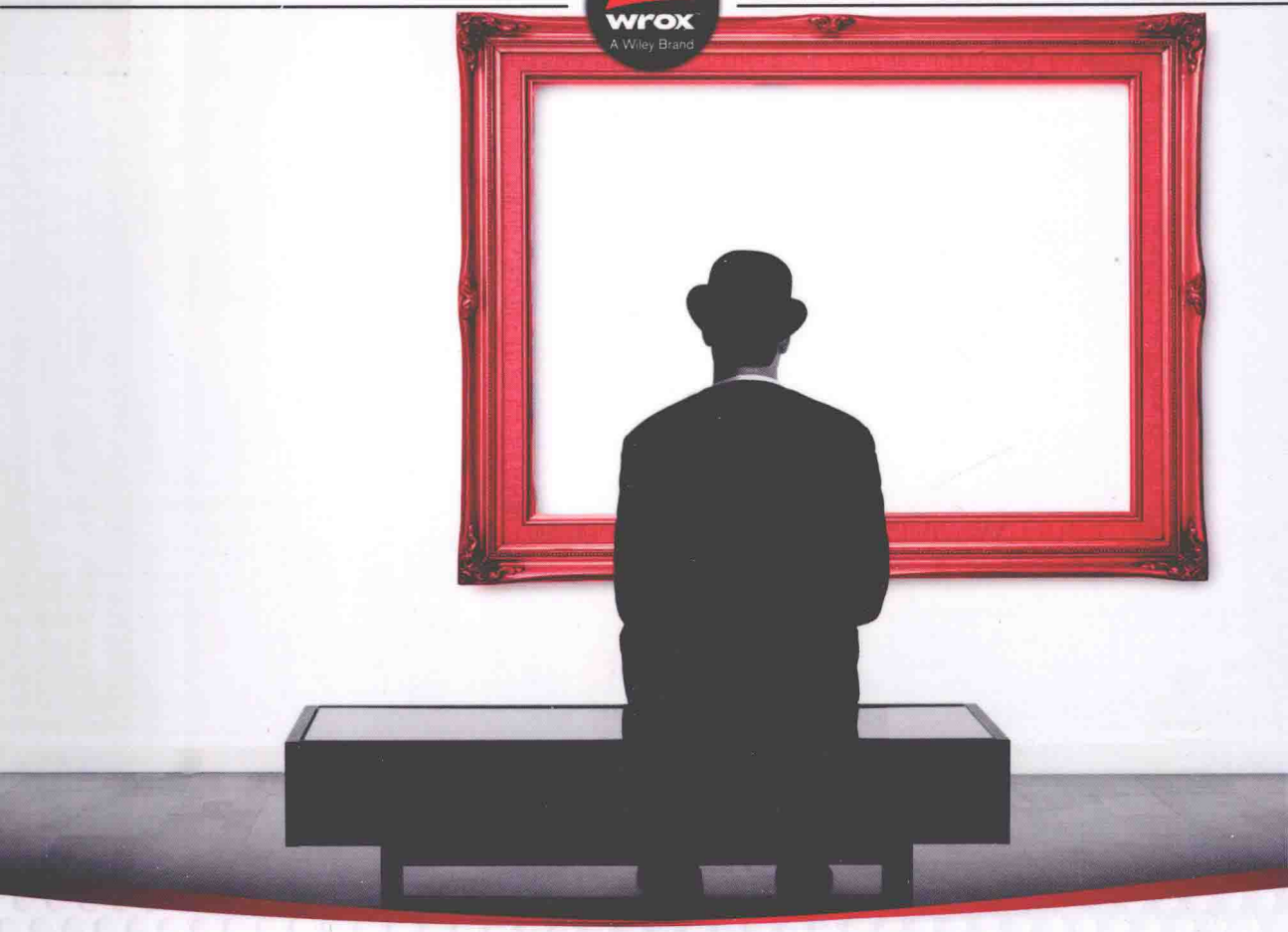

Python Projects

Python 项目开发实战

[美] Laura Cassell
Alan Gauld 著
高弘扬 卫莹 译

清华大学出版社

Python 项目开发实战

[美] Laura Cassell
Alan Gauld 著
高弘扬 卫莹 译

清华大学出版社
北 京

Laura Cassell, Alan Gauld
Python Projects
EISBN：978-1-118-90866-2

图书在版编目(CIP)数据

Python 项目开发实战 / (美) 卡塞尔 (Cassell，L.)，(美) 高尔德 (Gauld，A.) 著；高弘扬，卫莹 译.
—北京：清华大学出版社，2015
书名原文：Python Projects
ISBN 978-7-302-41587-9

