关于 Python

Python 是一种极少数能声言兼具 *简单与 功能强大* 的编程语言。你将惊异于发现你正在使用的这门编程语言是如此简单，它专注于如何解决问题，而非拘泥于语法与结构。

官方对 Python 的介绍如下：

> Python 是一款易于学习且功能强大的编程语言。 它具有高效率的数据结构，能够简单又有效地实现面向对象编程。Python 简洁的语法与动态输入之特性，加之其解释性语言的本质，使得它成为一种在多种领域与绝大多数平台都能进行脚本编写与应用快速开发工作的理想语言。

我将会在下一节详细讨论这些特性。

名字背后的故事

Python 的创造者吉多·范罗苏姆（Guido van Rossum）采用 BBC 电视节目《蒙提·派森的飞行马戏团（Monty Python's Flying Circus，一译巨蟒剧团）》的名字来为这门编程语言命名。尽管他本人并不特别喜欢蟒蛇这种通过在猎物身边卷曲自己的身体以此来碾碎猎物身体来进食的动物。

Python 的特色

简单

Python 是一门简单且简约的语言。阅读一份优秀的 Python 程序代码就如同在阅读英语文章一样，尽管这门英语要求十分严格！Python 这种伪代码式的特质正是它的一大优势。它能够让你专注于解决问题的方案，而不是语言本身。

易于学习

正如你接下来将看到的，Python 是一门非常容易入门的语言。正如前面所提到的，Python 有一套极其简单的语法体系。

自由且开放

Python 是 FLOSS （自由/开放源代码软件）的成员之一。简单来说，你可以自由地分发这一软件的拷贝，阅读它的源代码，并对其作出改动，或是将其的一部分运用于一款新的自由程序中。FLOSS 基于一个可以分享知识的社区理念而创建。这正是 Python 为何能如此优秀的一大原因——它由一群希望看到 Python 能变得更好的社区成员所创造，并持续改进至今。

高级语言

当你在用 Python 编写程序时，你不必考虑诸如你的程序应当如何使用内存等底层细节。

跨平台性

由于其开放源码的特性，Python 已被移植到其它诸多平台（意即它们已经过改动以保证其能正常工作）。如果你小心地避开了所有系统依赖型的特性。你所有的 Python 程序可以在其中任何一个平台上工作，不必作出任何改动。

你可以在 GNU/Linux、Windows、FreeBSD、Macintosh、 Solaris、 OS/2、 Amiga、 AROS、 AS/400、 BeOS、 OS/390、 z/OS、 Palm OS、 QNX、 VMS、 Psion、 Acorn RISC OS、 VxWorks、 PlayStation、 Sharp Zaurus、 Windows CE 以及 PocketPC 平台上运行 Python！

你甚至可以通过诸如 Kivy 一类的平台来制作可在你的电脑 *以及* iPhone、iPad 或安卓手机上运行的游戏。

解释性

有关这一特性，需要一些详细的解释。

在你使用诸如 C 或 C++ 等编译语言编写程序时，需要将这些语言的源代码通过编译程序配合其中不同的标记（Flags）与选项，来将它们转换成你的电脑所使用的语言（例如 0 与 1 构成的二进制码）。当你运行这些程序时，链接程序或载入程序将会从硬盘中将程序拷贝至内存中并将其运行。

另一方面，Python 不需要将其编译成二进制码。你只需要直接从源代码 *运行* 该程序。在程序内部，Python 会将源代码转换为称为字节码的中间形式，尔后再转换成你的电脑所使用的语言，并运行它。实际上，这一流程使得 Python 更加易于使用，你不必再担心该如何编译程序，或如何保证适当的库被正确的链接并加载等等步骤。这也同样使得 Python 程序更便携且易于迁移，你只需要将 Python 程序拷贝到另一台电脑便可让它立即开始工作！

面向对象

Python 同时支持面向过程编程与面向对象编程。在 *面向过程* 的编程语言中，程序是由仅仅带有可重用特性的子程序与函数所构建起来的。在 *面向对象* 的编程语言中，程序是由结合了数据与功能的对象所构建起来的。与 C++ 或 Java 这些大型语言相比，Python 具有其特别的、功能强大又简单的方式来实现面向对象编程。

可扩展性

如果你需要代码的某一重要部分能够快速地运行，或希望算法的某些部分不被公开，你可以在 C 或 C++ 语言中编写这些程序，然后再将其运用于你的 Python 程序中。

可嵌入性

你可以在你的 C 或 C++ 程序中嵌入 Python，从而向你的程序用户提供 *脚本* 功能。

丰富的库

实际上 Python 标准库的规模非常庞大。它能够帮助你完成诸多事情，包括正则表达式、文档生成、单元测试、多线程、数据库、网页浏览器、CGI、FTP、邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI（图形用户界面），以及其它系统依赖型的活动。只需记住，只要安装了 Python，这些功能便随时可用。它们的存在被称作 Python *自备电池（Batteries Included）* 式的哲学。

