

Universidad Privada Domingo Savio



Integrantes :

Flavia Gutiérrez Soliz

Docente:

Jimmy Requena

AGENDA

Configure ReplShell

~/workspace: python3 contactos.py

MENÚ DE CONTACTOS

1Agregar nuevo contacto

2Mostrar todos los contactos

3Mostrar contactos por organización

4Editar contacto

5Eliminar contacto

0Salir

• Elige una opción: 1

👤 Nombre: Wilson

☎ Teléfono: 75963215

📁 Organización (familia, amigo, trabajo, otro): otro

✅ Contacto 'Wilson' añadido con éxito.

MENÚ DE CONTACTOS

1Agregar nuevo contacto

2Mostrar todos los contactos

3Mostrar contactos por organización

4Editar contacto

5Eliminar contacto

0Salir

• Elige una opción: 2

LISTA DE CONTACTOS

ID 3

👤 Jefe

☎ 77837829

📁 Trabajo

ID 1

👤 Mama

☎ 75599103

📁 Familia

ID 4

👤 Wilson

☎ 75963215

📁 Otro

ID 2

👤 Yo

☎ 77837829

📁 Familia

README.md

contactos.py

contactos.py > ...

88

if contacto:

89

print(f"\n❌ Editando contacto ID {id_contacto} (deja vacío para no cambiar)")

90

nuevo_nombre = input(f"👤 Nuevo nombre [{contacto['nombre']}]: ") or contacto['nombre']

91

nuevo_telefono = input(f"☎ Nuevo teléfono [{contacto['telefono']}]: ") or contacto['telefono']

92

nueva_org = input(f"📁 Nueva organización [{contacto['organizacion']}]: ") or contacto['organizacion']

93

contacto.update({

94

"nombre": nuevo_nombre,

95

"telefono": nuevo_telefono,

96

"organizacion": nueva_org.lower().strip()

97

})

98

guardar_contactos()

99

print("✅ Contacto actualizado con éxito.\n")

100

else:

101

print(f"❌ No se encontró contacto con ID {id_contacto}.")

102

103

104

105

Eliminar

106

Ln 118, Col 3 • Spaces: 4 Histor

BATALLA NAVAL

Configure ReplShell

~/workspace: python3 batallanaval.py

0 1 2 3

0 X 🚢 🚢 🚢

1 🚢 🚢 🚢 🚢

2 🚢 🚢 🚢 🚢

3 🚢 🚢 🚢 🚢

Ingres a fila (0-3): 2

Ingres a columna (0-3): 2

🌊 Agua.

Intentos restantes: 5 | Barcos restantes: 1

0 1 2 3

0 X 🚢 🚢 🚢

1 🚢 🚢 🚢 🚢

2 🚢 🚢 X 🚢

3 🚢 🚢 🚢 🚢

Ingres a fila (0-3): 3

Ingres a columna (0-3): 1

🔥 ¡Tocado!

Intentos restantes: 5 | Barcos restantes: 0

0 1 2 3

0 X 🚢 🚢 🚢

1 🚢 🚢 🚢 🚢

2 🚢 🚢 X 🚢

3 🚢 🚢 🚢 🚢

🏆 ¡Ganaste! Hundiste todos los barcos.

README.md

batallanaval.py

batallanaval.py > f jugar_batalla_naval > ...

58

barcos.remove((fila, col))

59

aciertos += 1

60

else:

61

print("🌊 Agua.")

62

tablero[fila][col] = "X"

63

intentos -= 1

64

65

print(f"Intentos restantes: {intentos} | Barcos restantes: {NUM_BARCOS - aciertos}\n")

66

67

final

68

imprimir_tablero(tablero)

69

if aciertos == NUM_BARCOS:

70

print("\n🏆 ¡Ganaste! Hundiste todos los barcos.")

