Simulación de acceso concurrente seguro a un archivo de registro

En este ejercicio, vamos a simular varios hilos que intentan escribir en un archivo de registro simultáneamente. Utilizaremos un bloqueo (lock) para asegurar que solo un hilo pueda escribir en el archivo a la vez, evitando condiciones de carrera.

```
import threading
import time
import random

# Definimos el bloqueo global
lock = threading.Lock()

def escribir_en_registro(hilo_id):
    with lock:
        with open("registro.log", "a") as archivo:
              tiempo = time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime())
             mensaje = f"{tiempo} - Hilo {hilo_id} escribiendo
en el registro\n"
            archivo.write(mensaje)
            print(mensaje)
        time.sleep(random.uniform(0.1, 0.5))

def crear_hilos(num_hilos):
    hilos = []
    for i in range(num_hilos):
         hilo = threading.Thread(target=escribir_en_registro,
args=(i,))
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
```

```
        hilo.join()

if __name__ == "__main__":
    crear_hilos(10)
```

Explicación

1. **Importaciones y definición del bloqueo global**

```
import threading
import time
import random

lock = threading.Lock()
```

Importamos los módulos necesarios: threading para manejar los hilos, time para obtener la hora actual y random para generar pausas aleatorias. Definimos un bloqueo global lock que será usado para sincronizar el acceso al archivo de registro.

2. **Función para escribir en el archivo de registro**

```
def escribir_en_registro(hilo_id):
    with lock:
        with open("registro.log", "a") as archivo:
              tiempo = time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime())
            mensaje = f"{tiempo} - Hilo {hilo_id} escribiendo
en el registro\n"
            archivo.write(mensaje)
            print(mensaje)
        time.sleep(random.uniform(0.1, 0.5))
```

La función escribir_en_registro toma un identificador de hilo (hilo_id) como argumento. Utiliza un bloqueo para asegurar que solo un hilo pueda escribir en el archivo a la vez. Dentro del bloqueo, abre el archivo registro.log en modo de adición (a), escribe un mensaje con la hora actual y el identificador del hilo, y luego cierra el archivo. También imprime el mensaje en

la pantalla. Finalmente, el hilo duerme por un período aleatorio entre 0.1 y 0.5 segundos para simular trabajo.

