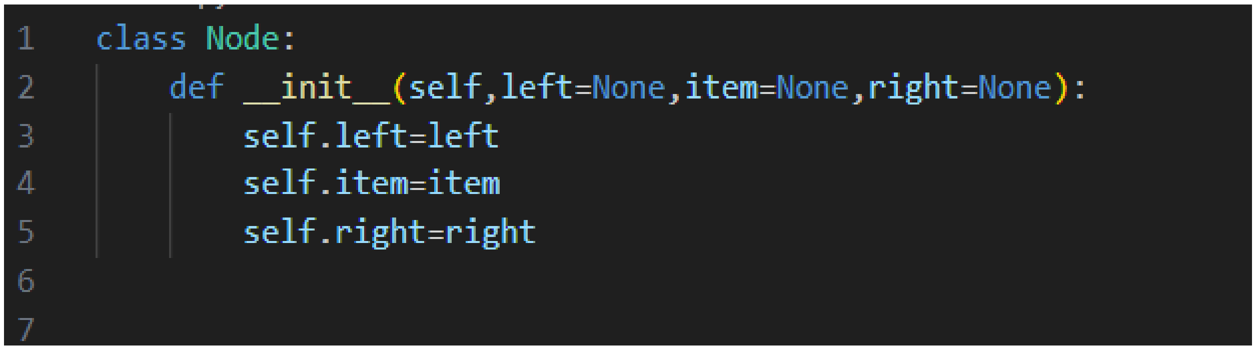
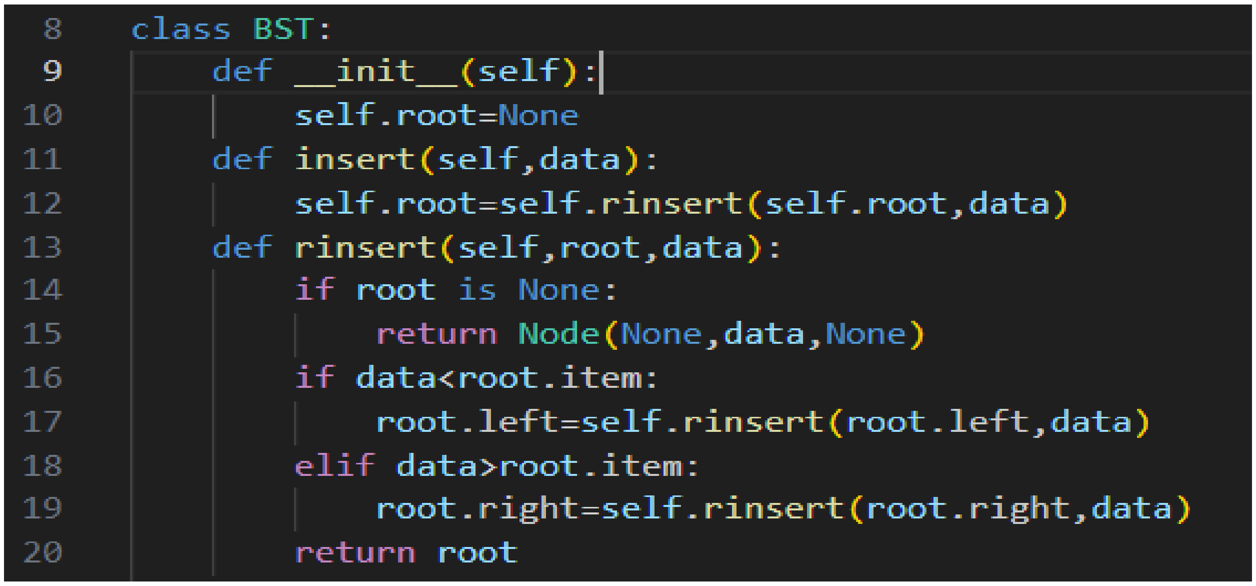
LAB#13

Example#1: Write a program to insert data into the binary search tree and then traverse it by using an inorder traversal.

