Linked List

- Creat a LinkList [L]
- len [L]
- insert fron head [L]
- transverse/print [L]
- insert from tail (append) [L]
- insert in middle (after) [L]

- clear (Empty) [L]

- delete from head [L]
- delete from tail (pop) [L]
- delete by value (remove) [L]

- search by value(find) [L]

- delete by index (del L[0])
- search by index(indexing) L[1]

In [1]:

```
class Node:
    def __init__(self,value):
        self.data=value
        self.next=None
```

In [2]:

```
#Head=None it means linklist is empty
#length of linklist is number of nodes
# in Linklist there are 4 main operations 1.Insert 2.Traverse 3.Delete 4.Search
```

In [51]:

```
class LinkList:
    def __init__(self):

        #Empty LinkList
        self.head=None
        self.n=0
    #Len
    def __len__(self):
        return self.n

    #Insert
    def insert_head(self,value):
        #new node
        new_node=Node(value)

        #creat Connection
        new_node.next=self.head

        #reassign head
        self.head=new_node

        #increment n
        self.n=self.n+1
    #print
    def __str__(self):
        curr = self.head

        result=''
        while curr != None:
            result=result+str(curr.data)+'->'
            curr=curr.next
        return result[:-2]

    #Append
    #traverse to the last node
    #set the next of tail to new node
    def append(self,value):
        new_node=Node(value)
        if self.head==None:
            #empty
            self.head=new_node
            self.n=self.n+1
            return

        curr=self.head

        while curr.next != None:
            curr = curr.next

        #you are at the lasr node
        curr.next=new_node
        self.n=self.n+1
    #Insert
    def insert_after(self,after,value):
        new_node=Node(value)

        curr = self.head
        while curr!=None:
            if curr.data==after:
                break
            curr=curr.next

        if curr !=None:
            #logic
            new_node.next=curr.next
            curr.next=new_node
            self.n=self.n+1
        else:
            return "Item not found"
    #clear
    def clear(self):
        self.head=None
        self.n=0
    #delete head
    def delete_head(self):
        if self.head==None:
            #empty
            return "Empty LinkList"
        self.head=self.head.next
        self.n=self.n-1
    #pop
    def pop(self):
```

```python
        if self.head==None:
            #empty
            return "Empty Linklist"
        curr=self.head

        if curr.next==None:
            #delete from head
            return self.delete_head()

        while curr.next.next!=None:
            curr=curr.next

        #curr is the 2nd last node
        curr.next=None
        self.n=self.n-1

    #remove
    def remove(self,value):

        if self.head==None:
            return 'Empty LinkList'
        if self.head.data==value:
            #you want to remove head node
            return self.delete_head()
        curr = self.head

        while curr.next != None:
            if curr.next.next == value:
                break
            curr = curr.next


            if curr.next==None:
                #item not found
                return 'Not Found'
            else:
                curr.next=curr.next.next
    #Search
    def search(self,item):
        curr=self.head
        pos=0

        while curr !=None:
            if curr.data == item:
                return pos
            curr = curr.next
            pos=pos+1

        return 'Not Found'

    #Get Item
    def __getitem__(self,index):
        curr=self.head
        pos=0

        while curr !=None:
            if pos == index:
                return curr.data
            curr=curr.next
            pos=pos+1
        return 'IndexError'
```

In [52]:

```
L=LinkList()
```

In [53]:

```
L.insert_head(1)
L.insert_head(2)
L.insert_head(3)
L.insert_head(4)
L.insert_head(5)
L.insert_head(6)
L.insert_head(7)
L.insert_head(8)
L.insert_head(9)
L.insert_head(10)
```

In [54]:

```
print(L)
```

```
10->9->8->7->6->5->4->3->2->1
```

In [55]:

```
L[6]
```

Out[55]:

```
4
```

In [56]:

```
L.search(5)
```

Out[56]:

```
5
```

In [57]:

```
L.search(6)
```

Out[57]:

```
4
```

In [58]:

```
L.remove(10)
print(L)
```

```
9->8->7->6->5->4->3->2->1
```

In [59]:

```
L.pop()
print(L)
```

```
9->8->7->6->5->4->3->2
```

In [60]:

```
L.delete_head()
print(L)
```

```
8->7->6->5->4->3->2
```

In [61]:

```
L.insert_after(2,200)
print(L)
```

```
8->7->6->5->4->3->2->200
```

In [62]:

```
L.append(10)
print(L)
```

```
8->7->6->5->4->3->2->200->10
```

In [63]:

```
L.clear()
print(L)
```

In [64]:

```
print(L)
```

In [ ]: