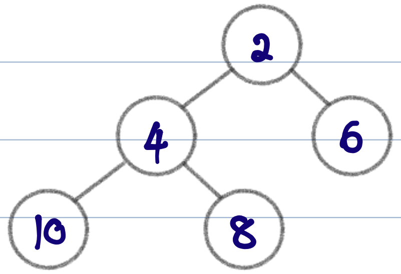
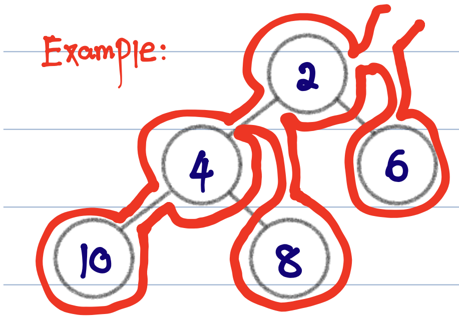
Inorder Traversal: Given a Binary Tree retur its key using inorder traversal.

Example:



Just loops are are used to do any operation on lists, recursion is same for trees, to traverse a tree, we use recursion.

Example:

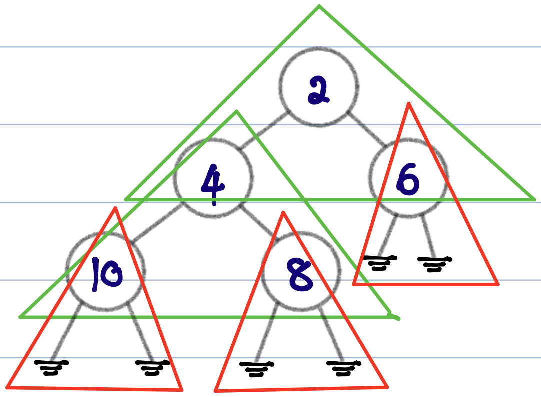


Based on the given tree, the values should be in this order : [10, 4, 8, 2, 6]

The idea is, until you have not reached the left most node, keep going and once reached there do the following :

1. get the left most node's key.
2. move to root of that node and get its key.
3. move to the right of that node and get its key.

The above three operations needs to be done for every single sub tree and one tree can have multiple sub tree.



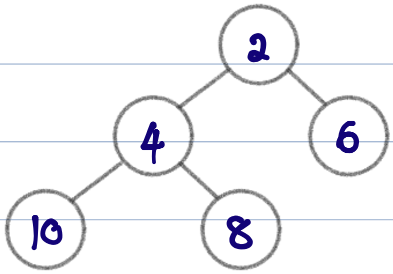
It can observed that we are having multiple sub tree, so for each of them the following operation will be done.

Algorithm:

