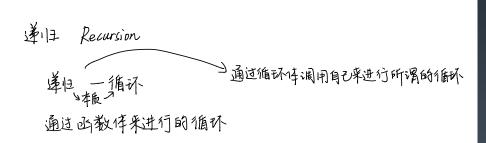
递归 一节点的定义 一重复性 (自相似性)



递归 Recursion

- 1. 从前有个山
- 2. 山里有个庙
- 3. 庙里有个和尚讲故事
- 4. 返回1

```
计算 n!

n! = 1*2*3*...*n

def Factorial(n):
    if n <= 1:
        return 1
    return n * Factorial(n - 1)

6*factorial(6)
6*factorial(5)
6*(5*factorial(4))
6*(5*(4*factorial(3)))
6*(5*(4*(3*factorial(2))))
6*(5*(4*(3*(2*factorial(1)))))
6*(5*(4*(3*(2*1))))
6*(5*(4*(3*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))
6*(5*(4*(3)*2)))</pre>
```

```
Java 代码模版
 public void recur(int level, int param) {
                                               - Recursion Terminator 美田终结条件
    if (level > MAX_LEVEL) {
                                               写递且的函数开始的似,一定要先把函数
                                               i.e. 递归终止条件写上 要不然: 死循环
      return;
                                                -Process Logic in current level
    // process current logic
process(level, param);
                                                处理新题辑
                                                 -Driu Down 下探到下-层
                                              高数标记至前是哪一层, level +1
同时把相应跨数
    recur( level: level + 1, newParam);
                                                               PI, Pa, Pa 一放下去···
                                              the current level
                                              needed 清理蓟鼠
```

思维要点

- 1. 不要人肉进行递归(最大误区)
- 江 找最近重复的题
- 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)

比较复杂的程序(逻辑较多) 却可以用五行/+行语句解决 why -> 2 复杂 文和多可重复性 逻辑