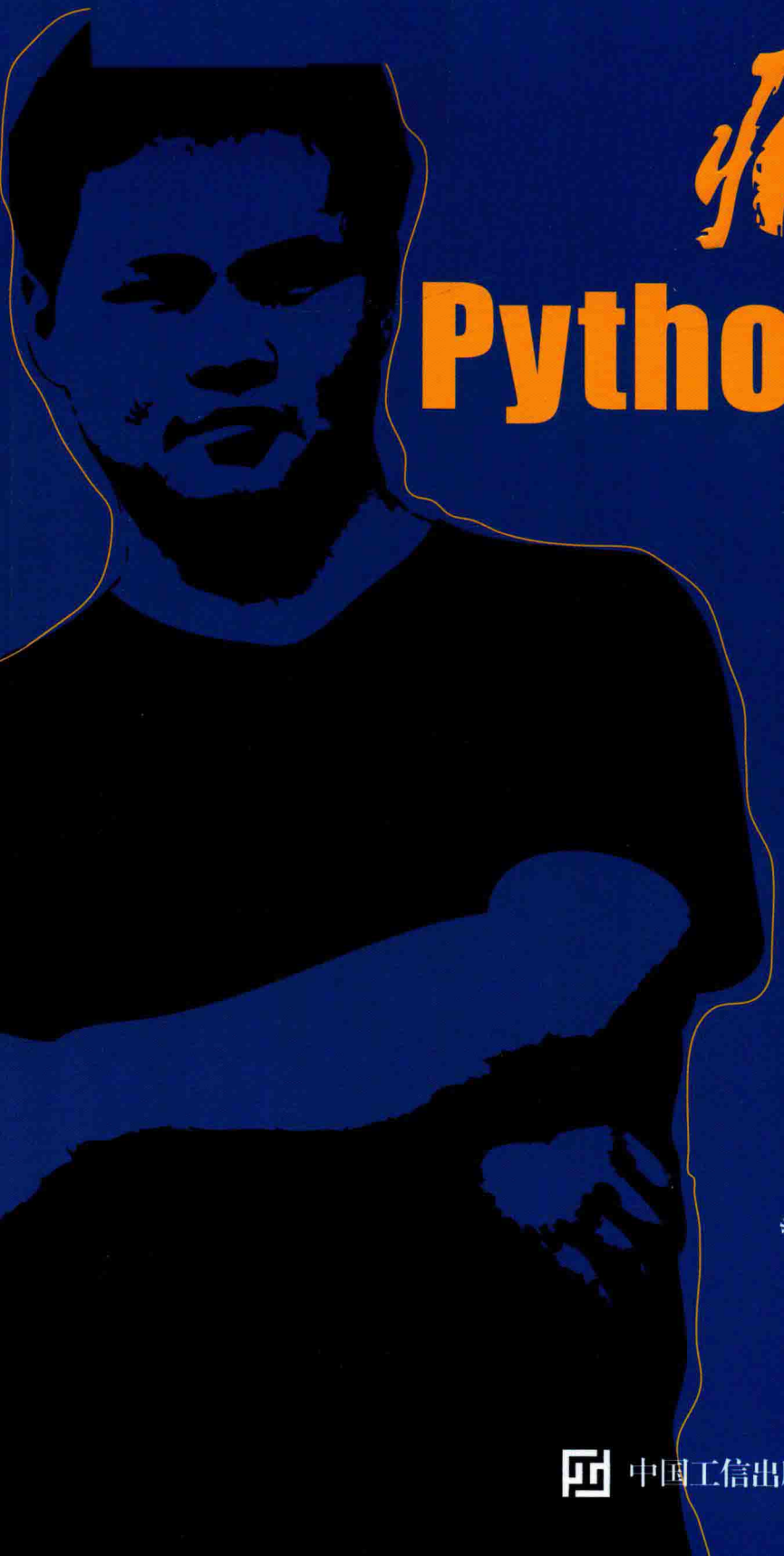
Broadview
www.broadview.com.cn

FK 疯狂软件教育 www.fkjava.org 标准教材

疯狂 Python讲义

李刚 编著



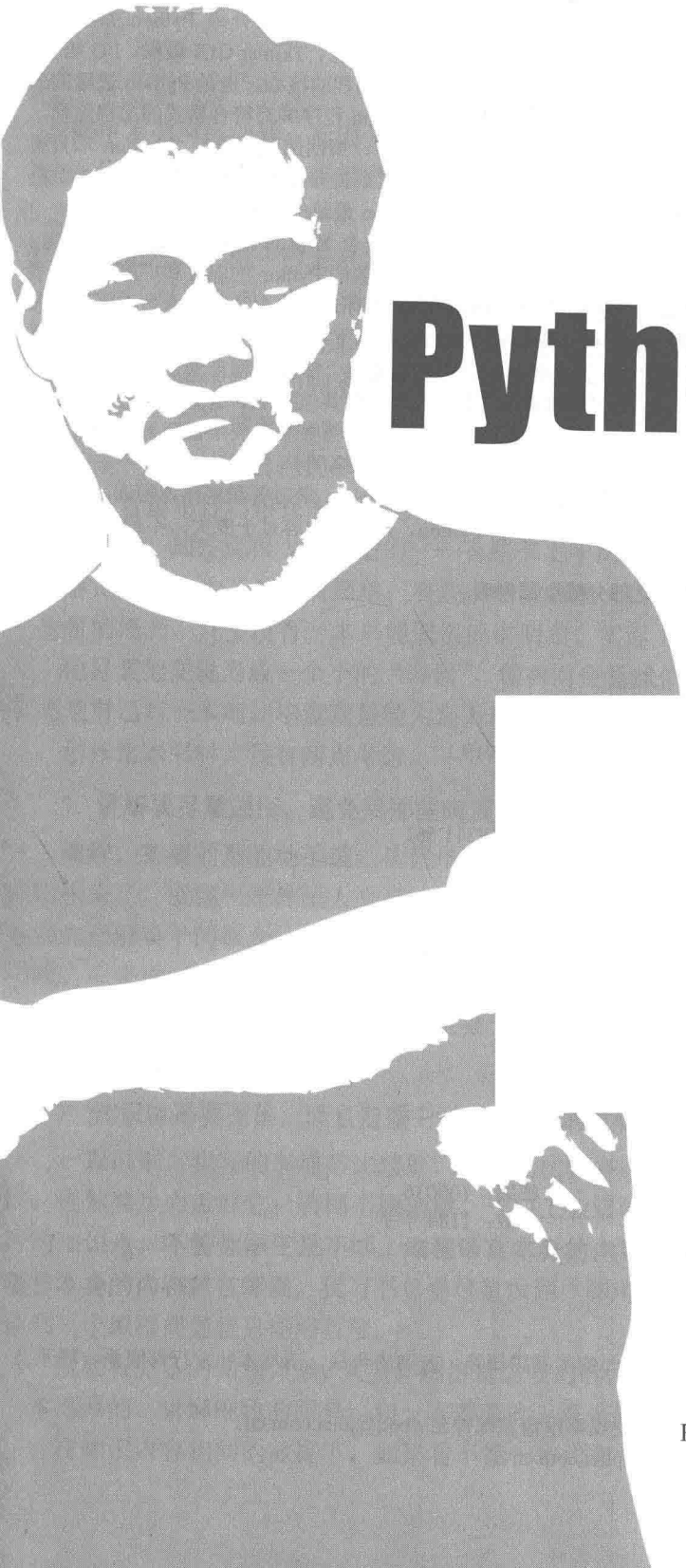
疯狂源自梦想

技术成就辉煌

疯狂源自梦想
技术成就辉煌

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn



疯狂 Python讲义

李刚 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

这既是一本适合初学者入门 Python 的图书（一个 8 岁的小朋友在本书出版前已学习了很多章节并动手写出了自己的程序）；也是一本适合 Python 就业的图书，因为本书涵盖了网络编程、数据分析、网络爬虫等大量企业实用的知识。

