21世纪高等教育计算机规划教材

COMPUTER

Python程序设计

Python Programming

赵英良 主编

卫颜俊 仇国巍 郑义 编著

- 化繁为简，降低初学者入门难度
- 突出重点，帮助学习者掌握核心
- 内容全面，开阔程序员视野范围

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等教育计算机规划教材

COMPUTER

Python程序设计

Python Programming

赵英良 主编
卫颜俊 仇国巍 郑义 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

Python程序设计 / 赵英良主编 ; 卫颜俊, 仇国巍, 郑义编著. -- 北京 : 人民邮电出版社, 2016.5
21世纪高等教育计算机规划教材
ISBN 978-7-115-41834-0

Ⅰ. ①P… Ⅱ. ①赵… ②卫… ③仇… ④郑… Ⅲ. ①软件工具－程序设计－高等学校－教材 Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2016)第036966号

内 容 提 要

本书以 Python 3.x 为编程环境，系统介绍了 Python 语言的特点、语法规则、应用方法以及程序设计的基本思想和基本方法。内容包括：Python 环境的基本使用方法、Python 的基本语法规则、数据类型（含列表等复杂类型）、运算符、表达式、控制结构、异常处理、函数、文件、迭代器、面向对象程序设计、图形界面程序设计、数据库程序设计以及网络程序设计等。书中贯穿的算法包括累加和的计算、排序、查找、一元非线性方程求根、积分、递推、递归、逻辑推理等。

本书内容全面、语言简练、实例丰富、习题量大，注重培养计算机程序求解问题的思维方式。本书可作为高等学校计算机程序设计课程的教材或参考书，也可供程序设计爱好者、工程技术和软件开发人员学习参考。

◆ 主　　编　赵英良
编　　著　卫颜俊　仇国巍　郑　义
责任编辑　邹文波
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：13.5　　2016 年 5 月第 1 版
字数：354 千字　　2016 年 5 月北京第 1 次印刷

定价 39.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言

五年前我接触到了 Python，当初只是觉得它是一种简单易学的语言，就用它作为学生学习算法的入门语言。因为要给学生讲，就需要不断学习，渐渐地，发现它非常强大，第三方的功能库也很多，使用非常方便。在这个上 GB 的时代，Python 的最新版的 IDE 安装包也只有二十多 MB。它的体积虽小，却既支持面向对象程序设计，又可编写图形界面软件，甚至可以在手机上编写手机应用程序。不少同事也喜欢上了这门语言，将它作为科研的工具。Python 有些功能让使用过多种语言的人印象深刻。比如，它变量的类型是可以改变的；函数的返回值可以有多个；函数的参数个数甚至可以是不确定的；实参的顺序和形参的顺序可以不同；赋值时，可以同时给多个变量赋值；典型的交换两个变量的值的三行程序，在 Python 中只要一行，等等。如果用其他语言，当你还在琢磨用什么数据结构，控制结构如何表达的时候，Python 一条语句就解决了。如果认识了 Python，用神奇、震撼形容它不足为过。所以，有一个愿望，愿意将它推荐给更多的人。

本书的特点如下。

（1）内容简洁，开门见山，直奔主题。

（2）内容比较全面，本书不仅较全面地介绍了 Python 的基本语法，还涉及网络编程、数据库编程和图形界面编程等较高级的编程内容。

（3）内容充实，每一部分都能使读者学到有用的知识。

本书语言简练、实例丰富、习题量大，不仅涵盖了 Python 的基础及特色语法知识，而且注重讲述计算机程序求解问题的思想方法。本书可作为高等学校计算机程序设计课程的教材或参考书，也可供程序设计爱好者、工程技术和软件开发人员学习参考。

