HZ BOOKS
华章教育

计 算 机 科 学 丛 书

Pearson

程序设计导论

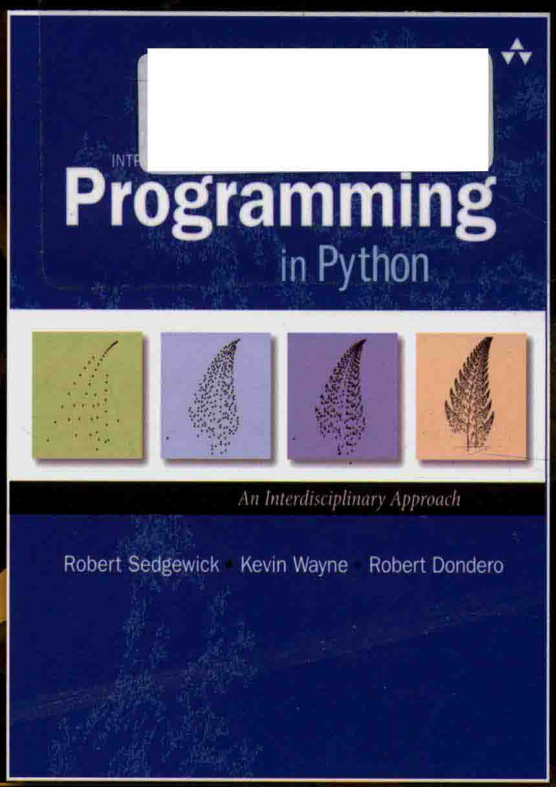
Python语言实践

罗伯特·塞奇威克（Robert Sedgewick）
[美] 凯文·韦恩（Kevin Wayne） 著
罗伯特·唐德罗（Robert Dondero）

汪红 余青松 译

Introduction to Programming in Python

An Interdisciplinary Approach



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

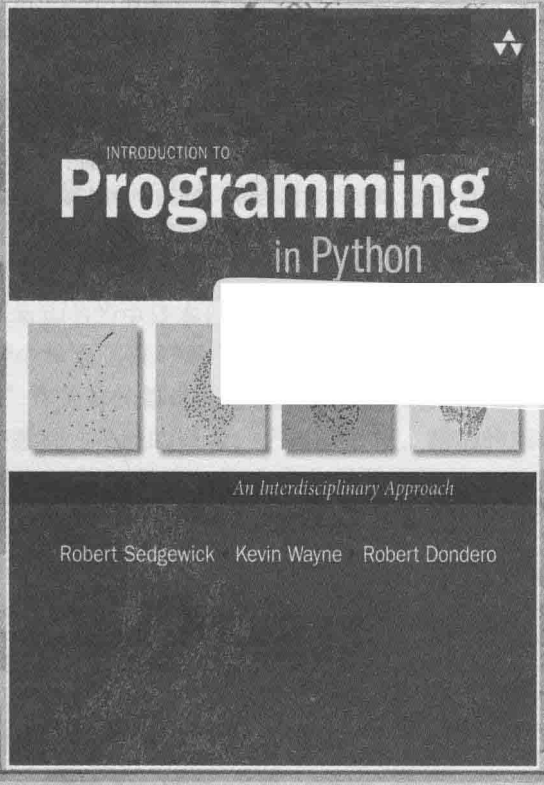
程序设计导论

Python语言实践

罗伯特·塞奇威克（Robert Sedgewick）
[美] 凯文·韦恩（Kevin Wayne） 著
罗伯特·唐德罗（Robert Dondero）
江红 余青松 译

Introduction to Programming in Python

An Interdisciplinary Approach



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

程序设计导论：Python 语言实践 /（美）罗伯特·塞奇威克（Robert Sedgewick）等著；江红，余青松译 . —北京：机械工业出版社，2016.9
（计算机科学丛书）
书名原文：Introduction to Programming in Python: An Interdisciplinary Approach

