Python语言程序设计
Design and Programming of The Python Language

主讲教师：张小东
联系方式：z_xiaodong7134@163.com
答疑地点：宋健研究院514

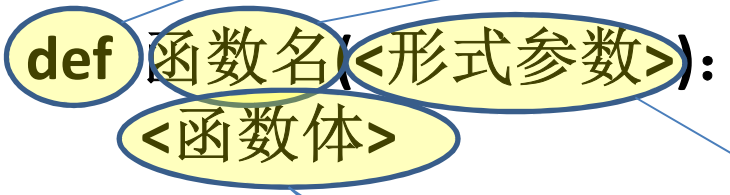
第4章 函数

主要内容

- 函数的定义与调用
- 变量的作用域
- 递归函数
- 结构化设计方法浅析

➢ 函数的定义

◆ 函数定义的一般格式



def定义函数的关键字

符合标符的命名规则

形式参数可没有，多个时用逗号隔开

函数体所有语句必须相对于第一行缩进

函数可以没有返回值，如果有使用**return**将数据返回，其格式为：

return <表达式>

➢ 函数的定义

【例4-1】编写函数求出区间[i, j]内所有整数的和

```
def mySum(i,j):
    s=0
    for k in range(i,j+1):
        s=s+k
    return s
```

◆ 参数传递方式：位置绑定

实参与形参按出现的位置一一对应

```
info('张三',30,'男')
```

```
def info(name,age,sex):
    print('name:',name,'age:',age,'sex:',sex)
```

➢ 函数的定义

◆ 参数传递方式：关键字绑定

实参与形参采用“形式参数名=数值”对应

```
def info(name,age,sex):
    print('name:',name,'age:',age,'sex:',sex)
```

```
info(age=30, name='张三',sex='男')
```

◆ 为形参指定默认值

```
info(age=30, name='张三')
```

```
def info(name,age,sex='男'):
    print('name:',name,'age:',age,'sex:',sex)
```

➢ 函数的定义

◆ 设定两种可变长参数

def 函数名(arg1, arg2,…,*tuple_args, **dic_arg)

元组变长参数　字典变长参数

```
def varArg(arg,other_arg='default',*tuple_arg,**dic_arg):
    print('无默认参数：',arg)
    print('有默认参数：',other_arg)
    print('元组变长参数：',tuple_arg)
    print('字典变长参数：',dic_arg)
```

➢ 函数的定义

◆返回多个数值

【例4-3】编写函数，计算三门课程的总分和平均分

```
def calc_grade(math, english, chinese):
    Sum=math+english+chinese
    Avg=float(Sum/3)
    return Sum,Avg
```

```
>>> a,b=calc_grade(88,76,85)
>>> a
249
>>> b
83.0
```

➢ 函数的定义

◆ lambda函数的定义

用单 行的表达式定义一个函数—lambda函数

函数名=labmbda 参数：表达式

```
g = lambda x,y,z:x*x+y*y+z*z
```

使用默认参数

```
g = lambda x,y=0,z=0:x*x+y*y+z*z
```

➢ 函数的调用

◆ 格式

函数名(<实际参数>)

◆ 函数出现的位置

（1）作为单独的语句出现，如

```
>>> calc_grade(88,76,85)
 (249, 83.0)
```

（2）出现在表达式里，如

```
a,b=calc_grade(88,76,85)
```

（3）作为实际参数出现在其他函数中，如

```
M=max(5000, mySum(1,100))
```

第4章 函数

主要内容

- 函数的定义与调用
- 变量的作用域
- 递归函数
- 结构化设计方法浅析

➢ 局部变量和全局变量

局部变量：只能在程序特定部分使用
全局变量：可以在文件中的任何地方使用

```
gl=9
def myAdd():
    ll=3
    return gl+ll
```

全局变量　局部变量

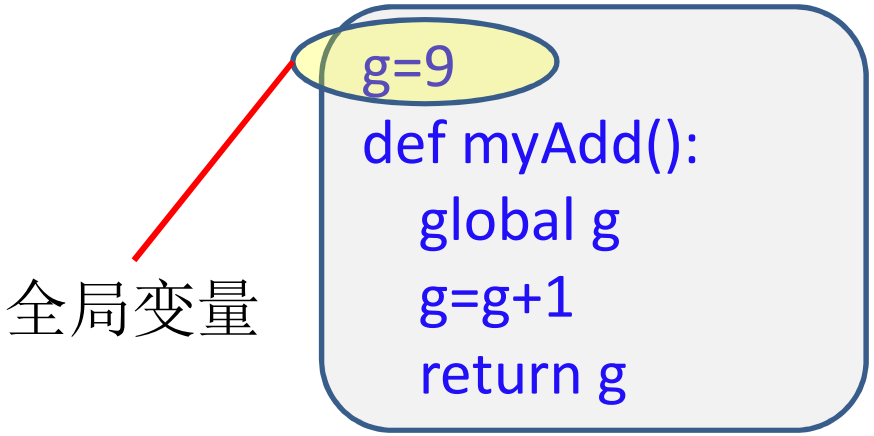
```
>>>print(myAdd())
>>>print(gl)
>>>print(ll)
```

```
gl=0
def myAdd():
    gl=3
    return gl+11
```