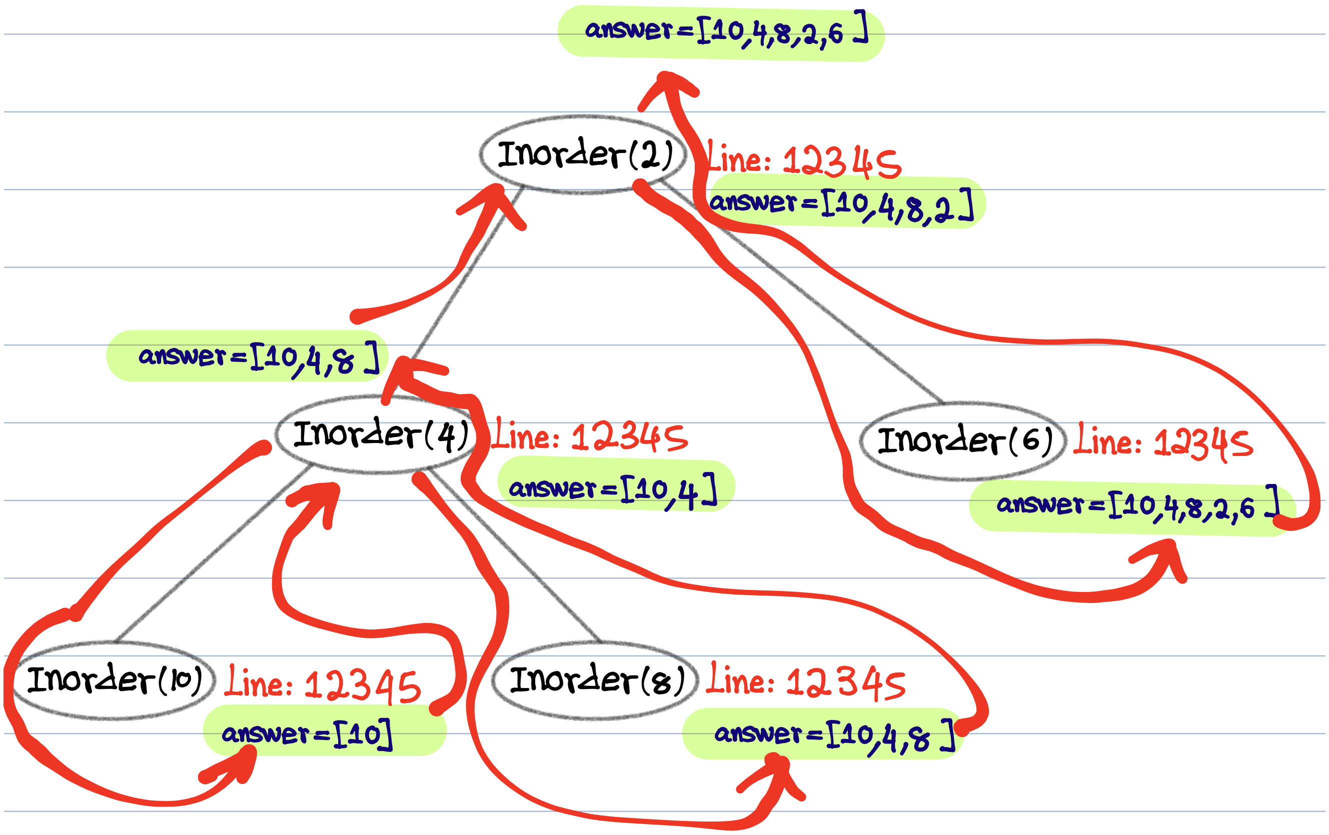
```
For every node ⓧ :
        nodes in ⓧ.left comes before ⓧ
        node in ⓧ.right comes after ⓧ
```

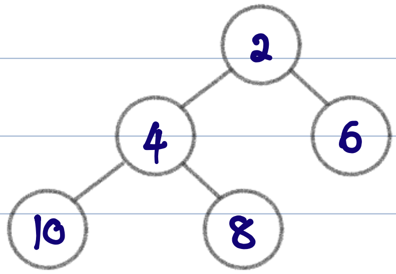
because we are doing the same task for every sub tree, recursive approach works best here.

```
def MainInorder(root):
Line  answer =[ ]
1     if root== None:
          return None
2     else:
          def Inorder(root):
3             Inorder(root.left)
4             answer.append(root.data)
5             Inorder(root.right)
          return answer
```



Implementation: our Initial root is ②, we will start from there. based on the algorithm everything on the left of ② comes before ②.


answer=[10,4,8,2,6]
Inorder(2)
Line: 12345
answer=[10,4,8,2]
answer=[10,4,8]
Inorder(4)
Line: 12345
answer=[10,4]
Inorder(6)
Line: 12345
answer=[10,4,8,2,6]
Inorder(10)
Line: 12345
answer=[10]
Inorder(8)
Line: 12345
answer=[10,4,8]



Inorder(root)
2

Is 2==None? No

Inorder(root.left)

Inorder(4)

Inorder(4)

Is 4==None? No

Inorder(root.left)

Inorder(10)

Inorder(10)

