TURING 图灵程序设计丛书

Pearson

- 引领读者巧妙避开学习陷阱，掌握Python思考方式
- 示例演示与练习相结合
- 涵盖Python中高级功能

Python Without Fear

A Beginner's Guide That Makes You Feel Smart

零压力学Python

[美] 布莱恩·奥弗兰德 著

袁国忠 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Python Without Fear

A Beginner's Guide That Makes You Feel Smart

零压力学Python

[美] 布莱恩·奥弗兰德 著

袁国忠 译

人民邮电出版社

北 京

图书在版编目（CIP）数据

零压力学Python / (美) 布莱恩·奥弗兰德
(Brian Overland) 著 ; 袁国忠译. -- 北京 : 人民邮电
出版社, 2018.6
（图灵程序设计丛书）
ISBN 978-7-115-48314-0

Ⅰ. ①零… Ⅱ. ①布… ②袁… 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067596号

内 容 提 要

本书以 Python 为背景讲述计算机编程，用示例强化理解、条分缕析 Python 工作原理，给出了经验总结及其语言特性的缘由和技巧。内容包括 Python 编程基础知识、如何独立编程、程序片段的功能及来由、创建实用和可重用代码、面向对象编程等。

本书简单易懂，适合 Python 初学者阅读。

◆ 著 [美] 布莱恩·奥弗兰德
译 袁国忠
责任编辑 朱 巍
执行编辑 张海艳
责任印制 周昇亮

◆ 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：800×1000 1/16
印张：20
字数：473千字 2018年6月第1版
印数：1－3 500册 2018年6月北京第1次印刷
著作权合同登记号 图字：01-2018-0948号

定价：79.00元

读者服务热线：(010)51095186转600 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

版权声明

谨以此书献给我亲爱的朋友——Skyler、Orlando、Madison、Cleo 和 Pogo。

前　言

免费的编程指南汗牛充栋，其中很多都是介绍 Python 的。有鉴于此，一本书必须非常好，才值得你花时间去阅读。

我编写本书是因为它正是我多年前刚接触 Python 时想要拥有的。与其他人一样，我当时每搞懂一个概念，就需要阅读十几本不同的图书，还需要研究数十份网上材料。

可本书要介绍的是 Python，大家都认为它学起来不难！

问题是并非什么东西学起来都像期望的那样容易或快速，而且并非所有的图书和学习网站都那么有趣。例如，你可能需要在不同的网站之间转来转去，只为找到确实合乎情理的诠释。

本书采取了如下做法，要是我刚接触 Python 时能读到它就好了。

避开陷阱

在 Python 中，很多事情做起来相对容易，但也有一些原本应该容易的事情比其他语言中还难，如果你有编程经验，这一点尤其明显。Python 的行事风格常常与其他语言截然不同，你可能瞪着屏幕几小时，等有人指出简单的解决方案才恍然大悟。当然，你也可以购买本书。

如何以 Python 的方式思考

一个与陷阱紧密相关的问题是，明白如何以 Python 的方式思考。如果你不明白 Python 独特的模拟世界的方式，就可能会像 C 程序员那样编写程序。这样的程序管用，但未利用任何让 Python 成为快速开发工具的功能。

```
a_list = ['Don\'t', 'do', 'this', 'the' ,'C', 'way']
for x in a_list:
    print(x, end=' ')
```

这一小段代码打印如下输出：

```
Don't do this the C way
```

中高级功能

再重申一次，总体而言，Python 比其他语言容易，但并非在任何方面都如此。有些重要的 Python 中级功能难以理解，除非有人向你做出详尽的诠释。本书花了很大的篇幅来介绍中高级功能，包括列表推导、生成器、多维列表（矩阵）和装饰器。

以众多不同的方式学习

相比于其他图书，本书采用的教学方式更加多样化。当然，我使用了大量的示例，但有时候贴切的示意图或类比可让学习效果大不相同，而有时候通过练习来促使你举一反三也能获得同样的效果。本书所有的教学方式都旨在加深你对概念的理解。

走进幕后

虽然本书是为编程新手编写的，但也适合想了解 Python 的工作原理及其截然不同的幕后的人阅读。Python 在内部如何执行操作呢？如果你想对此进行深入的了解，本书就是为你编写的。