局部变量
屏蔽全局

➢ 全局变量局部使用

使用**global**



在函数内部使用**global**声明的全局变量，在函数外部也可以使用

➢ 内嵌函数及其作用域

【例4-2】内嵌函数

```
def f1():
    x=y=2
    def f2():
        y=3
        print('f2:x=',x)
        print('f2:y=',y)
    f2()
    print('f1:x=',x)
    print('f2:y=',y)
```

第4章 函数

主要内容

- 函数的定义与调用
- 变量的作用域
- 递归函数
- 结构化设计方法浅析

➢ 递归算法思想

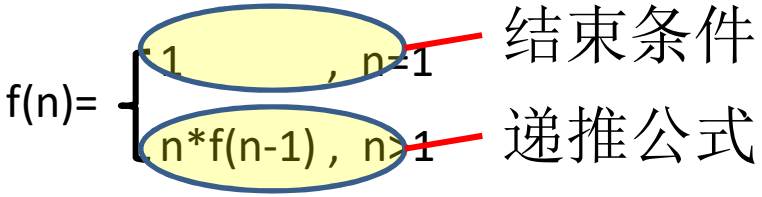
把一个复杂的大问题逐步转换为与原问题相似的小问题进行求解

递归算法的设计要点：

（1）递推公式

（2）递归结束条件

【例4-2】编程求n!




```
def f(n):
    if n==1:
        return 1
    return n*f(n-1)
```

➢ 递归算法思想

【例4-3】汉诺(Hanoi)塔问题。

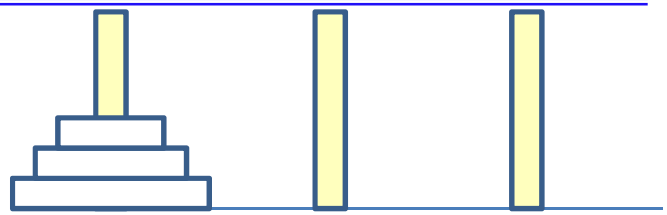


图4-1 汉诺塔

借助B将n个盘子从A移到C =
- 将一个盘子从A移到C　　n=1 — 结束条件
- 借助C将n-1个盘子从A移到B
  将一个盘子从A移到C
  借助A将n-1个盘子从B移到C　　n>1 — 递推公式

```
def Hanoi(n,ch1,ch2,ch3):
    if n==1:
        print(ch1,'->',ch3)
    else:
        Hanoi(n-1,ch1,ch3,ch2)
        print(ch1,'->',ch3)
        Hanoi(n-1,ch2,ch1,ch3)
```

谨慎使用递归算法，因为它们的简洁可能会掩盖其低效率的事实。

- 计算模型

$$\begin{cases} f(n) = \text{move}(1) & n = 1 \\ f(n) = 2f(n-1) + \text{move}(1) & n > 1 \end{cases}$$

1秒1个盘子

$2^{64}-1=18446744073709551615$秒

- n为规模，也是计算规模
- 核心操作为移动盘子
- 依据递推公式，两次递推之间，执行一次移动操作，因此有如下推导过程：

$T(n) = 2T(n-1)+1$

$=2[2T(n-2)+1]+1=2^2T(n-2)+2+1$

$=2^2[2T(n-3)+1]+2+1=2^3T(n-3)+2^2+2+1$

……

$=2^{i-1}[2T(n-i)+1]+2^{i-2}+2^{i-3}\ldots+2^0=2^iT(n-i)+2^i-1$

……

$=2^{n-1}T(n-(n-1))+2^{n-1}-1=2^{n-1}T(1)+2^{n-1}-1=2^n-1$

```
def Hanoi(n,ch1,ch2,ch3):
  if n==1:
    print(ch1,'->',ch3)
  else:
    Hanoi(n-1,ch1,ch3,ch2)
    print(ch1,'->',ch3)
    Hanoi(n-1,ch2,ch1,ch3)
```

第4章 函数

主要内容

- 函数的定义与调用
- 变量的作用域
- 递归函数
- 结构化设计方法浅析

➢ 自顶向下逐步求精的思想

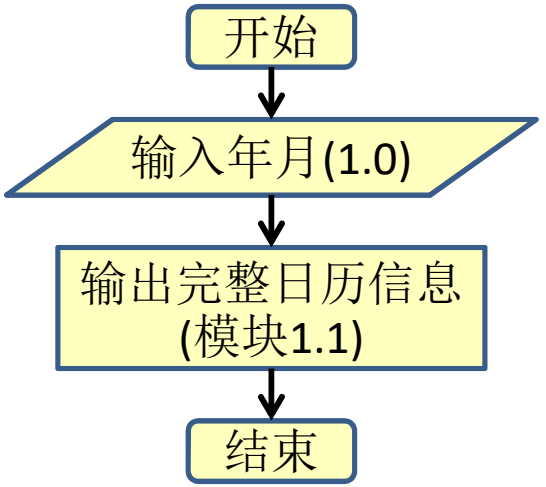
- ◆ 采用分治思想，将一个大的模块分解为不同的功能模块，简化问题，提高代码的复用性
- ◆ 从信息处理流程出发，将问题分解为不同的模块，细化模块，逐层向下分解。