本书由赵英良担任主编，卫颜俊、仇国巍、郑义共同编著，其中，赵英良编写了第 1 章、第 2 章、第 8 章，郑义编写了第 3 章，仇国巍编写了第 4 章、第 5 章、第 6 章，卫颜俊编写了第 7 章、第 9 章、第 10 章。本书在编写过程中得到国家级教学名师冯博琴教授的关心和帮助，在此表示感谢，同时也向参考文献的作者及相关人员表示感谢。

编 者

2016 年 3 月

目录

第1章 Python入门

计算机是一种自动计算装置。然而计算什么？怎么计算？需要人们以命令的形式告诉计算机。所有命令、符号及使用规则的集合，就是计算机语言。人们用自然语言写成文章，记录事实，表达意愿，交流信息。为了用计算机解决问题，人们需要按一定的顺序使用计算机命令，这种为解决问题，用计算机语言表达的命令的序列就是计算机程序。计算机程序及相关文档的集合称为计算机软件。没有计算机软件，再好的硬件也无法发挥其性能，所以有人称软件是计算机的“灵魂”。

1.1 计算机语言的发展

计算机语言的发展经历了机器语言、汇编语言和高级语言等几个阶段。

1. 机器语言

电子计算机是由电子元件和线路组成的，用电子信号表示数据和要执行的操作（也就是命令，计算机中称指令）。命令的表现形式就是“0”“1”组成的序列。不同的序列，可以表示不同的指令，称为**指令的编码**，这样的编码系统称为**机器语言**。人们把要做的事情用机器语言的指令序列表达出来，这便是计算机程序（机器语言程序）。机器语言是计算机可以直接“理解”的语言，机器语言的程序是计算机可以“看得懂”的“文件”，它可以遵照执行，完成人们交给它的任务。

机器语言用二进制数表示命令，计算机可以直接执行。然而，无论是程序的编写还是阅读，对于程序员来说，都是一件困难的事情，特别是当程序有错误时，要想查找并修改错误，更是非常困难，因为程序员看到的是一系列的数字。

2. 汇编语言

20世纪40年代，研究人员为了简化程序设计的过程，开发了记号系统，使用单词的缩写符号来表示指令，而不再使用数字形式，这些符号称为**指令助记符**。同时也用符号表示数据（在汇编语言中称操作数）、数据的存放地址以及CPU中暂时存放数据的装置——寄存器（Register）等。例如，使用ADD表示加，MOV表示移动数据，JZ表示转移等。

用指令助记符、地址符号等符号表示的指令称为**汇编格式指令**（Assemble Instruction）。汇编格式指令及其表示和使用这些指令的规则，称为**汇编语言**(Assembly Language)。用汇编语言编写的程序称为**汇编语言程序**或**汇编语言源程序**，或简称**源程序**。

由于汇编语言使用了助记符和用符号表示数据的存储位置（称为存储单元的地址，简称地址），

因此它自然比机器语言更容易掌握和使用。然而，机器只能识别机器语言表示的指令，如“MOV CX,E024”要翻译为 B924E0H，所以用汇编语言编写的程序并不能被计算机直接执行，还需要将它们翻译为一系列的机器指令。实际上，翻译工作并不需要人来做，可以用机器语言编写一个程序来做这项工作，这个程序称为**汇编程序**（Assembler）。翻译的过程称为**汇编**（Assemble），翻译的结果称为**目标程序**（Object Program）。

汇编语言是在机器语言基础上的巨大进步，被称为第二代程序设计语言。然而，由于汇编格式指令是机器指令的符号表示，而不同的 CPU 能识别的机器指令可能是不同的，所以汇编格式指令与机器有着密切的关系，也就是说，就一种 CPU 编写的求解某一问题的程序在另一种 CPU 的机器上不一定能正确执行。另一个缺点是程序员在编写求解问题的程序时，不仅需要关心如何求解工程问题还需要考虑计算机中的寄存器、数据的存储位置、内存的容量等细节问题。所以机器语言、汇编语言又被人们称为**低级语言**。

