DAY-6:

TREES:  
PROGRAM:  
class Node:

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

self.left=None

self.right=None

self.data=None

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)

OUTPUT:  
1

2

3

PROGRAM:  
class Node:

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

self.left=None

self.right=None

self.data=None

def inorder\_traversal(self,Node):

if Node:

self.inorder\_traversal(Node.left)

print(Node.data)

self.inorder\_traversal(Node.right)

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.inorder\_traversal(Node=tree)

OUTPUT:  
4

2

1

3

PREORDER:  
PROGRAM:

class Node:

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

self.left=None

self.right=None

self.data=None

def preorder\_traversal(self,Node):

if Node:

print(Node.data)

self.preorder\_traversal(Node.left)

self.preorder\_traversal(Node.right)

tree=Node()

tree.data=1 #root node

tree.left=Node()#creating node in left

tree.left.data=2

tree.right=Node()#creating node in right

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.preorder\_traversal(Node=tree)

OUTPUT:  
1

2

4

3

POSTORDER:

PROGRAM:

class Node:

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

self.left=None

self.right=None

self.data=None

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data)

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.postorder\_traversal(Node=tree)

OUTPUT:  
4

2

3

1

INORDER, PREORDER, POSTORDER :

PROGRAM:  
class Node():

def \_init\_(self):

self.left=None

self.right=None

self.data=None

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=" ")

tree=Node()

tree.data=1

tree.left=Node()

tree.left.data=2

tree.right=Node()

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.left.right=Node()

tree.left.right.data=5

tree.right.left=Node()

tree.right.left.data=6

tree.right.right=Node()

tree.right.right.data=7

tree.inorder\_traversal(Node=tree)

print()

tree.preorder\_traversal(Node=tree)

print()

tree.postorder\_traversal(Node=tree)

print()

OUTPUT:

Inorder Traversal:

4 2 5 1 6 3 7

Preorder Traversal:

1 2 4 5 3 6 7

Postorder Traversal:

4 5 2 6 7 3 1

SUM OF DATA IN TREE:

PROGRAM:

class Node:

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

self.left=None

self.right=None

self.data=None

def postorder\_traversal(self,Node):

if Node:

self.postorder\_traversal(Node.left)

self.postorder\_traversal(Node.right)

print(Node.data,end=" ")

def sum\_of\_nodes(self,Node):

if Node is None:

return 0

return Node.data+self.sum\_of\_nodes(Node.left)+self.sum\_of\_nodes(Node.right)

tree=Node()

tree.data=1 #root node

tree.left=Node()#creating node in left

tree.left.data=2

tree.right=Node()#creating node in right

tree.right.data=3

tree.left.left=Node()

tree.left.left.data=4

tree.postorder\_traversal(Node=tree)

OUTPUT:

4 2 3 1 10