

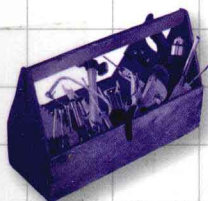
深入浅出程序设计 (中文版)

Head First Programming

A learner's guide to
programming, using
the Python language

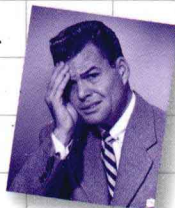


Load important
coding concepts
straight into
your brain



Add methods,
functions, and objects
to your toolbox

Avoid
embarrassing
mishaps with
input/output



Process
your data
like a pro

Build a
functional
and attractive
graphical
application



Learn how to automate
repetitive tasks

O'REILLY® 东南大學 出版社

Paul Barry & David Griffiths 著
蒋雁翔 童健 译

深入浅出程序设计

希望有一本介绍编程的书，可以让
人不会产生从计算机前逃开的冲动，
这难道不是一个梦吗？我猜这只能
是一个幻想……



Paul Barry & David Griffiths 著

蒋雁翔 童健 译

O'REILLY®

Beijing • Cambridge • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社
·南京·

图书在版编目 (CIP) 数据

深入浅出程序设计：中文版 / (英) 巴里 (Barry, P.) ,
(美) 格里匪思 (Griffiths, D.) 著; 蒋雁翔, 童健译. —南
京: 东南大学出版社, 2012.1

书名原文: Head First Programming
ISBN 978-7-5641-2966-8

I. ①深… II. ①巴… ②格… ③蒋… ④童… III. ①软
件设计—程序设计 IV. ① TP311.56

中国版本图书馆 CIP 数据核字 (2011) 第 174008 号

江苏省版权局著作权合同登记
图字: 10-2010-275 号

©2009 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press,
2012. Authorized translation of the English edition, 2009 O'Reilly Media, Inc., the owner of all rights to
publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2009。

简体中文版由东南大学出版社出版 2012。英文原版的翻译得到 O'Reilly Media, Inc. 的授权。此简体中
文版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

深入浅出程序设计

出版发行: 东南大学出版社

地 址: 南京四牌楼 2 号 邮编: 210096

出 版 人: 江建中

网 址: <http://www.seupress.com>

电子邮件: press@seupress.com

印 刷: 扬中市印刷有限公司

开 本: 787 毫米 × 980 毫米 12 开本

印 张: 36 印张

字 数: 454 千字

版 次: 2012 年 1 月第 1 版

印 次: 2012 年 1 月第 1 次印刷

书 号: ISBN 978-7-5641-2966-8

印 数: 1~3500 册

定 价: 98.00 元 (册)

本社图书若有印装质量问题, 请直接与读者服务部联系。电话 (传真): 025-83792328

对《深入浅出程序设计》的高度赞誉

“本书在使用迭代法来教授编程技能方面做得很好。添加一点点，解释一点点，让程序变得更好一点点。这就是真实世界中的编程工作，而本书在教学讨论中用到了这一过程。我把本书推荐给所有那些想要开始学习编程却不知道从哪儿入手的人。我也把本书推荐给那些对编程本身并不陌生，但是对Python感到好奇的人。对于普通的编程和用Python编程本书都是一个很好的开始。”

—— Jeremy Jones, 《Python for Unix and Linux System Administration》的共同作者

“David Griffiths 和 Paul Barry 在 ‘Head First’ 系列中打造了当前最流行的好东西。你有没有使用着一台计算机，但是对总是使用别人的软件感到厌烦？有没有什么是你希望你的计算机能做到，但不是现在已经被设计好了的？在本书中，你将会学习如何写代码，以及如何让计算机按照你自己的方式工作。”

—— Bill Mietelski, 软件工程师

“本书为一个复杂的问题提供了一种独特的方法。前面几章使用了非常棒的比喻来介绍一些基本的编程概念，这些概念为本书后面的章节奠定了基础。这本书里什么都有，从网站开发到图形用户界面以及游戏编程。”

—— Doug Hellmann, 高级软件工程师, Racemi

“本书使用最好的编程语言之一，对如何编程做了一个很好的介绍，采用一种独特的方式把视觉的感知、学习中的疑惑以及编程教学的练习结合起来，结果变得既好玩又易学。”