3. 高级语言

1953 年，美国 IBM 公司的约翰·贝克斯(John W.Backus)向他的主管提出一项建议，开发一种更实用的计算机语言，代替汇编语言，为他们的计算机 IBM 704 编写程序，这就是 FORTRAN 语言（IBM mathematical Formula translating system）。1954 年，他完成了计算机语言的详细说明书的编写。1956 年 10 月第一本 FORTRAN 指南问世。1957 年 4 月第一个 FORTRAN 编译器被开发出来。约翰·贝克斯（John W.Backus）说：“我的工作来源于懒惰。我不喜欢写程序，所以当我参加 IBM 704 项目，为计算弹道写程序时，我开始设计一套编程系统以使写程序更容易”。

从 FORTRAN 开始，计算机科学家又开发了多种语言，如 COBOL、BASIC、PASCAL、 C、C++等。它们的特点一是与机器无关，使用这些语言编写的程序可以较容易地移植到不同的计算机上；二是它们的命令注重描述解决问题的方法和步骤，而不是某种机器的指令。所以它们又称为**高级语言**。

高级语言的命令也是用单词或缩写符号来表示的，更接近于问题的求解方法，因而容易被人理解，但这样的程序也不能被计算机直接识别，所以，也需要翻译成机器语言命令的程序，这样的程序称为**编译器**（Complier）。通常，一条高级语言的命令（有时称为语句）编译后会对应几条机器指令。对不同的计算机系统，可能翻译后的机器指令序列也不同。

编译器一次将高级语言程序翻译成可执行的机器指令序列，以后再执行程序时不再需要翻译。还有另外一种翻译的策略，就是翻译的同时执行指令，实际是翻译一条高级语言命令，接着就执行这些机器指令，然后再翻译下一条高级语言命令并执行。这样的翻译方式称为**解释执行**，这样的翻译程序称为**解释器**（Interpreter）。

语言是一套规则，编译和解释是语言的实现方式。一般以编译方式实现的语言称为**编译型语言**，如 FORTRAN、C、C++等。一般以解释方式实现的语言称为**解释型语言**，如 BASIC、PHP、Python 等。但这种划分不是绝对的。

1.2 Python 简介

Python 是一种面向对象的解释型高级计算机程序设计语言，1989 年由吉多·范罗苏姆（Guio Van Rossum）开发设计。

1.2.1　Python 的特点

1. 简单

Python 的设计哲学是优雅、明确、简单。Python 关键字少，结构简单，语法清晰，易读，易维护。学习 Python 可以在短时间内轻松上手。Python 使用缩进格式。

2. 高级

Python 是高级语言，内置高级数据结构，程序员无需关心底层细节（如内存的分配和回收），可以更高效地编写求解工程、科学问题的应用程序。

3. 面向对象

Python 支持面向过程的编程，也支持面向对象的编程，还是一种函数式编程语言。

4. 可扩展

Python 提供丰富的 API 和工具，以便程序员能够轻松地使用 C、C++语言来编写扩充模块。

5. 免费和开源

Python 是自由/开放源码软件(Free/Libre and Open Source Software，FLOSS），允许自由地发布此软件的拷贝，阅读和修改其代码，或将其中的一部分用于新的自由软件中。

6. 可移植

Python 程序可以在 Unix/Linux、Windows、Macintosh 等不同的平台上运行。

7. 丰富的库

Python 语言提供功能丰富的标准库，如网络、文件、数据库、图形界面、正则表达式、文档生成、单元测试等。用 Python 开发，许多功能不必从零编写，直接使用库中已有程序即可。除了内置的库外，Python 还有大量的第三方库，如科学计算库 NUmPy、SciPy、Matplotlib 等。自己编写的程序，也可以作为第三方库给别人使用。

8. 丰富的接口

Pytohn 提供面向其他系统和专用库的接口，如数据库管理系统、计算机视觉库 OpenCV、三维可视化库 VTK、医学图像处理库 ITK 等。还可以将 Python 嵌入 C、C++程序中。