本书全面而深入介绍了 Python 编程的相关内容，全书内容大致可分为四个部分。第一部分系统讲解 Python 的基本语法结构、函数编程、类和对象、模块和包、异常处理等核心语法；第二部分主要介绍 Python 常用的内置模块和包，包括 Python 的 JSON 和正则表达式支持、容器相关类、collections 包、Tkinter GUI 编程、I/O 编程、数据库编程、并发编程、网络通信编程等内容，这部分内容既是掌握 Python 编程的核心，也是 Python 进阶的关键基础；第三部分主要介绍 Python 开发工程化方面的内容，包括如何为 Python 程序编写符合格式的文档注释、提取文档注释生成帮助文档、为 Python 程序编写测试用例、程序打包等内容；第四部分则属于“Python 项目实战”，这部分引入了 pygame、Matplotlib、Pygal、Scrapy 这些第三方包，通过项目介绍 Python 游戏开发、大数据展示、网络爬虫等热门技能，尤其是网络爬虫和大数据展示，均是当下 Python 最热的就业岗位。

与“疯狂体系”图书类似，虽然我会尽量让本书的讲解通俗易懂（毕竟一个 8 岁的小朋友也能阅读此书），但我创作“疯狂体系”图书的初衷从来就不是“简单”和“入门”，本书所覆盖的 Python 的深度和广度，是很多书籍所不能比拟的。本书涉及大量实用案例开发：五子棋游戏、画图板、桌面弹球、合金弹头、大数据展示、基于网络的各国人均 GDP 对比、基于爬虫的招聘热点分析、基于爬虫的高清图片下载、基于 Scrapy+Selenium 的微博登录……设计这些案例的初衷不是“简单”和“入门”，而是让读者学以致用、激发编程自豪感，进而引爆内心的编程激情。因此对于那些仅图简单的读者，建议不要选择此书。本书课后习题共包括 110 道循序渐进的 Python 练习题（面试题），读者可通过这些练习题巩固所学、为面试做准备。如果读者需要获取关于课后习题的解决方法、编程思路，可以登录 http://www.crazyit.org 站点或关注“疯狂图书”微信服务号。

本书为所有打算深入掌握 Python 编程的读者而编写，适合各种层次的 Python 学习者和工作者阅读，也适合作为大学教育、培训机构的 Python 教材。但如果只是想简单涉猎 Python，则本书内容过于庞大，不适合阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

疯狂 Python 讲义 / 李刚编著. —北京：电子工业出版社，2019.1
ISBN 978-7-121-35197-6

Ⅰ. ①疯… Ⅱ. ①李… Ⅲ. ①软件工具—程序设计 Ⅳ. ①TP311.561

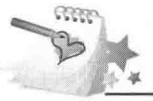
中国版本图书馆 CIP 数据核字（2018）第 234007 号

策划编辑：张月萍
责任编辑：葛　娜
印　　刷：北京京科印刷有限公司
装　　订：北京京科印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：39.75　　字数：1184 千字
版　　次：2019 年 1 月第 1 版
印　　次：2019 年 1 月第 1 次印刷
印　　数：5000 册　　定价：118.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。



前　言

创作本书纯属偶然，起因是我儿子想学编程。当他想报编程兴趣班时，居然没报上、满额了，而他是一个对生活充满好奇的小孩，望着他满是失落的眼睛，我想不如我来教吧，毕竟我曾经教了那么多别人的孩子。

我的想法是：挑一门上手足够容易的语言来教，毕竟他只是一个 8 岁的小孩。首先排除了 Java 和 C，虽然我自己用这两种语言比较多，但对于小孩来说，上手它们显得有些枯燥；也考虑过 Swift 或 Kotlin，能迅速带着做点手机小游戏比较酷，后来又觉得搭建运行环境有点费事；还是选一种能解释执行的脚本语言吧，我想到了 Python 或 Ruby，后来又了解到那个兴趣班教的就是 Python，那就选 Python 吧。