71

else:

72

print("\n💀 ¡Perdiste! Los barcos restantes estaban en:")

73

for fila, col in barcos:

74

print(f" - ({fila}, {col})")

75

76

Ejecutar

77

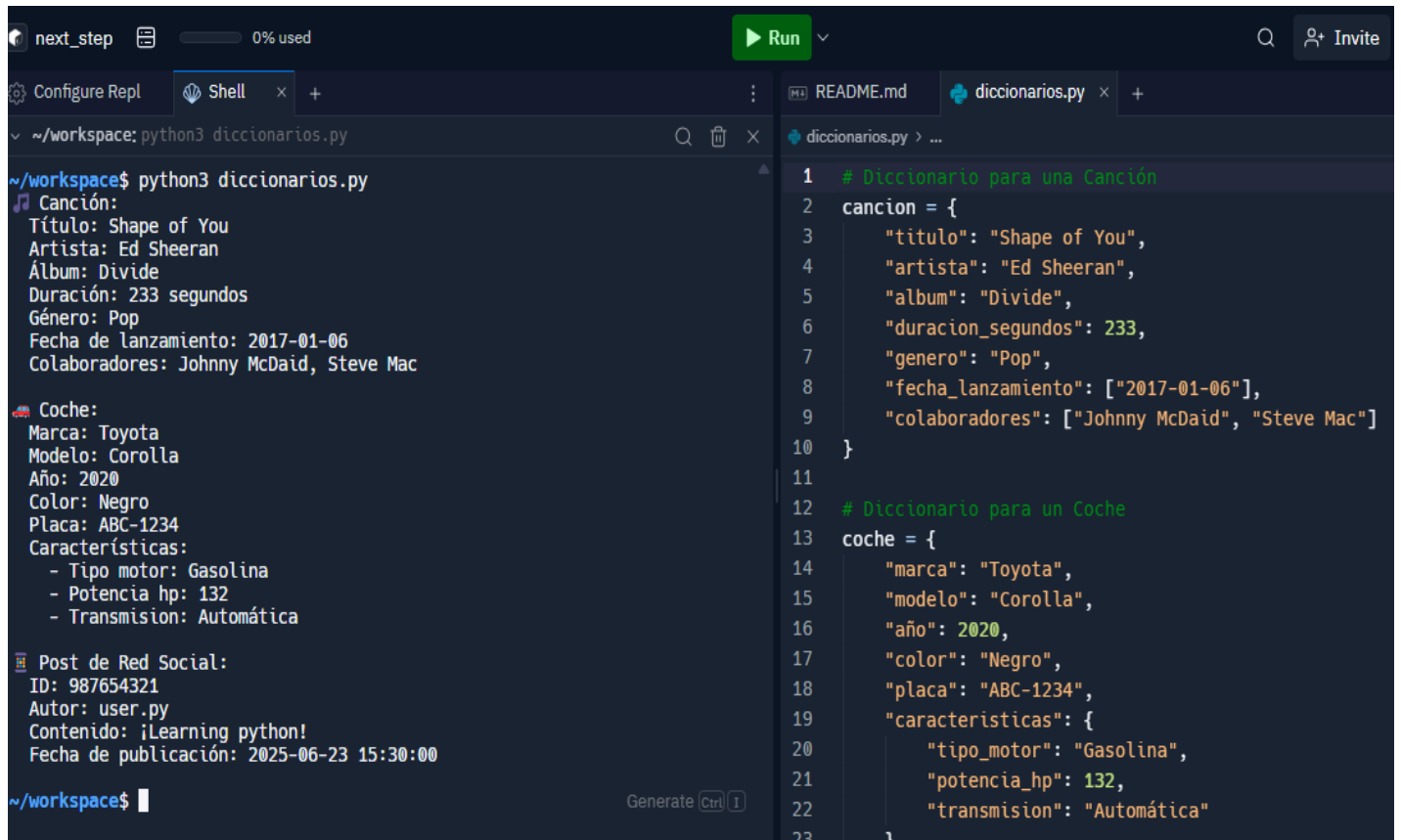
if __name__ == "__main__":

78

jugar_batalla_naval()

