Implementation of Stack

In [1]:
```
class Stack(object):

    def __init__(self):

        self.items = []

    def is_Empty(self):

        return self.items == []

    def push(self,items):

        self.items.append(items)

    def pop(self):

        return self.items.pop()

    def peek(self):

        return self.items[len(self.items)-1]

    def size(self):

        return len(self.items)
```

In [2]:
```
s = Stack()
```

In [3]:
```
print (s.is_Empty())
```

```
True
```

In [4]:
```
s.push(1)
```

In [5]:
```
s.push('two')
```

In [6]:
```
s.peek()
```

Out[6]:
```
'two'
```

In [7]:
```
s.push('three')
```

In [8]:
```
s.push(4)
```

In [9]:
```
s.peek()
```

Out[9]:
```
4
```

In [10]:
```
s.size()
```

Out[10]: 4

In [11]:
```
s.pop()
```

Out[11]: 4

In [12]:
```
s.size()
```

Out[12]: 3

In [13]:
```
s.pop()
```

Out[13]: 'three'

In [14]:
```
s.pop()
```

Out[14]: 'two'

In [15]:
```
s.peek()
```

Out[15]: 1

In [16]:
```
s.pop()
```

Out[16]: 1

In [17]:
```
s.is_Empty()
```

Out[17]: True

Implementation of Queue:

In [20]:
```
class Queue(object):

    def __init__(self):

        self.items = []

    def is_Empty(self):

        return self.items == []

    def enqueue(self,items):

        self.items.insert(0,items)

    def dequeue(self):

        return self.items.pop()

    def size(self):

        return len(self.items)
```

In [21]:
```
q = Queue()
```

In [22]:
```
q.size()
```

Out[22]:
```
0
```

In [23]:
```
q.is_Empty()
```

Out[23]:
```
True
```

In [24]:
```
q.enqueue(1)
```

In [25]:
```
q.enqueue('two')
q.enqueue(3.01)
```

In [26]:
```
q.size()
```

Out[26]:
```
3
```

In [27]:
```
q.dequeue()
```

Out[27]:
```
1
```

Implementation of a Dequeue:

In [28]:
```
#For Dequeue, addition and deletion of items can happen at either end(front/rear)
#It doesnt follow LIFO/FIFO concept
```

In [59]:
```
class Dequeue(object):

    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def size(self):
        return len(self.items)

    def add_front(self,items):
        self.items.insert(0,items)

    def add_rear(self,items):
        self.items.append(items)

    def remove_front(self):
        return self.items.pop(0)

    def remove_rear(self):
        return self.items.pop()
```

In [60]:
```
d = Dequeue()
```

In [61]:
```
d.add_front(1)
```

```
d.add_rear(2)
```

In [62]:
```
d.size()
```

Out[62]:
```
2
```

In [63]:
```
print(d.remove_front() , " " , d.remove_rear())
```

```
1   2
```

In [64]:
```
d.size()
```

Out[64]:
```
0
```

In [67]:
```
d.is_empty()
```

Out[67]:
```
True
```

In [ ]:
```

```