于是，他开始了自己的 Python 学习之旅。而我完全被困住了：每当他遇到一点问题就要来问我。这肯定不行，得找本书让他自己看，这样他就不用来烦我了。我是一个非常挑剔的人，找了不少书，却发现很少有合适的——有些书上手简单，但完全没有按照 Python 本身的知识体系讲解，单纯地为了简单而简单；有些书略微系统一些，却讲得晦涩难懂。典型来说，仅仅一个变量的概念，几乎没有一本书能通俗地讲明白。实际上，初学者并不需要知道变量的概念定义，他只要把变量当成一个小的“容器”，懂得对变量赋值就是把东西“装入”变量即可。那么我还是自己写一本吧，毕竟我曾经为别人写了那么多书。

创作这本书时，我有两点考虑。

1. 讲解要尽量通俗，避免搞那些晦涩的概念

编程，首要的是能动手编，让简单的程序跑起来。动手编得多了，那些概念的意义自然就浮现出来了。就像一个外星人来到地球，从未见过桌子，找个人一直给他讲桌子的概念，要他务必先理解桌子的概念，外星人的感觉一定是非常困惑；尝试用不同的方法：找一堆桌子放在一起，一张桌子、一张桌子给他看，让他在桌子上写字、用电脑、吃饭，甚至把桌子拆开给他看，相信外星人很快就能理解“什么是桌子”了。对于编程初学者而言，他们何尝不是刚来到地球的外星人？

2. 知识体系要完善，而且遵循 Python 内在的逻辑

一直以来，我写的书通常比较厚、内容也比较多。这和我挑剔的个性有关：既然做一件事情，当然要尽力做好它；否则干脆别做。一门工业级的编程语言，它不是玩具，它本身有那么多的知识点。**不管你学还是不学，编程语言本身的内容就在那里！不管作者写还是不写，编程语言本身的内容就在那里。**我写书总会尽量做到“够用”，起码认真学完这本书之后，不会随便遇到一个编程问题就只能问百度。

既要有完备的知识体系，又要详细讲透这些内容，书的篇幅自然就多了。同样的知识内容，一本厚厚的、讲解细致的图书，和一本薄薄的、浮光掠影的图书，哪本更容易看懂？

在知识内容相同的前提下，如果看不懂一本内容丰富、讲解细致的书，看一本薄薄的、浮

光掠影的图书反而能看懂？这完全没道理。

但有些读者确实这样说过，这一点我也能理解，存在“鸵鸟心态”的人，他并不是第一个：看不到的就当它不存在。有些书之所以薄，无非是两个知识点不讲：这也不讲，那也不讲！读者阅读的时候固然是轻松，因为内容少呀。就像学数学，如果只教一加一等于二，当然讲得简单、学得轻松；但等到真正做事时才发现：啊？还有二加三等于五？数学还有乘法？还有除法？然后发现这也不会，那也看不懂，后果就是遇到问题就上百度。这就是有些所谓的开发者，他们是“面向百度”编程的。这些开发者往往哀叹：做程序员太累了，一个问题往往要调半天甚至一天，其实他们根本不是调试，只是在找别人的代码、试别人的代码，运气好找到了合适的代码，问题就解决了；找不到合适的代码就只能哀叹了。

正因为基于以上两点考虑进行创作，因此初学者上手本书的门槛比较低，大部分读者都能迅速地通过学习本书内容写出自己的 Python 程序、运行自己的 Python 程序；但要坚持把本书学完也需要一定的毅力：书中内容确实比较多，而且后面内容更偏向实际应用开发。

编程图书不仅是用来“看”的，更是需要动手“练”的，正如先圣王阳明所倡导的：知行合一。学习本书需要读者认真练习书中每个示例程序，还需要读者认真完成全书在各章节后所配的 110 道 Python 练习题（面试题），如果读者需要获取关于课后习题的解决方法、编程思路，可以登录 http://www.crazyit.org 站点或关注“疯狂图书”微信公众号（拿出手机扫描封面勒口处的二维码）。

