**Module 16: Threads ESSENTIAL**

* Class and Threads
* A thread is a lightweight process. In Python, threads allow a program to run multiple tasks seemingly at the same time within the same process.
* Threading module provides Thread class to create and manage threads.

Example:-

import threading

def greet():

    print("Hello from a thread!")

# Create thread

t = threading.Thread(target=greet)

# Start thread

t.start()

# Wait until thread finishes

t.join()

print("Main program ends.")

* Multi-threading

Running multiple threads concurrently to perform different tasks at the same time. Useful for I/O-bound tasks (like downloading files, reading databases, network calls).

Example:-

import threading

import time

def task(name):

print(f"Task {name} started")

time.sleep(2)

print(f"Task {name} finished")

# Create multiple threads

t1 = threading.Thread(target=task, args=("A",))

t2 = threading.Thread(target=task, args=("B",))

t1.start()

t2.start()

t1.join()

t2.join()

print("All tasks done.")

* Synchronization  
  When multiple threads access shared resources (like variables, files, databases), synchronization is required to avoid conflicts (race conditions).
* Python provides Lock from threading module.

Example:-

import threading

lock = threading.Lock()

counter = 0

def increment():

global counter

for \_ in range(1000):

lock.acquire()

counter += 1

lock.release()

threads = []

for \_ in range(5):

t = threading.Thread(target=increment)

threads.append(t)

t.start()

for t in threads:

t.join()

print("Final counter value:", counter)

* Thread Life Cycle
* **States**:
  1. **New** – Thread created but not started.
  2. **Runnable** – Thread ready to run (after start() call).
  3. **Running** – Thread currently executing.
  4. **Waiting/Blocked** – Thread waiting for resources or synchronization.
  5. **Terminated/Dead** – Thread execution finished.

Example to show life cycle:

import threading

import time

def worker():

print("Thread is running...")

time.sleep(3)

print("Thread finished.")

t = threading.Thread(target=worker)

print("Thread created (New state)")

t.start()

print("Thread started (Runnable -> Running)")

t.join()

print("Thread terminated (Dead state)")

* Use Cases of Threads

Threads are mainly used for concurrent execution of tasks.

* **Common Use Cases**:
  + Downloading multiple files at once.
  + Handling multiple client requests in a server.
  + Performing background tasks (logging, monitoring).
  + GUI applications where UI remains responsive.
  + Running timers or schedulers.

Example (Downloader simulation):-

import threading

import time

def download\_file(file\_name):

print(f"Downloading {file\_name}...")

time.sleep(2)

print(f"{file\_name} downloaded.")

files = ["file1.pdf", "file2.mp3", "file3.jpg"]

threads = []

for f in files:

t = threading.Thread(target=download\_file, args=(f,))

threads.append(t)

t.start()

for t in threads:

t.join()

print("All files downloaded successfully.")

**Real-Life Example of Multi-threaded File Downloading**

Imagine you are downloading **multiple files from the internet** (like PDFs, songs, or images).

* If you do it **without threads**, the program will download **one file at a time** (very slow).
* If you use **multi-threading**, all files start downloading **simultaneously**, saving time.

It’s like having **4 people in your family** download 4 different files at the same time, instead of one person doing all downloads one by one.

import threading

import requests

def download\_file(url, filename):

print(f" Downloading: {filename}")

response = requests.get(url)

with open(filename, "wb") as f:

f.write(response.content)

print(f"✅ Finished: {filename}")

# URLs of real files

files = {

"Resume.pdf": "https://www.w3.org/WAI/ER/tests/xhtml/testfiles/resources/pdf/dummy.pdf",

"Song.mp3": "https://file-examples.com/storage/fe8e6a4e9d6b5d3/example.mp3",

"HolidayPhoto.jpg": "https://www.w3.org/People/mimasa/test/imgformat/img/w3c\_home.jpg",

"Presentation.pptx": "https://file-examples.com/storage/fe8e6a4e9d6b5d3/example.pptx"

}

threads = []

for filename, url in files.items():

t = threading.Thread(target=download\_file, args=(url, filename))

threads.append(t)

t.start()

for t in threads:

t.join()

print("🎉 All real files downloaded successfully!")