Ln 64, Col 1 • Spaces: 4

DICCIONARIO 1



```
~/workspace: python3 diccionarios.py
Canción:
Título: Shape of You
Artista: Ed Sheeran
Álbum: Divide
Duración: 233 segundos
Género: Pop
Fecha de lanzamiento: 2017-01-06
Colaboradores: Johnny McDaid, Steve Mac

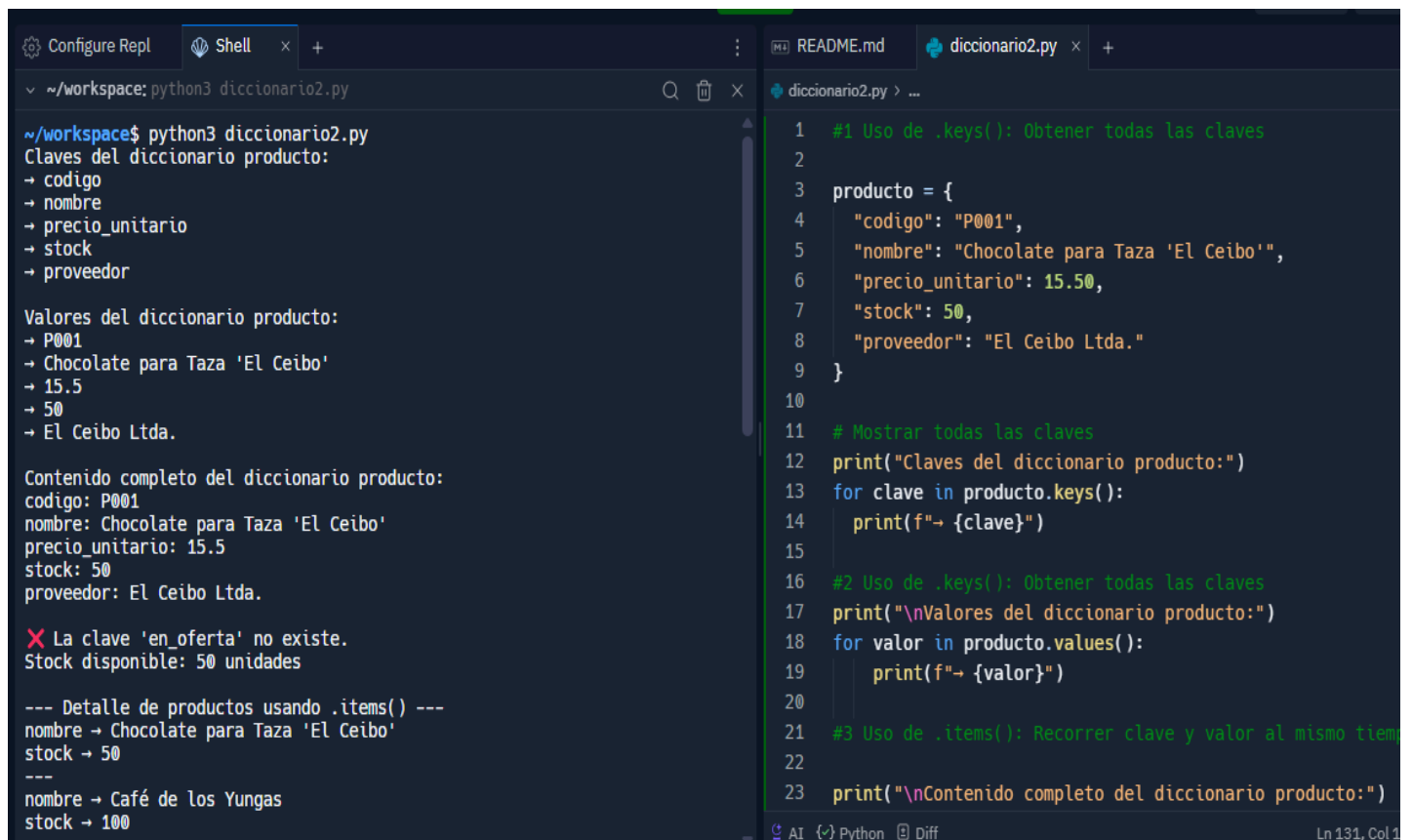
Coche:
Marca: Toyota
Modelo: Corolla
Año: 2020
Color: Negro
Placa: ABC-1234
Características:
- Tipo motor: Gasolina
- Potencia hp: 132
- Transmisión: Automática

Post de Red Social:
ID: 987654321
Autor: user.py
Contenido: ¡Learning python!
Fecha de publicación: 2025-06-23 15:30:00

~/workspace$
```

```
1 # Diccionario para una Canción
2 cancion = {
3     "titulo": "Shape of You",
4     "artista": "Ed Sheeran",
5     "album": "Divide",
6     "duracion_segundos": 233,
7     "genero": "Pop",
8     "fecha_lanzamiento": ["2017-01-06"],
9     "colaboradores": ["Johnny McDaid", "Steve Mac"]
10 }
11
12 # Diccionario para un Coche
13 coche = {
14     "marca": "Toyota",
15     "modelo": "Corolla",
16     "año": 2020,
17     "color": "Negro",
18     "placa": "ABC-1234",
19     "características": {
20         "tipo_motor": "Gasolina",
21         "potencia_hp": 132,
22         "transmisión": "Automática"
23     }
```

