

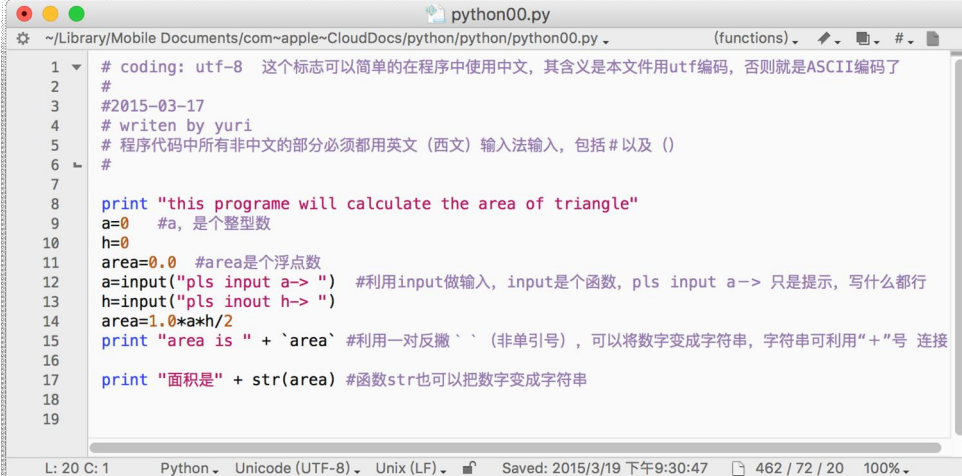
第一章 通过求三角形面积步入 Python 程序世界

本章通过一个求三角形面积的案例带领读者快速进入 Python 世界，案例涉及一些编程必须知道的内容，包括：

- 1) Python 程序的体例
- 2) 语句间的分隔
- 3) 变量的定义和使用
- 4) 利用 print 输出
- 5) 利用 input 输入
- 6) 数据类型
- 7) 将数字转换成字符串
- 8) 如何使用中文
- 9) 给程序标注注释

案例 1-1 求三角形面积

例程 1-1：求三角形面积，并输出。代码见图 1-1。



```
python00.py
~/Library/Mobile Documents/com~apple~CloudDocs/python/python/python00.py (functions) #
1 # coding: utf-8 这个标志可以简单的在程序中使用中文，其含义是本文文件用utf编码，否则就是ASCII编码了
2 #
3 #2015-03-17
4 # writen by yuri
5 # 程序代码中所有非中文的部分必须都用英文（西文）输入法输入，包括#以及()
6 #
7
8 print "this programe will calculate the area of triangle"
9 a=0 #a,是个整数
10 h=0
11 area=0.0 #area是个浮点数
12 a=input("pls input a-> ") #利用input做输入，input是个函数，pls input a-> 只是提示，写什么都行
13 h=input("pls inout h-> ")
14 area=1.0*a*h/2
15 print "area is " + `area` #利用一对反撇`（非单引号），可以将数字变成字符串，字符串可利用"+"号连接
16
17 print "面积是" + str(area) #函数str也可以把数字变成字符串
18
19
L: 20 C: 1 Python, Unicode (UTF-8), Unix (LF) Saved: 2015/3/19 下午9:30:47 462 / 72 / 20 100%
```

图 1-1 求三角形面积代码

导读

Python 程序非常简单，从书写的第一行开始执行，到最后一行结束，语句之间用“回车”分隔，即每行为一个单独语句。程序中用“#”标识“注释”，所有的注释都是不执行的，程序中若无特殊说明包括需要输出的字符串在内都应该是 ASCII 编码的英文，但随着各种语言使

用的增多，若在程序中使用“# coding: utf-8”标记，那么自该行以后，程序中可以出现 UTF8 编码的字符，例如中文。

第 1 行，利用 `print` 输出一个字符串，Python 的字符串写在双引号中：“象这样”，单引号也可以：'看这里'。

第 2, 3, 4 行分别定义 3 个变量：`a`, `h` 和 `area`。Python 与大多数其他语言一样，变量需要先定义再使用，但是它没有显式的变量声明形式。而是以赋初值形式完成声明的工作，这种做法虽然不同寻常，但是避免了无初值变量的产生。

