Write a Class to implement a Singly Linked List .

In [43]:

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

    def addElement(self, data):

        new_node = Node(data)
        if  self.head is None:
            self.head=new_node
            return
        temp=self.head
        while(temp.next!=None):
            temp=temp.next
        temp.next=new_node
        return
        temp.next= new_node


    def DeleteElement(self,key):
        temp=self.head
        if(temp!=None):
            if(temp.data==key):
                self.head=temp.next
                temp = None
                return
        while(temp!=None):
            if(temp.data==key):
                break
            pre = temp
            temp = temp.next
        if(temp==None):
            print("Element is Not Present in Linked List for Deletion ",-1)
        if (temp==None):
            return
        pre.next=temp.next
        temp=None

    def DeletePosition(self,position):
        if(self.head==None):
            return
        temp=self.head
        if(position==0):
            self.head=temp.next
            temp=None
            return
        for i in range(position-1):
            temp=temp.next
            if(temp==None):
                break
        if(temp==None):
            return
        if(temp.next==None):
            return
        pre = temp
        temp = temp.next
```

```python
        if (temp==None):
            return
        pre.next=temp.next
        temp=None

    def Search(self,key):
        if(self.head==None):
            print("-1")
            return

        temp=self.head
        while(temp!=None):
            if(temp.data==key):
                print("Element is " + str(temp.data) +" presented")
            temp=temp.next
        return


    def length(self):
        len1=0
        if(self.head==None):
            return

        temp=self.head
        while(temp!=None):
            temp=temp.next
            len1+=1
        print("Linked list Length ",len1)
        return

    def print1(self):
        temp=self.head
        while(temp):
            print(temp.data,end=" ")
            temp=temp.next
        print()

list1 = CreateLinkList()
list1.addElement(1)
list1.addElement(2)
list1.addElement(3)
list1.addElement(4)
print("Created Linked List",end=" ")
list1.print1()
list1.DeleteElement(5)
print("After Delete Linked List",end=" ")
list1.print1()
list1.Search(3)
list1.length()
list1.DeletePosition(3)
print("Delete the element by position ",end=" ")
list1.print1()
```

```
Created Linked List 1 2 3 4
Element is Not Present in Linked List for Deletion  -1
After Delete Linked List 1 2 3 4
Element is 3 presented
Linked list Length  4
Delete the element by position  1 2 3
```

Write a Class to implement a Doubly Linked List .

In [21]:

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None
class CreateLinkList:
    def __init__(self):
        self.head=None

    def addElement(self, data):

        new_node = Node(data)
        new_node.next=None
        if  self.head is None:
            new_node.prev=None
            self.head=new_node
            return
        temp=self.head
        while(temp.next!=None):
            temp=temp.next
        temp.next=new_node
        new_node.prev=temp

    def print1(self):
        temp=self.head
        while(temp):
            print(temp.data,end=" ")
            temp=temp.next
        print()


    def DeleteElement(self,key):
        temp=self.head
        if(self.head==None or key is None):
            return

        if(temp!=None):
            if(temp.data==key):
                temp.prev=None
                self.head=temp.next
                temp = None
                return
            while(temp!=None):
                if(temp.data==key):
                    break
                pre = temp
                temp = temp.next
            if(temp==None):
                print("Element is Not Present in Linked List for Deletion ",-1)
            if (temp==None):
                return
            pre.prev=temp
            pre.next=temp.next
            temp=None


    def DeletePosition(self,position):
        if(self.head==None):
            return
```

```python
        temp=self.head
        if(position==0):
            self.head=temp.next
            temp=None
            return
        for i in range(position-1):
            temp=temp.next
            if(temp==None):
                break
        if(temp==None):
            return
        if(temp.next==None):
            return
        pre = temp
        temp = temp.next
        if (temp==None):
            return
        pre.prev=temp
        pre.next=temp.next
        temp=None

    def Search(self,key):
        if(self.head==None):
            print("-1")
            return

        temp=self.head
        while(temp!=None):
            if(temp.data==key):
                print("Element is " + str(temp.data) +" presented")
            temp=temp.next
        return
    def length(self):
        len1=0
        if(self.head==None):
            return

        temp=self.head
        while(temp!=None):
            temp=temp.next
            len1+=1
        print("Linked list Length ",len1)
        return

list1 = CreateLinkList()
list1.addElement(1)
list1.addElement(2)
list1.addElement(3)
list1.addElement(4)
list1.addElement(5)
print("Created Linked List",end=" ")
list1.print1()
list1.DeleteElement(2)
print("After Delete Linked List",end=" ")
list1.print1()
list1.DeletePosition(2)
print("Delete the element by position ",end=" ")
list1.print1()
list1.Search(3)
list1.length()

```

```
Created Linked List 1 2 3 4 5
After Delete Linked List 1 3 4 5
Delete the element by position  1 3 5
Element is 3 presented
Linked list Length  3
```

Write a Class to implement a Stack of Integers using Linked List .

In [16]:

```
class Node:

    def __init__(self,data):
        self.data = data
        self.next = None

class Stack:
    def __init__(self):
        self.head = None
    def isempty(self):
        if self.head == None:
            return True
        else:
            return False
    def push(self,data):

        if self.head == None:
            self.head=Node(data)
        else:
            newnode = Node(data)
            newnode.next = self.head
            self.head = newnode
    def pop(self):

        if self.isempty():
            return None

        else:
            poppednode = self.head
            self.head = self.head.next
            poppednode.next = None
            return poppednode.data
    def getTop(self):

        if self.isempty():
            return None

        else:
            return self.head.data
    def print1(self):

        iternode = self.head
        if self.isempty():
            print("Stack Underflow")

        else:

            while(iternode != None):

                print(iternode.data,end = " ")
                iternode = iternode.next
            return
s = Stack()
s.push(1)
s.push(2)
s.push(3)
s.push(4)
s.push(5)
s.push(6)
```

```
print("Create a Stack:-",end=" ")
s.print1()
print()
print("Top of stack:-",s.getTop())
print("Pop Element:-",s.pop())

```

```
Create a Stack:- 6 5 4 3 2 1
Top of stack:- 6
Pop Element:- 6
```

In [ ]:

```

```