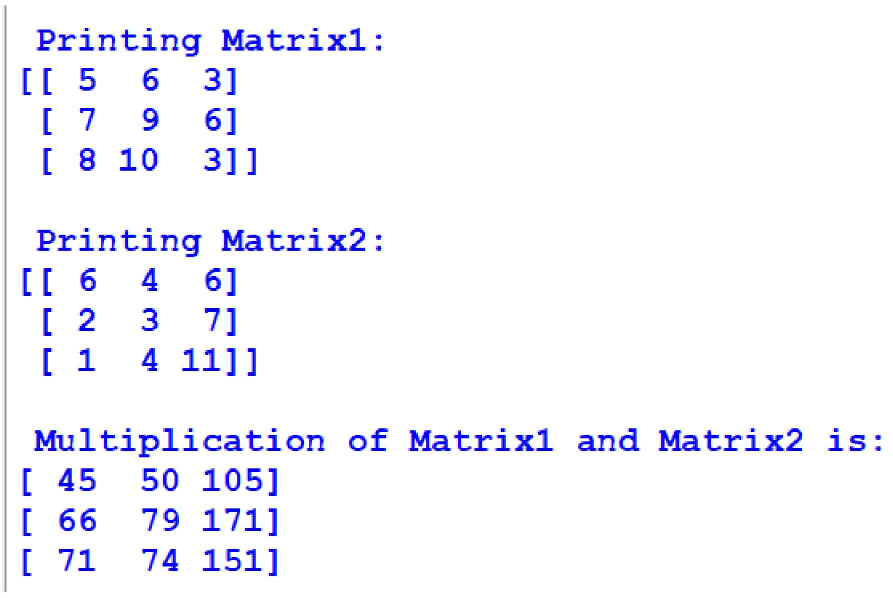
1. Write Python program to perform matrix multiplication. Discuss the complexity of algorithm used

```python
#1. Write Python program to perform matrix multiplication. Discuss the complexity of algorithm
#used
from numpy import *

Matrix1=array([[5,6,3],
                [7,9,6],
                [8,10,3]
                ])
Matrix2=array([[6,4,6],
                [2,3,7],
                [1,4,11]
                ])
Matrix3=array([[0,0,0],
               [0,0,0],
                [0,0,0]
                ])
print("\n Printing Matrix1:")
print(Matrix1)
print("\n Printing Matrix2:")
print(Matrix2)
for i in range(len(Matrix1)):
    for j in range(len(Matrix2)):
        Matrix3[i][j]=0
        Sum=0
        for k in range(len(Matrix2)):
            Sum=Sum+(Matrix1[i][k]*Matrix2[k][j])
        Matrix3[i][j]=Sum
print("\n Multiplication of Matrix1 and Matrix2 is:")
for R in Matrix3:
    print(R)
```

