图书在版编目（CIP）数据

Python数据分析基础教程 : NumPy学习指南 : 第2版/（印尼）伊德里斯（Idris,I.）著 ; 张驭宇译. -- 北京 : 人民邮电出版社, 2014.1
（图灵程序设计丛书）
书名原文: NumPy beginner’s guide,second edition
ISBN 978-7-115-33940-9

Ⅰ. ①P… Ⅱ. ①伊… ②张… Ⅲ. ①软件工具－程序设计－教材 Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2013)第296079号

内 容 提 要

本书是 NumPy 的入门教程，主要介绍 NumPy 以及相关的 Python 科学计算库，如 SciPy 和 Matplotlib。本书内容涵盖 NumPy 安装、数组对象、常用函数、矩阵运算、线性代数、金融函数、窗函数、质量控制、Matplotlib 绘图、SciPy 简介以及 Pygame 等内容，涉及面较广。另外，Ivan Idris 针对每个知识点给出了简短而明晰的示例，并为大部分示例给出了实用场景（如股票数据分析），在帮助初学者入门的同时，提高了本书可读性。

本书适合正在找寻高质量开源计算库的科学家、工程师、程序员和定量管理分析师阅读参考。

◆ 著　　[印尼] Ivan Idris
译　　张驭宇
责任编辑　毛倩倩
执行编辑　程　芃
责任印制　焦志炜

◆ 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　　印刷

◆ 开本：800×1000　1/16
印张：15.25
字数：371千字　　2014年 1 月第 1 版
印数：1－4 000册　　2014年 1 月北京第 1 次印刷
著作权合同登记号　图字：01-2013-5239号

定价：49.00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：京崇工商广字第 0021 号

版 权 声 明

译 者 序

NumPy，即Numeric Python的缩写，是一个优秀的开源科学计算库，并已经成为Python科学计算生态系统的重要组成部分。NumPy为我们提供了丰富的数学函数、强大的多维数组对象以及优异的运算性能。尽管Python作为流行的编程语言非常灵活易用，但它本身并非为科学计算量身定做，在开发效率和执行效率上均不适合直接用于数据分析，尤其是大数据的分析和处理。幸运的是，NumPy为Python插上了翅膀，在保留Python语言优势的同时大大增强了科学计算和数据处理的能力。更重要的是，NumPy与SciPy、Matplotlib、SciKits等其他众多Python科学计算库很好地结合在一起，共同构建了一个完整的科学计算生态系统。毫不夸张地讲，NumPy是使用Python进行数据分析的一个必备工具。

说起NumPy，我是在数据的“赛场”上与之结缘的。出于对数据挖掘和机器学习的爱好，我与中国科学院和北京大学的同学一起组建了一支名为BrickMover的“比赛小分队”，队友包括庞亮、石磊、刘旭东、黎强、孙晗晓、刘璐等。我们先后参加了一些国内外的数据挖掘比赛，包括百度电影推荐系统算法创新大赛、首届中国计算广告学大赛暨RTB算法大赛、RecSys Challenge 2013、ICDM Contest 2013等，并且取得了还算不错的成绩。无一例外的是，这些比赛均需要对数据进行快速、全面的分析。感谢NumPy，它正是我们使用的数据分析利器之一。

本书作为NumPy的入门教程，从安装NumPy讲起，涵盖NumPy数组对象、常用函数、矩阵运算、线性代数、金融函数、窗函数、质量控制、Matplotlib绘图、SciPy简介以及Pygame等内容，涉及面较为广泛。书中对每个知识点均给出了简短而明晰的示例，很适合初学者上手。大部分示例都有真实的应用场景（如股票数据分析），可读性远远好于枯燥的官方文档，帮助读者在掌握NumPy使用技能的同时拓宽视野、拓展思维。本书的阅读门槛不高，读者只需具备基本的Python编程知识。

在翻译本书的过程中，我发现原作有不少地方不够严谨，甚至还有一些错误。经过反复核对确认，我已经修改了发现的错误，并在我认为不够严谨的地方以译者注的形式给出了自己的理解，供读者参考。此外，原书中的配图为屏幕截图，清晰度较低。为了读者获得最佳的阅读体验，我将书中的代码全部运行了一遍，并输出矢量图以替换原有的配图。需要说明的是，书中部分代码将会在线下载最近一年的股价数据，即数据的时间区段取决于代码运行时的日期。因此，这部分代码对应的新配图会与原书中的配图稍有差异，但对读者不会有任何影响。

感谢我的研究生导师王斌老师的力荐，他让我有幸成为本书的译者。王老师已有《信息检索导论》《大数据：大规模互联网数据挖掘与分布式处理》《机器学习实战》等诸多译著。他并不满足于自己畅读国外的优质书籍，而是字斟句酌、不辞辛劳地完成同样优质的译作与广大中文读者共飨，这份精神让我深受鼓舞。也正因为此，我才鼓起勇气接受了这次自我挑战，不遗余力地完成了人生第一次翻译工作。在这个过程中，要感谢图灵公司的李松峰老师对译稿提出细致严谨的修改意见，感谢傅志红、朱巍、毛倩倩、程芃等出版界同仁在审校、编辑阶段给予的帮助。最后，感谢我的家人，以及BrickMover Team的小伙伴们对翻译本书的支持。

由于自己的专业水平和翻译能力十分有限，加上时间仓促，译稿中的疏漏之处在所难免，恳请读者谅解。希望读者通过新浪微博@张驭宇UCAS和个人邮箱i@zhangyuyu.com，不吝提出宝贵的修改意见和建议，共同努力不断完善译稿。

对于每一个期望快速了解NumPy，却又担心自己迷失在浩如烟海的官方文档中的人，这本书值得一读。

张驭宇

2013年11月于中关村

献给我的家人和朋友们。

关于审稿人

Jaidev Deshpande是Enthought公司的一位实习生，他在那里主要做数据分析和数据可视化方面的工作。他是个狂热的科学计算程序员，在信号处理、数据分析和机器学习的很多开源项目中都有贡献。

Alexandre Devert博士在中国科学技术大学从事数据挖掘和软件工程的教学工作。他同时也是一位研究员，从事最优化问题的研究，并在生物技术创业公司中研究数据挖掘问题。在所有这些工作中，Alexandre非常乐于使用Python、NumPy和SciPy。

Mark Livingstone曾为三家跨国计算机公司（如今已不复存在）工作，在工程、产品支持、编程和培训等部门任职。他厌倦了被裁员的遭遇。2011年，他从澳大利亚黄金海岸的Griffith大学毕业，获得信息技术学士学位。目前是他攻读信息技术荣誉学士学位的最后一个学期，研究的是蛋白质相关的算法。研究工作中用到的软件均是在Mac电脑上用Python写成的。他的导师以及整个研究小组都感受到了Python编程的乐趣。

