HZ BOOKS
华章教育

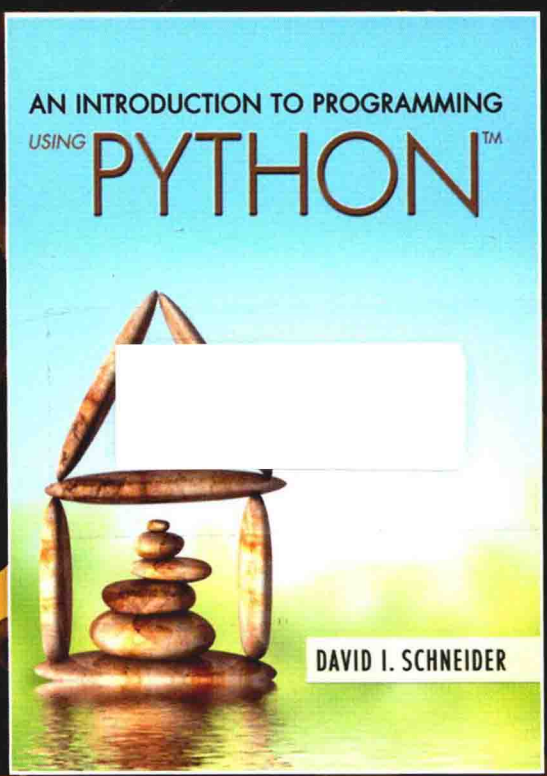
计 算 机 科 学 丛 书

PEARSON

Python程序设计

[美] 戴维 I. 施奈德（David I. Schneider） 著
车万翔 等译

An Introduction to Programming Using Python



机械工业出版社
China Machine Press

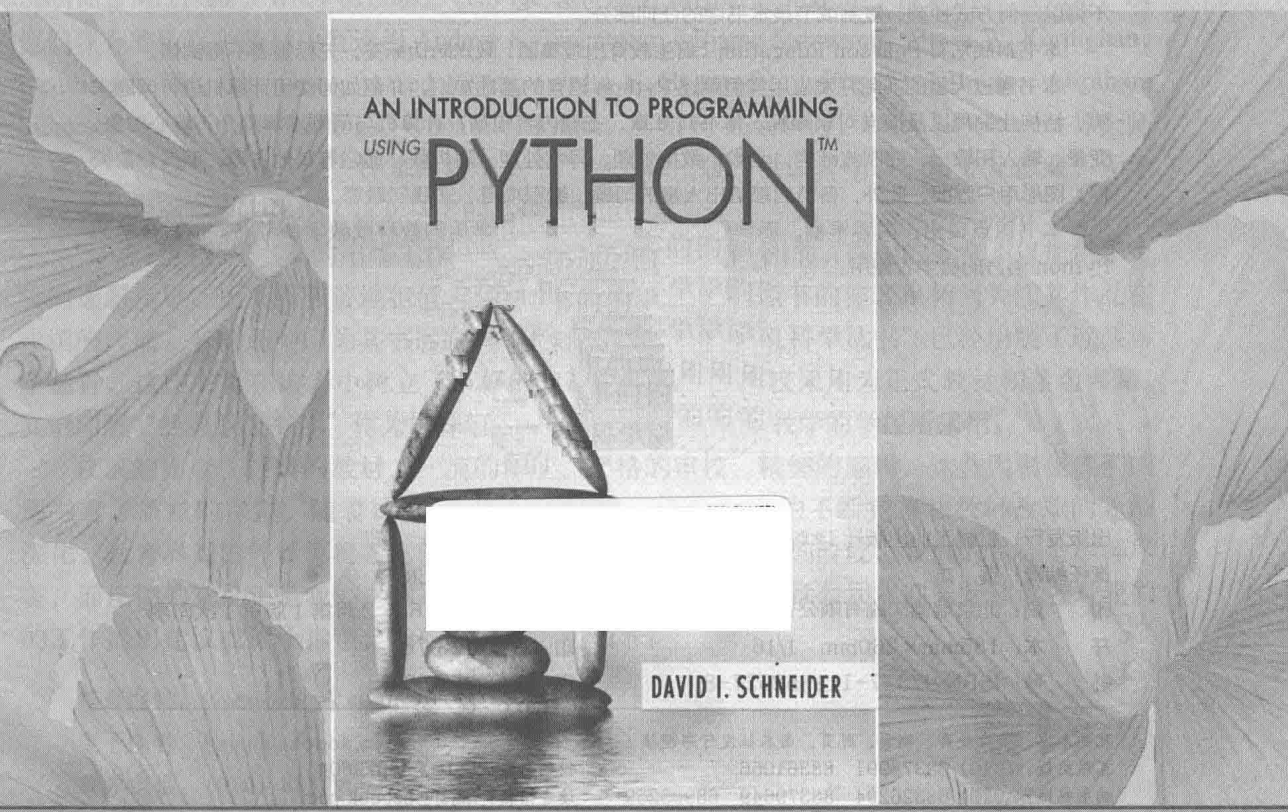
计 算 机 科 学 丛 书

Python程序设计

[美] 戴维 I. 施奈德 (David I. Schneider) 著

车万翔 等译

An Introduction to Programming Using Python



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Python 程序设计 /（美）施奈德（Schneider, D. I.）著；车万翔等译 . —北京：机械工业出版社，2016.2
（计算机科学丛书）
书名原文：An Introduction to Programming Using Python

ISBN 978-7-111-52627-8

I. P… II. ① 施… ② 车… III. 软件工具－程序设计－教材 IV. TP311.56

中国版本图书馆 CIP 数据核字（2016）第 014229 号

本书版权登记号：图字：01-2015-5403

本书通过大量的实际开发应用实例阐述 Python 语言的基础知识，介绍如何使用计算机进行问题求解、结构化编程以及面向对象编程。本书共 8 章，主要内容包括：计算机与问题求解简介，核心对象、变量、输入和输出，控制流结构，函数，数据处理，异常处理、随机数、递归等其他主题，面向对象编程，图形用户界面。此外，各节后都给出大量的习题、编程项目、实践问题等。

