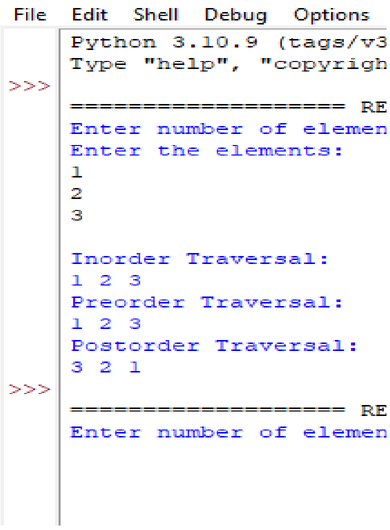
```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None
# Binary Search Tree class
class BinarySearchTree:
    def __init__(self):
        self.root = None
  # Function to insert a new node
    def insert(self, root, data):
        if root is None:
            return Node(data)
        else:
            if data < root.data:
                root.left = self.insert(root.left, data)
            else:
                root.right = self.insert(root.right, data)
        return root
   # Inorder Traversal (Left -> Root -> Right)
    def inorder(self, root):
        if root:
            self.inorder(root.left)
            print(root.data, end=" ")
            self.inorder(root.right)
 # Preorder Traversal (Root -> Left -> Right)
    def preorder(self, root):
        if root:
            print(root.data, end=" ")
            self.preorder(root.left)
            self.preorder(root.right)
    # Postorder Traversal (Left -> Right -> Root)
    def postorder(self, root):
        if root:
            self.postorder(root.left)
            self.postorder(root.right)
            print(root.data, end=" ")


# Main Program
if __name__ == "__main__":
    bst = BinarySearchTree()
    root = None
  # Taking input from user
    n = int(input("Enter number of elements to insert: "))
    print("Enter the elements:")
    for i in range(n):
        el = int(input())
        root = bst.insert(root, el)

    print("\nInorder Traversal: ")
    bst.inorder(root)

    print("\nPreorder Traversal: ")
    bst.preorder(root)

    print("\nPostorder Traversal: ")
    bst.postorder(root)
```



```
>>> Python 3.10.9 (tags/v3
    Type "help", "copyrigh

    =================== RE
    Enter number of elemen
    Enter the elements:
    1
    2
    3

    Inorder Traversal:
    1 2 3
    Preorder Traversal:
    1 2 3
    Postorder Traversal:
    3 2 1
>>>
    =================== RE
    Enter number of elemen
```