Python 的应用非常广泛，如科学计算、自然语言处理、图形图像处理、游戏开发、日常管理小工具、系统管理、Web 应用等。许多大型网站就是用 Python 开发的，如 YouTube、Instagram。很多大公司的应用，包括 Google、Yahoo 等，甚至 NASA（美国航空航天局）都大量地使用 Python。在约 600 种计算机语言中，后起的 Python 受关注程度在逐年上升。

1.2.2　Python 的版本

Python 语言目前主要有两个版本：Python 2.x 和 Python 3.x。

Python 3.0 于 2008 年发布，为了不带入过多的累赘，没有考虑向下兼容。这使得 Python 2.x 的程序一般无法在 Python 3 环境下执行。由于大量的 Python 2 的程序库的存在，所以，尽管 Python 3 发布多年，目前版本已到 3.5，但 Python 2 的使用热度仍然不减。不过毕竟不断改进是趋势，本书采用 3.x 版本。对一般的学习来说，常见的 Python 2 和 Python 3 的区别之一是输出语句。Python 2 下使用：

```
print "Hello World"
```

在屏幕上显示“Hello World”，而在 Python 3 中使用：

```
print("Hello World")
```

所以，不管是 Python 2 还是 Python 3，不管是书籍还是环境，都不太影响 Python 程序设计的学习。

1.2.3 Python 语言的实现

语言只是符号、语法、语义定义及使用规则的集合。使用这些规则编写的程序（就是 Python 源程序）并不能被计算机直接执行。解释执行 Python 源程序的程序叫做 Python 解释器。由解释器解释执行的过程就是 Python 的实现。Python 解释器有以下几种。

1. CPython

官方提供的解释器是用 C 语言实现的，所以称为 CPython。这是最常用的版本。本书使用的就是这样的解释器。

2. Jython

Jython 是使用 Java 语言实现的 Python 解释器，可以直接把 Python 代码编译成 Java 字节码执行。

3. IronPython

IronPython 是运行在微软.Net 平台上的 Python 解释器，可以直接把 Python 代码编译成.Net 的字节码。

4. PyPy

PyPy 是用 Python 语言实现的 Python 解释器，它的目标是提高执行速度。PyPy 采用 JIT 技术，对 Python 代码进行动态编译（而不是解释），所以可以显著提高 Python 代码的执行速度。

1.3 Python 编程环境

本节介绍 CPython 环境的安装和使用。

1.3.1 Python 环境的安装

1. 下载 Python 开发环境

读者可到 Python 官网下载 Python 开发环境。Python 支持多平台，用户下载适合自己平台的版本即可。Python 的 3.x 与 2.x 不太兼容。本书使用 3.x。编写本书时，Python 3 的最新版本是 3.4.2（Windows x86-64 MSI installer，64 位；Windows x86 MSI installer，32 位），3.5 版的 Alpha 版也已发布。如果读者与本书使用的版本不同，没有任何关系，这并不影响大家学习 Python 程序设计。Python 下载地址：http://www.python.org。

2. 安装

安装时采用默认安装即可。不过请注意一下安装路径，知道安装到哪个文件夹下，以后查找文件方便。Python 3.4.2 的默认安装路径是 C:\Python34。

1.3.2 Python 环境的使用

编写、编译和运行 Python 程序有以下 3 种方法。

1. 使用交互式解释器

在 Windows 开始菜单中找到安装的 Python 菜单项，选择“Python 3.4|Python3.4(command line..)”，启动 Python 的交互式解释器窗口（图 1-1）。

这是一个交互式解释执行 Python 程序的环境，其中的“>>>”常称为**提示符**，提示用户可以输入命令（或语句，或者说程序）。在提示符下输入 Python 语句，按回车键，系统就会立即执行这条语句。例如，输入 print("Hello Python")按回车键，它会在下一行显示“Hello Python”，输入“8+4”，它会在下一行显示“12”(图 1-2)。

