Experiment No:5

Aim: To implement menu driven programs for Link List, Stack and Queue in python

Theory:

A linked list is a sequential collection of data elements, which are connected together via links. A linked list consists of independent nodes containing any type of data and each node holds a reference or a link to the next node in the list.

The beginning node of a linked list is called the **head** and the end node is called the **tail.** All nodes of a linked list are independent and are not stored contagiously in memory.

Types of Linked Lists
There are 4 types of linked lists that can be created in python.
Singly Linked List
Circular Singly Linked List
Doubly Linked List
Circular Doubly Linked List

Stack:
In python, the stack is an abstract data structure that stores elements linearly. The items in a stack follow the Last-In/First-Out (LIFO) order. This means that the last element to be inserted in a stack will be the first one to be removed.

Stack Operations
Various operations can be performed on a stack in python.
Create Stack
Push

Pop
Peek
isEmpty
isFull
deleteStack

Queue
In python, the queue is an abstract data structure that stores elements linearly. The items in a queue follow the First-In/First-Out (FIFO) order. This means that the first element to be inserted in a queue will be the first one to be removed.

Queue Operations
Various operations can be performed on a queue in python.
Create Queue .
Enqueue
Dequeue
Peek
isEmpty
isFull
deleteQueue

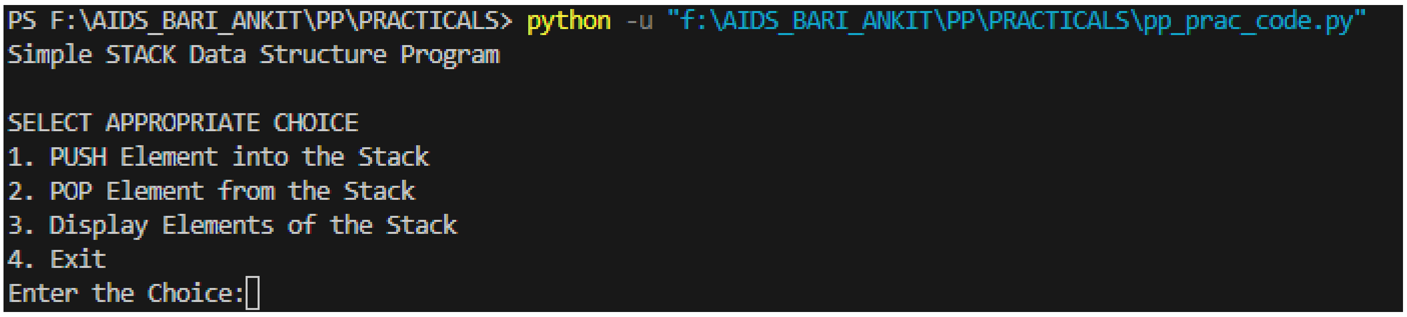
PROGRAM

Program 5.1: Stack

```
# Program introduction statement
print("Simple STACK Data Structure Program")

# Initial empty STACK
stack = []

# Display Menu with Choices
while True:
    print("\nSELECT APPROPRIATE CHOICE")
    print("1. PUSH Element into the Stack")
    print("2. POP Element from the Stack")
    print("3. Display Elements of the Stack")
    print("4. Exit")

    # Taking input from the user regarding choice
    choice = int(input("Enter the Choice:"))

    # USER enter option 1 then PUSH elements into the STACK
    if choice == 1:
        # append() function to PUSH elements into the STACK
        stack.append("Monday")       # PUSH element Monday
        stack.append("Tuesday")      # PUSH element Tuesday
        stack.append("Wednesday")    # PUSH element Wednesday
        stack.append("Thursday")     # PUSH element Thursday
        stack.append("Friday")       # PUSH element Friday
        stack.append("Saturday")     # PUSH element Saturday
        stack.append("Sunday")       # PUSH element Sunday
        stack.append('8')            # PUSH element 8
        print('\nTotal 8 elements PUSH into the STACK')

    # USER enter option 2 then POP one element from the STACK
    elif choice == 2:
        if len(stack) == 0:
            # Check whether STACK is Empty or not
            print('The STACK is EMPTY No element to POP out')
        else:
            # pop() function to POP element from the STACK in LIFO order
            print('\nElement POP out from the STACK is:')
            print(stack.pop())   # Display the element which is POP out from the STACK
```

```
# USER enter option 3 then display the STACK
elif choice == 3:
    if len(stack) == 0:
        # Check whether STACK is Empty or not
        print('The STACK is initially EMPTY') # Display this message if STACK is Empty
    else:
        print("The Size of the STACK is: ", len(stack)) # Compute the size of the STACK
        print('\nSTACK elements are as follows:')
        print(stack)    # Display all the STACK elements

# User enter option 4 then EXIT from the program
elif choice == 4:
    break

# Shows ERROR message if the choice is not in between 1 to 4
else:
    print("Oops! Incorrect Choice")
```

