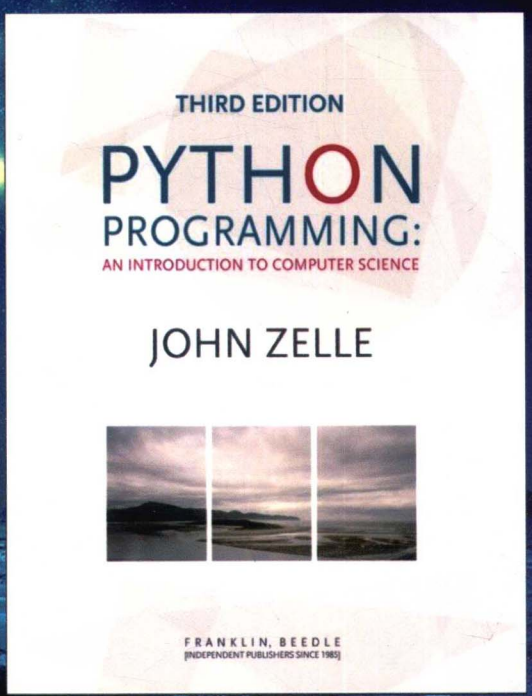
国外著名高等院校
信息科学与技术优秀教材

FRANKLIN, BEEDLE & ASSOCIATES INC.

异步图书
www.epubit.com.cn

# Python程序设计（第3版）

[美] 约翰·策勒（John Zelle）著　王海鹏 译

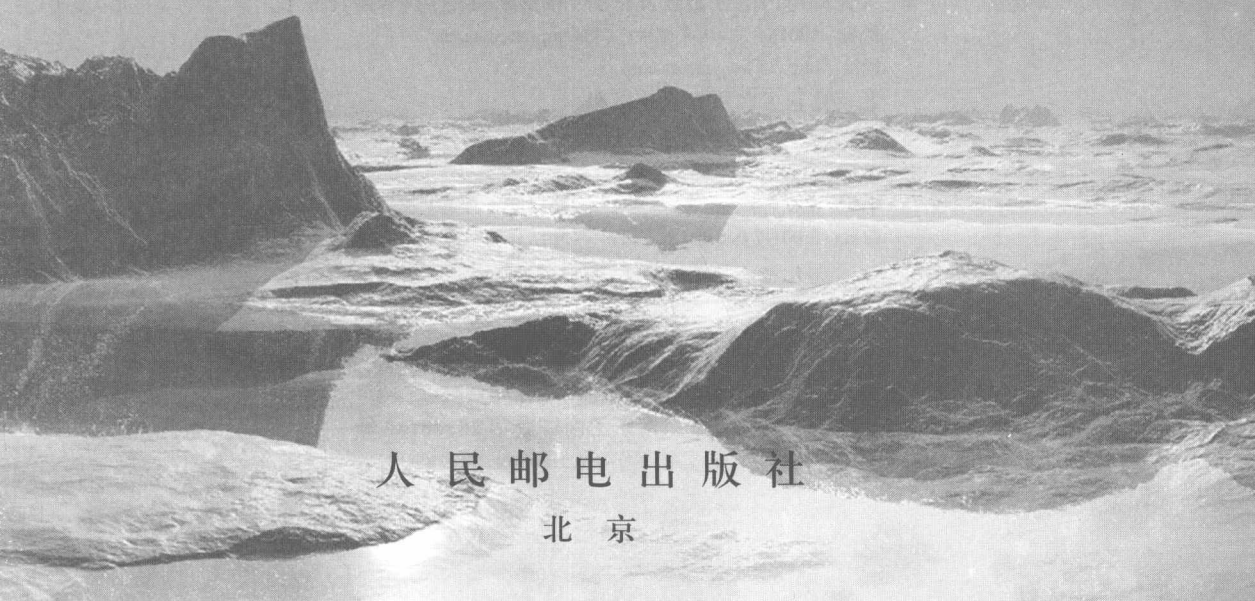



中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

国外著名高等院校
信息科学与技术优秀教材

# Python程序设计
# （第3版）

[美] 约翰·策勒（John Zelle）著　王海鹏 译



人民邮电出版社
北京

图书在版编目（CIP）数据

Python程序设计 : 第3版 / (美) 策勒(John Zelle) 著 ; 王海鹏译. -- 北京 : 人民邮电出版社, 2018.1(2018.3重印)
国外著名高等院校信息科学与技术优秀教材
ISBN 978-7-115-28325-2

