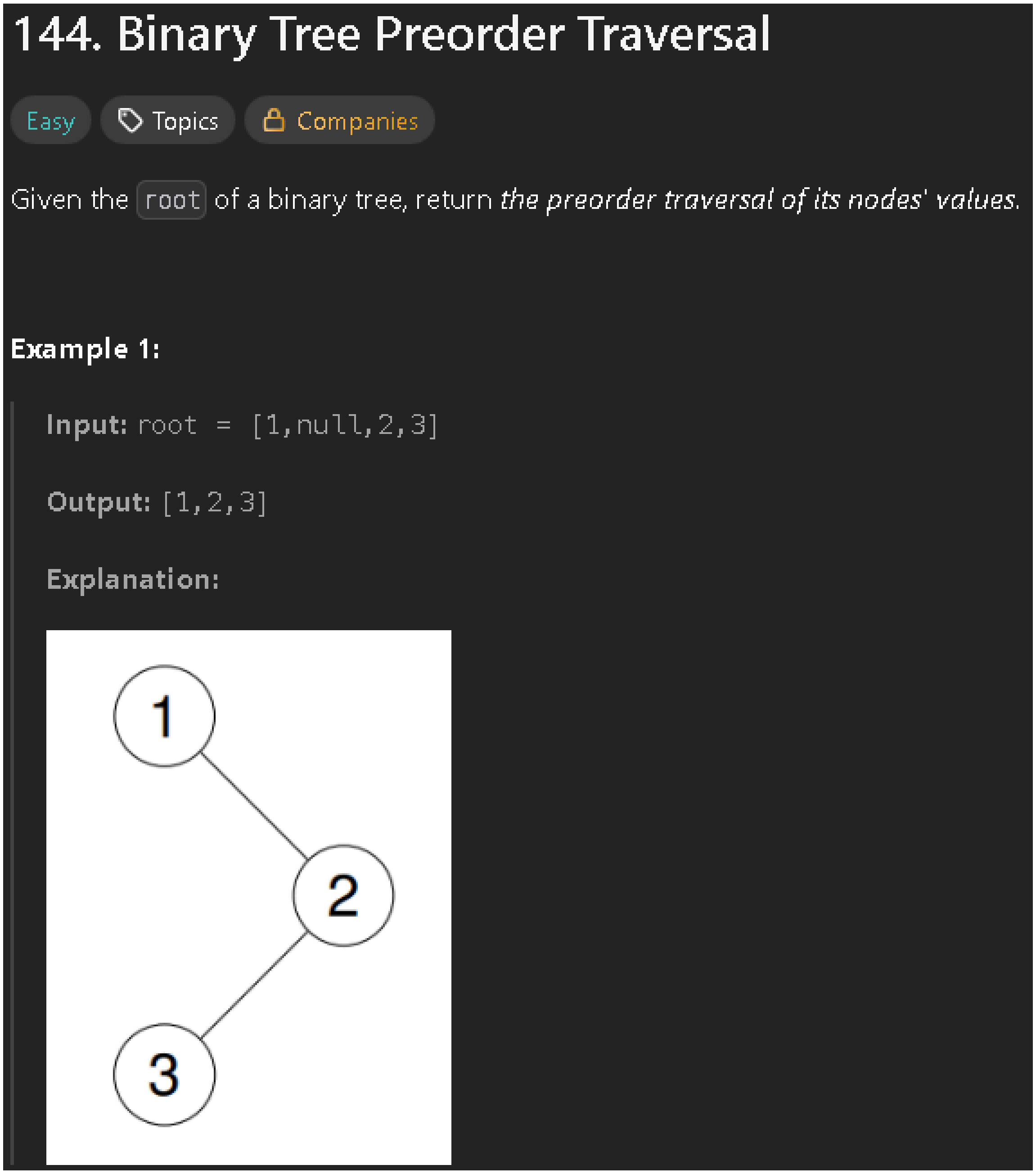
