Assignment - 3

Aim : Implement semaphore for the process synchronisation

Theory :

Semaphore:
A semaphore is a special kind of synchronization data that can be used only through specific synchronization primitives.
When a process performs a wait operation on a semaphore, the operation checks whether the value of the semaphore is >0. If so, it decrements the value of the semaphore and lets the process continue its execution; otherwise, it blocks the process on the semaphore.
A signal operation on a semaphore activates a process blocked on the semaphore if any, or increments the value of the semaphore by 1.
Due to these semantics, semaphores are also called counting semaphores. The initial value of a semaphore determines how many processes can get past the wait operation.

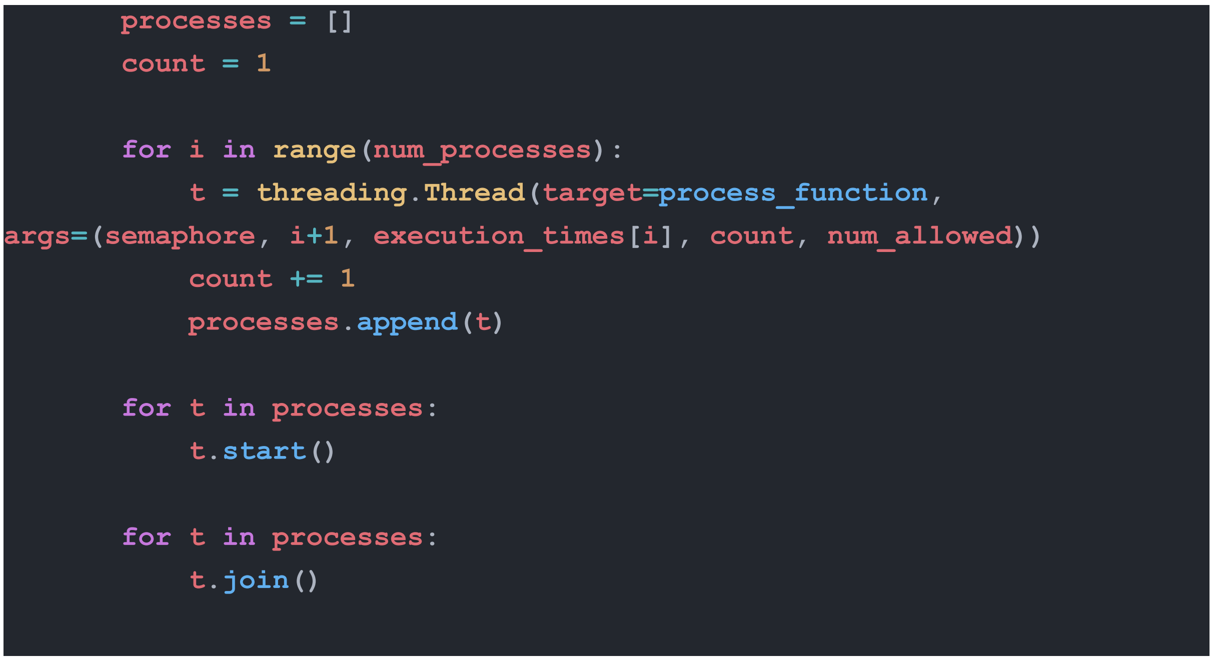
Two types of Semaphores :

1. **Binary Semaphore:-**
   This is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes.
2. **Counting Semaphore:-**
   Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

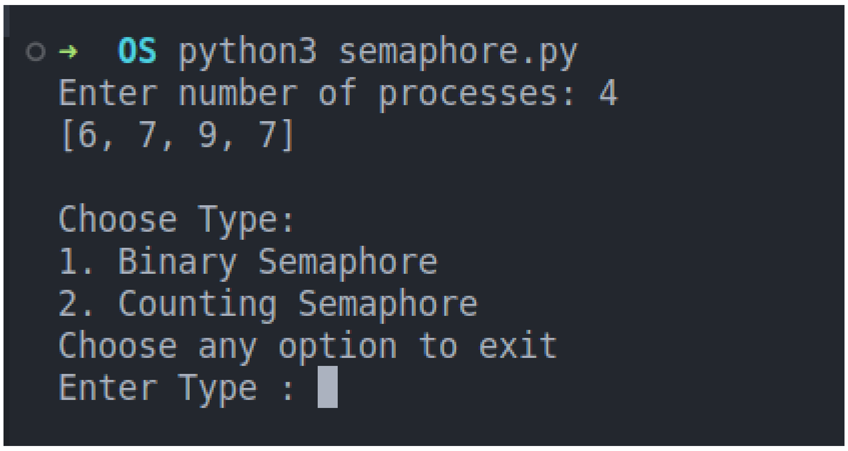
Program & Output :

```
import threading
import random
import time

def acquire(semaphore):
   with semaphore['cv']:
       while semaphore['s'] >= semaphore['max_allowed']:
           semaphore['cv'].wait()
       semaphore['s'] += 1
```