Ⅰ. ①P… Ⅱ. ①策… ②王… Ⅲ. ①软件工具－程序设计－高等学校－教材 Ⅳ. ①TP311.561

中国版本图书馆CIP数据核字(2017)第293398号

## 版权声明



◆ 著　[美] 约翰·策勒（John Zelle）
译　王海鹏
责任编辑　陈冀康
责任印制　焦志炜

◆ 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
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：21.5
字数：506 千字　2018 年 1 月第 1 版
印数：3 001-7 000册　2018 年 3 月北京第 2 次印刷
著作权合同登记号　图字：01-2016-3755 号

定价：69.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

# 内容提要

本书是面向大学计算机科学专业的教材。本书以 Python 语言为工具，采用相当传统的方法，强调解决问题、设计和编程是计算机科学的核心技能。

全书共 13 章，此外，还包含两个附录。第 1 章到第 5 章介绍计算机与程序、编写简单程序、数字计算、对象和图形、字符串处理等基础知识。第 6 章到第 8 章介绍函数、判断结构、循环结构和布尔值等话题。第 9 章到第 13 章着重介绍一些较为高级的程序设计方法，包括模拟与设计、类、数据集合、面向对象设计、算法设计与递归等。附录部分给出了 Python 快速参考和术语表。每一章的末尾配有丰富的练习，包括复习问题、讨论和编程联系等多种形式，帮助读者巩固该章的知识和技能。

本书特色鲜明、示例生动有趣、内容易读易学，适合 Python 入门程序员阅读，也适合高校计算机专业的教师和学生参考。

# 序

当出版商第一次发给我这本书的草稿时，我立刻感到十分兴奋。它看起来像是 Python 教科书，但实际上是对编程技术的介绍，只是使用 Python 作为初学者的首选工具。这是我一直以来想象的 Python 在教育中最大的用途：不是作为唯一的语言，而是作为第一种语言，就像在艺术中一样，开始学习时用铅笔绘画，而不是立即画油画。

作者在本书前言中提到，Python 作为第一种编程语言是接近理想的，因为它不是“玩具语言”。作为 Python 的创建者，我不想独占所有的功劳：Python 源于 ABC，这种语言在 20 世纪 80 年代初由阿姆斯特丹国家数学和计算机科学研究所（CWI）的 Lambert Meertens、Leo Geurts 等人设计，旨在教授程序设计。如果说我为他们的工作添加了什么东西，那就是让 Python 变成了一种非玩具语言，具有广泛的用户群、广泛的标准和大量的第三方应用程序模块。

我没有正式的教学经验，所以我可能没有资格来评判其教育效果。不过，作为一名具有将近 30 年经验的程序员，读过本书，我非常赞赏本书对困难概念的明确解释。我也喜欢书中许多好的练习和问题，既检查理解，又鼓励思考更深层次的问题。

恭喜本书读者！学习 Python 将得到很好的回报。我保证在这个过程中你会感到快乐，我希望你在成为专业的软件开发人员后，不要忘记你的第一种语言。

——Guido van Rossum，Python 之父

# 前　言

本书旨在作为大学的一门计算课程的主要教材。它采用相当传统的方法，强调解决问题、设计和编程是计算机科学的核心技能。但是，这些思想利用非传统语言（即 Python）来说明。在我的教学经验中，我发现许多学生很难掌握计算机科学和程序设计的基本概念。这个困难可以部分归咎于最常用于入门课程的语言和工具的复杂性。因此，这本教材只有一个总目标：尽可能简单地介绍基础计算机科学概念，但不是过于简单。使用 Python 是这个目标的核心。

