链滴

python 学习 --- 函数（二）

作者：oldcaptain

原文链接：https://ld246.com/article/1374321788217

来源网站：链滴

<h1> <a>简介 </h1>
 函数是为了达到代码重用，而封装的一种逻辑处理块。在高级的编程语言中都有这样的功能，在ython中也不例外。
 函数通过def关键字定义。def关键字后跟一个函数的 标识符 名称，然后跟一对圆括号。圆括号之中可以包括一些变量名，该行以冒号结尾。接下来是一块语句，们是函数体。
 1、定义函数
<pre>>>> def sayHello():
    print 'hello world'

>>> sayHello()
hello world</pre>

 2、函数形参
<pre>>>> def printMax(a,b):
    if a>b:
        print a,"is max"
    else:
        print b,"is max"

>>> printMax(1000,89)
1000 is max</pre>

 形参为a，b，实参则为实际传入的参数1000，89
 3、局部变量
 当你在函数定义声明变量的时候，它们与函数外具有相同名称的其他变量没有任何关系，即变量名称对于函数来说是nbsp;局部 的。这称为变量的 作用域 。所有变量的作用域是它们被定义的块，它们的名称被定义的那点开始
<pre>>>> def func(x):
    print 'x is ',x
    x =2
    print 'change local x to ',x

>>> x =50
>>> func(x)
x is 50
change local x to 2
>>> print 'x is still',x
x is still 50
>>>   </pre>

 4、gobal（全局）变量
 如果你想要为一个定义在函数外的变量赋值，那么你就得告诉Python这个变量名不是局部的，是 全局 的。我们使用global语句完成这一功能。没有global语句，是不可能为定义在数外的变量赋值的。
 你可以使用定义在函数外的变量的值（假设在函数内没有同名的变量）。然而，我并不鼓励你样做，并且你应该尽量避免这样做，因为这使得程序的读者会不清楚这个变量是在哪里定义的。使用gobal语句可以清楚地表明变量是在外面的块定义的。
<pre>>>> def func():
    global x

```
    print 'x is ',x
    x = 2
    print 'changed local x to ',x

&gt;&gt;&gt; x =50
&gt;&gt;&gt; func()
x is  50
changed local x to  2
&gt;&gt;&gt; print 'value of x is ',x
value of x is  2
&gt;&gt;&gt; x =40
&gt;&gt;&gt; print x
40
&gt;&gt;&gt;                                    </pre>

<p> <br /> 5、默认参数值<br /> 对于一些函数，你希望他的参数是可选的，如果用户不想为这些数赋值的话，那么这些参数将使用默认值。这个功能借助于默认参数值。你可以在函数定义的形参名加上赋值运算符（=）和默认值，从而给函数指定默认参数值。<br /> 注意，<a></a>默认参数值该是一个参数。更加准确的说，<a></a>默认参数值应该是不可变的——这会在后面的章节中做详解释。从现在开始，请记住这一点<br />  </p>
<pre>&gt;&gt;&gt; def say(message,times =1):
    print message * times

&gt;&gt;&gt; say('hello')
hello
&gt;&gt;&gt; say('hello',5)
hellohellohellohellohello
&gt;&gt;&gt;                    </pre>

<p> 重要：  </p>
<p> 只有在形参表末尾的那些参数可以有<a></a>默认参数值，即你不能在声明函数形参的时候，声明有默认值的形参而后声明没有默认值的形参。<br /> 这是因为赋给形参的值是根据位置而赋值。例如，def func(a, b=5)是有效的，但是def func(a=5, b)是&amp;nbsp;无效&amp;nbsp;的。 </p>
<p> 6、关键参数 </p>
<p> 如果你的某个函数有许多参数，而你只想指定其中的一部分，那么你可以通过命名来为这些参赋值——这被称作&amp;nbsp;<a></a>关键参数&amp;nbsp;——我们使用名字（关键字）而不是位置（我前面所一直使用的方法）来给函数指定实参。 </p>
<p> 这样做有两个&amp;nbsp;优势&amp;nbsp;——一，由于我们不必担心参数的顺序，使用函数变得更加单了。二、假设其他参数都有默认值，我们可以只给我们想要的那些参数赋值。 </p>
<pre>&gt;&gt;&gt; def func(a,b=5,c=10):
    print 'a is ',a,' and b is ',b,' and c is',c

&gt;&gt;&gt; func(3,7)
a is  3  and b is  7  and c is 10
&gt;&gt;&gt; func(25,c=24)
```

a is 25 and b is 5 and c is 24

>>> func(c=80,a=100)

a is 100 and b is 5 and c is 80

>>> </pre>

 7、return语句
 return语句用来从一个函数 返回 即跳出函数。我们也可选从函数 返回一值 。

<pre>>>> def maximum(x,y):
    if(x >y):
        return x
    else:
        return y

>>> print maximum(2,3)

3

>>> </pre>

 注意，没有返回值的return语句等价于return None。None是Python中表示没有任何东西的特类型。例如，如果一个变量的值为None，可以表示它没有值。

 除非你提供你自己的return语句，每个函数都在结尾暗含有return None语句。
 8、DocStrings
 Python有一个很奇妙的特性，称为 文档字符串 ，它通常被简称为 docstrigs 。<a>DocStrings是一个重要的工具，由于它帮助你的程序文档更加简单易懂，你应尽量使用它。你甚至可以在程序运行的时候，从函数恢复文档字符串！

<pre>>>> def printMax(x, y):
    '''Prints the maximum of two numbers.

The two values must be integers.'''
x = int(x) # convert to integers, if possible
y = int(y)

if x > y:
    print x, 'is maximum'
else:
    print y, 'is maximum'

>>> printMax(3,5)

5 is maximum

>>> print printMax. **doc**

Prints the maximum of two numbers.

The two values must be integers.

>>> help(printMax) </pre>

 在函数的第一个逻辑行的字符串是这个函数的 文档字符串 。注意，<a>Doc trings也适用于模块和类，我们会在后面相应的章节学习 们。
 文档字符串的惯例是一个多行字符串，它的首行以大写字母开始，句号结尾。第二行是空行， 第三行开始是详细的描述。 强烈建议 你在你的函数中使用文档字符串时遵循这个惯例。
 你可以使用__doc__（注意双下划线）调用printMax函数的文档字符串属性（属于函数的名称） 请记住Python把 每一样东西 都作为对象，包括这个函数。我们会在后面的类一章学习更多关于对象的知识。
 如果你已经在Python中使用过help()，那么你已经看到过DocStings的使用了！它所做的只是 取函数的__doc__属性，然后整洁地展示给你。你可以对上面这个函数尝试一下——只是在你的程序 包括help(printMax)。记住按q退出help。
 自动化工具也可以以同样的方式从你的程序中提取文档。因此，我 强烈建议 你对 所写的任何正式函数编写文档字符串。随你的Python发行版附带的pydoc命令 与help()类似地使用<a>DocStrings。
