<<Python标准库>>

图书基本信息

书名：<<Python标准库>>

13位ISBN编号：9787111378105

10位ISBN编号：7111378105

出版时间：2012-6-15

出版时间：机械工业出版社华章公司

作者：Doug Hellmann

页数：1016

译者：刘炽

前言

序　　今天是2010年的感恩节。
不论人们是否身在美国，在这个节日里，大家都一边品尝丰盛的食物，一边欣赏橄榄球比赛，有些人可能会出门逛逛。
对我（以及其他很多人）来说，会借此机会回顾一下过去的岁月，想想那些让我们的生活充满色彩的人和事，向他们致以感谢。
当然，我们每天都该这么做，不过专门有一天来表达谢意有时会让我们想得更深远一些。
现在我坐在这里为本书写序，非常感谢能有机会做这件事，不过我想到的不只是本书的内容，也不只是作者本人（一个无比热情的社区成员），我所想的是这个主题本身—Python，具体来讲，还有它的标准库。
当前发布的每版Python都包含数百个模块，它们是多年来多位开发人员针对多个主题和多个任务共同完成的。
这些模块涵盖一切，从发送和接收Email，到GUI开发，再到内置的HTTP服务器都一应俱全。
就其本身而言，标准库的开发和维护是一项极其庞大的工作。
如果没有多年来一直维护它的人们，没有数以千计的人提交补丁、文档和提供反馈，它绝不会成为今天的模样。
这是一个惊人的成就，在Python日益普及的今天（不论是作为语言还是一种生态系统），标准库已经成为其中不可或缺的重要组成部分。
如果没有标准库，没有核心团队和其他人员的“内含动力”（batteries included）口号，Python绝对不可能走这么远。
它已经被成千上万的人和公司下载，并已安装在数百万服务器、台式机和其他设备上。
即使没有标准库，Python仍是一种很不错的语言，在教学、学习和可读性方面有扎实的基础。
基于这些优点，它本身也能发展得足够好。
不过标准库把它从一种有趣的体验变成为一个强大而有效的工具。
每一天，全世界的开发人员都在构建工具和完整的应用，他们所基于的只是核心语言和标准库。
你不仅要能够明确概念，描述汽车是什么（语言），还要得到足够的部件和工具来自行组装一辆基本的汽车。
它可能并不完善，不过可以使你从无到有，这将是一个很好的奖励，会赋予你巨大的动力。
我曾反复对那些骄傲地看着我的人说：“看看我构建的应用，除了Python提供的，其他的工具通通都没有用到！
”　　不过，标准库也不是完美无缺的，它也有自己的缺陷。
由于标准库的规模、广度和它的“年龄”，毫无疑问有些模块有不同层次的质量、API简洁性和覆盖性。
有些模块存在“特性蔓延”的问题，或者无法跟进其覆盖领域中的最新进展。
通过众多不计酬劳的志愿者的帮助和辛勤工作，Python还在继续发展、壮大和改进。
不过，有些人对Python还有争议，不仅由于它的缺点，而且因为标准库并不一定构成其模块涵盖领域中“最顶尖的”解决方案（毕竟，“最佳”是一个不断改变和调整的目标），因而认为应当将它完全舍弃，尽管它还在不断改善。
这些人遗漏了一个事实：标准库不仅是促使Python不断成功的一个重要组成部分，而且尽管存在瑕疵，它还是一个绝妙的资源。
不过我有意忽略了一个巨大的领域：文档。
标准库的文档很不错，还在继续改进和发展。
由于标准库的庞大规模和广度，相应的文档规模也很惊人。
在数以千计的开发人员和用户的努力下，我们有成百上千页文档，实在让人佩服。
每天都有数万人在使用这些文档创建应用—可能简单到只是一页的脚本，也可能很复杂，比如控制大型机械手的软件。