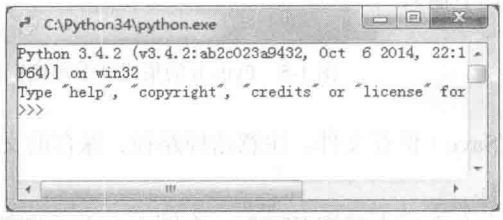

图 1-1　Python 交互式解释器

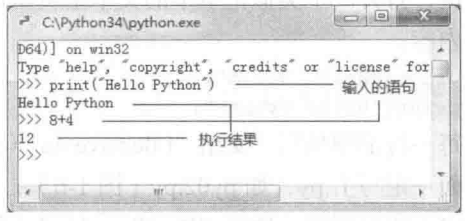

图 1-2　在交互式解释器中输入程序

使用解释器的好处就是马上能看到每一行程序的结果。但如果程序很长，这样一问一答的方式就显得啰唆。所以交互式解释器通常用于不长的程序、进行简单的计算、验证某条语句的写法、学习 Python 语法，有时也将它作为一个计算器使用。

2. 使用 Windows 命令行命令执行 Python 程序

（1）创建保存 Python 程序的文件夹。如在 C 盘根目录下创建“prog_python”文件夹。

（2）使用 Windows 自带的记事本，输入下列内容（每行顶格写）：

```
a=8
b=12
c=a+b
print(a,"+",b,"=",c)
```

（3）将写好的文件保存在 C:\prog_python 下，文件名为 py01.py。注意，保存时，选择保存类型为“所有文件（*.*）”，编码选择“UTF-8”（图 1-3）。

（4）执行 Windows 开始菜单附件中的“命令提示符”命令。打开“命令提示符”窗口，其中“>”也叫命令提示符，提示用户可以输入命令。在命令提示符下输入以下命令：

```
>c:\python34\python.exe c:\prog_python\py01.py
```

按回车键。结果显示：

```
8+12=20
```

就是刚才在记事本中输入的程序的执行结果（图 1-4）。

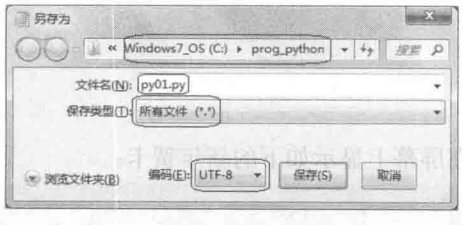

图 1-3　使用记事本保存文件

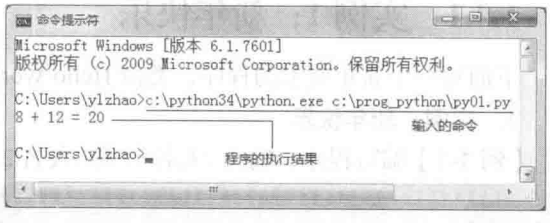

图 1-4　在 Windows 命令提示行中执行 Python 程序

输入的命令中，空格前是 Python 解释器所在的路径和解释器的可执行文件名，空格后是编写的 Python 程序的路径和文件名，就是用前面的解释器解释执行后面的程序。

3. 使用集成开发环境编写和执行 Python 程序

（1）启动软件。在 Windows 开始菜单中找到安装的 Python 菜单项，选择“Python 3.4|IDLE(Python 3.4 GUI-64 bit)”，启动 Python 的集成开发窗口（图 1-5）。

这个窗口和图 1-1 窗口基本是一样的，实际也是一个交互式的窗口，在命令提示符“>>>”下也可以输入程序，一行一行交互执行。但也有不同，因为它有菜单栏。

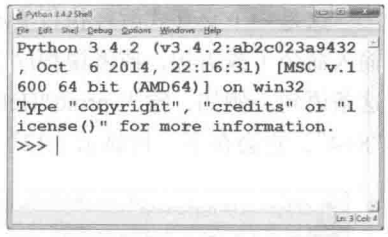

图 1-5 Python 的集成开发环境窗口

（2）编写程序。执行“File|New File”命令或直接按 Ctrl+N 组合键，打开一个类似记事本的文本编辑窗口，在其中输入下列内容：

```
print("Hello World")
print("Hello Python")
```