Mark喜欢指导需要帮助的大一新生，他是Griffith大学IEEE学生分会主席，也是Courthouse地区的太平绅士，曾经担任过信用合作社的主管，并将在2013年年底完成100次献血的计划。

他在业余时间也很多产，曾与人合作开发了S2 Salstat Statistics Package（项目主页[①]：http://code.google.com/p/salstat-statistics-package-2/）。这是一个跨平台的统计工具包，其中用到了wxPython、NumPy、SciPy、SciKit、Matplotlib等许多Python模块。

Miklós Prisznyák是一位有自然科学背景的资深软件工程师。他毕业于匈牙利历史最悠久、规模最大的大学Eötvös Lóránd大学，从事物理专业工作。他于1992年完成了硕士论文，研究了非阿贝尔的晶格量子场论的蒙特卡罗仿真。在匈牙利物理研究中心工作三年后，他加入了布达佩斯的MultiRáció Kft。这家公司是由一群物理学家创办的，专注于用数学方法分析数据和预测经济数据。在那里，他的主要项目是小区域内的失业统计分析系统，从那时候起，这一系统就一直被匈牙利政府用于公共就业服务。2000年他开始学习Python编程。2002年，他创办了自己的咨询公司，将Python用于所有能用的项目，服务于各种各样的保险、医药和电子商务公司。他还曾在意大利

① 该项目主页已过期并迁移至SourceForge：http://sourceforge.net/projects/s2statistical/。——译者注

的一家欧盟研究所工作，负责测试和优化基于Python的Zope/Plone分布式网络应用程序。2007年他移居英国，先是在一家苏格兰的创业公司工作，使用Twisted Python；随后在英国航天工业部门工作，使用PyQt窗口工具包、Enthought应用程序框架，以及NumPy和SciPy。2012年，他回到匈牙利并重新加入MultiRáció，目前的工作主要涉及OpenOffice/EuroOffice上的Python扩展模块，并再次使用NumPy和SciPy来让用户求解非线性问题和随机优化问题。Miklós喜欢旅行和阅读，他兴趣广泛，对自然科学、语言学、历史、政治、跳棋等均有涉猎。香浓的咖啡是他的最爱。不过，最美好的莫过于和他聪明的10岁大的儿子Zsombor一起享受时光。

Nikolay Karelin拥有光学博士学位，有近20年利用各种方法进行数值仿真和分析的经验，先后在学术界和工业界（从事光纤通信链路的仿真）工作。在初识Python和NumPy后，他在过去的五年内逐渐将这些优秀的工具用于几乎所有数值分析和脚本编写工作。

感谢我的家人，感谢他们在我审阅本书的一个个漫漫长夜所给予的理解与支持。

前　言

如今，科学家、工程师以及定量管理分析师面临着众多的挑战。数据科学家们希望能够用最小的编程代价在大数据集上进行数值分析，他们希望自己编写的代码可读性好、执行效率高、运行速度快，并尽可能地贴近他们熟悉的一系列数学概念。在科学计算领域，有很多符合这些要求的解决方案。

在这方面，C、C++和Fortran等编程语言各有优势，但它们不是交互式语言，并且被很多人认为过于复杂。常见的商业产品还有Matlab、Maple和Mathematica。这些产品提供了强大的脚本语言，但和通用编程语言比起来，功能依然很有限。另外还有一些类似于Matlab的开源工具，如R、GNU Octave和Scilab。显然，作为编程语言，它们都不如Python强大。

Python是一种流行的通用编程语言，在科学领域被广泛使用。你很容易在Python代码中调用以前的C、Fortran或者R代码。Python是面向对象语言，比C和Fortran更加高级。使用Python可以写出易读、整洁并且缺陷最少的代码。然而，Python本身并不具有与Matlab等效的功能块，而这恰恰就是NumPy存在的意义。本书就是要介绍NumPy以及相关的Python科学计算库，如SciPy和Matplotlib。

NumPy是什么

NumPy（Numerical Python的缩写）是一个开源的Python科学计算库。使用NumPy，就可以很自然地使用数组和矩阵。NumPy包含很多实用的数学函数，涵盖线性代数运算、傅里叶变换和随机数生成等功能。如果你的系统中已经装有LAPACK，NumPy的线性代数模块会调用它，否则NumPy将使用自己实现的库函数。LAPACK是一个著名的数值计算库，最初是用Fortran写成的，Matlab同样也需要调用它。从某种意义上讲，NumPy可以取代Matlab和Mathematica的部分功能，并且允许用户进行快速的交互式原型设计。

在本书中，我们不会从程序开发者的角度来讨论NumPy，而是更多地立足于用户，从他们的角度来分析它。不过值得一提的是，NumPy是一个非常活跃的开源项目，拥有很多的贡献者，也许有一天你也能成为其中的一员！

NumPy的由来

NumPy的前身是Numeric。Numeric最早发布于1995年，如今已经废弃了。由于种种原因，不管是Numeric还是NumPy，都没能进入Python标准库，不过单独安装NumPy也很方便。关于NumPy的安装，我们将在第1章中详细介绍。

早在2001年，一些开发者受Numeric的启发共同开创了一个叫做SciPy的项目。SciPy是一个开源的Python科学计算库，提供了类似于Matlab、Maple和Mathematica的许多功能。那段时间，人们对于Numeric越来越不满。于是，Numarray作为Numeric的替代品问世了。Numarray在某些方面比Numeric更强大，但是它们的工作方式却截然不同。鉴于此，SciPy继续遵循Numeric的工作方式，并延续了对Numeric数组对象的支持。虽然人们总是倾向于使用“最新最好”的软件，但是Numarray依然催生出了一整套的系统，包括很多周边的实用工具软件。

2005年，SciPy的早期发起人之一Travis Oliphant决定改变这一状况，他开始将Numarray的一些特性整合到Numeric中。一整套的代码重构工作就此开始，并于2006年NumPy 1.0发布的时候全部完成。于是NumPy拥有了Numeric和Numarray的所有特性，并且还新增了一些功能。SciPy提供了一个升级工具，可以让用户方便地从Numeric和Numarray升级到NumPy。由于Numeric和Numarray均不再活跃更新，升级是必然的。