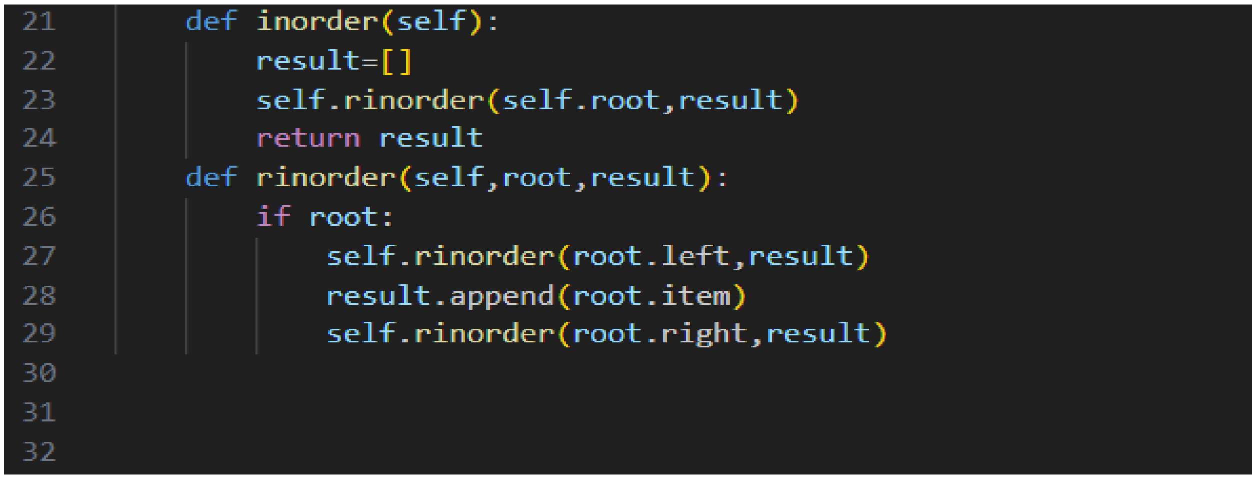
Solution:



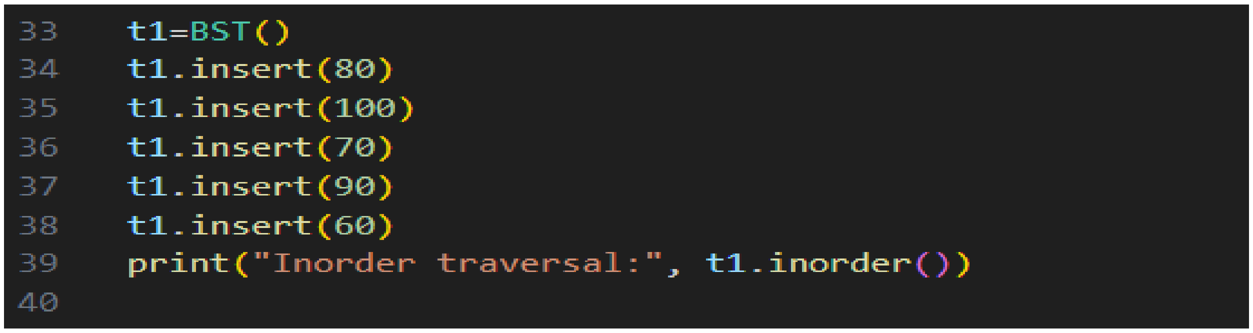
```python
class Node:
    def __init__(self,left=None,item=None,right=None):
        self.left=left
        self.item=item
        self.right=right

```



```python
class BST:
    def __init__(self):
        self.root=None
    def insert(self,data):
        self.root=self.rinsert(self.root,data)
    def rinsert(self,root,data):
        if root is None:
            return Node(None,data,None)
        if data<root.item:
            root.left=self.rinsert(root.left,data)
        elif data>root.item:
            root.right=self.rinsert(root.right,data)
        return root
```