每一行都顶格写。使用“File|Save As”（或 File|Save）保存文件，注意选择路径，保存的文件名后面一定写上.py，如 py02.py（图 1-6）。

（3）执行程序。执行“Run|Run Module”菜单命令（或直接按 F5 键），在图 1-5 所示的窗口中显示结果（图 1-7）。

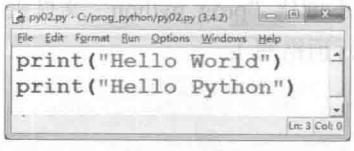

图 1-6 Python 集成环境的程序编辑窗口

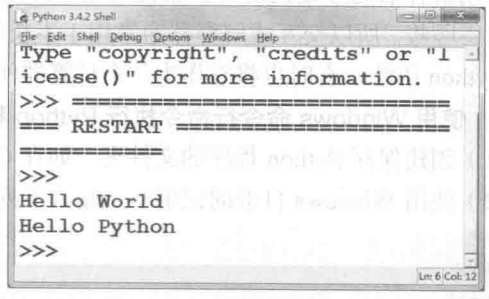

图 1-7 Python 集成环境程序的执行结果

以后讲简单的语法时会使用交互式方式。编写稍复杂的程序时会使用集成方式，先编写文件，再执行 run 命令。

1.4 Python 基础

本节将通过几个简单的小程序来学习 Python 程序的基本编写方法。

1.4.1 实例 1：新年快乐

下面写一个新年贺卡的程序，类似 Hello World。

1. 实例：新年快乐

【例 1-1】编写程序，输入<人名 1>和<人名 2>，在屏幕上显示如下的新年贺卡：

```
##########################################
<人名 1>
    Happy New Year to you.

          Yours <人名 2>
##########################################
```

【解】

（1）在集成环境下创建新的 Python 程序文件，文件名为“python0101.py”。

（2）编写文件的内容如下（注意，全部要顶格写）：

```
###################################
# 新年贺卡
# python0101.py
# 2016
###################################
name1=input("请输入收卡人:")
name2=input("请输入送卡人:")
print("############################")
print(name1)
print()
print("Happy New Year to you.")
print()
print("          Yours ",name2)
print("############################")
```

（3）运行程序，结果如图 1-8 所示。

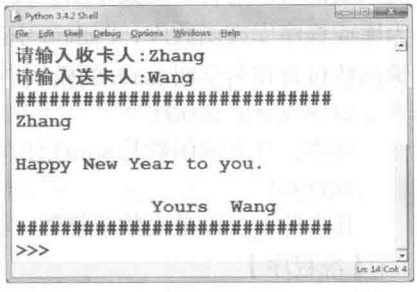

图 1-8　新年贺卡程序的运行结果

对照结果看程序，程序最前面的五行，没有对应的输出结果，它们叫**注释**，是为读懂程序而作的说明，开头是“#”；后面每行程序与结果均有一行对应，前两行是输入，输入的名字分别用 name1 和 name2 表示，后面是若干行的输出，程序的双引号中的内容是照原样输出的，name1、name2 不加引号，输出的是它代表的内容，print 后面只写一对圆括号表示输出空行。

2. 输入 input

输入的基本格式是：

```
input([promt])
```

这是一个函数，函数名是 input，小括号中是它的参数，其中 prompt 是提示信息，如本例的"请输入收卡人:"，方括号表示可选，即可以没有。这个函数接收用户从键盘上输入的一行字符，得到的是一个字符串。

