An Introduction to Scaling Distributed Python Applications.

- **Nombre:** Jesus Eduardo Jimenez Covarrubias
- **Materia:** Computación Tolerante a Fallas
- **Profesor:** Michel Emanuel Lopez Franco
- **NRC:** 179961
- **Sección:** D06
- **Ciclo:** 2025-A

Instrucciones:

Generar un programa utilizando hilos, procesos, demonios o concurrencia.

https://www.educative.io/blog/scaling-in-python

https://statusneo.com/concurrency-in-python-threading-processes-and-asyncio/

Desarrollo:

Código (Python):

```
import threading
import multiprocessing
import time


# ================== Sección de Hilos (I/O-bound) ==================
def tarea_io(nombre):
    print(f'Hilo {nombre}: Iniciando tarea I/O')
    time.sleep(2)  # Simula operación I/O (ej. lectura de archivo/red)
    print(f'Hilo {nombre}: Tarea I/O completada')


def ejecutar_hilos():
    hilos = []
    for i in range(3):
        hilo = threading.Thread(target=tarea_io, args=(i,))
        hilos.append(hilo)
        hilo.start()

    for hilo in hilos:
        hilo.join()


# ============== Sección de Procesos (CPU-bound) ==============
def tarea_cpu(numero):
    print(f'Proceso {numero}: Calculando...')
    resultado = sum(i * i for i in range(10 ** 6))  # Operación intensiva
de CPU
    print(f'Proceso {numero}: Resultado = {resultado}')


def ejecutar_procesos():
    procesos = []
    for i in range(3):
        proceso = multiprocessing.Process(target=tarea_cpu, args=(i,))
        procesos.append(proceso)
        proceso.start()

    for proceso in procesos:
        proceso.join()


# ================== Demonio (Tarea en segundo plano) ==================
```

```
def demonio():
    while True:
        print("Demonio: Ejecutando tarea en segundo plano...")
        time.sleep(1)


# ================= Ejecución principal =================
if __name__ == "__main__":
    print("\nEjemplo completo de concurrencia en Python\n")

    # Iniciar demonio (se detendrá automáticamente al finalizar el
programa)
    d = threading.Thread(target=demonio)
    d.daemon = True
    d.start()

    print("\n=== Ejecutando hilos para tareas I/O ===")
    inicio = time.time()
    ejecutar_hilos()
    print(f'Tiempo hilos: {time.time() - inicio:.2f}s')

    print("\n=== Ejecutando procesos para tareas CPU ===")
    inicio = time.time()
    ejecutar_procesos()
    print(f'Tiempo procesos: {time.time() - inicio:.2f}s')

    print("\nPrograma principal terminado. El demonio será eliminado
automáticamente.")
```

Funcionamiento del código:

Importación de módulos:

import threading: Importa el módulo de threading para manejar hilos.

import multiprocessing: Importa el módulo de multiprocessing para manejar procesos.

import time: Importa el módulo de time para manejar funciones relacionadas con el tiempo.

Sección de Hilos (I/O-bound):

Esta sección se enfoca en la ejecución de tareas I/O (input/output) que generalmente se benefician del uso de hilos.

def tarea_io(nombre): Define una función que simula una tarea I/O, como lectura de archivos o redes.

def ejecutar_hilos(): Define una función para ejecutar múltiples hilos.

Crea una lista de hilos.

Inicia 3 hilos que ejecutan la función tarea_io.

Usa join para esperar que todos los hilos terminen.

Sección de Procesos (CPU-bound):

Esta sección se enfoca en la ejecución de tareas intensivas de CPU que se benefician del uso de procesos separados.

def tarea_cpu(numero): Define una función que simula una tarea intensiva de CPU

def ejecutar_procesos(): Define una función para ejecutar múltiples procesos.

Crea una lista de procesos.

Inicia 3 procesos que ejecutan la función tarea_cpu.

Usa join para esperar que todos los procesos terminen.

Demonio (Tarea en segundo plano):

Define una tarea que se ejecuta continuamente en segundo plano.

def demonio(): Define una función que imprime un mensaje cada segundo.

Inicia un hilo que ejecuta la función demonio como un hilo demonio. Los hilos demonio se detienen automáticamente cuando el programa principal termina.

