```
import random
list=[]
n=int(input("enter the length of list:"))
for i in range(n):
    j=int(input("enter the element"))
    list.append(j)
random.shuffle(list)
print("shuffled list:",list)
```

```
enter the length of list: 2
enter the element 1
enter the element 2
```

```
shuffled list: [2, 1]
```

```
class MyQueue(object):

    def __init__(self):
        self.main=[]
        self.temp=[]
    def push(self,x):
        while self.main:
            e=self.main.pop()
            self.temp.append(e)
        self.main.append(x)

        while self.temp:
            k=self.temp.pop()
            self.main.append(k)

    def pop(self):
        return self.main.pop()
    def peek(self):
        if self.main:
            return self.main[-1]
        else:
            'stack is empty'
    def empty(self):
        return len(self.main)==0
```

```
o=MyQueue()
o.push(20)
o.push(90)
o.push(50)
o.push(70)
o.push(60)
```

```
o.peek()
```

```
20
```

```
o.pop()
```

```
20
```

```
o.empty()
```

```
False
```

```
class MyLL:
    class mynodes:
        def __init__(self,data):
            self.data = data
            self.next = None
    def __init__(self):
        self.head=None
    def insertb(self,data):
        new_node =MyLL.mynodes(data)
        new_node.next = self.head
        self.head = new_node
    def inserted(self,data):
        new_node=MyLL.mynodes(data)
        if self.head == None:
            self.head = new_node
            return
            current = self.head
        while current.next:
            current =current.next
            current.next=new_node
    def insertaft(self,data,prev_node):
        new_node = MyLL.mynodes(data)
        new_node.next = prev_node.next
        prev_node.next = new_node
    def print_LL(self):
        current=self.head
        while current:
            print(current.data, end='-->')
            current = current.next
    def search(self,val):
        current=self.head
        while current:
            if current.data ==val:
                return True
                current = current.next
            return False
    def deletefirst(self):
        if self.head == None:
            return
            current=self.head
            while current.next.next:
                current=current.next
                current.next=None
                print(current)
    def delbyval(self,x):
        if self.head.data == x:
            self.head = self.head.next
            current = self.head
        while current.next:
            if current.next.data!=x:
                current = current.next
            else:
                break
        if current.next:
                current.next = current.next.next
        else:
            return 'not found'
```

```
j =MyLL()
j.insertb(55)
j.insertb(40)
j.insertb(90)
j.print_LL()
```

```
90-->40-->55-->
```

```
j.deletefirst()
j.print_LL()
```

```
90-->40-->55-->
```

```
j.delbyval()
j.print_LL()
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
```

```
Cell In[10], line 1
----> 1 j.delbyval()
      2 j.print_LL()

TypeError: MyLL.delbyval() missing 1 required positional argument: 'x'

def del_by_val(self,x):
    if self.head.data == x:
        self.head = self.head.next
        current = self.head
    while current.next:
        if current.next.data!=x:

            current = current.next
        else:
            break
    if current.next:
        current.next = current.next.next
    else:
        return 'not found'
```