Ⅰ.①P… Ⅱ.①卡…②高…③高…④卫… Ⅲ.①软件工具—程序设计 Ⅳ.①TP311.56

中国版本图书馆 CIP 数据核字(2015)第 225360 号

责任编辑：王　军　于　平
装帧设计：牛静敏
责任校对：成凤进
责任印制：何　芊

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
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：21.25　　字　　数：517 千字
版　　次：2015 年 10 月第 1 版　　印　　次：2015 年 10 月第 1 次印刷
印　　数：1～3500
定　　价：59.80 元

产品编号：062782-01

译 者 序

Python 是一门脚本语言。在面世之初，Python 并没有得到很多人的青睐。但是自从 2004 年以来，越来越多的公司发现了 Python 的强大之处。Python 的使用率也呈现线性增长。它甚至在 2011 年被评为年度编程语言。Python 语言简洁、易读、可扩展，可以用于很多方面。一些国外的科研机构采用 Python 作为开发语言。一些知名大学甚至开设了教授 Python 的教程，例如卡耐基梅隆大学和麻省理工学院。除了 Python 语言本身的特性之外，各种各样的扩展库也是 Python 的强大之处。这些丰富的扩展库可以帮助 Python 在各个领域都大显身手。Pypi 的出现更是为获得这些扩展库提供了极大的便利。

Python 的语言特性决定了它非常容易上手。市面上也已经有很多 Python 的入门书籍。这些入门书籍从语言本身出发，会详细介绍 Python 的一些重要特性。通过这些入门书籍，读者可以足够了解 Python 的基础。一些书籍会涉及 Python 的高级特性或一些额外的扩展包。然而，这些书并没有告诉读者：使用 Python 可以在实际中做什么。本书的最主要目的就是告诉 Python 初学者：Python 在真实世界中有哪些用途。然而在阅读本书之前，读者最好已经对 Python 有了初步的了解。本书循序渐进，逐步地为读者揭开 Python 在实际世界中的用途。

本书的第 1 章简要回顾了 Python 的核心知识。通过第 1 章，读者可以巩固 Python 的基础知识。接下来的一章主要介绍了 Python 的最重要的应用之一：脚本语言。Python 可以被用作脚本来协调操作系统上不同的应用并完成一个任务。第 3 章讲述了如何使用 Python 管理数据。第 4 章描述了如何使用 Python 创建桌面应用。在第 5 章中，Python 被用来开发网络应用。第 6 章讲述了如何在更大的项目中使用 Python。第 7 章探索了 Python 的前沿技术。

为了让读者能够更加深刻地理解书中所涉及的知识，作者在本书中插入了大量的示例。读者在使用本书时，应该动手试试这些示例。纸上学来终觉浅，要知此事须躬行。如果有条件，推荐读者去阅读感兴趣模块的源代码，这样可以快速地提高 Python 编程能力。

在此，非常感谢清华大学出版社的编辑们。没有她们的帮助，本书不可能顺利付梓。同时也感谢本书的其他译者。本书的全部章节由高弘扬、卫莹翻译，参与翻译的还有卫玲、刘扬、于洋、金宏兵、戴云、张国健、王辉、袁传清、张秀丽、高庆宝、陈守范。

在翻译中，译者本着信达雅的原则，反复揣摩唯恐不能完全表述作者的原文。然而介于译者能力有限，如有纰漏之处，在此向各位读者道歉，还望海涵，不吝指正。

译者

作 者 简 介

Laura Cassell 从 1997 年就开始接触网络编程。在 21 世纪初，她自学了 Perl。那时，她发现编程资料急需改善和补充，但是教授编程的门槛却非常高。因此，她开始学习编程，这样就可以向更多人教授编程。

在乔治亚州的亚特兰大，Laura 创建了 PyLadies Atlanta，并开始为 Big Nerd Ranch 教授 Python 和 JavaScript。从此，她开始从事工程管理，并且现在定居波兰。在那里，她管理一组 Python 支持者，为 New Relic 有限公司做软件分析。同时，她在时间允许时会做义务教学和演讲。

Alan Gauld 是一位拥有超过 40 年 IT 从业经验的企业架构师。他主要的工作领域是电子通信和用户服务。他使用过的编程语言超过 20 种，创建的产品包罗万象，从大型机计费系统到嵌入式微控制器。在过去 15 年里，他主要的编程语言是 Python。他撰写过一本 Python 入门书籍，也是 python-tutor 邮件列表的联合负责人。

