计算机系列教材

董付国　编著

Python程序设计基础

清华大学出版社
北京

内 容 简 介

全书共 9 章，主要内容组织如下：第 1 章介绍 Python 基本知识与概念；第 2 章讲解列表、元组、字典、集合等序列的常用方法和基本操作；第 3 章讲解 Python 选择结构、for 循环与 while 循环、break 与 continue 语句；第 4 章讲解字符串编码格式，字符串格式化、替换、分割、连接等基本操作方法，正则表达式语法、正则表达式对象、子模式与 match 对象，以及正则表达式模块 re 的应用；第 5 章讲解函数的定义与使用，关键参数、默认值参数、长度可变参数、变量作用域以及 lambda 表达式；第 6 章讲解类的定义、类成员与实例成员、私有成员与公有成员、特殊方法与运算符重载；第 7 章讲解文件操作基本知识，文本文件内容读取与写入，二进制文件操作与对象序列化，文件复制、移动、重命名、MD5 值计算、压缩与解压缩等文件级操作以及目录操作有关知识；第 8 章讲解 Python 异常类层次结构，不同形式的异常处理结构，以及如何调试 Python 程序；第 9 章讲解如何使用 wxPython 进行 GUI 编程，主要包括窗体、按钮、文本框、单选钮、复选框等控件以及各种对话框的运用。

本书对 Python 内部工作原理进行了一定程度的剖析，对 Python 2. x 和 Python 3. x 之间的区别进行了深入对比和分析，并适当介绍 Python 程序优化和安全编程的有关知识，可以满足不同层次读者的需要。

图书在版编目(CIP)数据

Python 程序设计基础/董付国编著. —北京：清华大学出版社，2015
计算机系列教材
ISBN 978-7-302-41058-4

Ⅰ. ①P…　Ⅱ. ①董…　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(2015)第 173346 号

责任编辑：白立军　李　晔
封面设计：常雪影
责任校对：焦丽丽
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16　　**字　　数**：372 千字
版　　次：2015 年 8 月第 1 版　　**印　　次**：2015 年 8 月第1 次印刷
印　　数：1～2000
定　　价：29.00 元

产品编号：065428-01

Python 由 Guido van Rossum 于 1989 年底研制，第一个公开发行版本发行于 1991 年。Python 推出不久就迅速得到了各行业人士的青睐，经过二十多年的发展，已经渗透到计算机科学与技术、统计分析、移动终端开发、科学计算可视化、逆向工程与软件分析、图形图像处理、人工智能、游戏设计与策划、网站开发等几乎所有专业和领域。目前，Python 已经成为卡耐基-梅隆大学、麻省理工学院、加州大学伯克利分校、哈佛大学等国外很多大学计算机专业或非计算机专业的程序设计入门教学语言，国内也有不少学校的多个专业陆续开设了 Python 程序设计课程。Python 语言连续多年在 TIOBE 网站的编程语言排行榜上排名前十位，并于 2011 年 1 月被 TIOBE 网站评为 2010 年度语言。在 2014 年 12 月 IEEE Spectrum 推出的编程语言排行榜中，Python 更是取得了第 5 位的好名次。

Python 是一门免费、开源的跨平台高级动态编程语言，支持命令式编程、函数式编程，完全支持面向对象程序设计，语法简洁清晰，并且拥有大量功能丰富而强大的标准库和扩展库以及众多狂热的支持者，可以帮助各领域的科研人员或策划师甚至管理人员快速实现和验证自己的思路与创意。Python 用户可以把主要精力放在业务逻辑的设计与实现上，而不用过多考虑语言本身的细节，开发效率非常高，其精妙之处令人击节赞叹。

Python 是一门快乐的语言，学习和使用 Python 也是一个快乐的过程。与 C 语言系列和 Java 等语言相比，Python 更加容易学习和使用，但这并不意味着可以非常轻松地掌握 Python。熟练掌握和运用 Python 仍需要通过大量的练习来锻炼自己的思维和熟悉 Python 编程模式，同时还需要经常关注 Python 社区优秀的代码以及各种扩展库的动态。当然，如果能够适当了解 Python 及其扩展库的内部工作原理，对于编写正确而优雅的 Python 程序也是有很大帮助的。

Python 是一门优雅的语言。Python 语法简洁清晰，并且提供了大量的内置对象和内置函数，编程模式非常符合人类的思维方法和习惯。在有些编程语言中需要编写大量代码才能实现的功能，在 Python 中仅需要调用内置函数或内置对象的方法即可实现。如果有其他程序设计语言的基础，那么在学习和使用 Python 的时候，一定不要把其他语言的编程习惯和风格带到 Python 中来，因为这不仅可能会使得代码变得非常冗长、烦琐，还可能会严重影响代码的效率。应该尽量尝试从最自然、最简洁的角度出发去思考和解决问题，这样才能写出更加优雅、更加 Pythonic 的代码。

本书内容组织

对于 Python 程序员来说，能够熟练运用各种扩展库毫无疑问是非常重要的，使用优秀、成熟的扩展库可以帮助我们快速实现自己的业务逻辑和创意。但是也必须清楚地认识到，Python 语言基础知识和基本数据结构的熟练掌握是理解和运用其他扩展库的必备条件之一。因此，本书把重点和主要篇幅放在 Python 编程基础知识的介绍上，通过大量案例介绍 Python 在实际开发中的应用，关于不同应用领域的扩展库可以参考附录 B，并结合自己的专业领域查阅相关文档。全书共 9 章，主要内容组织如下：

第 1 章　基础知识。介绍如何选择 Python 版本，Python 对象模型，数字、字符串等基本数据类型，运算符与表达式，内置函数，基本输入输出，Python 程序文件名，扩展库的管理与使用，Python 代码编写规范，等等。

第 2 章　Python 序列。讲解序列常用方法和基本操作，成员测试运算符，切片操作，列表基本操作与常用方法，列表推导式，元组与生成器推导式，序列解包，字典、集合基本操作和常用方法，以及如何使用 Python 基本数据类型实现栈、二叉树、有向图等复杂数据结构。

第 3 章　选择与循环。讲解 Python 选择结构，for 循环与 while 循环，带有 else 子句的循环结构，break 与 continue 语句，选择结构与循环结构的综合运用。