本书有什么特点

本书并不是一本简单的 Python 入门教材，虽然本书上手门槛很低，但本书的知识体系很丰富。总结起来，本书具有如下三个特点。

1. 讲解通俗，上手门槛低

创作本书的最初目的决定了本书的上手门槛，本书不会故弄玄虚地纠缠于晦涩的概念，而是力求用浅显易懂的比喻引出概念、用口语化的方式介绍编程、用清晰的逻辑解释思路。

为了降低读者阅读的难度，书中代码的注释非常详细，几乎每两行代码就有一行注释。本书所有程序中关键代码以粗体字标出，也是为了帮助读者能迅速找到这些程序的关键点。

2. 案例驱动，引爆编程激情

本书不是知识点的铺陈，而是致力于将知识点融入实际项目的开发中，所以书中涉及大量 Python 案例：五子棋游戏、画图板、桌面弹球、合金弹头、大数据展示、各国人均 GDP 对比、基于爬虫的招聘热点分析、基于爬虫的高清图片下载、基于 Scrapy+Selenium 的微博登录……希望读者通过编写这些程序找到编程的乐趣。

3. 知识体系完备，直面企业开发实战

虽然本书在讲解上力求简单，但本书内容并不简单，全书知识体系完备且系统，不仅全方位地覆盖 Python 语言本身的语法，而且覆盖大数据展示、爬虫等 Python 的热门技术，这些内容能带领读者直面企业开发实战。

试读结束：需要全本请在线购买： www.ertongbook.com

本书写给谁看

如果你仅仅想对 Python 有所涉猎，那么本书并不适合你；如果小朋友有兴趣学习本书，可先引导他阅读本书前半部分；如果你想全面掌握 Python 编程，并使用 Python 解决大数据分析、网络爬虫等实际企业开发项目，那么你应该选择本书，并认真学完此书。希望本书能引爆你内心潜在的编程激情，让你废寝忘食。

2018-09-20

目 录 CONTENTS

试读结束：需要全本请在线购买： www.ertongbook.com

CHAPTER 1

第 1 章 Python 语言概述和开发环境

本章要点

- Python 的发展历史
- Python 语言的特点
- Python 程序的运行机制
- 在 Windows 上安装 Python 开发环境
- 在 Linux 上安装 Python 开发环境
- 在 Mac OS X 上安装 Python 开发环境
- 编写第一个 Python 程序
- 运行第一个 Python 程序
- 掌握交互式解释器的用法

伴随着大数据和人工智能的兴起，Python 这门“古老”的语言重新焕发出耀眼的光彩。实际上 Python 一直是一门优秀的编程语言，不仅简洁、易用，而且功能强大，它能做到的事情太多了——既可用于开发桌面应用，也可用于做网络编程，还可用于开发 Web 应用……可能正因为它能做到的方面太多，反而显得没有特别突出的一面。另外，由于 Python 非常简单，很多非专业人士都可使用 Python（我儿子 8 岁开始学编程，也是从 Python 开始的），这可能导致一些专业程序员对 Python 抱有偏见。

现在情况发生了改变，Python 在大数据和人工智能两个领域大放异彩，使得 Python 语言变得非常流行（目前 Python 排在商用语言排行榜的第 4 位），本书将会向读者详细介绍 Python 这门优秀的编程语言。本章重点介绍如何搭建 Python 的开发环境。

1.1 Python 简介

虽然软件产业的历史相对于人类历史只是白驹过隙，但世界上却存在非常多的编程语言，Python 就是其中之一。Python 语言算得上一门“古老”的编程语言，Python 流行这么久，必然有它的独到之处，下面我们简单介绍 Python 的相关情况。

1.1.1 Python 简史

Python 由 Guido van Rossum 于 1989 年年底出于某种娱乐目的而开发，Python 语言是基于 ABC 教学语言的，而 ABC 这种语言非常强大，是专门为非专业程序员设计的。但 ABC 语言并没有获得广泛的应用，Guido 认为是非开放造成的。

Python 的“出身”部分影响了它的流行，Python 上手非常简单，它的语法非常像自然语言，对非软件专业人士而言，选择 Python 的成本最低，因此某些医学甚至艺术专业背景的人，往往会选择 Python 作为编程语言。

Guido 在 Python 中避免了 ABC 不够开放的劣势，Guido 加强了 Python 和其他语言如 C、C++ 和 Java 的结合性。此外，Python 还实现了许多 ABC 中未曾实现的东西，这些因素大大提高了 Python 的流行程度。

