Título de la práctica: Paradigma Orientado a Objetos		
Asignatura: POO	**Fecha de inicio: 17 sep. 25**	**Hoja:** 1 **de 2**
Unidad temática: Unidad I	**Fecha de entrega: 17 sep. 25**	**Grupo:**
No. de participantes recomendados: Equipos	**Elaboró:** ISC Adriana Y. Contreras Álvarez	
Lugar Asignado: Laboratorio	**Revisó:**	
Alumno:		

Objetivo de la práctica:

Que el estudiante sea capaz de:

1. Definir clases con atributos y métodos.
2. Crear objetos a partir de las clases.
3. Utilizar métodos para manipular datos internos de un objeto.
4. Comprender y aplicar la relación de composición (un objeto contiene a otro).

Resultados esperados.

1. Código ejecutado paso a paso.
2. Capturas de pantalla de cada tarea.
3. ¿Cuál es la diferencia entre que un libro exista **fuera** de la biblioteca y que la biblioteca lo **contenga** en su lista?
4. ¿Cómo cambia el comportamiento del sistema si quitamos la composición?
5. ¿Qué ventajas nos da estructurar el sistema en **clases y objetos** en lugar de variables sueltas?

Marco teórico.

Material requerido:

1. Computadora
2. https://www.programiz.com/python-programming/online-compiler

Desarrollo de la práctica

Imagina que vamos a modelar una Biblioteca Universitaria.

- Una Biblioteca tiene un nombre y una colección de libros.
- Cada Libro tiene título, autor y un estado (prestado o disponible).
- La biblioteca puede agregar libros y mostrar su catálogo.

1. Crear la clase `Libro`

- Atributos: titulo, autor, disponible (por defecto True).
- Métodos:
  - prestar(): cambia el estado a **no disponible**.
  - devolver(): cambia el estado a **disponible**.
  - __str__(): devuelve una representación en texto del libro.

```
class Libro:
    def __init__(self, titulo, autor):
        self.titulo = titulo
        self.autor = autor
        self.disponible = True

    def prestar(self):
        if self.disponible:
            self.disponible = False
            print(f"El libro '{self.titulo}' ha sido prestado.")
        else:
            print(f"El libro '{self.titulo}' no está disponible.")

    def devolver(self):
        self.disponible = True
        print(f"El libro '{self.titulo}' ha sido devuelto.")

    def __str__(self):
        estado = "Disponible" if self.disponible else "Prestado"
        return f"{self.titulo} - {self.autor} ({estado})"
```

2. Crear la clase `Biblioteca` (Composición)

- Atributos: `nombre`, `libros` (lista vacía al inicio).
- Métodos:
  - `agregar_libro(libro)`: agrega un objeto de tipo `Libro` a la lista.
  - `mostrar_libros()`: imprime todos los libros de la biblioteca.

```
class Biblioteca:
    def __init__(self, nombre):
        self.nombre = nombre
        self.libros = []   # Aquí ocurre la composición

    def agregar_libro(self, libro):
        self.libros.append(libro)
        print(f"Se agregó '{libro.titulo}' a la biblioteca.")

    def mostrar_libros(self):
        print(f"\nCatálogo de la Biblioteca {self.nombre}:")
        for libro in self.libros:
            print(libro)
```

3. Crear objetos y probar la composición

```
# Crear objetos de tipo Libro
libro1 = Libro("Cien Años de Soledad", "Gabriel García Márquez")
libro2 = Libro("1984", "George Orwell")
libro3 = Libro("El Principito", "Antoine de Saint-Exupéry")

# Crear una biblioteca
mi_biblioteca = Biblioteca("Central Universitaria")

# Agregar libros a la biblioteca
mi_biblioteca.agregar_libro(libro1)
```

```
mi_biblioteca.agregar_libro(libro2)
mi_biblioteca.agregar_libro(libro3)

# Mostrar los libros
mi_biblioteca.mostrar_libros()

# Prestar y devolver un libro
libro1.prestar()
libro1.prestar()  # Intento de préstamo fallido
libro1.devolver()

# Ver catálogo actualizado
mi_biblioteca.mostrar_libros()
```

Código completo:

https://www.programiz.com/online-compiler/60mS9hzzJU6lx