本书重点突出，内容丰富，适合作为计算机及相关专业学生的教材或教学参考书，也适合学习 Python 语言的初学者使用。

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：缪　杰	责任校对：董纪丽
印　刷：北京瑞德印刷有限公司	版　次：2016 年 2 月第 1 版第 1 次印刷
开　本：185mm×260mm　1/16	印　张：23.75
书　号：ISBN 978-7-111-52627-8	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，Brain 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

乔布斯曾经说过："每个人都应该学习编程，因为它教你如何思考。"计算已经成为继实验归纳和理论推演之后的一种新型的科研手段，越来越多的学科被冠以计算之名，如计算生物学、计算物理学、计算语言学、计算社会学等等。以计算社会学为例，过去社会学研究一般是以问卷调查的方式进行。现在随着互联网、尤其是社交网络的蓬勃发展，社会学家已经可以用大数据计算的方式来研究社会学问题了，而且往往会获得更真实有趣的结论。因此，无论你学习的是什么专业，都应该开始学习编程。

然而，过去很长一段时间，很多人都是以C语言作为入门编程语言。C语言具有代码简洁紧凑、执行效率高、贴近硬件、可移植性好等优点，广泛应用于开发系统软件、嵌入式软件等。但是，随着计算机技术的发展，我们可以使用它来解决越来越多的问题，其应用场合也不局限于计算机学科。在这种情况下，C语言这种"低级"语言已经越来越难适应计算机发展的潮流，我们需要一种简单易学且能解决更实际问题的语言。Python语言恰好能弥补C语言的不足，它不但功能强大，而且易学易用，有人甚至说，读Python代码就像在读英语。

