

11.

"""

Write a program on threading using python.

QAYAM 231P038/ 02

"""

```
print("*****")
```

```
print("Threading using Python")
```

```
print("Mohd Qayam")
```

```
print("*****")
```

```
# Importing required modules
```

```
from threading import Thread
```

```
from time import sleep
```

```
class Theatre:
```

```
    # Constructor
```

```
    def __init__(self, message):
```

```
        self.message = message
```

```
    # Method to simulate movie ticket and seat allocation
```

```
    def movieshow(self):
```

```
        for i in range(1, 6):
```

```
            print(self.message, ":", i)
```

```
            sleep(0.5) # Simulating delay
```

```
# Creating two instances of Theatre class
```

```
obj1 = Theatre("Cut Ticket")
```

```
obj2 = Theatre("Show Chair")
```

```
# Creating two threads to run movieshow()
```

```
t1 = Thread(target=obj1.movieshow)
```

```
t2 = Thread(target=obj2.movieshow)
```

```
# Starting the threads
```

```
t1.start()
```

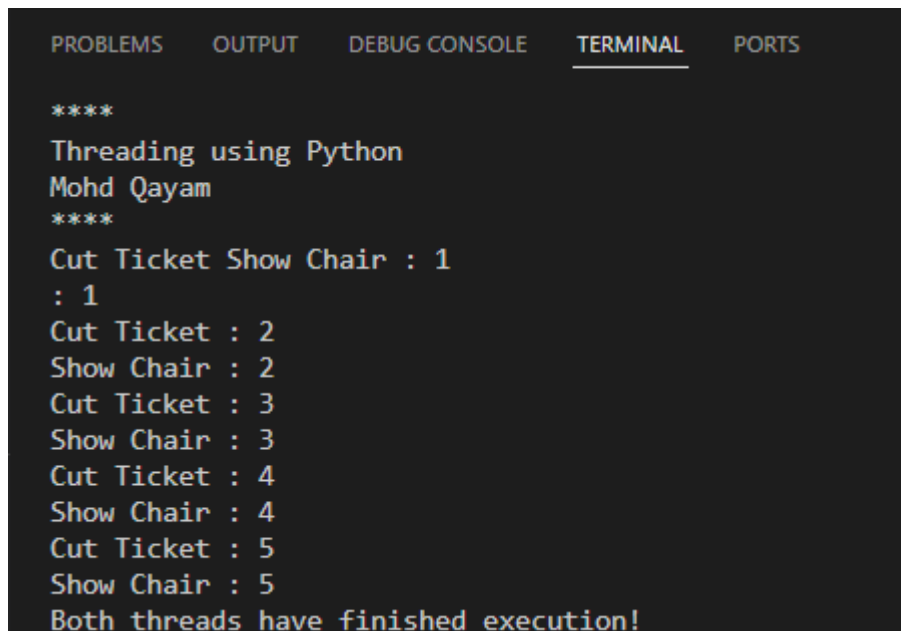
```
t2.start()
```

```
# Ensuring both threads finish before the program exits
```

```
t1.join()
```

```
t2.join()
print("Both threads have finished execution!")
```

Output:

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal output is as follows:

```
****
Threading using Python
Mohd Qayam
****
Cut Ticket Show Chair : 1
: 1
Cut Ticket : 2
Show Chair : 2
Cut Ticket : 3
Show Chair : 3
Cut Ticket : 4
Show Chair : 4
Cut Ticket : 5
Show Chair : 5
Both threads have finished execution!
```

11a.

```
"""
```

Write a program for single thread.