ISBN 978-7-111-54924-6

I. 程… II. ① 罗… ② 江… ③ 余… III. 软件工具－程序设计－高等学校－教材 IV. TP311.56

中国版本图书馆 CIP 数据核字（2016）第 230536 号

本书版权登记号：图字：01-2015-5193

本书从跨学科的角度入手，强调应用，介绍 Python 最有用的功能，包括编程的基本要素、功能、模块、面向对象编程、数据抽象对象、算法与数据结构，融汇了作者丰富的课堂教学经验，提供了大量源代码、I/O 库和精选实例。本书适合作为高校计算机专业编程课程的教材。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：刘诗灏　　责任校对：殷　虹
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2016 年 11 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：33.5（含 0.25 印张彩插）
书　　号：ISBN 978-7-111-54924-6　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章科技图书出版中心

译者序

Introduction to Programming in Python: An Interdisciplinary Approach

本书介绍程序设计的基本概念，而不仅仅是 Python 本身。本书的侧重点在于讲授使用程序设计解决各学科（从材料科学到基因组学、天体物理学、网络系统等）中的计算问题。本书除了讲述 Python 语言基础知识之外，还涉及许多新的研究领域（例如，随机 Web 冲浪模型、渗透原理、多体模拟、数据挖掘、小世界现象等），能激发学生对科学探究的求知欲，为以后专业课的学习打下坚实的基础。

本书采用跨学科的方法，重点讲述计算在其他学科中的重要地位。这种跨学科的方法向学生强调一种基本思想，即在当今世界中，数学、科学、工程和计算紧密结合在一起。本书面向对使用计算机程序解决数学、科学和工程问题感兴趣的大学生或研究生，作为教材的同时也可用于自学，或作为与其他领域相结合的程序设计课程的补充材料。

本书内容根据学习程序设计的四个阶段来组织：基本元素、函数和模块、面向对象的程序设计、算法和数据结构。本书由浅入深，将理论知识和实际应用相结合，逐步引导读者掌握通过计算机程序设计解决各种科学和技术研究问题的方法。本书的最大特色是提供丰富的实际应用示例，用于分析和解决各学科中涉及的计算问题。本书的应用示例涉及应用数学、物理、生物科学、计算机科学、物理系统、数字方法、数据可视化、声音合成、图像处理、金融模拟和信息技术等方面，真正体现了其跨学科的特点。

另外，本书包括大量的习题和创新习题，可引导读者进一步拓展通过程序设计解决科学和技术问题的能力。

本书配套课程是普林斯顿大学的精品课程，在其提供的教学官网（http://introcs.cs.princeton.edu/python）中包含大量的教学辅助内容，无论是教师、助教、学生还是一般读者，均可以从中获取与本书内容相关的所有资源库。

本书由华东师范大学江红和余青松共同翻译。衷心感谢本书的编辑王颖老师和刘诗灏老师，敬佩他们的睿智和敬业。我们在翻译过程中力求忠于原著，但由于时间和学识有限，且本书涉及各个领域的专业知识，故书中的不足之处在所难免，敬请诸位同行、专家和读者指正。

江红　余青松

21 世纪以前的教育基础是“读、写和算术”，而现在的教育基础则是“读、写和计算”。学习编程是每个科学和工程专业学生教育过程中的重要部分。除了直接的应用外，学习编程是了解计算机科学本质的第一步。计算机科学对现代社会产生了毋庸置疑的影响。本书的目的是在科学环境中为需要编程或想学习编程的人讲授程序设计的基本方法和应用技巧。

我们的主要目标是通过提供经验和必要的基本工具使得学生更加有效地进行计算。我们的方法是向学生灌输这样的理念：编写程序是一种自然而然、富有成就感和充满创造性的体验。我们将循序渐进地介绍基本概念，并使用应用数学和科学中的典型应用来阐述这些概念，并为学生提供编写程序以解决相关问题的机会。

