FP.057 - (P) Seguridad, procesos y zócalos
AA2. Programación de hilos

Alumno: JAVIER DE LA VEGA EDER

Consultor: Lean Zabala Iglesias
CFGS en Desarrollo de Aplicaciones Multiplataforma(ICB02)
FP UOC - Fundació Jesuïtes Educació

Índice

Conceptos básicos del funcionamiento de hilos de ejecución 3

Ejecutar el programa "ACT2_EJ1_v2.py" y explicar el funcionamiento del mismo 4

Modificar el programa para que se ejecuten cada una de las funciones definidas 6

Conceptos básicos del funcionamiento de hilos de ejecución

En programación y sistemas informáticos, un hilo de ejecución (thread) se refiere a una unidad individual de ejecución dentro de un proceso. Los procesos son programas en ejecución que pueden tener múltiples hilos de ejecución.

Conceptos clave para entender cómo funcionan los hilos:

Multitarea: Los hilos permiten que un programa realice múltiples tareas en paralelo o aparentemente en paralelo. Esto significa que un programa puede ejecutar varias secuencias de instrucciones simultáneamente en lugar de una a la vez.

Proceso vs. Hilo: Un proceso es una instancia de un programa en ejecución que tiene su propio espacio de memoria y recursos. En cambio, un hilo comparte la memoria y los recursos con otros hilos dentro del mismo proceso. Los hilos son más ligeros en términos de recursos que los procesos, lo que los hace adecuados para realizar tareas concurrentes.

Concurrencia: Los hilos se utilizan para lograr concurrencia, lo que significa que múltiples tareas pueden avanzar al mismo tiempo. La concurrencia es útil para mejorar el rendimiento y la capacidad de respuesta de una aplicación.

Sincronización: La sincronización es un desafío importante cuando se trabaja con hilos. Puesto que los hilos pueden acceder a recursos compartidos, es necesario coordinar y sincronizar sus acciones para evitar problemas como condiciones de carrera y acceso simultáneo a datos compartidos.

