SCNU - UOA
OBJECT-ORIENTED PROGRAMMING

Lecture 09： Recursion

By Robert from UOA & Jingong Li from SCNU

1. Objectives

- To understand that complex problems that may otherwise be difficult to solve may have a simple recursive solution.
- To learn how to formulate programs recursively.
- To understand and apply the **three laws of recursion**.
- To understand **recursion as a form of iteration**.
- To implement the **recursive formulation of a problem**.
- To understand how recursion is implemented by a computer system.

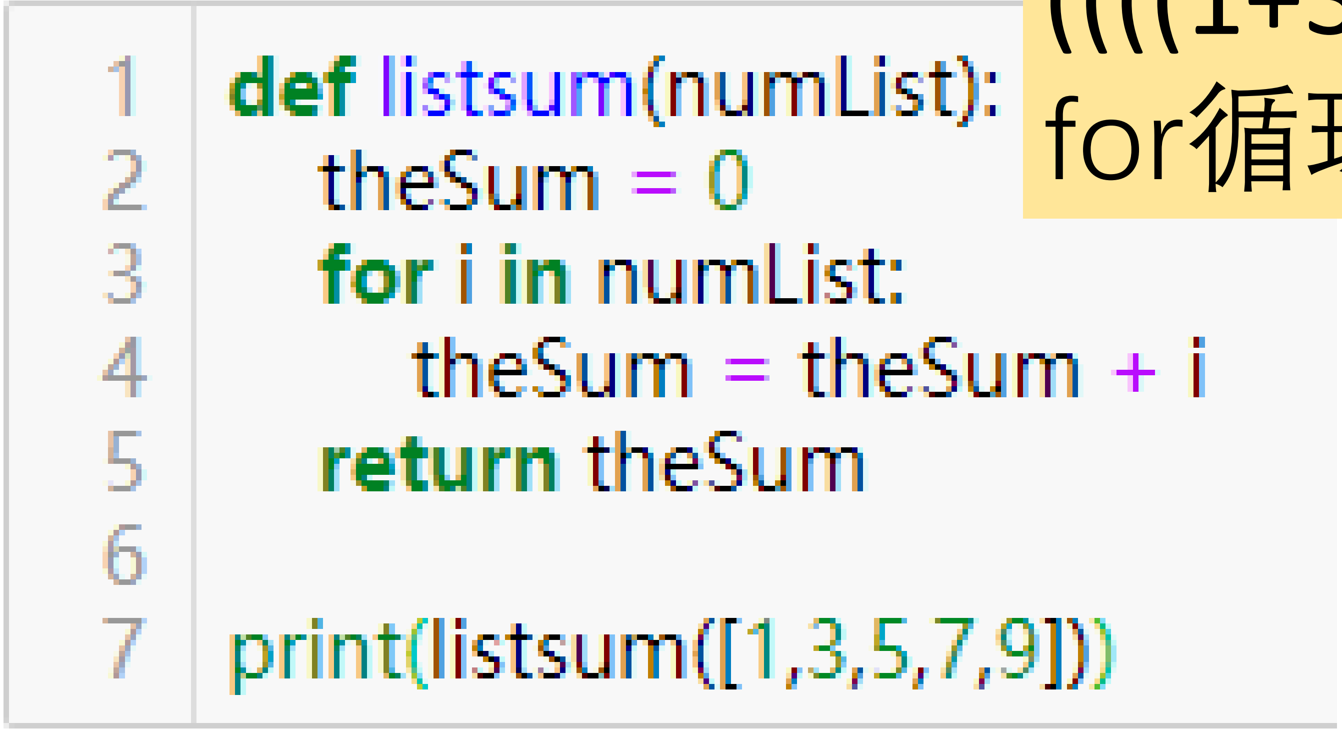
2. What Is Recursion?

- **Recursion** is a method of solving problems that involves breaking a problem down into smaller and smaller subproblems until you get to a small enough problem that it can be solved trivially. Usually recursion involves a function calling itself. While it may not seem like much on the surface, recursion allows us to write elegant solutions to problems that may otherwise be very difficult to program.

