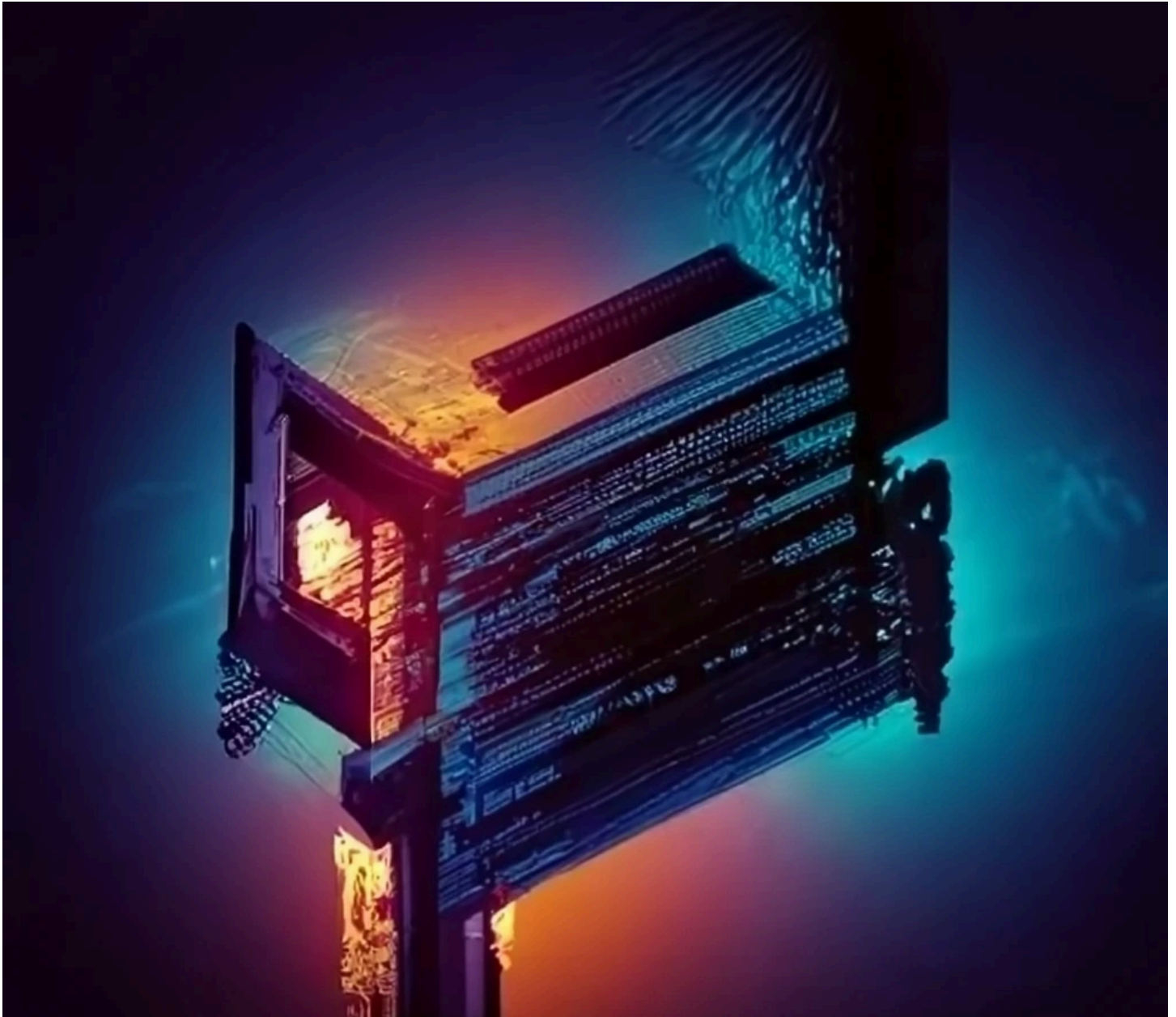




Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías



Sección: D06 Profesor: MICHEL EMANUEL LOPEZ FRANCO
Tema: An Introduction to Scaling Distributed Python Applications
Ciclo: 2024A

Introducción

se observa la utilización y definición de lo que son los demonios en la programación para tolerante a fallas en el cual la verificación es funcional para que estas mismas sean de manera controlada y escalables.

Contenido

se muestra en continuación unos ejemplos de lo que ya es existente y código simple en el cual veremos las funciones en python solo lo que es código y salidas para comprensión de cada uno.

Example

```
import threading

def worker():
    print(f"Thread {threading.current_thread().name} is running")

threads = []
for i in range(5):
    t = threading.Thread(target=worker)
    t.start()
    threads.append(t)



for t in threads:
    t.join()

print("All threads have completed")
```

Output

```
Thread Thread-1 (worker) is running
Thread Thread-2 (worker) is running
Thread Thread-3 (worker) is running
Thread Thread-4 (worker) is running
Thread Thread-5 (worker) is running
All threads have completed
```

Codigo de otro ejemplo realizado aparte

Tolerante a fallas >  aplicaciones distribuidas de python.py >  ejecutar_concurrencia

```
1  import threading
2  import multiprocessing
3  import concurrent.futures
4
5  # Función para calcular la suma de los cuadrados de los primeros N números naturales
6  def calcular_suma_cuadrados(N):
7      suma = 0
8      for i in range(1, N + 1):
9          suma += i ** 2
10     print(f"La suma de los cuadrados de los primeros {N} números es: {suma}")
11
12     # Función que se ejecutará en un hilo
13     def ejecutar_hilo(N):
14         threading.current_thread().name = f"Hilo - N={N}"
15         calcular_suma_cuadrados(N)
16
17     # Función que se ejecutará en un proceso
18     def ejecutar_proceso(N):
19         multiprocessing.current_process().name = f"Proceso - N={N}"
20         calcular_suma_cuadrados(N)
21
22     # Función que se ejecutará en concurrencia
23     def ejecutar_concurrencia(N):
24         with concurrent.futures.ThreadPoolExecutor() as executor:
25             for i in range(N):
26
27
28     if __name__ == "__main__":
29         N = 5 # Número de iteraciones para calcular la suma de los cuadrados
30
31         # Ejecución en un hilo
32         hilo = threading.Thread(target=ejecutar_hilo, args=(N,))
33         hilo.start()
34         hilo.join()
35
36         # Ejecución en un proceso
37         proceso = multiprocessing.Process(target=ejecutar_proceso, args=(N,))
38         proceso.start()
39         proceso.join()
40
41         # Ejecución en concurrencia
42         ejecutar_concurrencia(N)
```

Salidas previstas.

```
exico/Tolerante a fallas/aplicaciones distribuidas de python.py"
La suma de los cuadrados de los primeros 5 números es: 55
La suma de los cuadrados de los primeros 5 números es: 55
La suma de los cuadrados de los primeros 1 números es: 1
La suma de los cuadrados de los primeros 2 números es: 5
La suma de los cuadrados de los primeros 3 números es: 14
La suma de los cuadrados de los primeros 4 números es: 30
La suma de los cuadrados de los primeros 5 números es: 55
```

Referencias

- *An introduction to scaling distributed Python applications.* (s/f). Educative. Recuperado el 16 de febrero de 2024, de <https://www.educative.io/blog/scaling-in-python>
- Dixit, P. (2023, mayo 3). StatusNeo. *StatusNeo - Cloud Native Technology Services & Consulting.*
<https://statusneo.com/concurrency-in-python-threading-processes-and-asyncio/>
-