



4.2 函数和模块



4.2.1 函数

■ 函数 (function)

- 实现某种**特定功能**的代码块
- 程序简洁，可重复调用、封装性好、便于共享
- 系统函数和用户自定义函数

Python3.6.2版本，一共提供了**68个内置函数**。



■ Python内置函数

- 数学运算函数
- 输入输出函数
- 类型转换函数
- 逻辑判断函数
- 序列操作函数
- 对象操作函数



□ 数学运算函数

函数	原型	具体说明
abs()	abs(x)	返回x的 绝对值
pow()	pow(x,y)	返回x的y次 幂
round()	round(x[,n])	返回浮点数x的 四舍五入 值，参数n指定保留的小数位数，默认为0
divmod()	divmod(a,b)	返回a除以b的 商 和 余数 ，返回一个元组。divmod(a,b)返回(a/b, a%b)

```
>>>abs(-1)
1
>>>pow(2,3)
8
>>>round(3.1415,2)
3.14
>>>round(3.54)
4
>>>divmod(5,3)
(1,2)
```



□ 常用Python内置函数

函 数	描 述	函 数	描 述
len()	返回长度	list()	转换为列表
max()	返回最大值	help()	显示帮助信息
sum()	返回总和	dir()	显示属性
str()	转换成字符串	type()	显示类型
float()	转换为浮点型	range()	返回一个整型列表
int()	转换为整型表示	open()	打开文件



■ 用户自定义函数

□ 定义函数

形式参数 / 形参

```
def 函数名 (参数列表):  
    函数体
```

- **形式参数**在定义时并没有确定的值，只是用来说明函数的功能。在程序执行过程中，当函数被调用时，形参才得到具体的值，并参与运算，求得函数值
- **参数列表**可以为空，即函数可以没有参数；
- 如果函数中包含**多个参数**，参数之间用**逗号**分隔。
- **函数体**可以是一条语句，也可以是一个语句块。



例：自定义函数

□ 一个返回值

```
def add(a,b):  
    c=a+b  
    return c
```

□ 多个返回值

```
def add_mul(a,b):  
    add=a+b  
    mul=a*b  
    return add,mul
```

□ 无返回值

```
def say_hello(your_name):  
    print("Hello,%s!" %your_name)
```



□ 调用函数

实际参数 / 实参

函数名 (参数列表)

定义函数:

```
def add(a,b):  
    c=a+b  
    return c
```

调用函数:

```
add(1,2)
```

运行结果: 3



□ 通过**多元赋值语句**，同时获取多个返回值多个返回值

```
def add_mul(a,b):                #定义函数
    add=a+b
    mul=a*b
    return add,mul

x,y=add_mul(1,2)                #调用函数
print("add:",x,";mul:",y)
```

运行结果：

```
add: 3 ;mul: 2
```



□ 无形式参数

```
def say_hello():          #定义函数
    print("Hello! ")

say_hello()               #调用函数
```

调用函数时，函数名后面必须有小括号，即使没有参数，这个小括号也不能省略。

运行结果：

```
Hello!
```



□ 变量的作用域

- **局部变量** (Local Variable) : **在函数中定义**的变量, 仅在定义它的**函数内部有效**。
- **全局变量** (Global Variable) : **在函数体之外定义**的变量, 在**定义后的代码中都有效**, 包括在它之后定义的函数体内。



例：局部变量

```
def setNumber():  
    a= 9  
    a=a+1  
    print("setNumber:", a)
```

```
setNumber()
```

运行结果：

```
setNumber: 10
```

定义局部变量的函数中，**只有局部变量是有效的**

```
>>>print(a)  
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-9-5315f3e3adca> in <module>()  
----> 1 print(a)  
NameError: name 'a' is not defined
```



例：全局变量&局部变量

```
a=100                                #定义全局变量
def setNumber():                     #定义函数
    a= 9                             #定义局部变量
    a=a+1                             #打印局部变量
    print("setNumber:", a)

setNumber()                          #调用函数，打印局部变量
print(a)                             #打印全局变量
```

运行结果：

```
setNumber: 10
100
```

- 在函数外部定义的变量a是**全局变量**，当它与函数内部定义的局部变量同名时，在函数内部失效。
- 在函数内定义的变量a是**局部变量**，它只在函数体内局部有效，并不影响全局变量a的取值。



□ 函数的参数——参数的传递

- **按值传递**：形参和实参**分别存储，相互独立**。
- 在内部函数改变形参的值时，实参的值不会随之改变。

```
def func(num):           #定义函数
    num+=1               #函数内部改变形参的值
    print("num:",num)    #打印形参

a=10
func(a)                  #调用函数，a为实参
print("a:",a)            #输出变量a
```

运行结果：

```
num:11
a:10
```



□ 函数的参数——参数的默认值

```
def 函数名 (参数1=默认值, 参数2=默认值...):  
    函数体
```

```
def add(a,b=2):  
    return a+b
```

```
print(add(1))  
print(add(2,3))
```

运行结果:

```
3  
5
```

```
def add (a, b=1, c=2):  
    return a+b+c
```

当函数中有多个参数时, 参数的默认值只能**从最右面开始依次设置**



□ 函数的参数——向函数内部批量传递数据

- 可以使用**列表**、**字典**变量作为参数，向函数内部批量传递数据。
- 当使用列表或字典作为函数参数时，在函数内部对列表或字典的元素所做的修改，**会改变实参的值**。

```
def sum(list):  
    sum=0  
    for x in list:  
        sum+=x  
    return sum  
  
lst_1=[1,2,3,4,5]  
print(sum(lst_1))
```

运行结果：15