递归：一个调用自身的函数

3. Calculating the Sum of a List of Numbers

- Suppose that you want to calculate the sum of a list of numbers such as: **[1,3,5,7,9].**



```
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum = theSum + i
    return theSum

print(listsum([1,3,5,7,9]))
```

((((1+3)+5)+7)+9)
for循环求和

3. Calculating the Sum of a List of Numbers

- Pretend for a minute that you do not have while loops or for loops. How would you compute the sum of a list of numbers?

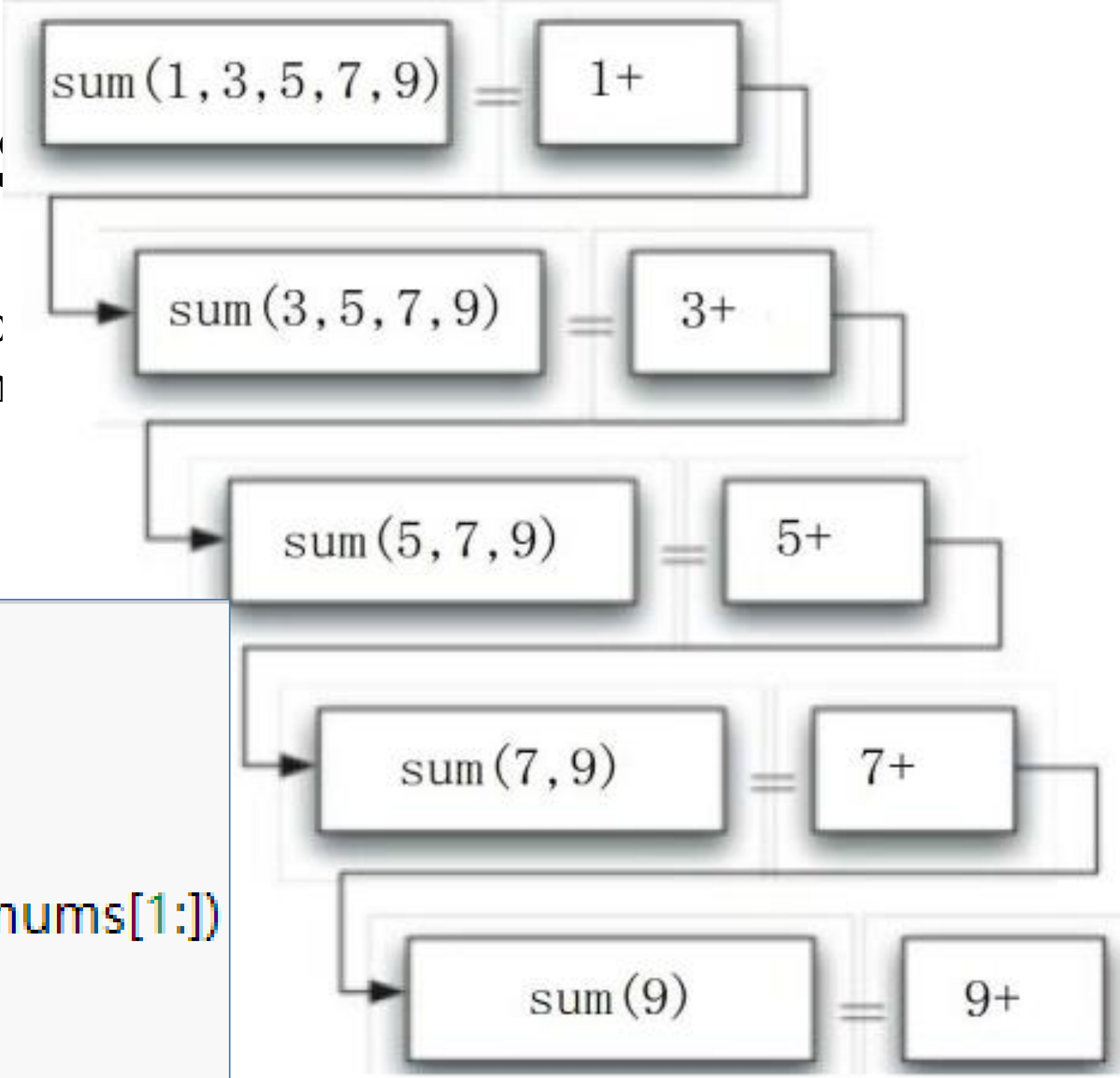
$$\begin{aligned} total &= (1+(3+(5+(7+9)))) \\ total &= (1+(3+(5+16))) \\ total &= (1+(3+21)) \\ total &= (1+24) \\ total &= 25 \end{aligned}$$