除了编程，他喜欢登山、徒步旅行和滑雪。同时，他也是一位摄影师、艺术家和声乐爱好者。他和妻子 Heather 生活在苏格兰。

技术编辑简介

Alex Bradbury 是一位编译器黑客、Linux 极客和自由软件爱好者，他是 Raspberry Pi 项目的长期贡献者，同时也是 *Learning Python with Raspberry Pi* 的作者之一。现在，他是剑桥大学计算机实验室的一名研究员，也是 lowRISC 项目的创建者之一。lowRISC 是一个用来生产完全开源的单晶片系统(System-on-Chip，SoC)的非营利性项目。

Todd Shandelman 非常怀念在 IBM System/370 主机的打孔卡上用汇编语言编程的日子。多年来，他使用过多种软件技术(C、C++和 Perl 等)。现在，Todd 把他最多的精力都放在 Linux 命令行的 Python 2.x 和 Python 3.x 编程上。在空闲时间，他是一个俄语和希伯来语的专业译者，擅长外文排版设计，并且对处理 Unicode 和 UTF-8 很有心得。Todd 在纽约州立大学企业管理系获得理学学士学位。他和妻子、儿子一起生活在德克萨斯州休斯顿市。

致　　谢

非常感谢 Alan Gauld，他对于本书做出了巨大贡献。感谢 Mary James 和 Jennifer Lynn，他们对本书提出了一些宝贵意见。因为有你们，本书质量才得以精益求精。

同样也感谢 Python 社区。迄今为止，你是我见到的最热情的社区。在社区中，人们可以感受到热情，可以接触到从专家到新手的每一个人。继续加油，感谢你让我成为这么优秀社区中的一员。

——Laura Cassell

非常感谢 Laura Cassell 推动了这个项目。感谢 Jennifer Lynn 一直激励着我们。感谢 Python 社区在过去 15 年的支持。

——Alan Gauld

前　言

在某年的一次会议后，发到 PyLadies 组织者邮件列表的一封邮件问："有人有兴趣写一本 Python 方面的书吗？"当时，我已经考虑撰写一本编程书很久了。在多年教课以及在 PyLadies 和其他编程聚会做指导之后，我意识到需要一本新的、特定类型的编程书。但是我并没有马上回复那封邮件。我知道写一本书是一项巨大的工程(确实是！)，会耗费我大量的时间和精力。我在周末和节假日也要工作(是的，我又对了！)。我也知道我有一份教授编程课程的全职工作，同时也是乔治亚州亚特兰大本地 PyLadies 的主要组织者。我的孩子也会开始问我："这个周末你要写书吗？"

以上内容都是真实的(实际上比我最初的想法还要多)，但是我知道书籍很重要。非常多的学生会在课后问我："现在我已经了解 Python 基础知识了，我能做些什么呢？"我的答案总是："你可以参与一些开源项目！"或者"参加一些 Python 高级课程"。但是这些答案既不能让他们满意也不能让我满意。正确答案应该是："你必须真正地寻找一些事情来做：解决一个问题或实现一个需求"。这是因为，真正理解编程和一门编程语言的唯一方式就是用这门语言去解决问题。

然而，另一个问题又出现了："我没有真正需要解决的问题"。所以，虽然我可以让我的学生去了解开源项目，而这实际上也非常有帮助，但是如果不了解技术，他们可能会迷失，甚至放弃。这样社区就又失去了一位可能带来有趣东西的程序员。所以，在与家人和朋友做大量交流之后，我意识到需要撰写这本书。

本书目的

多年以来，一直有人问我们，"我在熟悉 Python 基础后能做些什么？"，"我能学到什么？"，"我该何去何从？"。解决以上问题就是撰写本书的目的所在。

对于编程书籍来说，很多人都曾经历过的一个长期问题是它们都是从语言基础到深层概念。这些概念只有拥有计算机科学学位的人才能理解。但这并不酷！编程的大门应该向任何有兴趣的人敞开。我们都应该致力于降低编程的门槛。我们觉得 Python 做到了这一点，但是我们需要更进一步，并且开始理解人们是如何学习抽象想法和概念的，帮助他们学习编程。

可以将编程想象成学习如何盖房子，只知道需要木料，但是不知道如何用木料盖房子。你仍然需要理解结构工程、电气、水管设施、通风、高压交流电(High Voltage Alternating Current，HVAC)等。编程也是一样。语言只解释了盖房子需要木料。还有很多与木料相关的东西。我们希望帮助你了解这些概念。

