**一、递归的定义**

函数作为一种代码封装，可以被其他程序调用，当然，也可以被函数内部代码调用。这种函数定义中调用函数自身的方式称为递归。就像一个人站在装满镜子的房间中，看到的影像就是递归的结果。递归在数学和计算机应用上非常强大，能够非常简洁的解决重要问题。

**二、递归的思想**

把规模大的问题转化为规模小的、具有与原来问题相同解法的问题来解决。在函数实现时，因为解决大问题的方法和解决小问题的方法往往是同一个方法，所以就产生了函数调用它自身的情况。

**三、递归的使用方法**

1、找到递归关系，即把一个复杂的问题转化为与它形式相似、但规模较小的问题

2、找到递归出口，即问题转化时，当规模足够小，可以直接求解

**四、递归法练习**

1. **数字乘阶段**

# 1、理解题目：  
# 阶乘： 从1开始乘到自己本身  
# 假设有3阶，它的阶乘： 1\*2\*3=6  
# 假设有4阶，它的阶乘： 1\*2\*3\*4=24  
  
# 2、写代码

# n 是阶数 func999 是计算阶乘的函数  
def func999(n):  
 if n == 0:  
 return 1 #程序出口，最后一次递归 返回1  
 else:  
 return n \* func999(n-1)  
  
# 调用阶乘函数  
x = func999(4)  
print(x)

**2、字符串反转**

**对于用户输入的字符串s，输出反转后的字符串。解决这个问题的基本思想是把字符串看作一个递归对象。**

def rev(s): # 反转字符串  
 if len(s) == 1:  
 return s  
 else:  
 return s[-1] + rev(s[:len(s) - 1])  
# 调用递归函数  
s = input()  
print(rev(s))

**3、斐波那契数列（1、1、2、3、5、8、13、21、34、……）**

**兔子繁殖问题：**

在700多年前，意大利著名数学家斐波那契在《算盘全集》中提到这样一个问题：一对兔子，从出生后第3个月起每个月都生一对兔子。小兔子长到第3个月后每个月又生一对兔子。假如兔子都不死，请问第1个月出生的一对兔子，第n个月有多少对兔子？

假设：F代表兔子数量，F(1)是第一个月，F(2)是第二个月.......F(N)是第N个月；

得出以下推论：

F(1)=1

F(2)=1

F(3)=F(1)+F(2)=1+1=2

F(4)=F(2)+F(3)

…………

F(N)=F(N-2)+F(N-1)

**代码：**

def fab(n):  
 if n <= 2:  
 return 1  
 else:  
 return fab(n - 1) + fab(n - 2)  
n = eval(input())  
print(fab(n))

**4、赶鸭子问题**

**题目描述：**

一个人赶着鸭子去每个村庄卖，每经过一个村子卖去所赶鸭子的一半又一只。这样他经过了七个村子后还剩两只鸭子，问他出发时共赶多少只鸭子？经过每个村子卖出多少只鸭子？

**分析**

设经过的村子为n (n = 0,1,2,...,7)，根据题目分析可知递归结束的出口: n = 7时，剩余鸭子数duck = 2;

分析递归体：从后向前推 n=7时 ，duck = 2， 由于每经过一个村子，卖去所赶鸭子的一半又一只，因此七个村子后剩余的鸭子数 duck[7]=duck[6]-(duck[6]/2+1)

反推duck[6] = (duck[7] + 1) \* 2

最终递归体: (duck[n+1] + 1) \* 2;

**综上**

n=7 duck=2

0<=n<7 duck=(duck(n+1) + 1) \* 2

**代码**

def duck(n):  
 if n == 7:  
 return 2  
 else:  
 return (duck(n + 1) + 1) \* 2  
  
print("鸭子的总数为:{}".format(duck(0)))  
sum = duck(0)  
for i in range(1, 8):  
 print("第{}个村庄卖出{}只鸭子，剩余鸭子数为{}".format(i, sum - duck(i), duck(i)))  
 sum = duck(i)