第 4 章　字符串与正则表达式。讲解字符串编码格式，字符串格式化、替换、分割、连接等基本操作方法，正则表达式语法、正则表达式对象、子模式与 match 对象，以及 Python 正则表达式模块 re 的应用。

第 5 章　函数设计与使用。讲解函数的定义与使用，关键参数、默认值参数、长度可变参数等不同参数类型，全局变量与局部变量，参数传递时的序列解包，return 语句，lambda 表达式，等等。

第 6 章　面向对象程序设计。讲解类的定义与继承、self 与 cls 参数、类成员与实例成员、私有成员与公有成员、特殊方法与运算符重载等内容。

第 7 章　文件操作。讲解文件操作基本知识与 Python 文件对象，文本文件内容读取与写入，二进制文件操作与对象序列化，文件复制、移动、重命名、文件类型检测、MD5 值计算、压缩与解压缩等文件级操作以及目录操作有关知识。

第 8 章　异常处理结构与程序调试。讲解 Python 异常类层次结构与自定义异常类，

多种不同形式的异常处理结构，以及如何使用 IDLE 和 pdb 模块调试 Python 程序。

第 9 章　GUI 编程。讲解如何使用 wxPython 进行 GUI 编程，主要包括窗体、按钮、文本框、单选钮、复选框、组合框、列表框、树形等控件以及各种对话框的运用。

本书最大特点是信息量大、知识点紧凑、案例丰富。全书没有多余的文字和软件安装截图，充分利用宝贵的篇幅来介绍和讲解尽可能多的知识点，可以说是物超所值。本书作者具有 15 年程序设计教学经验，讲授过汇编语言、C/C++/C#、Java、PHP、Python 等多门程序设计语言，并编写过大量的应用程序。在本书内容的组织和安排上，结合了多年教学与开发过程中积累的许多案例，并巧妙地糅合进了相应的章节。

本书对 Python 内部工作原理进行了一定程度的剖析，对 Python 2.x 和 Python 3.x 之间的区别进行了深入对比和分析，并适当介绍了 Python 程序优化和安全编程的有关知识，可以满足不同层次读者的需要。

本书适用读者

本书可以作为(但不限于)：

- 数字媒体技术、软件工程、网络工程、信息安全、会计、经济、金融、心理学、统计以及其他非计算机专业本科或专科的程序设计教材。如果作为本科非计算机专业程序设计语言公共课或选修课教材，建议采用 64 学时或 48 学时边讲边练的教学模式。
- 具有一定 Python 基础的读者进阶学习资料。
- 打算利用业余时间学习一门快乐的程序设计语言并编写几个小程序来娱乐的读者首选学习资料。
- 少数对编程具有浓厚兴趣和天赋的中学生课外阅读资料。

教学资源

本书提供全套教学课件、源代码、课后习题答案与分析以及授课计划和学时分配表，配套资源可以登录清华大学出版社官方网站下载或与作者联系索取，作者 QQ 号码是 306467355，微信号是 Python_dfg，电子邮箱地址是 dongfuguo2005@126.com。

由于时间仓促，作者水平有限，书中难免出现错误，不足之处还请指正并通过作者联系方式进行反馈，作者将不定期在 QQ 空间和微信发布和更新勘误表。

感谢

首先感谢父母的养育之恩，在当年那么艰苦的条件下还坚决支持我读书，而没有让我像其他同龄的孩子一样辍学。感谢姐姐、姐夫多年来对我的爱护以及在老家对父母的照顾，感谢善良的弟弟、弟媳在老家对父母的照顾，正是有了你们，我才能在远离家乡的城市安心工作。感谢我的妻子在生活中对我的大力支持，也感谢懂事的小女儿在我工作的时候能够在旁边安静地读书而尽量不打扰我，并在定稿前和妈妈一起帮我阅读全书并检查出了几个错别字。

感谢每一位读者，感谢您在茫茫书海中选择了本书，并衷心祝愿您能够从本书中受益，学到您需要的知识！同时也期待每一位读者的热心反馈，随时欢迎您指出书中的不足！

本书的出版获 2014 年山东省普通高校应用型人才培养专业发展支持计划项目资助。我校专业共建合作伙伴——浪潮优派科技教育有限公司总裁邵长臣先生——审阅了全书，并提出了很多宝贵的意见，在此致以诚挚的谢意。本书在编写出版过程中也得到清华大学出版社的大力支持和帮助，在此表示衷心的感谢。

董付国

于山东烟台

2015 年 5 月

第1章 基础知识

Python是一门跨平台、开源、免费的解释型高级动态编程语言，同时也支持伪编译，即将Python源程序转换为字节码来优化程序和提高运行速度，并且支持使用py2exe工具将Python程序转换为扩展名为exe的可执行程序，可以在没有安装Python解释器和相关依赖包的Windows平台上运行；Python支持命令式编程、函数式编程，完全支持面向对象程序设计，语法简洁清晰，并且拥有大量的几乎支持所有领域应用开发的成熟扩展库；Python就像胶水一样，可以把多种不同语言编写的程序融合到一起实现无缝拼接，更好地发挥不同语言和工具的优势，满足不同应用领域的需求。

1.1 如何选择Python版本

众所周知，Python官方网站目前同时发行Python 2.x和Python 3.x两个不同系列的版本，并且互相之间不兼容，除了输入输出方式有所不同，很多内置函数的实现和使用方式也有较大的区别，Python 3.x对Python 2.x的标准库也进行了一定程度的重新拆分和整合。在本书开始编写的时候，最新版本分别为Python 2.7.8和Python 3.4.2，本书编写完成时最新版本分别为Python 2.7.10和Python 3.4.3，并且已发布Python 3.5.0的第三个测试版。对于很多初级用户而言，最纠结的一个问题很可能是自己到底应该选择哪个版本，是选择Python 2.x还是Python 3.x，是选择Python 2.7.x还是Python 2.6.x呢？对于Python的版本演化历史，这里不多解释，需要说明的是，并不是数字越大表示版本越新，例如Python 2.7.9就比Python 3.2.6晚几个月发行，并且Python 3.2.6比Python 3.4.1也晚几个月，类似的情况还有很多。另外，虽然同系列版本中高版本比低版本更加完善和成熟，但这并不意味着最新的才是最合适的。很多扩展库的发行总是滞后于Python发行的版本，甚至目前还有很多扩展库不支持Python 3.x。因此，在选择Python的时候，一定要先考虑清楚自己学习Python的目的是什么，打算做哪方面的开发，有哪些扩展库可用，这些扩展库最高支持哪个版本的Python。这些问题全部确定以后，再做出自己的选择，这样才能事半功倍，而不至于把太多时间浪费在Python以及各种扩展库的反复安装和卸载上。同时还应该注意，当较新的Python版本推出之后，不要急于更新和替换已安装版本，而是应该在确定自己必须使用的扩展库也推出了较新版本之后再一起进行更新。

