Python及其应用

主讲人：钱惠敏
E-mail: amandaqian@hhu.edu.cn

第5讲 函数

5.1 函数的定义
5.2 函数的参数
5.3 变量作用域
5.4 函数的返回
5.5 递归函数
5.6 匿名函数

5.1 函数的定义

- 函数：可重复使用的，用来实现单一或相关联功能的代码段。

- 作用：能提高应用的模块性，和代码的重复利用率

（1）通过函数组织程序为一个个代码块，更有利阅读代码，更容易调试代码。

（2）可减少重复代码的使用，也使代码修改变得简单。

5.1 函数的定义（续1）

- 函数包括：
  - ✓ 自定义函数
  - ✓ 内建函数，如Python自带print(),int()，标准库中的函数（如math库中的sqrt()），其它功能库的方法（如PIL库中的image.open()）

5.1 函数的定义（续2）

```
def function_name（par1, par2, ...）:
    expression
    return expression
```

➢自定义函数的规则：

（1）函数代码块以def关键词开头，后接函数标识符名称和圆括号()。

（2）圆括号中间是传入参数，个数可以是0,1或多个。

（3）函数内容以冒号起始，并且缩进。

（4）函数体由一个或多个语句组成；

（5）return [表达式] 结束函数，选择性地返回一个值给调用方法。不带表达式的return相当于返回 None。

5.1 函数的定义（续3）

```
def function_name（par1, par2, …）:
    expression
    return expression
```

注：

（1）函数的名字给出规则同变量名；

（2）注意函数内的语句的缩进；保持函数体中同一层级代码的缩进一致；缩进结束的地方，表示函数结束；

（3）return语句是可选的，可以出现在函数体的任何一个位置；如果没有return语句，函数执行完毕后也会返回结果，只是结果为None。

5.2 函数的参数

- 形式参数（简称形参）：定义函数时，函数名后圆括号中的变量。形参只在函数内部有效。
- 实际参数（简称实参）：调用函数时，函数名后面圆括号中的变量。

```
#定义函数paramone()
def paramone(str):
        print('the param is:',str)
        print('我是一个传入参数，我的值是：',str)
#调用函数
paramone('hello,world’)
```

5.2 函数的参数（续1）

➢参数的类型：

（1）必须参数

（2）关键字参数

（3）默认参数

（4）不定长参数

（5）组合参数

5.2 函数的参数（续2）

- 必须参数：须以正确的顺序传入函数。调用时的数量必须和声明时的一样。

```
#定义函数paramone()
def paramone(str):
        print('the param is:',str)
        print('我是一个传入参数，我的值是：',str)
#调用函数
paramone('hello,world')
```

```
the param is: hello,world
我是一个传入参数，我的值是： hello,world
```

5.2 函数的参数（续3）

➢关键字参数：函

传入的参数值。

```
def personinfo(age,name)
    print('年龄：',age)
    print('名称：',name)
    return
print('-------按参数顺序传入参数-------')
personinfo(21,'小萌')
print('-------不按参数顺序传入参数，指定参数名-------')
personinfo(name='小萌',age=21)
print('-------按参数顺序传入参数，并指定参数名-------')
personinfo(age=21,name='小萌')
```

```
-------按参数顺序传入参数-------
年龄： 21
名称： 小萌
-------不按参数顺序传入参数，指定参数名-------
年龄： 21
名称： 小萌
-------按参数顺序传入参数，并指定参数名-------
年龄： 21
名称： 小萌
```

5.2 函数的参数（续4）

- 默认参数：在定义函数时，给参数一个默认值，当没有给调用该函数时的该参数赋值时，调用的函数就使用这个默认的值。

```
def defaultparam(name,age=23):
    print('hi,我叫:',name)
    print('我今年:',age)
    return
defaultparam('小萌')
```

```
hi,我叫: 小萌
我今年: 23
```

5.2 函数的参数（续5）

➢默认参数注意点：

（1）默认参数不能在必须参数之前。

（2）若不传入默认参数值，都会使用默认值。

（3）若只想更改某一个默认参数值，可以通过参数名来更改。

（4）若有一个默认参数是通过传入参数名更改参数值，则其他任何想要更改的默认参数都需要传入参数名来更改参数值。

（5）更改默认参数的值时，传入默认参数的顺序不需要根据定义的函数中的默认参数的顺序进行传入，不过最好同时传入参数名，否则容易出现执行结果与预期不一致的情况。

5.2 函数的参数（续6）

➢不定长参数：调用时能处理比函数声明时更多的参数，不定长参数声明时不会命名。

```
def functionname([formal_args,] *var_args_tuple ):
   "函数_文档字符串"
   function_suite
   return [expression]
```

注：加了星号（*）的变量名会存放所有未命名的变量参数。如果变量参数在函数调用时没有指定参数，它就是一个空元组。我们也可以不向可变函数传递未命名的变量。

```
def personinfo(arg,*vartuple):
    print(arg)
    for var in vartuple:
        print('我属于不定长参数部分:',var)
    return
print('------------不带不定长参数--')
personinfo('小萌')
print('------------带上两个参数----------------')
personinfo('小萌',21,'beijing')
print('------------带上五个参数---------------')
personinfo('小萌',21,'beijing',123,'shanghai','happy')
```

```
------------不带可变参数----------
小萌
------------带上两个可变参数----------
小萌
我属于不定长参数部分: 21
我属于不定长参数部分: beijing
------------带上五个可变参数----------
小萌
我属于不定长参数部分: 21
我属于不定长参数部分: beijing
我属于不定长参数部分: 123
我属于不定长参数部分: shanghai
我属于不定长参数部分: happy
```

