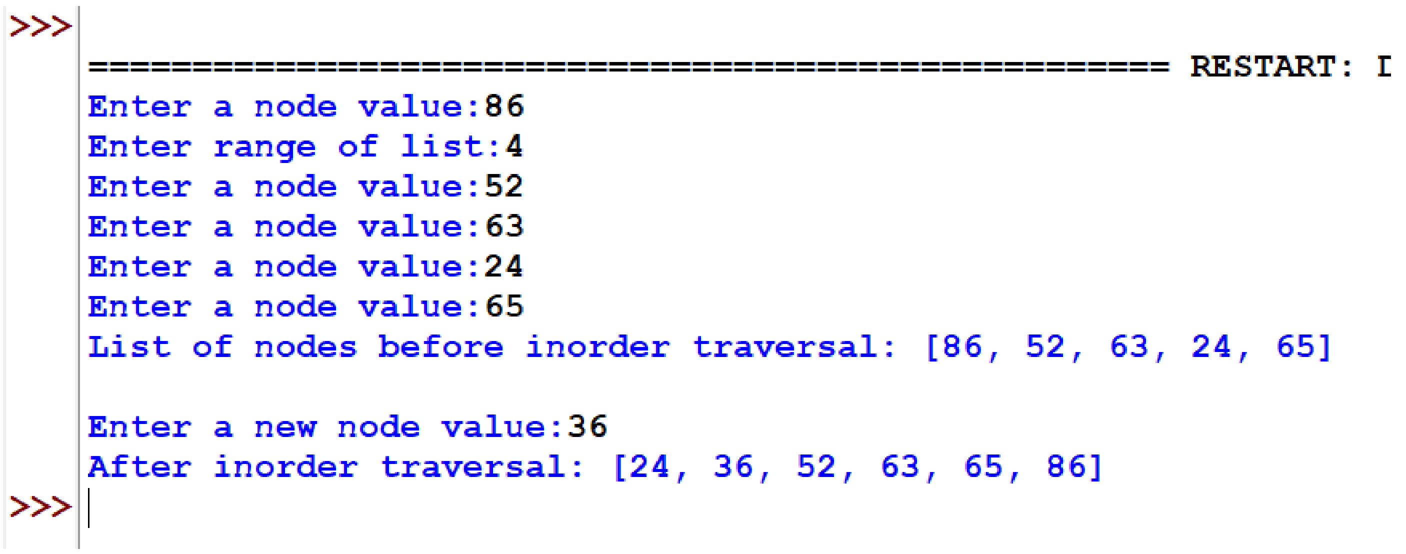
04)Write Python program for inserting an element into binary tree.

```
#04)Write Python program for inserting an element into binary tree.
#Python program to demonstrate insert operation in binary tree.
class BST_Node:
        def __init__(self,val):
                self.val=val
                self.left=None
                self.right=None
#A function to insert a new node
        def insert_Node(self,New_Node):
                if self.val is None:
                        self.val=New_Node
                        return
                if self.val==New_Node:
                    return


                if self.val>New_Node:
                        if self.left is None:
                                self.left=BST_Node(New_Node)

                        else:
                                self.left.insert_Node(New_Node)

                else:
                        if self.right is None:

                                self.right=BST_Node(New_Node)
                        else:
                                self.right.insert_Node(New_Node)

#A function for inorder tree traversal
def inorder_Tree(root):
        if root:
                inorder_Tree(root.left)
                #print(root.val)
                li1.append(root.val)
                inorder_Tree(root.right)
li=[]
#root means self because selft represent object itself.