OUTPUT:



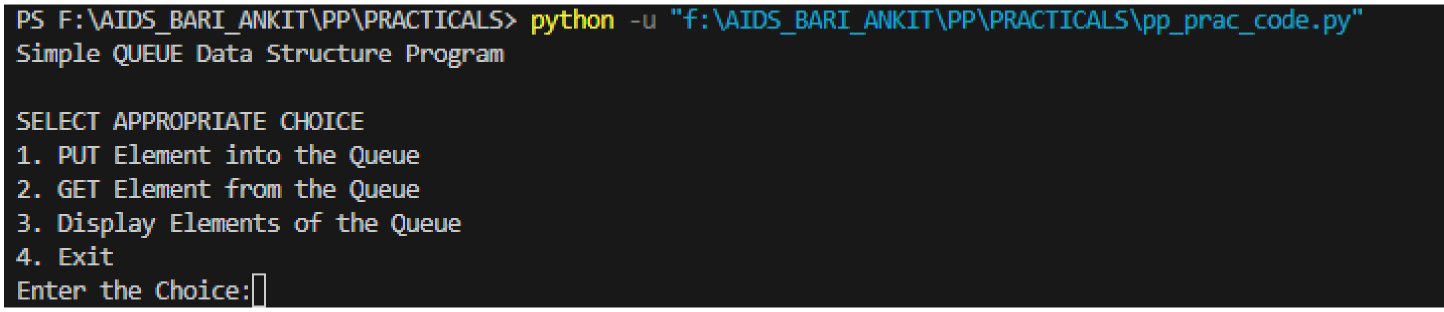
```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Simple STACK Data Structure Program

SELECT APPROPRIATE CHOICE
1. PUSH Element into the Stack
2. POP Element from the Stack
3. Display Elements of the Stack
4. Exit
Enter the Choice:
```

Program 5.2:Queue

```
# Import Python Package
from queue import Queue

# Program introduction statement
print("Simple QUEUE Data Structure Program")

# Initial empty QUEUE
queue = Queue()

# Display Menu with Choices
while True:
   print("\nSELECT APPROPRIATE CHOICE")
   print("1. PUT Element into the Queue")
   print("2. GET Element from the Queue")
   print("3. Display Elements of the Queue")
   print("4. Exit")

   # Taking input from the user regarding choice
   choice = int(input("Enter the Choice:"))

   # USER enter option 1 then PUT elements into the QUEUE
   if choice == 1:
      # put() function to PUT elements into the QUEUE
      queue.put("Monday")      # PUT element Monday
      queue.put("Tuesday")     # PUT element Tuesday
      queue.put("Wednesday")   # PUT element Wednesday
      queue.put("Thursday")    # PUT element Thursday
      queue.put("Friday")      # PUT element Friday
      queue.put("Saturday")    # PUT element Saturday
      queue.put("Sunday")      # PUT element Sunday
      queue.put('8')           # PUT element 8
      print('\nTotal 8 elements PUT into the QUEUE')

   # USER enter option 2 then GET one element from the QUEUE
   elif choice == 2:
      if queue.empty():
         # Check whether QUEUE is Empty or not
```

```
            print('The QUEUE is EMPTY No element to GET out')
        else:
            # get() function to GET element out from the QUEUE in FIFO order
            print('\nElement GET out from the QUEUE is:')
            print(queue.get())   # Display the element which is GET out from the QUEUE







    # USER enter option 3 then display the QUEUE
    elif choice == 3:
        if queue.empty():
            # Check whether QUEUE is Empty or not
            print('The QUEUE is initially EMPTY') # Display this message if QUEUE is Empty
        else:
            print("The Size of the QUEUE is: ", queue.qsize()) # Compute the size of the QUEUE
            print('\nQUEUE elements are as follows:')
            print(list(queue.queue))    # Display all the QUEUE elements

    # User enter option 4 then EXIT from the program
    elif choice == 4:
        break

    # Shows ERROR message if the choice is not in between 1 to 4
    else:
        print("Oops! Incorrect Choice")
```

OUTPUT:

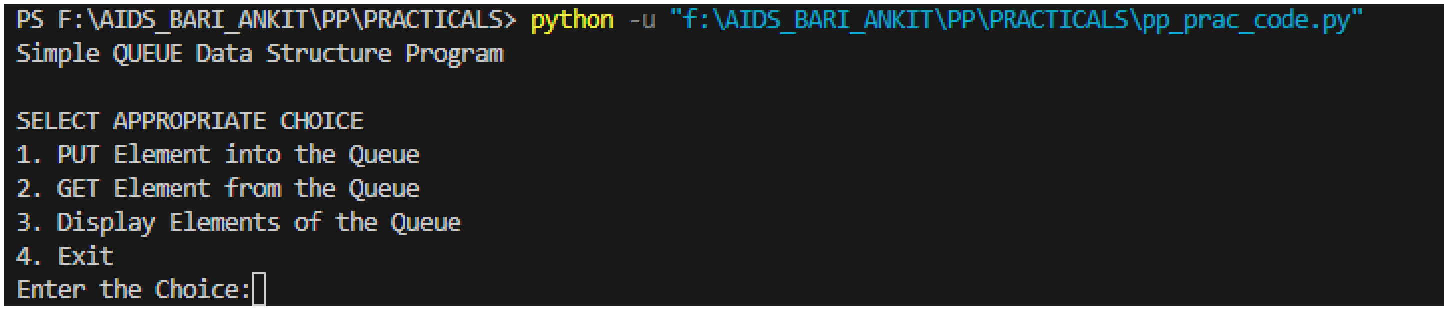


```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Simple QUEUE Data Structure Program

SELECT APPROPRIATE CHOICE
1. PUT Element into the Queue
2. GET Element from the Queue
3. Display Elements of the Queue
4. Exit
Enter the Choice:
```

Program 5.3:Linked List

```
# Importing module
import collections

# Program introduction statement
print("Simple LINKED LIST Data Structure Program")

# Initialising a deque() to create Linked List
linked_lst = collections.deque()

# Display Menu with Choices
while True:
    print("\nSELECT APPROPRIATE CHOICE")
    print("1. INSERT elements into Linked List")
    print("2. INSERT element at a Specific Position")
    print("3. Display all the elements of the Linked List")
    print("4. DELETE the last element from the Linked List")
    print("5. DELETE the specific element from the Linked List")
    print("6. Exit")

    # Taking input from the user regarding choice
    choice = int(input("Enter the Choice:"))



    # USER enter option 1 then INSERT element in the Linked List
    if choice == 1:
        # append() function to fill deque() with elements and inserting into the Linked List
        linked_lst.append("Monday")    # INSERT element Monday
        linked_lst.append("Tuesday")   # INSERT element Tuesday
        linked_lst.append("Wednesday") # INSERT element Wednesday
        linked_lst.append("Sunday")    # INSERT element Sunday
        print('\nTotal 4 elements INSERTED into the Linked List')

    # USER enter option 2 then INSERT element at a specific position in the Linked List
    elif choice == 2:
        # insert() function add element after the specified position in the Linked List
        linked_lst.insert(3, 'Thursday')   # INSERT element Thursday after 3rd element of Linked
List
        linked_lst.insert(5, 'Saturday')   # INSERT element Saturday after 5th element of Linked
List
        linked_lst.insert(4, 'Friday')     # INSERT element Friday after 4th element of Linked List
        print('\nTotal 3 new elements INSERTED at specific position in the Linked List')

    # USER enter option 3 then display the Linked List
    elif choice == 3:
        if len(linked_lst) == 0:
            # Check whether Linked List is Empty or not
            print('The Linked List is initially EMPTY')
        else:
            print("The Size of the Linked List is: ", len(linked_lst)) # Compute the size of the Linked
List
            print('\nLinked List elements are as follows:')
            print(linked_lst)

    # USER enter option 4 then DELETE last element in the Linked List
    elif choice == 4:
        if len(linked_lst) == 0:
            # Check whether Linked List is Empty or not
            print('The Linked List is EMPTY No element to DELETE')
        else:
            # pop() function to DELETE last element in the Linked List
            print('\nLast element DELETED from the Linked List is:')
            print(linked_lst.pop())   # Display the element which is Deleted from the Linked List

    # USER enter option 5 then DELETE the specific element from the Linked List
    elif choice == 5:
        if len(linked_lst) == 0:
            # Check whether Linked List is Empty or not
            print('The Linked List is EMPTY No element to DELETE')
        else:
            # remove() function to DELETE the specific element from the Linked List
            print('\nSpecific element "Monday" DELETED from the Linked List')
            linked_lst.remove('Monday')    # Remove Monday from the Linked List









    # User enter option 6 then EXIT from the program
    elif choice == 6:
        break

    # Shows ERROR message if the choice is not in between 1 to 6
    else:
        print("Oops! Incorrect Choice")
```

OUTPUT:



```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.py"
Simple QUEUE Data Structure Program

SELECT APPROPRIATE CHOICE
1. PUT Element into the Queue
2. GET Element from the Queue
3. Display Elements of the Queue
4. Exit
Enter the Choice:
```

Conclusion:

The implementation of menu-driven programs for linked lists, stacks, and queues inPython has demonstrated their versatility and efficiency in managing data structures. Through this experiment, we have gained insights into the practical applications of these fundamental data structures, paving the way for further exploration and optimization in programming solutions.