

重视大脑的学习指南

# Head First Python (中文版)



将重要的编程  
概念直接送入  
你的大脑

将数据保存  
在pickle中



把你的定制应用  
移植到Web上



用列表、  
集合和字  
典对数据  
建模

连接JSON、  
Android和  
App Engine



在PyPI上与全世界共  
享你的代码

O'REILLY® 中国电力出版社

Paul Barry 著  
林琪 郭静 等译

---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**

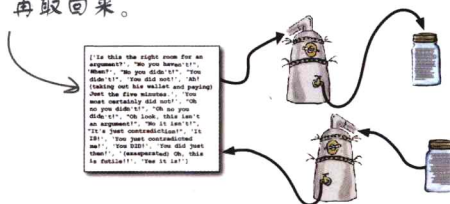
# Head First Python (中文版)

Python/Programming Languages

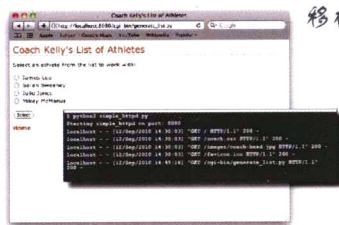
## 您将从本书学会什么?

你想过可以通过一本书就学会Python吗?《Head First Python (中文版)》超越枯燥的语法和用法手册,通过一种独特的方法教你学习这种语言。你会迅速掌握Python的基础知识,然后转向持久存储、异常处理、Web开发、SQLite、数据加工和Google App Engine。你还将学习如何为Android编写移动应用,这都要归功于Python为你赋予的强大能力。本书会提供充分并且完备的学习体验,帮助你成为一名真正的Python程序员。

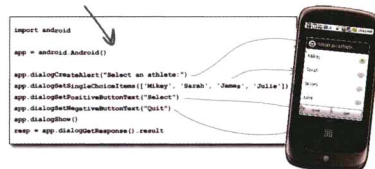
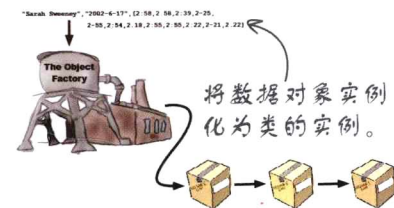
将数据放入一个pickle.....  
再取回来。



将脚本从Python shell  
移植到Web。



将Web应用作为一个Android  
应用移植到手机上。



## 这本书为何与众不同?

我们觉得你的时间相当宝贵,不应当过多地花费在与新概念的纠缠之中。通过应用认知科学和学习理论的最新研究成果,《Head First Python (中文版)》可以让你投入一个需要多感官参与的学习体验,这本书采用丰富直观的形式使你的大脑真正开动起来,而不是长篇累牍地说教,让你昏昏欲睡。

“《Head First Python (中文版)》不单纯是一本优秀的Python语言入门书,更棒的是,它充分展示了Python在现实世界中如何使用。这本书并不是罗列干巴巴的语法,它会教你如何为Android手机、Google App Engine等创建应用程序。”

—— David Griffiths,  
图书作者和敏捷教练

“其他书总是先从理论入手,然后过渡到示例,《Head First Python (中文版)》则不然,它直接进入代码,并随着内容的展开逐步对理论做出解释。书中提供的大量示例和解释足以涵盖你在日常工作中将要用到的大部分内容。”

—— Jeremy Jones,  
《Python for Unix and Linux  
System Administration》  
作者之一

O'Reilly Media, Inc. 授权中国电力出版社出版

O'REILLY®  
oreilly.com.cn  
headfirstlabs.com

ISBN 978-7-5123-2223-3



9 787512 322233 >

定价: 68.00元

此简体中文版仅限于在中华人民共和国境内(但不允许在中国香港、澳门特别行政区和中国台湾地区)销售发行  
This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

# Head First Python

## 中文版

不是在做梦吧？一本Python书居然  
能让你不再对坐在计算机前编写  
代码深恶痛绝？可能只是异想天  
开吧……



Paul Barry 著  
林琪 郭静 等译

**O'REILLY®**

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权中国电力出版社出版

**中国电力出版社**



## 图书在版编目 (CIP) 数据

深入浅出Python: 中文版/ (美) 巴里 (Barry, P.) 著; 林琪等译, -北京: 中国电力出版社, 2011.10

书名原文: Head First Python

ISBN 978-7-5123-2223-3

I. ①深… II. ①巴… ②林… 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 (2011) 第211690号

北京市版权局著作权合同登记

图字: 01-2011-5738号

©2010 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and China Electric Power Press, 2011.

Authorized translation of the English edition, 2010 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2010。

简体中文版由中国电力出版社出版, 2011。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。

书 名/	深入浅出 Python (中文版)
书 号/	ISBN 978-7-5123-2223-3
责任编辑/	刘炽
封面设计/	Karen Montgomery, 张健
出版发行/	中国电力出版社
地 址/	北京市东城区北京站西街 19 号 (邮政编码 100005)
印 刷/	航远印刷有限公司
开 本/	880 毫米×1230 毫米 20 开本 25 印张 665 千字
版 次/	2012 年 3 月第 1 版 2012 年 6 月第 2 次印刷
印 数/	3001—6000 册
定 价/	68.00 元 (册)

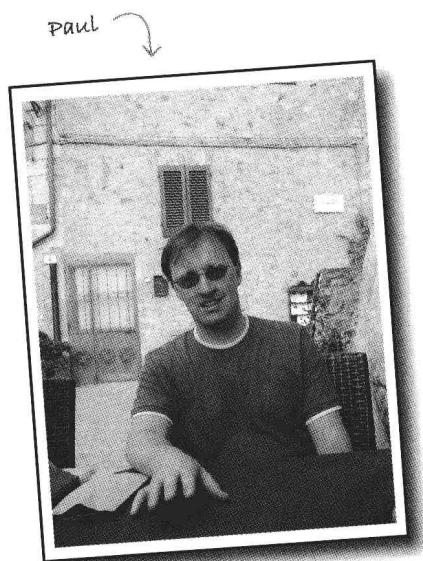
### 敬告读者

本书封面贴有防伪标签, 加热后中心图案消失

本书如有印装质量问题, 我社发行部负责退换

版 权 专 有 翻 印 必 究

## Head First Python的作者



Paul Barry最近发现他的编程生涯已近四分之一世纪，这个事实着实让人有些震惊。在此期间，Paul使用过多种不同的编程语言编写程序，他在两个大洲的两个国家生活并工作过，并且娶妻生子，如今已经有3个孩子（当然……实际上孩子们都是他妻子在悉心照顾，不过Paul确实在他们身边），另外他还攻读了计算机的学士和硕士学位，编写或合作编写了另外3本书，还为《Linux Journal》（他是这家杂志的特约编辑）撰写了大量技术文章。

Paul从第一眼看到《Head First HTML with CSS & XHTML》就爱不释手，当时就意识到“Head First”方法必将成为教授编程的一种绝妙方法。那时他欣喜万分，同样兴奋的还有David Griffiths，他们共同完成了《Head First Programming》来证明当初的预感并非妄想。

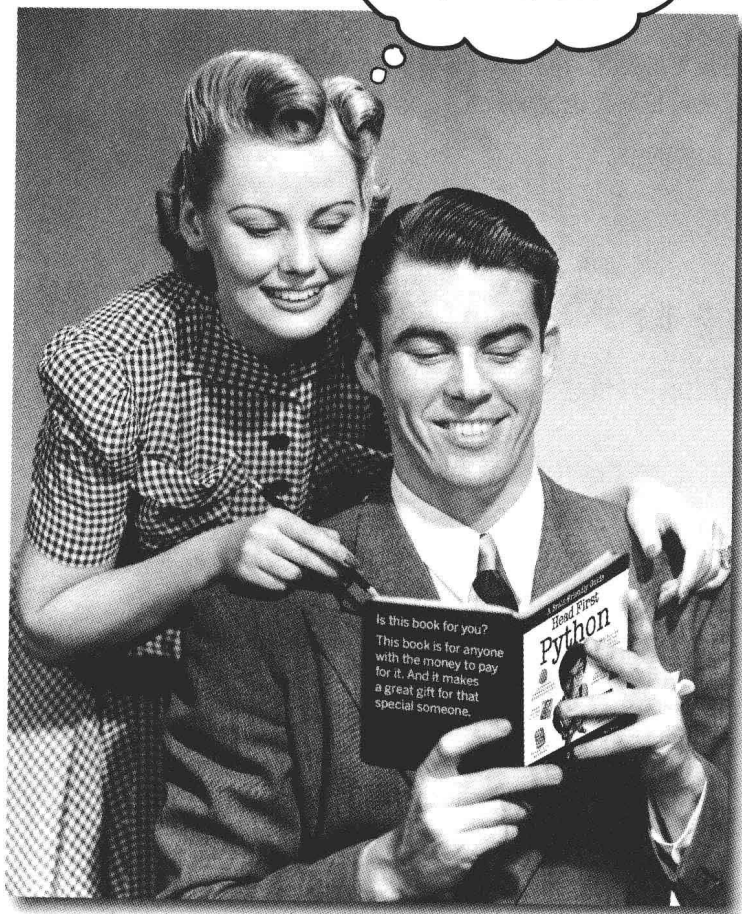
Paul平日的工作是爱尔兰卡罗理工学院的一名讲师。作为计算与网络系的老师，Paul每天都在研究、学习以及向学生们传授编程技术，其中也包括Python。

最近Paul拿到了“课程与教学”研究生毕业证书，终于放心地发现他所做的大多数工作确实符合当今的第三级最佳实践。

# 如何使用这本书

## 引子

真是无法相信，这样一些东西也能放在一本Python编程书里！



有一个问题真是听得我们耳朵都磨出茧了，这就是：“你们到底为什么要把这样一些东西放在一本Python书里呢？”这一部分正是要回答这个问题。

## 谁适合看这本书？

如果对下面的所有问题都能肯定地回答“是”：

- ❶ 你是不是已经知道如何用另外一种编程语言编程？
- ❷ 你是不是希望掌握Python编程的诀窍，想把它补充到你的工具集中，并用它完成一些新的创举？
- ❸ 你是不是更愿意亲自动手，在实践中应用所学，而不只是听别人长篇大论地说教？

那么，这正是你要的书。

## 谁可能不适合看这本书？

如果满足下面任何一种情况：

- ❶ 你是不是已经了解Python编程中需要知道的绝大多数内容？
- ❷ 你是不是正在找一本Python参考书，希望它能极其详尽地涵盖所有细节？
- ❸ 你是不是宁愿脚趾甲被15只尖叫的猴子拔掉也不愿意学新东西？是不是认为Python书就应该无所不包，即使这会让读者厌烦不已，也觉得这样反而更好？

那么，这本书并不适合你。

[来自市场的声音：任何一个有信用卡的人都可以拥有这本书……当然，我们也收支票。]





## 我们知道你在想什么

“这算是一本正式的Python书吗？”

“这些图用来做什么？”

“我真的能这样学吗？”

## 我们也知道你的大脑正在想什么

你的大脑总是渴求一些新奇的东西。它一直在搜寻、审视、期待着不寻常的事情发生。大脑的构造就是如此，正是这一点才让我们不至于墨守成规，能够与时俱进。

我们每天都会遇到许多按部就班的事情，这些事情很普通，对于这样一些例行的事情或者平常的东西，你的大脑又是怎么处理的呢？它的做法很简单，就是不让这些平常的东西妨碍大脑真正的工作。那么什么是大脑真正的工作呢？这就是记住那些确实重要的事情。它不会费心地去记乏味的东西。就好像大脑里有一个筛子，这个筛子会筛掉“显然不重要”的东西，如果遇到的事情枯燥乏味，这些东西就无法通过这个筛子。

