Broadview®
www.broadview.com.cn

从Python开始学编程

Vamei 著　　雷雨田 插画

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

从Python开始学编程

Vamei 著　　雷雨田 插画

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

内 容 简 介

本书以 Python 为样本，不仅介绍了编程的基本概念，还着重讲解了编程语言的范式（面向过程、面向对象、面向函数），并把编程语言的范式糅在 Python 中，让读者不仅学会 Python，未来在学习其他编程语言时也变得更加容易。

图书在版编目（CIP）数据

从 Python 开始学编程 / Vamei 著. —北京：电子工业出版社，2017.1
ISBN 978-7-121-30199-5

Ⅰ. ①从… Ⅱ. ①V… 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（2016）第 258912 号

责任编辑：安　娜
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：787×980　1/16　　印张：13　　字数：162 千字
版　　次：2017 年 1 月第 1 版
印　　次：2017 年 1 月第 2 次印刷
定　　价：49.00 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

从读博士起，我对编程的兴趣忽然浓厚起来。当时做大规模并行运算，需要自己写很多程序和脚本。作为新进研究组的新人，我自觉负担起很多写程序的活儿。写得多了，兴趣也变得浓厚。

那个时候抓紧一切机会学习编程。在我读博的研究所里，有一位英国教授也喜欢编程。她叫爱玛·希尔（Emma Hill），教我们用编程语言处理地球科学的数据。有一天，我路过她的办公室。她问我最近的学习进度。

“准备学 Perl 呢，”我回答说，“感觉 Perl 在地理领域应用很广。”

“你为什么不学学 Python 呢？”爱玛问我，“这门语言发展很快。你学会了或许可以教教我。”

我之前听过 Python 的一些传闻，比如那句著名的“人生苦短，我用 Python”。但我担心 Python 在地球科学研究方面不如 Perl 积累深厚。有了爱玛的鼓励，我下定决心去研究 Python。Python 学起来确实很快。没过多久，我就可以用 Python 来解决我在科研中遇到的大部分问题了。记忆比较深刻的是，有一次下载来自美国研究所的一批气象数据。我用 Python

中的多线程并发下载，创造了大学中网络传输的纪录。学习加实践，让我爱上了这门语言。

随后，我开始写一系列博客，记录自己学习 Python 的过程。这一系列的文章叫“Python 快速教程”。我想在这些文章中呈现出 Python 简单易学的特点，以便让更多的人也来享受编程的乐趣。在写作过程中我意识到，要想讲明白一门编程语言，还要引入额外的背景知识。我的编程博客也从 Python 开始，拓展到网络协议、操作系统、算法、数据分析等方面。写的时间越长，收获的读者也越来越多。每当有人告诉我看着我的文章学会编程时，我总会感到惊喜。因此，我非常感谢爱玛给我推开的这扇门。

完成博士学业之后，我需要在科研和编程之间选择。由于编程带给我的美好体验，我毫不犹豫地选择了编程。将近三十岁的我，和二十出头的年轻人一起做产品、调试、debug。我必须要非常努力，才能赶上这群富有天赋而精力旺盛的年轻人。但我并不觉得辛苦。辛苦是学习的台阶。在编程中，我享受着脑细胞的疯狂激活，享受着未知错误的折磨，以及苦苦思索之后的豁然开朗。更棒的是，我的伙伴总是以乐观的态度来看待技术，以享受的心态来享受编程。我从中受益良多。更何况，计算机浪潮已经并将继续改变世界。我很幸运，能加入浪潮中。

“Python 快速教程”得到了不少编辑的认可。他们希望我能把博客文章改编成一本书。写书当然是莫大的荣幸，我很感谢每一位编辑的赏识。可在博士学业的压力下，我能抽出的时间实在有限。终于拖到博士毕业，我才开始认真整理之前的文章。把略显凌乱的博客文章改编成书，工作量比我想象的要大得多。在此期间，我也开始了一个新的项目，研发一款用于畜牧的智能芯片。生活的节奏又变得忙碌，能分给写书的时间大大减少。结果，从签合约到完稿，我花了超过半年的时间。幸好编辑安娜对我的拖延症格外包容。

这本书的最终诞生，有赖于许多人的支持。感谢父母对我的激励和教育，感谢妻子一直以来的陪伴。雷雨田绘制的精美插画，让枯燥的技术书变得生动有趣。在写作博客的过程中，许多读者都指正过文章中的错误，或者对写作方向提出建议。在成书过程中，王豪、周昕梓和黄杜立对文章进行审阅校正。正是因为他们的审阅校正，我才能放心地交稿。此外还有很多帮助过我的人，不能一一列举，只好一并表达感激。