2008 年 12 月，Python 发布了 3.0 版本（也常常被称为 Python 3000 或简称 Py3k）。Python 3.0 是一次重大的升级，为了避免引入历史包袱，Python 3.0 没有考虑与 Python 2.x 的兼容。这样导致很长时间以来，Python 2.x 的用户不愿意升级到 Python 3.0，这种割裂一度影响了 Python 的应用。

毕竟大势不可抵挡，开发者逐渐发现 Python 3.x 更简洁、更方便。现在，绝大部分开发者已经从 Python 2.x 转移到 Python 3.x，但有些早期的 Python 程序可能依然使用了 Python 2.x 语法。

2009 年 6 月，Python 发布了 3.1 版本。
2011 年 2 月，Python 发布了 3.2 版本。
2012 年 9 月，Python 发布了 3.3 版本。
2014 年 3 月，Python 发布了 3.4 版本。
2015 年 9 月，Python 发布了 3.5 版本。
2016 年 12 月，Python 发布了 3.6 版本。
……

提示

本书将以 Python 3.x 来介绍 Python 编程，也会简单对比 Python 2.x 与 Python 3.x 的语法差异。

目前，由于大数据、人工智能（AI）的流行，Python 变得比以往更加流行。在最新的 TIOBE 编程语言排行榜上，Python 已经迅速上升到第 4 位，仅次于 Java、C、C++。

提示

Java 占据了世界上绝大部分电商、金融、通信等服务端应用开发，而 C、C++占据了世界上绝大部分贴近操作系统的硬件编程，这三门语言的地位太难动摇了。

1.1.2 Python 的特点

Python 是一种面向对象、解释型、弱类型的脚本语言，它也是一种功能强大而完善的通用型语言。

相比其他编程语言（比如 Java），Python 代码非常简单，上手非常容易。比如我们要完成某个功能，如果用 Java 需要 100 行代码，但用 Python 可能只需要 20 行代码，这是 Python 具有巨大吸引力的一大特点。

Python 的两大特色是清晰的语法和可扩展性。Python 的语法非常清晰，它甚至不是一种格式自由的语言。例如，它要求 if 语句的下一行必须向右缩进，否则不能通过编译。

Python 的可扩展性体现为它的模块，Python 具有脚本语言中最丰富和强大的类库（这些类库被形象地称为“batteries included，内置电池”），这些类库覆盖了文件 I/O、GUI、网络编程、数据库访问、文本操作等绝大部分应用场景。此外，Python 的社区也很发达，即使一些小众的应用场景，Python 往往也有对应的开源模块来提供解决方案。

Python 作为一门解释型的语言，它天生具有跨平台的特征，只要为平台提供了相应的 Python 解释器，Python 就可以在该平台上运行。

提示

解释型语言几乎天然是跨平台的。

Python 自然也具有解释型语言的一些弱点。

- 速度慢：Python 程序比 Java、C、C++等程序的运行效率都要慢。
- 源代码加密困难：不像编译型语言的源程序会被编译成目标程序，Python 直接运行源程序，因此对源代码加密比较困难。

上面两个问题其实不是什么大问题，关于第一个问题，由于目前计算机的硬件速度越来越快，软件工程往往更关注开发过程的效率和可靠性，而不是软件的运行效率；至于第二个问题，则更不是问题了，现在软件行业的大势本来就是开源，就像 Java 程序同样很容易反编译，但丝毫不会影响它的流行。

1.2 Python 程序运行机制

Python 是一门解释型的编程语言，因此它具有解释型语言的运行机制。

计算机程序，其实就是一组计算机指令集，能真正驱动机器运行的是机器指令，但让普通开发者直接编写机器指令是不现实的，因此就出现了计算机高级语言。高级语言允许使用自然语言（通常就是英语）来编程，但高级语言的程序最终必须被翻译成机器指令来执行。

高级语言按程序的执行方式可以分为编译型和解释型两种。

编译型语言是指使用专门的编译器，针对特定平台（操作系统）将某种高级语言源代码一次性“翻译”成可被该平台硬件执行的机器码（包括机器指令和操作数），并包装成该平台所能识别的可执行程序的格式，这个转换过程称为编译（Compile）。编译生成的可执行程序可以脱离开发环境，在特定的平台上独立运行。

