1. class Node:

def \_\_init\_\_(self):

self.right=None

self.left=None

self.data=None

tree=Node()

tree.data=1 #root Node

tree.left=Node() #creating new node for left

tree.left.data=2 #inserting data in left child

#creating new node for right child

tree.right=Node()

# inserting data in right child

tree.right.data=3

print(tree.data) #root Node

print(tree.left.data) #root child

# right child

print(tree.right.data) #right child

1. class Node:

def \_\_init\_\_(self):

self.right=None

self.left=None

self.data=None

def inorder\_traversal(self,Node):

if Node:

self.inorder\_traversal(Node.left)

print(Node.data)

self.inorder\_traversal(Node.right)

def preorder\_traversal(self,Node):

if Node:

print(Node.data,end=" ")

self.preorder\_traversal(Node.left)

self.preorder\_traversal(Node.right)

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data,end=" ")

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

print(tree.data)

print(tree.left.data)

print(tree.right.data)

tree.inorder\_traversal(tree)

tree.preorder\_traversal(tree)

tree.postorder\_traversal(tree)

1. class Node:

def \_\_init\_\_(self,data=None):

self.right=None

self.left=None

self.data=data

def inorder\_traversal(self,Node):

if Node:

self.inorder\_traversal(Node.left)

print(Node.data,end=" ")

self.inorder\_traversal(Node.right)

def preorder\_traversal(self,Node):

if Node:

print(Node.data,end=" ")

self.preorder\_traversal(Node.left)

self.preorder\_traversal(Node.right)

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data,end=" ")

def count(self,Node):

if(Node!=None)

count+=1

self.count\_of\_nodes(left,right)

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

print(tree.data)

print(tree.left.data)

print(tree.right.data)

tree.inorder\_traversal(Node=tree)

tree.preorder\_traversal(Node=tree)

tree.postorder\_traversal(Node=tree)

1. class Node:

def \_\_init\_\_(self,data):

self.data=data

self.next=None

class LinkedList:

def \_\_init\_\_(self):

self.head=None #starting point of LL

def insert(self,data):

new\_node=Node(data)

if self.head is None:

self.head=new\_node

else:

temp=self.head

while temp.next:

temp=temp.next

temp.next=new\_node

def display(self):

temp=self.head

while temp:

print(temp.data,end="->")

temp=temp.next

print("None")

ll=LinkedList()

ll.insert(10)

ll.insert(20)

ll.insert(30)

ll.insert(40)

ll.display()