我们使用 Python 程序设计语言来编写本书中的所有程序——在本书的标题中，我们在“程序设计”之后提及“ Python”以强调本书是关于程序设计的基本概念，而不仅仅是 Python 本身。这本书讲授了许多解决计算问题的基本技能，这些技能可以应用于许多现代计算环境中。本书自成体系，其目标人群是没有任何编程经验的人。

相对于传统的 CS1 课程而言，本书提供了一种跨学科的方法。我们将重点讲述计算在其他学科（材料科学、基因组学、天体物理学、网络系统等）中的重要地位。跨学科的方法向学生强调一种基本思想，即在当今世界中，数学、科学、工程和计算紧密结合在一起。同时，作为 CS1 的课本，本书主要面向对数学、科学和工程感兴趣的大学一年级学生。当然，本书也可用于自学，或者作为程序设计与其他领域相结合的课程的补充材料。

内容范围

本书根据学习编程的四个阶段来组织：基本元素、函数和模块、面向对象的程序设计、算法和数据结构。在进入编程的下一阶段之前，我们将向读者提供他们需要的基本信息，使读者有信心编写每个阶段的程序。本书所讲授方法的基本特征是使用示例程序解决感兴趣的问题，并提供各种练习题，从自学练习题到需要创新解决方案的挑战性难题。

*基本元素*包括变量、赋值语句、内置数据类型、控制流程、数组和输入 / 输出，以及图形和声音。

*函数和模块*为学生揭开了模块化程序设计的面纱。我们使用熟悉的数学函数来介绍 Python 函数，然后讨论使用函数编程的意义，包括库函数和递归函数。贯穿本书，我们强调一种基本理念，即把一个程序分解为可以独立调试、维护和重用的模块。

*面向对象的程序设计*是我们对数据抽象的介绍。我们强调数据类型的概念，并使用 Python 的类机制实现数据类型。我们将教会学生如何使用、创建和设计数据类型。模块化、封装和其他现代程序设计理念是面向对象程序设计阶段的中心概念。

*算法和数据结构*把这些现代程序设计理念与组织和处理数据的经典方法结合起来，因为经典方法依旧可以有效地用于现代应用程序。我们介绍了经典的排序和搜索算法，同时也介绍了基本的数据结构及其应用，强调了使用科学方法来理解实现的性能特征。

*在科学和工程中的应用*是本书的一个主要特点。我们通过其对具体应用的影响来强调我们所讨论的每一个程序设计概念。我们的示例来源于应用数学、物理学、生物科学、计算机科学本身，并包括物理系统模拟、数值方法、数据可视化、声音合成、图像处理、金融模拟和信息技术。具体的示例包括第 1 章用于页面排名的马尔可夫链以及渗透问题、*n* 体模拟、小世界现象的案例研究。这些应用都是正文不可分割的组成部分。它们为学生提供了资料，阐述程序设计概念的重要性，并提供了计算在现代科学和工程中扮演着重要角色的令人信服的证据。

我们的主要目标是教授学生学会有效解决任何程序设计问题所需要的具体机制和技能。我们完全使用 Python 程序，并鼓励读者也使用 Python 程序。我们关注个人的程序设计，而不是大型的程序设计。

本书在大学课程中的使用

本书主要面向大学一年级课程，其目标是教授新生在科学应用的背景下进行程序设计。根据本书所讲授的内容，将来主修科学或工程技术的学生都将学会在熟悉的背景下学习程序设计。修完基于本书课程的学生将为在后续科学和工程技术课程中应用他们的技能做好准备，并会意识到本书所讲授的内容对进一步学习计算机科学是非常有益的。

特别是将来主修计算机科学的学生将会受益于在科学应用的背景下学习程序设计。与生物学家、工程师和物理学家一样，计算机科学家在科学方法中也需要相同的基本背景，并且要承担科学计算的任务。