—— Ted Leung, 首席软件工程师, Sun Microsystems

对其他“Head First”书籍的称誉

“Kathy 和 Bert的《深入浅出Java》把印刷的书页转变成了你曾经见过的和GUI最接近的东西。以一种随意的轻松的方式，作者把学习Java变成了一种吸引人的发现‘他们接下来会干什么？’的经历。”

—— Warren Keuffel, 软件开发杂志

“除了以引人入胜的风格把你从一窍不通带到高级Java的境界，《深入浅出Java》还提供了大量的实用内容，而这些是被其他类似的书籍当作可怕的‘留给读者的练习……’遗留下来的。它是机灵的，随意的，时尚的和实用的——没有很多书能实现这样的效果，同时还能教会你对象序列化和网络装载协议。”

—— Dr. Dan Russell, User Sciences and Experience Research的主管
IBM Almaden 研究中心 (还在斯坦福大学教授人工智能)

“它是快速的，随意的，有趣的，吸引人的。留心——你真的能学会些什么！”

—— Ken Arnold, Sun Microsystems的前高级软件工程师
(和James Gosling一起，他是Java的创造者)，《The Java Programming Language》
的共同作者

“我的感觉就像一千磅重的书从我的脑袋里被扔出去了。”

—— Ward Cunningham, Wiki的发明者和Hillside Group的奠基者

“正是我们中的那些沉迷于技术，随性低调的领袖级程序员们喜欢的风格。也是非常适合实际开发方案的参考资料——让我的大脑不需要与那些陈旧的、累人的教授们的演讲苦苦挣扎就运转起来。”

—— Travis Kalanick, Scour and Red Swoosh的奠基者
MIT TR100的成员

“有些书你会买，有些书你会一直保留，还有些书你会一直放在桌子上，感谢O'Reilly和‘Head First’的全体工作人员，‘Head First’丛书就是倒数第二种。它们就是那些书页中被折了角的，开裂的，被带到各种地方的书。《深入浅出SQL》是我的书堆中最上面的一本。甚至连我用来评阅的PDF版本都读得又旧又破了。”

—— Bill Sawyer, ATG Curriculum主管, Oracle

“这本书中让人敬佩的精确，幽默和扎扎实实的聪明让它可以帮非程序员出身的人也能很好地思考解决问题的方法。”

—— Cory Doctorow, Boing Boing的编辑之一
《Down and Out in the Magic Kingdom》、《Someone Comes to Town》和
《Someone Leaves Town》的作者

对其他“Head First”书籍的称誉

“我昨天才拿到这本书并开始读它……然后我就停不下来了。它真的很酷。很有趣的同时，它的内容也很丰富，而且说到了点子上。我真的留下了很深的印象。”

—— Erich Gamma, IBM 杰出工程师, 也是《Design Patterns》的共同作者

“我读过的关于软件设计的书中最好玩也最厉害的书之一。”

—— Aaron LaBerge, 技术VP, ESPN.com

“曾经漫长又错误百出的学习过程现在被利落地缩简为一本引人入胜的小册子。”

—— Mike Davidson, CEO, Newsvine, Inc.

“每一章的核心都有优雅的设计，每一个概念的表达都伴随着等量的实用性和机智。”

—— Ken Goldstein, Executive Vice President, Disney Online

“我爱《深入浅出HTML与CSS、XHTML》——它以一种有趣的形式教你学习你所需要知道的所有东西。”

—— Sally Applin, UI设计师与艺术家

“平时阅读关于设计模式的书或文章时，我不得不偶尔用什么东西压一压我的眼睛，来确保我的注意力集中。但是读这本书时就一点也不需要这样做。也许听起来有点奇怪，但是这本书确实让学习设计模式变得很好玩。”

“当其他关于设计模式的书说着‘Buehler…… Buehler…… Buehler……’的时候，这本书却在花车上高喊‘宝贝，摇摆起来!’”