➢ 自顶向下逐步求精的思想

【例4-4】已知1980年1月1日是星期二，现在要求根据用户输入的年份(≥1980)、月份在屏幕上打印出当月的日历，运行效果如下：

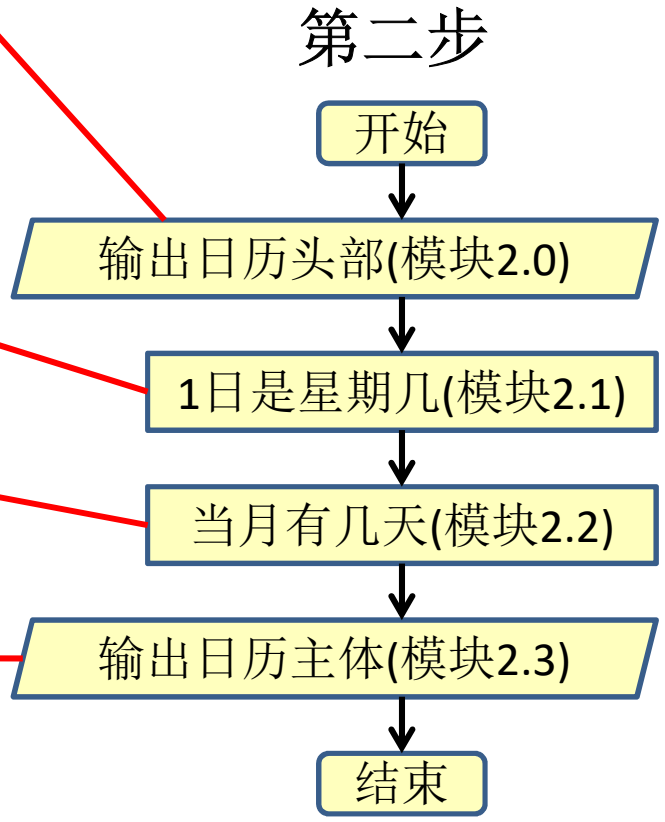
```
input year(yyyy):2017
input month(m):3
          March   2017
-----------------------------
 Sun Mon Tue Wed Thr Fri Sat
                   1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30  31
```

第一步



➢ 自顶向下逐步求精的思想

```
def pM(m,y):
    print('\t',gmn(m),' ',y)
    print('----------------------------')
    print(' Sun Mon Tue Wed Thr Fri Sat ')
    #计算星期几
    sD=gD(m,y)
    for i in range(0,sD):
        print('    ',end=" ")
    #计算这个月有多少天
    sDm=gDm(m,y)
    for i in range(1,sDm+1):
        if i<10:
            tme='   %d'%i
        else:
            tme='  %d'%i
        print(tme,end="")
        if (i+sD)%7==0:
            print()
```



===结构化设计方法浅析===

【例4-4】已知1980年1月1日是星期二，现在要求根据用户输入的年份(≥1980)、月份在屏幕上打印出当月的日历，运行效果如下：

```
def gmn(m):
    mn={1:"January",2:"February",
        3:"March",4:"April",5:"May",
        6:"June",7:"Jnly",8:"August",
        9:"September",10:"October",
       11:"November",12:"Deceember"}
    return mn[m]
```

```
def gDm(m,y):
    lm1=[1,3,5,7,8,10,12]
    lm2=[4,6,9,11]
    if m in lm1:
        return 31
    if m in lm2:
        return 30
    if m==2:
        if ly(y):
            return 29
        else:
            return 28
    return 0
```

```
def gD(m,y):
    t=1
    for i in range(1980,y):
        if ly(i):
            t+=366
        else:
            t+=365
    for i in range(1,m):
        t+=gDm(i,y)
    return t%7
```

```
def ly(y):
    return (y%4==0 and y%100!=0) or y%400==0
```

➢ 函数式编程

【例4-5】以普通编程方式实现计算列表中正数之和

```
lt=[2,-4,9,-5,6,13,-12,-3]
s=0
for i in range(len(lt)):
    if lt[i]>0:
        s+=lt[i]
print("sum=",s)
```

```
from functools import *
lt=[2,-4,9,-5,6,13,-12,-3]
s=filter(lambda x:x>0,lt)
sum1=reduce(lambda x,y:x+y,s)
print("sum=",sum1)
```

【例4-6】以函数编程方式实现计算列表中正数之和

➢ 闭包

函数的嵌套定义。可以在函数内部定义一个嵌套函数，将嵌套函数视为一个对象，所以可以将嵌套函数作为定义它的函数的返回结果

【例4-7】使用闭包

```
def func_lib():
    def add(x,y):
        return x+y
    return add
fadd=func_lib()
print(fadd(1,2))
```

本章小结

- 函数的定义与调用
- 变量的作用域
- 递归函数
- 结构化设计浅析