如上所述，最初的NumPy其实是SciPy的一部分，后来才从SciPy中分离出来。如今，SciPy在处理数组和矩阵时会调用NumPy。

为什么使用NumPy

对于同样的数值计算任务，使用NumPy要比直接编写Python代码便捷得多。这是因为NumPy能够直接对数组和矩阵进行操作，可以省略很多循环语句，其众多的数学函数也会让编写代码的工作轻松许多。NumPy的底层算法在设计时就有着优异的性能，并且经受住了时间的考验。

NumPy中数组的存储效率和输入输出性能均远远优于Python中等价的基本数据结构（如嵌套的list容器）。其能够提升的性能是与数组中元素的数目成比例的。对于大型数组的运算，使用NumPy的确很有优势。对于TB级的大文件，NumPy使用内存映射文件来处理，以达到最优的数据读写性能。不过，NumPy数组的通用性不及Python提供的list容器，这是其不足之处。因此在科学计算之外的领域，NumPy的优势也就不那么明显了。关于NumPy数组的技术细节，我们将在后面详细讨论。

NumPy的大部分代码都是用C语言写成的，这使得NumPy比纯Python代码高效得多。NumPy同样支持C语言的API，并且允许在C源代码上做更多的功能拓展。C API的内容不在本书讨论之

列。最后要记往一点，NumPy是开源的，这意味着使用NumPy可以享受到开源带来的所有益处。价格低到了极限——免费。你再也不用担心每次有新成员加入团队时，就要面对软件授权及更新的问题了。开源代码是向所有人开放的，对于代码质量而言这是非常有利的。

NumPy的局限性

如果你是Java程序员，可能会对Jython感兴趣。Jython是Python语言在Java中的完整实现。遗憾的是，Jython运行在Java虚拟机上，无法调用NumPy，因为大部分NumPy模块是用C语言实现的。Python和Jython可以说是完全不同的两个世界，尽管它们实现的是同一套语言规范。当然，仍然有一些变通方案，具体内容在本书作者的另一本著作《NumPy攻略》中有所讨论。

本书内容

第1章指导你在系统中安装NumPy，并创建一个基本的NumPy应用程序。

第2章介绍NumPy数组对象以及一些基础知识。

第3章教你使用NumPy中最常用的基本数学和统计分析函数。

第4章讲述如何便捷地使用NumPy，包括如何选取数组的某一部分（例如根据一组布尔值来选取）、多项式拟合，以及操纵NumPy对象的形态。

第5章涵盖了矩阵和通用函数的内容。矩阵在数学中使用广泛，在NumPy中也有专门的对象来表示。通用函数（ufuncs）是一个能用于NumPy对象的标量函数，该函数的输入为一组标量，并将生成一组标量作为输出。

第6章探讨通用函数的一些基本模块。通用函数通常可映射到对应的数学运算，如加、减、乘、除等。

第7章介绍NumPy中的一些专用函数。作为NumPy用户，我们时常发现自己有一些特殊的需求。幸运的是，NumPy能满足我们的大部分需求。

第8章介绍怎样编写NumPy的单元测试代码。

第9章深入介绍非常有用的Python绘图库Matplotlib。虽然NumPy本身不能用来绘图，但是Matplotlib和NumPy两者完美地结合在一起，其绘图能力可与Matlab相媲美。

第10章更详细地介绍SciPy。如前所述，SciPy和NumPy是有历史渊源的，SciPy是一套高端Python科学计算框架，可以与NumPy共同使用。

第11章是本书的“餐后甜点”，这一章介绍如何用NumPy和Pygame写出有趣的游戏。同时，我们也将从中“品尝”到人工智能的“滋味”。

阅读条件

要试验本书中的代码，你需要安装最新版NumPy，因此要先安装能够运行NumPy的任一版Python。本书部分示例代码采用Matplotlib进行绘图，这些代码不一定需要读者全部运行，但依然推荐安装Matplotlib。本书最后一章讲的是SciPy，会讨论一个使用SciKits的例子。

以下是开发及测试示例代码所需的软件：

- Python 2.7
- NumPy 2.0.0.dev20100915
- SciPy 0.9.0.dev20100915
- Matplotlib 1.1.1
- Pygame 1.9.1
- IPython 0.14.dev

当然，我并不是要你在计算机上装全这些软件或者必须装指定版本，但Python和NumPy是必须安装的。

读者对象

本书适合正在找寻高质量开源数学库的科学家、工程师、程序员和分析师阅读参考。读者应具备一些基本的Python编程知识。此外，读者应该是经常与数学和统计学打交道，或起码对它们感兴趣。

排版约定

本书会通过不同样式区别不同类型的内容。下面给出部分样式的示例及解释。

正文中的代码格式如此处所示：“注意`numpysum()`函数中没有使用`for`循环。”

代码段如下所示：

```
def numpysum(n):
  a = numpy.arange(n) ** 2
  b = numpy.arange(n) ** 3
  c = a + b
  return c
```

当我们希望你注意代码中的某一部分时，会将相关的行或项用粗体表示：

```
reals = np.isreal(xpoints)
print "Real number?", reals
Real number? [ True True True True False False False False ]
```

命令行输入输出如下所示：

```
>>>fromnumpy.testing import rundocs
>>>rundocs('docstringtest.py')
```

新术语和重要的名词将用楷体表示。你在屏幕、菜单或对话框中看到的文本会采用加粗样式：“单击**Next**按钮进入下一界面。”

警告或重要说明将写在这里。

小贴士和技巧将写在这里。

读者反馈

一直以来，我们都非常欢迎读者朋友的意见反馈。请告诉我们你对本书的看法，以及你喜欢还是不喜欢书中的内容。你的意见对我们非常重要，我们将努力使你从阅读中得到最大的收获。

如果希望提出一些反馈意见，敬请发送邮件至feedback@packtpub.com，并请在邮件标题中写上书名。

如果你想看某方面的书并希望我们出版，请通过www.packtpub.com上的SUGGEST A TITLE表单提交选题建议，或发送邮件至suggest@packtpub.com。

如果你是某个领域的专家，或有兴趣写书，欢迎访问www.packtpub.com/authors，里面有我们的作者指南。

售后支持

感谢你购买Packt出版的图书。我们有诸多售后支持服务，希望给你提供最大的附加价值。

示例代码下载

如果你是www.packtpub.com的注册用户并从那里购买了图书，可以从网站上下载配套的示例代码[①]。如果你是在别处购买了本书，可以访问www.packtpub.com/support并注册，我们会直接将示例代码以邮件形式发送给你。