—— Eric Wuehler

“简单地说，我爱这本书。实际上，我当着我妻子的面吻了它。”

—— Satish Kumar

译者序

《深入浅出程序设计》是东南大学出版社的O'Reilly计算机系列图书中的一本，该系列图书以内容丰富，实用，易学易懂，在读者中享有很好的口碑。

《深入浅出程序设计》的读者对象是没有任何编程基础的初学者，读者需要的唯一背景知识就是对编程的兴趣。只要你想自己动手写程序，哪怕你是一个对编程一窍不通的家庭主妇，本书也可以让你迅速上手。读完本书，并且实践过书中设计的编程练习之后，读者完全可以拥有一个程序员的基础技能，让“编程”不再具有神秘感。

最可贵的是，本书读来轻松有趣，全程就像在读一本漫画书。没有大段艰深复杂的文字，没有重重复杂的逻辑概念，拥有多年IT开发和教学经验的作者Paul Barry和David Griffiths精心设计的编程练习和穿插在编程练习中的程序概念的阐述，可以让读者忘了自己是在学习编程，更像是在做一个游戏，而且这个游戏还可以打关和升级。

译者第一次读完本书的最深感触就是，真希望当年自己学编程是由这本书开始的。

以下简单概括一下《深入浅出程序设计》中各章节的主要内容：

第1章列举了一些由普通用户转变为亲自动手编程者的情景和乐趣，并简洁地介绍了编程中变量、指令、分支和循环的概念。第2章介绍了字符串操作的常见方法，包括查找、替换、变换大小写以及如何把字符串转化为数字。第3章介绍了函数、参数、返回值、堆栈和作用域。第4章讲解了如何通过编程来管理文件、操纵数组和进行排序。第5章介绍了哈希和数据库，哈希可以帮助你更有效的组织数据。第6章讲述了如何格式化字符串和创建Python模块。第7章介绍了如何使用Python来编写图形界面。第8章介绍了图形界面中使用的各种数据输入组件，包括输入框、文本框和其他一些常见的组件。第8½章(很像魔法书吧?)介绍了如何捕获和处理异常以及各种消息框的使用。第9章介绍了窗口事件的处理和更复杂的组件类型，比如复选框和滑块。第10章讲解了如何定义你自己的组件和类，它可以让你的编程技能进入更高一个阶段。附录部分提供了更多的学习和使用Python的技巧与经验。

本书的翻译得到了东南大学移动通信国家重点实验室的支持，在此表示特别感谢！由于译者水平有限，加之时间仓促，译文中难免有不妥乃至错误之处，敬请读者不吝指正。译者感谢东南大学出版社对本书翻译工作的大力支持！

蒋雁翔 童健

我们把此书献给面对计算机问出这个问题“我想知道我怎么才能让它做这个……？”的第一人。

还有那些把编程变得足够复杂以至于人们需要我们这样的书才能学习编程的人。

David: 给Dawn，我所知道的最聪明的人。

Paul: 本书献给我的父亲，Jim Barry, 25年前当我需要有人推我一把的时候，就是他把我推向了计算机。推得好。

《深入浅出程序设计》的作者

Paul Barry



Paul Barry最近意识到他已经做了大概25年的编程了，这个事实有点让人惊讶。在这25年里，Paul使用过很多种不同的编程语言，在两个洲的两个国家生活和工作过，结了婚，有了三个孩子(好吧……其实是他的妻子Deirdre生了这三个孩子啦，但是Paul也干了他该干的活了嘛)，完成了计算机专业的学士和硕士学位的学习，写了另外两本书，还在《Linux Journal》上发表了大量的技术文章，而且也没有变成秃顶……可悲的是，这种情况可能正在改变了。

Paul在第一次看到《深入浅出HTML与CSS、XHTML》时就非常喜欢它，而且他立刻意识到这种深入浅出的教学风格用在教授编程上会很棒。他和David非常乐意写出这本书来证明他的直觉是正确的。

Paul在白天的工作是在爱尔兰卡罗理工学院做一名讲师。作为计算机和网络系工作的一部分，Paul把他的工作时间花在了探索、学习和教授那些很酷的编程技术上，这些在他看来是很有趣的事情(关于这个Paul可能还需要给出更多证明)，Paul也希望他的学生们觉得他教的那些东西是有趣的。



David Griffiths

David Griffiths在12岁那年看到了一篇有关Seymour Papert工作的文章之后便开始编程了。15岁的时候，他编写代码实现了Papert的计算机语言LOGO。在大学学习了纯数学之后，他开始给计算机写代码，也给杂志写文章。他做过敏捷开发的培训人员，开发人员，还有车库服务员，只不过不是以这样的顺序干活的。他能用10种以上的语言编写代码，或是一种语言写散文，而当他不在写作、编码或是培训时，他就把大部分的闲暇时间花在旅行上，和他那可爱的妻子——也是“Head First”系列的作者——Dawn一起。