正是由于文档，我们才会看到本书。
所有好的文档和代码都有一个起点—关于目标“是什么”以及“将是什么”要有一个核心概念。
从这个内核出发，才有了角色（API）和故事情节（模块）。
谈到代码，有时代码会从一个简单的想法开始：“我想解析一个字符串，查找一个日期。”不过等结束时，你可能已经查看了数百个单元测试、函数以及你编写的其他代码，你会坐下来，发现自己构建的东西远远超出了原先的设想。
文档也是如此，特别是代码文档。
在我看来，代码文档中最重要的部分就属于这种例子。
要写有关一个API中某一部分的描述，你可能会写上几本书，可以用华丽的文字和经过深思熟虑的用例来描述松耦合的接口。
不过，如果第一次查看这个描述的用户无法将这些华丽的文字、仔细考量的用例和API签名结合在一起，构建出有意义的应用并解决他们的问题，这一切就完全是徒劳。
人们建立重要连接所用的网关也属于这种例子，这些逻辑会从一个抽象概念跳转到具体的事物。“了解”思想和API是一回事；知道它如何使用则是另外一回事。
如果你不仅想要学到东西，还希望改善现状，这会很有帮助。
这就把我们重新引回Python。
本书作者Doug Hellmann在2007年创建了一个名为“Python Module of the Week”的博客。在这个博客中，他全面研究了标准库的众多模块，采用一种“示例为先”的方式介绍各个模块如何工作以及为什么。
从读到它的第一天起，它就成为我除了核心Python文档之外的又一个必访之地。
他的作品已经成为我以及Python社区其他人不可缺少的必备资源。
Doug的文章填补了当前我所看到的Python文档的一个重大空白：对例子的迫切需求。
用一种有效而简单的方式展示如何做以及为什么这么做，这绝非易事。
我们已经看到，这也是一项很重要、很有价值的工作，对人们每一天的工作都有帮助。
人们频繁地给我发邮件，告诉我：“你看过Doug的这个帖子吗？
实在太棒了！
”或者“为什么这个不能放在核心文档里呢？
它能帮助我了解到底是怎么做的！
”　　当我听说Doug准备花些时间进一步完善他现有的工作，把它变成一本书，让我能把它放在桌子上反复翻阅，以备急用，我真是太兴奋了。
Doug是一位非凡的技术作者，而且能敏锐地捕捉到细节。
有一整本书专门讲解实际例子，介绍标准库中一百多个模块是如何工作的，而且是他来写这本书，实在让我欣喜若狂。
所以，我要感谢Python，感谢标准库（包括它的瑕疵），感谢我拥有的这个活力充沛有时也存在问题的庞大Python社区。
我要感谢核心开发小组的辛勤工作，包括过去、现在，还有将来。
我还要感谢这么多社区成员提供的资源、投入的时间和做出的努力，其中Doug Hellmann更是卓越的代表，正是这些让这个社区和生态系统如此生机勃勃。
最后，我要感谢本书。
继续向它的作者表示敬意，这本书会在未来几年得到充分利用。
Jesse Noller　　Python核心开发人员　　PSF Board成员　　Nasuni公司首席工程师

<<Python标准库>>

内容概要

本书由资深Python专家亲自执笔，Python语言的核心开发人员作序推荐，权威性毋庸置疑。

对于程序员而言，标准库与语言本身同样重要，它好比一个百宝箱，能为各种常见的任务提供完美的解决方案，所以本书是所有Python程序员都必备的工具书！
本书以案例驱动的方式讲解了标准库中一百多个模块的使用方法（如何工作）和工作原理（为什么要这样工作），比标准库的官方文档更容易理解（一个简单的示例比一份手册文档更有帮助），为Python程序员熟练掌握和使用这些模块提供了绝佳指导。

全书一共19章，系统而全面地对Python标准库中的一百多个模块进行了生动的讲解。
这些模块主要包括：文本处理工具模块、与数据结构相关的模块、与算法有关的模块、管理日期和时间值的模块、用于数学计算的模块、管理文件系统的模块、用于数据存储与交换的模块、用于数据压缩与归档的模块、用于加密的模块、...
(展开全部)　　本书由资深Python专家亲自执笔，Python语言的核心开发人员作序推荐，权威性毋庸置疑。

