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

In [1]:
```
# Ans-The multiprocessing package supports spawning processes.
#     It refers to a function that loads and executes a new child processes.
#     For the child to terminate or to continue executing concurrent computing,
#     then the current process hasto wait using an API, which is similar to threading module
```

Q2. What are the differences between multiprocessing and multithreading?

In [2]:
```
# Ans -Multiprocessing uses two or more CPUs to increase computing power,
#      whereas multithreading uses a single process with multiple code segments to increase computing power.
```

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

In [3]:
```
import multiprocessing
print("Number of CPU : ",multiprocessing.cpu_count())
```

```
Number of CPU :  8
```

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

In [4]:
```
# Ans-Python multiprocessing Pool can be used for parallel execution of a function across multiple input values,
#     distributing the input data across processes (data parallelism)
```

Q5. How can we create a pool of worker processes in python using the multiprocessing module?

In [5]:
```
# Ans-Python Multiprocessing Pool: The Complete Guide
#      1.Create the Process Pool.
#      2.Submit Tasks to the Process Pool.
#      3.Wait for Tasks to Complete (Optional)
#      4.Shutdown the Process Pool.
```

Q6. Write a python program to create 4 processes, each process should print a different number using the multiprocessing module in python.

In [7]:
```
import multiprocessing
def helper(n):
    for i in [1,2,3,4,5,6]:
        n.put(i)
def worker(n):
    while True:
        item=n.get()
        if item is None:
            break
        print(item)

if __name__=="__main__":
    queue=multiprocessing.Queue()
    p=multiprocessing.Process(target=helper , args=(queue,))
    q=multiprocessing.Process(target=worker , args=(queue,))
    p.start()
    q.start()
    queue.put("10")
    p.join()
    q.join()
```

In [ ]:
```
```