***RESPUESTAS TP***

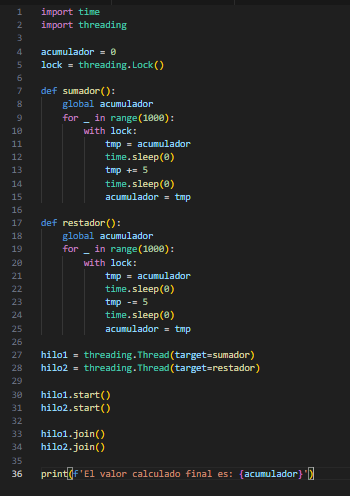
1. Tiempo de ejecución:
   * El código muestra tres tareas (tarea\_1, tarea\_2 y tarea\_3) que se ejecutan secuencialmente.
   * La tarea\_1 realiza un bucle simple sin pausas, la tarea\_2 espera 1 segundo y la tarea\_3 espera 4 segundos.
   * El tiempo de ejecución total es la suma de los tiempos de ejecución de las tres tareas.
2. ¿Qué se puede notar con respecto al tiempo de ejecución? ¿Es predecible?
   * El tiempo de ejecución total es predecible, ya que se compone de tiempos conocidos (1 segundo para tarea\_2 y 4 segundos para tarea\_3) más el tiempo de ejecución del bucle en tarea\_1.
   * Sin embargo, el tiempo de ejecución de tarea\_1 no es predecible debido a la naturaleza del bucle (depende de la velocidad de la máquina y otros factores).
3. Proceso de “máxima velocidad posible”:
   * Un ejemplo de un proceso que depende casi exclusivamente de la velocidad de la máquina es ordenar una lista.
   * Algoritmos de ordenamiento como QuickSort, MergeSort o TimSort son procesos que se ejecutan lo más rápido posible y su rendimiento está fuertemente influenciado por la velocidad de procesamiento.
4. Proceso de “velocidad de respuesta no dependiente de la velocidad de procesamiento” o impredecible:
   * Un ejemplo de proceso con velocidad de respuesta no dependiente de la velocidad de procesamiento es leer un archivo externo.
   * La velocidad de lectura de un archivo depende más de factores externos como la velocidad del disco duro, la latencia de red o la disponibilidad del archivo.

Estos factores pueden variar y no están directamente relacionados con la velocidad de la CPU.

1. Tiempo de ejecución y mejora con hilos:
   * El código utiliza hilos (threads) para ejecutar tres tareas (tarea\_1, tarea\_2 y tarea\_3) de forma concurrente.
   * Cada hilo ejecuta una tarea específica al mismo tiempo.
   * ¿Qué se puede notar con respecto al tiempo de ejecución? ¿Se mejoró el tiempo de respuesta con respecto al mismo programa sin hilos?
     + El tiempo total de ejecución se reducirá en comparación con la ejecución secuencial sin hilos.
     + Las tareas se ejecutan en paralelo, lo que permite aprovechar mejor los recursos de la CPU.
     + Sin embargo, la mejora en el tiempo de respuesta depende de la cantidad de núcleos de CPU disponibles y la naturaleza de las tareas.
2. Orden de ejecución de las funciones:
   * Los hilos se ejecutan de manera concurrente, pero no necesariamente en el orden establecido.
   * No hay garantía de que tarea\_1 termine antes que tarea\_2 o tarea\_3.
   * El orden de finalización puede variar según la asignación de recursos y la planificación del sistema operativo.
3. Escenario real donde el multi-hilado mejora el tiempo de respuesta:
   * Un ejemplo común es la carga de una página web en un navegador:
     + Cuando se carga una página web, el navegador realiza múltiples solicitudes (por ejemplo, para obtener HTML, CSS, imágenes, scripts, etc.).
     + Si estas solicitudes se manejan en hilos separados, el navegador puede descargar recursos en paralelo, lo que acelera la carga de la página.
     + Sin hilos, el navegador tendría que esperar a que se complete una solicitud antes de iniciar la siguiente, lo que sería menos eficiente.

En resumen, el uso de hilos puede mejorar significativamente el tiempo de respuesta en escenarios donde hay tareas concurrentes y recursos disponibles para ejecutarlas en paralelo.

1. Tiempo de ejecución:
   * El código utiliza dos hilos (hilo1 y hilo2) para ejecutar las funciones sumador y restador.
   * Cada función realiza operaciones en una variable compartida (acumulador).
   * ¿Qué se puede notar con respecto al tiempo de ejecución?
     + El tiempo de ejecución total es menor que si se ejecutaran las funciones secuencialmente.
     + Sin embargo, debido a la sincronización entre los hilos, no se garantiza que el tiempo total sea exactamente la mitad del tiempo secuencial.
2. Valor final del acumulador:
   * El valor final del acumulador no es predecible.
   * Esto se debe a una condición de carrera entre los hilos.
   * Ambos hilos leen y actualizan el valor de acumulador de manera concurrente, lo que puede llevar a resultados inesperados.
3. ¿Por qué sucede esto?:
   * La condición de carrera ocurre cuando dos o más hilos acceden a una variable compartida al mismo tiempo.
   * En este caso, ambos hilos leen el valor actual de acumulador, realizan una operación (suma o resta) y actualizan el valor.
   * Si los hilos se superponen en sus operaciones, pueden perder actualizaciones y producir resultados incorrectos.
4. Corrección de la condición de carrera:
   * Para evitar la condición de carrera, puedes utilizar bloqueos (locks) para sincronizar el acceso a la variable compartida.
   * Un bloqueo asegura que solo un hilo pueda acceder a la variable a la vez.
   * Aquí hay una versión corregida del código:



En el código proporcionado, se utilizan dos hilos (hilo1 y hilo2) para ejecutar las funciones sumadoras y restador. Ambas funciones modifican una variable compartida llamada acumulador. Vamos a analizar tus preguntas:

1. Valor final del acumulador:
   * Debido a la condición de carrera, el valor final del acumulador no es predecible.
   * La condición de carrera ocurre cuando dos o más hilos acceden y modifican una variable compartida al mismo tiempo.
   * En este caso, ambos hilos leen y actualizan el valor de acumulador de manera concurrente, lo que puede llevar a resultados inesperados.
2. Tiempo de ejecución:
   * El uso de hilos permite que las funciones sumador y restador se ejecuten en paralelo.
   * Sin embargo, debido a las operaciones de bloqueo (time.sleep(0)), el tiempo de ejecución total no necesariamente será la mitad del tiempo secuencial.
   * La sincronización entre los hilos (mediante el uso del bloqueo) introduce cierta latencia.

En resumen, el valor final del acumulador es impredecible debido a la condición de carrera, y el tiempo de ejecución puede variar según la planificación del sistema operativo y las operaciones de bloqueo.