所谓**字符串**就是一串字符，在 Python 中用一对单引号或一对双引号引起来表示字符串，例如，"Zhang"、'wang'等就是字符串。注意，字符串和数有本质区别，即使它们都是由数字组成的。例如，"123"，用双引号引起来，表示是字符串，只是代号，没有“数量”的概念，在计算机中保存的是 1、2、3 三个数字的 ASCII 码；而 123，没有引号，表示数量一百二十三，在计算机中存储的是 123 的二进制补码形式。

本例中输入的是人名，是代号的概念，输入后分别用 name1 和 name2 表示，“=”表示**赋值**，就是将等号右边的值赋给左边的符号。用 name1、name2 这样的符号表示数据，它们统称为**变量**。

3. 输出 print

在屏幕上显示信息，使用 print，基本格式是：

```
print([objects])
```

print 也是一个函数，objects 是参数，就是要输出的内容；可以没有，表示输出一个空行；如果有多个要输出的项目，多项用逗号隔开，如：

```
print("          Yours ",name2)
```

双引号中的内容称为**字符串常量**，照原样输出；不带双引号的 name2 是变量，输出它代表的内容。

1.4.2 实例 2：求直角三角形的斜边长度

【例 1-2】输入直角三角形的两个直角边的长度 a、b，求斜边 c 的长度，数学公式是：

$$c=\sqrt{a^2+b^2}$$

【问题分析】先说输入。input()函数得到的是字符串，是不代表数量的，即使输入的是数字。如果要使输入的数字代表实数，前面加 float,将 input()写在一对圆括号中，如：

```
a=float(input("请输入直角边 1 的长度: "))
```

float 也称为一个函数，把 input()输入的内容作为参数了，将它转换为实数，再赋值给 a，a 就代表输入的实数了。

a 的平方，只要两个 a 相乘。Python 中，乘法用“*”号表示，如 a*a 就是 a 的平方。

最后是开方运算。由于开方运算比较复杂，一般程序员不自己写开方的程序，通常设计语言的人或他人已经将一些数学函数或编程人员常用的程序写好了，放在一个文件中。这个文件常称为库或程序库或函数库，Python 中叫模块。要使用其中的功能，只要将它们导入自己的程序。数学函数包含在名字叫 math 的模块中。导入 math 模块的方法是：

```
from math import *
```

其中，开方的函数是 sqrt(),使用格式是：

```
sqrt(x)
```

其中的 x 可以是实数或整数，必须大于等于 0。

【源程序】

```
###################################
# 勾股定理求斜边长度
# python0102.py
# 2016
###################################
from math import *
a=float(input("请输入斜边 1 的长度")) #输入实数
b=float(input("请输入斜边 2 的长度")) #输入实数
c=a*a+b*b     #计算,得到的是斜边的平方
c=sqrt(c)     #开方，得到的是斜边长
print("斜边长为:",c) #显示，一项是字符串，一项是 c 表示的斜边长
```

【运行结果】

```
请输入斜边 1 的长度 3
请输入斜边 2 的长度 4
斜边长为: 5.0
```

【程序分析】本例学到了如何输入实数，学会了乘法，而加、减、除使用的符号分别是+、-和/；还学会了如何使用数学函数，其他的数学函数还有：

```
pow(x,y)      #计算 x 的 y 次方
log(x)        #以 e 为底 x 的对数
log10(x)      #以 10 为底 x 的对数
exp(x)        # e 的 x 次方
degree(x)     #弧度转角度
radians(x)    #角度转弧度
sin(x)        #x 的正弦，x 为弧度
```

```
cos(x)          #x 的余弦，x 为弧度
tan(x)          #x 的正切，x 为弧度
asin(x)         #反正弦，x∈[-1,1]
acos(x)         #反余弦，x∈[-1,1]
```

1.4.3　标识符和关键字

1. 标识符

标识符是程序中用来表示变量、函数、类、模块和其他对象的名称。如例 1-1 中的 name1、name2 都是标识符。标识符实际就是代表事物的一个名字。

Python 的标识符由字母、数字和下划线“_”组成，第一个字符不能是数字。Python 中的标识符不限长度，但区分大小写，如大写 A 和小写 a 被认为是两个不同的符号，可以分别代表不同的事物。Python 的标识符也可以使用 Unicode 字符，如汉字，但不推荐使用。

