🌳 1. What is a Tree?

A **tree** is a hierarchical data structure with nodes. Each node has a value and references to child nodes.
A **binary tree** is a tree where each node has at most two children (left and right).

🧱 2. Binary Tree Node Definition

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

📐 3. Binary Search Tree (BST)

A **BST** is a binary tree where:

- Left child < Parent node
- Right child > Parent node

🔁 4. Tree Traversals

a. Preorder Traversal (Root, Left, Right)

```
def preorder(root):
    if root:
        print(root.val, end=" ")
        preorder(root.left)
        preorder(root.right)
```

b. Inorder Traversal (Left, Root, Right)

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

c. Postorder Traversal (Left, Right, Root)

```
def postorder(root):
    if root:
        postorder(root.left)
        postorder(root.right)
        print(root.val, end=" ")
```

d. Level Order Traversal (Breadth-First)

```
from collections import deque

def level_order(root):
    if not root:
        return
    queue = deque([root])
    while queue:
        node = queue.popleft()
        print(node.val, end=" ")
        if node.left:
            queue.append(node.left)
        if node.right:
            queue.append(node.right)
```

🧪 5. Practice Problems

1. Create a Binary Tree and Print All Traversals

```
# Example tree:
#     1
#    / \
#   2   3
root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)

print("Preorder:")
preorder(root)     # Output: 1 2 3

print("\nInorder:")
inorder(root)      # Output: 2 1 3

print("\nPostorder:")
postorder(root)    # Output: 2 3 1

print("\nLevel Order:")
level_order(root)  # Output: 1 2 3
```

2. Insert a Value into a BST

```
def insert_bst(root, val):
    if not root:
        return TreeNode(val)
    if val < root.val:
        root.left = insert_bst(root.left, val)
    else:
        root.right = insert_bst(root.right, val)
    return root
```