勘误

尽管我们处处小心以保证图书内容的准确性，但错误仍在所难免。如果你在阅读过程中发现错误并告知我们，不管是文字还是代码中的错误，我们都将不胜感激。这样做可使其他读者免于困惑，也能帮助我们不断改进后续版本。如果你发现任何错误，敬请访问www.packtpub.com/support报告给我们，即在网页上选择你购买的图书，单击errata submission form（提交勘误[②]）链接，并输入详细描述。一旦你提出的错误被证实，你的勘误将被接受并上传至我们的网站，或加入到已有的勘误列表中。若要查看已有勘误，请访问www.packtpub.com/support并通过书名查找。

关于盗版

在网上，所有媒体都会遭遇盗版问题。Packt非常重视版权保护工作。如果你在网上发现Packt出版物的任何非法副本，请立即向我们提供侵权网站的地址或名称，以便我们采取补救措施。

敬请通过copyright@packtpub.com联系我们，告知涉嫌侵权内容的链接。

我们非常感激你的帮助。这将保护我们作者的利益，同时也使我们有能力继续提供高品质的内容。

疑难解答

如果对本书的任何方面有疑问，欢迎发送邮件至questions@packtpub.com，我们将尽最大努力为你答疑解惑。

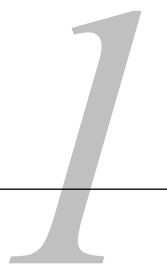
---

① 也可在图灵社区（iTuring.cn）本书网页免费注册下载。——编者注

② 关于本书中文版的勘误，请访问图灵社区（iTuring.cn）本书网页提交。——编者注

目　录

第 1 章 NumPy快速入门



让我们开始吧。首先，我们将介绍如何在不同的操作系统中安装NumPy和相关软件，并给出使用NumPy的简单示例代码。然后，我们将简单介绍IPython（一种交互式shell工具）。如前言所述，SciPy和NumPy有着密切的联系，因此你将多次看到SciPy的身影。在本章的末尾，我们将告诉你如何利用在线资源，以便你在受困于某个问题或不确定最佳的解题方法时，可以在线获取帮助。

本章涵盖以下内容：

- 在Windows、Linux和Macintosh操作系统上安装Python、SciPy、Matplotlib、IPython和NumPy；
- 编写简单的NumPy代码；
- 了解IPython；
- 浏览在线文档和相关资源。

1.1 Python

NumPy是基于Python的，因此在安装NumPy之前，我们需要先安装Python。某些操作系统已经默认安装有Python环境，但你仍需检查Python的版本是否与你将要安装的NumPy版本兼容。Python有很多种实现，包括一些商业化的实现和发行版。在本书中，我们将使用CPython[①]实现，从而保证与NumPy兼容。

1.2 动手实践：在不同的操作系统上安装 Python

NumPy在Windows、各种Linux发行版以及Mac OS X上均有二进制安装包。如果你愿意，也

① CPython是用C语言实现的Python解释器。——译者注

可以安装包含源代码的版本。你需要在系统中安装Python 2.4.x或更高的版本。我们将给出在以下操作系统中安装Python的各个步骤。

(1) Debian和Ubuntu Debian和Ubuntu可能已经默认安装了Python，但开发者包（development headers）[①]一般不会默认安装。在Debian和Ubuntu中安装python和python-dev的命令如下：

```
sudo apt-get install python
sudo apt-get install python-dev
```

(2) Windows Python的Windows安装程序可以在www.python.org/download下载。在这个站点中，我们也可以找到Mac OS X的安装程序，以及Linux、Unix和Mac OS X下的源代码包。

(3) Mac Mac OS X中预装了Python，而我们也可以通过MacPorts、Fink或者类似的包管理工具来获取Python。举例来说，可以使用如下命令安装Python 2.7：

```
sudo port install python27
```

LAPACK并不是必需的，但如果需要，NumPy在安装过程中将检测并使用之。我们推荐大家安装LAPACK以便应对海量数据的计算，因为它拥有高效的线性代数计算模块。

刚才做了些什么

我们在Debian、Ubuntu、Windows和Mac操作系统中安装了Python。

1.3　Windows

在Windows上安装NumPy是很简单的。你只需要下载安装程序，运行后在安装向导的指导下完成安装。

1.4　动手实践：在 Windows 上安装 NumPy、Matplotlib、SciPy 和 IPython

在Windows上安装NumPy是必需的，但幸运的是，安装过程并不复杂，我们将在下面详细阐述。建议你安装Matplotlib、SciPy和IPython，虽然这一操作对于使用本书不是必需的。我们将按照如下步骤安装这些软件。

(1) 从SourceForge网站下载NumPy的Windows安装程序：

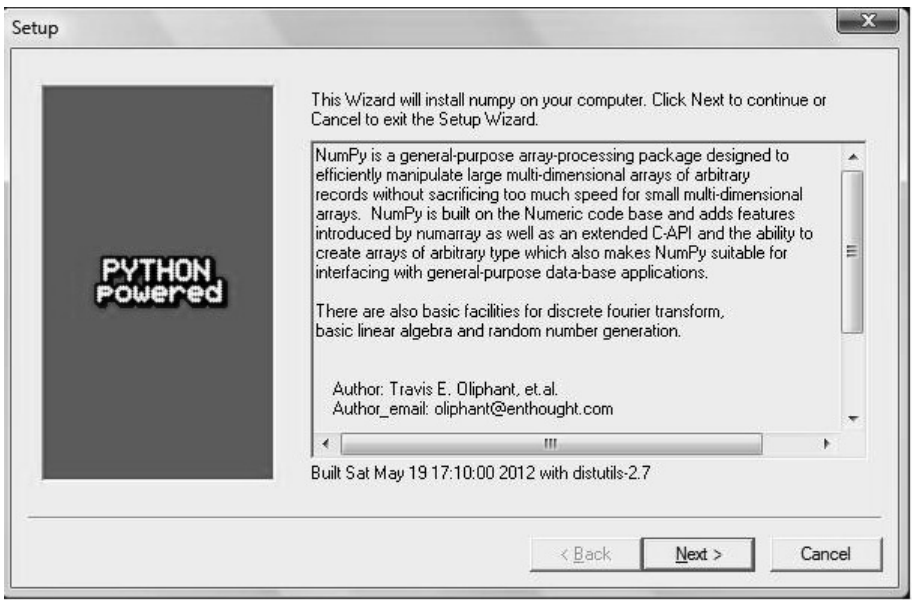
http://sourceforge.net/projects/numpy/files/