尽管如此，以目前来看Python 3.x毕竟是大势所趋，如果你暂时还没想到要做什么行业领域的应用开发，或者仅仅是为了尝试一种新的、好玩的语言，那么请毫不犹豫地选择Python 3.x系列的最高版本(目前正式发行版最高版本是Python 3.4.3)。我们相信，越来越多的扩展库将会在短时间内推出支持Python 3.x的版本。

安装好Python以后，在“开始”菜单中选择IDLE(Python GUI)命令，即可启动

Python 解释器并可以看到当前安装的 Python 版本号，如图 1-1 和图 1-2 所示。当然，如果你喜欢，也可以启动 Python(command line)来开始美妙的 Python 之旅。在 IDLE (Python GUI)和 Python(command line)两种界面中，都以三个大于号 >>> 作为提示符，可以在提示符后面输入要执行的语句。在本书所有章节给出的示例代码中，>>> 符号都不需要输入，仅表示该代码是在交互模式下运行，而不带有该提示符的代码则表示是以脚本程序的方式运行的。本书主要使用 IDLE(Python GUI)环境来介绍 Python 程序的开发与应用。

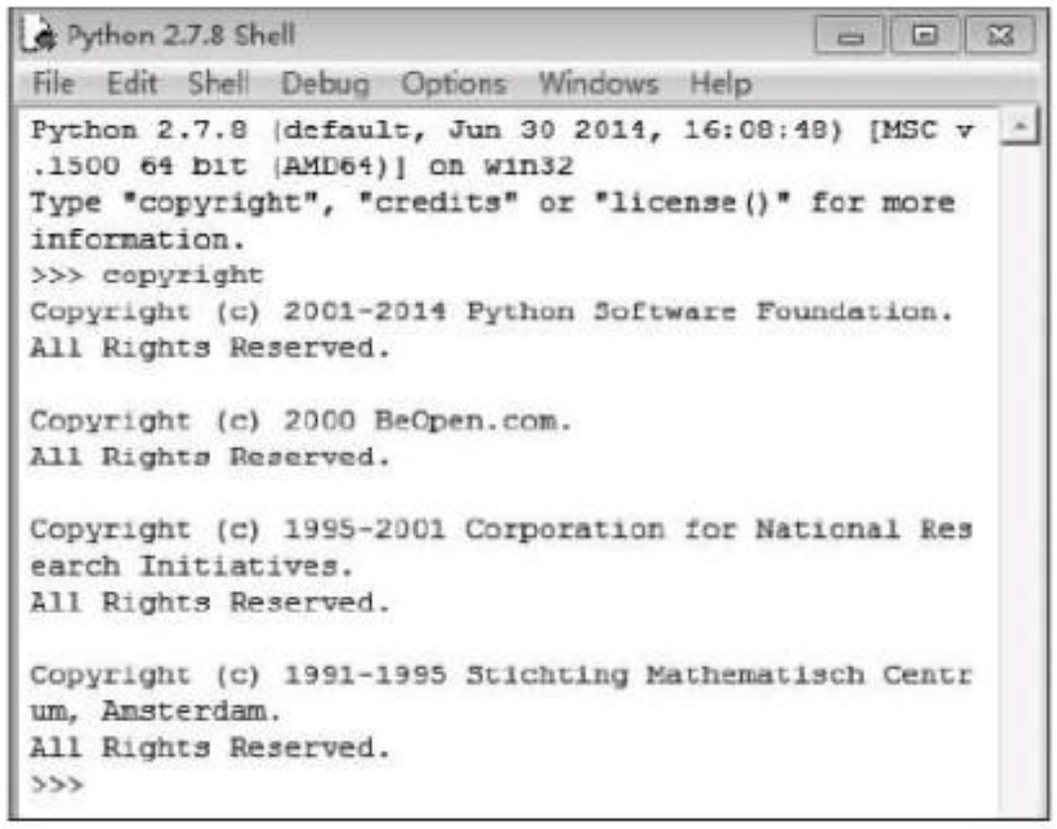

图 1-1 Python 2.7.8 主界面

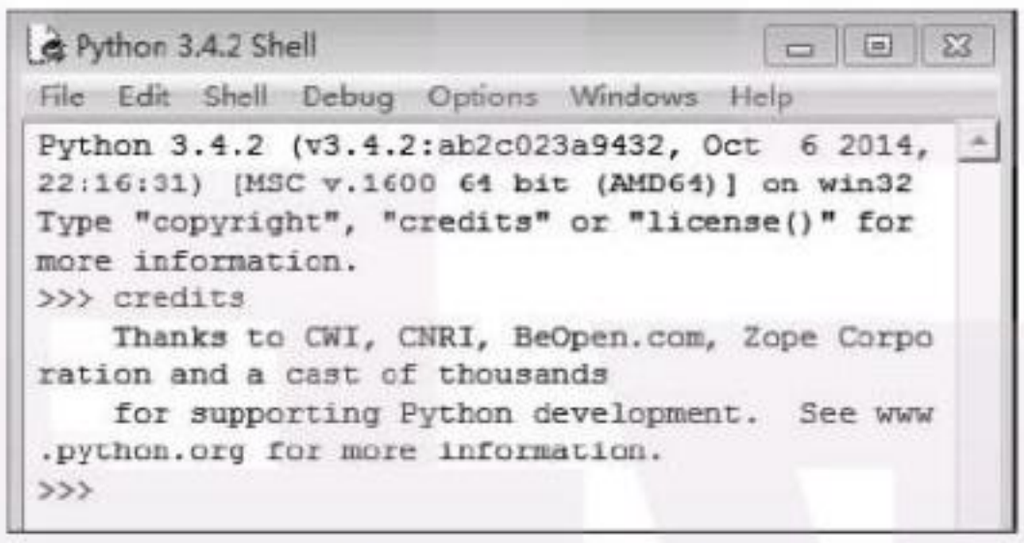

图 1-2 Python 3.4.2 主界面

除了在启动主界面上查看已安装的 Python 版本之外，还可以使用下面的命令随时进行查看。

```
>>> import sys
>>> sys.version
'3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)]'
>>> sys.winver
'2.7'
>>> sys.version_info
sys.version_info(major=2, minor=7, micro=8, releaselevel='final', serial=0)
```