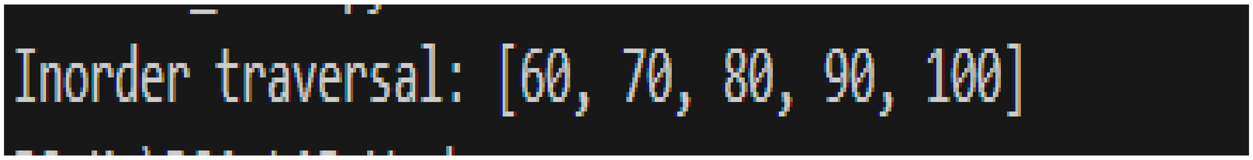


```python
    def inorder(self):
        result=[]
        self.rinorder(self.root,result)
        return result
    def rinorder(self,root,result):
        if root:
            self.rinorder(root.left,result)
            result.append(root.item)
            self.rinorder(root.right,result)
```



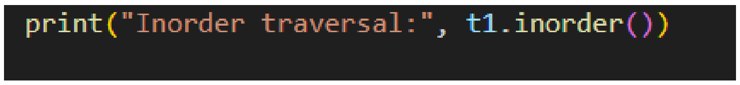
```python
t1=BST()
t1.insert(80)
t1.insert(100)
t1.insert(70)
t1.insert(90)
t1.insert(60)
print("Inorder traversal:", t1.inorder())
```

Result:

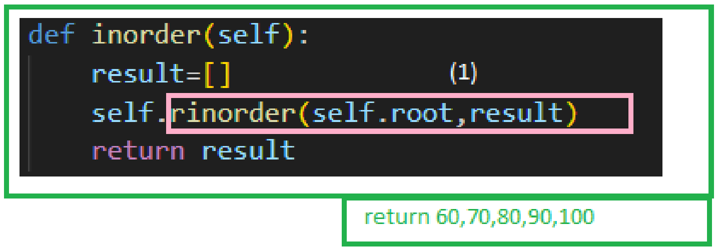


```
Inorder traversal: [60, 70, 80, 90, 100]
```

Explanation:



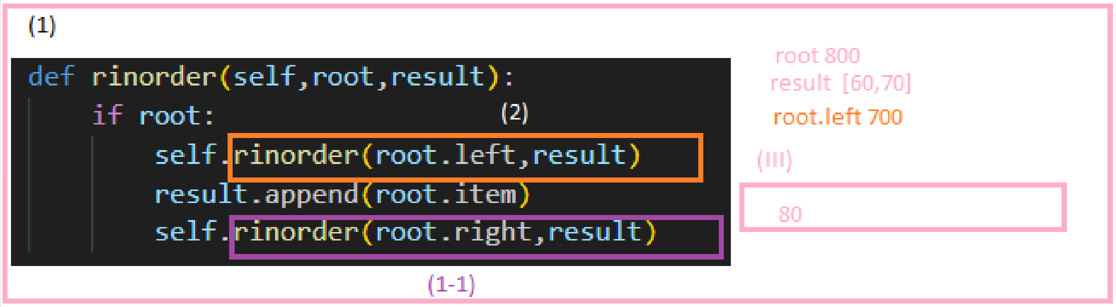
```python
print("Inorder traversal:", t1.inorder())
```



```python
def inorder(self):
    result=[]
    self.rinorder(self.root,result)
    return result
```

(1)

return 60,70,80,90,100



(1)

```python
def rinorder(self,root,result):
    if root:
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```

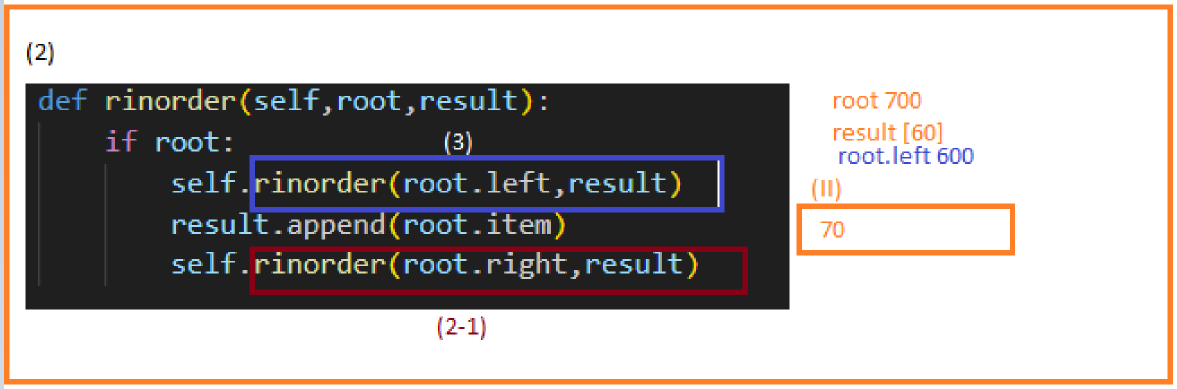
(2)

(1-1)

root 800
result [60,70]
root.left 700

(III)

80



(2)