第 4, 5 行利用 `input` 函数做输入，同 C 和 Java 不同，Python 的输入可以带一个文本提示，并会自动识别需要的数据类型，而不需要进行数据类型的限定与转换。

第 6 行是一个不同数据类型的混合计算过程，要注意数据类型，若只计算 `a*h/2`，则计算会将小数部分省略，而计算规则规定不同类型混合计算时简单类型会转为复杂类型，所以在 `a*h/2` 之前乘 1.0 从而得到浮点型的结果。

第 7, 8 行演示了数字与字符串的转换和输出，一对反撇号```（不是单引号）就可以便捷地将数字变成字符串，而字符串则可利用“+”号直接连接。

边学边练：

仿照上例完成一个梯形面积计算的程序。

知识梳理与扩展

1. 语句的缩进与结束

Python 用“分行”来表示一个语句的结束，一行就是一个语句，语句在书写格式上要严格遵守“缩进原则”。Python 没有利用“{ }”或“begin...end”来标志代码块的开始与结束，而是利用了“缩进”这种更接近人类书写的方式，但是程序员必须保证相同语句块的缩进保持一致，子块必须使用比父块更多的缩进，否则，就会引发“`IndentationError: unexpected indent`”错误。缩进这种强制规则使源文件的排版更加规则，更具有可读性。

2. 注释

评价程序好坏的一个重要依据是它的源代码是否能被人看懂，有时这甚至比它是否可以执行更重要。为此需要程序员通过标注来解释程序的目标、方法、意图、思路等等，这些标注称为注释。Python 用“#”来表示从#标记开始到本行结束是注释。注释的内容不会被程序执行，一定要注意 Python 的注释标记只在“当前行”有效。

如果需要进行多行的注释，Python 提供一种变通的方法：可以利用三个单引号，比如：`'''`。

```
'''
trans the data during down_edge,start in first clk
t_byte:the data(pyhton int)
B_width:how many bits in the data(8,16,32,64 ...)
LM_mask:LSB mask (... 0x800,0x80,0x8) or MSB mask (1)
'''
```

这时“三引号”中的内容被当做一个匿名的长文本，权当注释使用了。

3. 值与类型

按日常的概念，我们经常处理的信息应该可以分为：数字和非数字，由于计算机处理数据时需要做更详细的数据类型区分，所以我们进一步将数字分为：整数和带小数点的数。

其中整数分为 `int` 整型，`long` 长整型；而使用小数点的数据，我们称为实型数据：`float`。

在 Python 中，必须明确数据的使用类型，Python 的特点是不使用数据名称而使用数据实例来说明类型，这样的好处在于**永远也不会产生未定义初值的危险**，但是开发者要注意：若 $a=3$ ，则在计算完 $a=a/2$ 后， a 的值是 1，因为 a 被定义为整数，因而不可能拥有小数点；而若定义 $b=3.0$ ，则计算完 $b=b/2$ 后， b 的值是 1.5，因为 b 被定义为浮点数。

对于非数字的量，统称为字符串，可以这样表示：'使用单引号标志'，或"使用双引号标志"，双引号中可以嵌套使用单引号，比如："这是一个'单、双引号混合使用'的字符串示例"。

Python 中的数据类型还包括复数型：complex，数据示例如表 1-1 所示。

表 1-1 数据示例

int	long	float	complex
9	789191123L	0.0	123.45j
99	-0x15678L	15.2560	456.j
-78	01234L	-213.9	0.32e-36j
070	0x345AB789AEL	31.5+e18	35e+876j

Python 还提供了一个“内置函数”`type()`，用来观察数据的类型：若有定义 $a=3$ ，执行 `print type(a)` 之后，结果是：`int`。

4. 变量、标识符

Python 允许程序员用给数据量起名的方法区分和使用程序中的值，一般情况下，由于程序中的量参与计算时会按需要变化，所以它们被称为变量，程序员所起的名字被称为“标识符”，标识符必须遵守以下命名规范：

