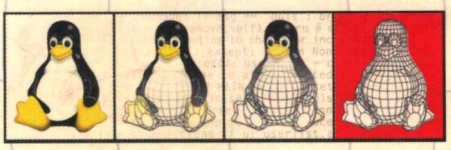
书中有40多个完整的、带有注释的脚本，这是学习Python更好的方法

Python 编程指南

Programming with Python



[美]Tim Altom, Mitch Chapman 著
云舟工作室 译

PRIMA TECH
中国水利水电出版社
www.waterpub.com.cn

Python 编程指南

[美] Tim Altom，Mitch Chapman 著

云舟工作室 译

中国水利水电出版社

内 容 提 要

本书是一本全面介绍 Python 语言的书籍，作者首先简明扼要地介绍了 Python 的语句、模块、数据类型、函数和模块等基础知识，然后通过大量的示例程序，详细介绍了 Python 在 Tkinter 脚本、数据库、数学/科学函数、服务器、字符串和其他数据类型、系统操作、游戏和人工智能方面的应用。并且作者将 Python 的 FAQ 放在本书的附录中，方便读者查阅。

本书适合 Python 初学者，对于有一定 C 语言基础的人，学习本书将更加容易。

北京市版权局著作权合同登记号：图字 01-2001-2163 号

图书在版编目（CIP）数据

Python 编程指南/（美）阿尔托马（Altom, T.），（美）查普曼（Chapman, M.）著；云舟工作室译．－北京：中国水利水电出版社，2001

ISBN 7-5084-0898-5

Ⅰ．P…　Ⅱ．①阿…②查…③云…　Ⅲ．Python 语言－程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2001）第 084632 号

书　　名	Python 编程指南
作　　者	[美]Tim Altom，Mitch Chapman　著
译　　者	云舟工作室　译
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）68359286（万水）、63202266（总机）、68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787×1092 毫米　16 开本　20.75 印张　452 千字
版　　次	2002 年 1 月第一版　2002 年 1 月北京第一次印刷
印　　数	0001—4000 册
定　　价	40.00 元（含 1CD）

译 者 序

Python 是一种脚本语言。脚本语言是类似 DOS 批处理、UNIX shell 程序的语言。脚本语言不需要每次编译再执行，并且在执行中可以很容易地访问正在运行的程序，甚至可以动态地修改正在运行的程序，适用于快速地开发以及完成一些简单的任务。在这样的情况下，Python 可能正好适合你的需要。

Python 是一门解释性的、面向对象的、动态语义特征的高层语言。Python 的简单而易于阅读的语法强调了可读性，因此降低了程序维护的费用。Python 支持模块和包，并鼓励程序模块化和代码重用。Python 的解释器和标准扩展库的源代码和二进制格式在各个主要平台上都可以免费得到，而且可以免费分发。

由于 Python 的诸多优点，所以其是学习脚本语言者的首选。本书由浅入深地举例说明了 Python 语言的使用，并提供了许多有关 Python 的网站的信息，可以从这些网站得到更多有用的资料。本书附带的光盘提供了书中的示例脚本，以帮助读者更好地学习 Python 语言。

全书分为两大部分，第一部分为第 1 章至第 5 章，介绍了 Python 的基础知识，如 Python 的语句、模块、数据类型、函数和模块等，第二部分为第 6 章至第 12 章，通过列举大量的示例程序，详细介绍了 Python 在 Tkinter 脚本、数据库、数学/科学函数、服务器、字符串和其他数据类型、系统操作、游戏和人工智能方面的应用。作者将 Python 的 FAQ 放在本书的附录中，方便读者查阅。

本书的最大特点在于“脚本”，提供了 40 多个大小不等的示例程序。学习一种语言的初学者最需要的往往是例子。有了例子的帮助，我们可以更快地学习语言，甚至可以从网页或光盘中收集示例代码满足自己的需要。