3. **Función para crear y gestionar los hilos**

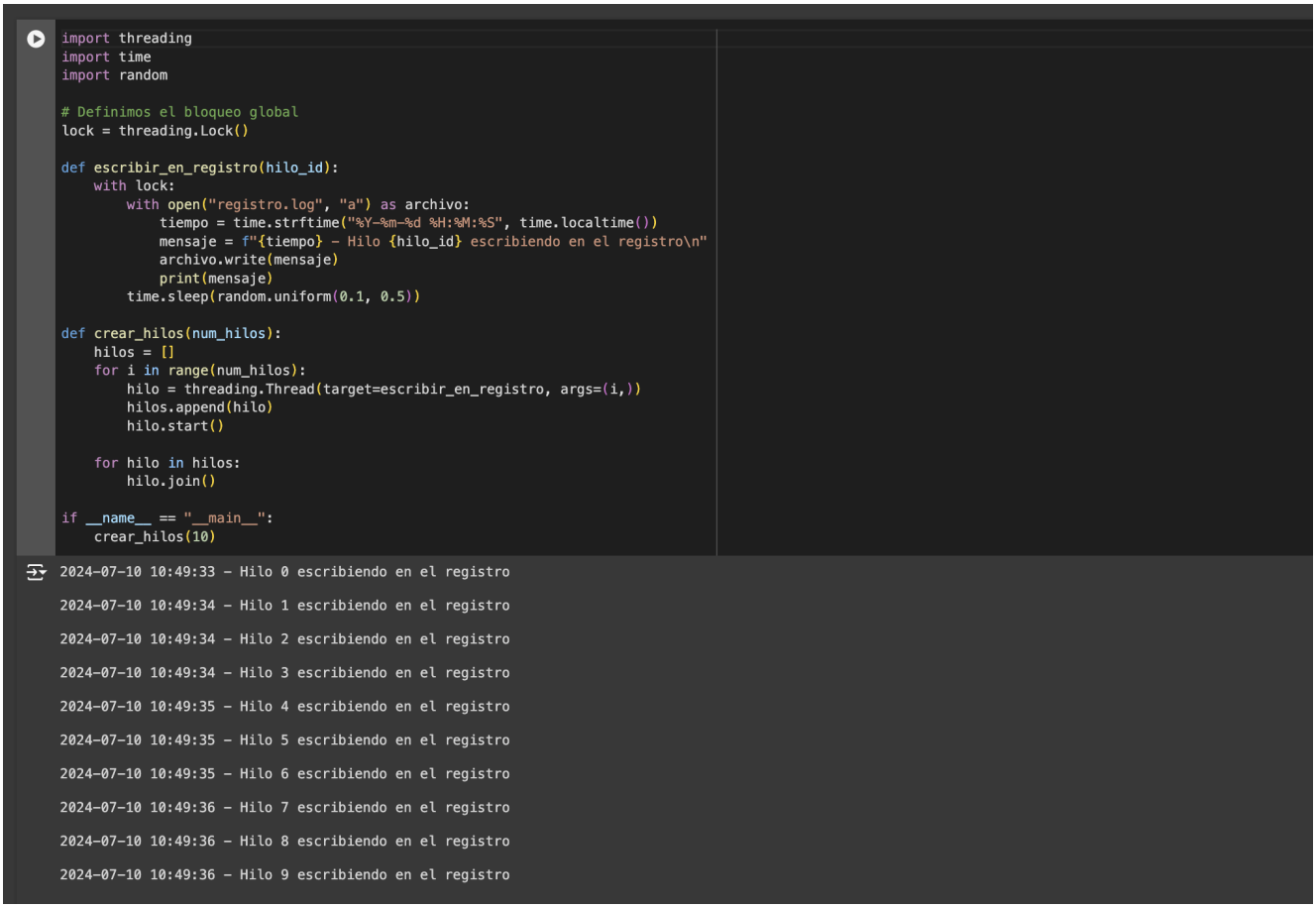
```
def crear_hilos(num_hilos):
    hilos = []
    for i in range(num_hilos):
         hilo = threading.Thread(target=escribir_en_registro,
args=(i,))
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
        hilo.join()
```

La función crear_hilos crea y gestiona un número de hilos especificado por num_hilos. Para cada hilo, crea un objeto Thread que ejecutará la función escribir_en_registro con el identificador del hilo como argumento. Añade cada hilo a una lista y lo inicia. Luego, espera a que todos los hilos terminen utilizando join.

4. **Bloque principal**

```
if __name__ == "__main__":
    crear_hilos(10)
```

El bloque principal llama a la función crear_hilos con 10 hilos. Esto inicia la simulación de acceso concurrente seguro al archivo de registro.



```python
import threading
import time
import random

# Definimos el bloqueo global
lock = threading.Lock()

def escribir_en_registro(hilo_id):
    with lock:
        with open("registro.log", "a") as archivo:
            tiempo = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
            mensaje = f"{tiempo} - Hilo {hilo_id} escribiendo en el registro\n"
            archivo.write(mensaje)
            print(mensaje)
        time.sleep(random.uniform(0.1, 0.5))

def crear_hilos(num_hilos):
    hilos = []
    for i in range(num_hilos):
        hilo = threading.Thread(target=escribir_en_registro, args=(i,))
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
        hilo.join()

if __name__ == "__main__":
    crear_hilos(10)
```

```
2024-07-10 10:49:33 - Hilo 0 escribiendo en el registro

2024-07-10 10:49:34 - Hilo 1 escribiendo en el registro

2024-07-10 10:49:34 - Hilo 2 escribiendo en el registro

2024-07-10 10:49:34 - Hilo 3 escribiendo en el registro

2024-07-10 10:49:35 - Hilo 4 escribiendo en el registro

2024-07-10 10:49:35 - Hilo 5 escribiendo en el registro

2024-07-10 10:49:35 - Hilo 6 escribiendo en el registro

2024-07-10 10:49:36 - Hilo 7 escribiendo en el registro

2024-07-10 10:49:36 - Hilo 8 escribiendo en el registro

2024-07-10 10:49:36 - Hilo 9 escribiendo en el registro
```