传统的系统语言（如 C++、Ada 和 Java）的发展是为了解决大规模编程中的问题，主要侧重于结构和纪律。它们不是为了易于编写中小型程序。最近脚本（有时称为“敏捷”）语言（如 Python）的普及程度上升，这表明了一种替代方法。Python 非常灵活，让实验变得容易。解决简单问题的方法简单而优雅。Python 为新手程序员提供了一个很好的实验室。

Python 具有一些特征，使其成为第一种编程语言的接近完美的选择。Python 基本结构简单、干净、设计精良，使学生能够专注于算法思维和程序设计的主要技能，而不会陷入晦涩难解的语言细节。在 Python 中学习的概念可以直接传递给后续学习的系统语言（如 C ++ 和 Java）。但 Python 不是一种“玩具语言”，它是一种现实世界的生产语言，可以在几乎每个编程平台上免费提供，并且具有自己易于使用的集成编程环境。最好的是，Python 让学习编程又变得有趣了。

虽然我使用 Python 作为语言，但 Python 教学并不是本书的重点。相反，Python 用于说明适用于任何语言或计算环境的设计和编程的基本原理。在某些地方，我有意避免某些 Python 的功能和习惯用法，它们通常不会在其他语言中使用。市面上有很多关于 Python 的好书，本书旨在介绍计算。除了使用 Python 之外，本书还有其他一些特点，旨在使其成为计算机科学的平台。其中一些特点如下。

- 广泛使用计算机图形学。学生喜欢编写包含图形的程序。本书提供了一个简单易用的图形软件包（以 Python 模块提供），允许学生们学习计算机图形学原理，并练习面向对象的概念，但没有完整的图形库和事件驱动编程中固有的复杂性。
- 有趣的例子。本书包含了完整的编程示例来解决实际问题。
- 易读的行文。本书的叙事风格以自然的方式介绍了重要的计算机科学概念，这是逐步讨论的结果。我试图避免随意的事实罗列，或稍微有点关系的侧边栏。
- 灵活的螺旋式介绍。因为本书的目的是简单地呈现概念，所以每一章的组织是为了逐渐向学生介绍新的思想，让他们有时间来吸收越来越多的细节。前几章介绍了需要更多时间掌握的思想，并在后面的章节中加以强化。
- 时机恰好地介绍对象。介绍面向对象技术的适当时机，是计算机科学教育中持续存在的争议。本书既不是严格的“早讲对象”，也不是“晚讲对象”，而是在命令式编程的基础上简要地介绍了对象概念。学生学习多种设计技巧，包括自顶向下

（函数分解）、螺旋式（原型）和面向对象的方法。另外，教科书的材料足够灵活，可以容纳其他方法。

- 大量的章末习题。每章末尾的练习为学生提供了充分的机会，强化对本章内容的掌握，并实践新的编程技巧。

## 第 2 版和第 3 版的变化

本书的第 1 版已经有些老旧，但它所采用的方法现在仍然有效，就像当时一样。

虽然基本原则并没有改变，但技术环境却变了。随着 Python 3.0 的发布，对原始资料的更新变得必要。第 2 版基本上与最初的版本相同，但更新使用了 Python 3.0。本书中的每个程序示例几乎不得不针对新的 Python 来修改。此外，为了适应 Python 中的某些更改（特别是删除了字符串库），内容的顺序稍做了调整，在讨论字符串处理之前介绍了对象术语。这种变化有一个好的副作用，即更早介绍计算机图形学，以激发学生的兴趣。

第 3 版延续了更新课本以反映新技术的传统，同时保留了经过时间考验的方法来教授计算机科学的入门课程。这个版本的一个重要变化是消除了 eval 的大部分用法，并增加了其危险性的讨论。在连接越来越多的世界中，越早开始考虑计算机安全性越好。

