定义函数

在Python中，定义一个函数要使用def语句，依次写出函数名、括号、括号中的参数和冒号:，然后，在缩进块中编写函数体，函数的返回值用return语句返回。

我们以自定义一个求绝对值的my\_abs函数为例：

**def** **my\_abs**(x):

**if** x >= 0:

**return** x

**else**:

**return** -x

请自行测试并调用my\_abs看看返回结果是否正确。

请注意，函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。因此，函数内部通过条件判断和循环可以实现非常复杂的逻辑。

如果没有return语句，函数执行完毕后也会返回结果，只是结果为None。

return None可以简写为return。

在Python交互环境中定义函数时，注意Python会出现...的提示。函数定义结束后需要按两次回车重新回到>>>提示符下：

如果你已经把my\_abs()的函数定义保存为abstest.py文件了，那么，可以在该文件的当前目录下启动Python解释器，用from abstest import my\_abs来导入my\_abs()函数，注意abstest是文件名（不含.py扩展名）：

import的用法在后续[模块](http://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000/0014318447437605e90206e261744c08630a836851f5183000)一节中会详细介绍。

空函数

如果想定义一个什么事也不做的空函数，可以用pass语句：

**def** **nop**():

**pass**

pass语句什么都不做，那有什么用？实际上pass可以用来作为占位符，比如现在还没想好怎么写函数的代码，就可以先放一个pass，让代码能运行起来。

pass还可以用在其他语句里，比如：

**if** age >= 18:

**pass**

缺少了pass，代码运行就会有语法错误。

参数检查

调用函数时，如果参数个数不对，Python解释器会自动检查出来，并抛出TypeError：

>>> my\_abs(1, 2)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: my\_abs() takes 1 positional argument but 2 were given

但是如果参数类型不对，Python解释器就无法帮我们检查。试试my\_abs和内置函数abs的差别：

>>> my\_abs('A')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 2, in my\_abs

TypeError: unorderable types: str() >= int()

>>> abs('A')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: bad operand type for abs(): 'str'

当传入了不恰当的参数时，内置函数abs会检查出参数错误，而我们定义的my\_abs没有参数检查，会导致if语句出错，出错信息和abs不一样。所以，这个函数定义不够完善。

让我们修改一下my\_abs的定义，对参数类型做检查，只允许整数和浮点数类型的参数。数据类型检查可以用内置函数isinstance()实现：

**def** **my\_abs**(x):

**if** **not** isinstance(x, (int, float)):

**raise** TypeError('bad operand type')

**if** x >= 0:

**return** x

**else**:

**return** -x

添加了参数检查后，如果传入错误的参数类型，函数就可以抛出一个错误：

>>> my\_abs('A')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 3, in my\_abs

TypeError: bad operand type

错误和异常处理将在后续讲到。

返回多个值

函数可以返回多个值吗？答案是肯定的。

比如在游戏中经常需要从一个点移动到另一个点，给出坐标、位移和角度，就可以计算出新的新的坐标：

**import** math

**def** **move**(x, y, step, angle=0):

nx = x + step \* math.cos(angle)

ny = y - step \* math.sin(angle)

**return** nx, ny

import math语句表示导入math包，并允许后续代码引用math包里的sin、cos等函数。

然后，我们就可以同时获得返回值：

>>> x, y = move(100, 100, 60, math.pi / 6)

>>> print(x, y)

151.96152422706632 70.0

但其实这只是一种假象，Python函数返回的仍然是单一值：

>>> r = move(100, 100, 60, math.pi / 6)

>>> print(r)

(151.96152422706632, 70.0)

原来返回值是一个tuple！但是，在语法上，返回一个tuple可以省略括号，而多个变量可以同时接收一个tuple，按位置赋给对应的值，所以，Python的函数返回多值其实就是返回一个tuple，但写起来更方便。

小结

定义函数时，需要确定函数名和参数个数；

如果有必要，可以先对参数的数据类型做检查；

函数体内部可以用return随时返回函数结果；

函数执行完毕也没有return语句时，自动return None。

函数可以同时返回多个值，但其实就是一个tuple。

练习

请定义一个函数quadratic(a, b, c)，接收3个参数，返回一元二次方程：

ax2 + bx + c = 0

的两个解。

提示：计算平方根可以调用math.sqrt()函数：