Name : Shawn Louis SE COMPS Roll No : 31

Experiment No: 6

Topic:	Implement stack, queue, deque and linked list.
Prerequisite:	Knowledge of some programming language like Data Structures, C, Java.
Mapping With COs:	CSL405.3
Objective:	- Ability to understand and implement the abstract data types stack, queue, deque, and Linked List.
Outcome:	- This will motivate you to write clean, readable and efficient code in Python. - To introduce the fundamental concept of data structures and to emphasize the importance of data structures in developing and implementing efficient algorithms. - To implement the abstract data types stack, queue, dequeue, and Linked List.
Bloom's Taxonomy	Apply
Theory/ Steps/ Algorithm/ Procedure:	**Stack:** Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order. Also, the inbuilt functions in Python make the code short and simple. To add an item to the top of the list, i.e., to push an item, we use **append()** function and to pop out an element we use **pop()** function. These functions work quiet efficiently and fast in end operations. **Queue**: A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource

where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Deque

Deque can be implemented in python using the module “**collections**“. Deque is preferred over list in the cases where we need quicker append and pop operations from both the ends of container, as deque provides an **O(1)** time complexity for append and pop operations as compared to list which provides O(n) time complexity.

Operations on deque :

1. append() :- This function is used to insert the value in its argument to the right end of deque.

2. appendleft() :- This function is used to insert the value in its argument to the left end of deque.

3. pop() :- This function is used to delete an argument from the right end of deque.

4. popleft() :- This function is used to delete an argument from the left end of deque.

Linked list

A linked list is a sequence of data elements, which are connected together via links. Each data element contains a connection to another data element in form of a pointer. Python does not have linked lists in its standard library. We implement the concept of linked lists using the concept of nodes as discussed in the previous chapter. We have already seen how we create a node class and how to traverse the elements of a node. In this chapter we are going to study the types of linked lists known as singly linked lists. In this type of data structure there is only one link between any two data elements. We create such a list and create additional methods to insert, update and remove elements from the list.

Creation of Linked list

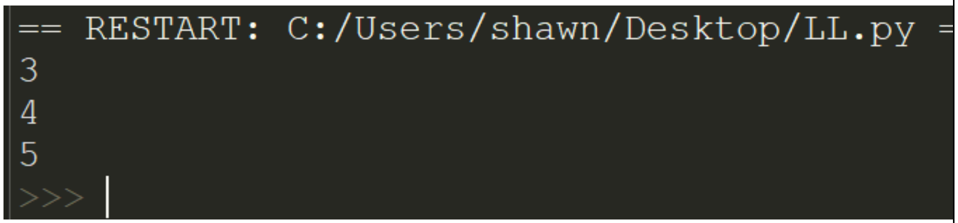
A linked list is created by using the node class we studied in the last chapter. We create a Node object and create another class to use this ode object. We pass the appropriate values through the node object to point the to the next data elements. The below program creates the linked list with three data elements. In the next section we will see how to traverse the linked list.

Experiments:

1. Implement Data Structures in Python (Linked List, Stack, Queues, Deque)

   Linked list:

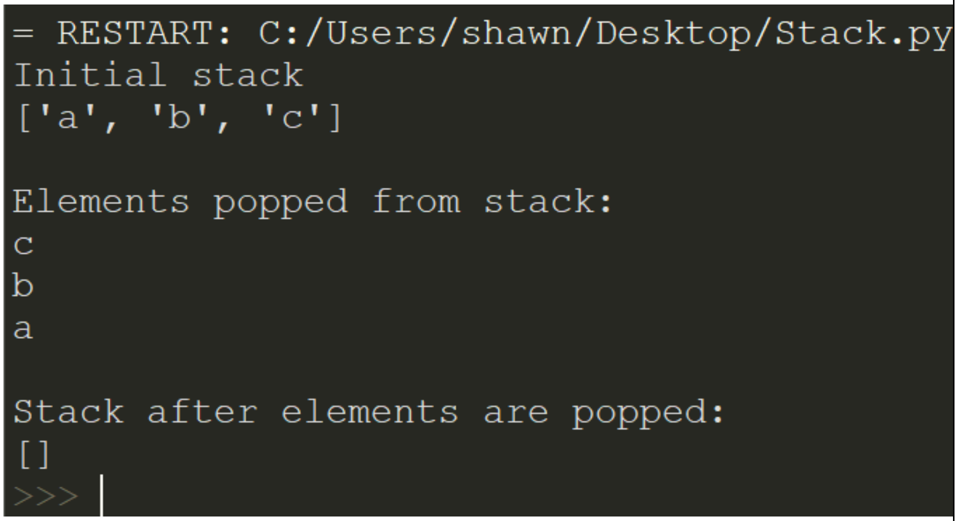
```
# A single node of a singly linked list
class Node:
  # constructor
  def __init__(self, data = None, next=None):
    self.data = data
    self.next = next

# A Linked List class with a single head node
class LinkedList:
  def __init__(self):
    self.head = None

  # insertion method for the linked list
  def insert(self, data):
    newNode = Node(data)
    if(self.head):
      current = self.head
      while(current.next):
        current = current.next
      current.next = newNode
    else:
      self.head = newNode

  # print method for the linked list
  def printLL(self):
    current = self.head
    while(current):
      print(current.data)
      current = current.next

# Singly Linked List with insertion and print methods
LL = LinkedList()
LL.insert(3)
LL.insert(4)
LL.insert(5)
LL.printLL()
```



```
== RESTART: C:/Users/shawn/Desktop/LL.py =
3
4
5
>>> |
```