本书添加了几个新的图形示例，在第 4 章到第 12 章中给出，以引入支持动画的图形库的新功能，包括简单的视频游戏开发。这使得最新的课本与大作业项目的类型保持一致，这些大作业常在现代的入门课程中布置。

在整个课本中还有一些较小的改动，其中包括：

- 第 5 章添加了文件对话框的内容；
- 第 6 章已经扩展并重新组织，强调返回值的函数；
- 为了一致地使用 IDLE（标准的“随 Python 分发的”开发环境），介绍范围已经改进并简化，这使得本书更适合自学和作为课堂教科书使用；
- 技术参考已更新；
- 为了进一步方便自学者，本版的章末习题答案可以在线免费获得。读者可访问异步社区（www.epubit.com.cn）并搜索本书页面，以下载示例代码、习题解答和教学 PPT。

## 本书主要内容

为了保持简单的目标，我试图限制 2 门课不会涵盖的内容数量。不过，这里的内容可能比较多，典型的一学期入门课程也许不能涵盖。我的课程依次介绍了前 12 章中的几乎所有内容，尽管不一定深入介绍每个部分。第 13 章（“算法设计与递归”）中的一个或两个主题通常穿插在学期中的适当时候。

注意到不同的教师喜欢以不同的方式处理主题，我试图保持材料相对灵活。第 1 章～第 4 章（“计算机和程序”“编写简单程序”“数字计算”“对象和图形”）是必不可少的介绍，应该按顺序进行说明。字符串处理的第 5 章（“序列：字符串、列表和文件”）的初始部分

也是基本的，但是稍后的主题（如字符串格式化和文件处理）可能会被延迟，直到后来需要。第 6 章～第 8 章（“定义函数”“判断结构”和“循环结构和布尔值”）设计为独立的，可以以任何顺序进行。关于设计方法的第 9 章～第 12 章是按顺序进行的，但是如果教师希望在各种设计技术之前介绍列表（数组），那么第 11 章（“数据集合”）中的内容可以很容易地提前。希望强调面向对象设计的教师不需要花费很多时间在第 9 章。第 13 章包含更多高级材料，可能会在最后介绍或穿插在整个课程的各个地方。

## 致谢

多年来，我教授 CS1 的方法受到了我读过并用于课堂的许多新教材的影响。我从这些书中学到的很多东西无疑已经融入了本书。有几位专家的方法非常重要，我觉得他们值得特别提及。A. K. Dewdney 一直有一个诀窍，找出说明复杂问题的简单例子。我从中借鉴了一些，装上了 Python 的新腿。我也感谢 Owen Astrachan 和 Cay Horstmann 的精彩教科书。我在第 4 章介绍的图形库直接受到 Horstmann 设计的类似库的教学经验启发。我也从 Nell Dale 那里学到了很多关于教授计算机科学的知识，当时我是得克萨斯大学的研究生，很幸运地担任了助教。

许多人直接或间接地为本书做出了贡献。我也得到了沃特伯格学院的同事（和前同事）的很多帮助和鼓励：Lynn Olson 在一开始就不动摇地支持，Josef Breutzmann 提供了许多项目想法，Terry Letsche 为第 1 版和第 3 版编写了 PowerPoint 幻灯片。

我要感谢以下阅读或评论第 1 版手稿的人：莫赫德州立大学的 Rus May、北卡罗莱纳州立大学的 Carolyn Miller、谷歌的 Guido Van Rossum、加州州立大学（Chico）的 Jim Sager、森特学院的 Christine Shannon、罗彻斯特理工学院的 Paul Tymann、亚利桑那大学的 Suzanne Westbrook。我很感激首都大学的 Dave Reed，他使用了第 1 版的早期版本，提供了无数有见地的建议，并与芝加哥大学的 Jeffrey Cohen 合作，为本版本提供了替代的章末练习。Ernie Ackermann 在玛丽华盛顿学院试讲了第 2 版。第 3 版是由位于 San Luis Obispo 的加州理工大学的 Theresa Migler 和我的同事 Terry Letsche 在课堂上试讲的。David Bantz 对草稿提供了反馈意见。感谢所有的宝贵意见和建议。