- 1) 可以由字母、数字、下划线组成
- 2) 标识符长度不限
- 3) 必须由字母和下划线开始
- 4) 大小写字母表达不同标识符
- 5) 不可以使用 Python 的关键字

另外还有一些约定俗成的标识符（起名）规范，违反了并不是错误，但可能会引起项目中其他合作伙伴的不快或困惑，比如：名字太长、太短或通过名字猜不出变量的大致作用。

下面比较好坏两种标识符：

好的标识符：`days_in_month`

不太好的标识符：`k1, m2`

在程序中也会临时使用一些简短的标识符来解决一些临时性的问题，但这种标识符的使用范围不要超过 10 行，而且要提供注释。

还有一些约定俗成的变量名称，例如：

`i`：经常用于循环

`tmp`：这是一个临时变量

`sum`：和

`max, min`：最大和最小

5. 常量

Python 没有常量机制，但是我们有时确实需要提醒自己或同伴某些数据不可改变，那么我们将其名称用大写。例如 $PI=3.14$ 。

6. 输入

程序的数据输入可以使用 `input`。`input` 是个输入函数，其格式是：

```
变量=input("提示信息")
```

但是 `input` 通常用来输入数值型数据，若要输入字符串的数据可使用：

```
变量=raw_input("提示信息")
```

简单地说，`input` 返回数值型数据，`raw_input` 返回字符串型数据，其实，`input` 函数只是对 `raw_input` 函数做了“数值转换”。

7. 输出

程序的输出可以使用 `print`。`print` 输出一个字符串，所以需要输出的数据转换为字符串，具体可参见本章的例程。

8. 字符串的连接、倍增和转换

字符串数据和字符串不能混合运算，但是我们可以利用“+”和“*”来进行字符串的连接和“倍增”：

例如 `'China'+''+'Beijing'` 的结果是 `'China Beijing'`。

而 `'A'*5` 的结果是：`'AAAAA'`。

最简单的将数字转换成字符串的方法有两种：

1) 利用一对反撇号（反撇号和单引号不一样）：``数值`` 比如：

```
a=123
```

```
b=456
```

```
c=a+b
```

```
d="a connect b =>"+`a`+`b`
```

那么 `c` 的结果是 579，`d` 的结果是：`a connect b => 123456`

2) 利用 `str` 函数也可以达到同样的效果

```
d="a connect b =>"+str(a)+str(b)
```

9. 程序中使用 UTF 编码的方法

最后和大家讨论使用其他文字编码的问题，最简单的方法是在程序的第一行以注释的方式作以下标注：

```
# coding: utf-8
```

这个标志的含义是：“本文件用 UTF 编码”，UTF 编码支持多种文字，包括中文，若不做此标注，那么程序就使用默认的 ASCII 编码，ASCII 编码只支持英文。要注意的是，即便使用了“`#coding: utf-8`”标记，程序代码中所有非中文的部分也必须都用英文输入法输入，比如本行开始的“#”必须是英文的，不能是中文的，包括运算符也必须使用英文。

小结

在 Python 中，变量必须先赋初值再使用，虽然没有其他语言的定义环节，但 Python 确

实是“强类型语言”，即所有的量必须有类型，只是它不用显式指定类型，而是第一次赋给数据的类型确定了它的类型，其实这种机制也不错，变量被强制赋了初值，这使得变量使用更安全。