5.2 函数的参数（续7）

➢组合参数：组合使用必须参数、关键字参数、默认参数和可变关键字参数。但参数定义的顺序必须是：必须参数、默认参数、不定长参数和关键字参数。

元组

字典

```
def exp(p1, p2, df=0, *vart, **kw):
    print('p1 =', p1, 'p2=', p2, 'df=', df, 'vart=', vart, 'kw =', kw)
exp(1,2)
exp(1,2,c=3)
exp(1,2,3,'a','b')
exp(1,2,3,'abc‘,’bcd’,x=9)
```

```
p1 = 1 p2= 2 df= 0 vart= () kw = {}
p1 = 1 p2= 2 df= 0 vart= () kw = {'c': 3}
p1 = 1 p2= 2 df= 3 vart= ('a', 'b') kw = {'x': 9}
p1 = 1 p2= 2 df= 3 vart= ('abc', 'bcd', 'cf') kw = {'x': 9}
```

5.3 变量作用域

- 作用域简单说就是一个变量的命名空间，即其访问权限。变量的作用域决定了哪一部分程序可以访问哪个特定的变量名称。

- 有两种最基本的变量作用域：
  （1）局部变量
  （2）全局变量

5.3 变量作用域（续1）

- 局部变量：在函数内定义的变量名，只能被函数内部引用，不能在函数外引用这个变量名，这个变量的作用域就是局部的。

```
def func():
    x = 100
    print x
```

局部变量只能在函数体中被访问，超出函数体的范围，访问报错。

5.3 变量作用域（续2）

- 全局变量：在函数外，一段代码最开始所赋值的变量，它可以被多个函数引用。全局变量可以在整个程序范围内访问。

```
total = 0; # 这是一个全局变量
def sum( arg1, arg2 ):
    total = arg1 + arg2; # total在这里是局部变量.
    print ("函数内是局部变量 : ", total)
    return total
def totalprint():
    print('total的值是',total)
    return total
```

5.4 函数的返回

- 若定义函数时没有使用return语句，默认返回None；若需返回具体的数值，使用return后面加上需要返回的内容；只写return返回None。
- 会产生并返回结果的函数，称为有返回值函数，如数学函数；只执行一些动作，但不返回任何值的函数为无返回值函数。
- 当调用有返回值函数时，可以使用返回的结果做相关操作，而使用无返回值或返回None的函数时，就只能得到一个None值。

5.4 函数的返回（续1）

函数可以有返回值，但除了返回值，函数中是否可以返回函数？

```
def calc_sum(*args):
    ax = 0
    for n in args:
        ax = ax + n
    return ax
```

返回值

```
def sum_late(*args):
    def calc_sum():
        ax = 0
        for n in args:
            ax = ax + n
        return ax
    return calc_sum
```

闭包

返回函数

5.5 递归函数

- 递归函数：一个函数在内部调用自身
- 递归函数特性：

（1）必须有一个明确的结束条件；

（2）每次进入更深一层递归时，问题规模相比上次递归都应有所减少

（3）相邻两次重复之间有紧密的联系，前一次要为后一次做准备（通常前一次的输出就作为后一次的输入）

（4）递归效率不高，递归层次过多会导致栈溢出

例：计算阶乘n! = 1×2 × 3 × ... × n

fact(n)用递归的方式的定义函数如下：

```
#! /usr/bin/python3
# -*- coding:UTF-8 -*-
 def fact(n):
    if n==1:
        return 1
    return n * fact(n - 1)
```

5.6 匿名函数

➢匿名函数：使用lambda来创建。

lambda [arg1 [,arg2,.....argn]]:expression

注：lambda的主体只是一个表达式，而不是一个代码块，其函数体比def简单很多。

```
def func(x,y,z):
    return 1*x+2*y**2+3*z**4
```

```
lambda x,y,z: 1*x+2*y**2+3*z**4
```

```
polynomial=lambda x,y,z: 1*x+2*y**2+3*z**4
```

➢一般应该在如下情况下多往匿名函数的方向考虑：

（1）程序一次性使用，不需要定义函数名时，用匿名函数可以节省内存中变量定义空间。

（2）如果想让程序更加简洁时，使用匿名函数可以做到。

案例讲解：定义函数1

1. 定义一个函数，输入不定个数的数字，返回所有数字的和

```
#求输入数字的和
def sumofdigits(*args):
    x=0
    for i in args:
        x+=i
    return x

print (sumofdigits(1,2,3,4,5,6,7,8,9,10))
```

案例讲解：定义函数2

2. 定义一个函数quadratic(a, b, c)，接收3个参数，返回一元二次方程： $ax^2 + bx + c = 0$ 的两个实解。

```
import math
def quadratic(a, b, c):
    if not isinstance(a, (int, float)):
        raise TypeError('a is not a number')
    if not isinstance(b, (int, float)):
        raise TypeErrot('b is not a number')
    if not isinstance(c, (int, float)):
        raise TypeError('c is not a number')
    derta = b * b - 4 * a * c
    if a == 0:
        if b == 0:
            if c == 0:
                return '方程根是全体实数'
            else:
                return '方程无根'
        else:
            x = -c / b
            print('方程只有一个实根')
            return x
    else:
        if derta < 0:
            return '方程无实根'
        elif derta=0:
            x=-b/(2*a)
            print('方程只有一个实根')
            return x
        else:
            x1 = (-b + math.sqrt(derta)) / (2 * a)
            x2 = (-b - math.sqrt(derta)) / (2 * a)
            return x1, x2

print(quadratic(2, 3, 1))
print(quadratic(1, 3, -4))
```