为何选择 Python

当然，如果你正为该选择哪种编程语言而犹豫不决，肯定想知道为何应首先学习 Python。

Python 正快速占领大片的编程江山。有些程序依然需要 C/C++的低级功能，但你会发现 Python 是一款快速开发应用程序的工具，能够让程序员事半功倍：常常只需编写几行代码就能完成令人诧异的任务。

具体地说，100 行的 Python 程序在功能上可能与 1000 乃至 2000 行的 C 程序相当。你可以使用 Python 来做可行性验证：花一个下午编写出程序，看看它能否满足项目的需求，确信程序管用后，可在必要时使用 C/C++重新编写，以便更高效地利用计算机资源。

了解这些后，但愿你会与我一道来完成这次有趣、激动人心而又愉快的旅行。可别忘了，Python 很酷哟！

```
x = ['Python', 'is', 'cool']
print(' '.join(x))
```

为方便地获取本书英文版的更新和修订，请前往 InformIT 网站注册。为此，请访问 informit.com/register 并登录（如果没有账户，请先创建账户再登录），再输入原书的 ISBN（9780134687476）并单击 Submit 按钮。在选项卡 Registered Products 中，找到并单击图书旁边的 Access Bonus Content 链接，以访问额外赠送的材料。读者也可登录图灵社区本书主页（http://www.ituring.com.cn/book/2556）提交反馈意见和勘误。

致　谢

对作者来说，写篇致谢是惯例，但就本书而言，这确有必要。本书每一章都打上了John Bennett深深的烙印。

John Bennett是本书的协作者之一，也是微软退休的程序员和软件开发工程师，有多年的Python使用经验——经常使用它来实现高级脚本语言。John Bennett指出本书应展示Python的行事风格，我接受了他的宝贵建议，不在书中演示如何将C++解决方案转换为Python版本，而演示如何充分利用Python概念，即如何以Python的方式思考。

还需指出的是，本书得以付梓离不开两位组稿编辑Kim Boedigheimer和Greg Doench的精神支持。Kim Boedigheimer在早期帮助启动了这个项目，而Greg Doench帮助这个项目顺利地推进。

感谢开发编辑Michael Thurston和技术编辑John Wargo，他们提出了重要的建议，极大地提高了本书的品质。还要感谢编辑小组的其他成员，让最后的出版阶段得以顺利、平稳地完成，他们是Julie Nahil、Kim Wimpsett、Angela Urquhart和Andrea Archer。

目　录

第 1 章 初识 Python

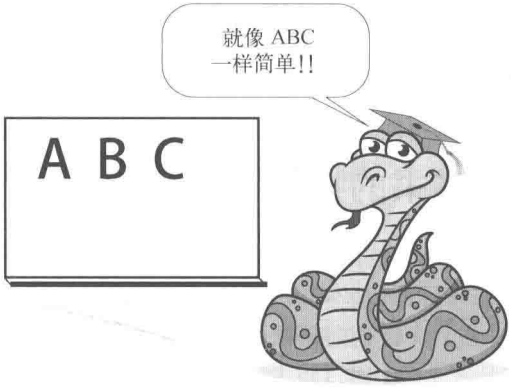
如果我告诉你，有一种计算机语言比其他计算机语言更好学、更易上手，且只需使用少量代码就能完成大量的工作，你会怎么想？

在很多人看来，这种语言就是 Python。它是从 ABC 语言（意思是就像 ABC 一样简单）衍生而来的，最近 20 年来在世界各地斩获了大量的拥趸。很多程序员都加入了 Python 社区，因为在这个社区，只需安装 Python，就可使用 100 000 多个免费包。

快来加入 Python 潮流吧。在本书中，我会引导你快速学会 Python，即便你编程经验有限；我还将引领你避开陷阱——Python 的行事方式如此不同，即便是经验丰富的程序员也会马失前蹄。本书适合编程新手阅读，也可供编程老手参考，因为其中讨论了幕后的情况。

1.1 Python 简史

Python 是荷兰程序员 Guido van Rossum 于 1991 年推出的，从很大程度上说，它是从 ABC 语言（请注意，不是 C 语言）衍生而来的。



Python 至今还保留着 ABC 的很多功能。Van Rossum 在 Python 界被称为“仁慈的独裁者”，他还在这门语言中融入了 Modula-3 语言的元素。

Van Rossum 给这门语言命名的灵感来自 BBC 的喜剧《巨蟒剧团之飞翔的马戏团》，因此这门语言与蟒蛇没有直接的联系，虽然巨蟒剧团的成员 John Cleese 最初这样给剧团命名隐含着“阴险、虚伪”的意思。Python 就这样诞生了，它与爬行动物还是有一定的联系的。