DICCIONARIO 2



```
~/workspace: python3 diccionario2.py
Claves del diccionario producto:
→ codigo
→ nombre
→ precio_unitario
→ stock
→ proveedor

Valores del diccionario producto:
→ P001
→ Chocolate para Taza 'El Ceibo'
→ 15.5
→ 50
→ El Ceibo Ltda.

Contenido completo del diccionario producto:
codigo: P001
nombre: Chocolate para Taza 'El Ceibo'
precio_unitario: 15.5
stock: 50
proveedor: El Ceibo Ltda.

❌ La clave 'en_oferta' no existe.
Stock disponible: 50 unidades

--- Detalle de productos usando .items() ---
nombre → Chocolate para Taza 'El Ceibo'
stock → 50
---
nombre → Café de los Yungas
stock → 100
```

```
1 #1 Uso de .keys(): Obtener todas las claves
2
3 producto = {
4     "codigo": "P001",
5     "nombre": "Chocolate para Taza 'El Ceibo'",
6     "precio_unitario": 15.50,
7     "stock": 50,
8     "proveedor": "El Ceibo Ltda."
9 }
10
11 # Mostrar todas las claves
12 print("Claves del diccionario producto:")
13 for clave in producto.keys():
14     print(f"→ {clave}")
15
16 #2 Uso de .values(): Obtener todas las claves
17 print("\nValores del diccionario producto:")
18 for valor in producto.values():
19     print(f"→ {valor}")
20
21 #3 Uso de .items(): Recorrer clave y valor al mismo tiempo
22
23 print("\nContenido completo del diccionario producto:")
```

INVENTARIO

```
Configure Repl  Shell  x  +  README.md  inventario.py  x  +  
~/workspace: python3 inventario.py  
Número de tipos de producto en el inventario: 3  
  
--- Inventario Actual ---  
- Short 'Guess': 15 unidades en stock.  
- Pantalón jean: 10 unidades en stock.  
- Crop Top 'Strapless': 12 unidades en stock.  
~/workspace$  
Generate Ctrl I
```

```
1 # 1. Crear una lista vacía llamada inventario  
2 inventario = []  
3  
4 # 2. Crear tres diccionarios con productos diferentes  
5 producto1 = {"nombre": "Short 'Guess'", "stock": 15}  
6 producto2 = {"nombre": "Pantalón jean", "stock": 10}  
7 producto3 = {"nombre": "Crop Top 'Strapless'", "stock": 12}  
8  
9 # 3. Añadir los diccionarios a la lista inventario  
10 inventario.append(producto1)  
11 inventario.append(producto2)  
12 inventario.append(producto3)  
13  
14 # 4. Imprimir la cantidad de tipos de producto en el inventario  
15 print(f"Número de tipos de producto en el inventario:  
16 {len(inventario)}\n")  
17  
18 # 5. Recorrer la lista inventario e imprimir el resumen de cada prod  
19 print("--- Inventario Actual ---")  
20 for producto in inventario:  
21     print(f"- {producto['nombre']}: {producto['stock']} unidades en  
22 stock.")
```