本书由云舟工作室编译，李国、杨水超、杨作梅、闫娅男、刘洋等同志承担了主要的编译工作。由于水平有限，书中还有很多不当之处，恳请广大读者批评指正。

译者

2001 年 9 月

作者简介

Tim Altom

在 Tim Altom 成年之后的前 15 年里，他做过电子技师和工业程序员。对于他来说能够文理兼顾是必须的，所以他寻求并最终在 Indiana 大学取得了英语/新闻学的学位。最近几年在 Indianapolis，他作过技术通信员、项目经理、撰稿人以及 Simply Written 公司的副总裁和首席技术顾问。他精通各种编程语言，几乎无所不通。他用过的编程语言有 Visual Basic for Applications、汇编、C、JavaScript、Java、Boolean、Basic、Lisp 和现在的 Python。

Tim 是技术通信协会的高级成员和 Hoosier 分会的前任会长。他在 Simply Written 公司关于文档编制方法的许多会议上发表过演讲，并且写过多篇专业的文章，题目从使用“big”一词的历史到为诵读有障碍者设计文件等等。他是 Prima 出版社的《Hands On HTML》一书和其他书籍的合作者。

他已婚，有两个女儿和一个继子，生活在 Indiana 州 Indianapolis 的东北面 Lawrence 的一个舒适、独立的社区中。Lawrence 是 Fort Benjamin Harrison（一个 Base Closure 运动的受害者）的新拥有者。Simply Written 公司很乐意地制定了“再利用”计划，一定程度上说服联邦政府将 Fort Harrison 卖给社区，目前那里已成为 Lawrence 社区生活的中心。

Mitch Chapman

Mitch Chapman 从 1980 年父母给了他一台 TI-55 计算器之后，就一直在编程。他在 Ohio 州 Dayton 的 Wright State 大学获得了计算机科学的学位，然后开始工作。他曾在 Wright-Patterson AFB 维护空气动力学建模软件，并且后来帮助进行瑞士 JAS-39 Gripen 的自动飞行测试。

Mitch 在 Ohio Electronic Engravers 度过了 6 年时间，他以非凡的努力制造出了世界最先进的出版影印系统。

Mitch 已经用 C、C++、Pascal、汇编语言和 Fortran 语言开发了软件，并且还使用过其他多种语言。1996 年他发现了 Python，到目前为止，这是他最喜爱的。

Mitch 目前受雇于 Bioreason，这是一家位于 Santa Fe，NM 刚刚新建的化学信息公司，Bioreason 致力于高吞吐量药物屏幕数据分析的自动化。他们用 Python 做一切事情，从 AI 到 GUI 应用程序。

Mitch 是 IEEE 计算机协会的成员，并很自豪、幸运地成为 Python 软件组织的第六个成员。

致　　谢

首先要感谢Prima出版社的Kim Spilker，他对Linux/GNU书籍有极好的洞察力；感谢Kevin Harreld，他作为编辑做了这么多工作。

另外对本书做出贡献的还有：

Mitch Chapman

Andrew Markebo

Sam Rushing

Joe Strout

Michel Vanaken

还有那些宁愿保持匿名的人，感谢你们中的每一位。

特别要感谢 Mitch Chapman，他和我在一起深入地研究和测试脚本。祝他的队伍日益壮大。

简　介

我不是每天都能写关于一些非常有趣的东西的书。的确，技术本身通常是有趣的，常常使人着迷，偶尔甚至有很高的报酬，但是有趣常常不是等式的一部分。

然而，Python 很有趣，真的很有趣！它强大、灵活、简单，尤其是非常好用。

我是一个交流者，我非常擅长于此，靠它足以维持生活；我还是一个爱开玩笑的人，因此我写、说、讲一些技术问题才是最自然的。但是，许多技术好得使人着迷，差得令人讨厌。多数软件在其界面弹出到表面时没有正确地发出应有的光芒，这么多我在文档里称之为封闭系统的东西，被编译者藏起来。当你掀起顶盖要修车的时候，风挡玻璃刮水器控制装置的资料该是多好的东西啊？

