Assignment 14: 14 Feb 2023

Q1. What is multithreading in python? Why is it used? Name the module used to handle threads in python.
Q2. Why is the threading module used? Write the use of the following functions:
1. activeCount()
2. currentThread()
3. enumerate()
Q3. Explain the following functions:
1 run()
2. stort()
3. join()
4. IsAlive()
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.
Q5. State advantages and disadvantages of multithreading.
Q6.Explain deadlocks and race conditions.

Q1. What is multithreading in python? Why is it used? Name the module used to handle threads in python.

Ans: Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching).
Benefits of Multithreading in Python

1. It ensures effective utilisation of computer system resources.
2. Multithreaded applications are more responsive.
3. It shares resources and its state with sub-threads (child) which makes it more economical.
4. It makes the multiprocessor architecture more effective due to similarity.
5. It saves time by executing multiple threads at the same time.
6. The system does not require too much memory to store multiple threads.

There are two main modules of multithreading used to handle threads in Python.

1. The thread module
2. The threading module

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

1. activeCount()
2. currentThread()
3. enumerate()

Ans: The threading module is a high-level implementation of multithreading used to deploy an application in Python. To use multithreading, we need to import the threading module in Python Program.
The following is Threading module function

- activeCount() − Returns the number of thread objects that are active.
- currentThread() − Returns the number of thread objects in the caller's thread control.
- enumerate() − Returns a list of all thread objects that are currently active.

Q3. Explain the following functions:

1 run()
2. stort()
3. join()
4. IsAlive()

Ans:

Methods	Description
start()	A start() method is used to initiate the activity of a thread. And it calls only once for each thread so that the execution of the thread can begin.
run()	A run() method is used to define a thread's activity and can be overridden by a class that extends the threads class.
join()	A join() method is used to block the execution of another code until the thread terminates.
IsAlive()	It is an inbuilt method of the Thread class of the threading module in Python. It uses a Thread object, and checks whether that thread is alive or not

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.

Ans:
```
def print_cube(num):
    # function to print cube of given num
    print("Cube: {}" .format(num * num * num))
def print_square(num):
    # function to print square of given num
    print("Square: {}" .format(num * num))
if __name__ =="__main__":
    # creating thread
    t1 = threading.Thread(target=print_square, args=(10,))
    t2 = threading.Thread(target=print_cube, args=(10,))
    # starting thread 1
    t1.start()
    # starting thread 2
    t2.start()
    # wait until thread 1 is completely executed
    t1.join()
    # wait until thread 2 is completely executed
    t2.join()
```
output
```
Square: 100
Cube: 1000
```

Q5. State advantages and disadvantages of multithreading.

Ans:

Advantages of Multithreading in Python

- Python multithreading enables efficient utilisation 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.

Disadvantages of Multithreading in Python

- Increases the complexity of the program.
- Synchronization of shared resources (objects, data) is necessary.
- Difficult to debug unpredictable results
- Constructing and synchronising threads is CPU/memory intensive.

Q6.Explain deadlocks and race conditions.

Ans:

Deadlock- Two threads hold locks on different resources, each waiting indefinitely for the other to release its lock.

Race condition- Two (or more) threads alter the state of a shared resource concurrently, leaving it in an unpredictable state.