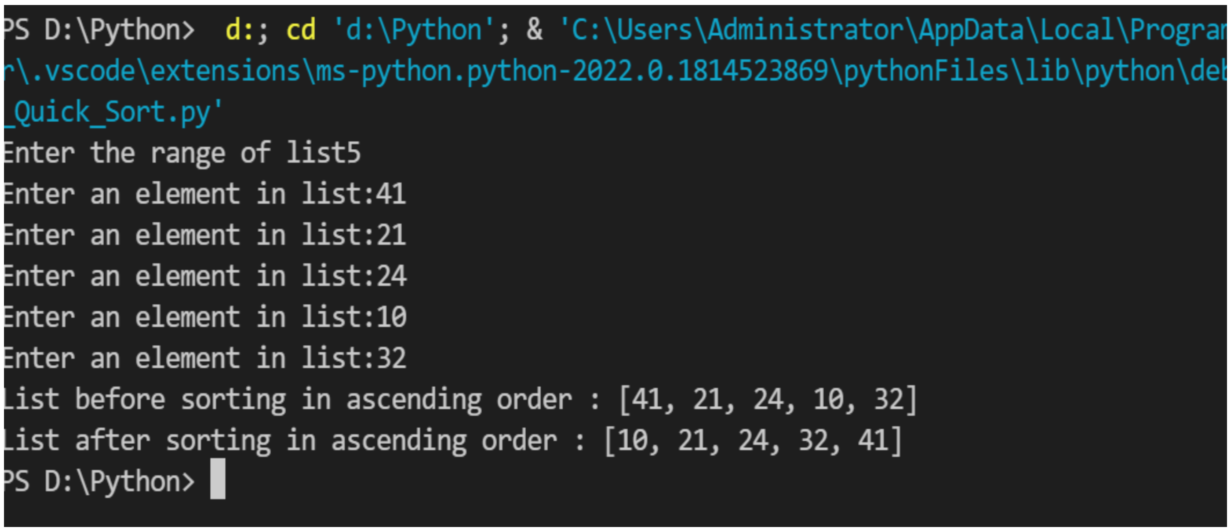


```
 Printing Matrix1:
[[ 5  6  3]
 [ 7  9  6]
 [ 8 10  3]]

 Printing Matrix2:
[[ 6  4  6]
 [ 2  3  7]
 [ 1  4 11]]

 Multiplication of Matrix1 and Matrix2 is:
[ 45  50 105]
[ 66  79 171]
[ 71  74 151]
```

2. Write Python program to sort n names using Quick sort algorithm. Discuss the complexity of algorithm used

```python
#2. Write Python program to sort n names using Quick sort algorithm. Discuss the
complexity of
```

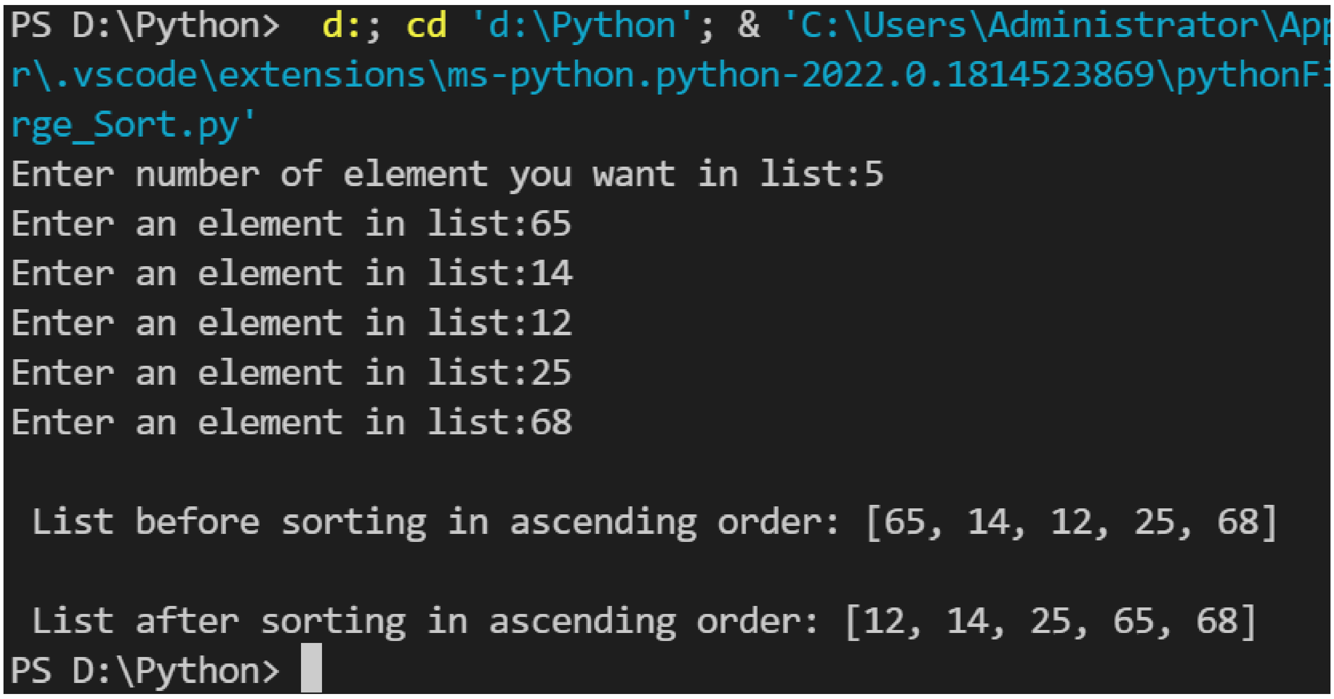
```python
#algorithm used

#Function toget correct position of pivot elemet
def Pivot_Position(List,First,Last):
    Pivot=List[First]
    left=First+1
    right=Last
    while True:
        while left<=right and List[left]<=Pivot:
            left=left+1
        while left<=right and List[right]>=Pivot:
            right=right-1
        if right<left:
            break
        else:
            List[left],List[right]=List[right],List[left]
    List[First],List[right]=List[right],List[First]
    return right

def Quick_Sort(List,First,Last):#Dividing the list according to pivot element
    if First<Last:#Bas case = First==Last ,Partition of list,When list contain single
element the stop everything
        pivot=Pivot_Position(List,First,Last)
        Quick_Sort(List,First,pivot-1)
        Quick_Sort(List,pivot+1,Last)

    #Function not returning any value because we didnot create new list just sorting
original list
#Calling the functions
Range=int(input("Enter the range of list"))
List=[]
for i in range(Range):
    List.append(int(input("Enter an element in list:")))
print("List before sorting in ascending order :",List)
length=len(List)
Quick_Sort(List,0,length-1)
print("List after sorting in ascending order :",List)
```



```
PS D:\Python>  d:; cd 'd:\Python'; & 'C:\Users\Administrator\AppData\Local\Progra
r\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\de
_Quick_Sort.py'
Enter the range of list5
Enter an element in list:41
Enter an element in list:21
Enter an element in list:24
Enter an element in list:10
Enter an element in list:32
List before sorting in ascending order : [41, 21, 24, 10, 32]
List after sorting in ascending order : [10, 21, 24, 32, 41]
PS D:\Python> █
```

