# Práctica 5

## Objetivos:

Practicar conceptos extendidos de funciones y módulos en Python.

## Ejercicio 1

Cree un programa que gestione operaciones matemáticas utilizando decoradores, funciones lambda y *args/**kwargs. Considere las siguientes tareas:

- Cree un decorador llamado `operation_logger` que registre el nombre de la operación, las entradas y el resultado.
- Cree funciones lambda para operaciones matemáticas básicas (suma, resta, multiplicación, división).
- Cree una función llamada `math_operation` que tome una operación (función lambda) y cualquier número de argumentos, y devuelva el resultado.
- Use el decorador en la función `math_operation`.
- Implemente manejo de errores para la división por cero.
- Pruebe el sistema con al menos las siguientes operaciones y entradas:

```
math_operation(add, 5, 3)
math_operation(subtract, 10, 4)
math_operation(multiply, 2, 6)
math_operation(divide, 15, 3)
math_operation(divide, 10, 0)  # Debe manejar la división por 0
math_operation(add, 1, 2, 3, 4, 5)  # Debe manejar múltiples
argumentos
```

## Ejercicio 2

Cree un sistema de gestión de bibliotecas que conste de los siguientes módulos.

- Un módulo `book.py` que defina una clase `Book` con atributos privados: título, autor, ISBN y estado (prestado o no). Utilice el decorador `@property` en todos los atributos. Deberá además implementar `__str__` para visualizar en forma de cadena todos los datos del libro.
- Un módulo `library.py` que defina una clase `Library` para gestionar una colección de libros. Debe incluir métodos para agregar libros, eliminar libros, prestar libros, devolver libros y buscar libros por título o autor.
- Un módulo `user.py` que defina una clase `User` con atributos privados: nombre, ID y libros prestados. Utilice el decorador `@property` en todos los atributos. Deberá además implementar `__str__` para visualizar en forma de cadena todos los datos del usuario.
- Extienda las funciones `leer_int, leer_float` y `crear_menu` de la práctica anterior para que permitan anotaciones y genere errores en el caso que los tipos no sean los esperados. Impleméntelas en un fichero denominado `utils.py`. Para crear anotaciones de tipo lista o diccionario debe utilizar el módulo

`typing`. Implemente en este fichero una nueva función que genere IDs únicos para usuarios y libros. Utilice el módulo `uuid` y la función `uuid4()` para crear un ID de versión 4 aleatorio. Devuelva el ID convertido en una cadena de texto con los 8 primeros caracteres.

- Un script principal `main.py` que utilice todos estos módulos para crear una biblioteca con al menos las siguientes opciones de menú: añadir libro, eliminar libro, registrar usuario, realizar préstamo, realizar devolución, y mostrar todos los libros.