MATRIZ 1

```
Configure Repl  Shell  x  +  README.md  Matrix.py  x  +  
~/workspace: python3 Matrix.py  
El elemento en (0,0) es 1  
El elemento en (0,1) es 2  
El elemento en (0,2) es 3  
1 2 3  
4 5 6  
7 8 9  
El elemento en (1,0) es 4  
El elemento en (1,1) es 5  
El elemento en (1,2) es 6  
1 2 3  
4 5 6  
7 8 9  
El elemento en (2,0) es 7  
El elemento en (2,1) es 8  
El elemento en (2,2) es 9  
1 2 3  
4 5 6  
7 8 9  
~/workspace$  
Generate Ctrl I
```

```
1 matriz= [  
2     [1, 2, 3],  
3     [4, 5, 6],  
4     [7, 8, 9]  
5 ]  
6 #opcion 1: recorriendo con indices  
7 num_filas=len(matriz)  
8 num_columnas=len(matriz[0])  
9 for i in range(num_filas):  
10     for j in range(num_columnas):  
11         elemento=matriz[i][j]  
12         print(f"El elemento en ({i},{j}) es {elemento}")  
13  
14 #opcion 2: recorriendo con elementos  
15 for fila_actual in matriz:  
16     for elemento in fila_actual:  
17         print(elemento, end=" ")  
18     print()
```

MATRIZ 2

The screenshot shows a code editor with two tabs: 'matrix1.py' and 'matrix'. The left pane shows the terminal output of running 'python3 matrix1.py'. The right pane shows the source code of 'matrix1.py'.

```
~/workspace$ python3 matrix1.py
1 2 3
4 5 6
7 8 9
Matriz con 0 en lugar de 1
0 2 3
4 5 6
7 8 9
~/workspace$
```

```
1 matriz= [
2     [1, 2, 3],
3     [4, 5, 6],
4     [7, 8, 9]
5 ]
6 for fila in matriz:
7     for elemento in fila:
8         print(elemento, end=" ")
9     print()
10
11 #Modificando matriz
12 print("Matriz con 0 en lugar de 1")
13
14 for i in range(len(matriz)):
15     for j in range(len(matriz[i])):
16         if matriz[i][j] == 1:
17             matriz[i][j] = 0
18 for fila in matriz:
19     for elemento in fila:
20         print(elemento, end=" ")
21     print()
```

MATRIZ 3

The screenshot shows a code editor with two tabs: 'matrix2.py' and 'matrix'. The left pane shows the terminal output of running 'python3 matrix2.py'. The right pane shows the source code of 'matrix2.py'.

```
~/workspace$ python3 matrix2.py
0 2 3
4 5 6
7 8 9
~/workspace$
```

```
1 matriz= [
2     [0, 2, 3],
3     [4, 5, 6],
4     [7, 8, 9]
5 ]
6 for fila in matriz:
7     for elemento in fila:
8         print(elemento, end="\t")
9     print()
10
```

MATRIZ 4

The screenshot shows a code editor with two tabs: 'matrix2-2.py' and 'matrix_5x5'. The left pane shows the terminal output of running 'python3 matrix2-2.py'. The right pane shows the source code of 'matrix2-2.py'.

```
~/workspace$ python3 matrix2-2.py
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
~/workspace$
```

```
1 matriz_5x5 = []
2
3 for i in range(5):
4     fila = []
5     for j in range(5):
6         fila.append(0)
7     matriz_5x5.append(fila)
8
9 # Imprimir la matriz
10 for fila in matriz_5x5:
11     for elemento in fila:
12         print(elemento, end="\t")
13     print()
14
```

POO 1

Configure ReplShell x +

~/workspace: python3 poo.py

~/workspace\$ python3 poo.py

=====

INFORMACIÓN DEL LIBRO

=====

Título: Cien años de soledad

Autor: Gabriel García Márquez

ISBN: 978-0307474728

Páginas: 417

Disponible: Sí

=====

Después de cambiar disponibilidad:

=====

INFORMACIÓN DEL LIBRO

=====

Título: Cien años de soledad

Autor: Gabriel García Márquez

ISBN: 978-0307474728

Páginas: 417

Disponible: No

=====

~/workspace\$

Generate Ctrl I

README.md

poo.py x +

poo.py > ...

39

40 # Ejemplo de uso (no te preocupes por crear objetos todavía, solo

define la clase!)