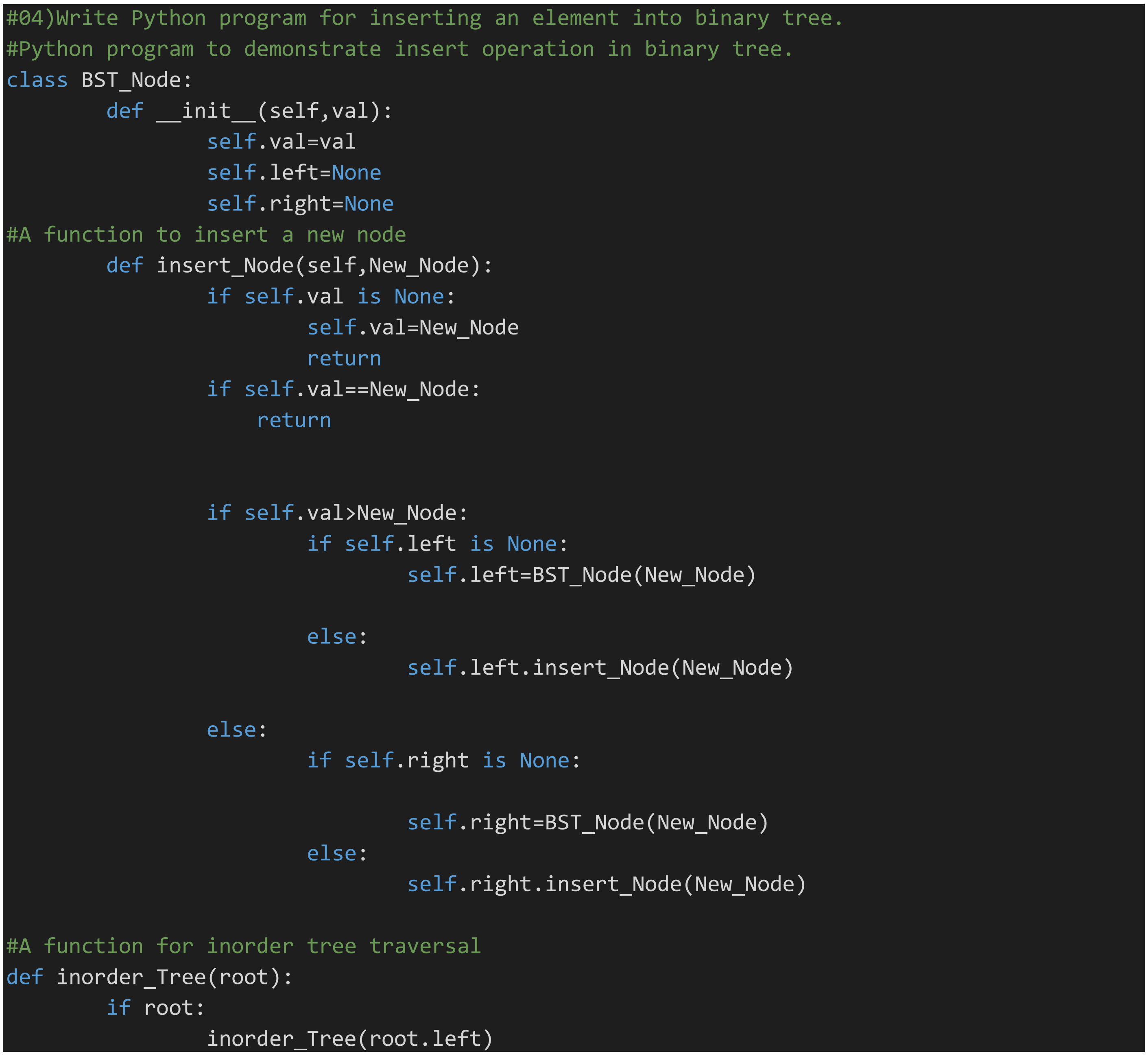
3. Write Python program to sort n numbers using Merge sort algorithm. Discuss the complexity of algorithm used.

```python
#3. Write Python program to sort n numbers using Merge sort algorithm. Discuss the
complexity
#of algorithm used.


#Divide the list into sublit untile it contain ingle element
def Merge_Sort(List):
    if len(List)>1:#Base condition if list contain single element the stop divide
        mid=len(List)//2
        Left_li=List[:mid]
        Right_li=List[mid:]
        Merge_Sort(Left_li)
        Merge_Sort(Right_li)

        i=0 #index of left list
        j=0 # index of right list
        k=0 #index of original list
        #For comparing and Merge value
        while i<len(Left_li)and j<len(Right_li):
            if Left_li[i]<Right_li[j]:
                List[k]=Left_li[i]
                i=i+1  # for checking next value
                k=k+1  #for adding value to the next position
            else:
                List[k]=Right_li[j]
                j=j+1
                k=k+1
         #For Merge value which is left
        while i<len(Left_li):
            List[k]=Left_li[i]
            i=i+1
            k=k+1
        while j<len(Right_li):
            List[k]=Right_li[j]
            j=j+1
            k=k+1

#Calling function
Range=int(input("Enter number of element you want in list:"))
List=[]
for i in range(Range):
    List.append(int(input("Enter an element in list:")))
print("\n List before sorting in ascending order:",List)
Merge_Sort(List)
print("\n List after sorting in ascending order:",List)
```