那么你的大脑怎么知道到底哪些东西重要呢？打个比方，假如你某一天外出旅行，突然一只大老虎跳到你面前，此时此刻，你的大脑还有身体会做何反应？

神经元会“点火”，情绪爆发，释放出一些化学物质。

好了，这样你的大脑就会知道……

这肯定很重要！可不能忘记了！

不过，假如你正待在家里或者坐在图书馆里，这里很安全、很舒适，肯定没有老虎。你正在刻苦学习，准备应付考试。也可能想学一些比较难的技术，你的老板认为掌握这种技术需要一周时间，最多不超过十天。

这就存在一个问题。你的大脑很想给你帮忙。它会努力地把这些显然不太重要的内容赶走，保证这些东西不去侵占本不算充足的脑力资源。这些资源最好还是用来记住那些确实重要的事情，比如大老虎，遭遇火灾险情等。再比如，你的大脑会让你记住，绝对不能把“聚会”时狂欢的照片放在你的Facebook网页上。没有一种简单的办法来告诉大脑：“嘿，大脑，真是谢谢你了，不过不管这本书多没意思，也不管现在我对它多么无动于衷，但我确实希望你能把这些东西记下来。”



你的大脑认为，这些根本不值得记下来。



## 我们认为“Head First”读者就是要学习的人。

那么，怎么学习呢？首先必须获得知识，然后保证自己确实不会忘记。这可不是填鸭式的硬塞。根据认知科学、神经生物学和教育心理学的最新研究，学习的途径相当丰富，绝非只是通过书本上的文字。我们很清楚怎么让你的大脑兴奋起来。

下面是一些Head First学习原则：

**看得到。**与单纯的文字相比，图片更能让人记得住，通过图片，学习效率会更高（对于记忆和传递型的学习，甚至能有多达89%的效率提升）。而且图片更能让人看懂。以往总是把图片放在一页的最下面，甚至放在另外的一页上，与此不同，把文字放在与之相关的图片内部，或者在图片的周围写上相关文字，学习者的能力就能得到多至两倍的提高，从而能更好地解决有关问题。

**采用一种针对个人的交谈式风格。**最新的研究表明，如果学习过程中采用一种第一人称的交谈方式直接向读者讲述有关内容，而不是用一种干巴巴的语调介绍，学生在学习之后的考试中成绩会提高40%。正确的做法是讲故事，而不是做报告。要用通俗的语言。另外不要太严肃。如果你面对着这样两个人，一个是你在餐会上结识的很有意思的朋友，另一个人学究气十足，喋喋不休地对你说教，在这两个人中，你会更注意哪一个呢？

**让学习的人想得更深。**换句话说，除非你很积极地让神经元活动起来，否则你的头脑里什么也不会发生。必须引起读者的好奇，促进、要求并鼓励读者去解决问题、得出结论、产生新的知识。为此，需要发出挑战，留下练习题和拓宽思路的问题，并要求读者完成一些实践活动，让左右脑都动起来，而且要利用到多种思维。

**引起读者的注意，而且要让他一直保持注意。**我们可能都有过这样的体验，“我真的想把这个学会，不过看过一页后实在是让我昏昏欲睡。”你的大脑注意的是那些不一般、有意思、有些奇怪、抢眼的、意料之外的东西。学习一项有难度的新技术并不一定枯燥。如果学习过程不乏味，你的大脑很快就能学会。

**影响读者的情绪。**现在我们知道了，记忆能力很大程度上取决于所记的内容对我们的情绪有怎样的影响。如果是你关心的东西，就肯定记得住。如果让你感受到了什么，这些东西就会留在你的脑海中。不过，我们所说的可不是什么关于男孩与狗的伤心故事。这里所说的情绪是惊讶、好奇、觉得有趣、想知道“什么……”还有就是一种自豪感，如果你解决了一个难题，学会了所有人都觉得很难的东西，或者发现你了解的一些知识竟是那些自以为无所不能的傲慢家伙所不知道的，此时就会有一种自豪感油然而生。

## 元认知：有关思考的思考

如果你真的想学，而且想学得更快、更深，就应该注意你怎样才会专注起来，考虑自己是怎样思考的，并了解你的学习方法。

我们中间大多数人长这么大可能都没有上过有关元认知或学习理论的课程。我们想学习，但是很少有人教我们怎么来学习。

不过，这里可以做一个假设，如果你手上有这本书，你想学习如何设计用户友好的网站，而且可能不想花太多时间。如果你想把这本书中读到的知识真正用起来，就需要记住你读到的所有内容。为此，必须理解这些内容。要想最大限度地利用这本书或其他任何一本书，或者掌握学习经验，就要让你的大脑负起责任，要求它记住这些内容。

怎么做到呢？技巧就在于要让你的大脑认为你学习的新东西确实很重要，对你的生活有很大影响。就像老虎出现在面前一样。如若不然，你将陷入旷日持久的拉锯战中，虽然你很想记住所学的新内容，但是你的大脑却会竭尽全力地把它拒之门外。

那么究竟怎样才能让你的大脑把编程看作是一只饥饿的老虎呢？

这两条路，一条比较慢，很乏味。另一条路不仅更快，还更有效。慢方法就是大量地重复。你肯定知道，如果反反复复地看到同一个东西，即便再没有意思，你也能学会并记住。如果做了足够的重复，你的大脑就会说，“尽管看上去这对他来说好像不重要，不过，既然他这样一而再、再而三地看同一个东西，所以我觉得这应该是重要的。”

更快的方法是尽一切可能让大脑活动起来，特别是开动大脑来完成不同类型的活动。如何做到这一点呢？上一页列出的学习原则正是一些主要的做法，而且经证实，它们确实有助于让你的大脑全力以赴。例如，研究表明，把文字放在所描述图片的中间（而不是放在这一页的别处，比如作为标题，或者放在正文中），这样会让你的大脑更多地考虑这些文字与图片之间有什么关系，而这就会让更多的神经元点火。让更多的神经元点火 = 你的大脑更有可能认为这些内容值得关注，而且很可能需要记下来。

交谈式风格也很有帮助，当人们意识到自己在与“别人”交谈时，往往会更专心，这是因为他们总想跟上谈话的思路，并能做出适当的发言。让人惊奇的是，大脑并不关心“交谈”的对象究竟是谁，即使你只是与一本书“交谈”，它也不会在乎！另一方面，如果写作风格很正统、干巴巴的，你的大脑就会觉得，这就像坐在一群人中被动地听人作报告一样，很没意思，所以不必在意对方说的是什么，甚至可能打瞌睡。

不过，图片和交谈风格还只是开始而已，能做的还有很多……



## 我们是这么做的：

我们用了很多图，因为你的大脑更能接受看得见的东西，而不是纯文字。对你的大脑来说，一幅图抵千言。如果既有文字又有图片，我们会把文字放在图片当中，因为文字处在所描述的图片中间时，大脑的工作效率更高，倘若把这些描述文字作为标题，或者“湮没”在别处的大段文字中，就达不到这种效果了。

我们采用了重复手法，会用不同方式，采用不同类型的媒体，运用多种思维手段来介绍同一个东西，目的是让有关内容更有可能储存在你的大脑中，而且在大脑中多个区域都有容身之地。

我们会用你想不到的方式运用概念和图片，因为你的大脑喜欢新鲜玩艺。在提供图和思想时，至少会含着一些情绪因素，因为如果能产生情绪反应，你的大脑就会投入更大的注意。而这会让你感觉到这些东西更有可能要被记住，其实这种感觉可能只是很幽默，让人奇怪或者比较感兴趣而已。

我们采用了一种针对个人的交谈式风格，因为当你的大脑认为你在参与一个会谈，而不是被动地听一场演示汇报时，它就会更加关注。即使你实际上在读一本书，也就是说在与书“交谈”，而不是真正与人交谈，但这对你的大脑来说并没有什么分别。

在这本书里，我们加入了80多个实践活动，因为与单纯的阅读相比，如果能实际做点什么，你的大脑会更乐于学习，更愿意去记。这些练习都是我们精心设计的，有一定的难度，但是确实能做出来，因为这是大多数人所希望的。

我们采用了多种学习模式，因为尽管你可能想循序渐进地学习，但是其他人可能希望先对整体有一个全面的认识，另外可能还有人只是想看一个例子。不过，不管你想怎么学，要是同样的内容能以多种方式来表述，这对每一个人都会有好处。

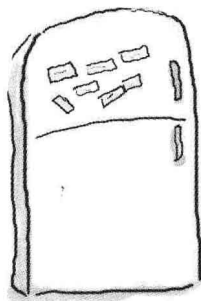
这里的内容不只是单单涉及左脑，也不只是让右脑有所动作，我们会让你的左右脑都动起来，因为你的大脑参与得越多，你就越有可能学会并记住，而且能更长时间地保持注意力。如果只有一半大脑在工作，通常意味着另一半有机会休息，这样你就能更有效率地学习更长时间。

我们会讲故事，留练习，从多种不同的角度来看同一个问题，这是因为，如果要求大脑做一些评价和判断，它就能更深入地学习。

我们会给出一些练习，还会问一些问题，这些问题往往没有直截了当的答案，通过克服这些挑战，你就能学得更好，因为让大脑真正做点什么的话，它就能学会并记住。想想吧，如果只是在体育馆里看着别人流汗，这对于保持你自己的体形肯定不会有什么帮助，正所谓临渊羡鱼，不如退而结网。不过另一方面，我们会竭尽所能不让你钻牛角尖，把劲用错了地方，而是能把功夫用在点子上。也就是说，你不会为搞定一个难懂的例子而耽搁，也不会花太多时间去弄明白一段艰涩难懂而且通篇行话的文字，我们的描述也不会太过简洁而让人无从下手。

我们用了拟人手法。在故事中，在例子中，还有在图中，你都会看到人的出现，这是因为你本身是一个人，不错，这就是原因。如果和人打交道，相对于某件东西而言，你的大脑会更为关注。





把这一页撕下来，  
贴到你的冰箱上。

## 可以用下面的方法让你的 大脑就范

好了，我们该做的已经做了，剩下的就要看你自己的了。以下提示可以作为一个起点：听一听你的大脑是怎么说的，弄清楚对你来说哪些做法可行，哪些做法不能奏效。要尝试新鲜事物。

### ❶ 慢一点。你理解的越多，需要记的就越少。

不要光是看看就行了。停下来，好好想一想。书中提出问题的時候，你不要直接去翻答案。可以假想真的有人在问你这个问题。你让大脑想得越深入，就越有可能学会并记住它。

### ❷ 做练习，自己记笔记。

我们留了练习，但是如果这些练习的解答也由我们一手包办，那和有人替你参加考试有什么分别？不要只是坐在那里看着练习发呆。拿出笔来，写一写画一画。大量研究都证实，学习过程中如果能实际动手，这将改善你的学习。

### ❸ 阅读“没有傻问题”。

顾名思义。这些问题不是可有可无的旁注，它们绝对是核心内容的一部分！千万不要跳过去不看。

### ❹ 上床睡觉之前不要再看别的书，至少不要看其他有难度的东西。

学习中有一部分是在你合上书之后完成的（特别是，要把学到的知识长久地记住，这往往无法在看书的过程中做到）。你的大脑也需要有自己的时间，这样才能再做一些处理。如果在这段处理时间内你又往大脑里灌输了新的知识，那么你刚才学的一些东西就会丢掉。

### ❺ 讲出来，而且要大声讲出来。

说话可以刺激大脑的另一部分。如果你想看懂什么，或者想更牢地记住它，就要大声地说出来。更好的办法是，大声地解释给别人听。这样你会学得更快，而且可能会有以前光看不说时不曾有的新发现。

### ❻ 要喝水，而且要大量喝水。

能提供充足的液体，你的大脑才能有最佳表现。如果缺水（可能在你感觉到口渴之前就已经缺水了），学习能力就会下降。

### ❼ 听听你的大脑怎么说。

注意一下你的大脑是不是负荷太重了。如果发现你自己开始浮光掠影地翻看，或者刚看的东西就忘记了，这说明你该休息一会了。达到某个临界点时，如果还是一味地向大脑里塞，这对于加快学习速度根本没有帮助，甚至还可能影响正常的学习进程。