本书读者对象

本书并不适合想要学习 Python 的初学者。实际上，作为本书的读者，你需要拥有一些 Python 编程基础。这意味着你已经学过一些教程。你也应该理解空格在 Python 中的作用、列表被包含在方括号([])中、但字典被包含在花括号({})中。本书适用于那些初学者，但应该已经学过一两个教程。这些人理解 Python 基础，但对 Python 可以实现的功能很感兴趣。

俗语说的好，需要是发明之母。在你学习编程时，这句话非常正确。如果你需要软件来执行特定函数或任务，那么围绕着需求学习一门语言就很容易。你有需求，语言就会帮助你，学习语言，解决问题，你学到了知识，并且立即付诸实践。这太棒了！然而，如果你觉得编程很有趣，但却没有需求，不知道要实现什么，结果会怎样呢？这就是本书要解决的问题。

本书会帮助你学习大部分人不会对初学者讲述的 Python 部分。书中涉及的大部分工具和技术只有在实践中才会遇到。然而，对于没有特定问题需要解决的新手程序员来说，学习这些工具可能比较困难。在很长一段时间里，没有人想要向开发者介绍这些工具，因为它们真的很常用。我们希望可以带你领略 Python 的能力和辉煌。

你将学习如何编写一个 Web 应用，以及如何使用 Python 库与数据库通信。如果你是一名系统管理员，还可以学到可以加速工作流的系统工具。我们将简要介绍诸如安全和最佳实践的话题，概述如何使用 Python 库创建图形用户界面(GUI)。还将介绍如何编写和使用应用编程接口(Application Programming Interfaces，API)，以及其他对 Python 程序员有用的话题。

本书内容简介

我们希望带你简要了解一下 Python 的基础知识，将向你介绍那些只有在解决问题时才会理解的概念。尽管我们不能在这里呈现所有将来可能需要解决的问题，但是我们希望可以为 Python 新手展示 Python 语言的强大特性和可以使用的包和技术。

首先，提供一个 Python 的速成课程，以防你已经忘记了所有东西。我们将复习基础知识，然后你可以决定是否完整阅读该章。接下来，将从脚本语言的角度来重新审视 Python。通过尝试使用 Python 编写一些小脚本来访问你的系统。这可以展示 Python 让你所拥有的非常基本的能力。之后会讨论数据，这其实就是编程的一切——操纵数据。你会使用 Python 提供的标准库来完成一些示例。我们甚至会讨论数据库，这样就可以对它有一个基本了解。我们想让你了解并接触系统中可能会接触到的每个部分。

在前三章之后，将介绍桌面应用。尽管这些在 Python 中并不常用，但也是语言的一个特性。在你的整个 Python 程序员的生涯中它都非常有用。接下来将介绍 Internet。这时，Python 会充当数据通信工具。你将学习有关 HTTP 和 Web 的所有知识，以及网站在

底层的工作方式，甚至会动手编写和使用 API。很多新手程序员对 API 都很迷惑。我们希望在本章揭开它神秘的面纱。

在最后几章，将介绍 Python 中一些更高级的话题，例如，如何在更大的项目中使用 Python、调试代码、创建测试模块、错误处理，以及创建自定义的异常和异常处理器。你在使用本书时、在将来查阅本书时、在使用 Python 编程时，都可以使用索引快速找到你想要的内容。

本书信息量很大，其中包含大量的工具和想法，可帮助你开始使用 Python。我们希望你可以自己动手尝试，并且花时间在你感兴趣的概念和想法上做更多功课。在本书中，已经包含了大量实践练习来帮助你尝试新概念。在大多数章节中，还包含了一些挑战性问题，以帮助你巩固新知识。

使用本书须知

为了更好地使用本书，建议你所使用的现代计算机能够运行 Python 3.3 或更新版本，有一个能够舒适使用的好的文本编辑器，具有 Internet 连接(本书一些部分会使用)，以及足够的耐心和求知欲。我们也建议你使用 Internet 搜索任何遇到的问题。专业程序员实际上并不是什么都会。他们通常只知道那些每天需要处理的问题，他们的大部分时间都花在搜索和追踪问题发生的原因上。不要觉得依赖 Google 解决问题是很让人沮丧的。有时，使用 Google 搜索问题的能力和你的编程能力是同样重要的。

在使用本书的示例和项目时，你可能需要源代码。示例的源文件可以通过 Wrox 网站 www.wrox.com/go/pythonprojects 和 http://www.tupwk.com.cn/downpage 下载。