在写《深入浅出程序设计》之前，他写过另外一本书，叫做《深入浅出 Rails》，那是一本很出色的读物，而且可以作为一份贴心的礼物送给任何一个亲近的朋友或是家庭成员。

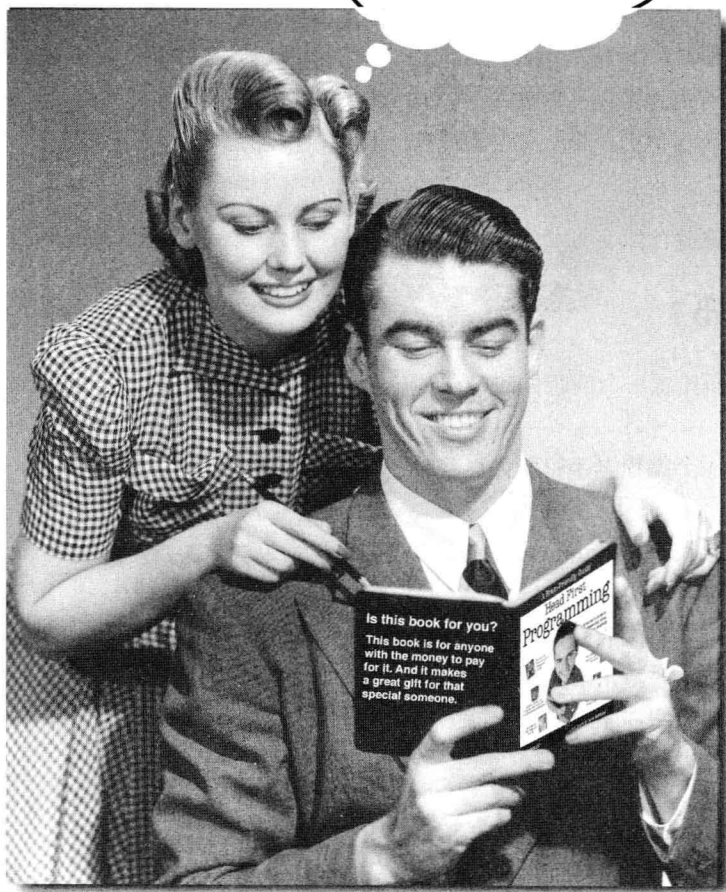
你可以在Twitter上关注他：

<http://twitter.com/dgriffiths>。

如何使用本书

介绍

我无法相信他们把这些东西放在了一本讲编程的书里。



在本节里，我们将回答一个棘手的问题：
“为什么他们把这样的内容放在了一本讲编程的书
中？”

谁适合读这本书？

如果对于下面所有的问题你都回答“是”：

- ❶ 你是否期望了解如何控制你的计算机，让它实现一些新的功能？
- ❷ 你是否想要学习如何编程，使得你能够在软件领域有所开拓，变得富有一些，退休以后可以去你自己的私人小岛？
- ❸ 你是否更愿意做一些实事，运用你所学到的知识，而不是在一个讲座里听某个人喋喋不休地讲上那么几个小时？

嗯，这一点也许扯得有些远了。
但是你需要从一个地方开始，
对不？

那么这本书就适合你。

谁应该远离这本书？

如果对于下面任意一个问题你回答“是”：

- ❶ 你是一个编程老手吗？你已经知道如何编程了吗？
- ❷ 你在寻找一本Python的快速介绍或者参考书吗？
- ❸ 你是否宁愿你的脚趾甲被15个尖叫的猴子拔出也不愿意学习一些新东西？你是否认为一本编程书籍需要包括所有一切，而且如果它能够让读者看得厌恶之极那就更好了？

那么这本书就不适合你。



[来自市场部门的补充：这本书适合于所有具备信用卡的人……我们也接受支票。]

我们知道你在想什么。

“这怎么可能是一本严肃的编程书？”

“那些图片都是些什么？”

“我真的可以通过这种方式从中学到知识？”

我们也知道你的大脑在想什么。

你的大脑渴望新鲜玩意儿。它总是寻找、扫描、等待一些不同寻常的东西。它就是那样被制造出来的，而且它让你能够生存下去。

那么你的大脑会如何对待那些你遇到的常规的、普通的、正常的事情呢？它会尽量阻止这些事情干扰大脑的真正工作——处理真正要紧的事。大脑不会保存那些乏味的事情，它们无法通过“这很显然不重要”的过滤器。