### ❽ 要有点感觉。

你的大脑需要知道这是很重要的东西。要真正融入到书中的故事里。为书里的照片加上你自己的图题。你可能觉得一个笑话很蹩脚，但这总比根本无动于衷要好。

### ❾ 编写大量软件！

要学习编程，没有别的办法，只能通过编写大量代码。这本书正是要这么做。编写代码是一种技巧，要想在这方面擅长，只能通过实践。我们会给你提供大量实践的机会：每一章都留有练习，提出问题让你解决。不要跳过这些练习，很多知识都是在完成这些练习的过程中学到的。我们为每个练习都提供了答案，如果你实在做不出来（很容易被一些小问题卡住），看看答案也无妨！不过在看答案之前，还是要尽力先自己解决问题。而且在读下一部分之前，一定要确实实地掌握前面的内容。

## 重要说明

要把这看做是一个学习过程，而不要简单地把它看成是一本参考书。我们在安排内容的时候有意做了一些删减，只要是对有关内容的学习有妨碍，我们都毫不留情地把这些部分删掉。另外，第一次看这本书时，要从第一页从头看起，因为书中后面的部分会假定你已经看过而且学会了前面的内容。

**这本书特别设计为使你能对Python尽快上手。**

既然你想学真功夫，这里就会教你真功夫。所以，在这本书中不会看到长篇大论的技术内容，这里不会用干巴巴的表格罗列Python的操作符，也不会给出枯燥的操作符优先级规则。所有这些都没有，不过我们会精心安排，尽可能涵盖所有基础知识，使你能把Python尽快记入大脑并永远留住。我们只做了一个假设，认为你已经知道如何用另外某种编程语言编写程序。

**这本书面向Python 3。**

这本书中使用Python编程语言的版本3，第1章会介绍如何得到和安装Python 3。当然，我们并不是完全忽略版本2，这一点你在第8章到第11章就会发现。不过请相信，你会庆幸使用Python，因为你根本不会注意到你编程实现的技术是在Python 2上运行。

**我们会直接让Python投入工作。**

从第1章开始你就会用Python做些有用的工作。这里不会绕弯子，因为我们希望你能立即用Python开展工作。

**书里的实践活动不是可有可无的。**

这里的练习和实践活动不是可有可无的装饰和摆设，它们也是这本书核心内容的一部分。其中有些练习和活动有助于记忆，有些则能够帮助你理解，还有一些对于如何应用所学的知识很有帮助。千万不要跳过这些练习不做。

我们有意安排了许多重复，这些重复非常重要。

Head First系列图书有一个与众不同的地方，这就是，我们希望你确实实地掌握这些知识，另外希望在学完这本书之后你能记住学过了什么。大多数参考书都不太重视重复和回顾，但是由于这是一本有关学习的书，你会看到一些概念一而再、再而三地出现。

代码例子尽可能短小精悍。

有读者告诉我们，如果查了200行代码才能找到要理解的那两行代码，这很让人郁闷。这本书里大多数例子往往都开门见山，作为上下文的代码会尽可能的少，这样你就能一目了然地看到哪些东西是需要你学习的。别指望这些代码很健壮，甚至别指望它们是完整的。我们特意把这些例子写得很简单，以便于你学习，它们的功能往往不太完备。

我们在网上放了大量代码示例，你可以根据需要复制和粘贴。可以从以下两个网址下载：

*<http://www.headfirstlabs.com/books/hfpython/>*

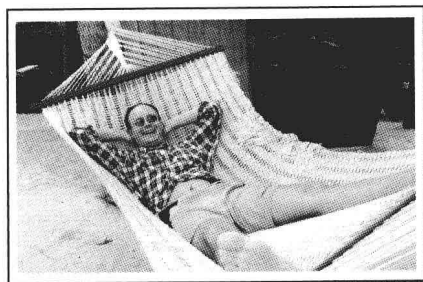
*<http://python.itcarlow.ie>*

“头脑风暴”练习没有答案。

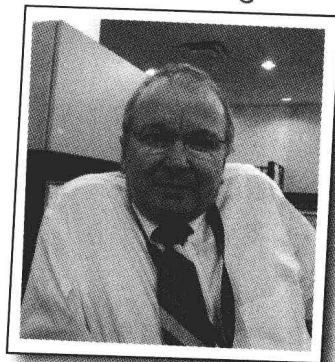
有一些头脑风暴练习根本没有正确的答案，对于另外一些练习，头脑风暴实践活动中有一部分学习过程就是让你确定你的答案是否正确以及在何种情况下正确。在其中一些头脑风暴练习中，你会得到一些提示来指出正确的方向。

## 技术审校团队

David Griffiths



Phil Hartley



Jeremy Jones



### 技术审校：

David Griffiths是《Head First Rails》的作者，同时也是《Head First Programming》的作者之一。他12岁时看到一份关于Seymour Papert工作的资料，从那时起就开始了他的编程生涯。15岁时，他编写了Papert的计算机语言LOGO的实现程序。在大学学习了纯粹数学之后，他开始为计算机编写代码，另外还为人们撰写杂志文章。他身兼多职，既是敏捷方法教练和开发人员，同时还是一个车厂修理工（不过这些角色并没有先后顺序）。他可以用十余种语言编写代码，也可以用一种语言完全搞定。如果不是在写书、编程或者辅导别人，他的闲暇时光大多会用来与他挚爱的妻子Dawn（也是Head First系列的作者之一）四处旅游。

Phil Hartley已获得苏格兰爱丁堡大学的计算机科学学位。经过在IT行业30多年的摸爬滚打，特别在OOP方面积累了丰富的经验。目前他在亚利桑那州滕比的高级科技大学任全职教师。空闲时间里，Phil是一个狂热的NFL球迷。

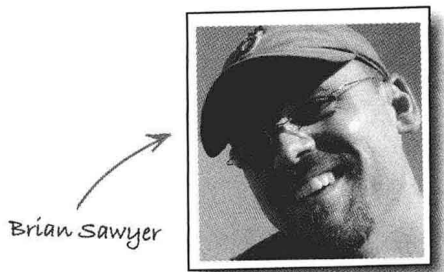
Jeremy Jones是《Python for Unix and Linux System Administration》的作者之一。2001年以来，他一直在积极地使用Python。他做过开发人员、系统管理员、质量工程师，还做过系统分析员。这些职业都各有回报和挑战，不过对他来说，最富有挑战同时回报最大的“职业”当属丈夫和父亲。



## 致谢

### 致我的编辑：

Brian Sawyer是这本书的编辑。如果不是在编辑图书，Brian空闲时喜欢跑马拉松。与我合作再出一本书（这是我们的二次联手）绝对是一个很好的训练。O'Reilly和Head First团队真的很幸运，能拥有像Brian这样有着过人才干的高手，使这本书以及其他书能够最完美地呈现在大家面前。



### 致O'Reilly团队：

Karen Shaner提供了管理支持，并且运用他的卓越才能很好地协调了技术审校过程，对于我提出的众多请求和询问，都迅速作答，并全力提供帮助。还要感谢那些幕后的人们——O'Reilly制作团队，是他们指导这本书顺利进入最终的出版环节，将我的InDesign原稿文件变成你手上这本精美的图书（也可能你在用iPad或Android手机阅读，或者在你的PC机上读这本书，那同样也很酷）。

还要感谢Head First系列的其他作者，在本书的整个编写过程中，他们通过Twitter给予了我充分的赞赏、建议和鼓励。你可能想不到140个字符能带来天壤之别，但事实上确实如此。

另外还要向Bert Bates以及Kathy Sierra致以诚挚的谢意，正是从他们绝妙的《Head First Java》开始，逐步成就了今天的Head First系列。最开始写这本书时，Bert曾与我做过一次长达90分钟的“马拉松”电话交流，帮助我确定这本书的基调，使我得以将思维延伸至极致，让这本书更为出色。如今，这个电话已经过去9个月了，我才刚刚从Bert带给我的震撼和兴奋中恢复过来。

### 致我的朋友和同事：

再次感谢卡罗理工学院计算与网络系主任Nigel Whyte能够支持我编写另一本书（特别是距上一本书的交付出版还时隔不久）。

我的学生们（游戏开发专业大三学生和软件工程专业大四学生）在过去18个月中曾以不同方式接触过这本书的内容。他们对Python的积极回应以及我在课上采用的方法都对这本书的结构以及最后内容的形成有很大帮助（没错，有些内容正是期末试卷上的题目）。

### 致我的家人：

我亲爱的家人Deirdre、Joseph、Aaron和Aideen不得不再一次忍受我的牢骚连连、吹胡子瞪眼，还有不时发作的坏脾气（不过，说实在的，与写《Head First Programming》时相比，这一次我发火的次数要少一点了）。完成上一本书之后，我答应“一段时间内”不再变成那个脾气暴躁的形象。可惜这“一段时间”仅仅保持了几个星期而已，我衷心感谢家人们没有因为我的失言集合起来把我扔出门外。如果没有他们的支持，特别是我的妻子Deirdre无尽的爱和支持，这本书绝无可能问世。

### 一个也不能少：

感谢我的技术审校团队出色的工作，是他们让我没有脱离正轨，保证我所说的准确无误。他们对正确内容做出确认，对不正确的部分提出质疑，不仅指出哪里有问题，还给出了具体的修改建议。特别是David Griffiths，他是《Head First Programming》的合作者，他做出的技术评论远远超出他的职责范围。虽然这本书封面上没有David的名字，但是书中很多内容都源于他的想法和建议。作为《Head First Python》的技术审校人员，他对这个角色如此热情的投入让我充满敬意，而且永远感激不尽。

## Safari®图书在线



Safari图书在线是一个按需提供资源的数字图书馆，从中可以很容易地搜索7500本技术书、参考书和视频，快速找到你想要的答案。

只需订阅，就可以从我们的在线图书馆阅读任何页面，观看任何视频。你可以在你的手机和移动设备上看书，能够在新书出版之前获得书目，还可以享有特权查看正在编写的书稿并向作者提出反馈意见。你可以剪切粘贴代码示例、整理最喜欢的图书、下载你需要的章节、为重要内容设置书签、创建笔记、打印页面，此外还有大量节省时间的特性可以让你从中受益。

O'Reilly Media公司已经将这本书英文版上传到Safari图书在线服务。要想通过数字方式全面访问这本书以及O'Reilly和其他出版商提供的其他类似图书，可以免费注册 <http://my.safaribooksonline.com/?portal=oreilly>。

目录（概览）

引子	xxiii
1 初识Python：人人都爱列表	1
2 共享你的代码：函数模块	33
3 文件与异常：处理错误	73
4 持久存储：数据保存到文件	105
5 推导数据：处理数据！	139
6 定制数据对象：打包代码与数据	173
7 Web开发：集成在一起	213
8 移动应用开发：小设备	255
9 管理你的数据：处理输入	293
10 扩展你的Web应用：来真格的	351
11 处理复杂性：数据加工	397
i 其他：（我们没有谈到的）十大问题	435
索引	447

详细目录

引子

**你的大脑与Python。**你想学些新东西，但是你的大脑总是帮倒忙，它会努力让你记不住所学的东西。你的大脑在想：“最好留出空间来记住那些确实重要的事情，比如要避开哪个野生动物，还有裸体滑雪是不是不太好？”那么，如何让你的大脑就范？让它认为如果不知道Python你将无法生存！

谁适合看这本书？	xxiv
我们知道你的大脑在想什么	xxv
元认知	xxvii
由你让你的大脑就范	xxix
重要说明	xxx
技术审校团队	xxxii
致谢	xxxiii