41 if __name__ == "__main__":

42 # Creación de ejemplo para demostrar el funcionamiento

43 libro1 = Libro("Cien años de soledad", "Gabriel García Márquez", "978-

0307474728", 417)

44 libro1.mostrar_info()

45

46 # Cambiar disponibilidad

47 libro1.disponible = False

48 print("\nDespués de cambiar disponibilidad:")

49 libro1.mostrar_info()

POO 2

Configure ReplShell x +

~/workspace: python3 poo2.py

--- INVENTARIO COMPLETO DE LA BIBLIOTECA ---

=====

INFORMACIÓN DEL LIBRO

=====

Título: Cien años de soledad

Autor: Gabriel García Márquez

ISBN: 978-0307474728

Páginas: 417

Disponible: Sí

=====

=====

INFORMACIÓN DEL LIBRO

=====

Título: 1984

Autor: George Orwell

ISBN: 978-0451524935

Páginas: 328

Disponible: Sí

=====

=====

INFORMACIÓN DEL LIBRO

=====

Título: El Principito

Autor: Antoine de Saint-Exupéry

ISBN: 978-0156013987

Páginas: 96

Disponible: Sí

=====

=====

README.md

poo2.py x +

poo2.py > ...

35

36

37 if __name__ == "__main__":

38 # Crear objetos de tipo Libro

39 libro1 = Libro("Cien años de soledad", "Gabriel García Márquez", "978-

0307474728", 417)

40 libro2 = Libro("1984", "George Orwell", "978-0451524935", 328)

41 libro3 = Libro("El Principito", "Antoine de Saint-Exupéry", "978-

0156013987", 96)

42

43 # Crear una lista vacía

44 mi_biblioteca = []

45

46 # Añadir libros a la lista

47 mi_biblioteca.append(libro1)

48 mi_biblioteca.append(libro2)

49 mi_biblioteca.append(libro3)

50

51 # Mostrar el inventario completo

52 print("\n\n--- INVENTARIO COMPLETO DE LA BIBLIOTECA ---")

53 for libro_actual in mi_biblioteca:

54 libro_actual.mostrar_info()

55 print("\n * 20) # Separador

POO 3-1

The screenshot shows a code editor with two panels. The left panel displays the output of running a Python script named `poo3.py`. The output shows the author and ISBN of two books, followed by a detailed display of book information for 'El Principito' and 'Raza de Bronce', and finally a test of the book's methods.

```
~/workspace$ python3 poo3.py

El autor del primer libro es: Antoine de Saint-Exupéry
El ISBN del segundo libro es: 978-99905-2-213-9

--- Mostrando información completa ---

=====
INFORMACIÓN DEL LIBRO
=====
Título: El Principito
Autor: Antoine de Saint-Exupéry
ISBN: 978-3-14-046401-7
Páginas: 120
Disponible: Sí
=====

INFORMACIÓN DEL LIBRO
=====
Título: Raza de Bronce
Autor: Alcides Arguedas
ISBN: 978-99905-2-213-9
Páginas: 250
Disponible: Sí
=====

--- Probando métodos de comportamiento ---
El libro 'El Principito' ha sido prestado.
El libro 'El Principito' ya está prestado.
El libro 'El Principito' ha sido devuelto.
El libro 'El Principito' ya estaba disponible.
```

The right panel shows the source code of `poo3.py`, which defines two book classes and tests their methods.

```
73 # Prestar libro1
74 libro1.prestar_libro()
75
76 # Intentar prestar libro1 otra vez (ya prestado)
77 libro1.prestar_libro()
78
79 # Devolver libro1
80 libro1.devolver_libro()
81
82 # Intentar devolver libro1 otra vez (ya disponible)
83 libro1.devolver_libro()
84
85 print("\n--- Probando con libro2 ---")
86
87 # Prestar libro2
88 libro2.prestar_libro()
89
90 # Devolver libro2
91 libro2.devolver_libro()
92
93 print("\n--- Estado final de los libros ---")
94 libro1.mostrar_info()
95 libro2.mostrar_info()
```