你的大脑如何知道什么是重要的呢？假定你有一天外出远足，然后突然一只老虎跳到了你的面前，你的头脑和身体会怎么做？

神经紧绷，情绪激动，肾上腺素激增！

这就是你的大脑“知道”的方式……

这很重要！请不要忘记它！

但是想象一下你呆在家里，或者在一个图书馆。这是一个安全、温暖、没有老虎的区域。你正在学习，为考试做准备，或者试图学习一些你的老板认为需要花上一周，最多十天就能学会的高深技术。

但是有一个问题：你的大脑在帮你一个大忙。它会试着确定那些显然不重要的内容不会挤占紧缺资源。毕竟，记忆资源最好用来存储真正重要的事情，如遇到老虎与火灾的危险，以及你绝不应该在你的Facebook网页上贴出那些“派对”照片。而且也不会有什么简单的方法来告诉你的大脑：“嘿，脑袋，非常感谢你，但是无论这本书有多无趣，以及我现在所调动的情绪里氏震级（emotion Richter scale）有多小，我真的需要你记住这些东西。”

你的大脑认为“这”才重要。



太好了。还只剩下464页枯燥、乏味、无趣的内容了。

你的大脑认为这不值得记忆。



我们把“Head First”的读者当作初学者。

那么，该怎样来学习新东西呢？首先，你必须了解它，然后你要确保你不会忘记它。这不是说用填鸭的方式对待你。根据认知科学、神经生物学、教育心理学的最新研究，学习过程所需要的不仅仅是纸上的文字。我们知道什么可以让你的大脑“开机”。

一些“Head First”学习守则：

让它可视化。图片比纯文字更容易让人记住，而且也让学习更有效率（在知识的回想和转化上最多有89%的效率提升）。它让学习内容更易理解。把文字放在相关的图片里面或者周围，而不是一页的底部或者下一页，这样初学者可在解读相关内容时达到事半功倍的效果。

使用对话式和拟人化风格。最新的研究表明，如果内容演讲者使用第一人称、对话式风格而不是过于正式的语气来直接与读者对话，学生在课后测试里的表现会有最多40%的提高。讲述故事而不是做报告，使用非正式的语言。不要太严肃。哪一种情况你更容易集中注意力：是晚宴伴侣的耳边细语，还是一场演讲？

让初学者更深入地思考。换一种说法，除非你积极地调动你的神经细胞，否则你的头脑中不会留下太多的东西。读者必须被刺激，亲自参与，产生好奇心并自发去解决问题，得出结论，最后形成新知识。而要做到这一点，你需要接受挑战，勤做练习，以问题诱导思考，用活动活化左右脑并触发多重的感知。

引起——并且保持——读者的注意力。我们都有“我真的想要学这个但是我没办法全神贯注于每一页上”的经验。你的大脑会注意于那些不寻常的、感兴趣的、引人注目的、意想不到的事情。学习一个全新的、困难的技术课题并不一定意味着枯燥。如果它可以变得有趣起来，你的大脑就会学习得快的多。

调动他们的情绪。我们现在知道你的记忆能力很大程度上依赖于情感。你会记住你关心的部分。你会记得你在乎的事，当你心有所感时，你就会记住它。我们不是在讨论一个男孩和他的小狗之间心有灵犀的伤心故事。我们在讨论如惊讶、好奇、有趣、“什么是……？”这类情绪，以及当你解决了一个谜题，学会了其他人认为很难的东西，或者意识到“我比你更懂技术”而工程师Bob就不行时你会拥有的“我说了算”的感觉。

元认知 (metacognition)：关于思考的思考 (想想你如何思考)

如果你真的希望学习，而且你希望学习得更快更深入，请观察你如何集中注意力。想想你如何思考，了解你如何学习。

在我们的成长过程中，我们中的大多数并没有学过元认知或者学习理论的课程。我们希望学习，但是很少被教导如何来学习。

我们假定当你拿着这本书的时候，你真的希望学习如何编程，而且你可能也不希望花费太多的时间。如果你希望使用你从本书中读到的知识，你需要记住你所读到的内容。而要做到这一点，你就需要理解读到的内容。为了从这本书（或者任何书和学习经验）里获得更多的知识，你需要对你的大脑负责。让你的大脑关注到这些内容上。

秘诀是让你的大脑认为你正在学习的新知识确实很重要，与你的生死存亡有关，就像噬人的老虎那样。否则，你就会陷入一场长期战役里，你的大脑依然会尽它最大的努力来使新东西无法留下痕迹。