从此以后，Python 出现了多个版本，每个版本都添加了重要的功能，而最新的版本为 Python 3.0。本书是针对 Python 3.0 编写的，但也指出了如何修改示例，以支持 Python 2.0。

1.2　Python 有何不同

对于 Python，首先要知道的是它是免费的。

很多 Python 扩展都是免费的，可随 Python 一起安装。这些模块提供了数学、日期/时间、分数、随机数生成以及跨平台用户界面创建功能。与 Python 一样，它们都是免费的。

Python 内置的数字功能令人叹为观止，它支持复数、浮点数、分数（模块 Fractions）和“无穷大整数”。

Python 吸引了大批的拥趸。有很多开发人员向 Python 程序员同仁提供库，这些库被称为**包**，大多是免费的。要获取这些包，可在浏览器中搜索 Python Package Index，并访问找到的网站。本书编写期间，这个网站提供的包已超过 107 000 个。

乍一看，Python 程序与使用其他语言编写的代码没什么不同，但仔细观察后，你会发现它们大不相同。

- 不同于其他大多数语言，Python 不提供代码块起始和结束语法，代码之间的各种关系都是使用缩进表示的！在 C 语言程序员看来，这好像很危险，但能确保代码外观的一致性，让初学者更容易理解。
- Python 没有变量声明的概念：变量是通过给它们赋值来创建的。这极大地简化了 Python 的语法，但也制造出了隐藏极深的陷阱。本书将引领你避开这些陷阱。
- Python 大量地使用了**迭代**的概念。所谓迭代，就是遍历序列。这个概念深植于列表、字典和集合等高级结构中。如果能够充分利用这些高级结构，你就能螺蛳壳里做道场。

考虑到 Python 的功能，它常被视为一种“原型设计”或“应用程序快速开发”语言。你可先使用 Python 快速编写出程序，在需要提升运行效率时，再使用 C 和 C++进行改写。

1.3　本书的特色

我深信通过示例进行学习绝不逊于通过理论进行学习。本书将通过如下方式介绍 Python 基本知识以及一些中高级功能。

- 通过语法图和短小精悍的示例介绍 Python 功能。
- 通过较大的示例演示功能的用法。
- 每个较大的示例都包含“工作原理”部分，对示例代码逐行进行解码。
- 通过一系列练习检查你举一反三的能力。

Python 提供了一个交互式开发环境——IDLE，建议你使用它来完成每个短小精悍的示例。

本书在书页边的空白内使用了大量的图标，旨在提供阅读线索。

这些部分描述了一些基本的 Python 语法规则。需要你通过键盘原样输入的内容（如关键字和标点）用粗体表示；需要你提供具体内容的占位符，用斜体表示。例如，介绍 `global` 语句的语法时，关键字 **`global`** 用粗体表示，而需要你提供的变量名用斜体表示：

```
global variable_name
```

这个图标表示伪代码块，它们使用自然语言而不是 Python 系统地描述了程序的每一步。然而，鉴于 Python 语句与自然语言并不总是相差甚远，所以并非在任何情况下都需要使用伪代码。话虽如此，在有些情况下，伪代码对概述程序设计依然大有裨益。

这些部分对较大示例的每一行（至少是所有不那么显而易见的代码行）进行解码。

这些部分包含与之前的示例相关的练习。你至少应该尝试完成其中的一些，这样学习 Python 的速度将快得多。

这些部分演示如何对示例进行修改或重大改进。并非每个示例都包含这部分。有些示例之所以包含这部分，是因为它们完成任务的方式平淡无奇，而优化后的方法演示了经验丰富的 Python 程序员会如何做。

1.4 安装 Python

无论你使用的是 Windows、Macintosh 还是 Python 支持的其他众多操作系统，安装 Python 的步骤都差不多。基本步骤如下。

(1) 前往 Python 官网。

(2) 打开 Downloads 菜单。

(3) 如果出现了 Downloads for Windows 页面，单击按钮 Python 3.6.1。如果你使用的不是 Windows 操作系统，就需要在菜单 Downloads 中选择相应的操作系统。