目前，市面上很多Python语言的图书多是面向有一定编程基础、想要尽快掌握一门新语言的读者。因此，它们更注重介绍Python语言的语法和应用特性。作为教科书，本书面向零基础编程的读者，以Python语言作为工具，介绍用计算机解决实际问题的基本思想和方法。因此，在内容组织上，不是面面俱到地介绍Python语言的诸多细节，而是只介绍那些最基本、最重要的概念，并将重点放到实际的应用上，为此提供了大量的编程项目、习题和实践问题。同时，本书还采用了大量真实数据的例子，通过解决这些问题，可以使读者亲身感受到编程魅力之所在。

我们要感谢对本书的翻译有所襄助的老师和学生们。本书由车万翔、赵妍妍、叶麟和袁永峰四位主译，他们都是哈尔滨工业大学的老师，并面向计算机学院主讲"高级语言程序设计（Python）"课程。李胜秋、徐梓翔也对本书部分内容的初译做了很多工作，机械工业出版社华章公司策划编辑朱劼、本书责任编辑缪杰在本书的整个翻译过程中提供了许多帮助，在此予以衷心感谢。

译文虽经多次修改和校对，但由于译者的水平有限，加之时间仓促，疏漏及缺点、错误在所难免，我们真诚地希望读者不吝赐教，不胜感激之至。

车万翔

2015年10月于哈尔滨工业大学

自 20 世纪 90 年代诞生以来，Python 已经成为软件行业中应用最为广泛的编程语言之一。学习第一门编程语言的学生也发现，Python 是理解计算机程序开发的理想工具。

这本书的写作目标如下：

1. 重点突出。集中介绍重要的主题，而不是泛泛地覆盖许多主题。

2. 使用学生耳熟能详的例子和相关习题。经常使用真实的数据。例子简洁明了，尽量向学生展示编程技术，而不去分散他们的注意力。

3. 通俗易懂，内容丰富。首先讨论每个主题的要点，然后再以注释的形式展示次要的细节。

4. 教授好的编程实践，与现代编程方法相接轨。全面讨论问题求解技术、结构化编程以及面向对象编程。

5. 深入介绍计算机的主要应用。

本书特色

编程项目。从第 2 章开始，每章都包含编程项目。编程项目反映了计算机的不同使用方法。数量多、难度不一的编程项目使本书适用于不同兴趣和层次的学生。后面章节的一些编程项目可以留作学期末项目。

大多数章节的习题。教授编程的每个章节都有一个习题集。这些习题既可以增强学生对这些章节关键思想的理解，又为学生探索更多的应用提出了挑战。大部分习题集需要学生跟踪程序、找到错误以及编写程序。本书中，除了 6.3 节以及第 8 章外，奇数编号习题的答案在本书的最后给出。几乎每个编程习题以及编程项目都配有一个可能的输出。

实践问题。在每节的习题之前，都有一些精选的实践问题。习题后面是完备的答案。实践问题经常集中于那些可能易混淆的要点上，在学生深入思考之后，就会透彻理解这些要点。在做习题之前，读者应该认真地试着解一下这些实践问题并且研究它们的答案。

注释。在每节最后，都有一些扩展以及新的主题，它们被置于“注释”部分，以便这些内容不会影响行文的流畅性。

关键术语和概念。在第 2 章到第 8 章中，在每章的最后给出关键术语和概念（以及示例）。

应用问题指南。该部分提供了程序的索引，讨论多种主题，包括商务、经济、数学以及体育。

解题手册。学生解题手册包括奇数编号习题的答案（不包括编程项目的答案）。教师解题手册包括所有习题和编程项目的答案。两本解题手册都是 PDF 格式的，并且能够从 Pearson 的网站上下载。

源代码和数据文件。全部示例的程序以及习题所需的数据文件都能从 Pearson 的网站上下载。

教师和学生资源[⊖]

