

# Python递归代码模板

```
def recursion(level, param1, param2, ...):  
    # recursion terminator  
    if level > MAX_LEVEL:  
        process_result  
        return  
  
    # process logic in current level  
    process(level, data...)  
  
    # drill down  
    self.recursion(level + 1, p1, ...)  
  
    # reverse the current level status if needed
```

递归终结条件

处理当前层逻辑

下探到下一层

清理当前层

首先，写上递归终止条件；第二步，处理当前层逻辑；第三步，下探到下一层；最后，清理当前层（有时不需要）

## 思维要点

1. 不要人肉进行递归（最大误区）
2. 找到最近最简方法，将其拆解成可重复解决的问题（重复子问题）
3. 数学归纳法思维

## 分治代码模板

# 分治代码模板

```
def divide_conquer(problem, param1, param2, ...):  
    # recursion terminator  
    if problem is None:  
        print_result  
        return  
    # prepare data  
    data = prepare_data(problem)  
    subproblems = split_problem(problem, data)  
    # conquer subproblems  
    subresult1 = self.divide_conquer(subproblems[0], p1, ...)  
    subresult2 = self.divide_conquer(subproblems[1], p1, ...)  
    subresult3 = self.divide_conquer(subproblems[2], p1, ...)  
    ...  
    # process and generate the final result  
    result = process_result(subresult1, subresult2, subresult3, ...)  
  
    # revert the current level states
```

首先，终止条件写出来，接着，处理当前逻辑（把这个大问题如何分解成子问题），第三步，调用函数，下探一层，最后，组装结果，返回

**关键：如何把大问题拆分成子问题，子结果如何合并**