Python 就不同，它是开放的。可以得到源代码，并且事实上它就在本书所附的光盘里。它很简单，用它写代码的人们颇能获益，并有一个很好的共同的幽默感觉。

我偶然发现 Python 是在几年前我装载了第一个 Linux 发行版的时候。在 Simply Written 公司，我们大多在 Microsoft Windows 下工作，这是一个错误（这里我要承认它，然后丢下它不管）。我们的客户大量运行 Windows，因此我们也这么做了。一到 Adobe 公司推出 Linux 平台的 FrameMaker 时，我们才立刻认真地讨论大规模的转化。

在很少的脚本语言的世界里，Python 还是一个很不成熟的选手，这是我也不得不承认的事实。Perl 是第一个而且仍是大多数 UNIX/Linux 用户所选择的脚本语言。与 Perl 相比，Python 更适合 Linux/GNU 来挑战 WenTel（对微软和英特尔结盟分享商业利益的戏称——译者注）这个大怪物。Python 比 Perl 更灵巧，更多地面向对象，更容易学。但是为数众多的用户已经学过 Perl 了，并且没有学过 Python。许多用户甚至没有见过 Python 或使用过它。然而，对那些日益被最后期限和迅速成长的需求折磨的管理员和 Webmaster 来说，Python 无疑是一个天赐之物。Python 可以减轻这些压力，需要它的人可以先快速地学习和应用 Python 的初步内容，同时他们可以逐渐地深入到 Python 的细节中去。

本书就是想帮助这些可怜的人。它有别于 Python 以前的书籍，因为它没有用任何渐进指南的方式来教你这门语言。如果你需要，其他几本好书可以做到这一点。

相反，本书介绍了仅有的 Python 骨架：它的语句、模块、数据类型等等。如果你已经非常熟悉 C 语言、数据库的术语和其他管理员类型的技术，你会觉得 Python 的大部分都比较熟悉。

然后，本书提供许多脚本。有些很短，其他的则很长、很复杂。以我的经验，一个语言的初学者最强烈需要的是例子。在可以下工夫研究和尝试的丰富的示例代码的支持下，我们大多数都可以更快地学习语言。你甚至可以从网页或光盘中收集示例代码满足自己的需要。

本书的编排

第 1 章至第 4 章是介绍性的。你将读到关于 Python 语句的内容，看到 Python 模块的列表，以及关于 Python 如何组合在一起的若干说明。然而别指望有深入的论述，我要为代码节省篇幅。

我通常使用和测试脚本的 Python 版本是 1.5.2。其他版本在你读本书的时候可能已经出来了，因此请到 www.python.org 上查看更新的信息。许多投稿的脚本是用旧版本写成的，某些情况下需要提供更新的版本。

其余的章节就是例子代码了。这涉及了简单到字符串处理之类，复杂到只有高级用户才能理解的代码。

每个例子脚本都伴有代码的分析和解释。我分解一些脚本的操作，以便你能看到操作的细节。深入内部的工作可以使你快速进入到 Python 中。

还有一个附录，是关于 Python 的一个 FAQ。你可以访问 Python 的 Web 站点（www.python.org）来获取最新的 FAQ。Python.org 的 FAQ 真是杰作，应受到赞扬。所有的 FAQ 都有如此好的组织、如此全面、如此广博。你几乎可以通过 FAQ 来学习 Python。它在所附光盘上也有 HTML 格式的文件。

更多的资料

Python 的交换中心在 python.org，下载、文档、SIG——这里应有尽有。

能提供每日帮助的主要来源在 comp.lang.python。在发表任何东西之前，先查看一下 FAQ。它在本书的光盘中，在附录中，在 python.org 上都有。列表的参与者当他们要写一个“查看 FAQ”的帖子时容易生气。正如两个 Monty Python 的人讲的非常著名的幽默一样：