我也要感谢 Franklin, Beedle and Associates 的朋友，特别是 Tom Sumner、Brenda Jones 和 Jaron Ayres，他们把我喜爱的项目变成一本真正的教科书。本版献给 Jim Leisy 作为纪念，他是 Franklin, Beedle and Associates 的创始人，在第 3 版正要付梓时意外过世。Jim 是个了不起的人，兴趣非常广泛。正是他的远见、指导、不懈的热情和不断的激励，最终让我成为一名教科书作者，让这本书成功。

特别感谢所有我教过的学生，他们在教学方面给我很多教益。还要感谢沃特伯格学院批准我休假，支持我写书。最后但最重要的是，我要感谢我的妻子 Elizabeth Bingham，她作为编辑、顾问和鼓舞士气者，在我写作期间容忍我。

# 目　录

# 第 1 章　计算机和程序

## 学习目标

- 了解计算系统中硬件和软件各自的作用。
- 学习计算机科学家研究的领域和他们使用的技术。
- 了解现代计算机的基本设计。
- 了解计算机编程语言的形式和功能。
- 开始使用 Python 编程语言。
- 学习混沌模型及其对计算的影响。

## 1.1　通用机器

几乎每个人都用过计算机。也许你玩过计算机游戏，或曾用计算机写文章、在线购物、听音乐，或通过社交媒体与朋友联系。计算机被用于预测天气、设计飞机、制作电影、经营企业、完成金融交易和控制工厂等。

你是否停下来想过，计算机到底是什么？一个设备如何能执行这么多不同的任务？学习计算机和计算机编程就从这些基本问题开始。

现代计算机可以被定义为“在可改变的程序的控制下，存储和操纵信息的机器”。该定义有两个关键要素。第一，计算机是用于操纵信息的设备。这意味着我们可以将信息放入计算机，它可以将信息转换为新的、有用的形式，然后输出或显示信息，让我们解释。

第二，计算机不是唯一能操纵信息的机器。当你用简单的计算器来加一组数字时，就在输入信息（数字），计算器就在处理信息，计算连续的总和，然后显示。另一个简单的例子是油泵。给油箱加油时，油泵利用某些输入：当前每升汽油的价格和来自传感器的信号，读取汽油流入汽车油箱的速率。油泵将这个输入转换为加了多少汽油和应付多少钱的信息。

我们不会将计算器或油泵看作完整的计算机，尽管这些设备的现代版本实际上可能包含嵌入式计算机。它们与计算机不同，它们被构建为执行单个特定任务。这就是定义的第二部分出现的地方：计算机在可改变的程序的控制下运行。这到底是什么意思？

“计算机程序”是一组详细的分步指令，告诉计算机确切地做什么。如果我们改变程序，计算机就会执行不同的动作序列，因而执行不同的任务。正是这种灵活性，让计算机在一个时刻是文字处理器，在下一个时刻是金融顾问，后来又变成一个街机游戏。机器保持不变，但控制机器的程序改变了。

每台计算机只是“执行”（运行）程序的机器。有许多不同种类的计算机。你可能熟悉Macintosh、PC、笔记本计算机、平板计算机和智能手机，但不论实际上还是理论上，都有数千种其他类型的计算机。计算机科学有一个了不起的发现，即认识到所有这些不同的计算机具有相同的力量，通过适当的编程，每台计算机基本上可以做任何其他计算机可以做的事情。在这个意义上说，放在你的办公桌上的PC实际上是一台通用机器。它可以做任何你想要它做的事，只要你能足够详细地描述要完成的任务。现在它是一台强大的机器！

## 1.2 程序的力量

你已经知道了计算的一个要点：“软件”（程序）主宰“硬件”（物理机器）。软件决定计算机可以做什么。没有软件，计算机只是昂贵的镇纸。创建软件的过程称为“编程”，这是本书的主要关注点。