实际上，跨学科的方法使得高等院校可给将来主修计算机科学或其他科学和工程技术的学生教授同一门课程。我们覆盖了 CS1 所规定的资料，但是我们对应用的关注给相关概念带来了生命，并激发了学生学习这些概念的兴趣。跨学科的方法向学生展示了许多不同学科中的问题，可帮助他们更明智地选择主修方向。

无论采用哪种具体机制，本书的使用最好安排在全部课程的早期。首先，这种安排允许我们利用高中数学和科学中所熟悉的资料。其次，学生在大学课程的早期学习程序设计将帮助他们在继续学习专业课程时有效地使用计算机。像阅读和写作一样，程序设计很显然也是任何科学家和工程师的一项基本技能。掌握本书概念的学生将终生不断发展这种技能，在其各自所选择的领域中，他们能够利用计算来解决或更好地理解问题和项目，并从这一过程中受益。

先修条件

本书非常适合于科学和工程技术专业的大学一年级学生。也就是说，我们不需要其他的预备知识，本书的先修条件和其他入门级科学和数学课程的要求基本一致。

*完备的数学知识*很重要。我们没有详细阐述相关的数学知识，但我们引用了学生在高中已经学习的数学课程，包括代数学、几何学和三角学。本书目标人群中的大多数学生都自动满足这些要求。事实上，我们充分利用了他们在基础课程中所熟悉的知识来介绍基本的编程概念。

*科学的求知欲*也是一个重要的部分。科学和工程技术专业的学生天生对进行科学探究以帮助解释自然本质的能力非常着迷。我们使用简单的关于自然界的程序示例支持这种偏爱。

本书任何特定的知识都没有超过高中课程中的数学、物理、生物和化学的知识范围。

程序设计经验不是必需的，但却是有益的。讲授程序设计是我们的主要目标，因此本书没有要求任何先行的程序设计经验。然而，编写一个程序解决一个新问题是一项富有挑战性的智力任务，所以在高中阶段编写了许多程序的学生会从选修基于本书的程序设计入门课程中受益。本书可满足各种不同背景的学生的授课需求，因为本书中的应用无论对于新手还是专家都具有吸引力。

使用计算机的经验也不是必需的，况且这根本不是问题。现在的大学生经常使用计算机与亲朋好友交流、听音乐、处理照片或进行许多其他活动。能够以有趣而又重要的方式驾驭自己的计算机需要扣人心弦和长期的训练。

总之，几乎所有科学和工程技术领域的学生都可以在他们第一个学期的课表中选修基于本书的课程。

目标

在科学和工程技术专业的高级课程中，教师希望完成基于本书课程的学生学到什么样的知识呢？

我们覆盖了 CS1 课程，但任何讲授入门级程序设计课程的教师都知道，教授后续课程的教师期望值很高：每个教师都希望学生已经熟悉所需使用的计算环境和方法。物理学教授可能期望某些学生在周末设计一个程序来运行模拟；工程学教授可能期望某些学生使用一个特定的软件包并基于数值方法求解微分方程；计算机科学教授可能期望学生掌握特定编程环境的详细知识。本书真的可以满足这些不同的期望吗？对于不同的学生群体，是否需要不同的入门级课程？

自从 20 世纪后期计算机被广泛使用以来，高等院校就一直被这些类似问题困扰。对于这些问题，我们给出的解答是本书介绍通用的程序设计入门方法，类似于数学、物理学、生物学和化学中普遍接受的入门级课程。本书努力为科学和工程技术专业的学生提供必要的基本准备，同时也清楚地传递这样的信息：理解计算机科学比程序设计更重要。学习过本书的学生，教师可期望他们拥有适应于新的计算环境和在不同应用中有效利用计算机的必要知识和能力。