POO 3-2

The screenshot shows a code editor with two panels. The left panel displays the output of running a Python script named `poo3.py`. The output shows the author and ISBN of two books, followed by a detailed display of book information for 'El Principito' and 'Raza de Bronce', and finally a test of the book's methods.

```
--- Probando métodos de comportamiento ---
El libro 'El Principito' ha sido prestado.
El libro 'El Principito' ya está prestado.
El libro 'El Principito' ha sido devuelto.
El libro 'El Principito' ya estaba disponible.

--- Probando con libro2 ---
El libro 'Raza de Bronce' ha sido prestado.
El libro 'Raza de Bronce' ha sido devuelto.

--- Estado final de los libros ---

=====
INFORMACIÓN DEL LIBRO
=====
Título: El Principito
Autor: Antoine de Saint-Exupéry
ISBN: 978-3-14-046401-7
Páginas: 120
Disponible: Sí
=====

INFORMACIÓN DEL LIBRO
=====
Título: Raza de Bronce
Autor: Alcides Arguedas
ISBN: 978-99905-2-213-9
Páginas: 250
Disponible: Sí
=====

~/workspace$
```

The right panel shows the source code of `poo3.py`, which defines two book classes and tests their methods.

```
73 # Prestar libro1
74 libro1.prestar_libro()
75
76 # Intentar prestar libro1 otra vez (ya prestado)
77 libro1.prestar_libro()
78
79 # Devolver libro1
80 libro1.devolver_libro()
81
82 # Intentar devolver libro1 otra vez (ya disponible)
83 libro1.devolver_libro()
84
85 print("\n--- Probando con libro2 ---")
86
87 # Prestar libro2
88 libro2.prestar_libro()
89
90 # Devolver libro2
91 libro2.devolver_libro()
92
93 print("\n--- Estado final de los libros ---")
94 libro1.mostrar_info()
95 libro2.mostrar_info()
```


SALA CINE

Configure ReplShell x +

~/workspace: python3 salacine.py

Sala actual:

	0	1	2	3	4	5	6	7
F 0	L	L	L	L	L	L	L	L
F 1	L	L	L	L	L	L	L	L
F 2	L	L	L	L	L	L	L	L
F 3	L	L	L	L	L	L	L	L
F 4	L	L	L	L	L	L	L	L

Asientos libres: 40

Menú:
1. Ocupar asiento individual
2. Buscar y ocupar N asientos juntos
0. Salir
Elige una opción: 1
Fila: 3
Columna: 5
Asiento (3, 5) reservado por Bs. 50

Sala actual:

	0	1	2	3	4	5	6	7
F 0	L	L	L	L	L	L	L	L
F 1	L	L	L	L	L	L	L	L
F 2	L	L	L	L	L	L	L	L
F 3	L	L	L	L	0	L	L	L
F 4	L	L	L	L	L	L	L	L

Asientos libres: 39

salacine.py > ...

```
10         else:
11             precio = 30 # Asientos de los costados
12             fila.append({"estado": "L", "precio": precio})
13             sala.append(fila)
14         return sala
15
16 # Mostrar la sala con precios y estados
17 def mostrar_sala(sala):
18     print("\n      " + " ".join(f"{j:^5}" for j in range(len(sala[0])))
19     print("      " + " ".join("-" * 5 for _ in range(len(sala[0])))
20     for i, fila in enumerate(sala):
21         estado_fila = " ".join(f"{a['estado']:^5}" for a in fila)
22         print(f"F{i:>2} | {estado_fila}")
23
24 # Ocupar asiento individual
25 def ocupar_asiento(sala, fila, columna):
26     if 0 <= fila < len(sala) and 0 <= columna < len(sala[0]):
27         asiento = sala[fila][columna]
28         if asiento["estado"] == "L":
29             asiento["estado"] = "0"
30             print(f"Asiento ({fila}, {columna}) reservado por Bs.
31             {asiento['precio']}")
32             return True
33     else:
```

TIENDA ONLINE

Configure ReplShell x +

~/workspace: python3 tiendaonline.py

~/workspace\$ python3 tiendaonline.py

Camiseta Roja ha sido agregado al carrito.

Pantalón Azul ha sido agregado al carrito.

Lo siento, Zapatillas Deportivas no está disponible.

Carrito de Juan Pérez:

=====

Producto: Camiseta Roja

Precio: \$19.99

Disponible: Sí

=====

Producto: Pantalón Azul

Precio: \$39.99

Disponible: Sí

=====

Total de la compra: \$59.98

El carrito ha sido vaciado.