```python
def rinorder(self,root,result):
    if root:
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```

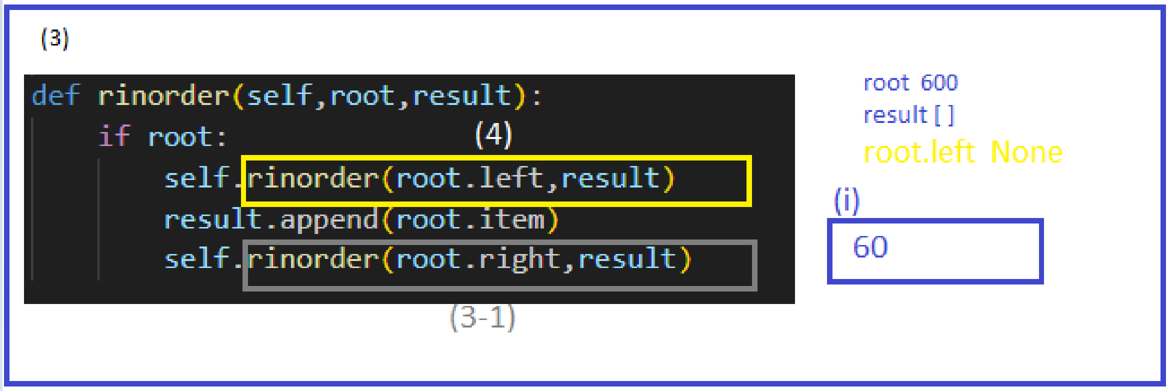
(3)

(2-1)

root 700
result [60]
root.left 600

(II)

70



(3)

```python
def rinorder(self,root,result):
    if root:
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```

(4)

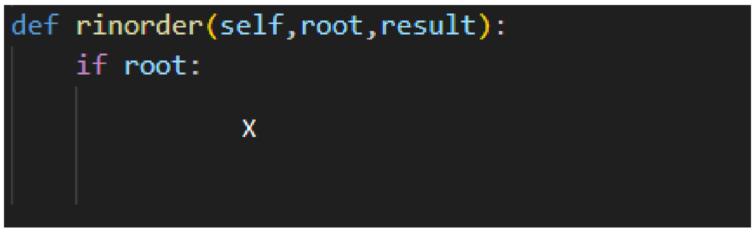
(3-1)

root 600
result [ ]
root.left None

(i)

60

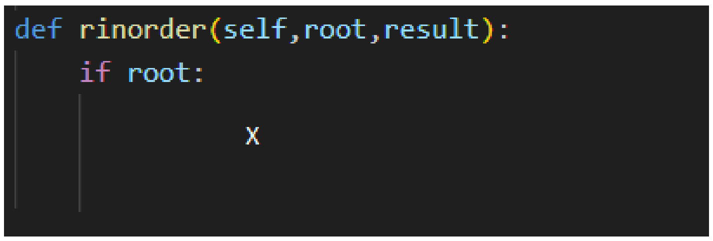
(4)