计算机编程是一项具有挑战性的活动。良好的编程既要有全局观，又要注意细节。不是每个人都有天赋成为一流的程序员，正如不是每个人都具备成为专业运动员的技能。然而，几乎任何人都可以学习如何为计算机编程。只要有一点耐心和努力，本书将帮助你成为一名程序员。

学习编程有很多好理由。编程是计算机科学的一个基本组成部分，因此对所有立志成为计算机专业人员的人都很重要。但其他人也可以从编程经验中受益。计算机已经成为我们社会中的常见工具。要理解这个工具的优点和局限性，就需要理解编程。非程序员经常觉得他们是计算机的奴隶。然而，程序员是真正的控制者。如果你希望成为一个更聪明的计算机用户，本书就是为你准备的。

编程也有很多乐趣。这是一项智力活动，让人们通过有用的、有时非常漂亮的创作来表达自己。不管你信不信，许多人确实爱好编写计算机程序。编程也会培养有价值的问题解决技能，特别是将复杂系统分解为一些可理解的子系统及其交互，从而分析复杂系统的能力。

你可能知道，程序员有很大的市场需求。不少文科生已经将一些计算机编程课程作为一种有利可图的职业选择。计算机在当今的商业世界中如此常见，以至于理解计算机和编程的能力可能就会让你在竞争中占据优势，不论你是何种职业。灵感迸发时，你就准备好写出下一个杀手级应用程序了。

## 1.3 什么是计算机科学

你可能会惊讶地得知，计算机科学不是研究计算机的。著名计算机科学家Edsger Dijkstra曾经说过，计算机之于计算机科学，正如望远镜之于天文学。计算机是计算机科学中的重要工具，但它本身不是研究的对象。由于计算机可以执行我们描述的任何过程，所以真正的问题是：“我们可以描述什么过程？”换句话说，计算机科学的根本问题就是“可以计算什么”。计算机科学家利用许多研究技术来回答这个问题。其中三种主要技术是设计、分析

和实验。

证明某个问题可以解决的一种方式就是实际设计解决方案。也就是说，我们开发了一个逐步的过程，以实现期望的结果。计算机科学家称之为“算法”。这是一个奇特的词，基本上意味着“菜谱”。算法设计是计算机科学中最重要的方面之一。在本书中，你会看到设计和实现算法的技术。

设计有一个弱点，它只能回答“什么是可计算的”。如果可以设计一个算法，那么问题是可解的。然而，未能找到算法并不意味着问题是不可解的。这可能意味着我只是不够聪明，或者碰巧还没有找到正确的想法。这就是引入分析的原因。

分析是以数学方式检查算法和问题的过程。计算机科学家已经指出，一些看似简单的问题不能通过任何算法解决。另一些问题是“难解的”（intractable）。解决这些问题的算法需要太长时间，或者需要太多存储器，因而没有实际价值。算法分析是计算机科学的重要组成部分，在整本书中，我们将探讨一些基本原则。第 13 章有不可解决和难解问题的例子。

一些问题太复杂或定义不明确，无法分析。在这种情况下，计算机科学家就依靠实验。他们实际实现一些系统，然后研究结果的行为。即使在进行理论分析时，也经常需要实验来验证和完善分析。对于大多数问题，底线是能否构建一个可靠的工作系统。通常我们需要对系统进行经验性测试，以确定这个底线已经满足。当你开始编写自己的程序时，会有很多机会观察你的解决方案的表现。

我已经从设计、分析和评估算法的角度定义了计算机科学，这当然是该学科的核心。然而，当今计算机科学家参与广泛的活动，所有这些活动都在计算这把大伞之下。一些例子包括移动计算、网络、人机交互、人工智能、计算科学（使用强大的计算机来模拟科学过程）、数据库和数据挖掘、软件工程、网络和多媒体设计、音乐制作、管理信息系统和计算机安全。无论在何处进行计算，计算机科学的技能和知识都有应用。

