昨天在群里问同学怎么使用 python 前序遍历求二叉树的深度，一个同学回答了一个后序遍历的答案。

```python
# 后续遍历，从下至上
class Solution:
    def maxDepth(self, root: TreeNode) -> int:
        if not root:
            return 0
        return 1 + max(self.maxDepth(root.left), self.maxDepth(root.right))
```

今天我又想了一下，顺便弄清楚了 python 写回溯时，在添加结果 res.append(path)时，有时候要用 res.append(list(path))，有时候要用 res.append(path)。不知道别的同学有没有这个疑惑：

例如力扣《面试题 08.07 无重复字符串的排列组合》，就直接使用 res.append(path)，但是如果题目给的是 S = [1, 2, 3]，再使用 res.append(path)最后就会返回[[],[],[],[],[],[]]，没有试验过的小伙伴可以试试。

原因在于 python 传递参数的时候，不允许选择采用传值还是传引用。Python 参数传递采用的是“传对象引用”的方式，这种方式相当于传值和传引用的一种综合：

如果函数收到的是一个可变对象（比如字典或者列表）的引用，就能修改对象的原始值，相当于通过“传引用”来传递对象；如果函数收到的是一个不可变对象（比如数字、字符或者元组）的引用，就不能直接修改原始对象，相当于通过“传值“来传递对象。

```python
# path可变的(列表、字典)，引用的是地址
res = []
path = [1,2,3]
res.append(path)
res.append(list(path))
res.append(path)
print(res)
path.pop()
print(res)
```

```
[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
[[1, 2], [1, 2, 3], [1, 2]]
```

```python
# path不可变（str、元组、数字），加入的是值
res = []
path = '123'
res.append(path)
res.append(path)
res.append(path)
print(res)
path = path[:-1]
print(res)
```

```
['123', '123', '123']
['123', '123', '123']
```

res.append(list(path))可以理解为 res.append(path.copy())。

可以理解为不可变的变量会在新函数中产生了新变量（只是名字一样，但地址不同），可变的变量始终为一个变量（地址相同）。

基于这个思路，我写了 python 前序遍历求二叉树的深度：

```python
# 前序遍历，从上至下
class Solution:
    def maxDepth(self, root: TreeNode) -> int:
        if not root:
            return 0

        def dfs(root, level):
            if not root.left and not root.right:
                if level > res[0]:
                    res[0] = level

            if root.left:
                # level为一个数字，是不可变的，每个新函数中都会新产生一个变量（变量的作用域）
                dfs(root.left, level + 1)
            if root.right:
                dfs(root.right, level + 1)

        # 因为列表可变，地址相同，操作的都是这一个变量
        res = [0]
        dfs(root, 1)
        return res[0]
```