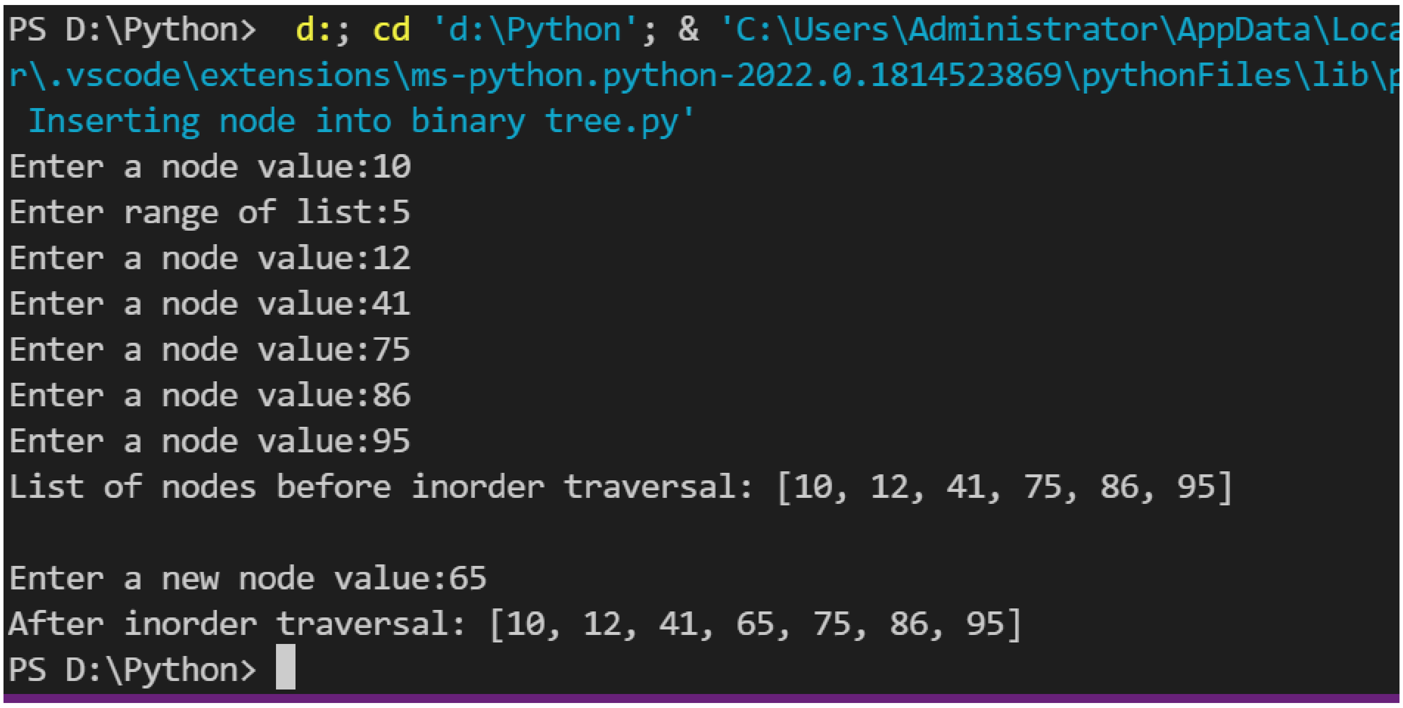


```
PS D:\Python>  d:; cd 'd:\Python'; & 'C:\Users\Administrator\App
r\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFi
rge_Sort.py'
Enter number of element you want in list:5
Enter an element in list:65
Enter an element in list:14
Enter an element in list:12
Enter an element in list:25
Enter an element in list:68

 List before sorting in ascending order: [65, 14, 12, 25, 68]

 List after sorting in ascending order: [12, 14, 25, 65, 68]
PS D:\Python> 
```