## 1.4 硬件基础

你不必知道计算机工作的所有细节，也能成为一名成功的程序员，但了解基本原理将有助于掌握让程序运行所需的步骤。这有点像驾驶汽车。了解一点内燃机知识，有助于解释为什么必须做一些事情，如加油、点火、踩油门等。你可以通过记住要做什么来学习驾驶，但拥有更多知识会让整个过程更容易理解。让我们花一点时间来看看计算机的内部构造。

虽然不同计算机在具体细节上会显著不同，但在更高的层面上，所有现代数字计算机是非常相似的。图 1.1 展示了计算机的功能视图。中央处理单元（CPU）是机器的“大脑”。这是计算机执行所有基本操作的地方。CPU 可以执行简单的算术运算，如两个数相加，也可以执行逻辑操作，如测试两个数是否相等。

存储器存储程序和数据。CPU 只能直接访问存储在“主存储器”（称为 RAM，即随机存取存储器）中的信息。主存储器速度快，但它也是易失性存储。也就是说，当电源关闭时，存储器中的信息会丢失。因此，还必须有一些辅助存储器，提供永久的存储。

在现代个人计算机中，主要的辅助存储器通常是内部的硬盘驱动器（HDD）或固态驱动器（SSD）。HDD 将信息以磁模式存储在旋转磁盘上，而 SSD 使用称为闪存的电子电路。

大多数计算机还支持可移动介质作为辅助存储器，如 USB 存储“棒”（也是一种形式的闪存）和 DVD 数字多功能光盘，后者以光学模式存储信息，由激光读取和写入。

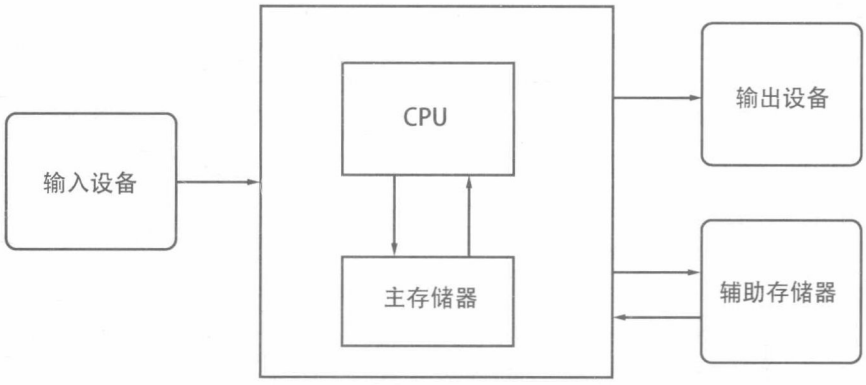



图 1.1 计算机的功能视图

人类通过输入和输出设备与计算机交互。你可能熟悉常见的设备，如键盘、鼠标和显示器（视频屏幕）。来自输入设备的信息由 CPU 处理，并可以被移动到主存储器或辅助存储器。类似地，需要显示信息时，CPU 将它发送到一个或多个输出设备。

那么，你启动最喜欢的游戏或文字处理程序时，会发生什么？构成程序的指令从（更）持久的辅助存储器复制到计算机的主存储器中。一旦指令被加载，CPU 就开始执行程序。

技术上，CPU 遵循的过程称为“读取—执行循环”。从存储器取得第一条指令，解码以弄清楚它代表什么，并且执行适当的动作。然后，取得并解码和执行下一条指令。循环继续，指令接着指令。这确实是所有的计算机从你打开它直到再次关闭时做的事情：读取指令、解码、执行。这看起来不太令人兴奋，是吗？但计算机能以惊人的速度执行这个简单的指令流，每秒完成数十亿条指令。将足够多的简单指令以正确的方式放在一起，计算机完成了惊人的工作。

## 1.5 编程语言

请记住，程序只是一系列指令，告诉计算机做什么。显然，我们需要用计算机可以理解的语言来提供这些指令。如果可以用我们的母语告诉计算机做什么，就像科幻电影中那样，当然很好。（“计算机，曲速引擎全速到达行星 Alphalpha 需要多长时间？”）计算机科学家在这个方向上取得了长足的进步。你可能熟悉 Siri（Apple）、Google Now（Android）和 Cortana（Microsoft）等技术。但是，所有认真使用过这种系统的人都可以证明，设计一个完全理解人类语言的计算机程序仍然是一个未解决的问题。