# 1 初识Python

## 人人都爱列表

你可能会问：“Python有什么与众不同的地方？”如果简短地回答这个问题，那么答案是：很多很多。要想更详细地回答，首先需要指出Python也有很多我们熟悉的东西。类似于所有其他通用编程语言，Python同样有语句、表达式、操作符、函数、模块、方法和类。这些确实都很普通。不过，Python还能提供一样东西，能让程序员（也就是你）的日子更好过一些。下面先从学习列表开始我们的Python之旅。不过，在此之前，还有一个重要的问题需要回答……

Python有什么过人之处?	2
安装Python 3	3
使用IDLE来帮助学习Python	4
有效地使用IDLE	5
处理复杂数据	6
创建简单的Python列表	7
列表就像是数组	9
向列表增加更多数据	11
处理列表数据	15
For循环处理任意大小的列表	16
在列表中存储列表	18
在列表中查找列表	20
复杂数据很难处理	23
处理多层嵌套列表	24
不要重复代码；应当创建一个函数	28
在Python中创建一个函数	29
解决之道：递归!	31
你的Python工具箱	32

The Holy Grail, 1975, Terry Jones & Terry Gilliam, 91 mins  
Graham Chapman  
Michael Palin, John Cleese, Terry Gilliam, Eric Idle & Terry Jones



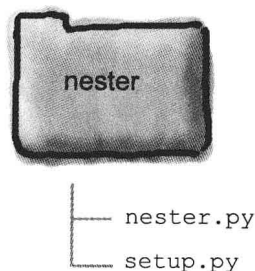
# 2

## 共享你的代码

### 函数模块

可重用的代码固然不错，不过可共享的模块更棒。通过作为Python模块共享代码，就可以向整个Python社区开放你的代码……共享总是一件好事，不是吗？在这一章中，你将学习如何创建、安装和发布你自己的可共享模块，然后把你的模块加载到Web上的Python软件共享网站，这样所有人都能受益于你的工作。在学习过程中，你还会了解一些与Python函数有关的新技巧。

太好了，所以应该分享	34
函数转换为模块	35
模块无处不在	36
注释代码	37
准备发布	40
构建发布	41
发布速览	42
导入模块并使用	43
Python的模块实现命名空间	45
注册PyPI网站	47
向PyPI上传代码	48
欢迎来到PyPI社区	49
用额外的参数控制行为	52
写新代码之前，先考虑BIF	53
Python会尽力运行你的代码	57
跟踪代码	58
找出哪里出了问题	59
用你的新代码更新PyPI	60
你改变了API	62
使用可选参数	63
模块支持两个API	65
API还是不对	66
模块重获声誉	70
你的Python工具箱	71

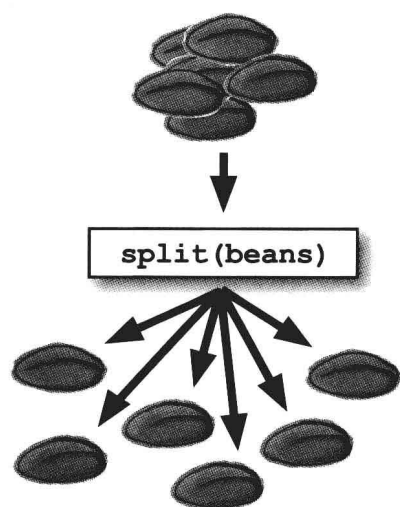


# 3

## 文件与异常

### 处理错误

只是在代码中处理列表数据还不够。你还需要把数据轻松地送入程序。如此说来，不难理解为什么利用Python可以很容易地从文件读取数据。这一点很棒，不过再想想看，与程序之外的数据交互时可能有麻烦……另外还有方方面面的问题在等着给你下绊！出问题时，需要一种策略使你能够从麻烦中脱身，利用Python的异常处理机制来处理异常情况就是这样一种策略，这一章将会介绍这种机制。



程序外部的数据	74
都是文本行	75
进一步查看数据	77
了解你的数据	79
了解你的方法，请求帮助	80
更好地了解你的数据	82
两种截然不同的方法	83
增加额外逻辑	84
处理异常	88
先尝试，然后恢复	89
找出要保护的代码	91
放过错误	93
其他错误呢？	96
增加更多错误检查代码……	97
……或者再增加一层异常处理	98
那么，哪种方法更好呢？	99
大功告成……不过还有一个小问题	101
特定指定异常	102
你的Python工具箱	103

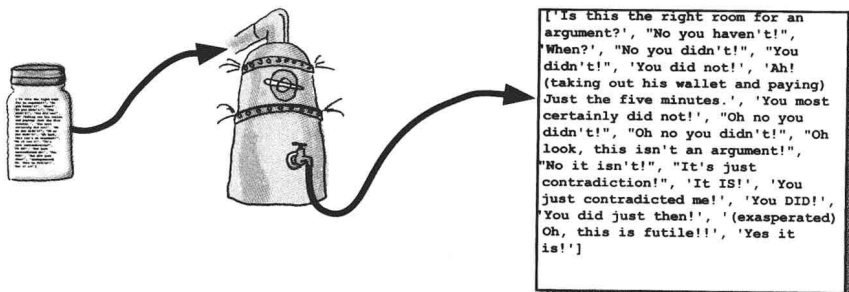
# 4

## 持久存储

### 数据保存到文件

能够处理基于文件的数据确实很棒。但是工作完成时你的数据会怎么样呢？当然，最好把数据保存到一个磁盘文件中，这样就能在以后某个时间再次使用这些数据。将基于内存的数据存储到磁盘上，这正是持久存储的含义。Python支持所有将数据写至文件的常用工具，另外还提供了一些很酷的工具，可以高效地存储Python数据。所以……请翻开下一页，下面开始学习这些内容。

程序生成数据	106
以写模式打开文件	110
发生异常后文件会保持打开！	114
用finally扩展try	115
知道错误类型还不够	117
用with处理文件	120
默认格式对文件并不合适	124
何不修改 print_lol()?	126
“腌制”数据	132
用dump保存，用load恢复	133
使用pickle的通用文件I/O才是上策！	137
你的Python工具箱	138



# 5 推导数据 处理数据！

数据各式各样，有不同的**大小、格式和编码**。要想有效地处理你的数据，通常需要把它处理并转换为一种常用的格式，以便高效处理、排序和存储。在这一章中，我们会研究Python的一些有助于有效处理数据的工具，让你感受到数据处理的非凡之处。好吧，翻开下一页，别让教练等久了……

Kelly教练需要你的帮助	140
排序有两种方式	144
时间的麻烦	148
推导列表	155
迭代删除重复项	161
用集合删除重复项	166
你的Python工具箱	172

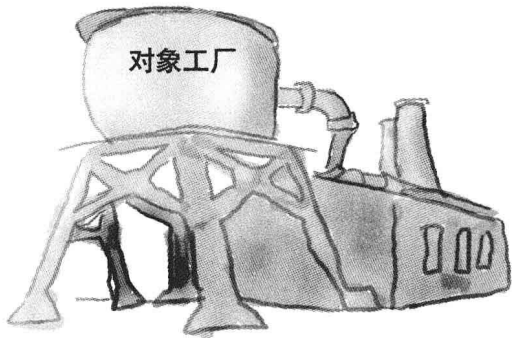


# 6 定制数据对象

## 打包代码与数据

你选择的数据结构要与数据匹配，这很重要。而且这个选择将对代码的复杂性带来很大差别。在Python中，尽管列表和集合确实很有用，但这并不是全部。Python还提供了字典，允许你有效地组织数据，可以将数据与名关联而不是与数字关联，从而实现快速查找。当Python的内置数据结构无法胜任时，Python class语句还允许你定义自己的数据结构。这一章就会介绍这些内容。

Kelly又来了（带来一种新的文件格式）	174
使用字典关联数据	178
将代码及其数据打包在类中	189
定义一个类	190
使用class定义类	191
self的重要性	192
每个方法的第一个参数都是self	193
继承Python内置的list	204
Kelly教练相当满意	211
你的Python工具箱	212



# 7

## Web 开发

### 集成在一起

你迟早会希望与很多人分享你的应用。要做到这一点，你有很多选择。可以把你的代码上传到PyPI，发送大量email邮件，把你的代码放在一个CD或USB上，或者只要有人需要，就直接把应用手动安装在他们的计算机上。听上去要做很多工作……更何况这会让人很头疼。另外，如果你开发出代码的下一个最佳版本又会发生什么情况？你将如何管理代码的更新？面对现实吧，要想出一个有创意的借口确实很困难。幸运的是，你根本不用去找借口，只需创建一个Web应用就行了。另外，正如这一章所要展示的，用Python完成Web开发确实非常轻松。

分享是好事	214
可以把你的程序放在Web上	215
Web应用需要做什么？	218
采用MVC设计Web应用	221
为数据建模	222
查看界面	226
控制你的代码	234
CGI让Web服务器运行程序	235
显示选手列表	236
可怕的404错误！	242
创建另一个CGI脚本	244
启用CGI跟踪来帮助解决错误	248
一个小改变会让一切大不同	250
你的Web应用妙极了！	252
你的Python工具箱	253



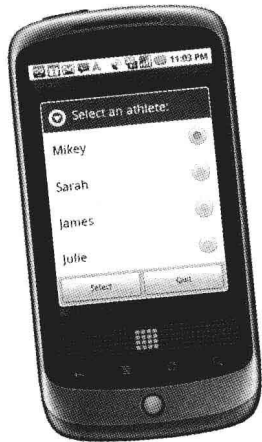
# 8

## 移动应用开发

### 小设备

数据放在Web上就像打开了潘朵拉的盒子。不仅任何人可以从任何地方与你的Web应用交互，而且越来越多的人在通过各种各样的计算设备访问你的Web应用，比如PC、笔记本电脑、平板电脑、掌上电脑，甚至移动电话。现在你不仅需要支持和考虑与Web应用交互的人，还要考虑机器人，也就是能自动完成Web交互的小程序，它们通常只想得到你的数据，并不需要对人类友好的HTML。这一章将在Kelly教练的移动电话上利用Python编写一个应用，来访问Web应用的数据。

世界越来越小	256
Kelly教练在使用Android	257
不用担心Python 2	259
建立开发环境	260
配置SDK和模拟器	261
安装和配置Android脚本环境	262
为SL4A安装增加Python	263
在Android上测试Python	264
定义应用的需求	266
SL4A Android API	274
在Android上选择列表	278
选手数据CGI脚本	281
看起来应该改变数据的类型	284
JSON无法处理你的定制数据类型	285
在真正的手机上运行你的应用	288
配置AndFTP	289
教练对应用大加赞赏	290
你的Python工具箱	291





9

管理你的数据

处理输入

Web和手机并不只是能很好地显示数据。它们也是从用户接收输入的绝妙工具。当然，一旦Web应用接收数据，肯定需要把数据放在某个地方。在决定把数据放在哪里时，你的选择往往会对Web应用带来巨大差别。如果选择得当，Web应用将很易于扩展，否则扩展可能很困难。在这一章中，你将扩展你的Web应用从Web（通过浏览器或从Android手机）接收数据，另外还将了解并改进后台数据管理服务。

你的选手时间应用已经声名远扬	294
使用表单或对话框接收输入	295
创建一个HTML表单模板	296
数据传送到CGI脚本	300
在Android手机上请求输入	304
该更新服务器数据了	308
避免竞态条件	309
需要一个更好的存储机制	310
使用数据库管理系统	312
Python包括SQLite	313
利用Python的数据库API	314
数据库API的相应Python代码	315
小小的数据库设计会带来很大不同	316
定义数据库模式	317
数据是什么样？	318
从pickle向SQLite传输数据	321
为选手指定了什么ID？	322
插入计时数据	323
SQLite数据管理工具	326
SQLite与现有Web应用集成	327
仍然需要名字列表	332
根据ID得到选手的详细信息	333
还需要修改Android应用	342
更新SQLite中的选手数据	348
NUAC非常满意！	349
你的Python工具箱	350



