Deque

1] Deque in Python :

```
print("**************** Using deque **********")
from _collections import deque
d=deque()
d.append(10)
d.append(20)
d.append(30)
d.appendleft(40)
print(d)
print(d.pop())
print(d.popleft())
print(d)
print()

print("*************** Using a List ***********")
from _collections import deque
d=deque([10,20,30,40])
d.insert(2,10)
print(d.count(10))
d.remove(10)
d.extend([50,60])
print(d)
d.extendleft([15,25])
print(d)
print()
```

OUTPUT :

```
**************** Using deque **********

deque([40, 10, 20, 30])

30

40

deque([10, 20])
```

```
*************** Using a List ***********

2

deque([20, 10, 30, 40, 50, 60])

deque([25, 15, 20, 10, 30, 40, 50, 60])
```

2] Using a List Doing a Rotation :

```python
from _collections import deque
d=deque([10,20,30,40,50])
d.rotate(2)  # right roatation (positive)
print(d)
d.rotate(-2)  #left roataion (negative)
print(d)
d.reverse()
print(d)
```

OUTPUT :

```
deque([40, 50, 10, 20, 30])

deque([10, 20, 30, 40, 50])

deque([50, 40, 30, 20, 10])
```

3] Index and access the index :

```
from _collections import deque
d=deque([10,20,30,40,50])
print(d[2])
d[2]=100
print(d)
print(d[0])
print(d[-1])
```

OUTPUT :

```
30

deque([10, 20, 100, 40, 50])

10

50
```

4] Design Data Structure with min and max operation :

```
from _collections import deque

class Myds:
    def __init__(self):
        self.dq=deque()

    def insertMin(self,x):
        self.dq.appendleft(x)

    def insertMax(self,x):
        self.dq.append(x)

    def extractMin(self):
        return self.dq.popleft()

    def extractMax(self):
        return self.dq.pop()

    def getMin(self):
        return self.dq[0]

    def getMax(self):
        return self.dq[-1]

    def printds(self):
        print(self.dq)

d=Myds()
d.insertMin(10)
d.printds()
d.insertMin(5)
d.printds()
d.insertMax(20)
d.printds()
d.insertMin(3)
d.printds()
print(d.extractMin())
d.printds()
print(d.extractMax())
d.printds()
print(d.getMin())
print(d.getMax())
d.printds()
```

OUTPUT :

```
deque([10])

deque([5, 10])

deque([5, 10, 20])

deque([3, 5, 10, 20])

3

deque([5, 10, 20])

20

deque([5, 10])

5

10

deque([5, 10])
```

5] Link List Implementation of deque :

```
class Node:
   def __init__(self,k):
      self.key=k
      self.next=None
      self.prev=None

class MyDeque:
   def __init__(self,c):
      self.front=None
      self.rear=None
      self.sz=0

   def size(self):
      return self.sz

   def isEmpty(self):
      return self.sz==0

   def inserRear(self,x):
      temp=Node(x)
      if self.rear==None:
         self.front=temp
      else:
         self.rear.next=temp
         temp.prev=self.rear
      self.rear=temp
      self.sz=self.sz+1

   def deletefront(self):
      if self.front==None:
         return None
      else:
         res=self.front.key
         self.front=self.front.next
         if self.front==None:
            self.rear=None

         else:
            self.front.prev=None
         self.sz=self.sz-1

         return res

   def getFront(self):
        if self.front:
            return self.front.key

    def getRear(self):
        if self.rear:
            return self.rear.key

#main
dq=MyDeque(3)

print(dq.isEmpty())
dq.inserRear(10)
print(dq.getFront(),dq.getRear())
dq.inserRear(20)
print(dq.getFront(),dq.getRear())
dq.inserRear(30)
print(dq.getFront(),dq.getRear())
dq.deletefront()
print(dq.getFront(),dq.getRear())
```

OUTPUT :

```
True

10 10

10 20

10 30

20 30
```

6] List implementation of deque in python :

```python
class MyDeque:
    def __init__(self,c):
        self.l=[None]*c
        self.cap=c
        self.size=0
        self.front=0

    def deleteFront(self):
        if self.size==0:
            return None
        else:
            res=self.l[self.front]
            self.front=(self.front+1)%self.cap
            self.size=self.size-1

            return res

    def insertFront(self,x):
        if self.size==self.cap:
            return
        else:
            self.front=(self.front-1)%self.cap
            self.l[self.front]=x
            self.size=self.size+1

    def insertRear(self,x):
        if self.size==self.cap:
            return
        new_rear=(self.front+self.size)%self.cap
        self.l[new_rear]=x
        self.size=self.size+1

    def deleteRear(self):
        sz=self.size
        if sz==0:
            return
        else:
            rear=(self.front+sz-1)%self.cap
            self.sizes=sz-1
            return self.l[rear]

    def frontEle(self):
        return self.l[self.front]
    def rearEle(self):
        rear=(self.front+self.size-1)%self.cap
        return self.l[rear]

dq=MyDeque(4)

dq.insertRear(10)
print(dq.frontEle(),dq.rearEle())
dq.insertFront(20)
print(dq.frontEle(),dq.rearEle())
dq.insertFront(30)
print(dq.frontEle(),dq.rearEle())
dq.deleteRear()
print(dq.frontEle(),dq.rearEle())
dq.insertRear(40)
print(dq.frontEle(),dq.rearEle())
dq.deleteRear()
print(dq.frontEle(),dq.rearEle())
```

OUTPUT :

```
10 10

20 10

30 10

30 10

30 40

30 40
```