Singly/Single Linked List

- Node
- head
- data
- ref/address #### Operations in LL
- Adding/Inserting
- Deleting/Removing
- Traversal
- Adding/Inserting(Insert at Begining,End and In between)
- Deleting(delete first node,last node and In between)

In [20]:

```
# Crearting a Node
class Node:
    def __init__(self,data):
        self.data=data
        self.ref=None
# Creating Linked List
class LinkedList:
    def __init__(self):
        self.head=None
# printing or traversal Operation
    def Traversal(self):
        if self.head is None:# self.head==None
            print("Linked List is Empty!!!")
        else:
            h=self.head
            while h is not None:
                print(h.data,"----->",end=" ")
                h=h.ref
# Inserting or Adding element at begin
    def add_begin(self,data):
        new_node=Node(data) # creating a new node
        new_node.ref=self.head # head ref is pointing to new ref
        self.head=new_node # link to new node from head
    def add_end(self,data):
        new_node=Node(data) # Creating a new Node
        if self.head is None: # Before adding element at end,first check wether the LL
 is empty or not?
            self.head=new_node # new_ node will be assigned to head
        else:
            h=self.head
            while h.ref is not None: # Moving to last node
                h=h.ref
            h.ref=new_node # new node address is stored in last node ref
# In -between Insertion
# After a Node ---insert an element
    def add_after(self,data,x):
        h=self.head
        while h is not None:
            if x==h.data:
                break
            h=h.ref
        if h is None:
            print("Node is Not Present in Linked List!!!")
        else:
            new_node=Node(data) # Creating a new node for ninsertion
            new_node.ref=h.ref # New node ref is pointing to the next node
            h.ref=new_node # new_node ref will copied prev(x) node ref
# Before a node ----insert an element
    def add_before(self,data,x):
        if self.head is None:
            print("Linked List is Empty You can't Insert!!!")
            return
        if self.head.data==x:
            new_node=Node(data) # creating a new node
            new_node.ref=self.head # head ref is pointing to new ref
            self.head=new_node # link to new node from head
            return
        h=self.head
        while h.ref is not None:
```

```
            if h.ref.data==x:
                break
            h=h.ref
        if h.ref is None:
            print("Node is Not Found to insert an element!!!")
        else:
            new_node=Node(data)
            new_node.ref=h.ref
            h.ref=new_node
# Deletion ---delete an first node
    def del_begin(self):
        if self.head is None:
            print("Linked List is Empty you can't delete!!!!")
        else:
            self.head=self.head.ref
# Delete a node at end
    def del_end(self):
        if self.head is None:
            print("Linked List is Empty you can't delete!!!!")
        else:
            h=self.head
            while h.ref.ref is not None:
                h=h.ref
            h.ref=None
# delete  by a value
    def del_value(self,x):
        if self.head is None:
            print("Linked List is Empty you can't delete!!!!")
            return
        if x==self.head.data:
            self.head=self.head.ref
        else:
            h=self.head
            while h.ref is not None:
                if x==h.ref.data:
                    break
                h=h.ref
            if h.ref is None:
                print("Node is Not Found!!!")
            else:
                h.ref=h.ref.ref




ll=LinkedList()
ll.add_begin(20)
ll.add_begin(10)
ll.add_end(100)
ll.add_after(1000,10)
ll.add_before(500,10)
ll.add_before(999,1000)
ll.del_begin()
ll.del_end()
ll.del_value(1000)
ll.Traversal()
```

```
10 -----> 999 -----> 20 ----->
```