```
def rinorder(self,root,result):
    if root:
            X
```

root None
result [ ]

(3-1)

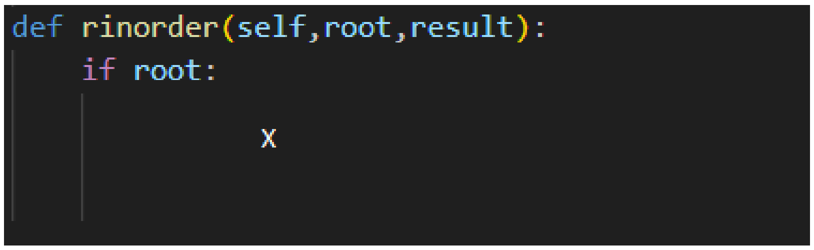


```
def rinorder(self,root,result):
    if root:
            X
```

root None
result [ ]

Section-3 has been executed completely.

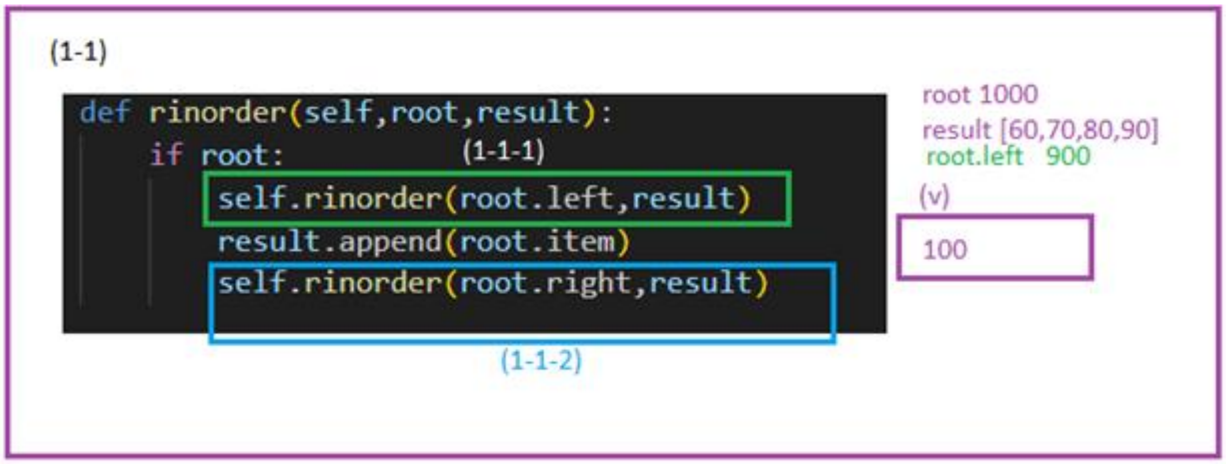
(2-1)



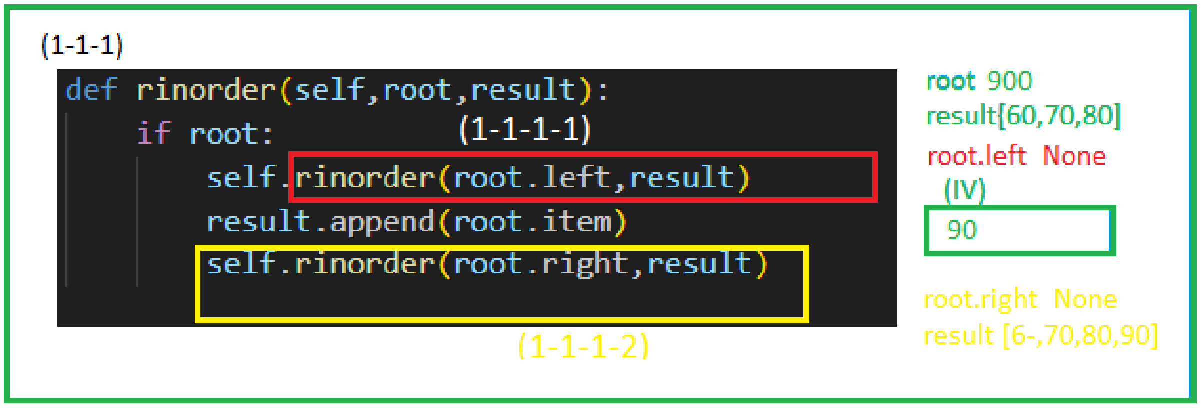
```
def rinorder(self,root,result):