完成基于本书课程的学生，他们期望在后续课程中学习到什么呢？

我们的观点是程序设计并不难学，但学会驾驭计算机意义深远。在未来的职业生涯中，掌握了本书知识的学生已为应对计算挑战做好准备。他们了解现代程序设计环境（例如本书介绍的 Python）将为未来可能遇见的任何计算问题打开一扇大门，同时他们也获得了学习、评价和使用其他计算工具的信心。对计算机科学感兴趣的学生将准备好进一步追寻这些兴趣，科学和工程技术专业的学生将准备好将计算融合到他们的研究中。

本书官网

在如下网站上，可以找到关于正文的大量补充信息：

http://introcs.cs.princeton.edu/python

为了方便，我们把这个站点称为本书官网。该网站包含了为使用本书的教师、学生和其他读者准备的资料。我们在这里简要描述一下这些资料，虽然所有的 Web 用户都知道，最

好的方法是通过浏览器纵览它们。除了少部分用于测试的资料，其他资料都是公开可用的。

本书官网的一个最重要的意义是让教师和学生可以使用自己的计算机教授或学习这些资料。任何拥有计算机和浏览器的人，均可按照本书官网提供的一些指示开始学习程序设计。这个过程并不比下载一个媒体播放器或一首歌更困难。和任何其他网站一样，我们的网站也一直保持更新。对于任何拥有本书的人而言，本书官网是一个非常重要的资源。特别是补充材料对于我们达到如下目标至关重要，那就是使得计算机科学成为所有科学家和工程师教育不可分割的有机组成部分。

对于教师，本书官网包含了与教学相关的信息。这些信息主要按照我们过去十几年开发的教学模式进行组织，我们每周为学生授课两次，并且每周对学生进行两次课外辅导，学生分成小组与任课教师或助教进行讨论。本书官网包括用于这些授课的演示幻灯片，教师可基于这些幻灯片根据需要进行补充和修改。

对于助教，本书官网包含了详细的问题集和编程项目，它们均基于本书的习题，但包含更多的详细信息。每个程序设计任务作业旨在基于一个有趣的应用环境教授一个相关的概念，同时为每个学生提出一个引人入胜的挑战。课外作业的进展体现了我们的教学方法。本书官网全面详细地说明了所有的作业，并提供详细的结构化信息帮助学生在规定时间内完成任务，包括有关建议方法的描述，以及在课堂中应该讲述的授课内容纲要。

对于学生，本书官网包含可快速访问的本书的大部分资料，包括源代码以及鼓励学生自学的额外资料。本书官网为书本中的许多习题提供了参考解答，包括完整的程序代码和测试数据。还有许多与程序设计作业相关的信息，包括建议的方法、检查清单、常见问题解答以及测试数据。

对于一般读者，本书官网是访问与本书内容相关的所有额外信息的资源库。所有的网站内容都提供 Web 超链接和其他路径，以帮助读者寻找有关讨论主题的更多信息。网站包含了非常多的信息，比任何个人所能想象和接受的信息多得多，因为我们的目标是为本书内容提供足够多的信息，以满足每位读者的需求。

致谢

这个项目自 1992 年开始启动，迄今为止，许多人为这个项目的成功做出了贡献，我们在此对他们表示诚挚的感谢。特别感谢 Anne Rogers 的大力帮助，使本项目得以顺利启动；感谢 Dave Hanson、Andrew Appel 和 Chris van Wyk 耐心地解释数据的抽象化；还要感谢 Lisa Worthington，她是第一个接受挑战，使用本书给大学一年级学生上课的老师。同时我们还要感谢 /dev/126 的努力；感谢过去 25 年中在普林斯顿大学致力于讲授本书内容的教师、研究生和教学人员；感谢成千上万努力学习本书的大学生们。