Compra realizada por Juan Pérez.

Carrito de Juan Pérez:

El carrito está vacío.

~/workspace\$

tiendaonline.py > ...

```
1 class Producto:
2     """
3     Clase que representa un producto en la tienda online.
4     """
5
6     def __init__(self, nombre, precio, disponible=True):
7         """
8         Constructor de la clase Producto.
9
10        Args:
11            nombre (str): Nombre del producto
12            precio (float): Precio del producto
13            disponible (bool): Disponibilidad del producto (por defecto
14            True)
15        """
16        self.nombre = nombre
17        self.precio = precio
18        self.disponible = disponible
19
20    def mostrar_info(self):
21        """
22        Método que imprime la información del producto.
23        """
```


TO DO LIST

Configure ReplShell x +

~/workspace: python3 clase13.py

~/workspace\$ python3 clase13.py

==== MENÚ TO-DO LIST ====
1. Agregar nueva tarea
2. Mostrar todas las tareas
3. Marcar tarea como completada
4. Eliminar tarea
0. Salir
Elige una opción: 1
Descripción de la nueva tarea: Limpiar
Prioridad (alta, media, baja): media
✅ Tarea 'Limpiar' añadida con éxito.

==== MENÚ TO-DO LIST ====
1. Agregar nueva tarea
2. Mostrar todas las tareas
3. Marcar tarea como completada
4. Eliminar tarea
0. Salir
Elige una opción: 2

--- 📖 LISTA DE TAREAS ---
■ ID: 1 | Limpiar (Prioridad: media)

==== MENÚ TO-DO LIST ====
1. Agregar nueva tarea
2. Mostrar todas las tareas
3. Marcar tarea como completada
4. Eliminar tarea
0. Salir
Elige una opción:

README.mdclase13.py x +

clase13.py > ...

1 # TO-DO LIST EN PYTHON
2 # =====
3
4 # Paso 1: Variables Globales
5 lista_de_tareas = []
6 proximo_id_tarea = 1 # Para generar IDs únicos
7
8 # Paso 2: Implementar agregar_tarea
9 def agregar_tarea(descripcion, prioridad='media'):
10 global proximo_id_tarea
11 nueva_tarea = {
12 "id": proximo_id_tarea,
13 "descripcion": descripcion,
14 "completada": False,
15 "prioridad": prioridad
16 }
17 lista_de_tareas.append(nueva_tarea)
18 proximo_id_tarea += 1
19 print(f"✅ Tarea '{descripcion}' añadida con éxito.")
20
21 # Paso 3: Implementar mostrar_tareas
22 def mostrar_tareas():
23 print("\n--- 📖 LISTA DE TAREAS ---")

AI Python DiffLn 1, Col 1

TRANSFORMACIONES

next_step0% usedRun

Configure ReplShell x +

~/workspace: python3 clase11transf.py

~/workspace\$ python3 clase11transf.py
¡Pruebas para es_simetrica pasaron! ✅
¡Pruebas para es_identidad pasaron! ✅
¡Prueba 1 (2x3) pasada! ✅
~/workspace\$

Generate Ctrl I

README.mdclase11transf.py x +

clase11transf.py > ...

1 def es_simetrica(matriz):
2 # Requisito 1: Debe ser cuadrada
3 num_filas = len(matriz)
4 if num_filas == 0:
5 return True # Una matriz vacía es trivialmente simétrica
6
7 for i in range(num_filas):
8 if len(matriz[i]) != num_filas:
9 return False # No es cuadrada
10
11 # Requisito 2: Comparar matriz[i][j] con matriz[j][i]
12 for i in range(num_filas):
13 for j in range(i + 1, num_filas): # Solo necesitamos chequear la triangular superior
14 if matriz[i][j] != matriz[j][i]:
15 return False # ¡Con una diferencia es suficiente!
16
17 return True # Si nunca encontramos diferencias, es simétrica
18
19 # Prueba tu función:
20 sim = [[1, 7, 3], [7, 4, -5], [3, -5, 6]]
21 no_sim = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
22 no_cuadrada = [[1, 2], [3, 4], [5, 6]]

AI Python DiffLn 26, Col 26 • Spaces: 4 Histor