有时候可能需要同时安装多个不同的版本，例如，同时安装 Python 2.7.8 和 Python 3.4.2，并根据不同的开发需求在两个版本之间进行切换。多版本并存一般不影响在 IDLE 环境中直接运行程序，只需要启动相应版本的 IDLE 即可。在命令提示符环境中运行 Python 程序时，如果无法正确运行，可以尝试在调用 Python 主程序时指定其完整路径，或者通过修改系统 Path 变量来实现不同版本之间的切换。在 Windows 7 系统下修改系统 Path 变量的步骤如下：单击“开始”菜单，右击“计算机”并执行“属性”命令，在弹出的对话框中单击“高级系统设置”选项，切换至“高级”选项卡，单击“环境变量”按钮，然后修改系统 Path 变量中的 Python 安装路径，如图 1-3 所示。

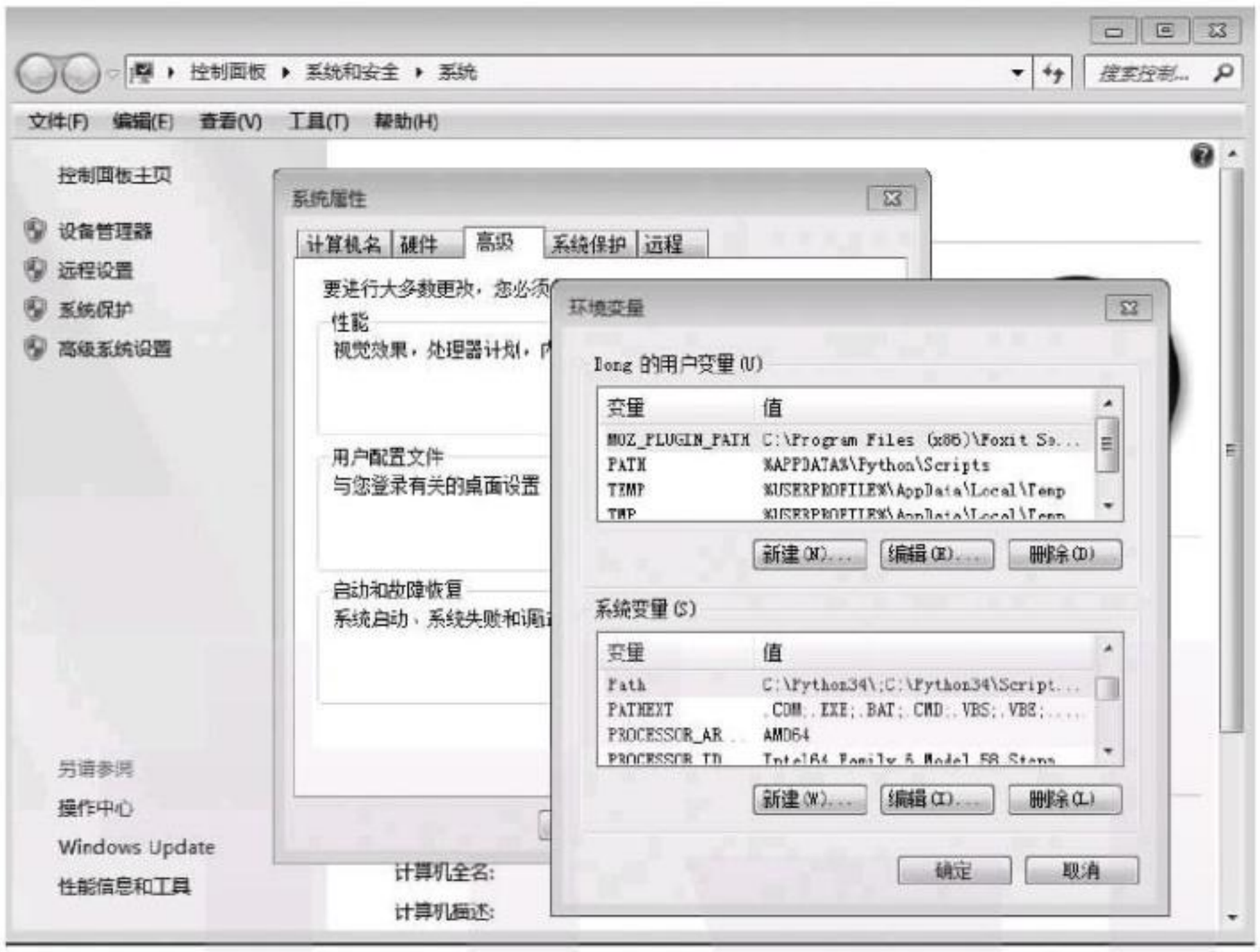

图 1-3 Windows 7 环境中系统 Path 变量修改方法

1.2 Python 安装与简单使用

为节约篇幅，这里不再详述 Python 的安装步骤，与大多数软件的安装并没什么明显的不同，打开 Python 官方主页 https://www.python.org/后选择适合自己的版本下载并安装即可。如果使用的是 Linux 系统，例如 Ubuntu，那么很可能已经预装了某个版本的 Python，请根据需要进行升级。若未经特别说明，本书所有示例均在 Windows 7 平台上使用 Python 3.4.2 和 Python 2.7.8 进行开发和演示。

安装好以后，默认以 IDLE 为开发环境，当然也可以安装使用其他的开发环境，例如 PythonWin。本书均以 IDLE 为例，如果使用交互式编程模式，那么直接在 IDLE 提示符

>>>后面输入相应的命令并回车执行即可，如果执行顺利，马上就可以看到执行结果，否则会抛出异常。

```
>>> 3+5
8
>>> import math
>>> math.sqrt(9)
3.0
>>> 3 * (2+6)
24
>>> 2/0
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    2/0
ZeroDivisionError: integer division or modulo by zero
```

