第5课 递归算法

递归算法

在定义一个函数或过程时出现调用自身的成分,称之为递归。

计算fx=1+2+3+4+5的值：

```
def  fx(a) :
    if a <= 1 :
        return 1
    else:
        return a + fx(a - 1)
                        #调用自身
print(fx(5))
```

递归算法

递归程序设计是程序设计中的一种重要的方法，它使许多复杂的问题变得简单，容易解决了。

就递归算法而言并不涉及高深数学知识，但要建立递归概念、深入了解递归过程也不容易。

递归应用实例——阶乘计算

数学中阶乘定义：

5的阶乘：5！=5×4×3×2×1

4的阶乘：4！=4×3×2×1

整数n的阶乘：n!=n×(n-1)×……2×1

递归应用实例——阶乘计算

问题分析：

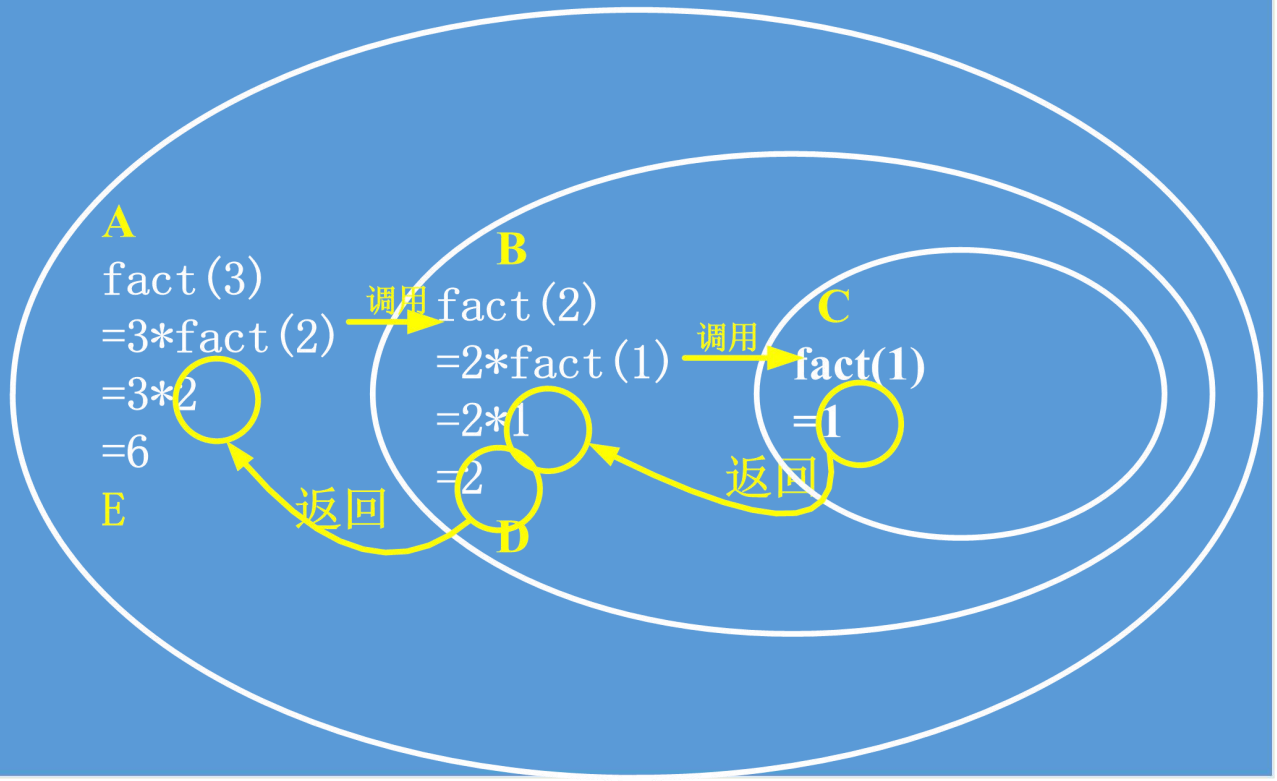
n!=n ×(n-1)!，（例如：5！=5×4！），而1！=1

求n的阶乘转化为求n-1的阶乘……

```
def fact(n):
    if n <= 1 :
        return 1
    else:
        return  n * fact (n - 1)
print(fact(3))
```