Robert Sedgewick
Kevin Wayne
Robert Dondero
2015.4

程序设计的基本元素

本章的目标是向读者证明编写一个程序比撰写一篇文章（例如一个段落或论文）更加容易。撰写散文十分困难：我们在学校中花费了多年的时间学习如何进行散文创作。比较而言，仅仅若干构造模块就足以使读者能够编写程序，用以解决各种有趣的问题（这些问题如果通过其他方法则难以解决）。在本章，我们引导读者学习这些构造模块，开始 Python 程序设计之旅，同时学习各种各样有趣的程序。仅仅需要几个星期的学习，读者就可以通过编写程序的方法表述自己的思想。正如撰写文章的技能一样，程序设计的技能是一个人的终生技能，它可以使你不断完善来更好地融入未来。

在本书中，读者将学习 Python 程序设计语言。这个任务十分容易，至少比学习一门外语要容易得多。实际上，程序设计语言仅仅由几十个词汇和语法规则组成。本书涉及的大多数内容也可以使用 Java 或 C++ 语言或任何其他现代程序设计语言来描述。我们特别使用 Python 描述一切，目的是使读者可以立即开始编写和运行程序。一方面，我们将侧重于学习如何编程，而不是学习 Python 的语言细节。另一方面，程序设计的挑战是了解在特定情况下有哪些相关的细节。Python 是一种广泛使用的语言，学习 Python 使得读者可在许多计算机上编写程序（例如，你自己的计算机）。另外，学习使用 Python 进行编程可让学习其他程序设计语言更加容易，包括低级语言（例如 C）以及专业语言（例如 Matlab）。

1.1 你的第一个程序

本节通过编辑并运行一个简单程序的基本步骤，引导读者进入 Python 程序设计的世界。

与其他熟知的应用程序（例如文字处理软件、电子邮件程序和 Web 浏览器）不同，Python 系统（简称 Python）是一系列应用程序的集合。当然，正如其他应用程序一样，使用 Python 前必须确保在计算机上已经正确安装了 Python 软件。许多计算机系统都预装了 Python 软件，你也可以非常容易地下载并安装 Python 软件。Python 程序设计环境还需要一个文本编辑器和控制台命令程序。读者首要的任务是通过访问下列网站获取在计算机上安装 Python 程序设计环境的说明书。

```
http://introcs.cs.princeton.edu/python
```

上述网站为本书的官网，网站上包含大量本书有关资料的补充信息，读者编程时可以参考和使用。

表 1-1-1 为本节所有程序的一览表，读者可以作为参考。

表 1-1-1　本节所有程序的一览表

程序名称	功能描述
程序 1.1.1（helloworld.py）	Hello, World
程序 1.1.2（useargument.py）	使用一个命令行参数

1.1.1 Python 程序设计

为了介绍 Python 程序开发，我们将整个编程过程分解为两个步骤。具体如下：

- 步骤 1：通过键入代码来编写程序并保存到一个文本文件中，例如，myprogram.py。
- 步骤 2：在控制台命令窗口中通过键入命令 python myprogram.py 运行（或执行）程序。

在步骤 1 中，从新建空白文件开始，输入一系列程序代码字符，过程就像写电子邮件或论文一样。程序员使用*代码*描述程序文本，而创建和编辑代码的行为则称为*编码*。在步骤 2 中，计算机的运行控制从系统转移到所运行的程序（程序运行结束后，运行控制重新返回系统）。许多系统支持各种不同的编写和运行程序的方法。本书统一选用上述步骤，因为该步骤非常易于描述和开发小程序。

1. 编写 Python 程序

Python 程序就是保存在后缀为 .py 的文件中的字符系列。只要使用文本编辑器就可以创建 Python 程序文件。可以使用任何文本编辑器，也可以使用本书官网推荐的功能强大的程序开发环境。

