Assignment -1

## Assignment (**Stack**)

| **Operation** | **Output** | **S (bottom,..., top)** |
| --- | --- | --- |
| S.Push(7) | - | 7 |
| S.Push(10) | - | 7,10 |
| S.Pop() | 10 | 7 |
| S.Pop() | 7 | - |
| S.Push(3) | - | 3 |
| S.Push(5) | - | 3,5 |
| S.Pop() | 5 | 3 |

## Assignment (**Queue)**

Codes -

class EmptyQueue():

def \_\_init\_\_(self):

self.items = []

def push(self, item): # will be enque

self.items.append(item)

def deque(self):

if self.isEmpty():

raise Exception("Queue is empty")

return self.items.pop(0)

def isEmpty(self):

return len(self.items) == 0

# def \_\_str\_\_(self):

# return str(self.items)

myQueue = EmptyQueue()

# a.

myQueue.push(2)

myQueue.push(4)

myQueue.push(3)

myQueue.push(7)

print(\*(f"{item} <-" for item in myQueue.items))

# Result ---- 2 <- 4 <- 3 <- 7 <-

# b. show in two-cell standard notation

print("Front :", myQueue.items[0], "| Back :", myQueue.items[-1])

# Result ---- Front : 2 | Back : 7

# c. show the result after deque -> deque -> front

myQueue.deque()

myQueue.deque()

print("After dequeuing twice: Front is -> ", myQueue.items[0])

# Result ---- After dequeuing twice: Front is -> 3