    if root:
            X
```

root None
result [60,70]

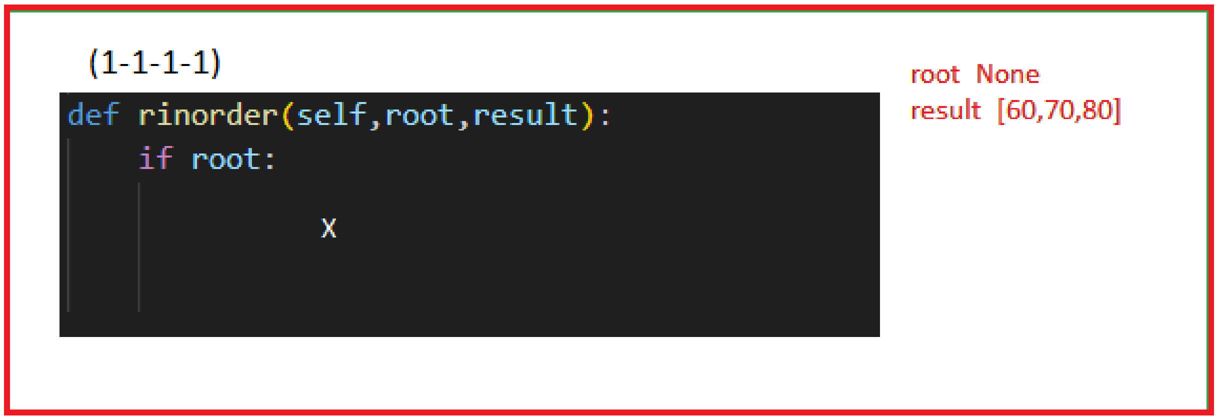
Section-2 has been executed completely.



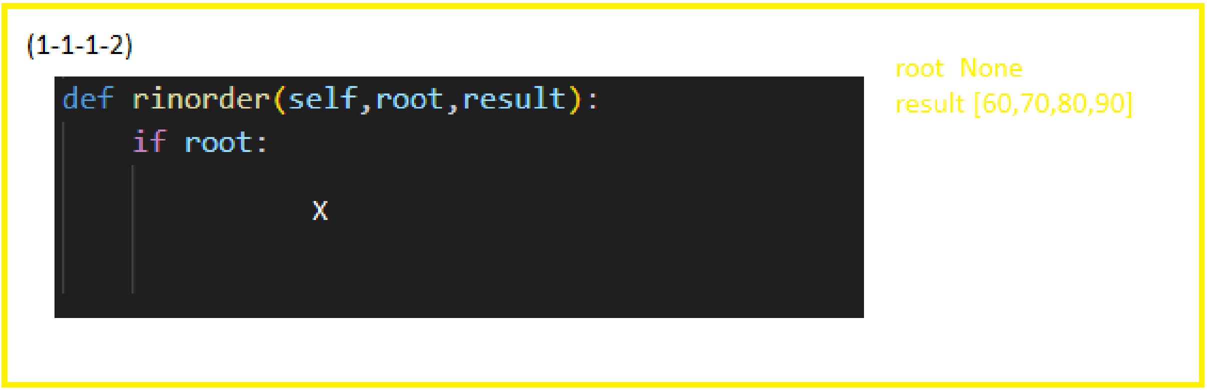
```
def rinorder(self,root,result):
    if root:
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```



```
def rinorder(self,root,result):
    if root:
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```