---

① 该软件包提供编译Python模块所需的静态库、头文件以及distutils工具等。——译者注

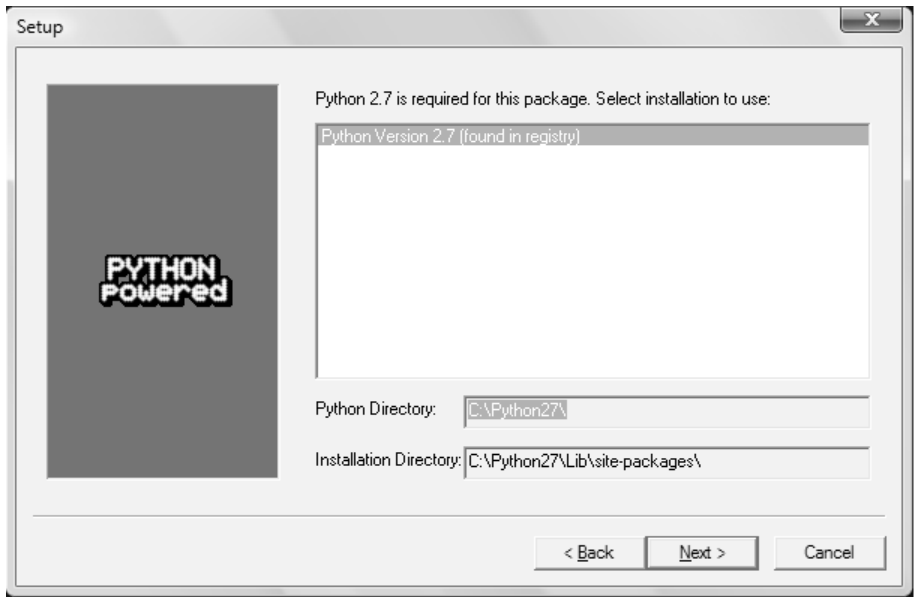
请选择合适的版本。在上图中，我们选择了numpy-1.7.0-win32-superpack-python2.7.exe。

(2) 下载完成后，双击运行安装程序。



(3) 现在，我们可以看到一段对NumPy的描述以及其特性，如上图所示。单击**Next**（下一步）按钮以继续安装。

(4) 如果你已经安装了Python，NumPy的安装程序应该能自动检测到。如果没有检测到Python，可能是你的路径设置有误。在本章的末尾，我们列出了一些在线资源，供安装NumPy时遇到问题的读者参考。



(5) 在上图中，安装程序成功检测到系统中已安装Python 2.7，此时应单击**Next**按钮继续安装；否则，请单击**Cancel**（取消）按钮并安装Python（NumPy不能脱离Python单独安装）。继续单击**Next**按钮，从这一步起就不能回退到上一步了，因此请你确认是否选择了合适的安装路径和其他安装选项。现在，真正的安装过程开始了，你需要等待一段时间。

(6) SciPy和Matplotlib可以通过Enthough安装，地址为www.enthought.com/products/epd.php。在安装过程中，你可能需要将一个文件msvcp71.dll放到目录C:\Windows\system32下。你可以从这里下载这个文件：www.dll-files.com/dllindex/dll-files.shtml?msvcp71。Windows下的IPython安装程序可以通过访问IPython的官网下载：http://ipython.scipy.org/Wiki/IpythonOnWindows。

刚才做了些什么

我们在Windows上安装了NumPy、SciPy、Matplotlib以及IPython。

1.5　Linux

在Linux上安装NumPy和相关软件的方法取决于具体使用的Linux发行版。我们将用命令行的方式安装NumPy，不过你也可以使用图形界面安装程序，这取决于具体的Linux发行版。除了软件包的名字不一样，安装Matplotlib、SciPy和IPython的命令与安装NumPy时是完全一致的。这几个软件包不是必需安装的，但这里建议你也一并安装。

1.6 动手实践：在 Linux 上安装 NumPy、Matplotlib、SciPy 和 IPython

大部分Linux发行版都有NumPy的软件包。我们将针对一些流行的Linux发行版给出安装步骤。

(1) 要在Red Hat上安装NumPy，请在命令行中执行如下命令：

```
yum install python-numpy
```

(2) 要在Mandriva上安装NumPy，请在命令行中执行如下命令：

```
urpmi python-numpy
```

(3) 要在Gentoo上安装NumPy，请在命令行中执行如下命令：

```
sudo emerge numpy
```

(4) 要在Debian或Ubuntu上安装NumPy，请在命令行中执行如下命令：

```
sudo apt-get install python-numpy
```

下表给出了各Linux发行版中相关软件包的名称以供参考。

Linux发行版	NumPy	SciPy	Matplotlib	IPython
Arch Linux	python-numpy	python-scipy	python-matplotlib	ipython
Debian	python-numpy	python-scipy	python-matplotlib	ipython
Fedora	numpy	python-scipy	python-matplotlib	ipython
Gentoo	dev-python/numpy	scipy	matplotlib	ipython
OpenSUSE	python-numpy,python-numpy-devel	python-scipy	python-matplotlib	ipython
Slackware	numpy	scipy	matplotlib	ipython

