**Labsheet 3**

**Binary Tree**

# Binary Tree in Python

class Node:

    def \_\_init\_\_(self, key):

        self.left = None

        self.right = None

        self.val = key

    # Traverse preorder

    def traversePreOrder(self):

        print(self.val, end=' ')

        if self.left:

            self.left.traversePreOrder()

        if self.right:

            self.right.traversePreOrder()

    # Traverse inorder

    def traverseInOrder(self):

        if self.left:

            self.left.traverseInOrder()

        print(self.val, end=' ')

        if self.right:

            self.right.traverseInOrder()

    # Traverse postorder

    def traversePostOrder(self):

        if self.left:

            self.left.traversePostOrder()

        if self.right:

            self.right.traversePostOrder()

        print(self.val, end=' ')

root = Node(1)

root.left = Node(2)

root.right = Node(3)

root.left.left = Node(4)

print("Pre order Traversal: ", end="")

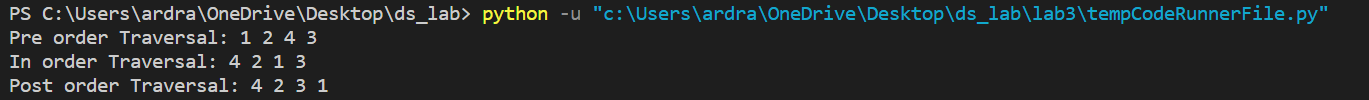
root.traversePreOrder()

print("\nIn order Traversal: ", end="")

root.traverseInOrder()

print("\nPost order Traversal: ", end="")

root.traversePostOrder()

****

**Binary Search Tree**

class Node:

    def \_\_init\_\_(self, key):

        self.key = key

        self.left = None

        self.right = None

def inorder(root):

    if root is not None:

        inorder(root.left)

        print(str(root.key) + "->", end=' ')

        inorder(root.right)

def insert(node, key):

    if node is None

        return Node(key)

    if key < node.key:

        node.left = insert(node.left, key)

    else:

        node.right = insert(node.right, key)

    return node

def minValueNode(node):

    current = node

    while(current.left is not None):

        current = current.left

    return current

def deleteNode(root, key):

    if root is None:

        return root

    if key < root.key:

        root.left = deleteNode(root.left, key)

    elif(key > root.key):

        root.right = deleteNode(root.right, key)

    else:

        if root.left is None:

            temp = root.right

            root = None

            return temp

        elif root.right is None:

            temp = root.left

            root = None

            return temp

        temp = minValueNode(root.right)

        root.key = temp.key

        root.right = deleteNode(root.right, temp.key)

    return root

root = None

root = insert(root, 8)

root = insert(root, 3)

root = insert(root, 1)

root = insert(root, 6)

root = insert(root, 7)

root = insert(root, 10)

root = insert(root, 14)

root = insert(root, 4)

print("Inorder traversal: ", end=' ')

inorder(root)

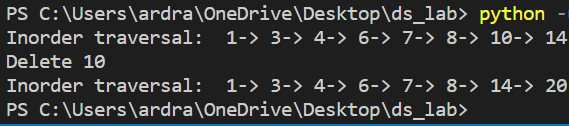
print("\nDelete 10")

root = deleteNode(root, 10)

root = insert(root, 20)

print("Inorder traversal: ", end=' ')

inorder(root)

****