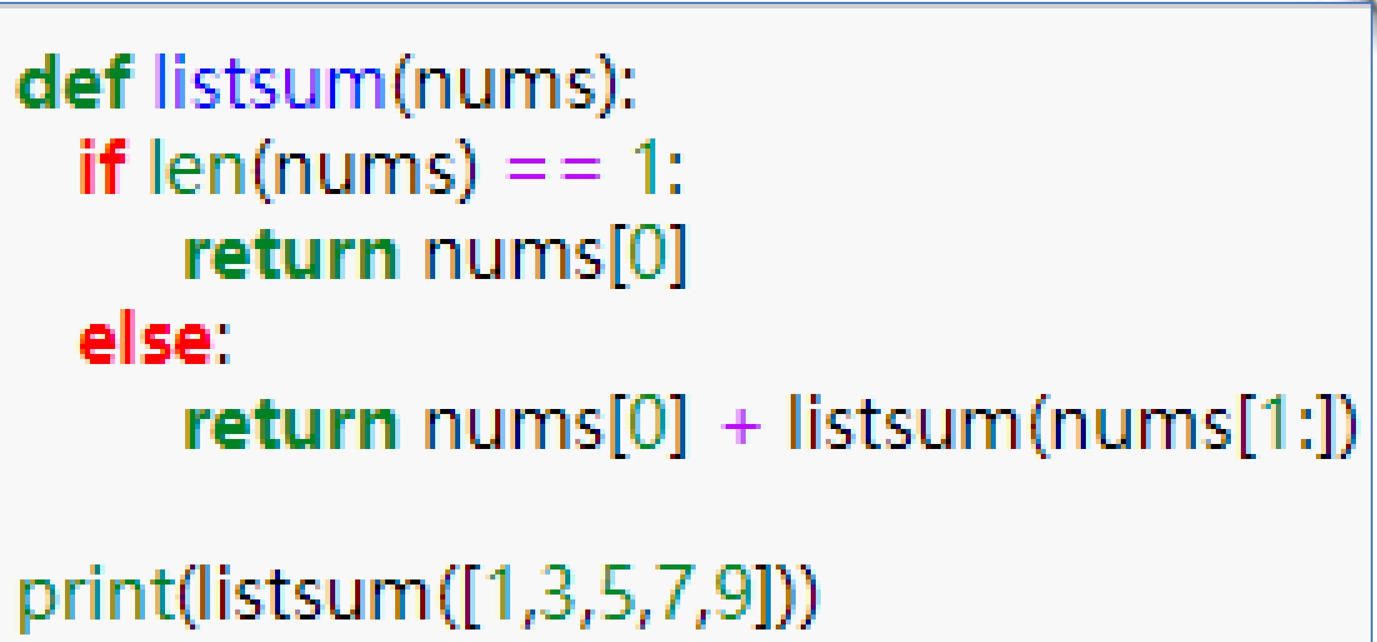
((((1+3)+5)+7)+9)

(1+(3+(5+(7+9))))

