02: 函数

2.1分类

函数分为内置函数和自定义函数。

函数定义:

def 函数名(参数列表)：

函数体

2.2 函数参数

2.2.1 参数种类

函数参数分为可变类型和不可变类型，其调用结果是不同的。

1. 可变类型：类似c++的引用传递，如列表、字典等。如果传递的参数是可变类型，则在函数内部对传入参数的修改会影响到外部变量。

例: def change\_int(my\_list):

"修改传入的列表"

my\_list.append([1, 2, 3])

print("函数内修改后的变量：", my\_list)

my\_list = [10, 20, 30]

change\_int(my\_list)

print("函数外变量的值：", my\_list)

在调用函数时，如果传入的参数是可变类型，则外部变量也会被更改。在上述例子中，传入函数的list对象和在末尾添加新内容的mylist对象用的是同一个引用。

1. 不可变类型：类似c++的值传递，如整数、字符串、元组等。如果传递的参数是不可变类型，则在函数内部对传入参数的修改不会影响到外部变量。

例: def change\_int(a):

a = 10

b = 2

change\_int(b)

print(b) # 结果是2

上述实例中，有int类型的对象2，指向它的变量是b。在传递给change\_int（）函数时，按传值方式复制了变量b，a和b都指向了同一个int对象。在a=10时，则新生成一个int值对象10，并让a指向它。

2.2.2 位置参数

调用函数时，Python语言必须将函数调用中的每个实参都关联到函数的相应形参。最简单的关联方式是基于实参的顺序，这种关联方式被称为位置实参。

2.2.3 默认参数

编写函数时，可给每个形参指定默认值。在调用函数时，如果给形参提供了实参，Python语言将使用指定的实参值；否则，将使用形参的默认值。给形参指定默认值后，可在函数调用中省略相应的实参。

2.2.4 不定长参数

在Python语言中，函数还可以定义不定长参数，也叫可变参数。

2.2.5 关键字参数

关键字实参是传递参数时使用“名称–值”对的方式，在实参中将名称和值关联起来，清楚地指出了函数调用中各个值的用途。

关键字参数有扩展函数的功能。

2.2.6参数组合

在Python语言中定义函数，可以组合使用这些参数（必选参数、默认参数、可变参数、关键字参数和命名关键字参数）。参数定义是有顺序的。定义的顺序必须是：必选参数、默认参数、可变参数、命名关键字参数和关键字参数。

2.2.7 参数组合

在Python语言中定义函数，开发者可以组合使用这些参数（必选参数、默认参数、可变参数、关键字参数和命名关键字参数）。注意参数定义是有顺序的。定义的顺序必须是：必选参数、默认参数、可变参数、命名关键字参数和关键字参数

2.4 函数式编程

函数式编程是一种编程范式，是面向数学的抽象，其将计算描述为一种表达式求值。

函数式编程中的“函数”不是指计算机中的函数，而是指数学中的函数，即自变量的映射。

函数式编程允许把函数本身作为参数传入另一个函数，还允许返回一个函数。

Python语言对函数式编程提供部分支持。由于允许使用变量，所以说Python语言不是纯函数式编程语言。

2.4.1 高阶函数

接受函数为参数，或者把函数作为结果返回的函数称为高阶函数。例

若要根据单词的长度排序，只需把len函数传给key参数。

fruits=['strawberry','fig','apple','cherry','raspberry','banana']

print(sorted(fruits,key=len))

# 输出结果如下：

# ['fig', 'apple', 'cherry', 'banana', 'raspberry', 'strawberry']

fruits = ['strawberry', 'fig', 'apple', 'cherry', 'raspberry', 'banana']

def reverse(word):

return word[::-1]

print(reverse('testing')) # 结果是：gnitset

print(sorted(fruits, key=reverse))

# 输出结果如下：

# ['banana', 'apple', 'fig', 'raspberry', 'strawberry', 'cherry']

2.4.2 匿名函数

所谓匿名函数，即不再使用def语句这样标准形式定义的函数。Python语言经常使用lambda来创建匿名函数。lambda 只是一个表达式，函数体比def定义的函数体要简捷。lambda函数的语法如下所示。

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

举例：

sum = lambda arg1, arg2: arg1 + arg2

print(sum(1, 2)) # 结果是：3

上述代码中，第一行定义了一个lambda函数，执行两个数的和运算，并且把该lambda函数命名为sum。会面的代码通过sum（）函数即实现了调用lambda函数的功能。