教师资源

- 测试项文件
- PPT 课件
- 教师解题手册
- 所有示例的程序以及习题和编程项目的答案（习题所需的数据文件在 Programs 文件夹下）

学生资源

- 学生解题手册
- 示例的程序（习题所需的数据文件在 Programs 文件夹下）

致谢

在本书写作过程中，许多优秀的研究者和程序员给予了富有建设性的意见和建议，我由衷地对他们的贡献表示感谢。以下审阅者为本书的编写提供了大量有价值的建议：

Daniel Solarek, 托莱多大学
David M. Reed, 首都大学
Debraj De, 佐治亚州
Desmond Chun, 夏波学院
Mark Coffey, 科罗拉多矿业大学
Randall Alexander, 查尔斯顿学院
Vineyak Tanksale, 鲍尔州立大学
Zhi Wei, 新泽西理工学院

很多人为本书的成功出版提供了帮助。我要感谢 Pearson 的团队，本书的出版离不开他们的支持和鼓励，特别要感谢计算机科学部门程序管理经理 Carole Synyder，计算机科学部门编辑助理 Kelsey Loanes 和产品经理主管 Scott Disanno。

我要感谢 Jacob Saina 在本书编写过程中各个阶段的帮助。出版编辑 Pavithra Jayapaul 和 Greg Dulles 为本书的出版做了许多工作，使之能按计划推进。我要感谢阿克伦大学的 Kathy Liszka 博士，他为本书制作了题库，感谢莱托诺大学的 Steve Armstrong 博士，他为本书制作了配套的 PPT 课件。本书在出版过程中也离不开来自 Jouve India 的 Shylaja Gattupalli 的帮助。

特别要感谢我的主编 Tracy Johnson。她的想法和热情为本书的筹备工作提供了巨大的帮助。

David I. Schneider
dis@alum.mit.edu

⊖ 关于本书教师资源，用书教师可向培生出版集团北京代表处申请，电话：010-57355169/57355171，电子邮件：service.cn@pearson.com。学生或读者如需要学生资源，请联系培生出版集团北京代表处购买 access code，登录原出版社网站下载。——编辑注

第 1 章
An Introduction to Programming Using Python

计算与问题求解简介

1.1 计算与 Python 简介

本书介绍如何使用计算机进行问题求解。虽然编程语言用 Python，但是原理适用于大部分现代编程语言。许多示例和习题展示了在真实世界中是如何使用计算机的。下面是你可能提出的计算机和编程方面一些的问题。

问：我们如何与计算机沟通？

答：使用编程语言与计算机沟通。最低级别的是机器语言（machine language），其能被微处理器直接理解，但是很难为人所理解。Python 是高级语言（high-level language）。它由人能理解的指令组成，如 print（输出）、if（如果）、input（输入）等。其他著名的高级语言如 Java、C++、Visual Basic 等。

问：如何让计算机执行复杂的任务？

答：任务可分解为一系列指令，称作**程序**（program），其能以编程语言表示。程序的大小从两三条指令到几百万条指令不等。执行指令的过程称作运行（running）程序。

问：为什么使用 Python 作为编程语言？

答：许多人认为 Python 是教初学者编程最好的语言，我们表示赞同。Python 也被主要软件公司所使用。Python 功能强大、易写易读、容易下载和安装，它能在 Windows、Mac 和 Linux 等操作系统上运行。

问：Python 的名字是怎么来的？

答：它的名字来源于英国喜剧剧团 Monty Python。Python 的创始人 Guido van Rossum（荷兰人）是该剧团的粉丝。

问：本书使用编辑器 IDLE 来生成程序。IDLE 是如何得名的？

答：IDLE 表示集成开发环境（Integrated DeveLopment Environment）。（一些人认为该名字是为了向 Monty Python 剧团的创始成员 Eric Idle 致敬。）IDLE 编辑器有许多特性（如对代码着色、格式化辅助等）以帮助程序员。