算术运算与平常没有区别，但是要注意计算中数据类型从简单（低级）向复杂（高级）变化，而赋值号（“=”）将把右边数据的类型转变为左边数据的类型。

此外本章还介绍了输入输出语句和字符串的一些简单使用方法。掌握了这些，我们完全可以进行简单的编程了。

练习一

1. 编程利用 $PI=3.1415$ 求圆的面积。
2. 编程打印下面表格：

姓名	年龄	籍贯
张三	32	北京
李四	45	天津
王五	28	河北

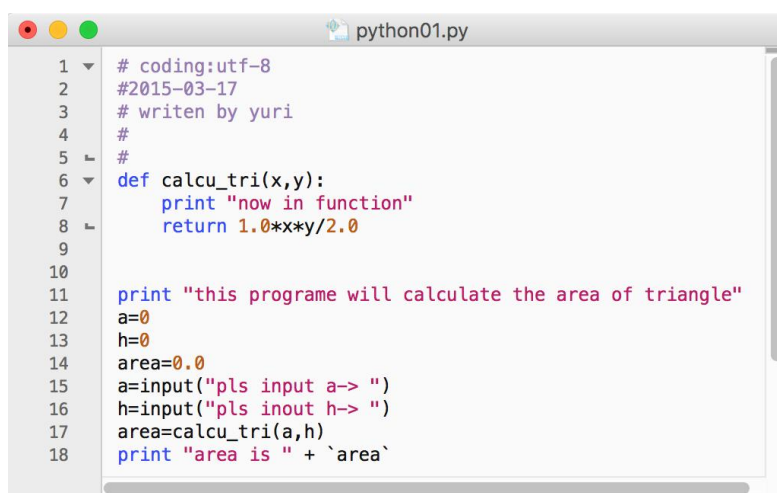
第二章 常用运算、使用自定义函数

在本章读者们将实践函数的用法，包括定义、使用、引用等等。本章所涉及的技术要点包括：

- 1) 函数的定义和使用
 - 2) 形参、实参、返回值
 - 3) 局部变量与全局变量
 - 4) 常用运算
 - 5) 利用 import 导入机制
 - 6) Python 的格式化输出
- 开门见山，看下面的例程。

案例 2-1 用函数的方法计算三角形面积

本案例代码见图 2-1。



```
python01.py
1 # coding:utf-8
2 #2015-03-17
3 # writen by yuri
4 #
5 #
6 def calcul_tri(x,y):
7     print "now in function"
8     return 1.0*x*y/2.0
9
10
11 print "this program will calculate the area of triangle"
12 a=0
13 h=0
14 area=0.0
15 a=input("pls input a-> ")
16 h=input("pls inout h-> ")
17 area=calcul_tri(a,h)
18 print "area is " + `area`
```

图 2-1 案例 2-1 代码

导读

无论已经提过多少遍都要再次强调的是：Python 是“先定义再使用”的语言。所以函数在使用之前也需要定义。

于是，上面的程序，把 6~8 行的函数定义放到 19 行之后就会有错误。

案例第 6 行，用 def 关键字定义了一个函数，名字叫 calcul_tri，这种形式就是函数的定义，calcul_tri 函数的参数是 x 和 y，然后使用缩进的方法标志函数的范围，calcul_tri 函数只有两条

语句，这两条语句的缩进（句首空格数）相同，而从第 11 行开始就不是 `calcu_tri` 函数的范围了。Python 利用排版的缩进格式表达语句的归属范围，第 7，8 行的缩进格式表明，这两句话隶属于第 6 行定义的函数。

第 7 行 `calcu_tri` 函数的功能，首先通过打印信息提示一下程序现在运行的位置。

第 8 行 `return` 也是一个 Python 语法关键字，顾名思义，函数将在此返回（到调用位置），并带回一个值，即： $1.0*x*y/2.0$ 。也就是说 `calcu_tri` 函数传入参数 `x,y`；并返回按公式 $1.0*x*y/2.0$ 计算所得到的值。

第 11 行取消了函数 `calcu_tri` 的缩进，表示回到程序主框架的范畴，到第 15，16 行，输入了 `a` 和 `h`。

第 17 行，像数学中调用函数一样，程序将 `a, h` 当作参数调用了 `calcu_tri` 函数，然后程序加载 `calcu_tri` 函数并运行，直至运行到 `calcu_tri` 函数的 `return` 语句，再回到函数被调用的位置，可以看到第 17 行利用一个赋值将 `calcu_tri` 函数的值给了 `area`，然后在第 18 行打印。