root=BST_Node(int(input("Enter a node value:")))
R=int(input("Enter range of list:"))
li.append(root.val)
for i in range(R):
        li.append(int(input("Enter a node value:")))

print("List of nodes before inorder traversal:",li)
li1=[]
for i in li:
        root.insert_Node(i)

#inorder_Tree(root)
#print(li1)
```

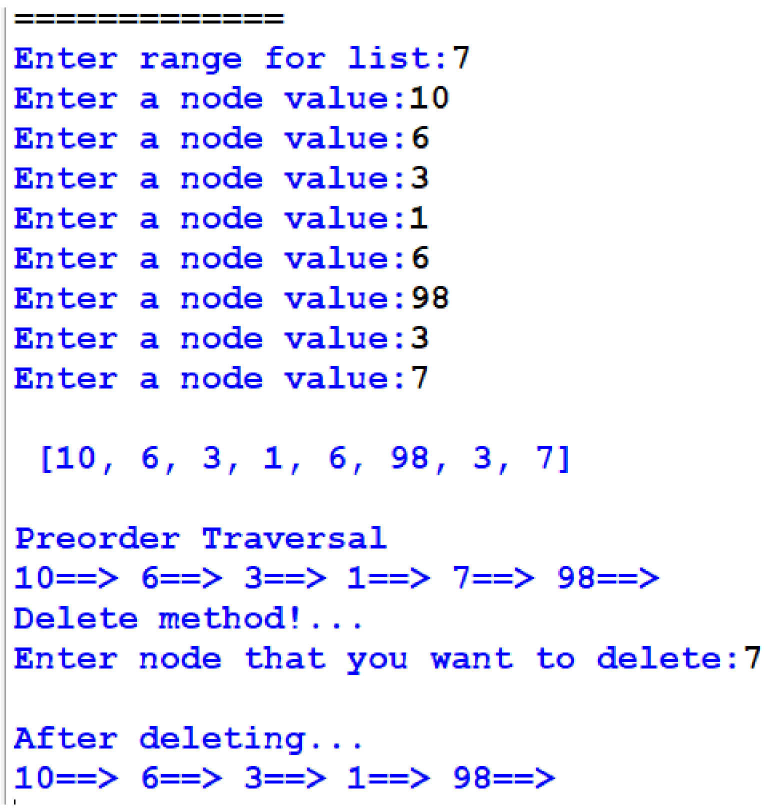
```python
root.insert_Node(int(input("\nEnter a new node value:")))
inorder_Tree(root)
print("After inorder traversal:",li1)
```



```
>>>
    ===================================================== RESTART: I
    Enter a node value:86
    Enter range of list:4
    Enter a node value:52
    Enter a node value:63
    Enter a node value:24
    Enter a node value:65
    List of nodes before inorder traversal: [86, 52, 63, 24, 65]

    Enter a new node value:36
    After inorder traversal: [24, 36, 52, 63, 65, 86]
>>>
```

05)Write Python program for deleting an element (assuming data is given) from binary tree.

```python
#05)Write Python program for deleting an element (assuming data is given) from binary
tree.

class BST_Tree:
    def __init__(self,value):
        self.lchild=None
        self.rchild=None
        self.value=value
    #Inserting new node in BST
    def insert_node(self,data):
        if self.value is None:
            self.value=data
        elif self.value ==data:
            return
        elif self.value>data:
            if self.lchild is None:
                self.lchild=BST_Tree(data)
            else:
                self.lchild.insert_node(data)
        else:
            if self.rchild:
                self.rchild.insert_node(data)
            else:
                self.rchild=BST_Tree(data)

    def delete_node(self,data):
        if self.value is None:     #Check Tree is empty or not
            print("Tree is empty")
        elif data<self.value:     #Find the position of the given node
            if self.lchild:
                self.lchild=self.lchild.delete_node(data)
            else:
                print("Given node is not presernt in tree..")
        elif data>self.value:
            if self.rchild:
                self.rchild=self.rchild.delete_node(data)
            else:
                print("Given node is not present in tree..")
        else:  #Node value store in self.lchild ,Check it contain 0,1 or 2 child
            if self.lchild is None: #---
                temp=self.rchild
                self=None
                return temp
            if self.rchild is None:
                temp= self.lchild
                self=None
                return temp #--Delete operation for node have 0 and 1 child
            #Delete operation for node have 2 child
            node=self.rchild
            while node.lchild:
                node=node.lchild
            self.value=node.value
            self.rchild=self.rchild.delete_node(node.value)
        return self


    #Traversal method
    def Preorder(self):
        print(self.value,end="==> ")
        if self.lchild :
            self.lchild.Preorder()
        if self.rchild:
            self.rchild.Preorder()


R=int(input("Enter range for list:"))
root=BST_Tree(int(input("Enter a node value:")))
li=[]
li.append(root.value)
for i in range(R):
    li.append(int(input("Enter a node value:")))
print("\n",li)
for i in li:
    root.insert_node(i)

print("\nPreorder Traversal")
root.Preorder()

print("\nDelete method!...")
root.delete_node(int(input("Enter node that you want to delete:")))
print("\nAfter deleting...")
root.Preorder()
```



```
==============
Enter range for list:7
Enter a node value:10
Enter a node value:6
Enter a node value:3
Enter a node value:1
Enter a node value:6
Enter a node value:98
Enter a node value:3
Enter a node value:7

 [10, 6, 3, 1, 6, 98, 3, 7]

Preorder Traversal
10==> 6==> 3==> 1==> 7==> 98==>
Delete method!...
Enter node that you want to delete:7

After deleting...
10==> 6==> 3==> 1==> 98==>
```