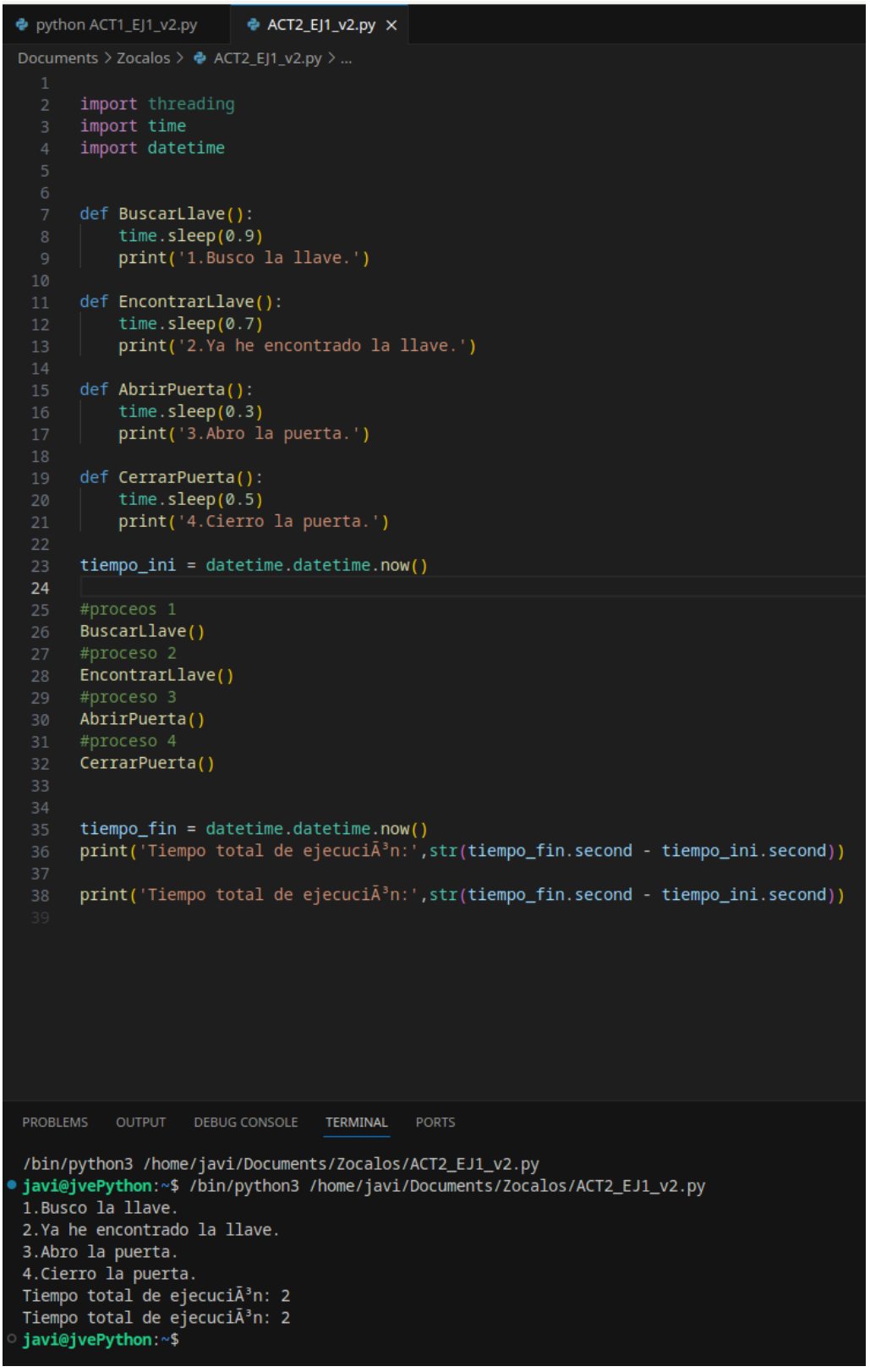
Modelo de Hilos: Los sistemas operativos proporcionan un modelo de hilos que permite a los desarrolladores crear, iniciar, detener y coordinar hilos. En Python, el módulo threading proporciona herramientas para trabajar con hilos.

Ventajas de los Hilos: Los hilos son beneficiosos en situaciones en las que se necesita realizar múltiples tareas concurrentemente, como aplicaciones de servidor, interfaces de usuario interactivas y procesamiento en segundo plano.

Desafíos de los Hilos: La programación con hilos puede ser compleja debido a la necesidad de gestionar la sincronización y evitar problemas de concurrencia. Los errores en la sincronización pueden llevar a resultados inesperados y comportamientos no deterministas.

Ejecutar el programa "ACT2_EJ1_v2.py" y explicar el funcionamiento del programa y el tiempo de ejecución

El programa "**ACT2_EJ1_v2.py**" es un ejemplo que ilustra la ejecución secuencial de varias funciones, que simulan pasos en una tarea, y mide el tiempo total de ejecución. Aquí se describen los aspectos clave del programa:



Funciones Secuenciales: El programa define cuatro funciones: **BuscarLlave**, **EncontrarLlave**, **AbrirPuerta**, y **CerrarPuerta**. Cada una de estas funciones representa una etapa en un proceso ficticio. Estas funciones se ejecutan una tras otra en un orden fijo.

Simulación de Retardos: Dentro de cada función, se utiliza la función time.sleep para simular un retraso en la ejecución. Cada función espera un período de tiempo específico antes de imprimir un mensaje en la consola. Esto simula la idea de que cada etapa de la tarea lleva un tiempo diferente en completarse.

Medición del Tiempo de Ejecución: El programa utiliza la biblioteca datetime para registrar el momento en que comienza y termina su ejecución. Al final de la ejecución, calcula el tiempo total de ejecución restando el segundo inicial del segundo final. Sin embargo, es importante notar que esta forma de medir el tiempo puede no ser precisa debido a las limitaciones de precisión de la función datetime. Para mediciones de tiempo más precisas, se pueden utilizar otras herramientas como el módulo time o la biblioteca timeit.