一般来讲，你可能更需要编写 Python 程序来实现特定的业务逻辑，同时也方便代码的不断完善和重复利用，毕竟直接使用交互编程模式不是很方便。在 IDLE 界面中使用菜单 File→New File 命令，创建一个程序文件，输入代码并保存为文件(务必要保证扩展名为 py，如果是 GUI 程序，可以保存为.pyw 文件。如果保存为其他扩展名的文件，一般并不影响在 IDLE 中直接运行，但是在“命令提示符”环境中运行时需要显式调用 Python 主程序，并且在资源管理器中直接双击该文件时可能会无法关联 Python 主程序，从而导致无法运行)，可以使用菜单 Run→Check Module 命令来检查程序中是否存在语法错误，或者使用菜单 Run→Run Module 命令运行程序，程序运行结果将直接显示在 IDLE 交互界面上。除此之外，也可以通过在资源管理器中双击扩展名为 py 或 pyc 的 Python 程序文件直接运行；在有些情况下，可能还需要在命令提示符环境中运行 Python 程序文件。选择“开始”→“附件”→“命令提示符”命令，然后执行 Python 程序。例如，假设有程序 HelloWorld.py 内容如下。

```
def main():
    print('Hello world')
main()
```

在 IDLE 环境中运行该程序结果如图 1-4 所示。

在命令提示符环境中运行该程序的方法与结果如图 1-5 所示，该图中演示了两种执行 Python 程序的方法，虽然第二种方法看上去更简单，但是请尽量使用第一种方法来运行 Python 程序，否则可能会影响某些程序的正确运行。

```
>>> ========== RESTART ==========
=
>>>
Hello world
>>>
```

图 1-4　在 IDLE 中运行程序

```
C:\Python34>python helloworld.py
Hello world

C:\Python34>helloworld.py
Hello world

C:\Python34>
```

图 1-5　在命令提示符中运行程序

在实际开发中，如果用户能够熟练使用集成开发环境 IDLE 提供的一些快捷键，将会大幅度提高编写速度和开发效率。在 IDLE 环境下，除了撤销(Ctrl+Z)、全选(Ctrl+A)、复制(Ctrl+C)、粘贴(Ctrl+V)、剪切(Ctrl+X)等常规快捷键之外，其他比较常用的快捷键如表 1-1 所示。

表 1-1 IDLE 常用快捷键

快 捷 键	功 能 说 明
Alt+P	浏览历史命令(上一条)
Alt+N	浏览历史命令(下一条)
Ctrl+F6	重启 Shell，之前定义的对象和导入的模块全部失效
F1	打开 Python 帮助文档
Alt+/	自动补全前面曾经出现过的单词，如果之前有多个单词具有相同前缀，则在多个单词中循环选择
Ctrl+]	缩进代码块
Ctrl+[	取消代码块缩进
Alt+3	注释代码块
Alt+4	取消代码块注释

1.3 使用 pip 管理 Python 扩展库

当前，pip 已经成为管理 Python 扩展库(或模块，一般不做区分)的主流方式，使用 pip 不仅可以实时查看本机已安装的 Python 扩展库列表，还支持纯 Python 扩展库的安装、升级和卸载等操作。使用 pip 工具管理 Python 扩展库只需要在保证计算机联网的情况下输入几个命令即可完成，极大地方便了用户。

对于 Python 2.7.9 和 Python 3.4.0 之前的版本，需要首先安装 pip 命令才能使用，而在 Python 2.7.9 以及 Python 3.4.0 之后的安装包中已经集成了该命令，不需要再单独进行安装。在较早的 Python 版本中要安装 pip，首先从 https://pypi.python.org/pypi/pip 下载文件 get-pip.py，然后在命令提示符环境中执行下面的命令：

```
python get-pip.py
```

即可自动完成 pip 的安装。当然，应保证计算机处于联网状态。

安装完成以后，就可以在命令提示符环境下使用 pip 来完成扩展库的安装、升级、卸载等操作了。如果某个模块无法使用 pip 进行安装，很可能是该模块依赖于某些动态链接库文件，此时需要登录该模块官方网站下载并单独进行安装。常用 pip 命令的使用方法如表 1-2 所示。

表 1-2 常用 pip 命令使用方法

pip 命令示例	说　　明
pip install SomePackage	安装 SomePackage 模块
pip list	列出当前已安装的所有模块
pip install --upgrade SomePackage	升级 SomePackage 模块
pip uninstall SomePackage	卸载 SomePackage 模块

1.4 Python 基础知识

本节主要介绍 Python 语言基础知识，包括对象模型、变量、运算符与表达式、内置函数以及数字、字符串等基本数据类型等。

1.4.1 Python 对象模型

对象是 Python 语言中最基本的概念之一。Python 中的一切都是对象，这一点可能与某些面向对象程序设计语言略有不同。Python 中有许多内置对象可供编程者直接使用，例如数字、字符串、列表、元组、字典、集合、del 命令以及 cmp()、len()、id()、type()等大量内置函数，表 1-3 中列出了其中一部分常见的 Python 对象类型；另外，有些对象需要导入特定模块(有些模块需要单独进行安装)后才能使用，如 math 模块中的正弦函数 sin()与常量 pi，random 模块中的随机数生成函数 random()，time 模块中用于返回当前时间的函数 time()，等等。

表 1-3 Python 内置对象

对象类型	示　　例	对象类型	示　　例
数字	1234, 3.14, 3+4j	文件	f=open('data.dat', 'r')
字符串	'swfu', "I'm student", '''Python'''	集合	set('abc'), {'a', 'b', 'c'}
列表	[1, 2, 3],['a', 'b', ['c', 2]]	布尔型	True, False
字典	{1:'food',2:'taste', 3:'import'}	空类型	None
元组	(2, −5, 6)	编程单元类型	函数(使用 def 定义) 类(使用 class 定义)

1.4.2 Python 变量

在 Python 中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。例如语句

```
>>> x=3
```