本书官网推荐的程序开发环境功能全面，适用于本书的所有范例，且使用并不复杂，并提供许多实用功能，被专业程序员广泛使用。

2. 执行程序

程序编写后，即可被运行（或称执行）。运行程序是令人激动的时刻，此时程序获取计算机的控制权（在 Python 允许的限制下）。准确地说是计算机执行程序的指令。更准确地说，是 Python *编译器*将你的 Python 程序编译成更适合在计算机上执行的计算机语言，然后 Python *解释器*指示计算机执行程序指令。在本书中，我们使用术语“*执行*”（executing）或“*运行*”（running）描述编译、解释以及执行一个程序的过程（例如，当 Python 运行该程序时……）。要使用 Python 编译器和解释器执行一个程序，可在控制台命令窗口中键入 Python 命令并加上程序的文件名。

程序 1.1.1 是一个完整的 Python 程序实例。程序代码位于文件 helloworld.py 中。该程序唯一的功能就是在控制台中输出一则信息。Python 程序由*语句*组成。一般情况下，一条语句占一行。

- helloworld.py 的第 1 行包含一条 import 语句。import 语句指示 Python 使用定义在 stdio *模块*（即 stdio.py 文件）中的功能。stdio.py 是我们为本书特别设计的模块文件，其中定义了用于输入和输出的函数。一旦导入了 stdio 模块，则随后可调用定义在该模块中的任意函数。
- 第 2 行为空白行。Python 忽略空白行，空白行主要用于分隔代码中的逻辑块。
- 第 3 行为*注释*。注释用于程序中的文档说明。Python 语言的注释从字符 '#' 开始，直至行结束。本书注释使用浅灰色字体格式，Python 忽略所有注释，注释仅仅用于增加程序的可阅读性。
- 第 4 行为本程序的核心。该语句调用函数 stdio.writeln()，在控制台输出指定文本。注意：调用其他模块中的函数时，使用“模块名加英文句号再加函数名”的格式。

程序 1.1.1 Hello, World (helloworld.py)

```
import stdio

# Write 'Hello, World' to standard output.
stdio.writeln('Hello, World')
```

上述 Python 程序代码完成一个简单任务。按惯例，该程序是初学者的第一个程序。程序的运行过程和结果如下所示。控制台命令程序显示命令提示符（本书为 %），用户键入 Python 命令（本书使用粗体）。使用 Python 执行程序代码，在控制台窗口输出 'Hello, World'，即第 4 行语句的运行结果。

```
% python helloworld.py
Hello, World
```

Python 程序开发流程如图 1-1-1 所示。

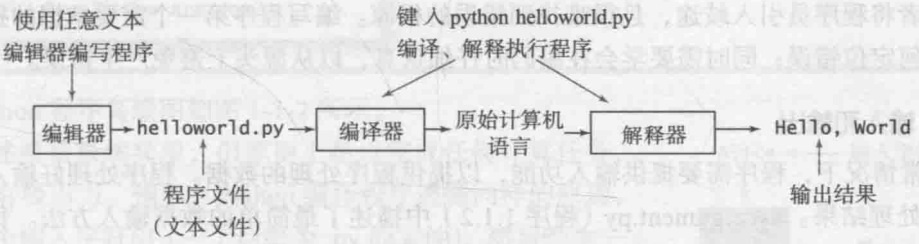


图 1-1-1 Python 程序开发流程

Python 2

本书使用的版本为 Python 3，因为 Python 3 是 Python 程序设计未来的发展方向。当然，我们尽量确保本书的代码可同时在 Python 3 和 Python 2 中运行。例如在 Python 2 中，helloworld.py 程序可以简单地包含一条语句 print 'Hello, World'，但该程序在 Python 3 中将报错。因此，本书特别编写并使用 stdio 模块，其中包含了同时适用于两个版本的输出函数。如果所涉及的知识点在两个版本中存在显著差别，本书将使用类似的方式提醒 Python 2 用户。

自 20 世纪 70 年代起形成一个惯例，初级程序员的第一个程序为输出 'Hello, World'。所以，首先在名为 helloworld.py 的文件中键入程序 1.1.1 的代码，然后执行该程序。成千上万的程序员就是按照上述步骤学习程序设计的。当然，学习程序设计还需要一个文本编辑器和控制台命令程序。当然，在控制台窗口输出内容似乎没有太大意义，但仔细思考后你会意识到，程序反馈并告知我们操作结果，是程序必须具备的最基本功能之一。

