Q1.what is multithreading in python? why is it used? Name the module used to handle threads in python

Multithreading refers to concurrently executing multiple threads by rapidly switching the control of the CPU between threads (called context switching). The Python Global Interpreter Lock limits one thread to run at a time even if the machine contains multiple processors.

Python multithreading enables efficient utilization of the resources as the threads share the data space and memory. Multithreading in Python allows the concurrent and parallel occurrence of various tasks. It causes a reduction in time consumption or response time, thereby increasing the performance.

modual named threading is used to handle threads in python

Q2). Why threading module used? Write the use of the following functions.

1) activeCount

2) currentThread

3) enumerate

Multithreaded programs can run faster on computer systems with multiple CPUs, because theses threads can be executed truly concurrent. A program can remain responsive to input. This is true both on single and on multiple CPU.

Q3). Explain the following functions.

1) run()

2) start()

3) join()

4) isAlive()

1.run() : The standard run() method invokes the callable object passed to the object’s constructor as the target argument, if any, with positional and keyword arguments taken from the args and kwargs arguments, respectively.

2.start() : Start the thread’s activity. It must be called at most once per thread object. It arranges for the object’s run() method to be invoked in a separate thread of control.

3.join(): Wait until the thread terminates. This blocks the calling thread until the thread whose join() method is called terminates – either normally or through an unhandled exception – or until the optional timeout occurs.

4.isalive(): Return whether the thread is alive. This method returns True just before the run() method starts until just after the run() method terminates. The module function enumerate() returns a list of all alive threads

Q4. Write a python program to create two threads. Thread one must print the list of squares and thread two must print the list of cubes.

In [2]:

```
import threading
def square(num):
    print(f'Square of {num} is {num * num}')


def cube(num):
    print(f'Cube of {num} is {num * num * num}')

thread1 = [threading.Thread(target = square , args =[i]  ) for i in range(1,11)]
thread2 = [threading.Thread(target = cube , args =[i]  ) for i in range(1,11)]

for i in thread1:
    i.start()
for i in thread2:
    i.start()
```

```
Square of 1 is 1
Square of 2 is 4
Square of 3 is 9
Square of 4 is 16
Square of 5 is 25
Square of 6 is 36
Square of 7 is 49
Square of 8 is 64
Square of 9 is 81
Square of 10 is 100
Cube of 1 is 1
Cube of 2 is 8
Cube of 3 is 27
Cube of 4 is 64
Cube of 5 is 125
Cube of 6 is 216
Cube of 7 is 343
Cube of 8 is 512
Cube of 9 is 729
Cube of 10 is 1000
```

Q5. State advantages and disadvantages of multithreading

Multithreading is a technique in Python (and other programming languages) that allows for the execution of multiple threads of code simultaneously within a single process.

Advantages of Multithreading:

1.mproved performance: By executing multiple threads simultaneously, a program can achieve better performance by utilizing available system resources more efficiently.

2.Concurrent operations: Multithreading allows a program to perform several operations concurrently, which can lead to increased productivity and faster completion of tasks.

3.Simplified code: Multithreading can make it easier to write code for complex operations by dividing them into smaller, more manageable pieces that can run concurrently.

Disadvantages of Multithreadin:

1.Increased complexity: Multithreading can make code more complex, as it requires careful synchronization of threads to avoid issues such as race conditions or deadlocks.

2.Debugging difficulties: Debugging multithreaded code can be challenging, as issues may arise due to the unpredictable order in which threads execute.

3.Memory usage: Multithreading can increase memory usage, as each thread requires its own stack space

6. Explain deadlocks and race conditions.

Deadlock : Deadlock describes a condition in which two or more threads are blocked (hung) forever because they are waiting for each other. There are many causes of deadlocks. The Thread Analyzer detects deadlocks that are caused by the inappropriate use of mutual exclusion locks. This type of deadlock is commonly encountered in multi-threaded applications. A process with two or more threads can deadlock when the following conditions hold:

-> Threads that are already holding locks request new locks

-> The requests for new locks are made concurrently

-> Two or more threads form a circular chain in which each thread waits for a lock which is held by the next thread in the chain

Race Conditions : A race condition occurs when two threads access a shared variable at the same time. The first thread reads the variable, and the second thread reads the same value from the variable. Then the first thread and second thread perform their operations on the value, and they race to see which thread can write the value last to the shared variable. The value of the thread that writes its value last is preserved, because the thread is writing over the value that the previous thread wrote.

```
In [ ]:
```