4. Write Python program for inserting an element into binary tree.



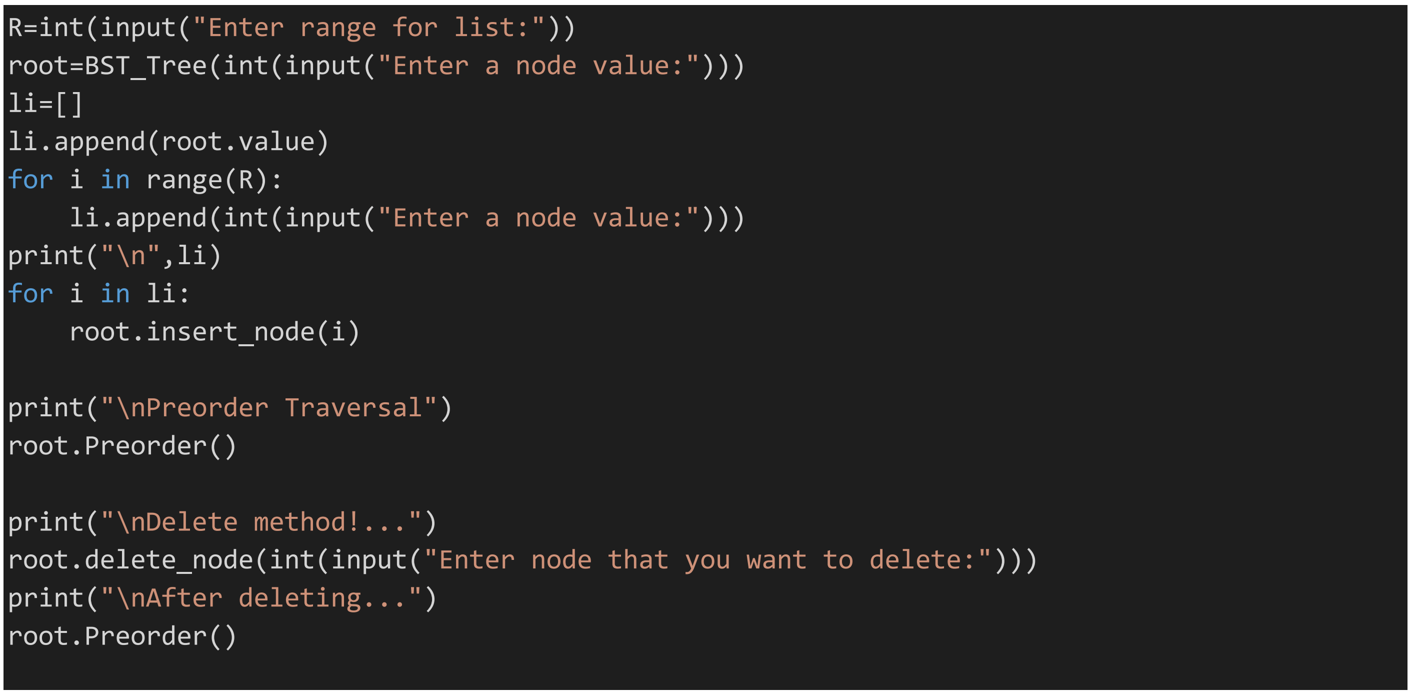
```python
#04)Write Python program for inserting an element into binary tree.
#Python program to demonstrate insert operation in binary tree.
class BST_Node:
        def __init__(self,val):
                self.val=val
                self.left=None
                self.right=None
#A function to insert 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)
root.insert_Node(int(input("\nEnter a new node value:")))
inorder_Tree(root)
print("After inorder traversal:",li1)
```