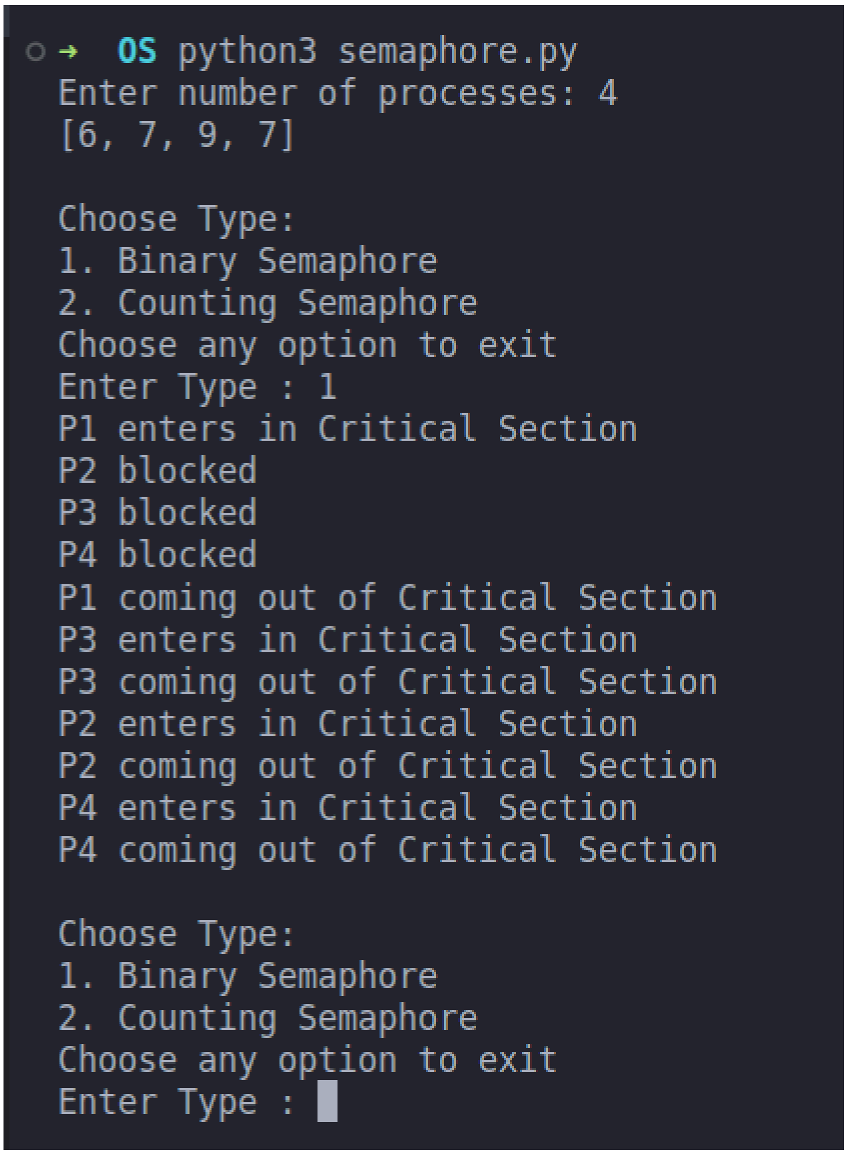
```python
def release(semaphore):
    with semaphore['cv']:
        semaphore['s'] -= 1
        semaphore['cv'].notify_all()

def process_function(semaphore, pid, execution_time, count,
num_allowed):
    if count > num_allowed:
        print(f"P{pid} blocked")
    acquire(semaphore)
    print(f"P{pid} enters in Critical Section")
    time.sleep(execution_time)
    print(f"P{pid} coming out of Critical Section")
    release(semaphore)

if __name__ == "__main__":
    num_processes = int(input("Enter number of processes: "))

    execution_times = [random.randint(1, 10) for _ in
range(num_processes)]
    print(execution_times)

    while True:
        print("\nChoose Type: ")
        print("1. Binary Semaphore")
        print("2. Counting Semaphore")
        print("Choose any option to exit")

        num_allowed = 1

        option = input("Enter Type : ")
        if option == "1":
            semaphore = {'max_allowed': 1, 's': 0, 'cv':
threading.Condition()}
        elif option == "2":
            num_allowed = int(input("Enter number of processes allowed
in critical section: "))
            semaphore = {'max_allowed': num_allowed, 's': 0, 'cv':
threading.Condition()}
        else:
            print("Exiting...")
            break
```