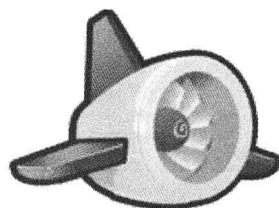
## 10

## 扩展你的Web应用

## 来真格的

你的应用确实很适合放在Web上……除非开始来真格的。总有一天你会走运，你的Web应用可能大获成功。等到那一天，你的Web应用不再每天只有寥寥无几的点击量，可能会达到上千、上万，甚至更多。做好准备了吗？你的Web服务器能处理这么大的负载吗？你又怎么知道它能不能应对？开销有多大？谁来承担费用？你的数据模型能不能扩展到包含数百万的数据项而不会导致应用缓慢如牛？利用Python可以很容易地启动和运行Web应用，而且现在有了Google App Engine，扩展Python Web应用也完全可以轻松实现。

到处都有人看到鲸	352
HFWWG需要自动化	353
用Google App Engine构建Web应用	354
下载和安装App Engine	355
确保App Engine正常工作	356
App Engine使用MVC模式	359
用App Engine对数据建模	360
如果没有视图，模型有什么用？	363
使用App Engine中的模板	364
Django的表单验证框架	368
检查表单	369
控制App Engine Web应用	370
提供选择来限制输入	376
遭遇“死亡白屏”	378
在Web应用中处理POST	379
把数据放在datastore中	380
不要破坏“健壮性原则”	384
接受几乎所有日期和时间	385
看起来你还没有完成	388
有时，最小的改变可能会带来天壤之别……	389
还要捕获用户的Google ID	390
将Web应用部署到Google云	391
HFWWG Web应用已经成功部署！	394
你的Python工具箱	395



# 11

## 处理复杂性

### 数据加工

能在一个特定领域应用Python确实很棒。不论是Web开发、数据库管理还是移动应用，Python都能帮助你完成任务，顺利地编写解决方案。不过除了这些问题，还存在另外一些问题：它们无法归类或关联到某一领域。这些问题本身很特别，必须用一种有区别的、非常特定的方式来考虑。为这些问题创建预定的软件解决方案正是Python擅长的领域。在最后这一章中，你的Python技能将会更上一层楼，从而能很好地解决这些问题。

下一次跑步有没有合适的目标时间?	398
那么……有什么问题吗?	400
从数据开始	401
将各个时间存储为字典	407
预测代码剖析	409
得到用户输入	413
获取输入产生了一个问题……	414
搜索最接近的匹配	416
时间有问题	418
时间一秒转换模块	419
时间还有问题……	422
移植到Android	424
你的Android应用就是一堆对话框	425
集成应用……	429
应用大功告成!	431
你的Python工具箱	432

File Edit View Insert Format Form Tools Help										
Formula: V02										
	A	B	C	D	E	F	G	H	I	
1	V02	84.8	82.9	81.1	79.3	77.5	75.8	74.2	72.5	
2	2mi	8:00	8:10	8:21	8:33	8:44	8:56	9:08	9:20	
3	5k	12:49	13:06	13:24	13:42	14:00	14:19	14:38	14:58	
4	5mi	21:19	21:48	22:17	22:47	23:18	23:50	24:22	24:56	
5	10k	26:54	27:30	28:08	28:45	29:24	30:04	30:45	31:26	
6	15k	41:31	42:27	43:24	44:23	45:23	46:24	47:27	48:31	
7	10mi	44:46	45:46	46:48	47:51	48:56	50:02	51:09	52:18	
8	20k	56:29	57:45	59:03	1:00:23	1:01:45	1:03:08	1:04:33	1:06:00	
9	13.1mi	59:49	1:01:09	1:02:32	1:03:56	1:05:23	1:06:51	1:08:21	1:09:53	
10	25k	1:11:43	1:13:20	1:14:59	1:16:40	1:18:24	1:20:10	1:21:58	1:23:49	
11	30k	1:27:10	1:19:08	1:31:08	1:33:11	1:35:17	1:37:26	1:39:37	1:41:52	
12	Marathon	2:05:34	2:08:24	2:11:17	2:14:15	2:17:16	2:20:21	2:23:31	2:26:44	

其他

(我们没有谈到的) 十大问题

你已经学到了不少。

不过学习Python是永无止境的。你编写的Python代码越多，就越需要了解新的方法来处理某些事情。你还要掌握新工具和新技术。这本书实在篇幅有限，无法向你全面展示关于Python所需了解的一切。所以，下面给出我们没有谈到的十大问题，你可能希望下一步对这些问题有更多了解。

#1: 使用一个“专业” IDE	436
#2: 处理作用域	437
#3: 测试	438
#4: 高级语言特性	439
#5: 正则表达式	440
#6: 关于Web框架	441
#7: 对象关系映射工具和NoSQL	442
#8: GUI编程	443
#9: 要避免的问题	444
#10: 其他Python书	445



# 1 初识Python

## 人人都爱列表



你可能会问：“Python有什么与众不同的地方？”

如果简短地回答这个问题，那么答案是：很多很多。要想更详细地回答，首先需要指出Python也有很多我们熟悉的东西。类似于所有其他通用编程语言，Python同样有语句、表达式、操作符、函数、模块、方法和类。这些确实都很普通。不过，Python还能提供一样东西，能让程序员（也就是你）的日子更好过一些。下面先从学习列表开始我们的Python之旅。不过，在此之前，还有一个重要的问题需要回答……

## Python有什么过人之处？

很多很多。可以这么说，这本书的目标就是向你展示Python的妙处。



深入研究Python之前，先来完成一些日常工作。

要使用和执行本书中的Python代码，你的计算机上需要有一个Python 3解释器。就像Python的很多其他方面一样，安装这个解释器并不难。当然，这里假设原先尚未安装Python……

## 安装Python 3

编写和运行Python代码之前，需要确保你的计算机上安装有Python解释器。在本书中，我们将采用Python 3，这是这个语言的新（也是最好）版本。

你的计算机上可能已经安装了某个版本的Python。Mac OS X会预装Python 2，Linux的大多数版本也是如此（也可能预装Python 3）。但Windows有所不同，它未内置任何Python版本。下面来检查你的计算机上是否安装有Python 3。打开一个命令行提示窗口，如果你使用的操作系统是Mac OS X或Linux，可以键入：

**python3 -V**

顺便说一句，这是一个大写的“V”。

在Windows上，则使用下面这个命令：

**c:\Python31\python.exe -V**

如果使用大写的“V”，会在屏幕上显示Python版本。

如果没有大写的“V”，则会进入Python解释器。

使用quit()命令会退出解释器，返回到操作系统提示符。

```
File Edit Window Help WhichPython?
$ python3 -V
Python 3.1.2
$
$ python3
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more info.
>>>
>>> quit()
$
```

**动手做!**

如果你的计算机上没有安装Python 3，可以从[www.python.org](http://www.python.org)网站针对你喜欢的操作系统下载一个合适的Python版本。

安装Python 3时，还会得到一个IDLE，这是Python的集成开发环境，尽管简单，但极其有用。IDLE包括一个能够利用颜色突出显示语法的编辑器、一个调试工具、Python Shell，以及一个完整的Python 3在线文档集。

下面先来简单看看IDLE。



## 使用IDLE来帮助学习Python

IDLE提供了一个功能完备的代码编辑器，允许你在这个编辑器中编写代码，另外还有一个 Python Shell，可以在其中试验运行代码。本书后面会使用这个代码编辑器，不过要知道，学习Python时，IDLE的shell真的很棒，因为利用它你可以在编写新Python代码的同时尝试运行。

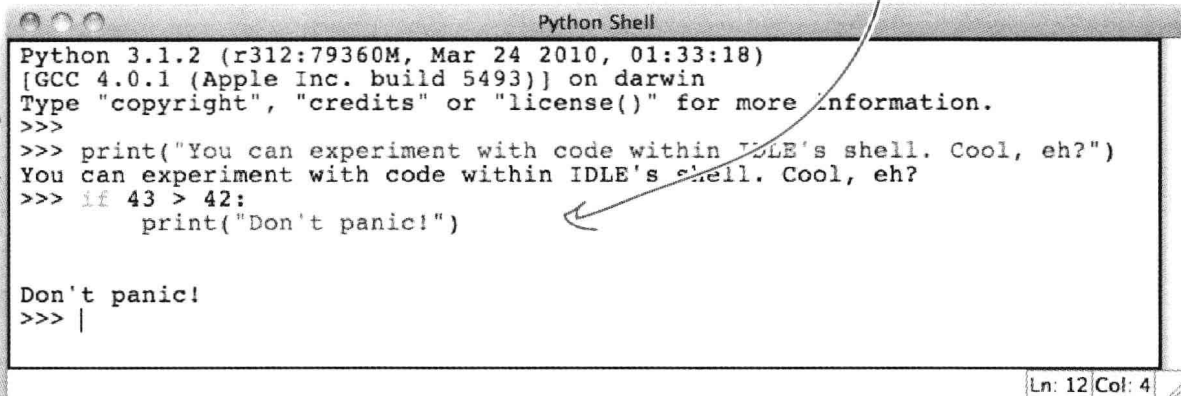
第一次启动IDLE时，会显示“三个大于号”提示符(>>>)，可以在这里输入代码。shell得到你的代码语句后会立即执行，并在屏幕上显示生成的结果。

IDLE知道所有Python语法，它会提供“完成提示”（当你使用一个内置函数（如print()）时就会弹出这种“完成提示”）。Python程序员通常把内置函数称为BIF（built-in functions）。print() BIF的作用是把消息显示到标准输出（通常就是屏幕）。

与其他基于C的语言不同，它们往往使用{和}来界定代码块，Python则利用缩进指示代码块。

在>>>提示符后面  
输入代码。

可以立即  
看到结果。



```
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>> print("You can experiment with code within IDLE's shell. Cool, eh?")
You can experiment with code within IDLE's shell. Cool, eh?
>>> if 43 > 42:
    print("Don't panic!")

Don't panic!
>>> |
```

IDLE使用区分颜色的语法来突出显示代码。默认地，内置函数都是紫色，字符串是绿色，Python语言的关键字（如if）为橙色。生成的所有结果显示为蓝色。如果你不喜欢这些颜色选择，也不用担心。只需调整IDLE的首选项就可以很容易地改变颜色选择。

另外，IDLE也很清楚Python的缩进语法（Python要求代码块缩进）。刚开始使用Python时，你可能很难适应这一点，不过IDLE可以减轻你的负担，它会根据需要自动缩进。

**IDLE了解  
Python的语法，  
并会帮助你遵  
循Python的缩  
进规则。**

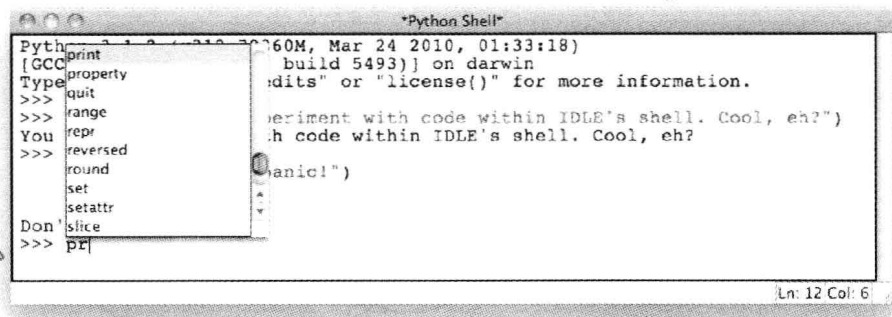
## 有效地使用IDLE

IDLE提供了大量特性，不过只需了解其中一小部分就能很好地使用IDLE。

### TAB完成

先键入一些代码，然后按下TAB键。IDLE会提供一些建议，帮助你完成这个语句。

在>>>提示符后面键入“pr”，然后按下TAB，可以看到IDLE给出的一个列表，其中包括完成这个命令的一组建议。



这是我的机器上的IDLE界面。看起来可能会与你的IDLE界面稍有不同，不过差别不会太大（没错，确实这么丑）。

### 回退代码语句

按下Alt-P，可以回退到IDLE中之前输入的代码语句，或者按下Alt-N可以移至下一个代码语句（如果有的话）。可以利用这两个按钮组合在IDLE中已输入的所有代码之间快速转换，根据需要重新执行其中的任何代码语句。