(4) 单击 Save File 按钮。

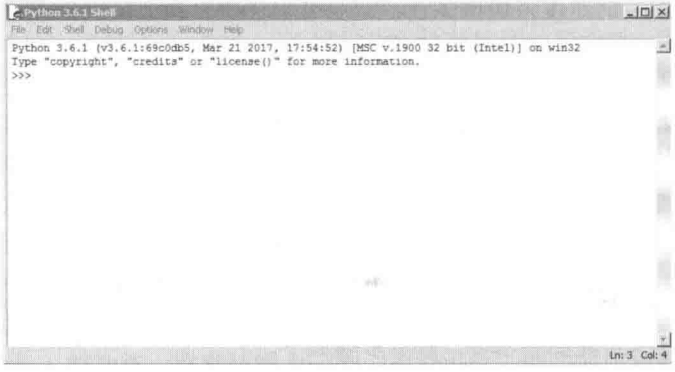
(5) 找到刚才保存的文件：无论你使用的是哪种操作系统，通常都有一个放置下载内容的目录。这个文件包含 Python 安装程序，双击它并按指示操作。

如果一切顺利，将在你的计算机上安装 Python 以及包括 tkinter（用于开发 GUI）在内的所有基本模块。现在你面临选择：为开始使用 Python，你可使用“基本交互模式”（这也可行，但没什么特别的），也可使用交互式开发环境——IDLE。

强烈建议你使用后者。基本交互模式能做的 IDLE 都能做，而且还能做很多其他的事情。下一节将介绍一些使用 IDLE 的方式，它们在后面将为你节省大量时间。

基本交互模式像下面这样，它只提供了基本编辑功能，根本不支持从文本文件中加载程序。

IDLE 像下面这样。注意，它提供了各种菜单；相比于使用基本交互模式，在 IDLE 中可做的事情要多得多，其中包括从文本文件中加载程序以及对其进行调试。



如果你使用的是 Windows，在“开始”菜单中就能够找到基本交互模式应用程序，但这不是你想要的。你应该花点时间选择“程序”>Python>IDLE，这样做是完全值得的。

在 Mac 系统中（假设你安装了 Python，包括 IDLE），要启动 IDLE，可能需要这样做：打开 Finder，然后依次选择“应用程序”、Python 和 IDLE。你下载的 Python 可能包含基本模式，也可能不包含。

1.5　开始通过 IDLE 使用 Python

启动交互式开发环境 IDLE。建议将它作为你的大本营，因为在学习 Python 期间，你的大部分时间都将在这里度过。你应将这个图标放到桌面，这样可随时轻松地启动 IDLE。

IDLE 启动后，你就会看到一个类似于下面这样的提示符：

```
>>>
```

在这个提示符下，你可以输入 Python 语句或表达式，还可使用命令 `help` 并指定类

型来获取帮助，如下所示：

```
>>>help(str)
```

对于用户输入（需要你通过键盘输入的字符），我将用粗体表示；对于 Python 的输出，则使用常规字体表示。全书都将遵循这样的约定。

1.6 在 IDLE 中修正错误

IDLE 的优点之一是，让你能够轻松地修正错误。假设你输入了如下代码：

```
>>>x = z
```

本书后面将指出，这是一条赋值语句，如果之前没有给变量 z 赋值，它将引发错误。IDLE 打印的消息类似于下面这样：

```
Traceback (most recent call last):
  File "<pyshell#205>", line 1, in <module>
    x = z
NameError: name 'z' is not defined
```

就这里而言，重新输入语句很容易，但如果导致错误的是一个长得多的代码块，你可能就不想重新输入整个代码块了。下面是一个这样的示例：

```
def print_nums(n):
    i = 1
    while i <= n:
        print(i, end='\t')
        i +++= 1
```

这个代码块存在的问题是，其中的最后一行为 `i +++= i` 而不是 `i += i`（只能有一个加号）。

你想修正这个错误，但不想重新输入所有的语句。所幸在 IDLE 中修正错误易如反掌，你只需这样做：

(1) 将光标放在代码块的任何一行中（如果代码块只包含一行代码，务必将光标放在行尾）；

(2) 按回车键。

整个代码块都奇迹般地再次出现了，且光标位于代码块的末尾。你可随便进行修复：使用箭头键移到要修复的地方，再进行修复。最后，要重新提交代码块，将光标放到最后一行的末尾并按回车键两次。

务必记住这种技巧，这将为你节省大量的时间。

1.7 续行

考虑到 Python 解读分行的方式，你不能像 C 语言中那样让代码横跨多行。如果你需要输入特别长的代码行，该怎么办呢？