Practica 2. Clases, Objetos, métodos y atributos

Practica 5. Singleton

Ejemplo de patrón de diseño Singleton - Sistema de registro de logs

class Logger: # Atributo de la clase para guardar la única instancia
_instancia = None

```
# Método __new__ controla la creación del objeto antes de __init__
# Se asegura que solo exista una única instancia de Logger
def __new__(cls):
    if cls._instancia is None:
        cls._instancia = super().__new__(cls)
        # Abrimos un archivo de logs en modo "append"
        cls._instancia.archivo = open("app.log", "a")
    return cls._instancia  # Devuelve siempre la misma instancia


def registro(self, mensaje):
    self.archivo.write(mensaje + "\n")
    self.archivo.flush()  # Fuerza a que el archivo se guarde en el dis
co
```

Creamos la única instancia SINGLETON

registro1 = Logger()

Devuelve la misma instancia sin crear una nueva

registro2 = Logger()

registro1.registro("Inicio de sesión en la aplicación")
registro2.registro("El usuario se autenticó")

print(registro1 is registro2) # True, ambas variables apuntan a la misma instancia

#1. ¿Qué pasaría si eliminamos la verificación if cls.instancia is None en el método __new_?no se crearia una nueva #2. En la línea print(registro1 is registro2). ¿Qué significa que devuelva True en el contexto del Singleton?Significa que ambas variables apuntan a la misma intancia #3. ¿Crees que siempre es buena idea usar Singleton para todo lo que es global? Da un ejemplo donde no sería recomendable.sirve para ahorrar memoria por lo tanto puede servir donde un programa sea demasiado grnde