What is Linkedlist

In [ ]:
```
--> Linkedlist is a linear datastructure.
--> It is a collections of nodes.
--> Nodes:Combination of Data and next node address
          Nodes --> Data|Adress of next Node

--> For each and every node we will store two things data and address of next node
    --> Head --> First node of a linkedlist is known as head
    --> Tail -->Last node of a linkedlist is known as tail(Address of Tail is always be None)
```

Steps for implementation of linkedlist

In [ ]:
```
--> Create a node --> Data and next
--> Make Connections
```

Implementation of Linkedlist

In [1]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None

c1=Node(10)
print(c1.data)
print(c1.next)
c2=Node(20)
print(c2.next)
print(c2.data)
c1.next=c2
print(c1.next)
print(c2)
c3=Node(30)
print(c3.data,c3.next)
print(c2.next)
c2.next=c3
print(c2.next,c3)
```

```
10
None
None
20
<__main__.Node object at 0x000001B548483580>
<__main__.Node object at 0x000001B548483580>
30 None
None
<__main__.Node object at 0x000001B54848 37C0> <__main__.Node object at 0x000001B54848 37C0>
```

Traversal (Printing) of a Linkedlist

In [2]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def printll(head):
    while head is not None:
        print(str(head.data)+"-->",end=" ")
        head=head.next
    print("None")

c1=Node(10)
c2=Node(20)
c3=Node(30)
c4=Node(40)
c1.next=c2
c2.next=c3
c3.next=c4
printll(c2)
```

```
20--> 30--> 40--> None
```

In [3]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def printll(head):
    while head is not None:
        print(head.data)
        head=head.next

c1=Node(10)
c2=Node(20)
c3=Node(30)
c4=Node(40)
c1.next=c2
c2.next=c3
c3.next=c4
printll(c2)
```

```
20
30
40
```

Operations of Linkedlist

In [ ]:
```
Basic Operations of Linkedlist:
    1.Insertion --> Inserting an element in linkedlist(start,end,pos)
    2.Deletion --> Deletion of element in  linkedlist(start,end,pos)
    3.Traversal  -->(Accessing) Visiting each and every element of the linkedlist.
```

Insertion Operations

Insertion at Beginning

Steps for Insertion at the Beginning

In [ ]:
```
#Insertion of node at begining.
1.Create a node
2.Make the connection with exsiting linkedlist
3.Change the head to the new node
```

Implementation

In [7]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def printll(head):
    while head is not None:
        print(str(head.data)+"-->",end="")
        head=head.next
    print("None")

def insert_at_begin(data,head):
    newNode=Node(data)
    newNode.next=head
    head=newNode
    return head

x=Node(10)
y=Node(20)
z=Node(30)
a=Node(40)
y.next=x
x.next=z
z.next=a
printll(y)
ll4=insert_at_begin(100,y)
printll(ll4)
```

```
20-->10-->30-->40-->None
100-->20-->10-->30-->40-->None
```

Insertion at Given Position

Steps for Insertion at the given position

In [ ]:
```
#Insertion at given pos
1.Create a node
2.Take two pointers prev and next node
3.Use while loop and move to that position
3.Prev.next=newNode
4.newNode.next=next
5.return head
```

Implementation

In [8]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def printll(head):
    while head is not None:
        print(str(head.data)+"-->",end="")
        head=head.next
    print("None")

def insert_at_pos(data,head,pos):
    prev=None
    curr=head
    i=0
    while i<pos:
        prev=curr
        curr=curr.next
        i=i+1
    newNode=Node(data)
    prev.next=newNode
    newNode.next=curr
    return head

x=Node(10)
y=Node(20)
z=Node(30)
a=Node(40)
y.next=x
x.next=z
z.next=a
printll(y)
ll4=insert_at_pos(100,y,2)
printll(ll4)
```

```
20-->10-->30-->40-->None
20-->10-->100-->30-->40-->None
```