问：Python 被认为是一种解释型语言。那么什么是解释型语言？

答：解释型语言使用一个叫作解释器（interpreter）的程序，一次将高级语言的一条语句翻译为机器语言，然后运行这段程序。解释器会发现几种类型的错误，一旦遇到一个错误，将终止程序的运行。

问：术语“程序员”和“用户”是什么意思？

答：程序员（也称为开发者）是指在计算机上编写程序来解决问题的人。在分析问题并

制定解决方案之后，程序员编写和测试相应的程序，这段程序用来指导计算机如何实现该方案。程序可能被程序员或者其他人运行多次。用户是任何运行该程序的人。在使用本书的过程中，你既是一个程序员，也是一个用户。

问： 术语“代码”是什么意思？

答： 程序员写的 Python 指令称为代码。编写程序的过程通常称作编码（coding）。

问： 所有的程序都具有某些共性吗？

答： 大部分程序做三件事：接受数据、操作数据和产生结果。这些操作称为输入（input）、处理（processing）和输出（output）。输入数据可能存放于程序中、位于磁盘上或者由用户提供，以响应程序运行时计算机的需求。输入数据的处理发生在计算机内部，可能花费几毫秒到几小时。输出数据显示在屏幕上、打印到打印机上或者记录在磁盘上。以一个计算营业税的程序作为一个简单的例子，输入数据是物品的售价，处理指将售价和营业税率进行相乘，输出数据是相乘的结果，即所需缴纳的营业税。

问： 术语“硬件”和“软件”是什么含义？

答： 硬件指计算机的物理部件，包括全部的外围设备、中央处理器（CPU）、磁盘驱动器以及全部的机电设备。软件指的是程序。

问： 问题是如何使用程序加以解决的？

答： 问题求解的过程是：通过仔细地阅读问题以确定给定的数据和所需的输出是什么，然后设计一步一步处理给定数据的过程，最后产生所需的输出。

问： 包括 Python 在内的许多编程语言使用基于零的计数系统。什么是基于零的计数系统？

答： 在基于零的计数系统中，从 0 开始，而不是从 1 开始计数。例如，在单词“code”中，“c”应该是第 0 个字母，“o”是第 1 个字母，以此类推。

问： 学习 Python 有什么先决条件？

答： 你应该熟悉在计算机上文件夹（也称作目录）和文件是如何管理的。文件位于存储设备上，如硬盘、U 盘、CD 和 DVD 等。传统上，个人计算机的主要存储设备是硬盘和软盘。因此，磁盘（disk）这个词经常用于表示任何存储设备。

问： 本书中一个开发好的程序示例是什么？

答： 图 1-1 展示了第 3 章中一个程序的可能输出结果。当其首次执行时，显示语句“Enter a first name:”。在用户键入一个名字并且按下回车（〈Enter〉或者〈return〉）键后，全部具有该名字的居民都会显示出来。

```
Enter a first name: James
James Madison
James Monroe
James Polk
James Buchanan
James Garfield
James Carter
```

图 1-1 第 3 章中一个程序的可能输出

问：程序员是如何生成上述程序的？

答：对于该程序，程序员编写了大概 10 行代码，以搜索一个名为 USpres.txt 的文件，并提取所需的姓名。

问：按键显示的约定是什么？

答：组合键〈key1+key2〉的含义是“按住〈key1〉，然后按下〈key2〉”。组合键〈Ctrl+C〉将所选的内容放入剪贴板。组合键〈key1/key2〉的含义是“按下并松开〈key1〉，然后按下〈key2〉”。组合键〈Alt/F〉打开菜单栏上的“文件”菜单。

问：如何获取本书中的程序示例？

答：阅读前言部分，获取如何从 Pearson 网站上下载示例程序的信息。

问：新的程序存储在哪里？

答：在编写第一个程序之前，应该创建一个专用的文件夹存储程序。

1.2 程序开发周期

1.1 节提过，硬件是指计算机系统中的机电设备（如显示器、键盘和 CPU 等），软件是指令的集合，也称作**程序**，用其指挥硬件。程序用于在计算机上解决问题或者执行任务。程序员将问题解决方案或者任务翻译成计算机能够理解的一种语言。当我们写程序时，我们必须牢记计算机只能做我们指示它做的事情。因此，在编写指令时，我们必须非常小心和认真。

