

Python语言基础与应用

计算和控制流 / 代码组织：函数(def)

陈斌 北京大学 gischen@pku.edu.cn



代码组织：函数(def)

- › 封装一个功能
- › 定义与调用函数
- › 变量的作用域
- › 函数小技巧

封装一个功能

› 封装

容器是对数据的封装

函数是对语句的封装

类是对方法和属性的封装

› 函数(function)

程序中实现明确功能的代码段可以封装成一个函数，以便复用 (reuse)

定义与调用函数

› 定义函数

用def语句创建一个函数

用return关键字指定函数返回的值

```
def <函数名> (<参数表>):  
    <缩进的代码段>  
    return <函数返回值>
```

› 调用函数

<函数名>(<参数>)

注意括号！

无返回值：<函数名>(<参数表>)

返回值赋值：v = <函数名>(<参数表>)

定义与调用函数

```
1  def sum_list(alist):  # 定义一个带参数的函数
2      sum_temp = 0
3      for i in alist:
4          sum_temp += i
5      return sum_temp  # 函数返回值
6
7
8  print(sum_list)  # 查看函数对象sum_list
9
10 my_list = [23, 45, 67, 89, 100]
11 # 调用函数, 将返回值赋值给my_sum
12 my_sum = sum_list(my_list)
13 print("sum of my list:%d" % (my_sum,))
```

→ <function sum_list at 0x10067a620>
sum of my list:324

变量的作用域

› 局部变量(Local Variable)

在函数内部定义的参数以及变量

只在该函数定义范围内有效，函数外边无法访问到

› 全局变量(Global Variable)

在函数外部定义的，作用域是整个代码段

```
>>> def addNum(num1, num2):  
    result = num1 + num2  
    return result  
  
>>> num1 = 5  
>>> num2 = 10  
>>> addNum(num1, num2)  
15  
>>> print(result)  
Traceback (most recent call last):  
  File "<pyshell#33>", line 1, in <module>  
    print(result)  
NameError: name 'result' is not defined
```

变量的作用域

› global关键字

可以在一个函数内部得到某个全局变量的值，但是无法进行修改，Python会在函数内部创建一个同名的局部变量

使用global关键字可以在函数中改变全局变量的值

```
>>> def addNum():  
    num1, num2 = 2, 3  
    return(num1 + num2)  
  
>>> addNum()  
5  
>>> num1, num2  
(1, 2)
```

```
>>> def addNum():  
    global num1, num2  
    num1, num2 = 2, 3  
    return(num1 + num2)  
  
>>> addNum()  
5  
>>> num1, num2  
(2, 3)
```

函数小技巧: map()函数

- › 有时需要对列表中每个元素做一个相同的处理, 得到新列表

例如: 所有数据乘以3

所有字符串转换为整数

两个列表对应值相加

- › **map(func, list1, list2....)**

函数func有几个参数, 后面跟几个列表

函数小技巧: map()函数

```
num = [10, 20, 40, 80, 160]
lst = [2, 4, 6, 8, 10]
def mul3(a):
    return a * 3

print (list( map(mul3, num) ))

def atob(a, b):
    return a + 1.0/b

print (list( map(atob, num, lst) ))
```

```
[30, 60, 120, 240, 480]
[10.5, 20.25, 40.166666666666664, 80.125, 160.1]
```

函数小技巧：匿名函数lambda

- › 有时函数只用一次，其名称也就不重要，可以无需费神去def一个
- › lambda表达式

返回一个匿名函数

lambda <参数表>:<表达式>

```
num = [10, 20, 40, 80, 160]
lst = [2, 4, 6, 8, 10]
def mul3(a):
    return a * 3

print (list( map(mul3, num) ))

def atob(a, b):
    return a + 1.0/b

print (list( map(atob, num, lst) ))

print (list( map(lambda a:a * 3, num)))
print (list( map(lambda a,b:a+1.0/b, num, lst)))
```