Alt-P表示“前一个”（Previous）

Alt-N表示“下一个”（Next）

除非你在使用Mac，这种情况下则使用Ctrl-P和Ctrl-N。

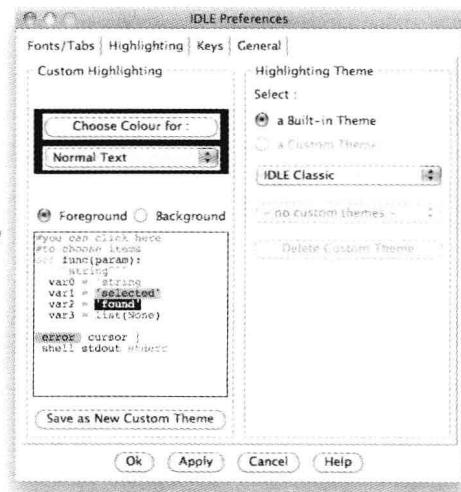
### 编辑回退的代码

一旦回退代码语句，还可以进行编辑，并使用箭头键切换语句。可以编辑之前输入的任何语句，甚至是跨多行的代码语句。

### 调整IDLE的首选项

IDLE的首选项对话框允许你根据个人口味调整IDLE的默认行为。有4个设置标签页可以调整。可以控制字体和tab行为，改变突出显示语法所用的颜色，调整某些按钮组合的行为，还可以改变IDLE的启动设置。所以，如果你确实喜欢鲜艳的粉红色字符串，IDLE会赋予你这个能力，可以改变代码在屏幕上如何显示。

按你心中所想来调整IDLE。

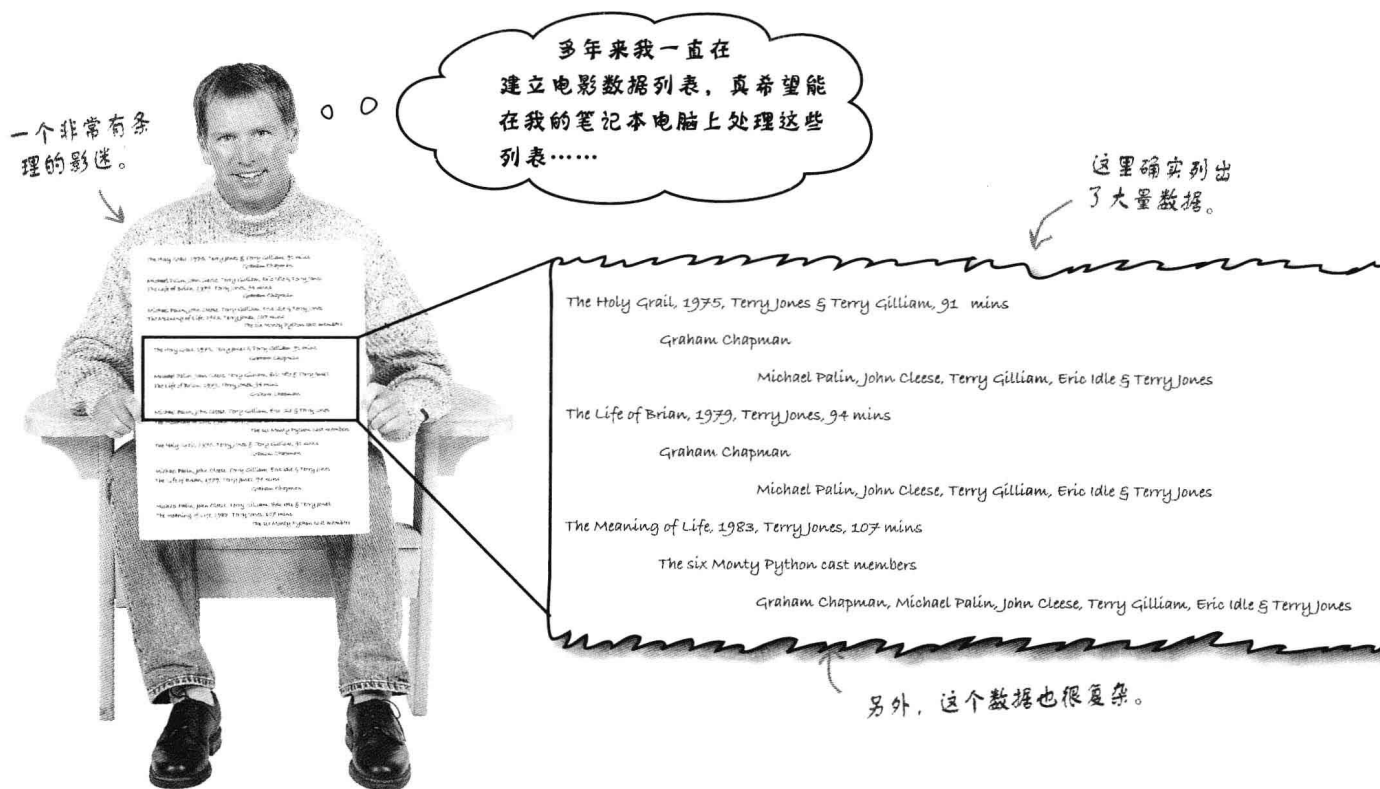


## 处理复杂数据

任何值得着手创建的程序都必然会处理数据。有些时候，数据很简单也很直接——很容易处理。但还有些情况下，所要处理的数据不论从结构还是含义上讲都很复杂，这就要求你花些工夫把它完全搞清楚，另外编写代码来处理这些数据更是需要费些心思。

为了降低复杂性，通常可以把数据组织为列表：可以有顾客列表、朋友列表、购物清单和待办事项清单等。由于用列表组织数据如此常见，Python中可以很容易地利用代码创建和处理列表。

在学习如何用Python创建和处理列表数据之前，先来看一些复杂的数据。



乍一看，这组数据确实相当复杂。不过，这组数据似乎符合某种结构：最前面一行给出一组电影基本信息，接下来一行是主要演员，再后面第三行列出了这部电影的配角。

看起来这个结构是可以处理的.....

# 创建简单的Python列表

先从下面这个简单的电影片名列表入手，在此基础上逐步深入：

The Holy Grail  
The Life of Brian  
The Meaning of Life

一个简短的Monty  
Python电影列表。

下面是同一个列表，不过这一次用Python能理解的方式来写：

```
movies = ["The Holy Grail",  
          "The Life of Brian",  
          "The Meaning of Life"]
```

为了把人可读的列表转换为Python可读的列表，需要遵循以下4个步骤：

- ❶ 在数据两边加引号，将各个电影名转换为字符串。
- ❷ 用逗号将列表项与下一项分隔开。
- ❸ 在列表的两边加上开始和结束中括号。
- ❹ 使用赋值操作符(=)将这个列表赋至一个标识符（以上代码中的movies）。

完全可以把创建列表的代码都放在一行上（当然，前提是假设你有足够的空间，全部代码都能在一行上放下）：

```
movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
```

这样也是可以的。



等一等！你是不是忘了点什么？难道不需要为列表声明类型信息吗？

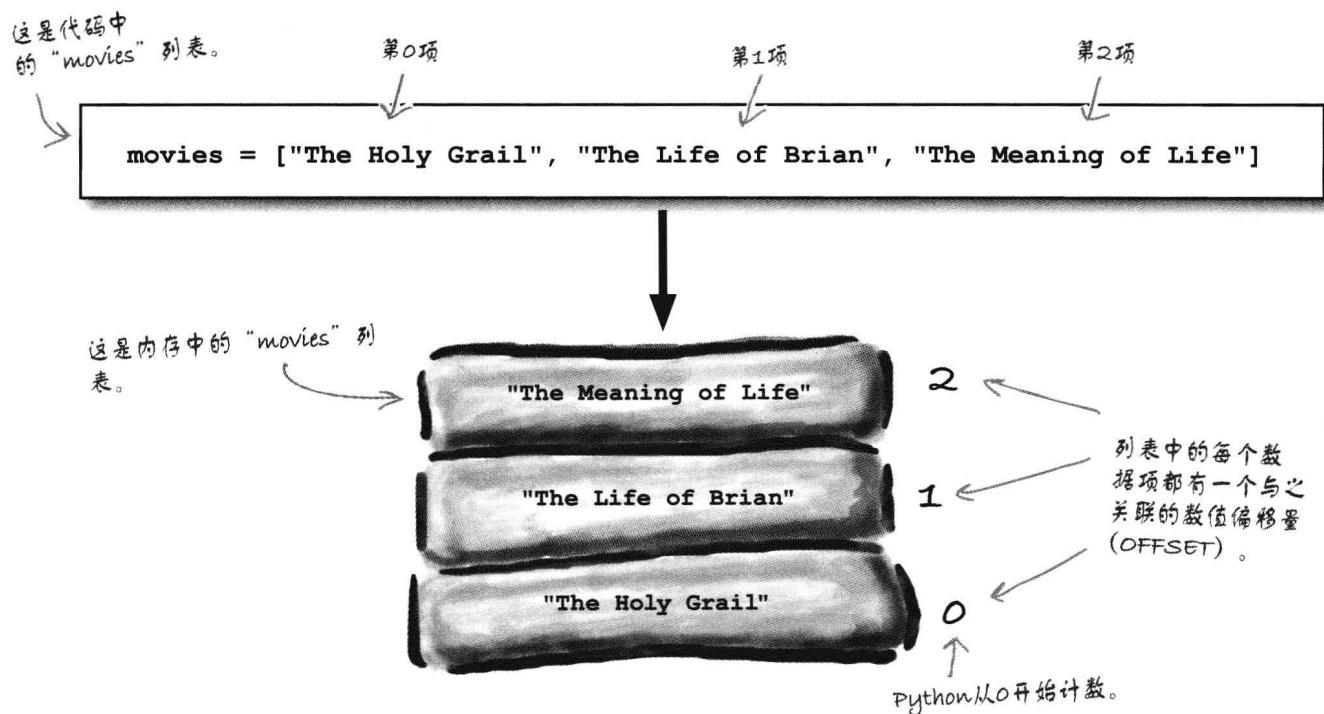
大可不必，因为Python的变量标识符没有类型。

很多其他编程语言坚持要求代码中使用的每一个标识符都必须声明有类型信息。但是这对Python并不适用：标识符只是名字，可以指示某个类型的数据对象。

可以把Python的列表想成是一个高层集合。对于列表来说，数据项的类型并不重要。当然可以说你的电影列表是一个“字符串集合”，不过Python并不需要知道这一点。Python所要知道的只是你需要一个列表，而且已经为它指定了一个名字，另外这个列表中包含有一些数据项。

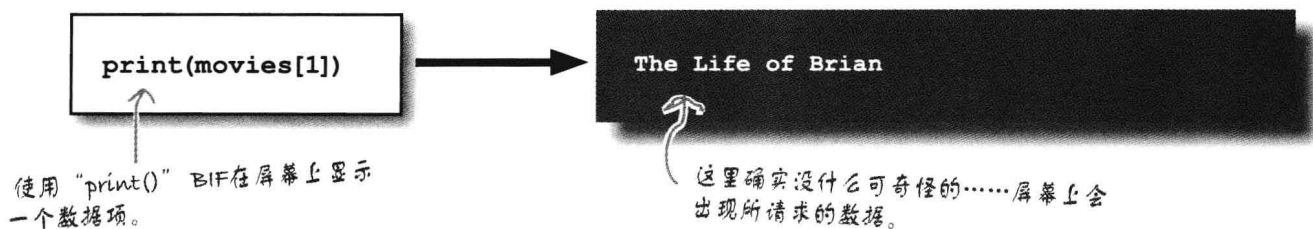
## 列表就像是数组

在Python中创建一个列表时，解释器会在内存中创建一个类似数组的数据结构来存储数据，数据项自下而上堆放（形成一个堆栈）。类似于其他编程语言中的数组技术，堆栈中的第一个槽编号为0，第二个槽编号为1，第3个编号为2，依此类推：



