1. **Python函数**

**第一部分**

**内容的引入：**

比如第一次计算sum=1+2+....+100

编写源代码一次

第二次又计算sum=1+2+....+100

编写源代码一次

比如第一次计算sum=1+2+....+100

编写源代码一次

第三次又计算sum=1+2+....+100

编写源代码一次

......

第100次，人崩溃

......

想办法，能不能只编写一次，调用100次了？

**一、函数的定义**

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。函数能提高应用的模块性，和代码的重复利用率。

**二、函数的分类**

1、系统函数，Python提供了许多内建函数。在python的命令行，输入dir(\_\_builtins\_\_)即可得到所有的内置函数。我们只管调用即可，比如print(),len(),next(),iter()等等。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [abs()](http://www.runoob.com/python3/python3-func-number-abs.html) | [dict()](http://www.runoob.com/python/python-func-dict.html) | [help()](http://www.runoob.com/python/python-func-help.html) | [min()](http://www.runoob.com/python3/python3-func-number-min.html) | [setattr()](http://www.runoob.com/python/python-func-setattr.html) |
| [all()](http://www.runoob.com/python/python-func-all.html) | [dir()](http://www.runoob.com/python/python-func-dir.html) | [hex()](http://www.runoob.com/python/python-func-hex.html) | [next()](http://www.runoob.com/python/python-func-next.html) | [slice()](http://www.runoob.com/python/python-func-slice.html) |
| [any()](http://www.runoob.com/python/python-func-any.html) | [divmod()](http://www.runoob.com/python/python-func-divmod.html) | [id()](http://www.runoob.com/python/python-func-id.html) | object() | [sorted()](http://www.runoob.com/python/python-func-sorted.html) |
| [ascii()](http://www.runoob.com/python3/python3-func-ascii.html) | [enumerate()](http://www.runoob.com/python3/python3-func-enumerate.html) | [input()](http://www.runoob.com/python/python-func-input.html) | [oct()](http://www.runoob.com/python/python-func-oct.html) | [staticmethod()](http://www.runoob.com/python/python-func-staticmethod.html) |
| [bin()](http://www.runoob.com/python/python-func-bin.html) | [eval()](http://www.runoob.com/python/python-func-eval.html) | [int()](http://www.runoob.com/python/python-func-int.html) | [open()](http://www.runoob.com/python/python-func-open.html) | [str()](http://www.runoob.com/python/python-func-str.html) |
| [bool()](http://www.runoob.com/python/python-func-bool.html) | [exec()](http://www.runoob.com/python3/python3-func-exec.html) | [isinstance()](http://www.runoob.com/python/python-func-isinstance.html) | [ord()](http://www.runoob.com/python/python-func-ord.html) | [sum()](http://www.runoob.com/python/python-func-sum.html) |
| [bytearray()](http://www.runoob.com/python/python-func-bytearray.html) | [filter()](http://www.runoob.com/python/python-func-filter.html) | [issubclass()](http://www.runoob.com/python/python-func-issubclass.html) | [pow()](http://www.runoob.com/python3/python3-func-number-pow.html) | [super()](http://www.runoob.com/python/python-func-super.html) |
| [bytes()](http://www.runoob.com/python3/python3-func-bytes.html) | [float()](http://www.runoob.com/python/python-func-float.html) | [iter()](http://www.runoob.com/python/python-func-iter.html) | [print()](http://www.runoob.com/python/python-func-print.html) | [tuple()](http://www.runoob.com/python3/python3-func-tuple.html) |
| [callable()](http://www.runoob.com/python/python-func-callable.html) | [format()](http://www.runoob.com/python/att-string-format.html) | [len()](http://www.runoob.com/python3/python3-string-len.html) | [property()](http://www.runoob.com/python/python-func-property.html) | [type()](http://www.runoob.com/python/python-func-type.html) |
| [chr()](http://www.runoob.com/python/python-func-chr.html) | [frozenset()](http://www.runoob.com/python/python-func-frozenset.html) | [list()](http://www.runoob.com/python3/python3-att-list-list.html) | [range()](http://www.runoob.com/python/python-func-range.html) | [vars()](http://www.runoob.com/python/python-func-vars.html) |
| [classmethod()](http://www.runoob.com/python/python-func-classmethod.html) | [getattr()](http://www.runoob.com/python/python-func-getattr.html) | [locals()](http://www.runoob.com/python/python-func-locals.html) | [repr()](http://www.runoob.com/python/python-func-repr.html) | [zip()](http://www.runoob.com/python/python-func-zip.html) |
| [compile()](http://www.runoob.com/python/python-func-compile.html) | [globals()](http://www.runoob.com/python/python-func-globals.html) | [map()](http://www.runoob.com/python/python-func-map.html) | [reversed()](http://www.runoob.com/python3/python3-func-reversed.html) | [\_\_import\_\_()](http://www.runoob.com/python/python-func-__import__.html) |
| [complex()](http://www.runoob.com/python/python-func-complex.html) | [hasattr()](http://www.runoob.com/python/python-func-hasattr.html) | [max()](http://www.runoob.com/python3/python3-func-number-max.html) | [round()](http://www.runoob.com/python3/python3-func-number-round.html) |  |
| [delattr()](http://www.runoob.com/python/python-func-delattr.html) | [hash()](http://www.runoob.com/python/python-func-hash.html) | [memoryview()](http://www.runoob.com/python/python-func-memoryview.html) | [set()](http://www.runoob.com/python/python-func-set.html) |  |