3. Calculating the S

- Pretend for a minute that yo
  How would you compute th

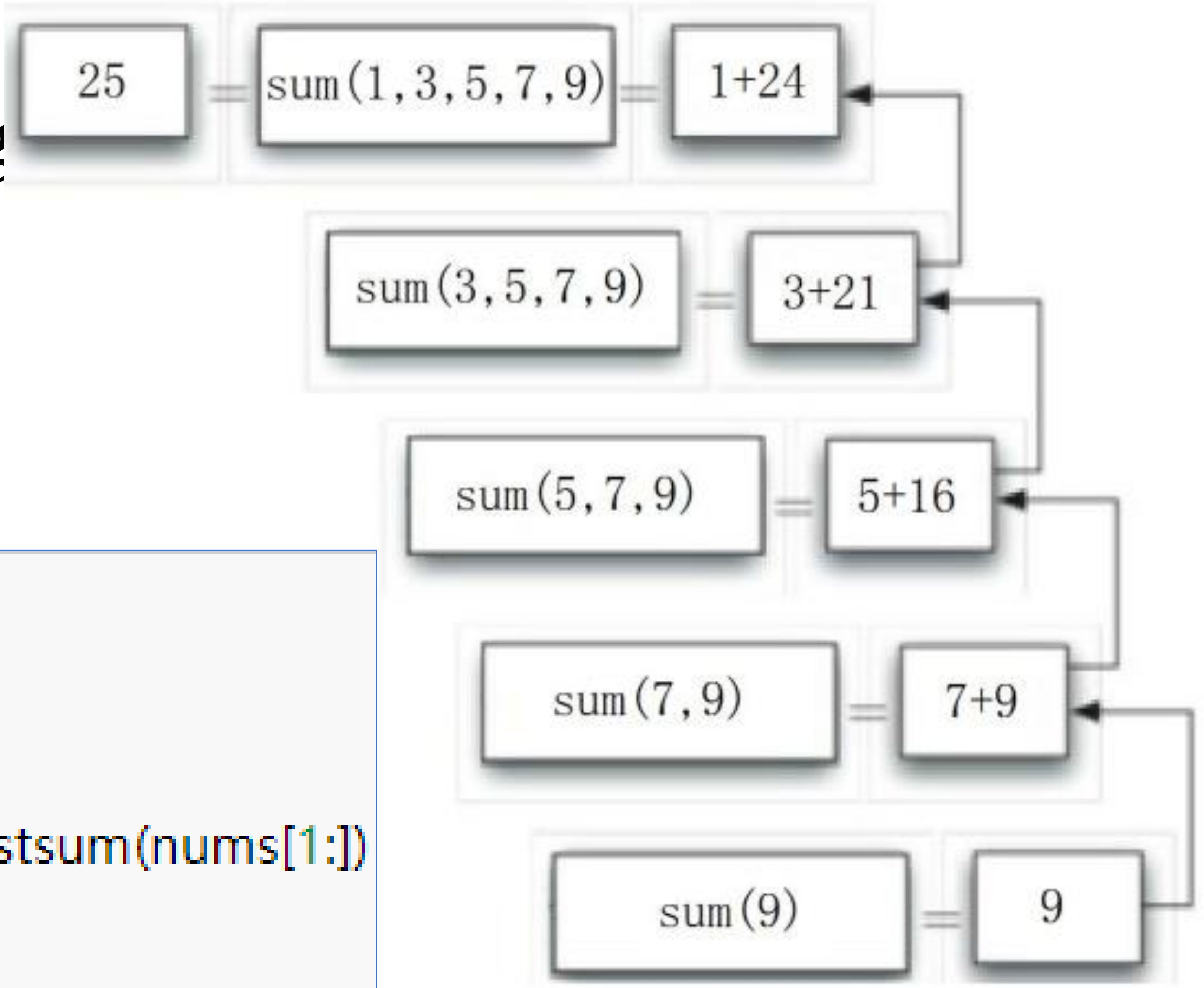




```
def listsum(nums):
   if len(nums) == 1:
        return nums[0]
   else:
        return nums[0] + listsum(nums[1:])

print(listsum([1,3,5,7,9]))
```

3. Calculating



```
def listsum(nums):
   if len(nums) == 1:
        return nums[0]
   else:
        return nums[0] + listsum(nums[1:])

print(listsum([1,3,5,7,9]))
```

4. The Three Laws of Recursion

递归算法三条重要的定律:

1、递归算法必须有个基本结束条件;