Stack:

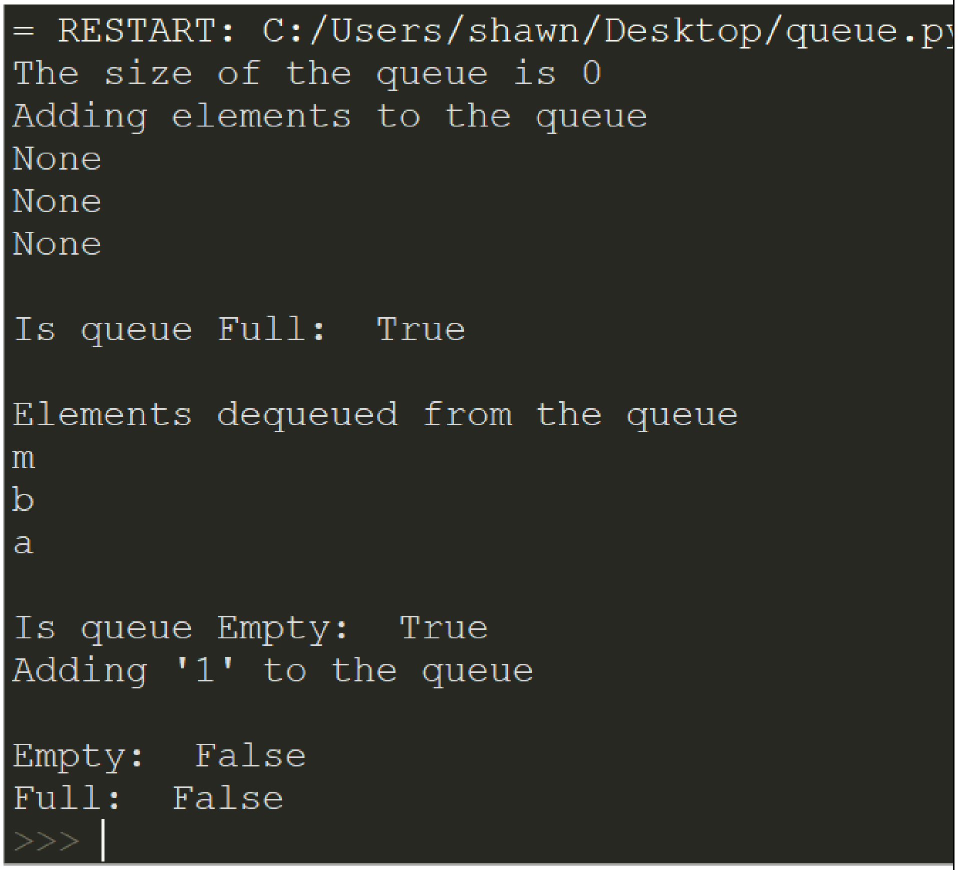
```
stack = []

stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)

# pop() fucntion to pop
# element from stack in
# LIFO order
print('\nElements poped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are poped:')
print(stack)
```



```
= RESTART: C:/Users/shawn/Desktop/Stack.py
Initial stack
['a', 'b', 'c']

Elements popped from stack:
c
b
a

Stack after elements are popped:
[]
>>>
```

Queue:

```
from queue import Queue

# Initializing a queue
q = Queue(maxsize = 3)

print('The size of the queue is {}'.format(q.qsize()))

# Adding of element to queue
print('Adding elements to the queue')
print(q.put('m'))
print(q.put('b'))
print(q.put('a'))
print("\nIs queue Full: ", q.full())

# Removing element from queue
print("\nElements dequeued from the queue")
print(q.get())
print(q.get())
print(q.get())

print("\nIs queue Empty: ", q.empty())
print("Adding '1' to the queue")
q.put(1)
print("\nEmpty: ", q.empty())
print("Full: ", q.full())
```

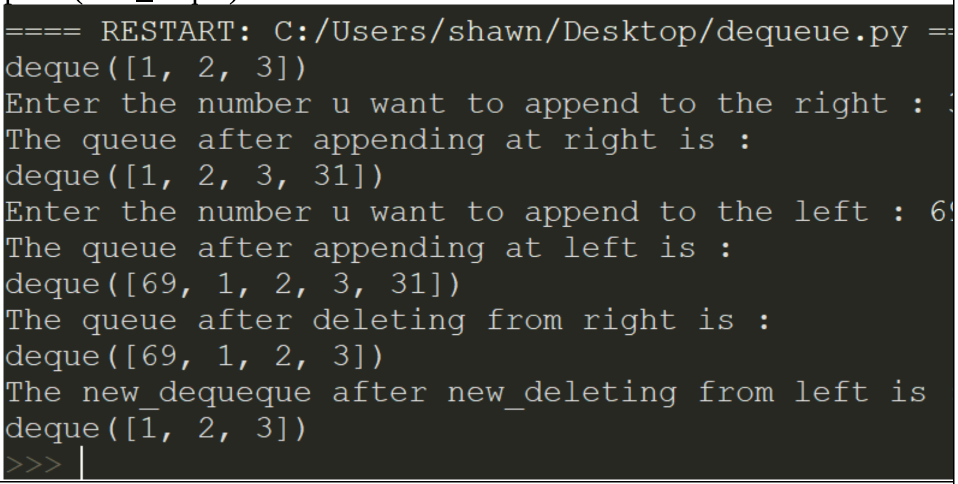


```
= RESTART: C:/Users/shawn/Desktop/queue.py
The size of the queue is 0
Adding elements to the queue
None
None
None

Is queue Full:  True

Elements dequeued from the queue
m
b
a

Is queue Empty:  True
Adding '1' to the queue

Empty:  False
Full:  False
>>> 
```

Deque:

```
import collections


new_deque = collections.deque([1,2,3])
print(new_deque)

n=int(input('Enter the number u want to append to the right'))
new_deque.append(n)

print ("The queue after appending at right is : ")
print (new_deque)
m=int(input('Enter the number u want to append to the left'))

new_deque.appendleft(m)
```

```
print ("The queue after appending at left is : ")
print (new_deque)

new_deque.pop()

print ("The queue after deleting from right is : ")
print (new_deque)


new_deque.popleft()


print ("The new_dequeque after new_dequeleting from left is : ")
print (new_deque)
```



```
==== RESTART: C:/Users/shawn/Desktop/dequeue.py =
deque([1, 2, 3])
Enter the number u want to append to the right :
The queue after appending at right is :
deque([1, 2, 3, 31])
Enter the number u want to append to the left : 6
The queue after appending at left is :
deque([69, 1, 2, 3, 31])
The queue after deleting from right is :
deque([69, 1, 2, 3])
The new_dequeque after new_deleting from left is
deque([1, 2, 3])
>>>
```

Deliverables:	All scripting code executed with their output.
Conclusion:	Thus we have successfully able to write a program to implement Stack, Queue, Deque and Linked List.
References:	https://www.geeksforgeeks.org/using-list-stack-queues-python/ https://www.tutorialspoint.com/python_data_structure/python_linked_lists.htm https://www.geeksforgeeks.org/deque-in-python/

Don Bosco Institute of Technology
Department of Computer Engineering

Academic year – 2019-20

<u>Open Source Technology Lab</u>

<u>Assessment Rubric for Experiment No.: 6</u>

Performance Date :
Submission Date :

Title of Experiment	: Implement stack, queue, deque and linked list
Year and Semester	: 2nd Year and IVth Semester
Batch	: Computer
Name of Student	: Shawn Louis
Roll No.	: 31

Performance	**Poor**	**Satisfactory**	**Good**	**Excellent**	**Total**
	2 points	3 points	4 points	5 points	
Results and Documentations	**Poor**	**Satisfactory**	**Good**	**Excellent**	
	2 points	3 points	4 points	5 points	
Timely Submission	**Submission beyond 14 days of the deadline**	**Late submission till 14 days**	**Late submission till 7 days**	**Submission on time**	
	2 points	3 points	4 points	5 points	

<u>Signature</u>

(Sana Shaikh)