在我现在的工作中，Python 依然占据着重要的地位。我会用 Python 进行网站开发和大数据分析，还会用 Python 来写一些在单片机上运行的脚本。当然，我也离不开其他语言，比如处理数据库的 SQL、编写安卓 App 的 Java、开发网页前端的 JavaScript 等。但 Python 让我爱上编程。我也希望，这本书能让读者也爱上 Python，并且继续像我的博客文章一样，能帮助到那些想学习编程的人。在此存一个美好心愿。

Vamei

目　录

第 1 章

用编程改造世界

本章将简要介绍计算机和编程的历史。从计算机出现以来，硬件性能得到飞跃式的发展。与此同时，编程语言也几经变迁，产生了多种编程范式。Python 语言以简洁灵活的特点，在诸多编程语言中打下一片天地。通过历史，我们不但能体验 Python 的特色，还能了解这门语言想要解决的痛点。本章将以一个简单的“Hello World!”程序结束，从此开启 Python 之旅。

1.1 从计算机到编程

人能运算和记忆，但更了不起的是善于借用工具。人类很早就开始利用方法和工具，辅助计算和记忆这样高度复杂的认知活动。古人用给绳子打结的方式来记录圈养的牛羊，我们的祖先很早就能以眼花缭乱的速度使用算盘。随着近代工业化的发展，社会对计算的需求越来越强烈。收税需要计算，造机器需要计算，开挖运河也需要计算。新的计算工具不断出现。利用对数原理，人们制造出计算尺。计算尺可以平行移动尺子来计算乘除法。19 世纪的英国人巴贝奇设计了一台机器，用齿轮的组合来进行高精度的计算，隐隐预示着机器计算的到来。20 世纪初有了机电式的计算机器。电动马达驱动变档齿轮“咯吱”转动，直到得到计算结果。

二战期间，战争刺激了社会的计算需求。武器设计需要计算，比如坦克的设计、潜艇的外形、弹道的轨迹。社会的军事化管理需要计算，比如火车调度、资源调配、人口动员。至于导弹和核弹这样的高科技项目，更需要海量的计算。美国研制原子弹的曼哈顿计划，除了使用 IBM 的计算机器外，还雇佣了许多女孩子进行手工运算。计算本身甚至可以成为武器。电影《模仿游戏》就讲述了英国数学家阿兰·图灵（Alan Mathison Turing）破解德国传奇密码机的故事。图灵的主要工具，就是机电式的计算机器。值得一提的是，正是这位图灵，提出了通用计算机的

理论概念，为未来的计算机发展做好了理论准备。现在，计算机学科的最高奖项就以图灵命名，以纪念他的伟大贡献。德国工程师康拉德·楚泽（Konrad Zuse）发明的 Z3 计算机能编写程序。这看起来已经是一台现代计算机的雏形了。

计算工具的发展是渐进的。不过历史把第一台计算机的桂冠颁给了战后宾夕法尼亚大学研制的埃尼阿克（ENIAC，Electronic Numerical Integrator And Computer）。埃尼阿克借鉴了前任的经验，远非横空出世的奇迹。但它最重要的意义，是向人们展示了高速运算的可能性。首先它是一台符合图灵标准的通用计算机，能通过编程来执行多样的计算任务。其次，与机电式计算机器不同，埃尼阿克大量使用真空管，从而能快速运算。埃尼阿克的计算速度比之前的机电型机器提高了一千倍，这是一次革命性的飞跃。因此，即便计算辅助工具同样历史悠久，但人们仍认为埃尼阿克引领了一个新的时代——计算机时代。

从埃尼阿克开始，计算机经历了迅猛的发展。计算机所采用的主要元件，从真空管变成大规模集成电路。计算机的运算性能，每一两年的时间就会翻倍。但计算机的大体结构，都采用了冯·诺依曼体系。这一体系是长期演化的结果，但冯·诺依曼进行了很好的总结。按照冯·诺依曼的设计，计算机采用二进制运算，包括控制器、运算器、存储器、输入设备和输出设备五个部分，如图 1-1 所示。五个部分相互配合，执行特定的操作，即指令。这五个部分各有分工。

1. **控制器**：计算机的指挥部，管理计算机其他部分的工作，决定执行指令的顺序，控制不同部件之间的数据交流。

2. **运算器**：顾名思义，这是计算机中进行运算的部件。除加减乘除之类的算数运算外，还能进行与、或、非之类的逻辑运算。运算器与控制器一起构成了中央处理器（CPU，Central Processing Unit）。