“我已经告诉过你一次了！”

“不，你没有啊？”

“我肯定讲过了？”

脚本的来源

你会注意到大多数的脚本都不是我的。本书的大多数脚本都是别人贡献出来的。其中一些是它们的作者正在使用的，其他则是一些写出来张贴给别人用于取乐的很有趣的程序。

我从 Python 程序员那里要脚本是因为，无论我写多少，我的代码总是我自己的风格。其他用 Python 写的代码则更好，因此我认为，拥有来自众多有经验的程序员的各种代码会更好地帮助你。这些脚本是公用领域的，请根据你的意愿使用它们。然而，大多数作者都希望你在将代码用于商业目的的时候让他知道。在得到许可的地方，我会标明程序员的名字和组织。你也许想联系他们，说声谢谢就可以了。

Python 是自由/开放源码世界的一部分。这意味着你可以根据自己的想法来获取、装载、

运行和修补 Python。这种方法与商业软件通用的做法形成鲜明的对照，商业软件给你一个盒子，要为它付钱，然后努力去用好它。许多 Python 程序员这样喜欢 Python 是因为它不是私有的。Python 的工具很小，可以像一个产品一样扩展。Python 程序员常常像 Python 本身那样开放和可用。他们通常慷慨地提供建议和帮助，并且在这里投稿的程序员毫无例外地应该受到赞誉。

虽然脚本的作者们很有帮助，但他们并不愚蠢。他们对你拿代码所做的任何事情都不负责任。在机器上尝试代码和排除错误可都是你自己的工作。

我编辑过某些脚本使它更适合我的目的。在这种情况下，我会先展示原始的代码，然后是我的改动。一般我会尝试以简单的例子来开始脚本的每个段落，然后渐渐深入，以给你一个最好的切入点。

Python 和平台

Python 适合每一种适当流行的平台，包括 Amiga。然而，这些平台没有相应对等地“Python 化”。本书中我使用一个 WinTel 的 Python 发行版和一个 Linux 上的版本。它们是不同的。在两个平台上有些脚本不能同样地运行。例如 Shaney.py，在 WinTel 上就不会停止运行，它持续地输出文本；而在 Linux 的机器上很正常。

因此，应该要小心对待仅仅拷贝一个脚本并指望它运行良好的想法。你在自己运行的时候可能会遇到这样或那样的问题。很基础的 Python 脚本应该各处都能运行，但是当你留下尽量简明的代码并做更多的尝试，脚本的可靠性会有所改变。

总之，我宁愿在 Linux 机器上写脚本。我的发行版本是 Caldera 的，我大多在 Xemacs 20.4 上写代码。Python 模式使得它对缩进的跟踪更容易。你可以在 Xemacs 中很好地测试脚本。但是谨慎对待在使用 Tkinter 的 Xemacs 中测试脚本，因为依我的经验，如果尝试在 Xemacs 之外运行，Xemacs 会变慢甚至停止。啊，还好吗？

本书中的脚本都是在 Linux/GNU 上写成并特别针对 Linux/GNU 的，在其他地方这些脚本也许能运行得很好，也许不能。

领略一下 Python

好，如果你能跟得上我这个速度，大概你已经渴望看一些代码了。这里有一个我编辑过的由 Joe Strout 提交的脚本。

```
#! usr/bin/python
# dog.py
### 获得初始年龄
def doggie():
    try:
        age = input("Enter your age (in human years): ")
```

```
        print # print a blank line
    ### 进行一些范围检查，然后打印结果
        if age < 0:
            print "Negative age?!? I don't think so."
        elif age < 3 or age > 110:
            print "Frankly, I don't believe you."
        else:
            print "That's", age*7, "in dog years."
    ### 暂停，等待键入Return键（所以窗口不出现）
        raw_input('press Return')
    except SyntaxError:
        print "Why didn't you type anything? Try again."

if __name__=="__main__":
    doggie()
```

