
# Universidad
## Nacional de San Martín

Guía de Ejercicios

Unidad 20: Decoradores

# 1 Ejercicios

## 1.1 Sistema de Autenticación de Usuarios

Desarrolla un sistema de autenticación que gestione el inicio de sesión de usuarios. Usa un decorador para verificar las credenciales del usuario antes de permitirle acceder a una función protegida (por ejemplo, una función que muestre un mensaje de bienvenida). Implementa un decorador adicional para limitar el número de intentos fallidos de inicio de sesión.

## 1.2 Sistema de Registro de Ejecuciones

Crea un programa con varias funciones que realicen operaciones matemáticas (suma, resta, multiplicación y división). Implementa un decorador @logger que registre cada llamada a estas funciones, imprimiendo el nombre de la función, los argumentos y el resultado cada vez que se ejecute alguna de las funciones.

## 1.3 Sistema de Reserva de Mesas

Diseña un programa para gestionar reservas de mesas en un restaurante, utilizando un diccionario donde las claves sean los números de mesa y los valores sean los nombres de los clientes. Crea un decorador que verifique si una mesa está disponible antes de agregar una reserva, y que permita cancelar reservas solo si la mesa está ocupada.

## 1.4 Validación de Entrada en Funciones Matemáticas

Crea varias funciones matemáticas que reciban argumentos numéricos, como cálculo de área y volumen. Implementa un decorador que valide si los parámetros son numéricos y mayores que cero antes de ejecutar la función, y que muestre un mensaje de error si algún parámetro no es válido.

## 1.5 Control de Acceso a Archivos

Desarrolla un sistema que permita a los usuarios abrir y editar archivos de texto. Cada archivo tendrá asignado un nivel de acceso (por ejemplo, “público”, “privado”, “confidencial”). Usa un decorador que verifique si el usuario tiene permiso para acceder al archivo en función de su nivel de acceso. Si el usuario no tiene acceso, muestra un mensaje de “Acceso denegado”.

## 1.6 Ejecutar Función Varias Veces

Escribe una función que imprima un mensaje y utiliza un decorador que permita ejecutarla un número específico de veces, determinado por un parámetro. Usa este decorador en funciones simples para verificar su funcionamiento y que el número de ejecuciones sea flexible.

## 1.7 Medidor de Tiempo de Ejecución

Crea un programa que incluya varias funciones de simulación que usen time.sleep() para simular una operación que toma tiempo (por ejemplo, descargar datos o procesar información). Implementa un decorador que mida el tiempo de ejecución de cada función y que muestre cuánto tiempo tomó ejecutarse.

## 1.8 Cálculo de Impuestos

Escribe un programa que permita calcular el precio final de un producto con un descuento o un impuesto añadido. Implementa un decorador que permita añadir diferentes tasas de impuestos a cada función que calcule precios. Usa este decorador para aplicar distintos porcentajes de impuestos a diferentes productos.

## 1.9 Sistema de Notificaciones de Ejecución

Desarrolla un sistema que envíe notificaciones de correo electrónico cuando se ejecuten ciertas funciones críticas, como transferencias de dinero o acceso a información confidencial. Usa un decorador para interceptar la ejecución de estas funciones y simula el envío de una notificación imprimiendo un mensaje de "Notificación enviada".

## 1.10 Sistema de Cacheo de Resultados

Implementa un sistema de cacheo para una función que calcule el factorial de un número. Usa un decorador que almacene los resultados de cada cálculo en un diccionario para que, si la función se llama nuevamente con el mismo número, el decorador devuelva el resultado de la cache sin realizar el cálculo otra vez.