有些程序编译结束后，还可能需要对其他编译好的目标代码进行链接，即组装两个以上的目标代码模块生成最终的可执行程序，通过这种方式实现低层次的代码复用。

因为编译型语言是一次性编译成机器码的，所以可以脱离开发环境独立运行，而且通常运行效率较高。但因为编译型语言的程序被编译成特定平台上的机器码，因此编译生成的可执行程序通常无法移植到其他平台上运行；如果需要移植，则必须将源代码复制到特定平台上，针对特定平台进行修改，至少需要采用特定平台上的编译器重新编译。

现有的 C、C++、Objective-C、Pascal 等高级语言都属于编译型语言。

解释型语言是指使用专门的解释器对源程序逐行解释成特定平台的机器码并立即执行的语言。解释型语言通常不会进行整体性的编译和链接处理，解释型语言相当于把编译型语言中的编译和解释过程混合到一起同时完成。

可以认为：每次执行解释型语言的程序都需要进行一次编译，因此解释型语言的程序运行效率通常较低，而且不能脱离解释器独立运行。但解释型语言有一个优势，就是跨平台比较容易，只需提供特定平台的解释器即可，每个特定平台上的解释器都负责将源程序解释成特定平台的机器指令。解释型语言可以方便地实现源程序级的移植，但这是以牺牲程序执行效率为代价的。

编译型语言和解释型语言的对比如图 1.1 所示。

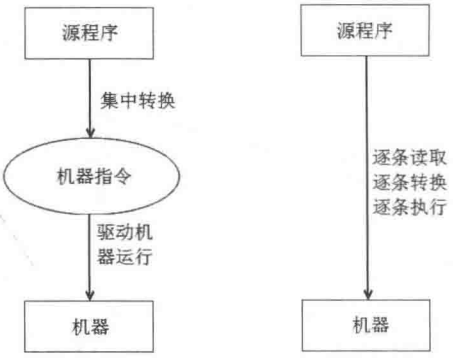


图 1.1 编译型语言和解释型语言

此外，还有一种伪编译型语言，如 Visual Basic，它属于半编译型语言，并不是真正的编译型语言。它首先被编译成 P-代码，并将解释引擎封装在可执行程序内，当运行程序时，P-代码会被解析成真正的二进制代码。从表面上看，Visual Basic 可以编译生成可执行的 EXE 文件，而且这个 EXE 文件也可以脱离开发环境，在特定平台上运行，非常像编译型语言。实际上，在这个 EXE 文件中，既有程序的启动代码，也有链接解释程序的代码，而这部分代码负责启动 Visual Basic 解释程序，再对 Visual Basic 代码进行解释并执行。

Python 语言属于解释型语言，因此运行 Python 程序时需要使用特定的解释器进行解释、执行。

解释型的 Python 语言天生具有跨平台的能力，只要为 Python 提供相应平台的解释器即可。接下来将会介绍为不同平台安装 Python 解释器。

1.3 开发 Python 的准备

在开发 Python 程序之前，必须先完成一些准备工作，也就是在计算机上安装并配置 Python 解释器。

1.3.1 在 Windows 上安装 Python

在 Windows 上安装 Python 请按如下步骤进行。

- 登录 https://www.python.org/downloads/页面，可以在该页面上看到两个下载链接。
  - Download Python 3.6.x：下载 Python 3.x 的最新版。
  - Download Python 2.7.x：下载 Python 2.x 的最新版。

从上面的链接可以看出，Python 同时维护 3.x 和 2.x 两个版本，这样既可让早期项目继续使用 Python 2.x，也可让新的项目使用 Python 3.x。

- 不要直接单击该页面上的两个下载链接，因为这两个链接总是下载 32 位的安装文件，但现在大部分 Windows 都是 64 位的，而 32 位 Python 无法在 64 位的 Windows 上运行。在该页面下方的“Looking for a specific release?”列表中选择“Python 3.6.x”，可以看到如图 1.2 所示的下载列表。

试读结束：需要全本请在线购买： www.ertongbook.com