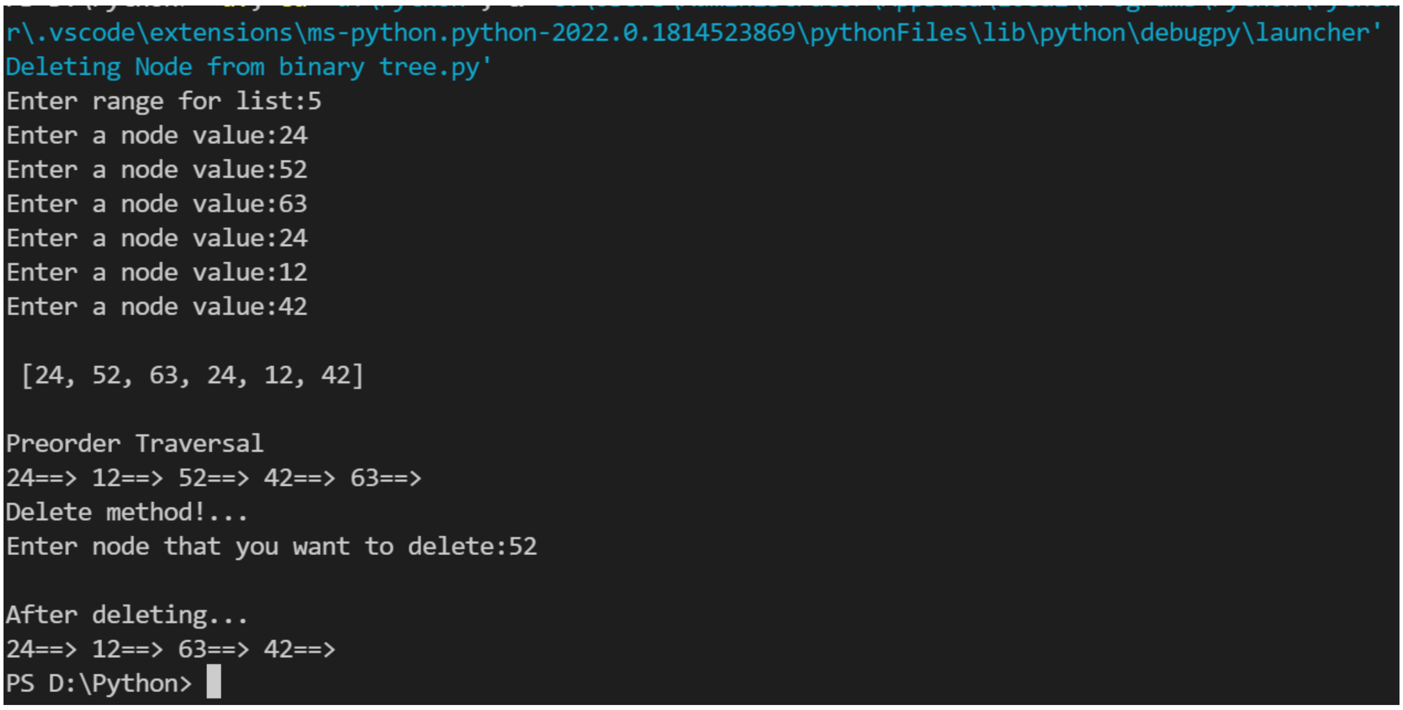
```
PS D:\Python>  d:; cd 'd:\Python'; & 'C:\Users\Administrator\AppData\Loca
r\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\p
 Inserting node into binary tree.py'
Enter a node value:10
Enter range of list:5
Enter a node value:12
Enter a node value:41
Enter a node value:75
Enter a node value:86
Enter a node value:95
List of nodes before inorder traversal: [10, 12, 41, 75, 86, 95]

Enter a new node value:65
After inorder traversal: [10, 12, 41, 65, 75, 86, 95]
PS D:\Python> 
```

5. 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()
```



```python
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()
```



```
[illegible]
r\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher'
Deleting Node from binary tree.py'
Enter range for list:5
Enter a node value:24
Enter a node value:52
Enter a node value:63
Enter a node value:24
Enter a node value:12
Enter a node value:42

 [24, 52, 63, 24, 12, 42]

Preorder Traversal
24==> 12==> 52==> 42==> 63==>
Delete method!...
Enter node that you want to delete:52

After deleting...
24==> 12==> 63==> 42==>
PS D:\Python>
```