源代码

在完成本书示例时，可以选择手动输入所有代码，也可以使用本书附带的源代码。本书中用到的所有源代码都可以从 www.wrox.com 下载。对于本书，源代码下载的具体位置在 www.wrox.com/go/pythonprojects 的 Download Code 选项卡下。

可以在 www.wrox.com 搜索本书的 ISBN(本书的 ISBN 是 978-1-118-90866-2)来寻找代码。www.wrox.com/dynamic/books/download.aspx 上列出了当前所有 Wrox 书籍的完整代码下载列表。

www.wrox.com 上的大部分代码是使用.ZIP、.RAR 或适用于当前平台的类似压缩格式压缩的。下载之后，使用合适的解压缩工具解压即可。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是错误总是难免的，如果你在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常

感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

要在网站上找到本书的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/ booklist.shtml。

如果在 Book Errata 页面上没有看到你找出的错误，请进入 www.wrox.com/contact/techsupport.shtml，填写表单，发电子邮件，我们就会检查你的信息，如果是正确的，就在本书的勘误表中粘贴一个消息，我们将在本书的后续版本中采用。

p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可以在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新帖子时，会给你发送你选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。

在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。要加入这个论坛，必须执行下面的步骤：

(1) 进入 http://p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

(4) 然后就可以收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

提示：不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的帖子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

目　录

第 1 章

Python 核心知识回顾

本章主要内容：

- Python 语言的基本特性
- 如何使用 Python 模块机制
- 如何创建新模块
- 如何创建新包

从 wrox.com 下载本章代码

可以在 wrox.com 网站找到本章涉及的代码。代码可以在 www.wrox.com/go/pythonprojects 的 Download Code 选项卡下找到。第 1 章代码位于 Chapter 1 download，每个文件都是根据本章提到的代码文件名命名的。

如果已经遗忘了一些基础知识，本章开始部分提供了 Python 的简短回顾以及一些基础知识。本书的内容都是基于这部分基础知识的。如果对自己的 Python 编程基础有足够的信心，则可以跳过这些内容直接去阅读感兴趣的部分。毕竟，之后当需要温习这些内容时，可以随时回到本章。

在本章，你首先会了解 Python 的生态系统、数据类型和主要的控制结构，以及函数和类的定义的知识。然后，你会看到 Python 模块和包系统。最后会创建一个基础的新模块包。这个包会包含数个模块。

在本章的结尾，你应该为进一步的内容做好准备，并且开始将标准 Python 模块应用在真实的项目任务中。

1.1 探索 Python 语言和解释器

Python 是一个动态的但是拥有严格类型的编程语言(Python 是严格类型的编程语言，

解释器会追踪每个变量的类型)。Python 代码既被解释又被编译。Python 源代码首先被编译成字节码，然后被解释器解释。但是这个过程对于用户来说是透明的。你不需要显式地调用 Python 来编译你的代码。

Python 语言有几种实现版本。但最常用的版本是用 C 语言实现的，通常被称为 CPython。其他实现版本包括 Java 语言实现的 Jython，以及专门为微软.NET 平台实现的 IronPython。本书所用的 Python 实现版本是 CPython。

注意：在本书成文时，Python 存在两种版本流：2.x 版本和 3.x 版本。本书将专注于 3 版本。本书所涉及的代码已经在 3.x 版本流的几个版本中测试过，包括最新的 3.4 版本。当涉及与 2.x 版本的较大兼容性问题时，我们通常指的是 2.7 版本。

Python 程序通常被写在带有.py 后缀的文本文件中。Python 解释器被称为 python(小写)。实际上，Python 解释器并不在意文件的后缀，添加这个后缀仅仅是为了方便用户(在有些操作系统中，这样做也允许文件和解释器建立关联)。

也可以直接在解释器中输入 Python 代码。这个方法适用于高度交互的开发风格。在这种开发风格中，想法首先在解释器中形成原型或被测试，然后转移到代码编辑器中。当开始使用一个新概念或代码模块时，Python 解释器是一个强大的学习工具。

