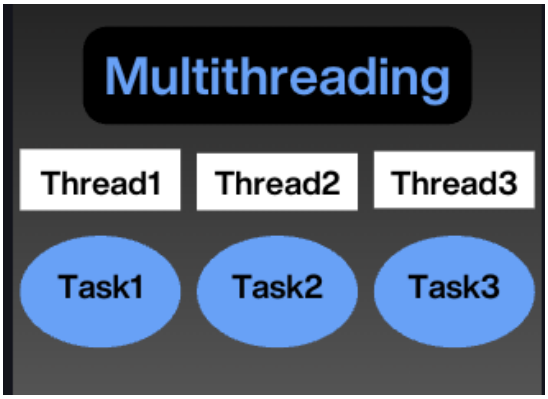
learnwithbhawana

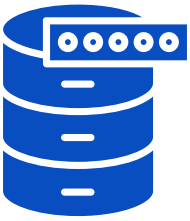
Python Multithreading for Beginners

Introduction

Multithreading is a programming technique where multiple threads are spawned by a process to do different tasks concurrently.

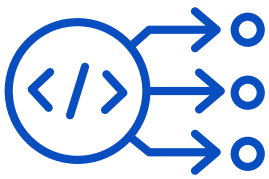


This is especially useful when you have I/O-bound tasks like downloading files or reading from a database.



Key Concepts

Thread: A lightweight subprocess, the smallest unit of processing.



GIL (Global Interpreter Lock): In CPython, GIL allows only one thread to execute at a time even on multi-core processors.

I/O-bound vs CPU-bound: Multithreading is best for I/O-bound tasks.

Use the '***threading***' module in Python to create and manage threads.

Basic Example with Explanation

Code:

```
import threading
import time
def task1():
    for i in range(3):
        print("Task 1 running. ")
        time.sleep(1)

def task2():
    for i in range(3):
        print("Task 2 running. ")
        time.sleep(1)

t1 = threading.Thread(target=task1)
t2 = threading.Thread(target=task2)
t1.start()
t2.start()
t1.join()
t2.join()

print("Both tasks completed!")
```

```
In [14]: import threading
         import time

         def task1():
             for i in range(3):
                 print("Task 1 running. ")
                 time.sleep(1)

         def task2():
             for i in range(3):
                 print("Task 2 running. ")
                 time.sleep(1)

         t1 = threading.Thread(target=task1)
         t2 = threading.Thread(target=task2)

         t1.start()
         t2.start()

         t1.join()
         t2.join()

         print("Both tasks completed!")

         Task 1 running.
         Task 2 running.
         Task 1 running.
         Task 2 running.
         Task 2 running. Task 1 running.

         Both tasks completed!
```

Explanation:

- We define two functions: task1 and task2.
- Create two threads using threading.Thread.
- Start the threads using .start().
- Use .join() to wait for both threads to finish.
- This allows both tasks to run concurrently.

Common Interview Questions and Answers

Q1: What is multithreading in Python?

A1: It allows concurrent execution of more than one part of a program to maximize CPU utilization.

Q2: Does Python support true multithreading?

A2: CPython does not allow true parallel execution of threads due to GIL, but it's useful for I/O-bound tasks.

Q3: Difference between thread and process?

A3: A thread is a subset of a process. Threads share memory space; processes do not.

Q4: What is GIL?

A4: Global Interpreter Lock ensures only one thread executes at a time in CPython.

Q5: When should you use multithreading?

A5: When dealing with I/O-bound tasks like network calls, file I/O, etc.

THE END