创建了整型变量 x,并赋值为 3,再例如语句

```
>>> x='Hello world.'
```

创建了字符串变量 x,并赋值为'Hello world.'。这一点适用于 Python 任意类型的对象。

虽然不需要在使用之前显式地声明变量及其类型,但是 Python 仍属于强类型编程语言,Python 解释器会根据赋值或运算来自动推断变量类型。每种类型支持的运算也不完全一样,因此在使用变量时需要程序员自己确定所进行的运算是否合适,以免出现异常或者意料之外的结果。同一个运算符对于不同类型数据操作的含义和计算结果也是不一样的,后面会进行介绍。另外,Python 还是一种动态类型语言,也就是说,变量的类型是可以随时变化的,下面的代码演示了 Python 变量类型的变化。

```
>>> x=3
>>> print(type(x))
<class 'int'>
>>> x='Hello world.'
>>> print(type(x))
<class 'str'>
>>> x=[1,2,3]
>>> print(type(x))
<class 'list'>
>>> isinstance(3, int)
True
>>> isinstance('Hello world', str)
True
```

其中,内置函数 type()用来返回变量类型,内置函数 isinstance()用来测试对象是否为指定类型的实例。代码中首先创建了整型变量 x,然后又分别创建了字符串和列表类型的变量 x。当创建了字符串类型的变量 x 之后,之前创建的整型变量 x 自动失效,创建列表对象 x 之后,之前创建的字符串变量 x 自动失效。可以将该模型理解为“状态机”,在显式修改其类型或删除之前,变量将一直保持上次的类型。

在大多数情况下,如果变量出现在赋值运算符或复合赋值运算符(例如+=、*=等等)的左边则表示创建变量或修改变量的值,否则表示引用该变量的值,这一点同样适用于使用下标来访问列表、字典等可变序列以及其他自定义对象中元素的情况。例如下面的代码:

```
>>> x=3                    #创建整型变量
>>> print(x**2)
9
>>> x+=6                   #修改变量值
>>> print(x)               #读取变量值并输出显示
9
>>> x=[1,2,3]              #创建列表对象
>>> print(x)
```

```
[1, 2, 3]
>>> x[1]=5                    #修改列表元素值
>>> print(x)                  #输出显示整个列表
[1, 5, 3]
>>> print(x[2])               #输出显示列表指定元素
3
```

后面会提到，字符串和元组属于不可变序列，这意味着不能通过下标的方式来修改其中的元素值，例如下面的代码试图修改元组中元素的值时抛出异常。

```
>>> x=(1,2,3)
>>> print(x)
(1, 2, 3)
>>> x[1]=5
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in<module>
    x[1]=5
TypeError: 'tuple' object does not support item assignment
```

在 Python 中，允许多个变量指向同一个值，例如：

```
>>> x=3
>>> id(x)
1786684560
>>> y=x
>>> id(y)
1786684560
```

继续上面的示例代码，需要注意的是，当为其中一个变量修改值以后，其内存地址将会变化，但这并不影响另一个变量，例如，接着上面的代码再继续执行下面的代码：

```
>>> x+=6
>>> id(x)
1786684752
>>> y
3
>>> id(y)
1786684560
```

在这段代码中，内置函数 id()用来返回变量所指值的内存地址。可以看出，在 Python 中修改变量值的操作，并不是修改了变量的值，而是修改了变量指向的内存地址。这是因为 Python 解释器首先读取变量 x 原来的值，然后将其加 6，并将结果存放于内存中，最后将变量 x 指向该结果的内存空间，如图 1-6 所示。

Python 采用的是基于值的内存管理方式，如果为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一块内存地址，前面的几段代码也说明了这个特点。再例如下面的代码：

```
>>> x=3
>>> id(x)
10417624
>>> y=3
>>> id(y)
10417624
>>> y=5
>>> id(y)
10417600
>>> id(x)
10417624
```

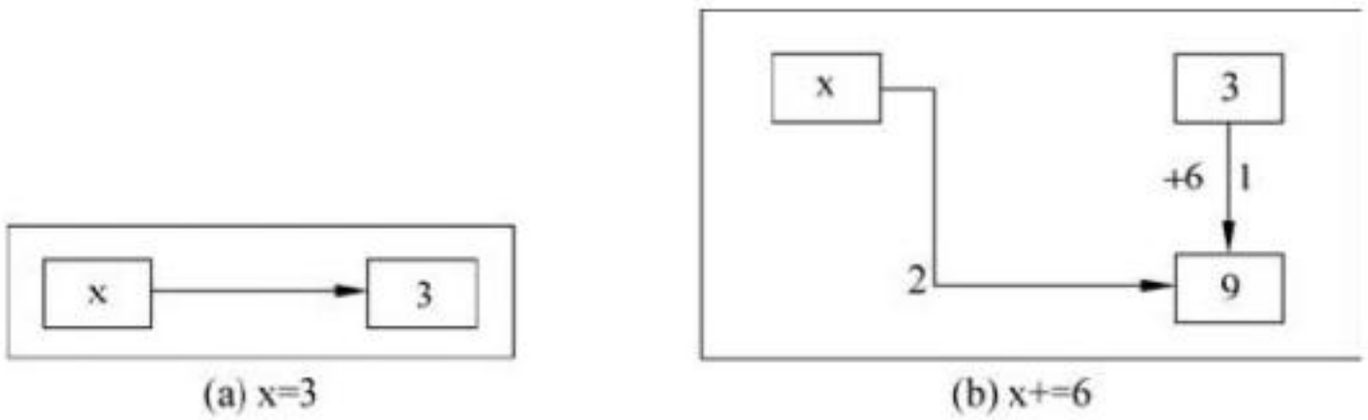

图 1-6　Python 内存管理模式

Python 具有自动内存管理功能，对于没有任何变量指向的值，Python 自动将其删除。Python 会跟踪所有的值，并自动删除不再有变量指向的值。因此，Python 程序员一般情况下不需要太多考虑内存管理的问题。尽管如此，显式使用 del 命令删除不需要的值或显式关闭不再需要访问的资源，仍是一个好的习惯，同时也是一个优秀程序员的基本素养之一。

最后，在定义变量名的时候，需要注意以下问题：

- 变量名必须以字母或下划线开头，但以下划线开头的变量在 Python 中有特殊含义，本书后面第 6 章会详细讲解；
- 变量名中不能有空格以及标点符号（括号、引号、逗号、斜线、反斜线、冒号、句号、问号等等）；
- 不能使用关键字作变量名，可以导入 keyword 模块后使用 print(keyword. kwlist) 查看所有 Python 关键字；

```
>>> import keyword
>>> keyword.kwlist
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif',
'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import',
'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try',
'while', 'with', 'yield']
>>> and=3
SyntaxError: invalid syntax
```