另外 Python 也支持格式化输出：第 18 行可以改成

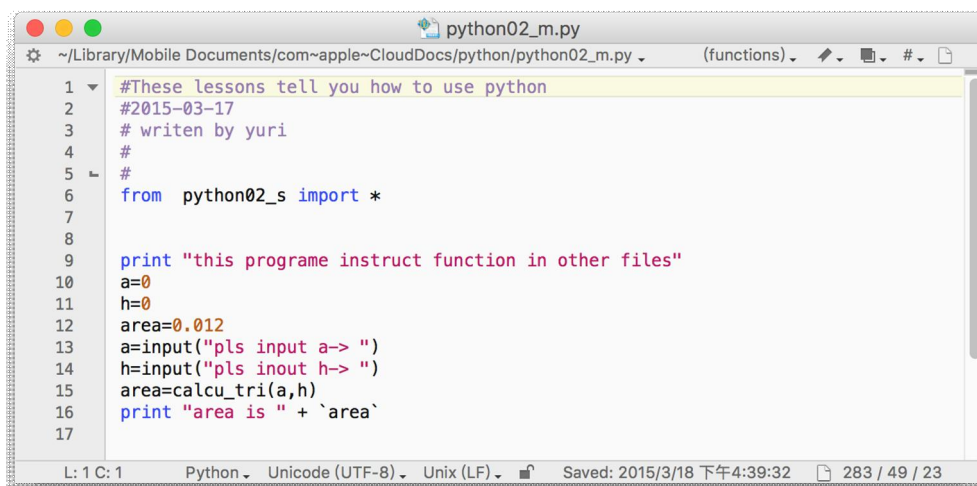
```
print "area is"+ '{:d}'.format(area)
```

这里先提一下格式化输出，Python 的格式化输出十分有特点，需要一段时间才能讲清楚，现在我们用不到很复杂的方式，暂时知道输出整型数用 `'{:d}'.format(整型数)`，就行了，不用细究。

在真实的应用中，形如案例 2-1 的函数使用方式没什么实际用途，最大的不方便就是函数不能作为一个“工具包”被其他程序利用（即复用）。所以把函数打包十分重要，请看案例 2-2。

案例 2-2 开发一个求三角形面积的工具包

本案例代码如图 2-2 和图 2-3 所示。

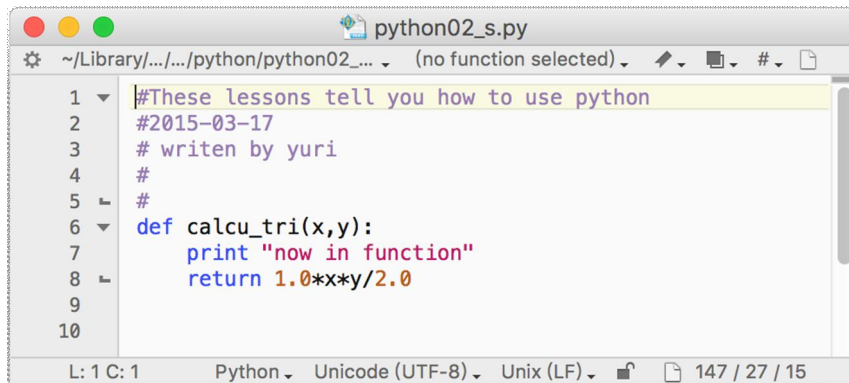


```

1 #These lessons tell you how to use python
2 #2015-03-17
3 # writen by yuri
4 #
5 #
6 from python02_s import *
7
8
9 print "this programe instruct function in other files"
10 a=0
11 h=0
12 area=0.012
13 a=input("pls input a-> ")
14 h=input("pls inout h-> ")
15 area=calcu_tri(a,h)
16 print "area is " + `area`
17

```

图 2-2 python02_m.py 代码



```

1  #These lessons tell you how to use python
2  #2015-03-17
3  # written by yuri
4  #
5  #
6  def calcul_tri(x,y):
7      print "now in function"
8      return 1.0*x*y/2.0
9
10