刚才做了些什么

我们在各种Linux发行版上安装了NumPy、SciPy、Matplotlib以及IPython。

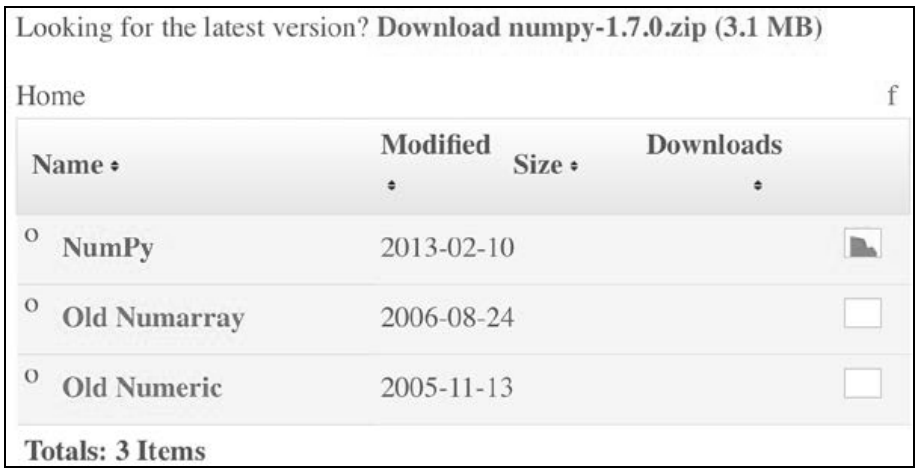
1.7 Mac OS X

在Mac上，你可以通过图形用户界面或者命令行来安装NumPy、Matplotlib和SciPy，根据自己的喜好选择包管理工具，如MacPorts或Fink等。

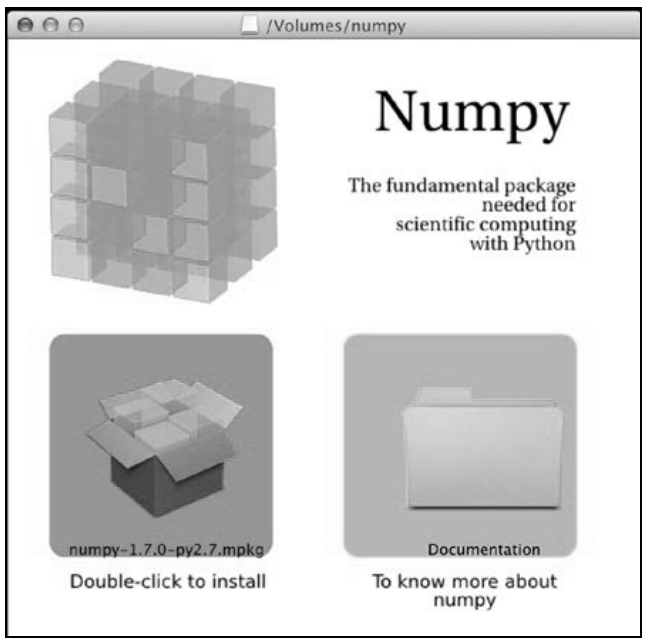
1.8 动手实践：在 Mac OS X 上安装 NumPy、Matplotlib 和 SciPy

我们将使用图形用户界面安装程序，安装步骤如下所示。

(1) 我们先从SourceForge页面下载NumPy的安装程序，地址为http://sourceforge.net/projects/numpy/files/。Matplotlib和SciPy也可以用类似的方式下载，只需将前面URL中的numpy修改为scipy或matplotlib。在我写作本书的时候，IPython还没有提供图形界面的安装程序。如下面的截图所示，请单击下载合适的DMG文件，且通常要选择最新版本的文件。

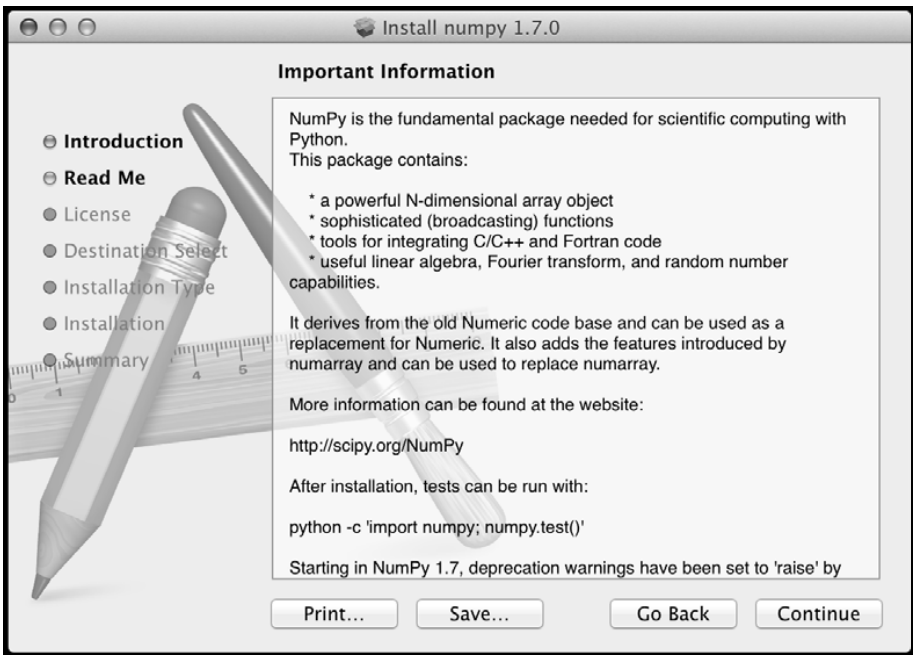


(2) 打开下载的DMG文件（示例中为numpy-1.7.0-py2.7-python.org-macosx10.6.dmg），如下图所示。



❑ 在打开的窗口中，双击那个下方文字以**.mpkg**结尾的图标，我们将看到安装程序的欢迎界面。

- 单击Continue（继续）按钮进入**Read Me**（自述页）界面，我们将看到一小段NumPy的描述文字，如下图所示 。



- 单击**Continue**按钮，可以看到关于软件许可协议的说明。

(3) 阅读软件许可协议，单击**Continue**按钮，在被提示是否接受协议时单击**Accept**（同意）按钮。继续安装，最后单击**Finish**（完成）按钮结束安装。

刚才做了些什么

我们在Mac OS X上用图形用户界面安装程序安装了NumPy。SciPy和Matplotlib的安装步骤与之很类似，使用上面第(1)步中提到的URL进行下去即可。

1.9 动手实践：使用 MacPorts 或 Fink 安装 NumPy、SciPy、Matplotlib 和 IPython

我们也可以选择另外一种安装方式，即使用MacPorts或Fink来安装NumPy、SciPy、Matplotlib以及IPython。下面给出的安装步骤将安装所有这些软件包。在本书中只有NumPy是必需的，如果你对其他软件包不感兴趣，也可以暂不安装。

(1) 输入以下命令，从MacPorts安装这些软件包：

```
sudo port install py-numpy py-scipy py-matplotlib py-ipython
```

(2) Fink也包含了相关软件包：NumPy的有`scipy-core-py24`、`scipy-core-py25`和`scipy-core-py26`；SciPy的有`scipy-py24`、`scipy-py25`和`scipy-py26`。执行如下命令，我们来安装基于Python 2.6的NumPy以及其他推荐安装的软件包：

```
fink install scipy-core-py26 scipy-py26 matplotlib-py26
```

刚才做了些什么

我们在Mac OS X上使用MacPorts和Fink安装了NumPy以及其他推荐安装的软件包。

1.10　编译源代码

NumPy的源代码可以使用`git`获取，如下所示：

```
git clone git://github.com/numpy/numpy.git numpy
```

使用如下命令将NumPy安装至/usr/local：

```
python setup.py build
sudo python setup.py install --prefix=/usr/local
```

我们需要有C编译器（如GCC）和Python开发者包（`python-dev`或`python-devel`），然后才可对源代码进行编译。

1.11　数组对象

在介绍完NumPy的安装步骤后，我们来看看NumPy中的数组对象。NumPy数组在数值运算方面的效率优于Python提供的list容器。使用NumPy可以在代码中省去很多循环语句，因此其代码比等价的Python代码更为简洁。