Orden Secuencial: En la versión original del programa, las funciones se ejecutan en un orden fijo: primero se busca la llave, luego se encuentra la llave, se abre la puerta y, finalmente, se cierra la puerta. Este enfoque secuencial significa que cada etapa debe completarse antes de que comience la siguiente.

Finalización de Tareas: Una vez que todas las funciones han terminado de ejecutarse, el programa muestra el tiempo total de ejecución en segundos.

Modificar el programa para que se ejecuten cada una de las funciones definidas en diferentes hilos de ejecución y en el orden correcto. Puedes utilizar la función "threading.Thread".

```python
import threading
import time
import datetime

def BuscarLlave():
    time.sleep(0.9)
    print('1.Busco la llave.')

def EncontrarLlave():
    time.sleep(0.7)
    print('2.Ya he encontrado la llave.')

def AbrirPuerta():
    time.sleep(0.3)
    print('3.Abro la puerta.')

def CerrarPuerta():
    time.sleep(0.5)
    print('4.Cierro la puerta.')

tiempo_ini = datetime.datetime.now()

# Crear los hilos
thread1 = threading.Thread(target=BuscarLlave)
thread2 = threading.Thread(target=EncontrarLlave)
thread3 = threading.Thread(target=AbrirPuerta)
thread4 = threading.Thread(target=CerrarPuerta)

# Iniciar y esperar a que todos los hilos terminen
thread1.start()
thread1.join()

thread2.start()
thread2.join()

thread3.start()
thread3.join()

thread4.start()
thread4.join()

tiempo_fin = datetime.datetime.now()
print('Tiempo total de ejecución:', str(tiempo_fin.second - tiempo_ini.second))
```

En este programa, he creado cuatro hilos (uno por cada función) y los he iniciado utilizando **start**. Luego, he utilizado **join** para esperar a que todos los hilos terminen antes de calcular el tiempo de ejecución. Esto permitirá que las funciones se ejecuten en paralelo en el orden correcto.

En el programa modificado, las funciones se ejecutarán en el orden en el que se iniciaron los **hilos: 1, 2, 3 y 4**. Tal y como indica el enunciado, utilizamos las funciones "**start**" y "**join**".

```
javi@jvePython:~$ /bin/python3 /home/javi/Documents/Zocalos/ACT2_EJ1_v2.py
1.Busco la llave.
2.Ya he encontrado la llave.
3.Abro la puerta.
4.Cierro la puerta.
Tiempo total de ejecuciÃ³n: 2
javi@jvePython:~$ /bin/python3 /home/javi/Documents/Zocalos/ACT2_EJ1_v2-mod.py
1.Busco la llave.
2.Ya he encontrado la llave.
3.Abro la puerta.
4.Cierro la puerta.
Tiempo total de ejecución: 3
javi@jvePython:~$
```

El programa modificado tarda 3 segundos mientras que el original tarda 2 segundos porque utiliza múltiples hilos de ejecución para realizar las tareas concurrentemente.

Se procedió a realizar ajustes en el código para su ejecución a través de hilos, aprovechando la utilidad proporcionada por la función threading, la cual simplifica la creación y administración de hilos. Con la introducción de **threading**.**Thread**, se han generado hilos independientes que **se ponen en marcha** mediante la **función .start()**, y se controla su **finalización** con **.join()**. El orden de conclusión se determina según el tiempo de espera asignado en cada función.

BuscarLlave() espera un tiempo de (0.9) antes de finalizar.
EncontrarLlave() espera un tiempo de (0.7) antes de finalizar.
AbrirPuerta() espera un tiempo de (0.3) antes de finalizar.
CerrarPuerta() espera un tiempo de (0.5) antes de finalizar.

Este enfoque **garantiza la ejecución de las funciones a través de hilos**, iniciando cada una con start() y asegurando su finalización con join(). Se controla la secuencia de ejecución de cada función mediante el tiempo total de ejecución, que se ha establecido en 3 segundos.