1. 用户自定义函数，用户用来完成某一个特定的功能的代码。

**三、使用自定义函数的步骤**

1、先定义一个函数

2、再调用函数

**四、自定义函数的语法**

Python 定义函数使用 def 关键字，一般格式如下：

def 函数名（参数列表）:

函数体

需要注意几点：

1、函数代码块以 **def** 关键词开头，后接函数名称和圆括号 **()**。

2、函数名称要符合Python的命名规范

3、任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义多个参数，每个参数用“，”隔开。

5、函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。

6、函数内容以冒号起始，并且缩进。

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

8、写在return语句下的语句是不会执行的，是"死"代码。

开发实例：定义一个函数sum(n),其功能是计算1+2+3+....+n的和。

#定义一个函数，函数的名称是sum,参数是n

def sum(n):

#函数体，用来处理具体的业务逻辑

a=0;

for t in range(n+1):

a+=t

#函数的返回值

return a

**四、调用函数的语法**

函数名(参数值）

比如下例：

#调用函数

print(sum(100))

**第二部分**

**五、参数的分类**

5.1 形式参数和实际参数

比如上面的sum(n）函数，n其实就是一个参数，而且叫做形式参数，因为它不表示一个具体的值，形式参数往往存在定义一个函数的时候，也就是说，定义函数时，函数圆括号中的参数都是形式参数。

如果在调用一个函数是，比如sum(100)，其中的100代表一个具体的数值，直接赋给形式参数，让它去执行函数体内的代码，这个数值就叫实际参数。

5.2 第二种分类方法

以下是调用函数时可使用的参数类型：

1、必需参数

2、 关键字参数

3、 默认参数

4、不定长参数

1、必需参数：必需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。

比如上面的sum(n)，其中n就是一个必须参数，这个参数必须传递，否则报错。再比如如下函数

def Person(name,age):

return "你的名字是%s,年龄是%d" %(name,age)

在调用函数Person时，print(Person("afeng", 23).decode("gb2312"))，参数的顺序是不可以改变的，如果改变了，函数的输出的内容也就改变了。

2、关键字参数

关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值。使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为 Python 解释器能够用参数名匹配参数值。

print(Person(age=12, name="afeng").decode("gb2312"))

3、默认参数

调用函数时，如果没有传递参数，则会使用默认参数。以下实例中如果没有传入 age和name 参数，则使用默认值：

def Person(name="afeng",age="35"):

return "你的名字是%s,年龄是%d" %(name,age)

print(Person().decode("gb2312"))

print(Person(age=45).decode("gb2312"))

print(Person(name="czf").decode("gb2312"))