1.2.1 在计算机上执行任务

编写指令执行任务的第一步是确定**输出**是什么——也就是说，明确此任务应该产生什么。第二步是明确获得输出所需的数据或**输入**。最后一步是确定如何**处理**输入以获得输出——也就是说，确定使用什么公式或者做事的方法以获得输出。

此解决问题的方法与在代数课上解决现实世界问题所使用的方法相同。例如，考虑以下代数问题：

如果一辆汽车在 2 小时内行驶了 50 英里[⊖]，它的速度是多少？

第一步是确定所需答案的类型。答案应该是一个以英里 / 每小时为单位的速度值（输出）。获取答案所需的信息是该车移动的距离和时间（输入）。公式

速度 = 距离 / 时间

用于处理行驶的距离和所花费的时间，以确定速度。也就是说，

速度 = 50 英里 /2 小时
= 25 英里 / 小时

该问题求解过程的图示化表示如图 1-2 所示。

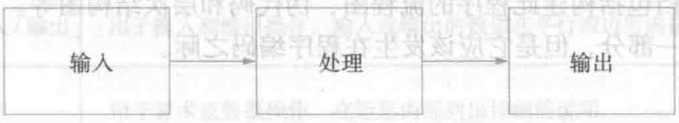


图 1-2 问题求解过程

⊖ 1 英里 =1 609.344 米

我们将所希望获得的作为输出，然后获取所需的输入并且处理该输入以产生所期望的输出。

后续章节将讨论如何编写程序实现上述操作。但是首先我们看一下编写程序的一般过程。

1.2.2 程序规划

一个烘焙食谱提供了规划的范例。原料和所需的数量由想要烘焙的东西所决定。也就是说，输出决定输入和处理。食谱或者方案，减少了你可能犯的错误。虽然很难想象一个建桥或者建工厂的建筑师会没有一个详细的计划，但是许多程序员（特别是第一门编程课上的学生）试图不事先制定一个细致的计划就编写程序。问题越复杂，计划就应该越复杂。如果你精心设计出一个一步步的计划并且在实际写程序之前测试它，你将在程序上花少得多的时间。

许多程序员使用一系列步骤作为他们程序的计划，称作**软件开发生命周期**。以下步骤将使你更有效地利用时间，并且帮助你设计能产生所期望输出的、零错误的程序。

1. **分析**：定义问题。

明确你理解程序应该做什么——也就是说，输出应该是什么。对给定的数据（或输入）以及输入和期望输出之间的关系做到心中有数。

2. **设计**：制定解决问题的计划。

找到解决问题的一系列精确的逻辑步骤。这一系列步骤称作**算法**（algorithm）。包括明显的步骤在内的每个细节都应出现在算法中。下一节将讨论三个用于制定逻辑计划的方法：流程图、伪代码和层次结构图。这些工具帮助程序员将一个问题分解为计算机能执行的一系列小任务，从而解决问题。制定计划也包含手工使用有代表性的数据来测试算法的逻辑性，以确定算法的正确性。

3. **编码**：将算法翻译为编程语言。

编码（coding）是编写程序的术语。在此步骤中，把使用 Python 编写的程序输入计算机中。程序员使用步骤 2 设计的算法以及 Python 的知识。

4. **测试和纠错**：定位并删除程序中的任何错误。

测试是找到程序中错误的过程。（程序中的错误称作 bug，测试和纠错通常称作**调试**。）随着程序键入，Python 会指出程序中的某些类错误。当程序执行的时候，Python 会检测出一些其他类型的错误——然而，许多错误，如输入错误、算法的瑕疵或者 Python 语言规则的不正确使用等，并不能被发现，只能通过认真的检测工作才能得到纠正。例如，本应使用乘号的地方使用了加号。