所以问题在于你如何让你的大脑把编程当成一只饥饿的老虎来看待？

有缓慢且繁琐的方法，也有快速且有效的方法。缓慢的方法就是纯粹的重复。你当然知道再乏味的知识只要你反复学习总能学会和记住。在足够多的重复下，你的大脑会说：“虽然这似乎对他不那么重要，但是他翻来覆去地看这个部分，所以我觉得它应该是重要的。”

而快速的方法就是做任何增强大脑活动的事，尤其是不同类型的大脑活动。前一页提到的就是解决方案的大部分内容，它们都被证明可以用你所喜欢的方式来帮助你的大脑工作。例如，研究表明把语句放到它们所描述的图片里面（而不是置于页面内其他地方，如图片说明或正文）会让你的大脑尝试理解语句和图片的关系，而这会触发更多的神经元。更多激活的神经元就等于让你的大脑有更多机会来知道这是一些值得注意的事情，并且记录下来。

对话风格的帮助是由于当人们感知到他们正在一个对话中时，他们会试图加强注意力，因为他们必须竖起耳朵，注意整个对话的进行。更让人惊喜的是，你的大脑并不在意这个“对话”是发生在你和书本之间！另一方面，如果写作风格是正式而枯燥的话，你的大脑觉得它和你正在经历一场讲座，自己只是一个被动听众，根本没有必要保持清醒。

但是图片和对话式风格还只是开始……



这儿是我们所做的：

我们使用图片，因为你的大脑对视觉化效果比较有感觉，而不是文字。只要能够引起你的大脑的关注，一幅图片胜过千言万语。当文字和图片一起出现时，我们把文字嵌入在图片中，因为当文字放置于它所涉及的图片中时（而不是出现在图片说明或者埋没在正文的某处）可以让你的大脑更有效率地工作。

我们重复表现相同内容，把一件事情用不同的方式和不同的媒介，以及多种感知来增加内容被记录到你的大脑的多个区域的机会。

我们以意想不到的方式来使用概念和图片，因为你的大脑更容易接受新奇的事物，而我们使用的图片和构想带有情感内容，因为你的大脑更容易关注生理情绪。这会导致让你有感觉的事物更容易被记住，即使这种感觉只是一点点幽默、惊讶或者有趣。

我们使用一种拟人化、对话式的风格，因为如果你的大脑认为你处于一个对话而不是被动的听讲时会更容易集中精力。即使你在阅读时，你的大脑也是这样工作的。

我们引入了超过80种活动，因为当你在做事情而不是在读事情时你的大脑会学习和记住更多的内容。而我们让练习维持虽然有挑战性但是依然能做到的程度，因为这是大多数人所期望的。

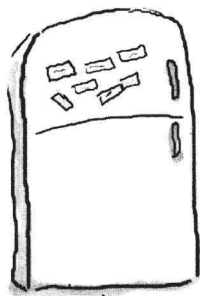
我们使用多种学习风格，因为你可能喜欢循序渐进的过程，有些人可能希望首先了解整体轮廓，而其他一些人则希望看到一个例子。但是无论你倾向于哪种学习方式，每个人都可以通过多种角度来看到相同的内容，从而受益。

我们为你的左右脑都准备了内容，因为你让你的大脑参与得更多，你就越有可能学习和记忆，以及保持更长时间的专注。由于让一边大脑工作通常意味着让另一边有机会休息，你可以有更长时间的学习，更有效率。

我们加入了呈现多个角度的故事和练习，因为你的大脑在它被迫作出评估和决定时能够学习得更深入。

书中也有相当多的挑战和练习，通过问问题的方式进行，答案不见得都很直接，我们的用意是让你的大脑深涉其中，学得更多，记得更牢。想一下——你不可能只是通过观察健身中心的其他人来让自己塑身。但是我们尽我们的最大努力来确保当你刻苦学习时，你所学的是正确的事情。也就是你并没有浪费额外的脑力去处理一个难以理解的例子，或者分析困难的、有着太多术语的或是过于简单的文字。