- 不建议使用系统内置的模块名、类型名或函数名以及已导入的模块名及其成员名作变量名，这将会改变其类型和含义，可以通过 dir(__builtins__)查看所有内置模块、类型和函数；
- 变量名区分英文字母的大小写，例如 student 和 Student 是不同的变量。

1.4.3 数字

数字属于 Python 不可变对象，1.4.2 节的例子已经说明了这一点，即修改整型变量值的时候并不是真的修改变量的值，而是先把值存放到内存中然后修改变量使其指向了新的内存地址，浮点数、复数等数字类型以及其他类型的变量具有同样的特点。列表、字典等可变类型对象的情况稍微复杂一些，将在第 2 章中详细介绍。

在 Python 中，数字类型变量可以表示任意大的数值。

```
>>> a=99999999999999999999999999999
>>> a * a
999999999999999999999999999998000000000000000000000000000001L
>>> a**3
999999999999999999999999999997000000000000000000000000029999999999
99999999999999999999L
```

如果你愿意，完全可以把 IDLE 当做计算器来使用，IDLE 可以实现复杂的数学运算。

```
>>> 3 * (2+5)/3.0
7.0
>>> import math                    #math 是 Python 标准库，其中包含大量用于数学计算的函数
>>> math.sqrt(3**2+4**2)          #平方根
5.0
```

Python 数值类型主要有整数、浮点数和复数。整数类型主要有：

- 十进制整数，如 0、－1、9、123。
- 十六进制整数，使用 16 个数字 0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f 来表示整数，必须以 0x 开头，如 0x10、0xfa、0xabcdef。
- 八进制整数，使用 8 个数字 0、1、2、3、4、5、6、7 来表示整数，必须以 0o 开头，如 0o35、0o11。
- 二进制整数，使用 2 个数字 0、1 来表示整数，必须以 0b 开头，如 0b101、0b100。

浮点数也称小数，例如，.3、15.0、0.37、－11.2、1.2e2、314.15e－2，都是合法的浮点数。

Python 中的复数与数学中复数的形式完全一致，都是由实部和虚部构成，并且使用 j 或 J 来表示虚部。

```
>>> a=3+4j
>>> b=5+6j
>>> c=a+b
```

```
>>> c
(8+10j)
>>> c.real                    #查看复数实部
8.0
>>> c.imag                    #查看复数虚部
10.0
>>> a.conjugate()             #返回共轭复数
(3-4j)
>>> a*b                       #复数乘法
(-9+38j)
>>> a/b                       #复数除法
(0.6393442622950819+0.03278688524590165j)
```

1.4.4 字符串

在Python中，字符串属于不可变序列，一般使用单引号、双引号或三引号进行界定，并且单引号、双引号、三单引号、三双引号还可以互相嵌套，用来表示复杂字符串。例如

```
'abc'、'123'、'中国'、"Python"、'''Tom said,"Let's go"'''
```

都是合法字符串，空字符串表示为""或""或''''，即一对不包含任何内容的任意字符串界定符。特别地，一对三单引号或三双引号表示的字符串支持换行，支持排版格式较为复杂的字符串，也可以在程序中表示较长的注释，在第4章和第5章中将分别进行介绍。

由于字符串类型应用非常广泛，其支持的操作也较多，这里先简单介绍一下，第4章再结合正则表达式全面展开进行详细讲解。

字符串支持使用+运算符进行合并以生成新字符串。

```
>>> a='abc'+'123'
>>> a
'abc123'
```

可以对字符串进行格式化，把其他类型对象按格式要求转换为字符串，并返回结果字符串，例如下面的代码：

```
>>> a=3.6674
>>> '%7.3f' %a
'  3.667'
>>> "%d:%c"%(65,65)
'65:A'
>>> """My name is %s, and my age is %d"""%('Dong Fuguo',38)
'My name is Dong Fuguo, and my age is 38'
```

Python支持转义字符，常用的转义字符如表1-4所示。

表 1-4 转义字符

转义字符	含　义	转义字符	含　义
\n	换行符	\"	双引号
\t	制表符	\\	一个\
\r	回车	\ddd	3 位八进制数对应的字符
\'	单引号	\xhh	2 位十六进制数对应的字符

需要特别说明的是，字符串界定符前面加字母 r 或 R 表示原始字符串，其中的特殊字符不进行转义，但字符串的最后一个字符不能是\符号。原始字符串主要用于正则表达式，也可以用来简化文件路径或 url 的输入，请参考第 4 章的内容。

1.4.5 运算符与表达式

与其他语言一样，Python 支持大多数算术运算符、关系运算符、逻辑运算符以及位运算符，并遵循与大多数语言一样的运算符优先级。除此之外，还有一些运算符是 Python 特有的，例如成员测试运算符、集合运算符、同一性测试运算符等等。另外，Python 很多运算符具有多种不同的含义，作用于不同类型操作数的含义并不相同，非常灵活。常用运算符如表 1-5 所示。

表 1-5 Python 运算符

运算符示例	功 能 说 明
x+y	算术加法，列表、元组、字符串合并
x-y	算术减法，集合差集
x * y	乘法，序列重复
x/y	除法（在 Python 3. x 中叫做真除法）
x//y	求整商
-x	相反数
x%y	余数（对实数也可以进行余数运算），字符串格式化
x**y	幂运算
x<y;x<=y;x>y;x>=y	大小比较（可以连用），集合的包含关系比较
x==y;x!=y	相等（值）比较，不等（值）比较
x or y	逻辑或（只有 x 为假才会计算 y）
x and y	逻辑与（只有 x 为真才会计算 y）
not x	逻辑非
x in y;x not in y	成员测试运算符
x is y;x is not y	对象实体同一性测试（地址）
|、^、&、<<、>>、~	位运算符
&、|、^	集合交集、并集、对称差集

需要说明的是，Python 中的除法有两种：/和//分别表示除法和整除运算，并且 Python 2.x 和 Python 3.x 对/运算符的解释也略有区别。Python 2.x 将/解释为普通除法，而 Python 3.x 将其解释为真除法。例如，在 Python 3.4.2 中运算结果如下：

```
>>> 3/5
0.6
>>> 3//5
0
>>> 3.0/5
0.6
>>> 3.0//5
0.0
>>> 13//10
1
>>> -13//10
-2
```

而上面的表达式在 Python 2.7.8 中运算结果如下：

```
>>> 3/5
0
>>> 3//5
0
>>> 3.0/5
0.6
>>> 3.0//5
0.0
>>> 13//10
1
>>> -13//10
-2
```