执行结果：

你的名字是afeng,年龄是35

你的名字是afeng,年龄是45

你的名字是czf,年龄是35

4、不定长参数

你可能需要一个函数能处理比当初声明时更多的参数。这些参数叫做不定长参数，和上述2种参数不同，声明时不会命名。基本语法如下：

def function\_name([formal\_args,] \*var\_args\_tuple )，加了星号（\*）的变量名会存放所有未命名的变量参数。如果在函数调用时没有指定参数，它就是一个空元组。我们也可以不向函数传递未命名的变量。

声明一个带有不定长参数的函数：

def Person(name="afeng",age=35,\*v):

for var2 in v:

print (var2)

return "你的名字是%s,年龄是%d" %(name,age)

调用不定长参数的函数

print(Person("czf",45,67,45,78,90).decode("gb2312"))

输出的结果：

67

45

78

90

你的名字是czf,年龄是45

**第三部分**

1. **匿名函数：lambda ['læmdə]**

**6.1、lambda是什么？**

　　　　看个例子：

g = lambda x:x+1

　　看一下执行的结果：

　　g(1)

　　>>>2

　　g(2)

>>>3

>>> type(g)

<class 'function'>

>>>

　　可以这样认为,lambda作为一个表达式，它就是一个函数，且是匿名函数，所谓匿名，意即不再使用 def 语句这样标准的形式定义一个函数，上例的代码x为入口参数，x+1为函数体，用函数来表示为：

def g(x):  
 return x+1

　　非常容易理解，在这里lambda简化了函数定义的书写形式。是代码更为简洁，但是使用函数的定义方式更为直观，易理解。

**6.2 语法**

python 使用 lambda 来创建匿名函数。

语法

lambda 函数的语法只包含一个语句，如下：

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

lambda语句中，冒号前是参数，可以有多个，用逗号分割；冒号右边是返回值。

sum = lambda arg1, arg2,arg3: arg1 + arg2 + arg3;

print("sum=", sum( 10,20,56 ))

lambda 匿名函数也是可以使用"关键字参数"进行参数传递

g= lambda x,y : x\*\*2+y\*\*2

g(y=3,x=2)

13

同样地,lambda 匿名函数也可以设定默认值

g= lambda x=0,y=0 : x\*\*2+y\*\*2

>>> g(2,3) 13

>>> g(2) 4

>>> g(y=3 )9

注意：如果只打算给其中一部分参数设定默认值，那么应当将其放在靠后的位置（和定义函数时一样，避免歧义），否则会报错。

lambda 来创建匿名函数,经常用在全局函数filter, map.

因为在python3里面，map(),filter()这些的返回值已经不再是list,而是iterators, 所以想要使用，只用将iterator 转换成list 即可， 比如 list(map())

1、比如把一个列表中的每个元素的值乘以2再加上1，生成一个新的列表

foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]  
*#方法1：定义一个函数***def** f(x):  
 **return** 2\*x+1  
print(list(map(f,foo)))  
*#方法2：使用lambda函数*print(list(map(**lambda** x: x\*2+1, foo)))  
*#方法3：使用for语句*print([x \* 2 + 1 **for** x **in** foo])

2、过滤：只留下一个列表中元素的值能被3整除的

foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]  
*#方法1：定义一个函数***def** f(x):  
 **if** x%3==0:  
 **return** x  
print(list(filter(f,foo)))  
*#方法2：使用lambda函数*print(list(filter(**lambda** x:x%3==0, foo)))

*#方法3：使用for语句结合if语句*print([x **for** x **in** foo **if** x % 3 == 0])

3、提供了两个列表，对相同位置的列表数据进行相加

list1=[12,34,56,78]  
list2=[34,56,89,90]  
print(list(map(**lambda** x,y:x+y,list1,list2)))

**6.3 应用**

要实现的功能：在python3.x中读取某个目录下的文件，并取出不带后缀的文件名。

*# 方法用于返回指定的文件夹包含的文件或文件夹的名字的列表*flist=os.listdir(**r'Q:\temp'**)  
print(flist)  
*#os.path.splitext：分离文件名与扩展名；  
# 默认返回(fname,fextension)元组，可做分片操作*flist=list(map(**lambda** x: os.path.splitext(x)[0] ,flist))  
print(flist)