除了标准库以外，你还可以在 Python 库索引（Python Package Index） 中发掘许多其它高质量的库。

总结

Python 着实是一门令人心生激动且强大的语言。它得当地结合了性能与功能，使得编写 Python 程序是如此简易又充满乐趣。

Python 3 VS Python 2

如果你对“Python 2”与“Python 3”之间的区别不感兴趣你可以略过本段。但务必注意你正在使用的版本。本书是以 Python 3 为对象撰写的。

只消记住一旦你正确理解并学习了其中一个版本的 Python，你便可以很容易地理解另一版本的区别，并能快速学习如何使用。困难的是学习如何编程以及理解 Python 语言本身的基础部分。这便是我们在本书中的目标，而一旦你达成了目标，你便可以根据你的实际情况，决定是该使用 Python 2 还是 Python 3。

要想了解有关 Python 2 和 Python 3 之间的区别的更多细节，你可以参阅：

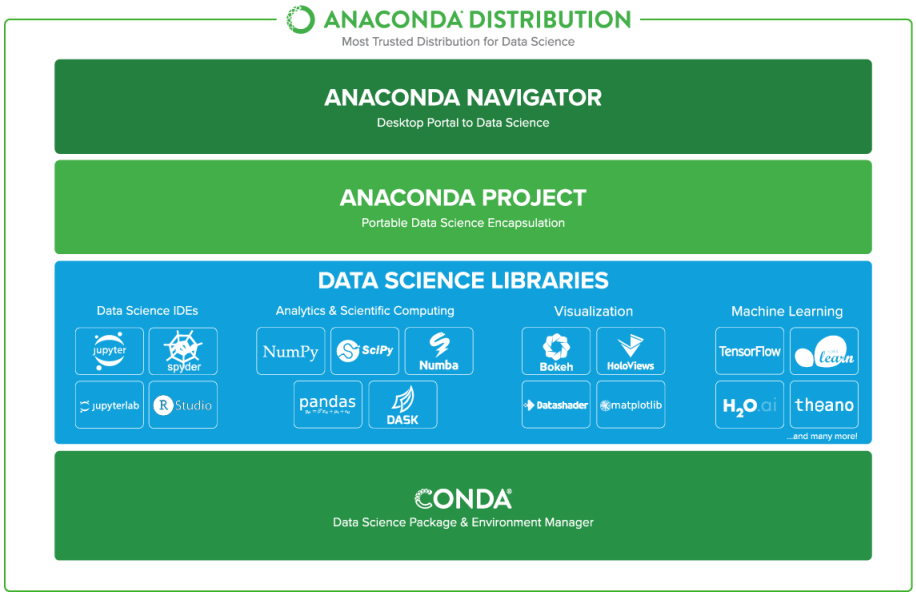
- The future of Python 2
- Porting Python 2 Code to Python 3
- Writing code that runs under both Python2 and 3
- Supporting Python 3: An in-depth guide

程序员怎么说

或许你在阅读诸如 ESR 等伟大的黑客是如何讨论 Python 时会有些有趣的发现：

- *埃里克·雷蒙（Eric S. Raymond）* 是《大教堂和市集（The Cathedral and the Bazaar）》的作者，同时也是 *开放源代码促进会* 的创始人之一。他曾说Python 已成为他所喜爱的一门编程语言。这篇文章给了我接触 Python 的最先鼓舞。
- *布鲁斯·埃克尔（Bruce Eckel）* 是《Java 编程思想（Thinking in Java）》与《C++ 编程思想（Thinking in C++）》的作者。他说没有一种编程语言能像 Python 这样使他更加高产。他说或许 Python 是唯一一门面向程序员且致力于使事情变得更加容易的语言。阅读 完整采访 以了解更多细节。
- *彼得·诺米格（Peter Norvig）* 是广为人知的 Lisp 作者，同时也是 Google 公司的搜索质量总监（Director of Search Quality，感谢吉多·范罗苏姆指出这一点）。他说写 Python 时就好像在写伪代码。他还说 Python 一直是构成 Google 整体的重要部分。你可以通过浏览 Google Jobs 页面并发现“Python 知识”是软件工程师所须具备的一项要求来验证这一说法。

关于本书所用的Anaconda Python发行版



如上图所示，开源的Anaconda Python有机的整合了科学计算领域中常用的Python库，使得Anaconda Python成为了目前在Linux,Windows和Mac上进行Python科学计算和Python机器学习最方便的平台，能够方便的安装和管理各种Python库和Python环境。在本书中，我们会详细介绍相关用法。