Ejecución principal:

Verifica si el script se ejecuta como el programa principal.

Inicia el hilo demonio.

Ejecuta los hilos para las tareas I/O y mide el tiempo de ejecución.

Ejecuta los procesos para las tareas CPU y mide el tiempo de ejecución.

Imprime un mensaje final indicando que el programa principal ha terminado.

Este código proporciona un ejemplo de cómo manejar tareas I/O y tareas intensivas de CPU usando hilos y procesos en Python. Además, demuestra cómo

```
C:\Users\jusue\.conda\envs\Ejercico_5.1\python.exe "G:\Septimo Semeste\COMPUTACION TOLERANTE A FALLAS\Ejercicios\Ejercico 5.1\Actividad_5.py"

Ejemplo completo de concurrencia en Python

Demonio: Ejecutando tarea en segundo plano...
=== Ejecutando hilos para tareas I/O ===

Hilo 0: Iniciando tarea I/O
Hilo 1: Iniciando tarea I/O
Hilo 2: Iniciando tarea I/O
Demonio: Ejecutando tarea en segundo plano...
Hilo 0: Tarea I/O completada
Demonio: Ejecutando tarea en segundo plano...
Hilo 1: Tarea I/O completada
Hilo 2: Tarea I/O completada
Tiempo hilos: 2.00s

=== Ejecutando procesos para tareas CPU ===
Proceso 2: Calculando...
Proceso 1: Calculando...
Proceso 0: Calculando...
Proceso 2: Resultado = 333332833333500000
Demonio: Ejecutando tarea en segundo plano...
Proceso 1: Resultado = 333332833333500000
Proceso 0: Resultado = 333332833333500000
Tiempo procesos: 1.05s

Programa principal terminado. El demonio será eliminado automáticamente.

Process finished with exit code 0
```

ejecutar una tarea en segundo plano utilizando un hilo demonio. ¿Hay alguna parte específica que te gustaría que profundizara más?

Inicia el programa principal y se imprime el mensaje: "Ejemplo completo de concurrencia en Python".

Inicia el demonio, que es un hilo en segundo plano que ejecuta la función demonio:

Demonio: Ejecutando tarea en segundo plano...

Ejecuta los hilos para tareas I/O:

Se imprime el mensaje: "=== Ejecutando hilos para tareas I/O ===".

Se inician tres hilos (Hilo 0, Hilo 1 y Hilo 2) que ejecutan la función tarea_io. Cada hilo imprime un mensaje al iniciar y después de 2 segundos imprime un mensaje indicando que la tarea I/O se completó.

Hilo 0: Iniciando tarea I/O

Hilo 1: Iniciando tarea I/O

Hilo 2: Iniciando tarea I/O

Demonio: Ejecutando tarea en segundo plano...

Hilo 0: Tarea I/O completada

Demonio: Ejecutando tarea en segundo plano...

Hilo 1: Tarea I/O completada

Hilo 2: Tarea I/O completada

El tiempo total de ejecución de los hilos es aproximadamente 2 segundos debido a la operación de time.sleep(2).

Ejecuta los procesos para tareas CPU:

Se imprime el mensaje: "=== Ejecutando procesos para tareas CPU ===".

Se inician tres procesos (Proceso 0, Proceso 1 y Proceso 2) que ejecutan la función tarea_cpu. Cada proceso imprime un mensaje al iniciar y después de calcular el resultado de la operación intensiva de CPU, imprime el resultado.

Proceso 2: Calculando...

Proceso 1: Calculando...

Proceso 0: Calculando...

Proceso 2: Resultado = 333332833333500000

Demonio: Ejecutando tarea en segundo plano...

Proceso 1: Resultado = 333332833333500000

Proceso 0: Resultado = 333332833333500000

El tiempo total de ejecución de los procesos es aproximadamente 1.05 segundos.

Finaliza el programa principal y se imprime el mensaje final:

Programa principal terminado. El demonio será eliminado automáticamente.

Salida de proceso:

Process finished with exit code 0

Esto demuestra cómo puedes manejar tareas de I/O y de CPU en paralelo utilizando hilos y procesos en Python. También se muestra cómo un hilo demonio puede ejecutar tareas en segundo plano mientras el programa principal está activo.