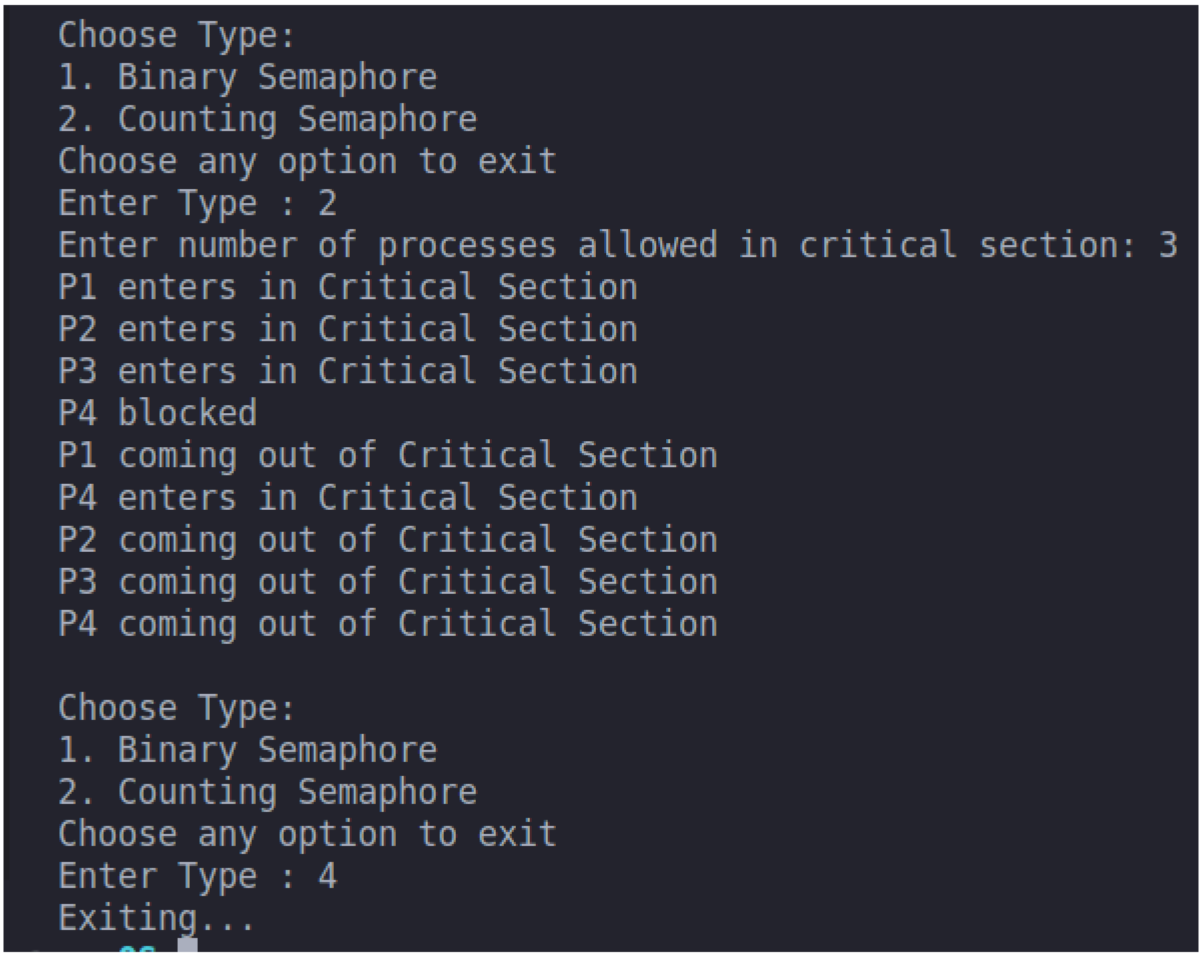
```python
        processes = []
        count = 1

        for i in range(num_processes):
            t = threading.Thread(target=process_function,
args=(semaphore, i+1, execution_times[i], count, num_allowed))
            count += 1
            processes.append(t)

        for t in processes:
            t.start()

        for t in processes:
            t.join()
```



```
○ ➜  OS python3 semaphore.py
  Enter number of processes: 4
  [6, 7, 9, 7]

  Choose Type:
  1. Binary Semaphore
  2. Counting Semaphore
  Choose any option to exit
  Enter Type : 
```



```
○ ➜  OS python3 semaphore.py
  Enter number of processes: 4
  [6, 7, 9, 7]

  Choose Type:
  1. Binary Semaphore
  2. Counting Semaphore
  Choose any option to exit
  Enter Type : 1
  P1 enters in Critical Section
  P2 blocked
  P3 blocked
  P4 blocked
  P1 coming out of Critical Section
  P3 enters in Critical Section
  P3 coming out of Critical Section
  P2 enters in Critical Section
  P2 coming out of Critical Section
  P4 enters in Critical Section
  P4 coming out of Critical Section

  Choose Type:
  1. Binary Semaphore
  2. Counting Semaphore
  Choose any option to exit
  Enter Type :
```



```
Choose Type:
1. Binary Semaphore
2. Counting Semaphore
Choose any option to exit
Enter Type : 2
Enter number of processes allowed in critical section: 3
P1 enters in Critical Section
P2 enters in Critical Section
P3 enters in Critical Section
P4 blocked
P1 coming out of Critical Section
P4 enters in Critical Section
P2 coming out of Critical Section
P3 coming out of Critical Section
P4 coming out of Critical Section

Choose Type:
1. Binary Semaphore
2. Counting Semaphore
Choose any option to exit
Enter Type : 4
Exiting...
```

Conclusion :

Hence, here we learnt about what semaphore is, what are the types and implemented those in python using threads.