Q1. What is multiprocessing in Python? Why is it useful?

Multiprocessing is a technique in which **multiple processes run in parallel**, each having its **own memory space and CPU execution**.
Python uses the `multiprocessing` **module** to implement this.

Why multiprocessing is useful:

- Utilizes **multiple CPU cores**
- Bypasses the **Global Interpreter Lock (GIL)**
- Best suited for **CPU-bound tasks**
- Improves performance for heavy computations

Example use cases:

- Image processing
- Scientific computations
- Data analysis
- Machine learning tasks

Q2. Differences between Multiprocessing and Multithreading

Multiprocessing	Multithreading
Uses multiple processes	Uses multiple threads
Each process has its own memory	Threads share the same memory
No GIL limitation	Affected by GIL
Best for CPU-bound tasks	Best for I/O-bound tasks

Higher memory usage	Lower memory usage
Safer (no shared memory issues)	Risk of race conditions

Q3. Write a Python code to create a process using the multiprocessing module

```
import multiprocessing

def task():
    print("This is a child process")

if __name__ == "__main__":
    p = multiprocessing.Process(target=task)
    p.start()
    p.join()
    print("Main process finished")
```

Q4. What is a multiprocessing pool in Python? Why is it used?

A **multiprocessing pool** is a collection of **worker processes** that execute tasks **in parallel**.

Why it is used:

- Manages multiple worker processes efficiently
- Reduces overhead of creating processes repeatedly
- Simplifies parallel execution of functions
- Automatically distributes tasks among processes

Q5. How can we create a pool of worker processes in Python?

We use the `Pool` **class** from the `multiprocessing` module.

Example:

```
from multiprocessing import Pool

def square(n):
    return n * n

if __name__ == "__main__":
    with Pool(4) as p:
        result = p.map(square, [1, 2, 3, 4])
    print(result)
```

Q6. Python program to create 4 processes

Each process prints a different number

```
import multiprocessing

def print_number(num):
    print("Process number:", num)

if __name__ == "__main__":
    processes = []

    for i in range(1, 5):
        p = multiprocessing.Process(target=print_number, args=(i,))
        processes.append(p)
        p.start()

    for p in processes:
        p.join()
    print("All processes completed")
```