## 使用中括号记法访问列表数据

像数组一样，可以使用标准的中括号偏移量记法来访问一个列表槽中的数据项：



下面利用IDLE进一步学习列表是如何工作的。



## An IDLE Session

Python中的列表看起来可能很像数组，不过还不只如此：列表是完备的Python集合对象。也就是说，列表通过列表方法的形式提供了一些现成的功能。

下面来了解Python列表的一些方法。打开IDLE，在>>>提示符后面输入如下代码。应该能看到与这里相同的输出。

首先定义一个名字列表，然后使用print() BIF在屏幕上显示这个列表。接下来，使用len() BIF得出列表中有多少个数据项，然后再访问并显示第2个数据项的值：

```
>>> cast = ["Cleese", 'Palin', 'Jones', "Idle"]
>>> print(cast)
['Cleese', 'Palin', 'Jones', 'Idle']
>>> print(len(cast))
4
>>> print(cast[1])
Palin
```

← 对一个BIF的结果再调用另一个BIF是完全可以的。

创建了列表之后，可以使用列表方法在列表末尾增加一个数据项（使用append()方法），或者从列表末尾删除数据（使用pop()方法），还可以在列表末尾增加一个数据项集合（利用extend()方法）：

```
>>> cast.append("Gilliam")
>>> print(cast)
['Cleese', 'Palin', 'Jones', 'Idle', 'Gilliam']
>>> cast.pop()
'Gilliam'
>>> print(cast)
['Cleese', 'Palin', 'Jones', 'Idle']
>>> cast.extend(["Gilliam", "Chapman"])
>>> print(cast)
['Cleese', 'Palin', 'Jones', 'Idle', 'Gilliam', 'Chapman']
```

← 使用常用的 "."（点记法）来调用方法。

← 这是另一个列表：各数据项之间用逗号隔开，整个列表用中括号括起。

最后，在列表中找到并删除一个特定的数据项（使用remove()方法），然后在某个特定的位置前面增加一个数据项（使用insert()方法）：

```
>>> cast.remove("Chapman")
>>> print(cast)
['Cleese', 'Palin', 'Jones', 'Idle', 'Gilliam']
>>> cast.insert(0, "Chapman")
>>> print(cast)
['Chapman', 'Cleese', 'Palin', 'Jones', 'Idle', 'Gilliam']
```

← 完成所有这些调用之后，最终会得到Monty Python影片Flying Circus的演员表！

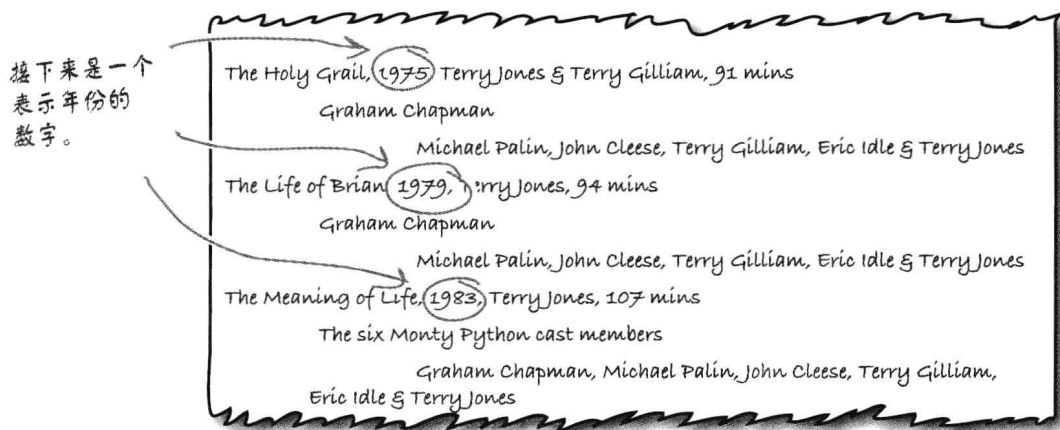


## 向列表增加更多数据

创建了电影片名列表之后，现在需要为它增加这个影迷的更多复杂数据。  
这里可以做一个选择：



这两种策略都是可行的。至于哪一种方法更合适，这取决于你打算做什么。下面回忆一下这个影迷的数据是怎样的：



向列表增加的下一个数据是一个数字（表示影片发行的年份），它必须插入到各个影片片名后面。下面就来做这个工作，看看会发生什么。



什么？在列表中混合不同的类型？列表中应该不能混合不同类型的数据，难道不是吗？这是不是太荒谬了？

不，一点也不荒谬，Python正是这样做的。

Python列表可以包含混合类型的数据。在同一个Python列表中混合存放字符串和数字是允许的。实际上，不光可以混合字符串和数字，只要你愿意，完全可以在列表中存储任意类型的数据。

应该还记得，Python列表是一个高层集合，原本设计为要存储一个“相关事物”的集合。列表并不关心这些事物的类型是什么，因为列表的存在只是为了提供一种机制，从而可以采用列表形式存储数据。

所以，如果确实需要在列表中存储混合类型的数据，Python绝对不会阻止你。



下面花点时间试试看，在这种情况下向列表增加数据应当采用哪种策略。  
给定以下创建列表的代码：

```
movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
```

❶ 得出向以上列表插入年份数值数据所需的Python代码，改变列表，使之最终为以下形式：

```
["The Holy Grail", 1975, "The Life of Brian", 1979, "The Meaning of Life", 1983]
```

在这里写出你的  
插入代码。 →

.....

.....

.....

.....

.....

.....

.....

❷ 现在用所需的数据从头重新创建列表，请写出重新创建列表所需的代码：

在这里写出重  
新创建列表的  
代码。 →

.....

.....

.....

对于这种情况，你认为以上两种方法中哪一个更好（请把你的答案圈出来）？

❶      或      ❷



## Exercise Solution

下面花点时间试试看，在这种情况下向列表增加数据应当采用哪种策略。

给定以下创建列表的代码：

```
movies = ["The Holy Grail", "The Life of Brian", "The Meaning of Life"]
```

❶ 得出向以上列表插入年份数值数据所需的Python代码：

将第1个年份插入到第2个列表项前面。

```
movies.insert(1, 1975)
```

将第2个年份插入到第4个列表项前面。

```
movies.insert(3, 1979)
```

你算对了吗？第一次插入之后，列表扩大了，所以在计算第二次插入的位置时需要考虑这一点。

```
movies.append(1983)
```

然后把最后一个年份追加到列表末尾。

❷ 还要写出用所需数据从头重新创建列表的相应Python代码：

```
movies = ["The Holy Grail", 1975,
```

```
"The Life of Brian", 1979,
```

```
"The Meaning of Life", 1983]
```

将所有数据赋至“movies”标识符。这会取代原来存储的数据。

对于这种情况，你认为以上两种方法中哪一个更好？（圈出你的答案）。

❶

或

❷

没错，在这里第2种方法看起来更好一些……我的意思是，对于这样一个小列表来说，第2种方法更可取。而且，第2种方法不需要做麻烦的计算。

## 处理列表数据

通常需要迭代处理列表，在这个过程中对每一个数据项完成某个动作。  
当然，可能经常这样做（这种做法可行，但是不具有伸缩性）：

定义一个列表，并用两个电影片名填充列表项。

在屏幕上显示各个列表项的值。

```
fav_movies = ["The Holy Grail", "The Life of Brian"]

print(fav_movies[0])
print(fav_movies[1])
```

这是处理列表的代码。

这个代码会得到我们预期的结果，使列表的数据都出现在屏幕上。不过，如果以后修改代码，在列表中再增加一个喜欢的电影，以上处理列表的代码就无法达到预期了，因为这个代码没有提到第3个数据项。

这有何难，只需要再增加一个print()语句，对不对？

没错，对于一个新增的电影，再加一个print()语句是可以的，但是，如果需要再加上百部喜欢的电影呢？这个问题的规模肯定会让你心生畏惧，因为另外增加所有这些print()语句实在太烦琐，你不会愿意这样做，肯定会想方设法逃避。

### 该迭代了

处理每一个列表项是一个相当常见的需求，所以Python通过提供内置的for循环可以非常方便地做到这一点。考虑以下代码，这里重写了前面的代码来使用一个for循环：

定义一个列表，并像前面一样填充这个列表。

使用“for”迭代处理列表，在这个过程中将各个列表项的值显示在屏幕上。

```
fav_movies = ["The Holy Grail", "The Life of Brian"]

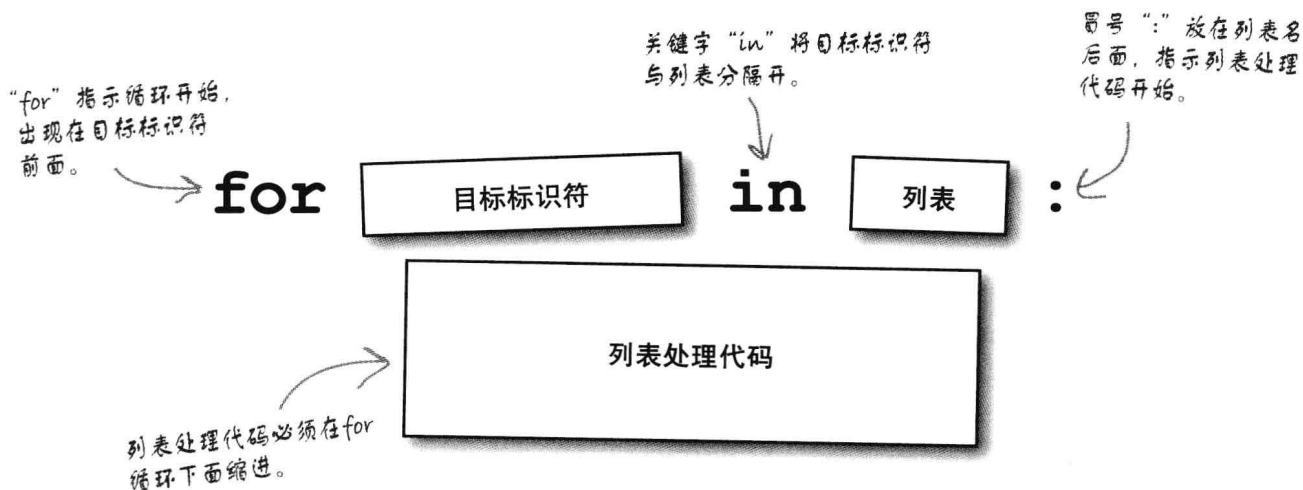
for each_flick in fav_movies:
    print(each_flick)
```

这是处理列表的代码，这里使用了一个for循环。

使用for循环是可伸缩的，适用于任意大小的列表。

## For循环处理任意大小的列表

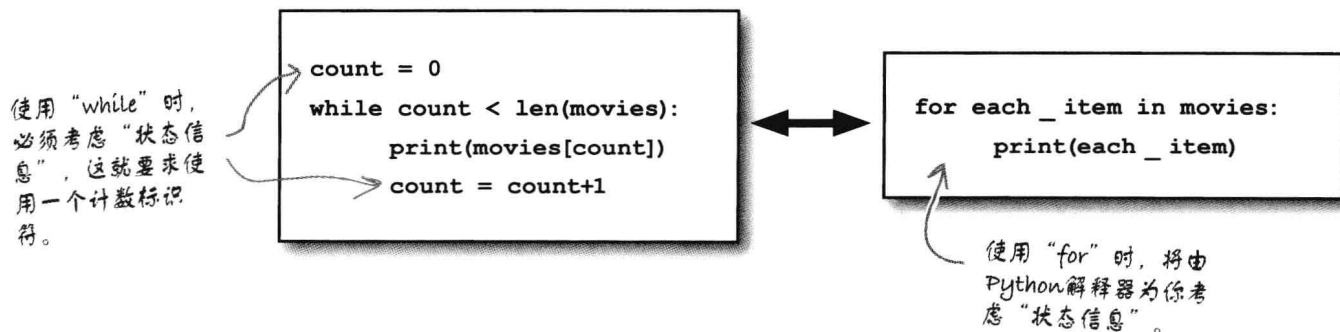
Python的for循环就是为了处理列表和Python中的其他迭代结构。列表是Python中最常用的迭代数据结构，需要迭代处理一个列表时，最好使用for循环：



