Practica 5

```
# Practica 5. Singleton
# Ejemplo de patrón de diseño Singleton - Sistema de Registro de logs.

class Logger:
    # Atributo de clase para guardar la única instancia
    _instance = None
# Método __new__ controla la creacion del objeto antes de init. Se asegura de que solo exista
    # Método __new__ controla la creación del objeto
    def __new__(cls, *args, **kwargs):
        if cls._instance is None:   # Se crea solo una vez
            cls._instance = super().__new__(cls)
            # Arbimos un archivo de logs en modo "append (agregar)".
            cls._instance.archivo = open("app.log", "a")
        return cls._instance # Devuelve siempre a la misma instancia.

    def log(self, mensaje):
        self.archivo.write(mensaje + "\n")
        self.archivo.flush()   # Fuerza a guardar en disco

# Probando el Singleton
registro1 = Logger() # Se crea la unica instancia SINGLETON.
registro2 = Logger() # Devuelve la misma instancia, sin crear una nueva.

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

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

```
True
```

1. ¿Qué pasaría si eliminamos la verificación if cls.instancia is None en el método ___new_? La clase deja de ser un Singleton. Se crearía una nueva instancia cada vez que se llama al constructor, resultando en múltiples objetos.

2. En la línea print(registro1 is registro2). ¿Qué significa que devuelva True en el contexto del Singleton? Significa que ambas variables referencian el mismo objeto único en la memoria, lo cual es la validación de que el patrón Singleton está funcionando correctamente.

3. ¿Crees que siempre es buena idea usar Singleton para todo lo que es global? Da un ejemplo donde no sería recomendable. No. Su uso excesivo puede: ocultar dependencias, dificultando las pruebas; violar el Principio de Responsabilidad Única (SRP); y causar problemas de concurrencia en entornos multi-hilo.

Ejemplo: Gestión de conexiones a bases de datos. Un Singleton de conexión puede ser rígido. Si necesitas múltiples conexiones o control individual sobre la apertura/cierre, es mejor usar la Inyección de Dependencias o el patrón Factory.