```

图 2-3 python02_s.py 代码

案例 2-2 示范了如何使用外部文件的资源，这样所有函数都可以被新程序复用了。本案例包括两个文件，一个叫 python02_m.py，程序将调用 calcul_tri 函数，但是这个函数在另一个文件 python02_s.py 中。于是在 python02_m.py 的第 6 行：

```
from python02_s import *
```

这句话的含义是“从 python02_s 中引入所有函数”，于是，在 python02_m.py 中可以使用 calcul_tri 函数了。

注意，使用 from 引用文件模块的时候不要带文件名的后缀“.py”。

为什么要分成两个文件呢？这样就可以方便地进行工具包（函数库）的复用了，比如开发人员完成了求各种图形面积的函数，那么求图 2-4 中的图形阴影部分面积时，只要把包含求简单图形面积的函数文件导入（import）进来，然后把各种函数组合一下，就完成了开发任务。这样开发就简单多了，所以尽量使用多文件机制定义函数，否则函数的功效将大打折扣。



图 2-4 图形阴影

知识梳理与扩展

1. 函数的定义和调用

函数在调用前必须定义，比如在函数 A 中调用函数 B，被调用的函数叫“被调函数”，例如同上面提到的函数 B，那么函数 A 就是主调函数，函数被调用时使用到的参数叫“实参”（实际参数），而函数定义时使用的参数叫形参（形式参数）。例如上面的示例，calcul_tri 函数按参数排列顺序接收 a, h 的值到 x, y 中，x, y 属于 calcul_tri 函数定义时使用的参数，即“形参”，而 a, h 即实参。calcul_tri 函数中 return 后面的叫“返回值”。之所以出现实参、形参、返回值的说法对编程来说没什么意义，只是教学和交流时指代比较明确。

在 Python 中形参和实参值间采用值传递的机制，形参的变化不会改变实参。例如：

```
#形参不改变实参的示例
def fun1(a):
    a=a+1
    print 'a='+`a`
    return 0
k=10
print 'Before k='+`k`
fun1(k)
print 'After k='+`k`
```

运行以上简短代码，读者会发现调用函数 fun1 前后 k 的值并没有发生变化。

2. 全局变量与局部变量

简单的说在函数内部定义的变量的使用范围仅限于函数内部，被称为局部变量，而不属于任何函数的变量就是全局变量。但是想要形象地理解全局变量和局部变量有一个简单的判断方法，那就是变量的作用范围由缩进格式标志的代码块确定，在一个代码块声明的变量仅限于本（级）代码块使用。

```
#全局变量与局部变量示例

def fun1(a):
    a=a+1
    print 'a='+`a`
    return 0
k=10
print 'Before k='+`k`
fun1(k)
print 'After k='+`k`
```

3. 常用运算

Python 提供丰富的计算功能，其中算术运算有：+（加），-（减），*（乘），/（除），%（取余）。它们的运算规则与常规算术运算一样，若要提高某部分表达式的运算优先级则在该部分表达式外面使用“()”，“()”的使用可以嵌套。另外进行除法运算时除数不能为 0，计算机语言中的计算与日常计算不同，需要注意数据的类型。将常用的运算总结如下：

(1) 括号()

括号内的数据优先处理，像其他语言一样，Python 只提供一种括号，可以嵌套使用。

(2) 算术运算符（见表 2-1）

表 2-1 算术运算符

算术运算符	意义	说明
+	加法运算符	遵循数学运算规则
-	减法运算符	

续表

算术运算符	意义	说明
*	乘法运算符	
/	除法运算符	
%	模运算、求余运算符	只能用于整型
**	幂运算	2**7, 即求 2 的 7 次幂

Python 支持我们常用的算术运算, 算术运算的对象必须是数值、加、减、乘、除和括号(), 但是有一个规则, 即同型数据计算只能获得同型的结果, 例如 $1/2$ 是两个整型数据计算, 其结果就是 0, 不同数据类型做算术运算, 结果类型会跟随“更复杂”的数据类型。比如 $1.0/2$ 的结果就是 0.5。

Python 世界对算术运算优先级的设置和现实世界是一样的。

(3) 关系运算符 (见表 2-2)