Explicación detallada

```
import threading
import time
import random
```

Importaciones: Se importan tres módulos:

- threading: Para manejar la creación y gestión de hilos.
- time: Para trabajar con el tiempo y las fechas.
- random: Para generar números aleatorios, utilizado aquí para crear pausas aleatorias.

```
# Definimos el bloqueo global
lock = threading.Lock()
```

Definición del bloqueo global:

- lock = threading.Lock(): Crea un objeto de bloqueo

(lock) que será utilizado para sincronizar el acceso al recurso compartido (archivo de registro) entre los hilos.

```
def escribir_en_registro(hilo_id):
    with lock:
        with open("registro.log", "a") as archivo:
              tiempo = time.strftime("%Y-%m-%d %H:%M:%S",
time.localtime())
             mensaje = f"{tiempo} - Hilo {hilo_id} escribiendo
en el registro\n"
            archivo.write(mensaje)
            print(mensaje)
        time.sleep(random.uniform(0.1, 0.5))
```

Función escribir_en_registro:

- def escribir_en_registro(hilo_id):: Define una función que toma un identificador de hilo (hilo_id) como argumento.
- with lock:: Utiliza el bloqueo para asegurar que solo un hilo pueda ejecutar el bloque de código dentro del with a la vez.
- with open("registro.log", "a") as archivo:: Abre el archivo registro.log en modo de adición ("a") para que las escrituras se añadan al final del archivo.
- tiempo = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()): Obtiene la fecha y hora actual en un formato legible.
- mensaje = f"{tiempo} - Hilo {hilo_id} escribiendo en el registro\n": Crea un mensaje que incluye la hora actual y el identificador del hilo.
- archivo.write(mensaje): Escribe el mensaje en el archivo de registro.
- print(mensaje): Imprime el mensaje en la salida estándar (pantalla).
- time.sleep(random.uniform(0.1, 0.5)): Pausa el hilo por un período aleatorio entre 0.1 y 0.5 segundos para

```
      simular trabajo.


def crear_hilos(num_hilos):
    hilos = []
    for i in range(num_hilos):
         hilo = threading.Thread(target=escribir_en_registro,
args=(i,))
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
        hilo.join()
```

Función crear_hilos:

- def crear_hilos(num_hilos):: Define una función que toma el número de hilos a crear (num_hilos) como argumento.
- hilos = []: Inicializa una lista vacía para almacenar los hilos.
- for i in range(num_hilos): Itera desde 0 hasta num_hilos-1.
- hilo = threading.Thread(target=escribir_en_registro, args=(i,)): Crea un nuevo hilo que ejecutará la función escribir_en_registro con el identificador del hilo (i) como argumento.
- hilos.append(hilo): Añade el hilo creado a la lista de hilos.
- hilo.start(): Inicia la ejecución del hilo.
- for hilo in hilos: Itera sobre la lista de hilos.
- hilo.join(): Espera a que cada hilo termine su ejecución antes de continuar.

```
if __name__ == "__main__":
    crear_hilos(10)
```

Bloque principal:

- if __name__ == "__main__": Comprueba si el script se está ejecutando directamente (no importado como módulo).

- crear_hilos(10): Llama a la función crear_hilos con 10 hilos, iniciando así la simulación de acceso concurrente seguro al archivo de registro.