另外一个需要说明的，也是与其他有些语言略有不同的运算符是%。在 Python 中，除去前面已经介绍过的字符串格式化用法之外，该运算符还可以对整数和浮点数计算余数。但是由于浮点数的精确度影响，计算结果可能略有误差。

```
>>> 3.1%2
1.1
>>> 6.3%2.1
2.0999999999999996
>>> 6%2
0
>>> 6.0%2
0.0
>>> 6.0%2.0
0.0
```

```
>>> 5.7%4.8
0.9000000000000004
```

如前所述，Python 中很多运算符有多重含义，在程序中运算符的具体含义取决于操作数的类型，将在第 2 章中根据内容组织的需要陆续进行展开。例如 * 运算符就是 Python 运算符中比较特殊的一个，它不仅可以用于数值乘法，还可以用于列表、字符串、元组等类型，当列表、字符串或元组等类型变量与整数进行 * 运算时，表示对内容进行重复并返回重复后的新对象。

```
>>> 3 * 2                    #整数相乘
6
>>> 2.0 * 3                  #浮点数与整数相乘
6.0
>>> (3+4j) * 2               #复数与整数相乘
(6+8j)
>>> (3+4j) * (3-4j)          #复数与复数相乘
(25+0j)
>>> '1' * 5                  #字符串重复
'11111'
>>> "a" * 10                 #字符串重复
'aaaaaaaaaa'
>>> [1,2,3] * 3              #列表重复
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> (1,2,3) * 3              #元组重复
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>> 3 * 'a'                  #字符串重复
'aaa'
```

在 Python 中，单个任何类型的对象或常数属于合法表达式，使用表 1-5 中运算符连接的变量和常量以及函数调用的任意组合也属于合法的表达式。

```
>>> a=[1,2,3]
>>> b=[4,5,6]
>>> c=a+b
>>> c
[1, 2, 3, 4, 5, 6]
>>> d=map(str, c)
>>> d
['1', '2', '3', '4', '5', '6']
>>> import math
>>> map(math.sin, c)
[0.8414709848078965, 0.9092974268256817, 0.1411200080598672, -0.7568024953079282,
-0.9589242746631385, -0.27941549819892586]
>>> 'Hello'+' '+'world'
'Hello world'
```

```
>>> 'welcome ' * 3
'welcome welcome welcome '
>>> ('welcome,' * 3).rstrip(',')+'!'
'welcome,welcome,welcome!'
```

需要注意的是，在 Python 中逗号“,”并不是运算符，而只是一个普通分隔符，例如下面的代码：

```
>>> 'a' in 'b', 'a'
(False, 'a')
>>> 'a' in ('b', 'a')
True
>>> x=3, 5
>>> x
(3, 5)
>>> 3==3, 5
(True, 5)
>>> x=3+5, 7
>>> x
(8, 7)
```

1.4.6 常用内置函数

内置函数是指不需要导入任何模块即可直接使用的函数，例如，在 1.4.5 节最后的例子中用到的 map()函数即属于 Python 内置函数，因此不需要导入任何模块就可以直接使用，该函数在本书后面会有讲解，当然你也可以直接跳至第 5 章进行阅读，或者使用 help(map)来查看该函数帮助文档进行学习。

执行下面的命令可以列出所有内置函数和内置对象：

```
>>> dir(__builtins__)
```

常用的内置函数及其功能简要说明如表 1-6 所示。

表 1-6 Python 常用内置函数

函　数	功能简要说明
abs(x)	返回数字 x 的绝对值
all(iterable)	如果对于可迭代对象中所有元素 x 都有 bool(x)为 True，则返回 True。对于空的可迭代对象也返回 True
any(iterable)	只要可迭代对象中存在元素 x 使得 bool(x)为 True，则返回 True。对于空的可迭代对象，返回 False
bin(x)	把数字 x 转换为二进制串

续表

函　　数	功能简要说明
callable(object)	测试对象是否可调用。类和函数是可调用的,包含__call__()方法的类的对象也是可调用的
chr(x)	返回 ASCII 编码为 x 的字符
cmp(x,y)	比较大小,如果 x<y 则返回负数;如果 x==y,则返回 0;如果 x>y 则返回正数。Python 3.x 不再支持该函数
dir()	返回指定对象的成员列表
eval(s[,globals[,locals]])	计算字符串中表达式的值并返回
filter(function or None,sequence)	返回序列中使得函数值为 True 的那些元素,如果函数为 None 则返回那些值等价于 True 的元素。如果序列为元组或字符串则返回相同类型结果,其他则返回列表
float(x)	把数字或字符串 x 转换为浮点数并返回
help(obj)	返回对象 obj 的帮助信息
hex(x)	把数字 x 转换为十六进制串
id(obj)	返回对象 obj 的标识(地址)
input([提示内容字符串])	接收键盘输入的内容,返回字符串。Python 2.x 和 Python 3.x 对该函数的解释不完全一样,详见 1.4.8 节
int(x[,d])	返回数字的整数部分,或把 d 进制的字符串 x 转换为十进制并返回,d 默认为十进制
isinstance(object,class-or-type-or-tuple)	测试对象是否属于指定类型的实例
len(obj)	返回对象 obj 包含的元素个数,适用于列表、元组、集合、字典、字符串等类型的对象
list([x])、set([x])、tuple([x])、dict([x])	把对象转换为列表、集合、元组或字典并返回,或生成空列表、空集合、空元组、空字典
map(函数,序列)	将单参数函数映射至序列中每个元素,返回结果列表
max(x)、min(x)、sum(x)	返回序列中的最大值、最小值或数值元素之和
oct(x)	把数字 x 转换为八进制串
open(name[,mode[,buffering]])	以指定模式打开文件并返回文件对象
ord(s)	返回 1 个字符 s 的编码
pow(x,y)	返回 x 的 y 次方,等价于 x**y
range([start,] end [,step])	返回一个等差数列(Python 3.x 中返回一个 range 对象),不包括终值
reduce(函数,序列)	将接收 2 个参数的函数以累积的方式从左到右依次应用至序列中每个元素,最终返回单个值作为结果

本书试读到此结束啦！

[Python程序设计基础](#)

作者：董付国, 编著

出版社：清华大学出版社