5. **完成文档**：组织全部描述程序的材料。

文档的目的是让其他人或者程序员将来能理解此程序。内部文档（注释）包括程序中不执行的语句，但是指出了程序各部分的目的。文档也可能包括程序能做什么事情以及如何使用它的详细描述（例如，所期望输入的类型）。对于商用程序，文档包括使用手册和在线帮助。其他类型的文档包括构建此程序的流程图、伪代码和层次结构图等。虽然文档被列为程序开发周期的最后一部分，但是它应该发生在程序编码之际。

1.3 编程工具

本节讨论一些特殊的算法，并描述三个用于将算法转化为计算机程序的工具：流程图、伪代码和层次结构图。

你每天都使用算法进行决策以及执行任务。例如，当你寄信时，你必须决定在信封上贴几张邮票。一个经验法则是，每五页或不足五页信纸使用一张邮票。假如一个朋友让你决定在信封上贴几张邮票，下面的算法将完成此任务：

1. 获取信纸的页数，称其为 Sheets。（输入）
2. 将 Sheets 除以 5。（处理）
3. 如果需要，将商向上取整，称其为 Stamps。（处理）
4. 返回 Stamps 数目。（输出）

上述算法将信纸的数目（Sheets）作为输入，处理此数据，产生所需邮票的数目（Stamps）作为输出。我们可以用有 16 页信纸的信测试此算法。

1. 获取信纸的页数，Sheets = 16。
2. 将 16 除以 5，得到 3.2。
3. 3.2 向上取整得 4，Stamps = 4。
4. 返回答案，4 张邮票。

此问题求解的例子如图 1-3 所示。

图 1-3　邮资问题的问题求解过程

三种流行的程序设计工具如下。

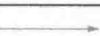
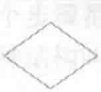
流程图：图形化地描述完成一项任务所需的逻辑步骤，并且展示各个步骤之间的关系。

伪代码：使用类似英语的短语以及一些 Python 语句概述一项任务。

层次图：展示程序不同部分之间的关系。

1.3.1　流程图

流程图由通过箭头连接的特殊几何符号构成。在每个符号内部，是描述该步动作的一个短语。符号的形状指明待发生的操作的类型。例如，平行四边形表示输入或输出。连接几何符号的箭头称作**流程线**（flowline），展示了步骤的进程。流程图应该从上“流”到下。虽然流程图中使用的符号是规范化的，但是每个符号的更多细节并没有什么标准。

符　号	名　称	含　义
	流程线	用于连接各符号，指明逻辑流
	终端	用于表示一个任务的开始和结束
	输入 / 输出	用于输入和输出操作。输入或输出的数据在平行四边形内描述
	处理	用于算术或数据操作。在矩形内部列出详细的说明
	决策	用于任何逻辑或比较操作。输入 / 输出和处理符号有一个进入流程线和一个退出流程线，而决策符号有一个进入路径和两个退出路径。路径的选择依赖于菱形内问题的答案是“yes”还是“no”