QAYAM 231P038/ 02

```
"""
```

```
import threading
```

```
import time
```

```
# Function to print numbers with a delay
```

```
def print_numbers():
```

```
    for i in range(1, 6):
```

```
        print(f"Number: {i}")
```

```
        time.sleep(1) # Delay of 1 second
```

```
# Creating a single thread
```

```
t1 = threading.Thread(target=print_numbers)
```

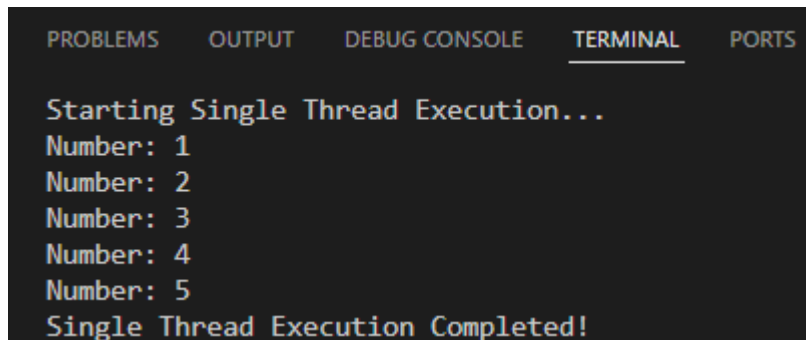
```
print("Starting Single Thread Execution...")
```

```
t1.start()
```

```
t1.join()
```

```
print("Single Thread Execution Completed!")
```

OUTPUT:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Starting Single Thread Execution...
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Single Thread Execution Completed!
```

11b.

"""

Write a program for multiple thread.

QAYAM 231P038/ 02

"""

```
import threading
```

```
import time
```

```
# Function to print numbers
```

```
def print_numbers():
```

```
    for i in range(1, 6):
```

```
        print(f"Thread 1 - Number: {i}")
```

```
        time.sleep(1) # Delay of 1 second
```

```
# Function to print alphabets
```

```
def print_alphabets():
```

```
    for ch in 'ABCDE':
```

```
        print(f"Thread 2 - Alphabet: {ch}")
```

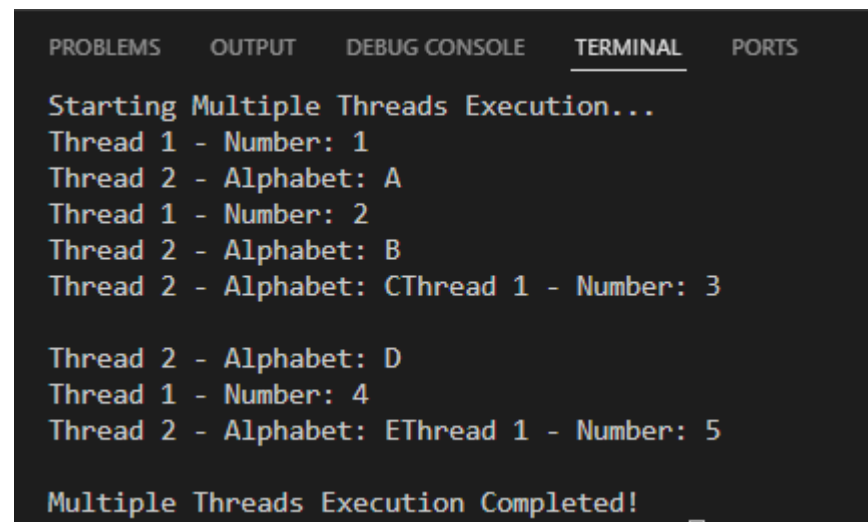
```
        time.sleep(1) # Delay of 1 second
```

```
# Creating multiple threads
```

```
t1 = threading.Thread(target=print_numbers)
```

```
t2 = threading.Thread(target=print_alphabets)
print("Starting Multiple Threads Execution...")
# Starting the threads
t1.start()
t2.start()
# Ensuring both threads complete execution before program exits
t1.join()
t2.join()
print("Multiple Threads Execution Completed!")
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Starting Multiple Threads Execution...
Thread 2 - Alphabet: A
Thread 1 - Number: 1
Thread 2 - Alphabet: B
Thread 1 - Number: 2
Thread 2 - Alphabet: C
Thread 1 - Number: 3
Thread 2 - Alphabet: D
Thread 1 - Number: 4
Thread 2 - Alphabet: E
Thread 1 - Number: 5

Multiple Threads Execution Completed!
```