Program:

class Node:

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

self.data = data

self.left = None

self.right = None

class BinaryTree:

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

self.root = None

def insert\_root(self, data):

self.root = Node(data)

def insert\_left(self, parent, data):

parent.left = Node(data)

return parent.left

def insert\_right(self, parent, data):

parent.right = Node(data)

return parent.right

# Inorder Traversal: Left -> Root -> Right

def inorder(self, node):

if node:

self.inorder(node.left)

print(node.data, end=' ')

self.inorder(node.right)

# Preorder Traversal: Root -> Left -> Right

def preorder(self, node):

if node:

print(node.data, end=' ')

self.preorder(node.left)

self.preorder(node.right)

# Postorder Traversal: Left -> Right -> Root

def postorder(self, node):

if node:

self.postorder(node.left)

self.postorder(node.right)

print(node.data, end=' ')

# Display the tree visually

def display(self, node, level=0, prefix="Root: "):

if node is not None:

print(" " \* level + prefix + str(node.data))

self.display(node.left, level + 1, "L---- ")

self.display(node.right, level + 1, "R---- ")

# Main Execution

if \_\_name\_\_ == "\_\_main\_\_":

tree = BinaryTree()

tree.insert\_root(input("enter root data:"))

B = tree.insert\_left(tree.root, input("enter left data:"))

C = tree.insert\_right(tree.root, input("enter right data:"))

tree.insert\_left(B, input("enter left data:"))

tree.insert\_right(B, input("enter right data:"))

tree.insert\_right(C, input("enter right data:"))

print("Inorder Traversal: ", end=''); tree.inorder(tree.root); print()

print("Preorder Traversal: ", end=''); tree.preorder(tree.root); print()

print("Postorder Traversal: ", end=''); tree.postorder(tree.root); print()

print("\nTree Structure:")

tree.display(tree.root)

output:

enter root data:A

enter left data:B

enter right data:C

enter left data:D

enter right data:E

enter right data:F

Inorder Traversal: D B E A C F

Preorder Traversal: A B D E C F

Postorder Traversal: D E B F C A

Tree Structure:

Root: A

L---- B

L---- D

R---- E

R---- C

R---- F