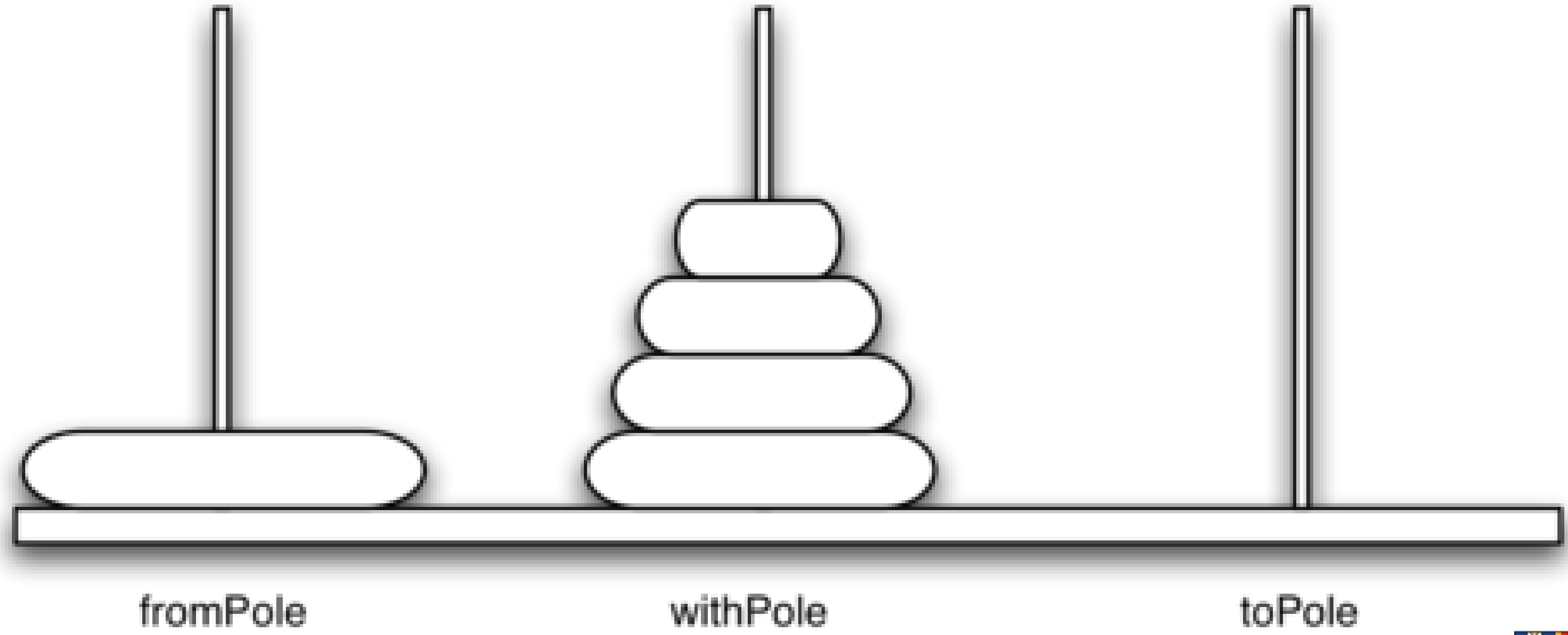
2、递归算法必须递归地调用自身 。

3、递归算法必须改变自己的状态、并向基本结束条件演进;

