perplexity

explain below code by visualization:

def inorder(root):
if root:
inorder(root.left)
print(root.val, end=' ')
inorder(root.right)

The code defines and visualizes an *inorder traversal* of a binary tree. Here's how it works, using the provided tree and the attached visualization:

Tree Structure

The binary tree in your image and code is:

```
      1
     / \
    2   3
   / \   \
  4   5   6
```

Inorder Traversal Algorithm

The `inorder` function processes the tree as follows:

- Traverse the left subtree recursively.
- Visit (print) the current node.
- Traverse the right subtree recursively.

Step-by-Step Traversal on the Tree

The traversal order for this tree is:

1. Go to the leftmost node first (4).
2. Visit node 4.
3. Go up to node 2.
4. Visit node 2.
5. Visit node 5 (right child of 2).
6. Go up to root node 1.
7. Visit node 1.
8. Visit node 3 (right child of 1).

9. Visit node 6 (right child of 3).

So, the output order is: **4 → 2 → 5 → 1 → 3 → 6**.

Visualization Explanation

- Each circle in the image is a node in the tree.
- The traversal order is shown in the plot title:
  Binary Tree and Inorder Traversal Order: 4 → 2 → 5 → 1 → 3 → 6
- The traversal visits nodes from the leftmost leaf, up through their parents, then down the right subtrees.

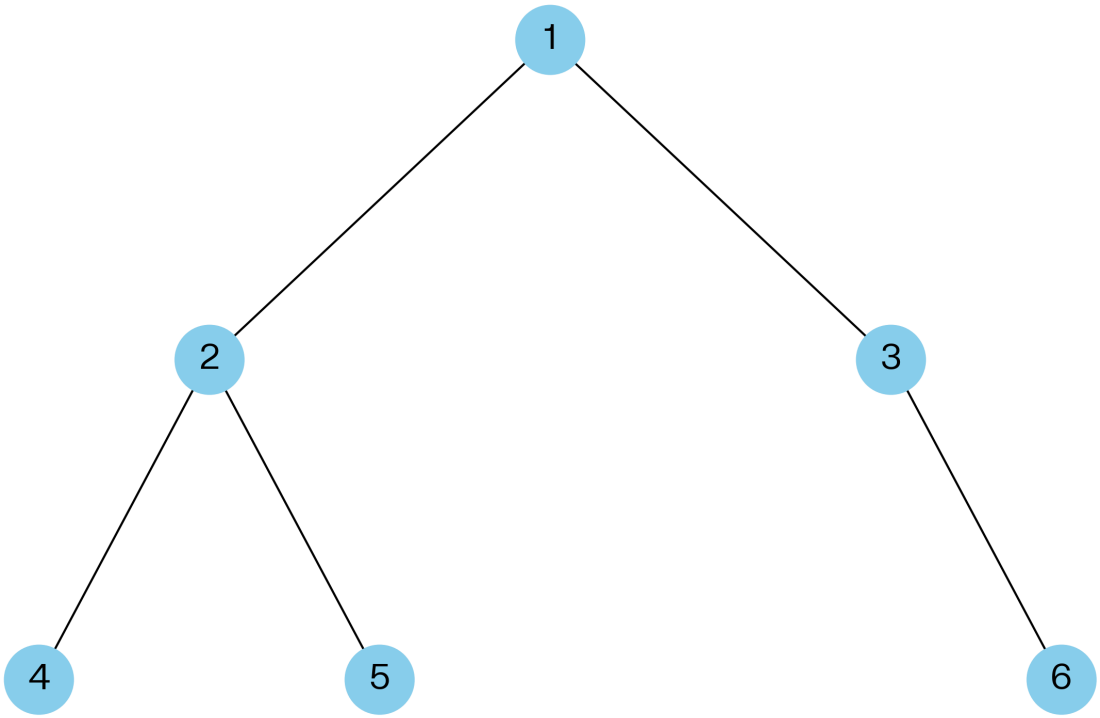
Summary Table

Step	Node Visited	Description
1	4	Leftmost node of the tree
2	2	Parent of 4
3	5	Right child of 2
4	1	Root node
5	3	Right child of 1
6	6	Right child of 3

Conclusion

The code and image together show how *inorder traversal* visits each node in the tree: left subtree first, then the node itself, then the right subtree. The result for this tree is ****.

⁂


1
2
3
4
5
6

⁂