Linked List

1] Simple Linked List Implementation in python

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None


temp1=Node(10)
temp2=Node(20)
temp3=Node(30)

temp1.next=temp2
temp2.next=temp3

head=temp1
print(head.key)
print(head.next.key)
print(head.next.next.key)
```

OUTPUT:

10

20

30

2] Traversing Linked List In Python

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


def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next

head=Node(10)
head.next=Node(20)
head.next.next=Node(15)
head.next.next.next=Node(30)

printList(head)
```

OUTPUT :

10 20 15 30

3] Search Linked List :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None


def search(head,x):
    curr=head
    pos=1
    while curr!=Node:
        if curr.key==x:
            return pos
        pos+=1
        curr=curr.next

head=Node(10)
head.next=Node(15)
head.next.next=Node(20)
head.next.next.next=Node(25)
x=20
print(search(head,x))
```

OUTPUT :

3

4] Insert at beginning of linked list in Python :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=Node

def insertBegin(head,key):
    temp=Node(key)
    temp.next=head
    return temp

head=None
head=insertBegin(head,10)
head=insertBegin(head,20)
head=insertBegin(head,30)

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next

printList(head)
```

OUTPUT :

30 20 10

5] Insert at end of Linked List :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def insertEnd(head,key):
    temp = Node(key)
    if head==None:
        return temp

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

    curr.next=temp
    return head

head=None
head=insertEnd(head,10)
head=insertEnd(head,20)
head=insertEnd(head,30)

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next

printList(head)
```

OUTPUT :

10 20 30

6] Insert at given position in linked list :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def insertPos(head,data,pos):
    temp=Node(data)
    if pos==1:
        temp.next=head
        return temp

    curr=head
    for i in range(pos-2):
        curr=curr.next
        if curr==None:
            return head
    temp.next=curr.next
    curr.next=temp

    return head

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)
head.next.next.next.next=Node(50)

printList(head)

head=insertPos(head,45,4)

printList(head)
```

OUTPUT : 10 20 30 40 50

10 20 30 45 40 50

7] Delete First Node Of singly Linked List :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def delFirst(head):
    if head==None:
        return None
    else:
        return head.next

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=delFirst(head)

printList(head)
```

OUTPUT :

10 20 30 40

20 30 40

8] Delete Last Node Of Linked List :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def delLastNode(head):
    if head==None:
        return None
    if head.next==None:
        return None
    curr=head
    while curr.next.next!=None:
        curr=curr.next

    curr.next=None
    return head

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()


head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=delLastNode(head)

printList(head)
```

OUTPUT :

10 20 30 40

10 20 30

9] Sorted Inserted Linked List In Python :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def sortedInsert(head,x):
    temp=Node(x)

    if head==None:
        return temp
    elif x<head.key:
        temp.next=head
        return temp
    else:
        curr=head

        while curr.next!=None and curr.next.key<x:
            curr=curr.next

        temp.next=curr.next
        curr.next=temp
        return head

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

h=head
while h!=None:
    print(h.key)
    h=h.next

print()

h=sortedInsert(head,35)

h=head
while h!=None:
    print(h.key)
    h=h.next
```

OUTPUT :

10

20

30

40

10

20

30

35

40

10] reverse a linked list using stack naïve solution :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None


def reverseList(head):
    stack=[]

    curr=head

    while curr is not None :
        stack.append(curr.key)
        curr=curr.next

    curr=head
    while curr is not None:
        curr.key=stack.pop()
        curr=curr.next

    return head

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=reverseList(head)

printList(head)
```

OUTPUT : 10 20 30 40

40 30 20 10

11] Reverse a Linked List Efficient :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def reverseList(head):
    curr=head
    prev=None

    while curr is not None:
        next=curr.next
        curr.next=prev
        prev=curr
        curr=next

    return prev

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()


head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=reverseList(head)

printList(head)
```

OUTPUT :

10 20 30 40

40 30 20 10

12] recursive reverse linked list part 1 :

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def reverseList(head):
    if head==None:
        return head
    if head.next==None:
        return head

    rest_head=reverseList(head.next)
    rest_tail=head.next
    rest_tail.next=head
    head.next=None

    return rest_head

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=reverseList(head)

printList(head)
```

OUTPUT :

10 20 30 40

40 30 20 10

13] recursive a reverse linked list part 2

(tail recursive which is slow in python but fast in other language):

```
class Node:
    def __init__(self,k):
        self.key=k
        self.next=None

def reverseList(curr,prev=None):
    if curr==None:
        return prev

    next=curr.next
    curr.next=prev

    return reverseList(next,curr)

def printList(head):
    curr=head
    while curr!=None:
        print(curr.key,end=" ")
        curr=curr.next
    print()

head=Node(10)
head.next=Node(20)
head.next.next=Node(30)
head.next.next.next=Node(40)

printList(head)

head=reverseList(head)

printList(head)
```

OUTPUT :

10 20 30 40

40 30 20 10