```
#Aim:Write a Programs on Threading using python.
# Branch: Comps
# Year: SE
# Sem: IV
# Subject: Python
# Name: Sidra Shaikh
# UIN: 231P064
# Roll No: 40
# Multitasking using two threads
from threading import *
from time import *

class Theatre:
    def __init__(self, str, lock):
        self.str = str
        self.lock = lock

    def movieshow(self):
        for i in range(1, 6):
            with self.lock:
                print(self.str, ":", i)
                sleep(0.5)
lock = Lock()
obj1 = Theatre("Cut Ticket", lock)
obj2 = Theatre("Show Chair", lock)
t1 = Thread(target=obj1.movieshow)
t2 = Thread(target=obj2.movieshow)
t1.start()
t2.start()
t1.join()
t2.join()


# Aim:Write a program for single thread.
# Branch:Comps
# Year:SE
# Sem:IV
# Subject:Python
# Name: Sidra Shaikh
# UIN:231P064
# Roll No:40
import time

def task1():
    print("Task 1 started.")
    time.sleep(2)
    print("Task 1 completed.")

def task2():
    print("Task 2 started.")
```

```
    time.sleep(3)
    print("Task 2 completed.")

def main():
    print("Program started.")
    task1()
    task2()
    print("Program completed.")

if __name__ == "__main__":
    main()


# Aim:Write a program for multiple thread.
# Branch:Comps
# Year:SE
# Sem:IV
# Subject:Python
# Name: Sidra Shaikh
# UIN:231P064
# Roll No:40
import threading
import time

def task1():
    print("Task 1 started.")
    time.sleep(2)
    print("Task 1 completed.")

def task2():
    print("Task 2 started.")
    time.sleep(3)
    print("Task 2 completed.")

def main():
    print("Program started.")

    thread1 = threading.Thread(target=task1)
    thread2 = threading.Thread(target=task2)

    thread1.start()
    thread2.start()

    thread1.join()
    thread2.join()

    print("Program completed.")

if __name__ == "__main__":
    main()
```

```
""" WAP to perform following operation on stack
1. push an element
2. pop an element
3. peep an element
4. search an element
5. Exit
"""
# Branch:Comps
# Year:SE
# Sem:IV
# Subject:Python
# Name: Sidra Shaikh
# UIN:231P064
# Roll No: 40
# Stack class - save this as stack.py
class Stack:
    def __init__(self):
        self.st = []

    def isempty(self):
        return self.st == []

    def push(self, element):
        self.st.append(element)

    def pop(self):
        if self.isempty():
            return -1
        else:
            return self.st.pop()

    def peep(self):
        n = len(self.st)
        return self.st[n - 1]

    def search(self, element):
        if self.isempty():
            return -1
        else:
            try:
                n = self.st.index(element)
                return len(self.st) - n
            except ValueError:
                return -2

    def display(self):
        return self.st
```

```
# using Stack class of stack.py program
from Stack import Stack
# Create empty stack object
s = Stack()

# Display menu
choice = 0
while choice < 5:
    print("STACK OPERATIONS")
    print("1. Push Element")
    print("2. Pop Element")
    print("3. Peep Element")
    print("4. Search an Element")
    print("5. Exit")

    choice = int(input("Enter Your Choice: "))

    # Perform task based on user's choice
    if choice == 1:
        element = int(input("Enter Element: "))
        s.push(element)

    elif choice == 2:
        element = s.pop()
        if element == -1:
            print("Stack is empty")
        else:
            print("Popped Element:", element)

    elif choice == 3:
        element = s.peep()
        print("Topmost element:", element)

    elif choice == 4:
        element = int(input("Enter Element: "))
        pos = s.search(element)
        if pos == -1:
            print("Stack is empty")
        elif pos == -2:
            print("Element not found in the stack")
        else:
            print("Element found at position:", pos)

    else:
        break

    # Display the content of the stack object
    print("Stack =", s.display())
```