到目前为止，本书所有的程序架构将与 helloworld.py 类似，不同之处在于不同的文件名、不同的注释和不同的语句系列。因而，编写程序时无须从新文件开始。替代方法为：

- 复制 helloworld.py 文件，并重新命名为你所需要的程序名称。注意，请确保新文件名的后缀为 .py。

- 替换注释内容。
- 将 stdio.writeln() 语句替换为不同的语句系列。

一个程序由文件名和文件中的语句系列来确定。按惯例，Python 程序包含在后缀为 .py 的文本文件中。

3. 错误

学习 Python 程序设计时，很容易模糊程序的编辑、编译和解释执行之间的界限。但如果要更好地学习程序设计，并理解程序设计过程中不可避免的错误的成因，则必须将这些概念区分开。本节后的“问题和解答”部分包括若干程序错误的例子。

编写程序时，通过仔细检查程序代码，可修正或避免大部分错误。就像编辑电子邮件信息时，修正拼写和语法错误的方法一样。有些错误称为*编译时错误*，在 Python 编译程序时产生。这些错误阻止编译器编译代码。Python 将编译错误显示为 SyntaxError。另一些错误为*运行时错误*，在 Python 解释执行程序时产生。例如，如果 helloworld.py 中漏写了语句 import stdio，则运行该程序时 Python 将抛出一个 NameError 错误。

一般而言，程序中的错误通常也称为 bug。错误是程序员的灾星：错误信息常常会令人困惑或者将程序员引入歧途，且很难找到错误的根源。编写程序第一个需要掌握的技能就是学习如何定位错误；同时需要学会在编码时仔细认真，以从源头上避免产生错误。

5

1.1.2 输入和输出

通常情况下，程序需要提供输入功能，以提供程序处理的数据。程序处理好输入的数据后输出处理结果。useargument.py（程序 1.1.2）中描述了最简单的数据输入方法。每次运行程序 useargument.py 时，程序接收命令行参数（运行时在程序名后键入），并作为消息的一部分输出到控制台窗口。程序运行的结果与程序名后键入的内容有关，使用不同的命令行参数运行该程序时，可以得到不同的输出结果。

在 useargument.py 中，语句 import sys 告知 Python 我们要使用定义在 sys 模块中的功能。sys 模块中的一个功能名为 argv，用于存储命令行参数（命令行中位于“python useargument.py”之后以空格分隔的内容）列表。本书 2.1 节将详细讨论其机制。目前仅需理解 sys.argv[1] 表示命令行中程序名后所键入的第 1 个参数，sys.argv[2] 表示命令行中程序名后键入的第 2 个参数，依此类推。在程序体中，可使用 sys.argv[1] 表示运行程序时在命令行中键入的第 1 个参数，具体可参照 useargument.py 中的代码。

除了使用 writeln() 函数，程序 1.1.2 还调用了 write() 函数。write() 函数与 writeln() 函数类似，但仅输出字符串（不换行）。

诚然，程序仅仅完成从控制台获取用户输入的内容并回显到控制台窗口的任务似乎没有太大意义，但通过仔细观察和思考你会意识到，如何使程序响应来自于用户的基本信息并控制程序的运行结果也是程序需要具备的另一个基本功能。程序 useargument.py 所展示的简单模型足以启发我们仔细思考 Python 程序的基本编程机制，以及如何借此解决各种有趣的计算问题。

回顾一下，useargument.py 程序的功能确实恰好完成了将一个字符串（参数）映射为另一个字符串（回显到控制台窗口的消息）的功能。使用该程序时，可以将该程序想象为一个将输入字符串转换为输出字符串的黑盒子。