即使计算机可以理解我们，人类语言也不太适合描述复杂的算法。自然语言充满了模糊和不精确。例如，如果我说“I saw the man in the park with the telescope”，是我拥有望远镜，还是那个人拥有望远镜？谁在公园里？我们大多数时间都相互理解，因为所有人都拥有广泛的共同知识和经验。但即便如此，误解也是很常见的。

计算机科学家已经设计了一些符号，以准确无二义的方式来表示计算，从而绕过了这个问题。这些特殊符号称为编程语言。编程语言中的每个结构都有精确的形式（它的“语法”）和精确的含义（它的“语义”）。编程语言就像一种规则，用于编写计算机将遵循的指令。实际上，程序员通常将他们的程序称为“计算机代码”（computer code），用编程语言来编写算法的过程被称为“编码”（coding）。

Python 是一种编程语言，它是我们在本书中使用的语言[①]。你可能已经听说过其他一些常用的语言，如 C ++、Java、Javascript、Ruby、Perl、Scheme 和 BASIC。计算机科学家已经开发了成千上万种编程语言，而且语言本身随着时间演变，产生多个、有时非常不同的版本。虽然这些语言在许多细节上不同，但它们都有明确定义的、无二义的语法和语义。

上面提到的所有语言都是高级计算机语言的例子。虽然它们是精确的，但它们的设计目的是让人使用和理解。严格地说，计算机硬件只能理解一种非常低级的语言，称为“机器语言”。

假设我们希望让计算机对两个数求和。CPU 实际执行的指令可能是这样的：

```
将内存位置 2001 的数加载到 CPU 中
将内存位置 2002 的数加载到 CPU 中
在 CPU 中对这两个数求和
将结果存储到位置 2003
```

两个数求和似乎有很多工作，不是吗？实际上，它甚至比这更复杂，因为指令和数字以二进制符号表示（即 0 和 1 的序列）。

在 Python 这样的高级语言中，两个数求和可以更自然地表达为 $c = a + b$。这让我们更容易理解，但我们需要一些方法，将高级语言翻译成计算机可以执行的机器语言。有两种方法可以做到这一点：高级语言可以被“编译”或“解释”。

“编译器”是一个复杂的计算机程序，它接受另一个以高级语言编写的程序，并将其翻译成以某个计算机的机器语言表达的等效程序。图 1.2 展示了编译过程的框图。高级程序被称为“源代码”，得到的“机器代码”是计算机可以直接执行的程序。图中的虚线表示机器代码的执行（也称为“运行程序”）。

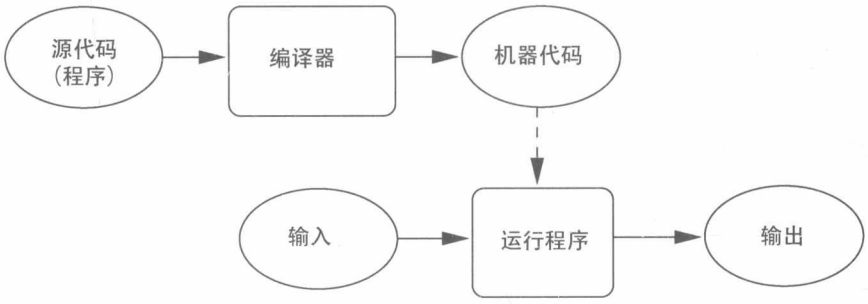



图 1.2 编译高级语言

“解释器”是一个程序，它模拟能理解高级语言的计算机。解释器不是将源程序翻译成

[①] 本书的这个版本使用 Python 3.4 版本开发和测试。Python 3.5 现在可用。如果你的计算机上安装了早期版本的 Python，则应升级到最新的稳定版 3.x，以便尝试这些例子。