3. **存储器**：存储信息的部件。冯·诺依曼根据自己在曼哈顿工程中的经验，提出了存储器不但要记录数据，还要记录所要执行的程序。

4. **输入设备**：向计算机输入信息的设备，如键盘、鼠标、摄像头等。

5. **输出设备**：计算机向外输出信息的设备，如显示屏、打印机、音响等。

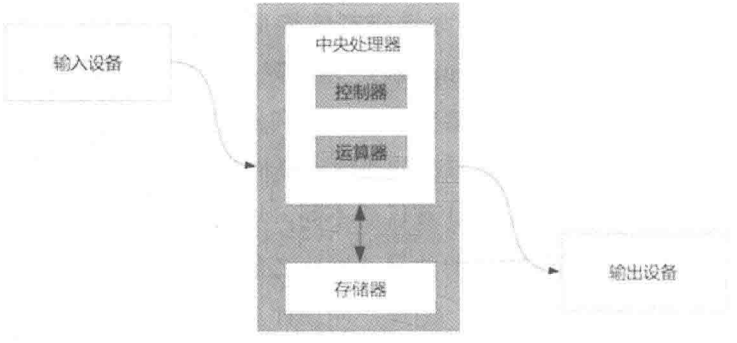

图 1-1 冯·诺依曼结构

人们最常想到的计算机是台式机和笔记本电脑。其实，计算机还存在于智能手机、汽车、家电等多种设备中。但无论外形如何多变，这些计算机都沿袭了冯·诺依曼结构。不过在具体细节上，计算机之间又有很大的差别。有的计算机使用了多级缓存，有的计算机只有键盘没有鼠标，有的计算机用磁带存储。计算机的硬件是一门很庞杂的学问。幸运的是，计算机用户大多不需要和硬件直接打交道。这一点是**操作系统**（Operating System）的功劳。

操作系统是运行在计算机上的一套软件，负责管理计算机的软硬件资源。有的时候我们请人修电脑，他可能会说“电脑需要重装一下”。这个“重装”，就是重新安装操作系统的意思。无论是微软的 Windows，还是苹果的 iOS，都属于操作系统。我们编程时，大多数时候都是通过操作系统这个“中间商”来和硬件打交道的。操作系统提供了一套**系统调用**（System Call），如图 1-2 所示，规定了操作系统支持哪些操作。当调用

某个系统调用时，计算机会执行对应的操作。这就像是按下钢琴上的一个键，钢琴就会发出对应的音律一样。系统调用提供的功能非常基础，有时调用起来很麻烦。操作系统因此定义一些**库函数**（Library Routine），将系统调用组合成特定的功能，如同几个音律组成的和弦。所谓的编程，就是用这些音律和和弦，来组成一首美妙的音乐。

图 1-2　硬件、操作系统和应用程序的关系

1.2　所谓的编程，是做什么的

编程中总是在调用计算机的基本指令。如果完全用基础指令来说明所有的操作，代码将超乎想象的冗长。IBM 前总裁小沃森的自传中就提到，他看到一位工程师想要做乘法运算，输入程序用的打孔卡叠起来有 1.2 米高。幸好，程序员渐渐发现，许多特定的指令组合会重复出现。如果能在程序中复用这些代码，则可以节省很多工作量。复用代码的关键是**封装**（Packaging），即把执行特殊功能的指令打包成一个程序块，然后给这个程序块起个容易查询的名字。如果需要重复使用这个程序块，则

可以简单地通过名字调用。就好像食客在点菜时，只需告诉厨师做“鱼香肉丝”，而不需要具体说明要多少肉、多少调料、烹制多久一样。刚才提到的操作系统，就是将一些底层的硬件操作组合封装起来，供上层的应用程序调用。当然，封装是有代价的，它会消耗计算机资源。如果使用的是早期的计算机的话，封装和调用的过程会非常耗时，最终得不偿失。

封装代码的方式也有很多种。根据不同的方式，程序员写程序时要遵循特定的编程风格，如面向过程编程、面向对象编程和函数式编程。用更严格的术语来说，每种编程风格都是一种**编程范式**（Programming Paradigm）。编程语言开始根据编程范式区分出阵营，面向过程的 C 语言、面向对象的 Java 语言、面向函数的 Lisp 语言等。任何一种编程范式编写出来的程序，最终都会翻译成上面所述的简单功能组合。所以编程的需求总是可以通过多种编程范式来分别实现，区别只在于这个范式的方便程度而已。由于不同的范式各有利弊，所以现代不少编程语言都支持多种编程范式，以便程序员在使用时取舍。Python 就是一门多范式语言。某一范式的代表性语言，也开始在新版本中支持其他范式。原本属于面向对象范式的 Java 语言，就在新版本中也开始加入函数式编程的特征。

