In [ ]:
```
"""
Que 1-.what is multithreading in python? why is it used? Name the module used
to handle threads in python
Ans 1-when a program creates multiple threads with execution cycling among them, so one longer-running task doesn't
block all the others.

The Threading Module is used for it




"""
```

In [1]:
```
"""Que 2-. why threading module used? rite the use of the following functions

 activeCount
 currentThread
 enumerate

Ans 2-
when a program creates multiple threads with execution cycling among them,
so one longer-running task doesn't block all the others

threading.activeCount() - Returns the number of thread objects that are active.

threading.currentThread() - Returns the number of thread objects in the caller's thread control.

threading.enumerate() - Returns a list of all thread objects that are currently active.

"""



# ActiveCount()-

# Program to count active threads
# active_count() method from Threading Module
import threading
import time
# Methods for three threads..
def thread1_Subroutine(i):
    time.sleep(2)
    print("Thread-1: Number of active threads:", threading.active_count())
    print('Thread 1 Value:', i)

def thread2_Subroutine(i):
    print("Thread-2: Number of active threads:", threading.active_count())
    print('Thread 2 Value:', i)

def thread3_Subroutine(i):
    time.sleep(5)
    print("Thread-3: Number of active threads:", threading.active_count())
    print("Thread 3 Value:", i)

# Creating sample threads
thread1 = threading.Thread(target=thread1_Subroutine, args=(100,), name="Thread1")
thread2 = threading.Thread(target=thread2_Subroutine, args=(200,), name="Thread2")
thread3 = threading.Thread(target=thread3_Subroutine, args=(300,), name="Thread3")
print("START: Current active thread count: ", threading.active_count())
# Calling start() method to initialize execution
thread1.start()
thread2.start()
thread3.start()
thread3.join() # Wait for thread-3 to join.
```

```
START: Current active thread count:  5
Thread-2: Number of active threads: 7
Thread 2 Value: 200
Thread-1: Number of active threads: 7
Thread 1 Value: 100
Thread-3: Number of active threads: 6
Thread 3 Value: 300
```

In [3]:
```
# currentThread


import time
import threading

def thread_1(i):
    time.sleep(2)
    print("Active current thread right now:", (threading.current_thread()))
    print('Value by Thread 1:', i)

def thread_2(i):
    time.sleep(5)
    print("Active current thread right now:", (threading.current_thread()))
    print('Value by Thread 2:', i)

def thread_3(i):
    print("Active current thread right now:", (threading.current_thread()))
    print("Value by Thread 3:", i)

# Creating sample threads
thread1 = threading.Thread(target=thread_1, args=(1,))
thread2 = threading.Thread(target=thread_2, args=(2,))
thread3 = threading.Thread(target=thread_3, args=(3,))

print("Active current thread right now:", (threading.current_thread()))
#3 Initially it is the main thread that is active

# Starting the threads
thread1.start()
thread2.start()
thread3.start()
```

```
Active current thread right now: <_MainThread(MainThread, started 11340)>
Active current thread right now: <Thread(Thread-8, started 16548)>
Value by Thread 3: 3
Active current thread right now: <Thread(Thread-6, started 16592)>
Value by Thread 1: 1
Active current thread right now: <Thread(Thread-7, started 18308)>
Value by Thread 2: 2
```

In [4]:

```
# threading.enumerate()

import time
import threading

def thread_1(i):
    time.sleep(5)
    print("Threads alive when thread_1 executes:")
    print(*threading.enumerate(), sep = "\n")

    print()

def thread_2(i):
    print("Threads alive when thread_2 executes")
    print(*threading.enumerate(), sep = "\n")
    print()

def thread_3(i):
    time.sleep(4)


def thread_4(i):
    time.sleep(1)
    print("Threads alive when thread_4 executes")
    print(*threading.enumerate(), sep = "\n")
    print()

# Creating sample threads
thread1 = threading.Thread(target=thread_1, args=(10,))
thread2 = threading.Thread(target=thread_2, args=(20,))
thread3 = threading.Thread(target=thread_3, args=(30,))
thread4 = threading.Thread(target=thread_4, args=(50,))

print("Threads alive in the starting:", threading.enumerate())
print()

# Starting the threads
thread1.start()
thread2.start()
thread3.start()
thread4.start()
```

```
Threads alive in the starting: [<_MainThread(MainThread, started 11340)>, <Thread(Thread-4, started daemon 12828)>, <Heartbeat
(Thread-5, started daemon 13316)>, <HistorySavingThread(IPythonHistorySavingThread, started 12888)>, <ParentPollerWindows(Threa
d-3, started daemon 15748)>]

Threads alive when thread_2 executes
<_MainThread(MainThread, started 11340)>
<Thread(Thread-4, started daemon 12828)>
<Heartbeat(Thread-5, started daemon 13316)>
<HistorySavingThread(IPythonHistorySavingThread, started 12888)>
<ParentPollerWindows(Thread-3, started daemon 15748)>
<Thread(Thread-9, started 8448)>
<Thread(Thread-10, started 1168)>
<Thread(Thread-11, initial)>

Threads alive when thread_4 executes
<_MainThread(MainThread, started 11340)>
<Thread(Thread-4, started daemon 12828)>
<Heartbeat(Thread-5, started daemon 13316)>
<HistorySavingThread(IPythonHistorySavingThread, started 12888)>
<ParentPollerWindows(Thread-3, started daemon 15748)>
<Thread(Thread-9, started 8448)>
<Thread(Thread-11, started 9624)>
<Thread(Thread-12, started 10764)>

Threads alive when thread_1 executes:
<_MainThread(MainThread, started 11340)>
<Thread(Thread-4, started daemon 12828)>
<Heartbeat(Thread-5, started daemon 13316)>
<HistorySavingThread(IPythonHistorySavingThread, started 12888)>
<ParentPollerWindows(Thread-3, started daemon 15748)>
<Thread(Thread-9, started 8448)>
```

In [ ]:
```
"""3. Explain the following functions
 run()
 start()
 join()
 isAlive()



Ans 3 -

run()

method is an inbuilt method of the Thread class of the threading
module in Python

start()

start() method is an inbuilt method of the Thread class of the threading
module in Python.

join()

Join in Python is an in-built method used to join an iterable's elements,
separated by a string separator, which is specified by you

alive()

The . is_alive() method returns True if the thread is still running and False ,
otherwise


"""
```

In [ ]:
```

```

In [ ]:
```
"""
Que 5 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 5

"""
```

In [ ]:
```
"""
Que 5-State advantages and disadvantages of multithreading
Ans 5-

This model provides more concurrency than the many-to-one model. It also
allows another thread to run when a thread makes a blocking system call.
It supports multiple threads to execute in parallel on microprocessors.
Disadvantage
of this model is that creating user thread requires the corresponding
Kernel thread.
"""
```

In [ ]:
```
"""
Que 6-Explain deadlocks and race conditions.
Ans 6

Deadblock-

A Deadlock is a situation where each of the computer process waits for a
resource which is being assigned to some another process. In this situation,
none of the process gets executed since the resource it needs, is held by some
other process which is also waiting for some other resource to be released.


Race Condition-

A race condition occurs when two threads access a shared variable at the same time
"""
```