Data Structure

Data Structure

- It is a logical way of organizing data that makes them efficient to use.
- Addition
- Modification
- Deletion
- Searching
- Traversing (display)

STACK AND QUEUE

STACK

- A Stack is linear structure implemented in LIFO(last in first out). A stack is a dynamic data structure, bcoz it an grow(increase number of elements) or shrink(decrease number of elements). A static data structure has fixed size.
- LIFO-means element last inserted would be the first one to be deleted.
- Two rules of Stack:
  - 1. Data can only be removed from top of stack(pop operation)
  - 2. New element can only be added to top of stack(push operation)

STACK

- Some terms related to Stack:
- **Peek/inspection**-inspecting the value at stack's top without removing it.
- **Overflow**: It is a situation when someone tries to push an item in stack, when stack if fixed and can't grow further or no memory left for new item.
- **Underflow**: It is a situation when someone tries to pop an item in empty stack.

Stack Application

- Stack can be used for various purposes. e.g.
- **Reversing a line-** to push each character on to a stack as it is read. When the line is finished, characters are then popped off the stack.
- **Polish Strings-** convert arithmetic expressions into polish strings by using stacks. It is of two types: postfix and prefix.
- **To reverse a word.**
- **An "undo" mechanism in text editors**; this operation is accomplished by keeping all text changes in a **stack**. ...
- **A stack of plates/books in a cupboard**.
- **Wearing/Removing Bangles**.

Types of Stack

- An item stored in a stack is also called item-code. We can implement stack which contain group information.

- Stack of integers

4
5
6
8

- Stack of strings

‘a’
‘b’
‘c’
‘d’

- Stack of lists

1,’abc’,98
2,’xyz’,78
3,’def’,98

Implementing Stack in Python

- We use lists to implement Stack.

```
#to insert element into stack
s=[]
def push(s,e):
    s.append(e)
    top=len(s)-1
```

Implementing Stack in Python

- We use lists to implement Stack.

```
#to delete element from stack
s=[]
def pop(s):
    if (s==[]):
        print("underflow")
    else:
        e=s.pop()
        if(len(s)==0):
            top=None
        else:
            top=len(s)-1
        return e
```

Implementing Stack in Python

- We use lists to implement Stack.

```
#to display element of stack
s=[]
def display(s):
        if(s==[]):
                print("underflow")
        else:
                top=len(s)-1
                print(s[top],"<--top")
                for d in range(top-1,-1,-1):
                        print(s[d])
```

QUEUE

- Queues are similar to stacks. But queue follow FIFO(first in first out).
- A queue has two ends:

  Front-end: items are removed from front

  Rear-end: items are added from back-end.
- Two rules of Queue:
  - 1. Data can only be removed (dequeue/pop) from front.
  - 2. New element can only be added (enqueue/push) to backend.

Applications of Queue

- Passengers leaving a bus
- Call center use queue
- Printing purposes
- Single lane vehicle into a vehicle
- Airport implement queue
- Runway for both landing and take off
- When running multiple programs on PC, CPU might use queues to process applications in phased manner.

Implementing Queue in Python

- We use lists to implement Queue.

```
#to insert element into queue
q=[]
f=0
r=0
def en(q,e):
    q.append(e)
    if(len(q)==1):
        f=r=0
    else:
        r=len(q)-1
```

Implementing Queue in Python

- We use lists to implement queue.

```
#to delete element from Queue
q=[]
f=0
r=0
def pop(s):
    if (q==[]):
        print("underflow")
    else:
        e=q.pop()
        if(len(q)==0):
            f=r=None
        return e
```

Implementing Queue in Python

```
#to display element of Queue
def display(q):
    if(q==[]):
        print("underflow")
    else:
        f=0
        r=len(q)-1
        print(q[f],"<--front")
        for d in range(1,r):
            print(q[d])
        print(q[r],"<-rear")
```

Variation in Queues

- Circular Queues- These queues are implemented in circular form not in straight line. In queues after some insertions and deletions, some unutilized space lies in the beginning of the queue. To overcome such problem, circular queues are used.
- Deque- Double ended queues
- It is a refined queues in which elements can be added or removed at either end but not in the middle.
- Deque is preferred over list in those cases where we need quicker append and pop operations from both the ends.

Variation in Dequeues

- There are two variations of deque-
  - Input restricted deque-allows insertions at only one end but allows deletions at both ends
  - Output restricted deque-allows deletions at only one end but allows insertions at both ends