我们使用人物。在各种故事、例子、图片中处处都是人物，因为，嗯，因为你是一个人。而你的大脑更关注于人而不是其他事物。



沿虚线剪下，粘贴到你的冰箱上。

让你的大脑顺从你的方法

那么，我们完成了我们的部分。剩下的就归你了。这些技巧只是一个开始：请聆听你的大脑，找出哪些适合于你，哪些不行。试试看吧！

1 减慢阅读速度，你理解的越多，你需要强记的就越少。

请不要只是阅读。有时你需要停下来进行思考。当这本书问你一个问题时，不要直接跳到答案部分。想象一下有人真的在问你这个问题。你让你的大脑思考得越深入，你就越有机会学会和记住更多知识。

2 做练习，记笔记。

我们安排了练习题，但是如果我们帮你完成练习题的话，那就像让其他人替你训练一样。不要只是看练习题。请使用你的铅笔。有很多证据表明在学习过程中的实质活动可以强化学习效果。

3 阅读“没有蠢问题”单元。

详细阅读所有的“没有蠢问题”。它们不是无关紧要的说明，而是核心内容的一部分，千万不要忽略它们！

4 让它成为你上床睡觉前最后阅读的内容，或者至少是最后一件有挑战性的事情。

部分学习过程（特别是需要转化为长期记忆的情况下）发生在你放下书本之后。你的大脑也需要一定的时间来对学习的内容进行处理。如果你在它正在处理的时候放入其他的新鲜事物，你刚刚所学习的部分内容就会丢失。

5 谈论它，响亮地说出来。

说话是大脑不同部分的活动。如果你试图理解什么，或者想强化记忆，你应该把它大声地说出来。而更好的做法是，尝试大声地解释给其他人听。这样你就会学得更快，你会发掘出一些你在阅读的时候没有想到的新想法。

6 喝水，大量喝水。

你的大脑浸泡在充沛的液体里时工作状态最佳。脱水状态（往往在你感觉口渴前会发生）会减缓认知能力。

7 聆听你的大脑。

注意你的大脑是否过度疲劳。如果你发现你已经在浮光掠影式地阅读或者开始遗忘你刚刚读到的内容，你就需要休息。一旦你过了一定的时间点，你就不可能通过强塞内容来加快学习速度，而且你这样做还可能破坏学习过程。

8 尽力感觉。

你的大脑需要知道什么是重要的事情。你应该进入到故事中。根据提供的照片展开自己的想象。即使因为一个糟糕的笑话而抱怨也比什么都感觉不到来得强。

9 编写大量的代码！

学习编程只有一种方法：编写大量代码。而这正是你通过这本书所要做的。编码是一种技能，熟能生巧。我们将给你大量的练习：每一章都会有各种练习，它们提出问题，等待你的解决。请不要跳过它们——大量的学习过程发生在你做练习的时候。我们为每个练习提供了答案——如果你做不出来，请大胆翻阅这些答案！（被一些小障碍给卡住是很正常的事情。）但是你需要在查看答案前尽可能地去解决问题。而且在你转到下一部分内容前确保你的答案可以奏效。

读我

这是一本讲述学习经验的书，而不是一本参考书。我们在这本书中特意去掉了那些可能会阻碍学习进程的内容。当你第一次阅读的时候，你需要从首页开始，因为这本书对你已经看过和学过的知识做了一些假设。

这不是《深入浅出Python》。

我们在全书中使用Python 3编程语言，但是这并不会让这本书变成《深入浅出Python》。我们选择Python是因为它是一门非常适合于初学阶段和成长阶段的编程语言。事实上，Python可能是你唯一需要学习和使用的编程语言（虽然你的老板可能不这么想）。当然，你必须以某门语言作为开始，而我们觉得没有什么比Python更适合作为初学编程时所使用的语言。也就是说，这本书不是用来教你如何使用Python，而是用来教你如何编程，所以我们展示给你的大部分内容会强调编程概念，而不是Python特性。

你需要在你的计算机上安装Python 3。

你需要下载和安装Python 3到你的计算机上，这样才能运行本书中的程序。这并不像听起来那么难。去一下Python下载网站，然后选择最适合你所使用的计算机的选项。只要确保你选择了Python 3，而不是Python 2：<http://www.python.org/download>。

我们一开始讲解一些基本的编程概念，然后我们就会立刻开始让你编程。

我们在第1章讲解了编程的基础。这样，当你到达第2章，你就会创建一些真实、有用以及——咕嘟！——有趣的程序。我们猜想你会在第2章大吃一惊的，因为你可以用少于十几行的代码来实现相当多的功能。剩下的部分会培养你的编程技能，快速地把你从一个编程新手转化为编程大师。