Summary of Threading & Async & Await

Threading (Multithreading)

Definition: A technique for running multiple tasks concurrently within the same process using multiple threads.

When to use? When executing CPU-intensive tasks (e.g., complex calculations) on a multi-core processor.

Disadvantages:
Managing threads is complex (Race Conditions, Deadlocks).
High resource consumption due to multiple threads

```python
def print_numbers():
    for i in range(5):
        print(i)

# Creating and starting a new thread
thread = threading.Thread(target=print_numbers)
thread.start()

# The main thread continues execution
print("Main thread does not wait for the new thread")
```

Async & Await (Asynchronous Programming)

Definition: Asynchronous programming allows tasks to run independently without blocking execution.

When to use? When handling slow I/O operations (e.g., web requests, file reading) without freezing the program.

How it works?

- **async defines an asynchronous function (Coroutine).**
- **await pauses execution until the awaited task completes but does not block other tasks**

```python
import asyncio

async def say_hello():
    print("Hello")
    await asyncio.sleep(2)  # Waits without blocking execution
    print("After 2 seconds")

async def main():
    await asyncio.gather(say_hello(), say_hello())  # Run two functions concurrently

asyncio.run(main())
```

by Ali Hatem
cis team