7. Complex Recursive Problems

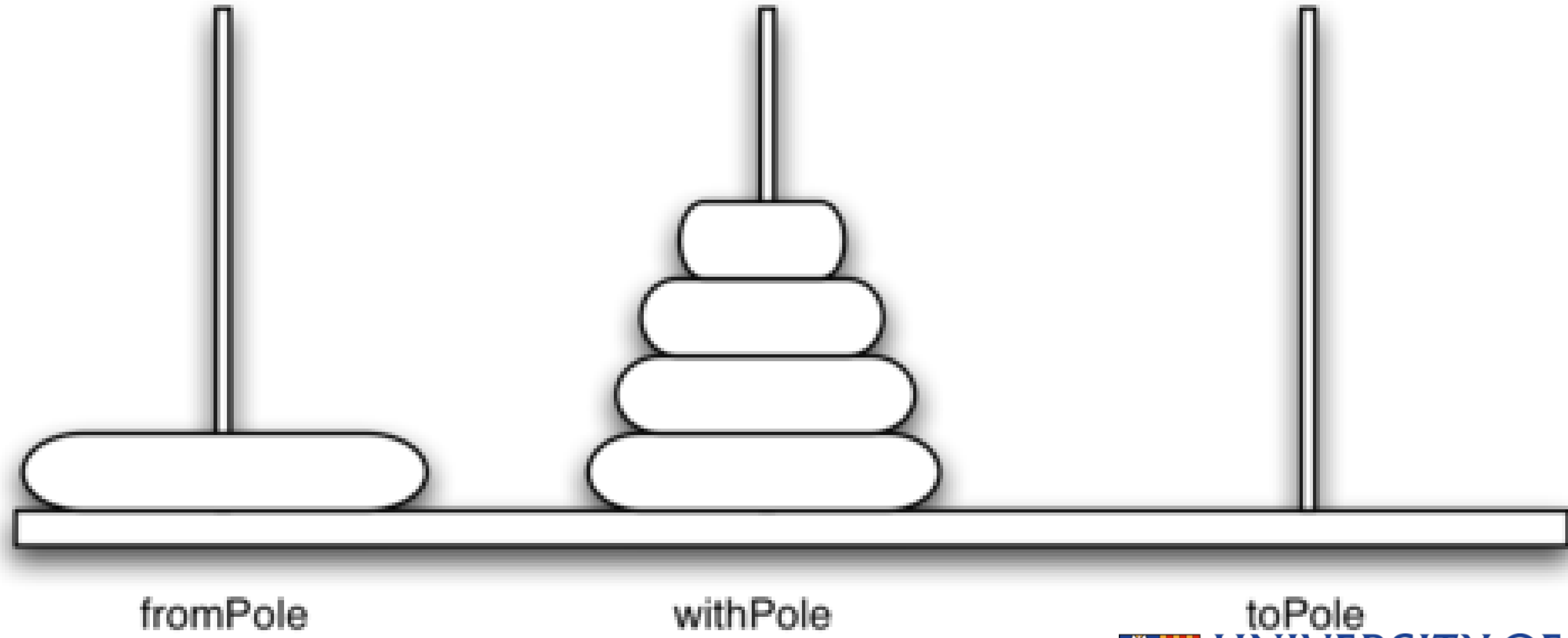
- Tower of Hanoi：

一次只能移动一个圆盘, 不能将大圆盘放在小圆盘之上



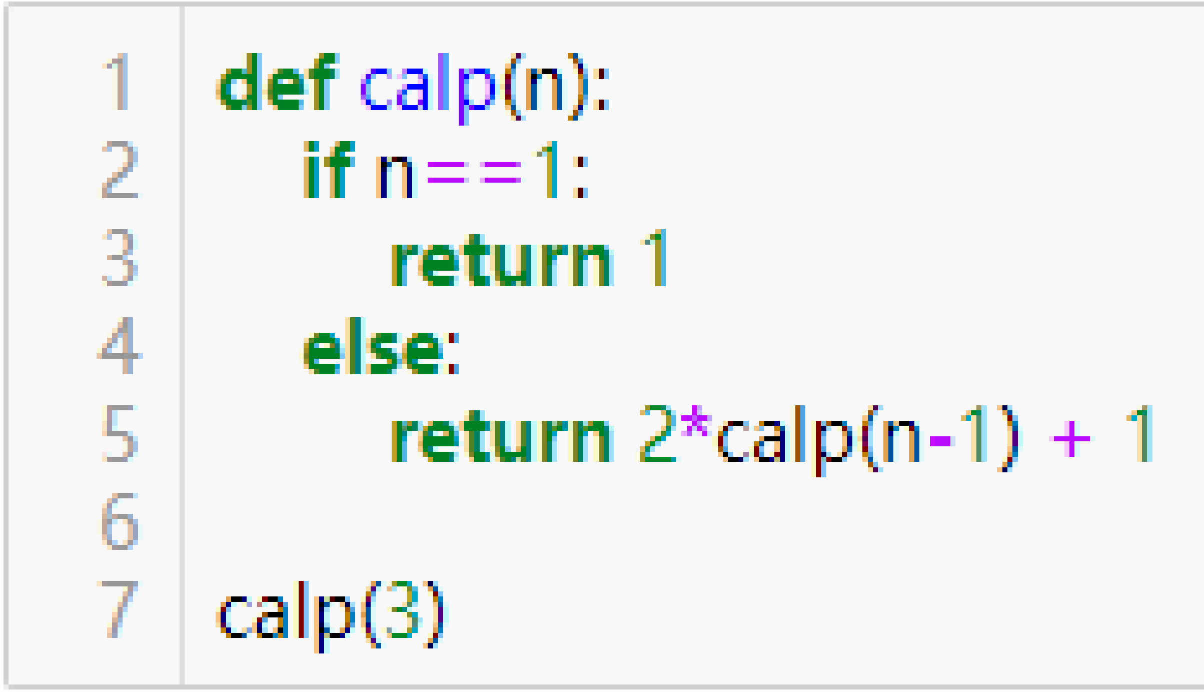
7. Complex Recursive Problems

- P(5) = P(4) + 1 + P(4) = 2*P(4) +1
- P(4) = 2P(3) + 1
- P(3) = 2P(2) + 1
- P(2) = 2P(1) + 1
- P(1) = 1



7. Complex Recursive Problems

