde las claves son los títulos de las películas y los valores son otro diccionario con los horarios como clave y los asientos disponibles como valor.

- La función debe verificar si la película y el horario existen, y si hay suficientes asientos.
- Si la venta es posible, debe restar la `cantidad_entradas` de los asientos disponibles y devolver `True`.
- Si no es posible, no debe modificar nada y debe devolver `False`.

Continúa en la  
siguiente página



## Ejemplo real 🔍

- El proceso de compra en una web de cine: seleccionas "Inception" a las 21:00, eliges 2 asientos, y el sistema confirma tu compra, reduciendo los asientos disponibles para esa función.

## Pruebas de validación 📄

```
def probar_vender_entrada():
    sesiones_base = {
        'Inception': {'18:00': 10, '21:00': 5},
        'The Matrix': {'17:30': 25, '20:30': 1}
    }

    # Prueba 1: Venta exitosa
    sesiones_p1 = {k: v.copy() for k, v in sesiones_base.items()}
    resultado1 = vender_entrada(sesiones_p1, 'Inception', '18:00', 3)
    print(f"Prueba 1 (Venta exitosa): {resultado1 is True and sesiones_p1['Inception']['18:00'] == 7}")

    # Prueba 2: Venta fallida por falta de asientos
    sesiones_p2 = {k: v.copy() for k, v in sesiones_base.items()}
    resultado2 = vender_entrada(sesiones_p2, 'The Matrix', '20:30', 2)
    print(f"Prueba 2 (Falta de asientos): {resultado2 is False and sesiones_p2['The Matrix']['20:30'] == 1}")

    # Prueba 3: Venta fallida porque el horario no existe
    sesiones_p3 = {k: v.copy() for k, v in sesiones_base.items()}
    resultado3 = vender_entrada(sesiones_p3, 'Inception', '19:00', 1)
    print(f"Prueba 3 (Horario no existe): {resultado3 is False}")

# Descomenta la siguiente línea cuando tengas tu función lista
# probar_vender_entrada()
```

Copie el código



Pegue estas líneas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 🤖🤖

# Nivel Avanzado: Generar Reporte de Taquilla

## Caso

Al final del día, el gerente del cine necesita saber qué películas generaron más ingresos para tomar decisiones de negocio, como qué películas mantener en cartelera o en qué salas proyectarlas.



## Objetivo

Escribe una función llamada **generar\_reporte\_taquilla** que reciba una `cartelera_completa`. Esta es una lista de diccionarios, donde cada uno tiene `'titulo'`, `'precio_entrada'` y un diccionario de `'sesiones'` con el formato `{'horario': {'total_asientos': 100, 'vendidas': 80}}`.

- La función debe calcular los ingresos totales para cada película (suma de `precio_entrada` \* `vendidas` de todas sus sesiones).
- Debe devolver una lista de tuplas, donde cada tupla contiene `('titulo', ingresos_totales)`.
- La lista debe estar ordenada de mayor a menor según los ingresos.

Continúa en la  
siguiente página



## Ejemplo real 🔎

Un sistema de Business Intelligence que genera un gráfico de barras mostrando los ingresos por película, permitiendo al gerente ver rápidamente que "Barbie" generó \$15,000 mientras que "Blue Beetle" generó \$4,500.

## Pruebas de validación 📄

```
def probar_generar_reporte_taquilla():
    cartelera = [
        {
            'titulo': 'Oppenheimer',
            'precio_entrada': 12.50,
            'sesiones': {
                '19:00': {'total_asientos': 100, 'vendidas': 80}, # 12.50 * 80 = 1000
                '22:00': {'total_asientos': 100, 'vendidas': 90} # 12.50 * 90 = 1125
            } # Total: 2125.0
        },
        {
            'titulo': 'Barbie',
            'precio_entrada': 10.00,
            'sesiones': {
                '18:30': {'total_asientos': 120, 'vendidas': 110}, # 10.00 * 110 = 1100
                '21:30': {'total_asientos': 120, 'vendidas': 115} # 10.00 * 115 = 1150
            } # Total: 2250.0
        },
        {
            'titulo': 'Gran Turismo',
            'precio_entrada': 11.00,
            'sesiones': {
                '20:00': {'total_asientos': 80, 'vendidas': 40} # 11.00 * 40 = 440.0
            } # Total: 440.0
        }
    ]

    # Prueba 1: Generar reporte ordenado
    reporte = generar_reporte_taquilla(cartelera)
    esperado1 = [
        ('Barbie', 2250.0),
        ('Oppenheimer', 2125.0),
        ('Gran Turismo', 440.0)
    ]
    print(f"Prueba 1: {reporte == esperado1}")

    # Prueba 2: Cartelera vacía
    reporte2 = generar_reporte_taquilla([])
    print(f"Prueba 2: {reporte2 == []}")

    # Descomenta la siguiente línea cuando tengas tu función lista
    # probar_generar_reporte_taquilla()
```

←  
Copie el código

Pegue estas líneas en su archivo Python donde lo desarrollará, debe de validarse exitosamente 🤖🤖



## Éxitos en los ejercicios

Un buen diseño agrega valor más rápido de lo que agrega costo

Thomas C. Gale

Aprende mucho