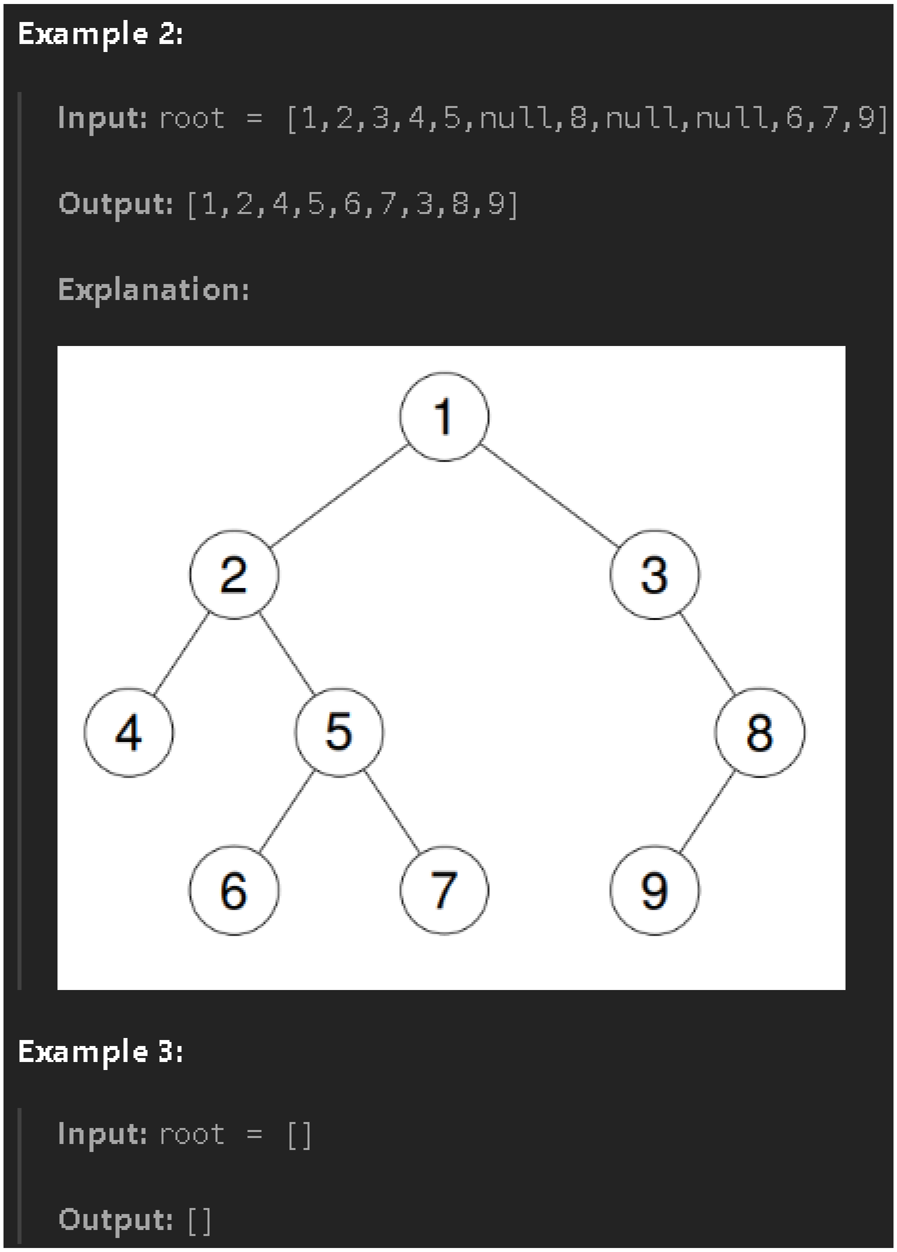


