```
# 1.

class Node:

    def __init__(self, data):
        self.data = data


class Stack:

    def __init__(self):
        self.head = None

    def __del__(self):
      pass

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    def printList(self):
        temp = self.head
        while (temp):
            print(temp.data)
            temp = temp.next

obj1 = Stack()

obj1.push(6)
obj1.push(7);
obj1.push(1);
obj1.push(4)
obj1.printList()
```

```
4
1
7
6
```

```
#2.

class AREA_OF_RECTANGLE_1 :

  def __init__(self , a , b):
    self.a = a
```

```
    self.b = b

  def area(self):
    return self.a * self.b

a1 = AREA_OF_RECTANGLE_1(10 , 90)
a1.area()
```

```
900
```

```
class AREA_OF_RECTANGLE_2 :

  def __init__(self):
    pass

  def area(self , a , b):
    return a * b

a2 = AREA_OF_RECTANGLE_2()
a2.area(10 , 90)
```

```
900
```

```
#3.

class queue1 :

  def __init__(self , size):
    self.queue = []
    self.size = size

  def __del__(self):
    pass

  def enqueue(self , value):
    if(self.isfull() != True) :
      self.queue.insert(0 , value)
    else :
      print("queue is full")

  def dequeue(self):

    if(self.isempty() != True) :
      return self.queue.pop()

    else :
      print("queue is empty")

  def peek(self):
```

```
        if(self.isempty() != True):
            return self.queue[-1]

        else:
            print("Queue is Empty")

    def isempty(self):
      return self.queue == []


    def isfull(self):
        return len(self.queue) == self.size



myQ = queue1(15)
myQ.enqueue(4)
myQ.enqueue(5)
myQ.enqueue(6)
myQ.enqueue(10)
myQ.enqueue(20)

print(myQ.queue)

myQ.enqueue(11)
myQ.enqueue(52)
myQ.enqueue(93)

print(myQ.queue)

myQ.dequeue()

print(myQ.queue)
print(myQ.peek())

myQ.dequeue()

print(myQ.peek())
```

```
[20, 10, 6, 5, 4]
[93, 52, 11, 20, 10, 6, 5, 4]
[93, 52, 11, 20, 10, 6, 5]
5
6
```

✓ 0s completed at 4:47 PM