2. 关键字

有些标识符已经被 Python 语言规定了语法含义，写程序时不能再用它表示其他的事物，这些标识符称为保留字或关键字。这些关键字如下：

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

可以在交互方式中输入 help()进入帮助系统，查看关键字信息。

```
>>>help()                #进入帮助系统
help>keywords           # 查看所有的关键字列表
help> return            #查看 return 这个关键字的说明
help>quit               #退出帮助系统
```

3. 预定义标识符

Python 语言包含许多预定义的内置类、异常、函数等，如 float、input、print 等，用户应避免再使用它们为自己的数据、函数、类来命名。使用 dir(__builtins__)可以查看所有的内置异常名和函数名，如：

```
>>> dir(__builtins__)   # builtins 前后都是两个下划线
```

预定义标识符很多，常用的避免就可以了。

4. 保留标识符类

_*是特殊的标识符，在交互式执行方式中使用，代表最后的计算结果，例如：

```
>>> 100+200
300
>>> _+200
500
```

_代表上一次的计算结果 300，所以最后得到 500。

__*__，两个下划线开始和两个下划线结束，这样的名字常常是系统定义的函数的名字，如__new__()是创建新对象的函数，__init__()是构造函数等。

1.4.4 Python 的语句

程序执行中不变的数据称为**常量**，如 1、2、3、"the result is"等。

程序中可变的数据称为**变量**，一般用符号表示数据，如 a=10，以后还可以说 a=20。a 表示的数据可变，a 就是变量。

计算机语言中表示运算的符号称为**运算符**，如前面的+、-、*、/等。

常量、变量以及由运算符和变量、常量连接起来的式子称为**表达式**，如 321、"please input a number"、4+5、a+b 等是**表达式**。

1. 语句

能表达完整意义的命令形成一条**语句**，如：

```
a=5
b=3
c=a+b
```

就是三条语句，等号（=）表示“赋值”，就是将右边的表达式的值赋给左边的变量。这样的语句称为**赋值语句**。

表达式也能构成语句，如：

```
3+5
```

2. 语句的书写

Python 中，通常一个物理行就是一条语句，如：

```
a=10
b=20
```

一行也可以写多条语句，中间用分号隔开，如：

```
a=10;b=20
```

这就是两条语句，一个说明用 a 表示 10，一个说明用 b 表示 20。

一条语句也可以分多行来写,用反斜杠（\）表示**续行**。如：

```
a=(6-4)*(8-2)*(4-2)\
*(35-23)
```

和

```
a=(6-4)*(8-2)*(4-2)*(35-23)
```

是相同的。一般避免将一条语句写在多行中，那样不易理解。

如果数据是元组、列表、字典，数据元素可以分多行书写不需续行符。

Python 中的语句有简单语句和复合语句。if 语句、while 语句、for 语句、try 语句、with 语句、函数定义、类定义等是复合语句，其他一般是简单语句。

简单语句在一行中一般从第 1 列开始书写，前面不留空格。复合语句的构造块必须缩进。构造块是由简单语句和复合语句组成的语句块，它们整体缩进，之间要对齐。第 3 章会讲具体的使用。

3. 缩进

Python 采用缩进格式标记一组语句。缩进就是在一行中输入若干空格或制表符（按 Tab 键产生）然后再开始书写字符。缩进量相同的是一组语句，也称为**构造块**或**程序段**。但缩进不是随意的，后面会讲到的分支语句、循环语句、函数定义等都采用规定的缩进格式。它们的特点都是物理行的末尾有一个冒号“:”，然后随后的若干行是向右缩进并对齐的。否则不能随便缩进。如 1.3 节中写过的两个程序，都强调左边顶格对齐，就是这个原因。