144. Binary Tree Preorder Traversal

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]`
>
> **Explanation:**

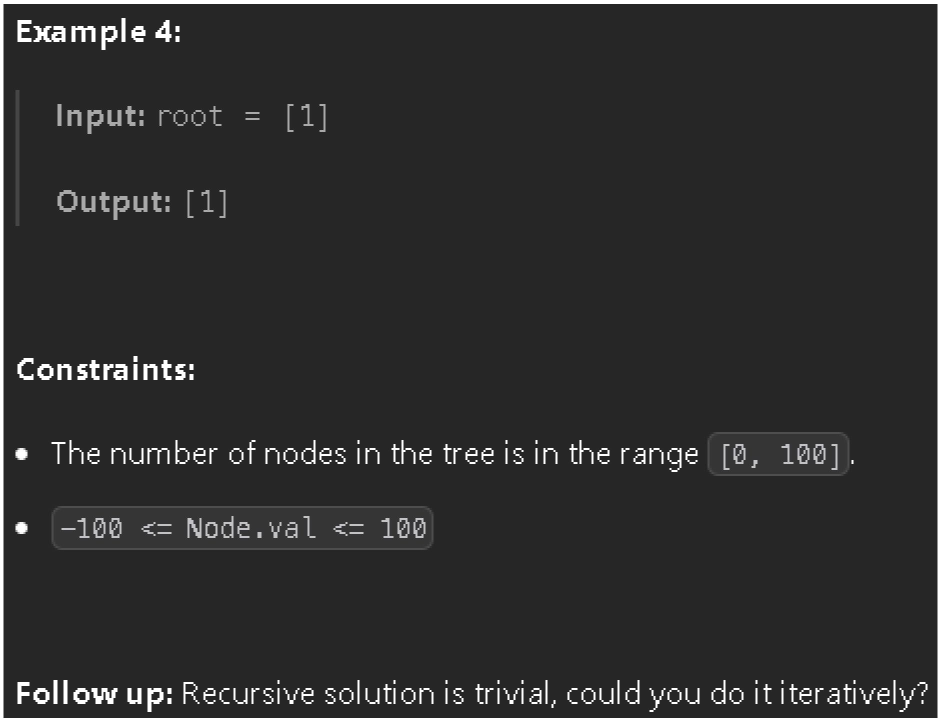


Example 2:

> **Input:** `root = [1,2,3,4,5,null,8,null,null,6,7,9]`
>
> **Output:** `[1,2,4,5,6,7,3,8,9]`
>
> **Explanation:**

Example 3:

> **Input:** `root = []`
>
> **Output:** `[]`



Example 4:

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

Constraints:

- The number of nodes in the tree is in the range `[0, 100]`.
- `-100 <= Node.val <= 100`

Follow up: Recursive solution is trivial, could you do it iteratively?

Python:

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right

from typing import List, Optional

class Solution:
    def preorderTraversal(self, root: Optional['TreeNode']) -> List[int]:
        if not root:
            return []

        stack, result = [root], []

        while stack:
            node = stack.pop()
            result.append(node.val)  # Visit the root

            # Push right first so left is processed first
            if node.right:
                stack.append(node.right)
            if node.left:
                stack.append(node.left)

        return result
```

JavaScript:

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val, left, right) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.left = (left===undefined ? null : left)
 *     this.right = (right===undefined ? null : right)
 * }
 */
/**
 * @param {TreeNode} root
 * @return {number[]}
 */
var preorderTraversal = function(root) {
    let result = [];

    function dfs(node) {
        if (!node) return;
        result.push(node.val);     // 1. Visit root
        dfs(node.left);            // 2. Traverse left
        dfs(node.right);           // 3. Traverse right
    }

    dfs(root);
    return result;
};
```

Java:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
import java.util.*;

class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> result = new ArrayList<>();
        preorder(root, result);
        return result;
    }

    private void preorder(TreeNode node, List<Integer> result) {
        if (node == null) return;
        result.add(node.val);           // 1. Visit root
        preorder(node.left, result);    // 2. Traverse left subtree
        preorder(node.right, result);   // 3. Traverse right subtree
    }
}
```