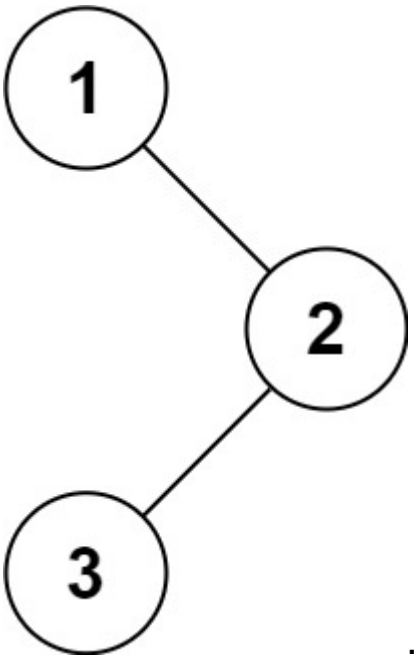
# 144. Binary Tree Preorder Traversal

Given the `root` of a binary tree, return *the preorder traversal of its nodes' values*.

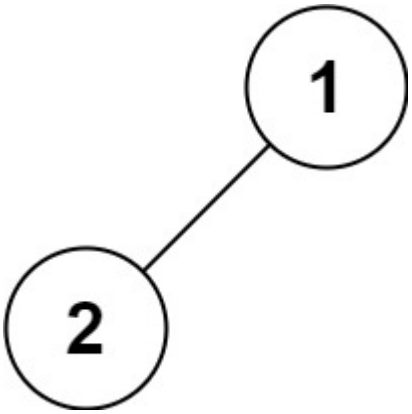
**Example 1:**




**Input:** root = [1,null,2,3]
**Output:** [1,2,3]
Ex2:

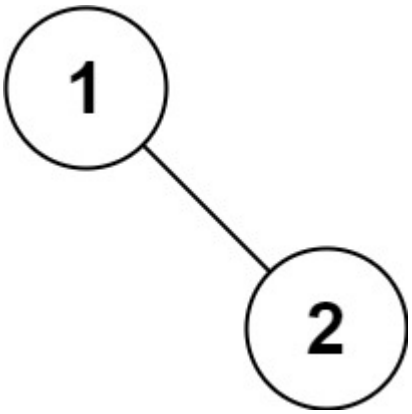
**Input:** root = []
**Output:** []
ex3:

**Input:** root = [1]
**Output:** [1]



**Input:** root = [1,2]
**Output:** [1,2]

**Example 5:**




**Input:** root = [1,null,2]
**Output:** [1,2]

```
class Pair:
    def __init__(self,node,state):
        self.node = node
        self.state = state
class Solution:
    def preorderTraversal(self, root: TreeNode) -> List[int]:
        if root is None:
            return
        pair = Pair(root,1)
        stack = [pair]
        res = []
        while len(stack)>0:
            temp= stack[-1]
            if temp.state==1:
                res.append(temp.node.val)
                temp.state = temp.state +1
                if temp.node.left:
                    stack.append(Pair(temp.node.left,1))
            elif temp.state==2:
                temp.state = temp.state +1
                if temp.node.right:
                    stack.append(Pair(temp.node.right,1))
            else:
                stack.pop()
        return res
```

```
def preorderTraversal(self, root):
        """
        :type root: TreeNode
        :rtype: List[int]
        """
        if not root: []
        res, stack = [], [root]
        while stack:
            node = stack.pop()
            if node:
                res.append(node.val)
                stack.append(node.right)
                stack.append(node.left)
        return res
```