编程范式是学习编程的一大拦路虎。对于一个程序员来说，如果他熟悉了某一种编程范式，那么他就能很容易地上手同一范式的其他编程语言。对于一个新手来说，一头扎进 Python 这样的多范式语言，会发现同一功能有不同的实现风格，不免会感到困惑。一些大学的计算机专业课程，选择了分别讲授代表性的范式语言，比如 C、Java、Lisp，以便学生在未来学习其他语言时有一个好的基础。但这样的做法，会将学习过程拉得很漫长。在我看来，Python 这样的多范式语言提供了一个对比学习多种编程范式的机会。在同一语言框架下，如果程序员能清晰地区分出不同的编程范式，并了解各自的利弊，将起到事半功倍的效果。这也是本书中想要做到的，从面向过程、面向对象、函数式三种主流范式出发，在一本书的篇幅内学三遍 Python。这样的话，读者将不止是学会了

一门 Python 语言，还能为未来学习其他语言打好基础。

学习了包括 Python 在内的任何一门编程语言后，就打开了计算机世界的大门。通过编程，你几乎可以发挥出计算机所有的功能，给创造力提供了广阔的施展空间。想到某个需求，比如统计金庸小说中的词频，自己就能写程序解决。有了不错的想法，例如想建立一个互助学习的网站，也可以立即打开电脑动手编写。一旦学会了编程，你会发现，软件主要比拼的就是大脑和时间，其他方面的成本都极为低廉。编写出的程序还会有许多回报。可以是经济性的回报，比如获得高工资，比如创立一家上市的互联网企业。也可以是声誉性的回报，比如做出了很多人喜爱的编程软件，比如攻克了困扰编程社区的难题等。正如《黑客与画家》一书中所说，程序员是和画家一样的创作者。无穷的创造机会，正是编程的一大魅力所在。

编程是人与机器互动的基本方式。人们通过编程来操纵机器。从 18 世纪的工业革命开始，人们逐渐摆脱了手工业的生产方式，开始转向机器生产。机器最开始用于棉纺工业，一开始纺出的纱线质量比不上手工纺制的。但机器可以昼夜工作，不知疲倦，产量也是惊人。因此，到了 18 世纪末，全球大部分的棉布都变成了机器生产。如今，机器在生活中已经屡见不鲜。人工智能这样的“软性机器”，也越来越多地进入生产和生活。工人用机器来制造手机、医生操纵机器来进行微创手术、交易员用机器进行高频的股票交易。残酷一点讲，对机器的调配和占有能力，将会取代血统和教育，成为未来阶级区分的衡量标准。这也是编程教育变得越来越重要的原因。

机器世界的变化，正在改变世界的工作格局。重复性工作消亡，程序员的需求量却在不断加大。很多人都在自学编程，以便跟上潮流。幸好，编程也变得越来越简单。从汇编语言，到 C 语言，再到 Python 语言，编程语言越来越亲民。以 Python 为例，在丰富的模块支持下，一个功能的实现只需要寥寥几个接口的调用，不需要费太多工夫。我们之前所说

的封装，也是把功能给打包成规范的接口，让别人用起来觉得简单。编程用精准的机器为大众提供了一个规范化的使用接口，无论这个接口是快速安全的支付平台，还是一个简单快捷的订票网站。这种封装和接口的思维反映在社会生活的很多方面。因此，学习编程也是理解当代生活的一个必要步骤。

1.3 为什么学 Python

正如 1.2 节所说，高级语言的关键是封装，让程序编写变得简单。Python 正是因为在这一点上做得优秀，才成为主流编程语言之一。Python 的使用相当广泛，是 Google 的第三大开发语言，也是 Dropbox、Quora、Pinterest、豆瓣等网站主要使用的语言。在很多科研领域，如数学、人工智能、生物信息、天体物理等，Python 都应用广泛，渐有一统天下的势头。 当然，Python 的成功并非一蹴而就。它从诞生开始，已经经历了二三十年的发展。回顾 Python 的历史，我们不但可以了解 Python 的发展历程，还能理解 Python 的哲学和理念。

Python 的作者是**吉多·范·罗苏姆**（Guido von Rossum）。罗苏姆是荷兰人。1982 年，他从阿姆斯特丹大学（University of Amsterdam）获得了数学和计算机硕士学位。然而，尽管他算得上是一位数学家，但他更加享受计算机带来的乐趣。用他的话说，尽管拥有数学和计算机双料资质，但他总是趋向于做计算机相关的工作，并热衷于做任何和编程相关的活儿。

在编写 Python 之前，罗苏姆接触并使用过诸如 Pascal、C、Fortran 等语言。这些语言的关注点是让程序更快运行。在 20 世纪 80 年代，虽然 IBM 和苹果已经掀起了个人电脑浪潮，但这些个人电脑的配置在今天看来十分低下。早期的苹果电脑只有 8MHz 的 CPU 主频和 128KB 的内存，稍微复杂一点的运算就能让电脑死机。因此，当时编程的核心是优化，