1.12　动手实践：向量加法

假设我们需要对两个向量`a`和`b`做加法。这里的向量即数学意义上的一维数组，随后我们将在第5章中学习如何用NumPy数组表示矩阵。向量`a`的取值为0~*n*的整数的平方，例如`n`取3时，向量`a`为0、1或4。向量`b`的取值为0~*n*的整数的立方，例如`n`取3时，向量`b`为0、1或8。用纯Python代码应该怎么写呢？我们先想一想这个问题，随后再与等价的NumPy代码进行比较。

(1) 以下的纯Python代码可以解决上述问题：

```
def pythonsum(n):
    a = range(n)
    b = range(n)
    c = []

    for i in range(len(a)):
        a[i] = i ** 2
        b[i] = i ** 3
        c.append(a[i] + b[i])

    return c
```

(2) 以下是使用NumPy的代码，它同样能够解决问题：

```
def numpysum(n):
  a = numpy.arange(n) ** 2
  b = numpy.arange(n) ** 3
  c = a + b
  return c
```

注意，numpysum()函数中没有使用for循环。同时，我们使用NumPy中的arange函数来创建包含0~*n*的整数的NumPy数组。代码中的arange函数前面有一个前缀numpy，表明该函数是从NumPy模块导入的。

下面我们做一个有趣的实验。在前言部分我们曾提到，NumPy在数组操作上的效率优于纯Python代码。那么究竟快多少呢？接下来的程序将告诉我们答案，它以微秒（10^{-6} s）的精度分别记录下numpysum()和pythonsum()函数的耗时。这个程序还将输出加和后的向量最末的两个元素。让我们来看看纯Python代码和NumPy代码是否得到相同的结果：

```
#!/usr/bin/env/python

import sys
from datetime import datetime
import numpy as np

"""
 本书第1章
 该段代码演示Python中的向量加法
 使用如下命令运行程序：

    python vectorsum.py n

 n为指定向量大小的整数

 加法中的第一个向量包含0到n的整数的平方
 第二个向量包含0到n的整数的立方
 程序将打印出向量加和后的最后两个元素以及运行消耗的时间
"""
```

```
def numpysum(n):
    a = np.arange(n) ** 2
    b = np.arange(n) ** 3
    c = a + b

    return c

def pythonsum(n):
    a = range(n)
    b = range(n)
    c = []

    for i in range(len(a)):
        a[i] = i ** 2
        b[i] = i ** 3
        c.append(a[i] + b[i])

    return c

size = int(sys.argv[1])

start = datetime.now()
c = pythonsum(size)
delta = datetime.now() - start
print "The last 2 elements of the sum", c[-2:]
print "PythonSum elapsed time in microseconds", delta.microseconds
start = datetime.now()
c = numpysum(size)
delta = datetime.now() - start
print "The last 2 elements of the sum", c[-2:]
print "NumPySum elapsed time in microseconds", delta.microseconds
```

程序在向量元素个数为1000、2000和3000时的输出分别为：

```
$ python vectorsum.py 1000
The last 2 elements of the sum [995007996, 998001000]
PythonSum elapsed time in microseconds 707
The last 2 elements of the sum [995007996 998001000]
NumPySum elapsed time in microseconds 171

$ python vectorsum.py 2000
The last 2 elements of the sum [7980015996, 7992002000]
PythonSum elapsed time in microseconds 1420
The last 2 elements of the sum [7980015996 7992002000]
NumPySum elapsed time in microseconds 168

$ python vectorsum.py 4000
The last 2 elements of the sum [63920031996, 63968004000]
PythonSum elapsed time in microseconds 2829
The last 2 elements of the sum [63920031996 63968004000]
NumPySum elapsed time in microseconds 274
```

如果你是www.packtpub.com的用户并从那里购买了图书，可以从网站上下载配套的示例代码。如果你是在别处购买了本书，可以访问www.packtpub.com/support进行登记，我们会直接将示例代码文件以邮件形式发送给你。

刚才做了些什么

显然，NumPy代码比等价的纯Python代码运行速度快得多。有一点可以肯定，即不论我们使用NumPy还是Python，得到的结果是一致的。不过，两者的输出结果在形式上有些差异。注意，`numpysum()`函数的输出不包含逗号。这是为什么呢？显然，我们使用的是NumPy数组，而非Python自身的list容器。正如前言中所述，NumPy数组对象以专用数据结构来存储数值。我们将在下一章中详细介绍NumPy数组对象。

突击测验：**arange**函数的功能

问题1 `arrange(5)`的作用是什么？

(1) 创建一个包含5个元素的Python列表（list），取值分别为1~5的整数

(2) 创建一个包含5个元素的Python列表，取值分别为0~4的整数

(3) 创建一个包含5个元素的NumPy数组，取值分别为1~5的整数

(4) 创建一个包含5个元素的NumPy数组，取值分别为0~4的整数

(5) 以上都不对

勇敢出发：进一步分析

我们用来比较NumPy和常规Python代码运行速度的程序不是特别严谨，如果将相同的实验重复多次并计算相应的统计量（如平均运行时间等）会更科学。你可以把实验结果绘制成图表，并展示给你的好友和同事。

我们在本章的最后给出了一些在线文档和相关资源，你可以在线获取帮助。顺便提一下，NumPy中的统计函数可以帮你计算平均数。建议你使用Matplotlib绘图。（第9章概述了Matplotlib。）

1.13 IPython：一个交互式 shell 工具

科学家和工程师都喜欢做实验，而IPython正是诞生于爱做实验的科学家之手。IPython提供

的交互式实验环境被很多人认为是Matlab、Mathematica和Maple的开源替代品。你可以在线获取包括安装指南在内的更多信息，地址为http://ipython.org/。

IPython是开源免费的软件，可以在Linux、Unix、Mac OS X以及Windows上使用。IPython的作者们希望那些用到IPython的科研工作成果在发表时能够提到IPython，这是他们对IPython使用者唯一的要求。下面是IPython的基本功能：

- Tab键自动补全；
- 历史记录存档；
- 行内编辑；
- 使用`%run`可以调用外部Python脚本；
- 支持系统命令；
- 支持pylab模式；
- Python代码调试和性能分析。

在pylab模式下，IPython将自动导入`SciPy`、`NumPy`和`Matplotlib`模块。如果没有这个功能，我们只能手动导入每一个所需模块。

而现在，我们只需在命令行中输入如下命令：

```
$ ipython --pylab
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
Type "copyright", "credits" or "license" for more information.

IPython 0.14.dev -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: MacOSX].
For more information, type 'help(pylab)'.
In [1]: quit()
```

使用`quit()`函数或快捷键Ctrl+D均可以退出IPython shell。有时我们想要回到之前做过的实验，IPython可以便捷地保存会话以便稍后使用。

```
In [1]: %logstart
Activating auto-logging. Current session state plus future input saved.
Filename       : ipython_log.py
Mode           : rotate
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
```