Is 10==None? No

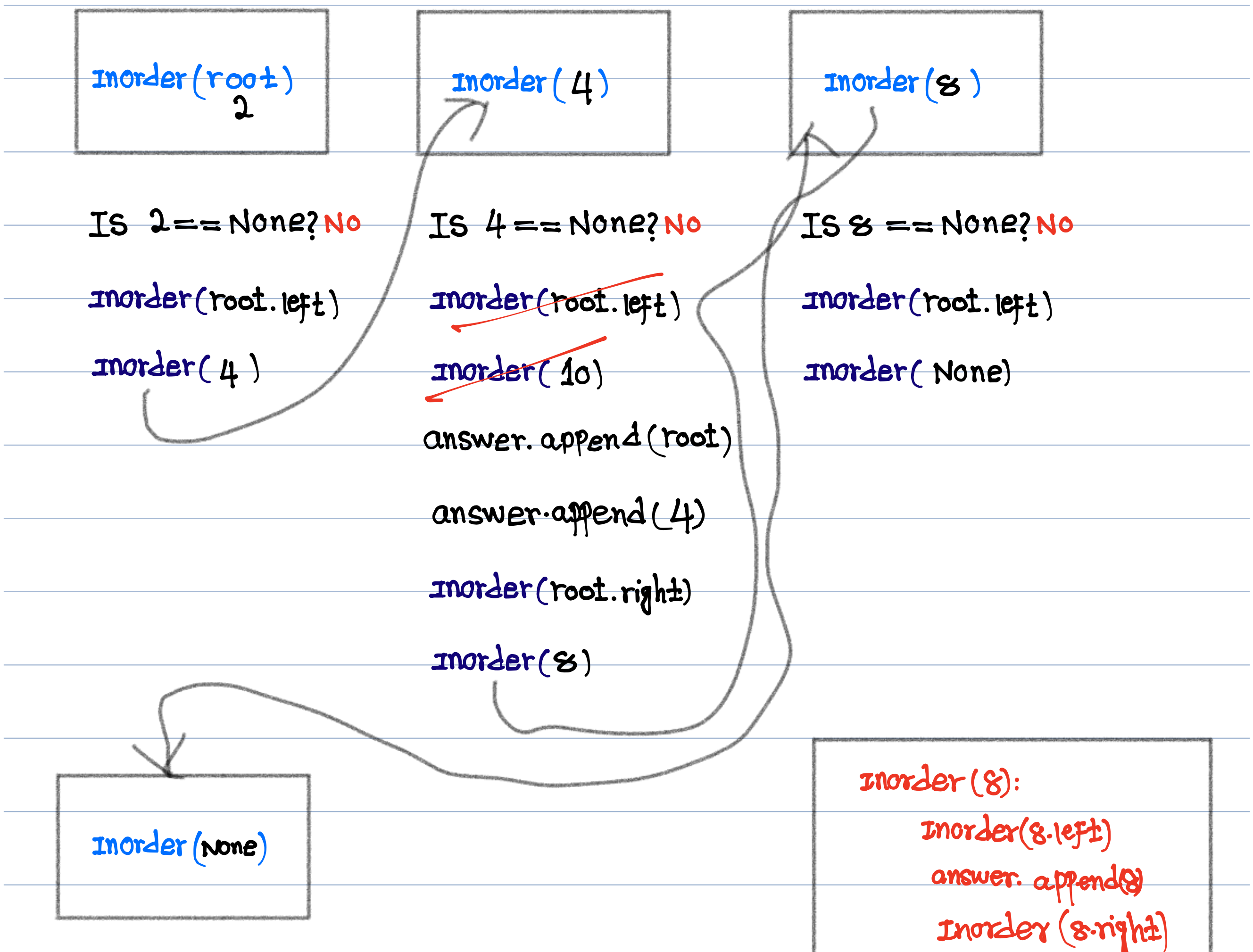
Inorder(root.left)

Inorder(None)

Inorder(None)

Is None==None? yes

answer.append(10) ← why? because the function that called Inorder(None) was Inorder(10).



Inorder(root)
2

Inorder(4)

Inorder(8)

Is 2==None? No

inorder(root.left)

inorder(4)

Is 4==None? No

~~inorder(root.left)~~

~~inorder(10)~~

answer.append(root)

answer.append(4)

inorder(root.right)

inorder(8)

Is 8 == None? No

inorder(root.left)

inorder(None)

Inorder(None)

Inorder(8):
Inorder(8.left)
answer.append(8)
Inorder(8.right)

Is None == None? yes

return // Move back the the function that called you.

answer.append(root) ← why? because the function that called

inorder(root.right) Inorder(None) was inorder(8).

inorder(None)