Práctica 7 - Tema Threading & Multiprocessing
(Quiz Filósofos - Plazo 23 de marzo)

Tiempo estimado: 60 minutos.
Video base:https://youtu.be/dNaeR3hLWaQ

Threading en Python: Se clarifica que Python admite threading, pero no de manera simultánea debido al Global Interpreter Lock (GIL), que asegura que solo un hilo se ejecute a la vez en un proceso. Esto significa que, aunque múltiples hilos puedan existir, no ejecutan código simultáneamente en un solo proceso, lo cual es crucial para tareas intensivas en CPU pero menos problemático para tareas limitadas por I/O.

Procesos en Python: A diferencia de los hilos, los procesos en Python se ejecutan en espacios de memoria separados, permitiendo la ejecución simultánea real en múltiples núcleos de CPU. Esto se demuestra como ventajoso para tareas intensivas en CPU, donde el multiprocessing puede aprovechar múltiples núcleos para mejorar significativamente el rendimiento en comparación con el threading.

Ejemplos prácticos: Se presentan una serie de experimentos para comparar el rendimiento de threading y multiprocessing en tareas limitadas por CPU e I/O. Estos ejemplos ilustran cómo el multiprocessing supera al threading en tareas intensivas en CPU, logrando un paralelismo real y una mejora en el rendimiento al distribuir la carga de trabajo en varios procesos.

Consideraciones de rendimiento: Aunque el multiprocessing ofrece ventajas significativas para ciertas tareas, también se discuten sus desventajas, como el mayor costo de configuración y la complejidad adicional al manejar la comunicación entre procesos. Se destaca la importancia de elegir la herramienta adecuada (threading vs. multiprocessing) basada en la naturaleza de la tarea a realizar.

Seguridad y sincronización: Se abordan las consideraciones de seguridad y sincronización al usar threading, como la necesidad de bloqueos para proteger los datos compartidos entre hilos. También se explora el uso de colas y tuberías para la comunicación segura entre hilos y procesos.

Este análisis proporciona una comprensión clara de cuándo y cómo utilizar threading y multiprocessing en Python, ofreciendo a los desarrolladores las herramientas necesarias para optimizar sus aplicaciones según las necesidades específicas de rendimiento y concurrencia.

Reto:
Se habilita un cuestionario de 10 preguntas sobre el video.