在这种模式下工作时，可以在操作系统命令提示符中输入 python 来启动解释器。系统会反馈给你一个包含 Python 版本和一些创建细节的信息。在这个信息的下面是一个交互式提示符。可在这里输入代码。它看起来如下所示：

```
ActivePython 3.3.2.0 (ActiveState Software Inc.) based on
Python 3.3.2 (default, Sep 16 2013, 23:10:06) [MSC v.1600 32 bit (Intel)]
  on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

从这个信息中，我们知道解释器适用于 3.3.2.0 版本的 Python。它是一个 ActiveState 的发行版本(而不是 python.org 发行版本)。它是基于 Windows 32 位操作系统创建的。你获得的信息可能会稍有不同，但应该包含相同类型的信息。

如果想执行一个存储在文件中的程序而不是与 Python 解释器交互，则可以在操作系统提示符中使用 python 命令，并在后面附上文件的名称：

```
$ python myscript.py
```

注意：通常，也可以在你的文件管理器工具中双击文件，操作系统会调用 python 来自动运行程序。然而，这通常会导致：程序打开一个窗口，运行结束，然后在你看到结果前关闭窗口。所以你可能会倾向于在一个命令行提示中完整地输入 python filename 命令。

Python 带有两个非常有帮助的函数：dir(name)和 help(name)。它们有助于你学习和探索这门语言。dir(name)会告诉你 name 所确定的对象有哪些可用的名称。而 help(name)将会展示被称为 name 的对象的信息。当第一次导入一个新模块时，通常是不知道这个模块包含哪些函数或类。通过查看 dir()得到的模块列表，可以知道有哪些方法和成员是可用的。然后可以在列出的任意特性上调用 help()。一定要试试这些函数，它们是非常重要的信息来源。

1.2　回顾 Python 数据类型

Python 支持很多强大的数据类型。表面上看，这些数据类型和其他编程语言中对应的数据类型一样。但是在 Python 中，这些数据类型通常拥有强大的能力。任何东西在 Python 中都是对象，也因此拥有方法。这意味着可以对任何变量执行一系列的操作。内置的 dir()和 help()函数可以帮助你了解全部信息。在本节，你会看到标准数据类型以及它们最重要的操作。

提示：如果需要，Python 参考手册提供了全部细节(http://docs.python.org/3.3/reference/)。

需要注意 Python 的一些基础概念。首先，Python 变量仅是名称。变量名的创建是通过把类型的实例赋值给它们。变量本身并没有类型，而与它们绑定在一起的对象拥有类型。名称只是个标签，同样，它也可以被一个完全不同的对象重新赋值。赋值操作使用=操作符，所以把一个值赋给一个变量就如下所示：

```
aVariable = aValue
```

这段代码把值 aValue 绑定到变量名 aVariable 上。如果此变量名不存在，解释器会把这个名称添加到合适的命名空间中。

因此，在 Python 中区别变量和它指向的对象是非常重要的。可以用双等号(==)来检查两个变量是否相等，也可以用 is 操作符来检查两个变量的对象身份(也就是两个名称是否

指向同一个对象)，如下所示：

```
>>> aString = 'I love spam'
>>> anotherString = 'I love spam'

>>> anInt = 6
>>> intAlias = anInt
>>> aString == anotherString  # test equality of value
True
>>> aString is anotherString  # test object identity
False
>>> anInt == intAlias         # same value
True
>>> anInt is intAlias         # also same object identity
True
```

Python 根据你使用类型的方式对它们进行分类。比如，所有类型都可以分类为可变的(mutable)或不可变的(immutable)。如果一个类型是不可变的，它意味着这种类型的对象一旦创建后就不能再改变。可以创建一个新的数据项并把它赋值给同一个变量，但是不能更改原来不可变的值。

Python 也支持多个容器类型，有时也被称作序列(严格来讲，容器是序列的子集，稍后会清晰阐述它们之间的差别)。尽管并不是所有的序列都支持所有的操作，但它们有一组共同的操作。

一些 Python 数据类型是可被调用的。这意味着可以像调用函数一样使用类型名来生成这种类型的一个新实例。如果没有给定初值，则会返回一个默认值。你将在下面每个数据类型的描述中看到相关的示例。

现在，你已经了解了操作 Python 数据类型的基础知识，下面会看到不同的数据类型，包括数值、布尔、None 类型以及各种容器类型。

1.2.1 数值类型：整数和浮点数

Python 支持多种数值类型，包括最基本的整数类型和浮点类型。

Python 整数类型的特点在于它在理论上是无限大的。事实上，整数的大小只被你的计算机的内存限制。整数类型支持所有常用的数值操作，比如加法、减法、乘法等。可以使用传统的中缀表示法进行算术运算。比如，当相加两个整数时:

```
>>> 5 + 4
9
```

或:

```
>>> result = 12 + 8
>>> print (result)
20
```