脚本中#号后面的任何东西都被当作注释。在上面的脚本中，另外两个#号只是为了强调；只有第一个起作用。

注意缩进的使用。Python 并不使用圆括号“()”或花括号“{}”来包括代码。它只用缩进来表示。def 表示一个函数的开始。

if、while、for 和 def 语句结尾带有一个冒号，提示解释器向下寻找和运行代码块。if 语句后还有 else 和 else if（缩写为 elif）。

这个脚本还有其他各种形式，但你学到了它的思想。你可能已经猜想出它的操作而不需要了解更多 Python 的知识。几分钟的学习，你已经能独立编写基于这个例子的脚本。所以 Python 非常简单，不是吗？

在解释器中运行这个脚本也不困难。在命令行或终端窗口的模式下键入 python，启动解释器。如果 Python 正确地安装了，就会得到如下所示的 Python 提示符，像三个尖括号：

```
>>>
```

确认脚本是 Python 可以访问到的。最简单的办法是将 dog.py 文件复制到 Python 库目录下。在我的 Linux 机器上，该是 usr/bin/lib/python1.5/。

键入如下的命令，运行脚本：

```
python dog.py
```

脚本加载然后运行。在提示符状态下重复上面的命令，就可以再运行一次。

也可以将脚本输入到 Python 会话中，如下：

```
>>>import dog
```

将回到 Python 提示符状态。dog.py 现在已经在 Python 的名字空间里了。要让它再做 doggie 的小把戏，键入：

```
>>>dog.dogie()
```

这使得解释器运行 dog.py 里的 doggie()函数。在屏幕上将看到：

```
Enter your age (in human years):
```

接下来键入你的年龄。不要试图用小于 3 或大于 110 的数来干扰脚本，它知道这一点。键入 12 然后按 Enter 键，你将得到：

```
That's 84 in dog years.
Press Return.
```

按下 Enter 键回到提示符状态。

如果还想运行这个小程序，只要再次键入 dog.doggie()然后按 Enter 键即可。

这个例子只触及了 Python 的一些皮毛，但它描绘了最基本的原理。

准备好学习更多知识了吗？来吧！

目　　录

第二部分 脚本

第一部分 “啊，你得到了什么？”

这一部分是对脚本语言 Python 的介绍，这部分没有 Python 脚本（脚本都在第 2 部分）。在第 1 部分，你将看见 Python 的语句、数据结构、模块和窗口小部件（Widget）怎样被放在一起。并且第 1 章你会读到与 Python 毫无关系的介绍，语言概述以及 Python 的细节。

第 1 章　Python 的介绍

本章要点：

- 脚本化：现在有些事情完全不同了
- 如何选择
- 在 Python 中面向对象编程：在其他东西上面再放些东西
- 如何使用本书中的脚本

Monty Python（必胜之蟒，来自一个著名电视短剧集《Monty Python's Flying Circus》的名称，该剧很荒诞、反传统——译者注）是 Python 编程语言的命名灵感，虽然这种联系不那么显而易见。除了名字之外，两者再没有任何东西是一样的。然而，这给了 Python 开发者无穷无尽的令人愉快的旁白和狡猾的影射。对 Monty Python 感到好奇的读者可参观 www.stone-dead.asn.au。

Python 语言是 Guido van Rossum 创建的。他仍然是这个语言的领袖和神父，并且是它的最终决策人。Van Rossum（当 van 在句子的前面出现时，应每位 Guido 的要求用大写字母开头）一直在编写语言已经有好多年了。Python 的起源日期回溯到 1989 年，虽然 van Rossum 直到 1991 年才发布了 Python。从那以后，它被稍微重组了一下，但它基本上仍然是他所创建的相同的语言。Van Rossum 的主页是 www.python.org/~guido/。在给他发送 e-mail 之前请阅读整个主页，因为这样做可以在你询问之前也许就可以找到问题的答案。