举例来说，我们将之前的向量加法程序放在当前目录下，可以按照如下方式运行脚本：

```
In [1]: ls
README          vectorsum.py
In [2]: %run -i vectorsum.py 1000
```

你可能还记得，这里的1000是指向量中元素的数量。%run的-d参数将开启ipdb调试器，键入c后，脚本就开始逐行执行了（如果脚本有*n*行，就一共执行n步直到代码结束）。在ipdb提示符后面键入quit可以关闭调试器。

```
In [2]: %run -d vectorsum.py 1000
*** Blank or comment
*** Blank or comment
Breakpoint 1 at: /Users/.../vectorsum.py:3
```

在ipdb>提示符后面键入c，从而开始运行代码。

```
><string>(1)<module>()
ipdb> c
> /Users/.../vectorsum.py(3)<module>()
      2
1---> 3 import sys
      4 from datetime import datetime
ipdb> n
>
/Users/.../vectorsum.py(4)<module>()
1     3 import sys
----> 4 from datetime import datetime
      5 import numpy
ipdb> n
> /Users/.../vectorsum.py(5)<module>()
      4 from datetime import datetime
----> 5 import numpy
      6
ipdb> quit
```

我们还可以使用%run的-p参数对脚本进行性能分析。

```
In [4]: %run -p vectorsum.py 1000
         1058 function calls (1054 primitive calls) in 0.002 CPU seconds
   Ordered by: internal time
ncallstottimepercallcumtimepercallfilename:lineno(function)
1 0.001    0.001    0.001    0.001 vectorsum.py:28(pythonsum)
1 0.001    0.001    0.002    0.002 {execfile}
1000 0.000    0.0000.0000.000    {method "append" of 'list' objects}
1 0.000      0.000     0.002    0.002 vectorsum.py:3(<module>)
1 0.000      0.0000.0000.000  vectorsum.py:21(numpysum)
3    0.000    0.0000.0000.000    {range}
1    0.000    0.0000.0000.000    arrayprint.py:175(_array2string)
3/1    0.000    0.0000.0000.000 arrayprint.py:246(array2string)
2    0.000    0.0000.0000.000    {method 'reduce' of 'numpy.ufunc' objects}
4    0.000    0.0000.0000.000    {built-in method now}
2    0.000    0.0000.0000.000    arrayprint.py:486(_formatlnteger)
2    0.000    0.0000.0000.000    {numpy.core.multiarray.arange}
1    0.000    0.0000.0000.000    arrayprint.py:320(_formatArray)
```

```
3/1    0.000    0.0000.0000.000 numeric.py:1390(array_str)
1    0.000    0.0000.0000.000    numeric.py:216(asarray)
2    0.000    0.0000.0000.000    arrayprint.py:312(_extendLine)
1    0.000    0.0000.0000.000    fromnumeric.py:1043(ravel)
2    0.000    0.0000.0000.000    arrayprint.py:208(<lambda>)
1    0.000    0.000    0.002    0.002<string>:1(<module>)
11    0.000    0.0000.0000.000 {len}
2    0.000    0.0000.0000.000    {isinstance}
1    0.000    0.0000.0000.000    {reduce}
1    0.000    0.0000.0000.000    {method 'ravel' of 'numpy.ndarray' objects}
4    0.000    0.0000.0000.000    {method 'rstrip' of 'str' objects}
3    0.000    0.0000.0000.000    {issubclass}
2    0.000    0.0000.0000.000    {method 'item' of 'numpy.ndarray' objects}
1    0.000    0.0000.0000.000    {max}
1    0.000    0.0000.0000.000    {method 'disable' of ' lsprof.Profiler'
objects}
```

根据性能分析的结果，可以更多地了解程序的工作机制，并能够据此找到程序的性能瓶颈。使用`%hist`命令可以查看命令行历史记录。

```
In [2]: a=2+2
In [3]: a
Out[3]: 4
In [4]: %hist
1: _ip.magic("hist ")
2: a=2+2
3: a
```

通过前面的介绍，希望你也认为IPython是非常有用的工具了！

1.14　在线资源和帮助

在IPython的pylab模式下，我们可以使用`help`命令打开`NumPy`函数的手册页面。你并不需要知道所有函数的名字，因为可以在键入少量字符后按下Tab键进行自动补全。例如，我们来查看一下`arange`函数的相关信息。

```
In [2]: help ar<Tab>
```

arange	arctan	argsort	array_equal	arrow
arccos	arctan2	argwhere	array_equiv	
arccosh	arctanh	around	array_repr	
arcsin	argmax	array	array_split	
arcsinh	argmin	array2string	array_str	

```
In [2]: help arange
```

另一种方法是在函数名后面加一个问号：

```
In [3]: arange?
```

可以访问http://docs.scipy.org/doc/查看NumPy和SciPy的在线文档。在这个网页上，你可以访问http://docs.scipy.org/doc/numpy/reference/浏览NumPy的参考资料、用户指南以及一些使用教程。

NumPy的wiki站点上也有许多相关文档可供参考：http://docs.scipy.org/numpy/Front%20Page/。

NumPy和SciPy的论坛地址为http://ask.scipy.org/en。

广受欢迎的开发技术问答网站Stack Overflow上有成百上千的提问被标记为`numpy`相关问题。你可以到这里查看这些问题：http://stackoverflow.com/questions/tagged/numpy。

如果你确实被某个问题困住了，或者想了解NumPy开发的最新进展，可以订阅NumPy的讨论组邮件列表，电子邮件地址为numpy-discussion@scipy.org。每天的邮件数量不会太多，并且基本不会讨论无意义的事情。最重要的是，NumPy的活跃开发者们也愿意回答讨论组里提出的问题。完整的邮件列表可以在这里找到：www.scipy.org/Mailing_Lists。

对于IRC[①]的用户，可以查看irc.freenode.net上的IRC频道。尽管该频道的名称是`#scipy`，但你也可以提出NumPy的问题，因为SciPy是基于NumPy的，SciPy的用户也了解NumPy的知识。SciPy频道上至少会有50人同时在线。

1.15 本章小结

在本章中，我们安装了NumPy以及其他推荐软件。我们成功运行了向量加法程序，并以此证明了NumPy优异的性能。随后，我们介绍了交互式shell工具IPython。此外，我们还列出了供你参考的NumPy文档和在线资源。

在下一章中，我们将深入了解NumPy中的一些基本概念，包括数组和数据类型。

① Internet Relay Chat，一种公开的协议，用于网络即时聊天。——译者注