表 2-2 关系运算符

关系运算符示例	说明
$a > b$	a 大于 b 时为真
$a \geq b$	a 大于等于 b 时为真
$a < b$	a 小于 b 时为真
$a \leq b$	a 小于等于 b 时为真
$a == b$	a, b 相等时为真
$a != b$ 或 $a <> b$	a, b 不相等时为真, 利用 $<>$ 表达“不等于”运算的方法会逐步淘汰

注意: Python 的关系运算和逻辑运算中用布尔值表达两种结果: True(真)以及 False(假)。

(4) 逻辑运算符 (见表 2-3)

表 2-3 逻辑运算符

逻辑运算符示例	说明
$a \text{ and } b$	a、b 皆不为假时为真, 否则为假
$a \text{ or } b$	a、b 皆为假时为假, 否则为真
$!a$	a 为假时为真, 否则为假

(5) 赋值运算符

=、+=、-=、*=、/=

(6) 运算优先级

以上列出的运算符优先次序为: 括号, 算术运算, 关系运算, 逻辑运算, 赋值运算。

4. import 模块导入机制

利用 import 导入机制可以将已有的函数功能模块导入程序。这个机制方便代码复用, 但要注意引用功能模块的文件时不需要后缀名。

5. 格式化输出

Python 支持数据的格式化，优势是可以方便地将数据格式化成相应的字符串。常用的格式化功能如下：

- 1) 基本格式：`STRING.format(表达式)`，`STRING` 表达式字符串。
- 2) 字符串中加入形如 `{:^nd}` 的格式控制符，用以控制 `format` 中表达式的输出形式。
- 3) 常用格式控制符有：`{:^nd}` 用来输出整型，`n` 代表输出所占字符列数；`{:^m.nf}` 用来输出浮点型，`m` 代表输出所占字符列数，`n` 代表小数点部分的位数。
- 4) 若指定输出的字符列数大于实际输出，则输出空格占位；若实际输出字符数大于指定，则按实际输出。

5) Python 还有一种类似 C 语言的格式化输出的机制，例如：

```
a=3
b=5
print "%d +%d=%d"%(a,b,(a+b))
```

运行结果是：`3+5=8`，分析这个简单的例子，可以得知，双引号中的“%”是格式控制，而双引号外部用一个“%”引导需要输出的变量表列。

总结格式控制符如表 2-4 所示。

表 2-4 格式控制符

格式描述	格式控制符	
占据 <code>n</code> 列的整数	<code>%nd</code>	<code>{:^nd}</code>
以十六进制格式输出整数	<code>%0x</code>	<code>{:^0x}</code>
占据 <code>m</code> 列保留 <code>n</code> 位小数的 <code>float</code> 型数据	<code>%m.nf</code>	<code>{:^m.nf}</code>
占据 <code>n</code> 列的字符串	<code>&ns</code>	<code>{:^ns}</code>

小结

对一个开发人员而言，代码复用意味着工作经验的积累，函数为开发人员提供了这种机制。开发人员可以把一些常用的功能抽象成函数形式，再把同类的多个函数（例如求各种不同图形面积的函数）打包成一个文件，这样就可以在开发其他程序时便利地使用以前的工作成果了。另外这种机制也可以支持小组开发，例如多名开发人员组成小组，规定好功能间的接口，然后分别提交不同的功能文件，通过函数的相互调用完成程序的功能，这样可以较大地提升开发效率。

在定义函数的过程中要注意函数代码的范围，特别是函数中的缩进格式是初学者最容易犯错误的地方。Python 鼓励开发人员尽量写短的代码，而使代码变短的方法当然包括将功能包装成函数。

Python 在函数调用过程中使用值传递的形式，所以主调函数中的实参不会因形参在被调用函数中变化而变化。

练习二

1. 尝试自编一个面积工具包求图 2-4 中阴影部分面积。
2. 海伦公式的用途是利用三角形三条边求面积，编写一个函数，可以利用海伦公式求三角形面积。
3. 编写一个求矩形面积的函数。
4. 利用第 2、3 题的成果编写程序，输入梯形的四条边，求梯形面积。