

**Write a Class to implement a Singly Linked List .**

In [43]:

```
1 class Node:
2     def __init__(self,data):
3         self.data=data
4         self.next=None
5 class CreateLinkedList:
6     def __init__(self):
7         self.head=None
8
9     def addElement(self, data):
10
11         new_node = Node(data)
12         if self.head is None:
13             self.head=new_node
14             return
15         temp=self.head
16         while(temp.next!=None):
17             temp=temp.next
18         temp.next=new_node
19         return
20         temp.next= new_node
21
22
23     def DeleteElement(self,key):
24         temp=self.head
25         if(temp!=None):
26             if(temp.data==key):
27                 self.head=temp.next
28                 temp = None
29                 return
30         while(temp!=None):
31             if(temp.data==key):
32                 break
33             pre = temp
34             temp = temp.next
35         if(temp==None):
36             print("Element is Not Present in Linked List for Deletion ",-1)
37         if (temp==None):
38             return
39         pre.next=temp.next
40         temp=None
41
42     def DeletePosition(self,position):
43         if(self.head==None):
44             return
45         temp=self.head
46         if(position==0):
47             self.head=temp.next
48             temp=None
49             return
50         for i in range(position-1):
51             temp=temp.next
52             if(temp==None):
53                 break
54         if(temp==None):
55             return
56         if(temp.next==None):
57             return
58         pre = temp
59         temp = temp.next
```

```

60         if (temp==None):
61             return
62         pre.next=temp.next
63         temp=None
64
65     def Search(self,key):
66         if(self.head==None):
67             print("-1")
68             return
69
70         temp=self.head
71         while(temp!=None):
72             if(temp.data==key):
73                 print("Element is " + str(temp.data) + " presented")
74                 temp=temp.next
75             return
76
77
78     def length(self):
79         len1=0
80         if(self.head==None):
81             return
82
83         temp=self.head
84         while(temp!=None):
85             temp=temp.next
86             len1+=1
87         print("Linked list Length ",len1)
88         return
89
90     def print1(self):
91         temp=self.head
92         while(temp):
93             print(temp.data,end=" ")
94             temp=temp.next
95         print()
96
97 list1 = CreateLinkList()
98 list1.addElement(1)
99 list1.addElement(2)
100 list1.addElement(3)
101 list1.addElement(4)
102 print("Created Linked List",end=" ")
103 list1.print1()
104 list1.DeleteElement(5)
105 print("After Delete Linked List",end=" ")
106 list1.print1()
107 list1.Search(3)
108 list1.length()
109 list1.DeletePosition(3)
110 print("Delete the element by position ",end=" ")
111 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]:

```
1 class Node:
2     def __init__(self,data):
3         self.data=data
4         self.next=None
5         self.prev=None
6 class CreateLinkedList:
7     def __init__(self):
8         self.head=None
9
10    def addElement(self, data):
11
12        new_node = Node(data)
13        new_node.next=None
14        if self.head is None:
15            new_node.prev=None
16            self.head=new_node
17            return
18        temp=self.head
19        while(temp.next!=None):
20            temp=temp.next
21        temp.next=new_node
22        new_node.prev=temp
23
24    def printl(self):
25        temp=self.head
26        while(temp):
27            print(temp.data,end=" ")
28            temp=temp.next
29        print()
30
31
32    def DeleteElement(self,key):
33        temp=self.head
34        if(self.head==None or key is None):
35            return
36
37        if(temp!=None):
38            if(temp.data==key):
39                temp.prev=None
40                self.head=temp.next
41                temp = None
42                return
43            while(temp!=None):
44                if(temp.data==key):
45                    break
46                pre = temp
47                temp = temp.next
48            if(temp==None):
49                print("Element is Not Present in Linked List for Deletion ",-1)
50            if (temp==None):
51                return
52            pre.prev=temp
53            pre.next=temp.next
54            temp=None
55
56
57    def DeletePosition(self,position):
58        if(self.head==None):
59            return
```

```

60     temp=self.head
61     if(position==0):
62         self.head=temp.next
63         temp=None
64         return
65     for i in range(position-1):
66         temp=temp.next
67         if(temp==None):
68             break
69     if(temp==None):
70         return
71     if(temp.next==None):
72         return
73     pre = temp
74     temp = temp.next
75     if (temp==None):
76         return
77     pre.prev=temp
78     pre.next=temp.next
79     temp=None
80
81     def Search(self,key):
82         if(self.head==None):
83             print("-1")
84             return
85
86         temp=self.head
87         while(temp!=None):
88             if(temp.data==key):
89                 print("Element is " + str(temp.data) + " presented")
90                 temp=temp.next
91         return
92     def length(self):
93         len1=0
94         if(self.head==None):
95             return
96
97         temp=self.head
98         while(temp!=None):
99             temp=temp.next
100             len1+=1
101         print("Linked list Length ",len1)
102         return
103
104 list1 = CreateLinkList()
105 list1.addElement(1)
106 list1.addElement(2)
107 list1.addElement(3)
108 list1.addElement(4)
109 list1.addElement(5)
110 print("Created Linked List",end=" ")
111 list1.print1()
112 list1.DeleteElement(2)
113 print("After Delete Linked List",end=" ")
114 list1.print1()
115 list1.DeletePosition(2)
116 print("Delete the element by position ",end=" ")
117 list1.print1()
118 list1.Search(3)
119 list1.length()
120

```

```
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]:

```
1 class Node:
2
3     def __init__(self,data):
4         self.data = data
5         self.next = None
6
7 class Stack:
8     def __init__(self):
9         self.head = None
10    def isempty(self):
11        if self.head == None:
12            return True
13        else:
14            return False
15    def push(self,data):
16
17        if self.head == None:
18            self.head=Node(data)
19        else:
20            newnode = Node(data)
21            newnode.next = self.head
22            self.head = newnode
23    def pop(self):
24
25        if self.isempty():
26            return None
27
28        else:
29            poppednode = self.head
30            self.head = self.head.next
31            poppednode.next = None
32            return poppednode.data
33    def getTop(self):
34
35        if self.isempty():
36            return None
37
38        else:
39            return self.head.data
40    def print1(self):
41
42        iternode = self.head
43        if self.isempty():
44            print("Stack Underflow")
45
46        else:
47
48            while(iternode != None):
49
50                print(iternode.data,end = " ")
51                iternode = iternode.next
52            return
53 s = Stack()
54 s.push(1)
55 s.push(2)
56 s.push(3)
57 s.push(4)
58 s.push(5)
59 s.push(6)
```



```
60 print("Create a Stack:-",end=" ")
61 s.print1()
62 print()
63 print("Top of stack:-",s.getTop())
64 print("Pop Element:-",s.pop())
65
```

Create a Stack:- 6 5 4 3 2 1

Top of stack:- 6

Pop Element:- 6

In [ ]:

1