Stack

- A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.

Stack in Python can be implemented using following ways:

- list
- collections.deque
- queue.LifoQueue

In [3]:

```
stack = []

# append() function to push
# element in the stack
stack.append('a')
stack.append('b')
stack.append('c')
stack.append(10)
stack.append(20)

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)
```

```
Initial stack
['a', 'b', 'c', 10, 20]

Elements poped from stack:
20
10
c

Stack after elements are poped:
['a', 'b']
```

In [1]:

```
# Stack implementation using List
stack=[]
def push():
    if len(stack)==size: # check wether the stack is full or not
        print("Stack is Full!!!!")
    else:

        element=input("Enter the element:")
        stack.append(element)
        print(stack)
def pop_element():
    if not stack:# or if len(stack)==0
        print("Stack is Empty!!!")
    else:
        e=stack.pop()
        print("element removed!!:",e)
        print(stack)
def display():
    print(stack)
size=int(input("Enter the size of Stack:"))
while True:
    print("Select the Operation:\n 1.Push \n 2.Pop \n 3. Display \n 4. Quit")
    ch=int(input())
    if ch==1:
        push()
    elif ch==2:
        pop_element()
    elif ch==3:
        display()
    elif ch==4:
        break
    else:
        print("Invalid Option!!!")
```

```
Enter the size of Stack:5
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
3
[]
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Enter the element:1000
['1000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Enter the element:AAA
['1000', 'AAA']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Enter the element:2000
['1000', 'AAA', '2000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Enter the element:5000
['1000', 'AAA', '2000', '5000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Enter the element:10
['1000', 'AAA', '2000', '5000', '10']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
1
Stack is Full!!!!
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
2
```

```
element removed!!: 10
['1000', 'AAA', '2000', '5000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
2
element removed!!: 5000
['1000', 'AAA', '2000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
3
['1000', 'AAA', '2000']
Select the Operation:
 1.Push
 2.Pop
 3. Display
 4. Quit
4
```

In [2]:

```
# Python program to
# demonstrate stack implementation
# using collections.deque


from collections import deque

stack = deque()

# append() function to push
# element in the 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)
```

```
Initial stack:
deque(['a', 'b', 'c'])

Elements poped from stack:
c
b
a

Stack after elements are poped:
deque([])
```

In [3]:

```
# Python program to
# demonstrate stack implementation
# using queue module


from queue import LifoQueue

# Initializing a stack
stack = LifoQueue(maxsize = 4)

# qsize() show the number of elements
# in the stack
print(stack.qsize())

# put() function to push
# element in the stack
stack.put('a')
stack.put('b')
stack.put('c')

print("Full: ", stack.full())
print("Size: ", stack.qsize())

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

print("\nEmpty: ", stack.empty())
```

```
0
Full:  False
Size:  3

Elements poped from the stack
c
b
a

Empty:  True
```

Queue

Queue is a linear data structure that stores items in First-In/First Out(FIFO) manner. The item which is inserted first will removed first.

Operations

- Enqueue:Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition – Time Complexity : O(1)
- Dequeue: Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition – Time Complexity : O(1)
- Front: Get the front item from queue – Time Complexity : O(1)
- Rear: Get the last item from queue – Time Complexity : O(1)

Implementation

Queue in Python can be implemented by the following ways:

- list
- collections.deque
- queue.Queue

In [7]:

```
# implementing Queue using List Method1:
from collections import deque
q=deque()
q.append(10)
q.append(100)
q.append(1000)
q.append(10000)
print("Initial Queue is:",q)
print(q.popleft())
print(q.popleft())
print("After Removing elements:",q)
```

```
Initial Queue is: deque([10, 100, 1000, 10000])
10
100
After Removing elements: deque([1000, 10000])
```

In [8]:

```
# implementing Queue using List Method2:
from collections import deque
q=deque()
q.appendleft(10000)
q.appendleft(1000)
q.appendleft(100)
q.appendleft(10)
print("Initial Queue is:",q)
print(q.pop())
print(q.pop())
print("After Removing elements:",q)
```

```
Initial Queue is: deque([10, 100, 1000, 10000])
10000
1000
After Removing elements: deque([10, 100])
```

In [9]:

```
# implementing Queue using List Method3:
q=[]
q.append(10)
q.append(100)
q.append(1000)
q.append(10000)
print("Initial Queue is:",q)
print(q.pop(0))
print(q.pop(0))
print(q.pop(0))
print("After Removing elements:",q)
```

```
Initial Queue is: [10, 100, 1000, 10000]
10
100
1000
After Removing elements: [10000]
```

In [10]:

```
# implementing Queue using List Method4:
q=[]
q.insert(0,10)
q.insert(0,20)
q.insert(0,30)
q.insert(0,40)
print("Initial Queue:",q)
print(q.pop())
print(q.pop())
print(q.pop())
print("After removal Queue:",q)
```

```
Initial Queue: [40, 30, 20, 10]
10
20
30
After removal Queue: [40]
```

In [15]:

```
# Implement Queue using List(Functions)
q=[]
def Enqueue():
    if len(q)==size: # check wether the stack is full or not
        print("Queue is Full!!!!")
    else:

        element=input("Enter the element:")
        q.append(element)
        print(element,"is added to the Queue!")
def dequeue():
    if not q:# or if len(stack)==0
        print("Queue is Empty!!!")
    else:
        e=q.pop(0)
        print("element removed!!:",e)
def display():
    print(q)
size=int(input("Enter the size of Queue:"))
while True:
    print("Select the Operation:1.Add  2.Delete  3. Display  4. Quit")
    choice=int(input())
    if choice==1:
        Enqueue()
    elif choice==2:
        dequeue()
    elif choice==3:
        display()
    elif choice==4:
        break
    else:
        print("Invalid Option!!!")
```

```
Enter the size of Queue:3
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
1
Enter the element:A
A is added to the Queue!
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
1
Enter the element:B
B is added to the Queue!
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
1
Enter the element:C
C is added to the Queue!
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
1
Queue is Full!!!!
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
3
['A', 'B', 'C']
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
2
element removed!!: A
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
2
element removed!!: B
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
2
element removed!!: C
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
2
Queue is Empty!!!
Select the Operation:1.Add  2.Delete  3. Display  4. Quit
4
```

In [17]:

```
# implment queue using queue module
from queue import Queue
q=Queue(maxsize=4)
print("Initial Size Before Insertion:",q.qsize())
q.put('A')
q.put('AA')
q.put('AAA')
q.put('AAAA')
print("After Insertion:",q.qsize())
print("Queue is Full or Not:",q.full())
print("Size of Queue:",q.qsize())
print("Removing Elements:")
print(q.get())
print(q.get())
print(q.get())
print("Empty or Not??",q.empty())
print(q.get())
print("Empty or Not??",q.empty())
print("Size of Queue:",q.qsize())
```

```
Initial Size Before Insertion: 0
After Insertion: 4
Queue is Full or Not: True
Size of Queue: 4
Removing Elements:
A
AA
AAA
Empty or Not?? False
AAAA
Empty or Not?? True
Size of Queue: 0
```

In [ ]:

```

```