列表处理代码被Python程序员称为“组”（suite）。

目标标识符（target identifier）类似于代码中的任何其他名。迭代处理列表时，相应的会把列表中的各个数据值分别赋至目标标识符。这说明，每次执行循环代码时，目标标识符都会指示一个不同的数据值。循环会一直迭代，直到处理完列表的所有数据（不论列表有多大或者多小）。

除了使用for，还有一种候选方法，可以使用while循环编写迭代代码。考虑以下两个Python代码段，它们都完成相同的动作：



以上while和for语句完成的工作是一样的。

## there are no Dumb Questions

**问：**这么说……迭代处理一个列表时，是不是总要用for而不是while？

**答：**对，除非你有非常充分的理由使用while循环（或者需要while循环提供的额外控制）。for循环会负责从列表起始位置开始，一直处理到列表末尾。如果使用for，几乎不太可能遭遇“大小差1”错误。但使用while循环时就不会这么走运了。

**问：**那么，列表并不真的像数组，因为列表能做的事情要多得多，是这样吗？

**答：**没错……你可以用标准的中括号记法访问列表中的单个数据项，从这一点来讲，列表和数组确实很像，不过你已经看到，Python的列表可以做更多的事情。在Head First Labs，我们喜欢把列表认为是“打了激素的数组”。

**问：**只有Python 3中有列表，是吗？

**答：**不对。尽管Python 3中确实对列表增加了一些改进，不过Python 2也有列表。到目前为止，你所学到的关于列表的内容对Python 3和Python 2中的列表都适用。

**问：**为什么我们要使用Python 3？Python 2到底有什么问题？看起来很多程序员都在使用Python 2。

**答：**确实有相当多的程序员在使用Python 2，不过Python 3才是Python发展的未来。当然，要让整个Python社区完全转向Python 3并不是一蹴而就的事情，所以在可预知的将来，仍会有大量项目继续使用Python 2。尽管Python 2目前占据主导，但在Head First Labs，我们认为Python 3中的新特性确实很妙，

很值得“追加投入”来深入学习。不用担心，如果你了解Python 2，Python 3也很容易掌握。

**问：**我看到Python的列表可以随需要伸缩，它们肯定不支持越界检查，对不对？

**答：**是这样的，列表可以伸缩，从这一点来讲，列表是动态的，不过它们并没有魔力，因为不能访问一个根本不存在的项。如果你试图访问一个不存在的数据项，Python会给出一个IndexError作为响应，这就表示“越界”。

**问：**那些奇怪的Monty Python引用到底是什么？

**答：**哈，你发现了？看起来Python创始人Guido van Rossum设计这个新的编程语言时肯定在看Monty Python电视剧的剧本。Guido需要为这个新语言找一个名字，他有点开玩笑地选择了“Python”（这也可能只是一个“谣传”）。

**问：**我是不是需要了解Monty Python才能理解这些例子？

**答：**大可不必，不过在官方的Python文档中指出：“如果你了解Monty Python会有帮助”。但不用担心，即使你从来没有听说过Monty Python，也一样能过得很好。

**问：**我注意到，有些字符串用双引号引起来，而另外一些却用单引号引起来。这有什么区别吗？

**答：**没有任何区别。Python中，单引号和双引号都可以用来创建字符串。对此只有一个规则，这就是如果字符

串前面使用了某个引号（单引号或双引号），那么字符串后面也要用同样的引号；不能在字符串前后混合使用不同的引号。你可能已经看到，IDLE在shell中显示字符串时使用了单引号。

**问：**如果我需要在一个字符串中嵌入一个双引号该怎么做呢？

**答：**你有两个选择：可以像这样对双引号转义：\'，或者用单引号引起这个字符串。

**问：**可以用任意字符来命名标识符吗？

**答：**不行。与大多数其他编程语言一样，创建名字时，Python也有一些必须遵循的规则。名字可以以一个字母字符或一个下划线开头，接下来可以包括任意个字母字符、数字和/或下划线。不允许有奇怪的字符（如%\$£），你肯定希望使用在代码上下文中有意义的名字。类似members、the\_time和people等名字就比m、t和p好得多，不是吗？

**问：**确实，好的命名实践通常很重要。那么大小写敏感性呢？

**答：**嗯，Python属于“敏感型”，因为Python代码区分大小写。这说明，msg和MSG是两个不同的名字，所以一定要当心。Python（和IDLE）会帮助解决可能因此出现的问题。例如，只有当标识符已经赋值后才能在代码中使用；未赋值的标识符会导致运行时错误。这说明，如果本来想键入msg，但实际上误写作mgs，你会很快发现问题，因为Python会指出代码存在一个NameError错误。

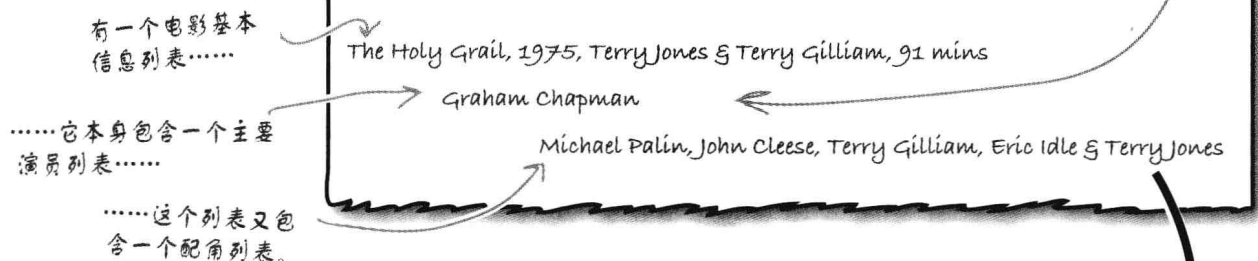


## 在列表中存储列表

我们已经看到，列表可以存放混合类型的数据。不过还有更棒的：列表能存放任何东西的集合，这也包括其他列表。只需根据需把内列表嵌在外围列表中。

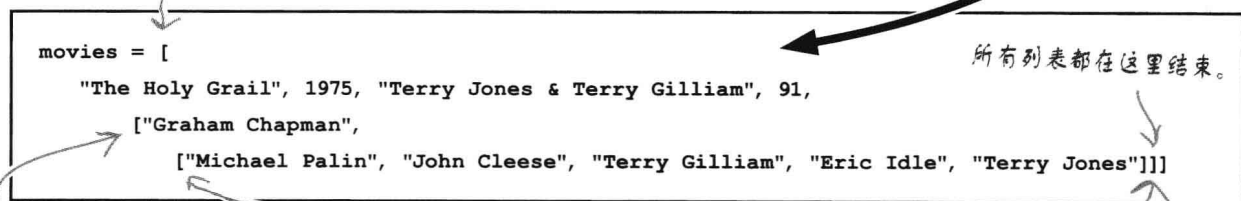
进一步分析影迷的数据，可以确定一个结构，看起来这非常像一个列表的列表：

这里只列出了一个主要演员，不过也可以有更多。



在Python中，稍微努力甚至毫不费力就可以把这个真实的数据列表转换为代码。只需记住，每个列表都是一个列表项集合，各列表项相互之间用逗号隔开，另外列表要用中括号括起。当然，任何列表项本身也可以是另一个列表：

第一个外列表的起始位置。

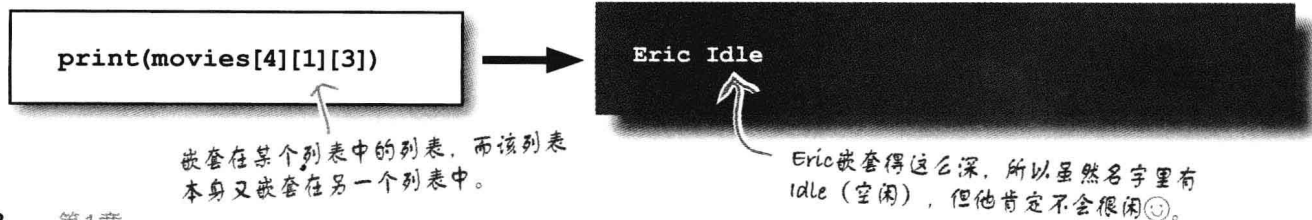


第二个内列表“movies[4]”的起始位置。

第3个下一级内列表“movies[4][1]”的起始位置。

看起来有点怪异……不过只要记住前面有3个开始中括号，所以肯定还有3个结束中括号，这样就不奇怪了。

所以，列表中嵌套列表是可以的，另外列表中嵌套的列表还可以嵌套下一级列表（前面的代码示例就是如此）。实际上，利用Python，完全可以在列表中嵌套任意多层的列表。可以使用每个列表自己的列表方法来管理，并使用中括号记法来访问：





## An IDLE Session

创建一个包含另一个列表的列表，这个工作很简单。不过，如果你想使用本章前面的for循环来处理一个包含另一个列表（或多个列表）的列表，会怎么样呢？

下面使用IDLE来看会发生什么。首先在内存中为“The Holy Grail”创建电影数据列表，并在屏幕上显示，然后用for循环处理这个列表：

```
>>> movies = ["The Holy Grail", 1975, "Terry Jones & Terry Gilliam", 91,
               ["Graham Chapman", ["Michael Palin", "John Cleese",
                                     "Terry Gilliam", "Eric Idle", "Terry Jones"]]]

>>> print(movies)

['The Holy Grail', 1975, 'Terry Jones & Terry Gilliam', 91, ['Graham Chapman', ['Michael Palin',
'John Cleese', 'Terry Gilliam', 'Eric Idle', 'Terry Jones']]]

>>> for each_item in movies:
    print(each_item)
```

The Holy Grail  
1975  
Terry Jones & Terry Gilliam  
91  
['Graham Chapman', ['Michael Palin', 'John Cleese', 'Terry Gilliam', 'Eric Idle', 'Terry Jones']]

已经在内存中创建这个三重嵌套的列表。

“for”循环只打印外列表的各个数据项。

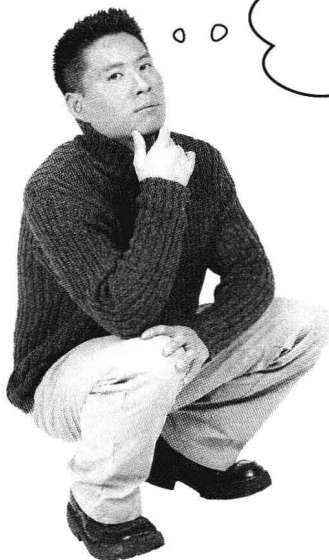
嵌套在内列表中的下一层内列表会原样打印。

你的for循环本身并没有问题。我想麻烦在于你没有告诉它如何处理找到的内列表，从而能打印所有内容，是不是？

没错，你说对了：这个循环代码并不完备。

目前，循环中的代码只是打印各个列表项，它在某个槽中找到一个列表时，只会在屏幕上把这个列表整个显示出来。毕竟，对于外围列表来说，内列表只是外列表中的一个列表项。这里我们需要一种机制来发现列表中的某一项实际上是另一个列表，并采取适当的行动。

听起来有点困难。不过Python能帮忙吗？



---

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

**备用QQ:2404062482**