PYTHON
编程基础

递归函数

概述

- 递归函数是指在一个函数内部通过调用自己来完成一个问题的求解。

- 当我们在进行问题分解时，发现分解之后待解决的子问题与原问题有着相同的特性和解法，只是在问题规模上与原问题相比有所减小，此时，就可以设计递归函数进行求解。

- 比如，对于计算n!的问题，可以将其分解为：n! = n*(n-1)!。可见，分解后的子问题(n-1)!与原问题n!的计算方法完全一样，只是规模有所减小。同样，(n-1)!这个子问题又可以进一步分解为(n-1)*(n-2)!，(n-2)!可以进一步分解为(n-2)*(n-3)!…，直到要计算1!时，直接返回1。

递归函数

例：编写递归函数计算n的阶乘。

```
1  def fac(n): #定义函数fac
2      if n==1: #如果要计算1的阶乘，则直接返回1（结束递归调用的条件）
3          return 1
4      return n*fac(n-1) #将计算n!分解为n*(n-1)!
5  print(fac(5)) #调用fac函数计算5的阶乘并将结果输出到屏幕
```

fac(5)=>5*fac(4)=>5*(4*fac(3))=>5*(4*(3*fac(2)))=>5*(4*(3*(2*fac(1))))
=>5*(4*(3*(2*1)))=>5*(4*(3*2))=>5*(4*6)=>5*24=>120

注意：

当问题规模较大时，递归调用会涉及到很多层的函数调用，一方面会由于栈操作影响程序运行速度，另一方面在Python中有栈的限制、太多层的函数调用会引起栈溢出问题（如将第5行的fac(5)改为fac(1000)则会报错）。

递归函数

例：使用非递归方式计算n的阶乘。

```
1  def fac(n): #定义函数fac
2      f=1 #保存阶乘结果
3      for i in range(2,n+1): #i依次取值为2至n
4          f*=i #将i乘到f上
5      return f #将计算结果返回
6  print(fac(5)) #调用fac函数计算5的阶乘并将结果输出到屏幕
```