```
def rinorder(self,root,result):
    if root:

              X
```

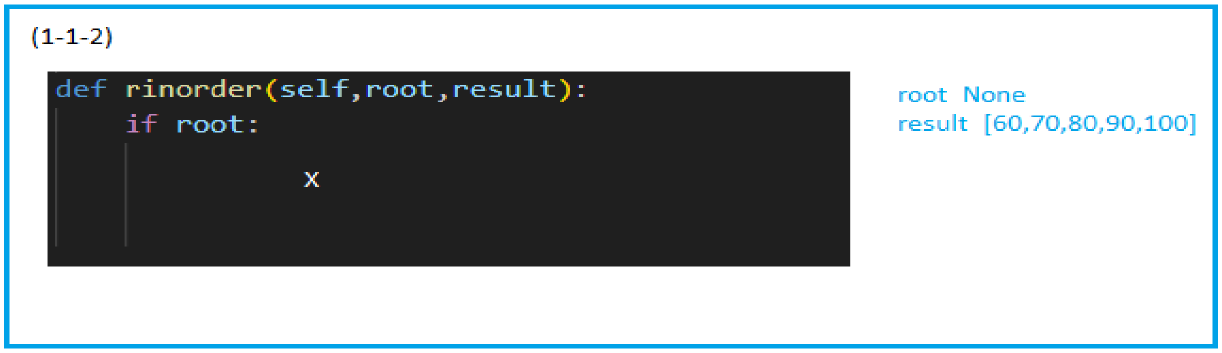


(1-1-1-2)

```
def rinorder(self,root,result):
    if root:
                X
```

root None
result [60,70,80,90]

Section-(1-1-1) has been executed completely.

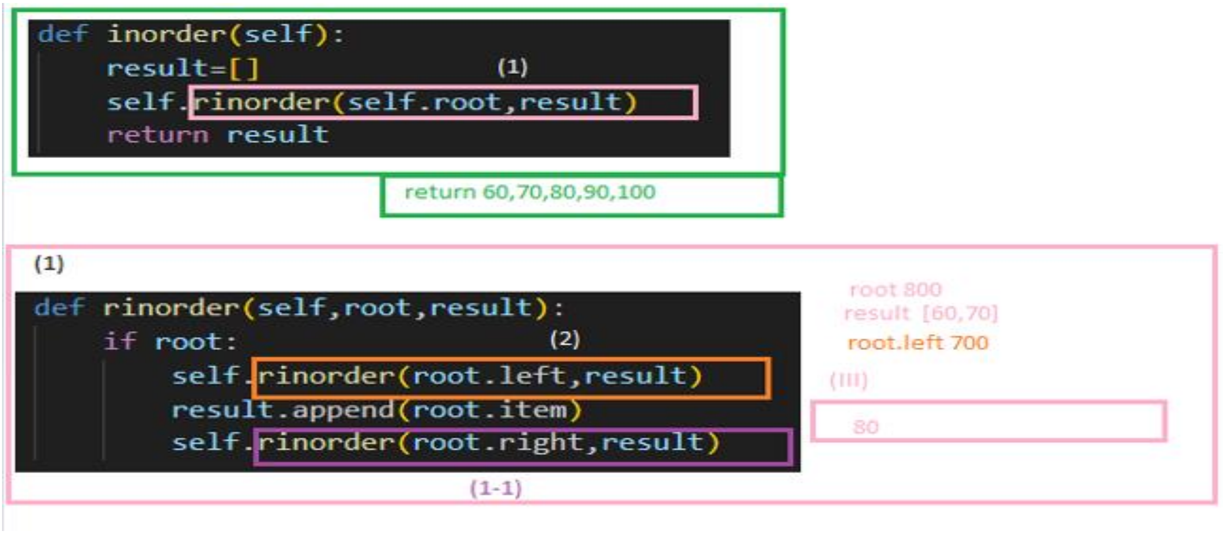


(1-1-2)

```
def rinorder(self,root,result):
    if root:
                X
```

root None
result [60,70,80,90,100]

Section-(1-1) has been executed completely which completes section-(1).



```
def inorder(self):
    result=[]                    (1)
    self.rinorder(self.root,result)
    return result
```

return 60,70,80,90,100

(1)

```
def rinorder(self,root,result):
    if root:                        (2)
        self.rinorder(root.left,result)
        result.append(root.item)
        self.rinorder(root.right,result)
```

(1-1)

root 800
result [60,70]
root.left 700
(III)
80

Class Assignment

Q: Add two more functions preorder and postorder traversal in the Example#1 Lab#13. Then explain the code of each function diagrammatically.