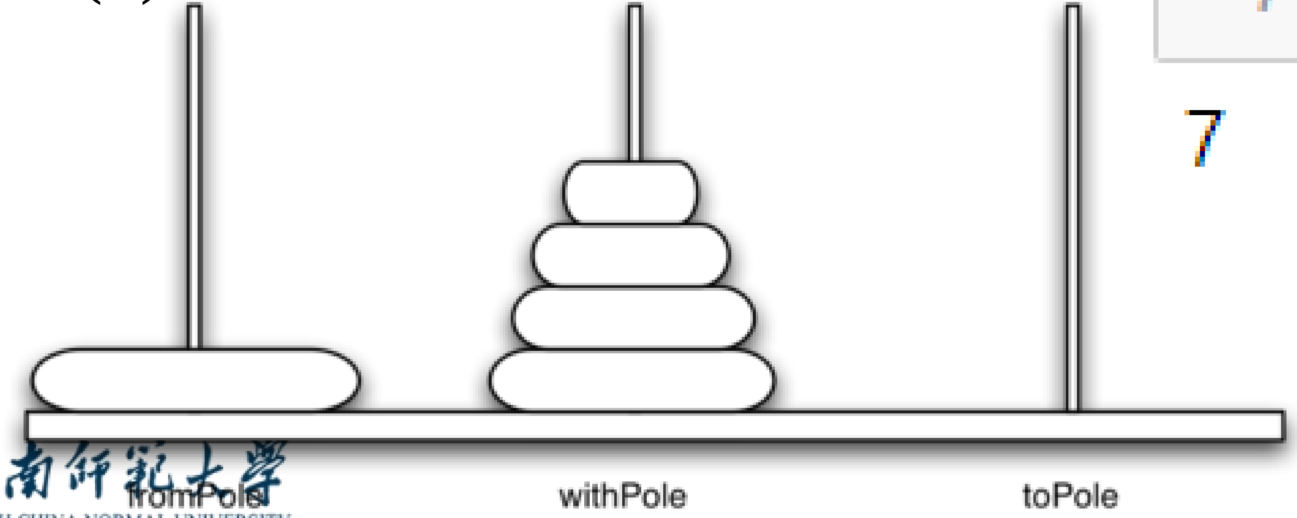
- P(5) = P(4) + 1 + P(4) = 2*P(4) +1
- P(4) = 2P(3) + 1
- P(3) = 2P(2) + 1
- P(2) = 2P(1) + 1
- P(1) = 1



```python
def calp(n):
    if n==1:
        return 1
    else:
        return 2*calp(n-1) + 1

calp(3)
```

```
7
```



SOUTH CHINA NORMAL UNIVERSITY

UNIVERSITY OF ABERDEEN

Thank You