递归应用实例——阶乘计算

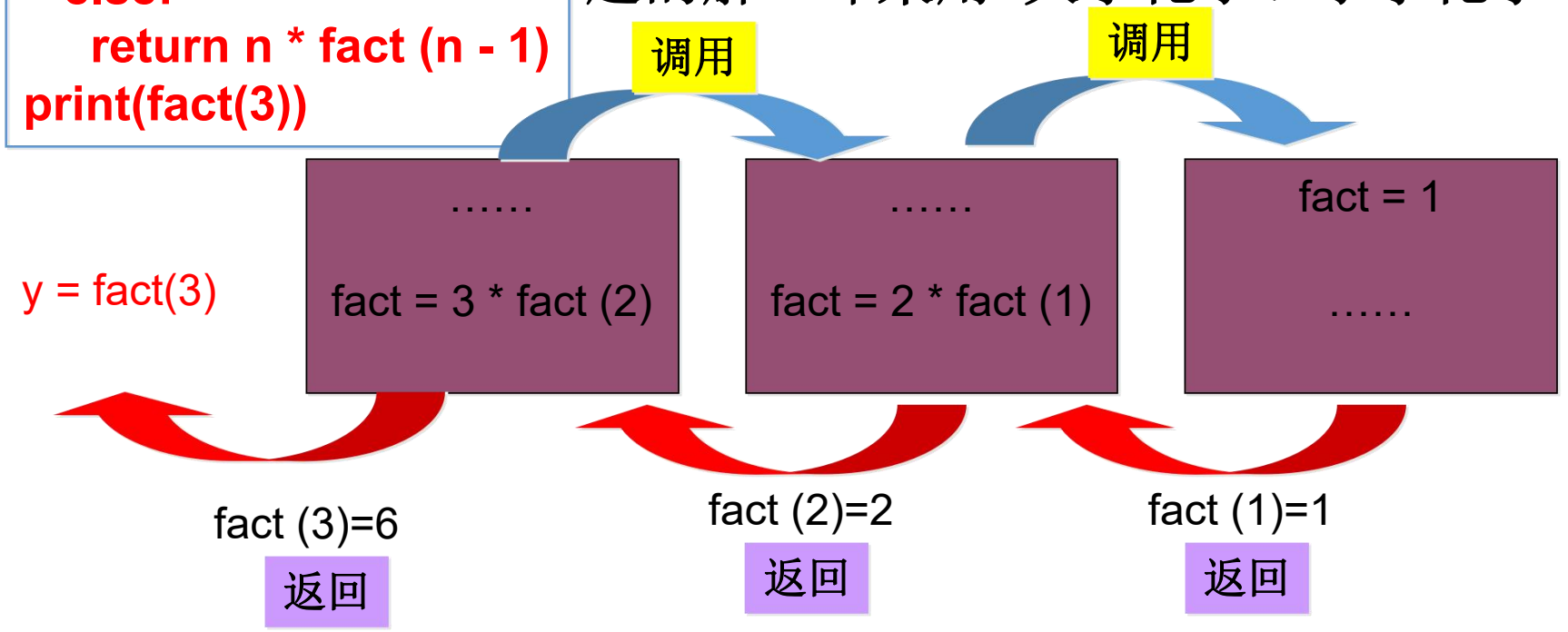
```
def fact(n) :
    if n <= 1:
        return 1
    else:
        return n * fact (n - 1)
print(fact(3))
```



当n=3时

```
def fact(n)
  if n <= 1:
    return 1
  else:
    return n * fact (n - 1)
print(fact(3))
```

递归算法的基本思想是把规模较大问题变成规模较小的、规模较小的问题又变成规模更小的问题，当问题小到一定程度时，可以直接得出它的解，从而得到原来问题的解。即采用“大事化小、小事化了”的基本思想。



递归算法的实现要点

(1)有明确的结束递归的边界条件(又称终止条件)以及结束时的边界值，可以通过条件语句(**If语句**)来实现

(2)函数的描述中包含其本身，即能用递归形式表示，且递归终止条件的发展。

```
def fact(n) :
  if n <= 1 :
    return 1
  else:
    return n * fact (n - 1)
```

边界条件

包含其本身

求阶乘使用循环实现

```
def fact(n) :
    s = 1
    for i in range(1,n+1)  :
        s = s * i
    return s
```

本例而言，同学们会认为递归算法可能是多余的，费力而不讨好。但许多实际问题不可能或不容易找到显而易见的递推关系，这时递归算法就表现出了明显的优越性。

小试牛刀：

1．下列有关递归的说法，错误的是(　D　)

A．递法算法的代码一般较少

B．递归算法一定要有终止条件

C．递归算法体现了大事化小的思想

D．递归函数中可以不包含条件控制语句

小试牛刀：

2．用递归算法求 1～n 个连续自然数的和的程序段代码如下：

```
def  sum (n):
    if n  =  1 :
     return 1
    else:
     return    (                    )
pritn(sun(5))
```

代码补全完整： n+sum(n-1)