对于程序员而言，标准库与语言本身同样重要，它好比一个百宝箱，能为各种常见的任务提供完美的解决方案，所以本书是所有Python程序员都必备的工具书！
本书以案例驱动的方式讲解了标准库中一百多个模块的使用方法（如何工作）和工作原理（为什么要这样工作），比标准库的官方文档更容易理解（一个简单的示例比一份手册文档更有帮助），为Python程序员熟练掌握和使用这些模块提供了绝佳指导。

全书一共19章，系统而全面地对Python标准库中的一百多个模块进行了生动的讲解。
这些模块主要包括：文本处理工具模块、与数据结构相关的模块、与算法有关的模块、管理日期和时间值的模块、用于数学计算的模块、管理文件系统的模块、用于数据存储与交换的模块、用于数据压缩与归档的模块、用于加密的模块、与进程和线程相关的模块、与网络通信和Email相关的模块、构建模块、支持处理多种自然语言和文化设置的模块、开发工具模块、与运行时特性相关的模块，等等。

作者简介

Doug Hellmann目前是Racemi公司的一位高级开发人员，也是Python Software Foundation的信息交流主管。
从1.4版开始他就一直在做Python编程，曾在大量UNIX和非UNIX平台上参与项目开发，涉及领域包括地图、医疗新闻播报、金融和数据中心自动化。
为《Python Magazine》做了一年普通专栏作家后，他在2008—2009年成为这家杂志的主编。
自2007年以来，Doug在他的博客上发表了颇受关注的“ Python Module of the Week ”系列。
他居住在乔治亚州的Athens。

<<Python标准库>>

书籍目录

章节摘录

版权页： 垂持久存储数据以供长期使用，这包括两个方面：在对象的内存中表示和存储格式之间来回转换数据，以及处理转换后数据的存储区。
标准库包含很多模块，可以在不同情况下处理这两个方面。
有两个模块可以将对象转换为一种可以传输或存储的格式[这个过程称为串行化（serializing）]。
最常用的是使用pickle完成持久存储，因为它可以与其他一些具体存储串行化数据的标准库模块集成，如shelve。
不过对基于Web的应用，json则更为常用，因为它能更好地与现有Web服务存储工具集成。
一旦将内存中对象转换为一种可以保存的格式，下一步就是确定如何存储这个数据。
如果数据不需要以某种方式索引，依序先后写入串行化对象的简单平面文件就很适用。
Python包括一组模块可以在一个简单的数据库中存储键一值对，需要索引查找时会使用某种变种DBM格式。
要利用DBM格式，最直接的方式就是shelve。
可以打开shelve文件，并通过一个类字典的API来访问。
保存到数据库的对象会自动pickle并保存，而无须调用者做任何额外的工作。
不过shelve有一个缺点，使用默认接口时，没有办法预测将使用哪一个DBM格式，因为它会根据创建数据库的系统上有哪些可用的库来进行选择。
如果应用不需要在库配置不同的主机之间共享数据库文件，那么选择哪种格式并不重要；不过，如果必须保证可移植性，可以使用这个模块中的某个类来确保选择一个特定的格式。
对于Web应用，由于这些应用处理的就是JSON格式的数据，因此使用json和anydbm可以提供另一种持久存储机制。
直接使用anydbm会比shelve稍微多做一些工作，因为DBM数据库键和值必须是字符串，而且在数据库中访问值时不会自动地重新创建对象。
大多数Python发布版本都提供了sqlite3进程中关系数据库，可以采用比键一值对更复杂的组织来存储数据。
它将数据库存储在内存中或者存储在一个本地文件中，所有访问都来自同一个进程，所以不存在网络通信延迟。
sqlite3的紧凑性使它尤其适合嵌入到桌面应用或Web应用的开发版本中。

<<Python标准库>>

编辑推荐

《Python标准库》的读者应该是中等水平的Python程序员，所以尽管书中对所有源代码都做了讨论，但只有一部分会逐行给出解释。
每节会通过源代码和完全独立的示例程序的输出来重点介绍一个模块的特性。
我会尽可能简洁地介绍各个特性，使读者能够把重点放在所展示的模块或函数上，而不会因支持代码而分心。

版权说明