（续）

符 号	名 称	含 义
◯	连接符	用于连接不同的流程线
---▭	标注	用于提供关于流程图符号的额外信息

流程图符号表已经被美国国家标准协会（ANSI）所采纳。图 1-4 展示了邮资问题的流程图。

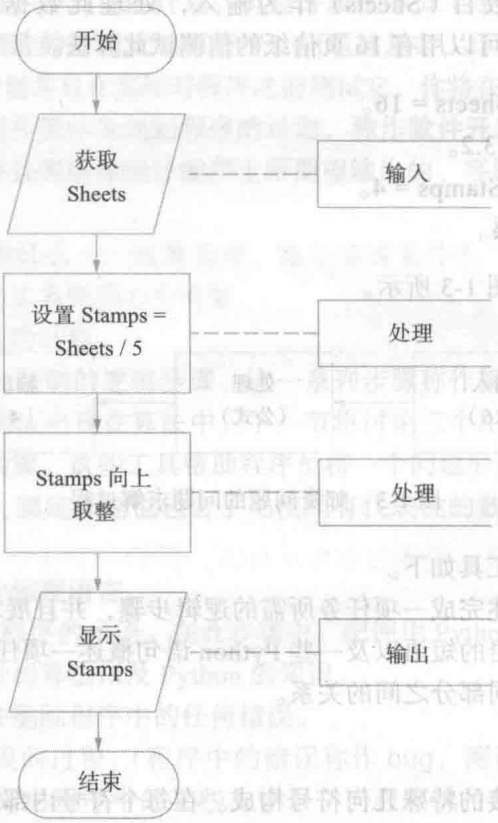


图 1-4 邮资问题的流程图

使用流程图来制定计划的主要好处是，其提供了任务的一个图形化表示，因此使得逻辑更容易理解。我们可以清晰地看到每个步骤以及步骤之间是如何连接的。主要不足是当一个程序非常大的时候，流程图可能需要画在多页上，使得他们很难阅读和修改。

1.3.2 伪代码

伪代码是实际计算机代码的英文简化版本[⊖]。在流程图中使用的几何符号被类似英语的语句所替代，它们概述了整个过程。结果就是，伪代码比流程图更像计算机代码。伪代码使得程序员能够专注于解决问题的步骤，而不是如何使用计算机语言。程序员能够使用类 Python 的形式描述算法，而不用为 Python 规则所限。当伪代码完成时，它们能够很容易地翻译为 Python 语言。

⊖ 伪代码也可以使用其他语言书写，如汉语。——译者注

邮资问题的伪代码如图 1-5 所示。

```
程序：决定一封信恰当的邮票数目。
获取信纸的页数（Sheets）                （输入）
将邮票数目置为 Sheets/5                 （处理）
将邮票数目向上取整                      （处理）
显示邮票数目                            （输出）
```

图 1-5 邮资问题的伪代码

伪代码有几个好处。它很紧凑，可能不会向流程图一样连续许多页。伪代码也看起来很像真实的代码，因此为许多程序员所偏爱。

1.3.3 层次结构图

我们将讨论的最后一个编程工具是**层次结构图**（Hierarchy Chart），其展示了程序结构的全貌。层次结构图也称作结构图、HIPO（Hierarchy plus Input-Process-Output）图、自顶向下图、VTOC（Visual Table of Contents）图。所有这些名字都指类似于公司组织结构图那样的规划图。

层次结构图描绘了程序的组织框架，而不关心具体的处理逻辑。它描述了程序的每一个部分在做什么，并且描述了程序的一个部分和其他部分之间的关系。然而，忽略了程序内部的处理细节。我们应该按照从上至下、从左至右的顺序来阅读层次结构图。程序的一个部分可能被划分成几个子部分，在图中以分支的形式来描述。通常，当一个部分的子部分的工作都执行完毕后，下一个待执行的部分是这个部分右面的那个部分。我们可以通过层次结构图，很快地了解到程序分为几个部分，当前执行到哪一个部分。图 1-6 展示了邮资问题的层次结构图。

图 1-6 邮资问题的层次结构图

层次结构图最主要的作用是用来做程序的初期规划。一旦我们划分出程序的主要部分，我们就可以知道在总体上需要做什么。接着，在此基础上，我们再进一步将每个部分用流程图或者伪代码细化。这个过程就是所谓的**分治**（divide-and-conquer）法。

1.3.4 判断结构

为了解决邮资问题，我们设计了一系列指令来获取数据、执行运算、显示结果。其中每一个步骤都是按顺序执行的，也就是说，我们一行接着一行执行指令，并不跳过任何一行。这种结构叫作**顺序结构**（sequence structure）。然而，许多问题需要判断，来决定是否要执行