

Modern Python Cookbook

Python

经典实例

[美] 史蒂文·F. 洛特◎著 闫兵◎译

解决实际场景中的具体问题, 全面了解Python语言特性



中国工信出版集团



人民邮电出版社

POSTS & TELECOM PRESS

Modern Python Cookbook

Python

经典实例

[美] 史蒂文·F.洛特◎著 闫兵◎译

人民邮电出版社
北 京

图书在版编目 (CIP) 数据

Python经典实例 / (美) 史蒂文·F. 洛特
(Steven F. Lott) 著 ; 闫兵译. — 北京 : 人民邮电出版社, 2019.3

(图灵程序设计丛书)
ISBN 978-7-115-50717-4

I. ①P… II. ①史… ②闫… III. ①软件工具—程序设计 IV. ①TP311.561

中国版本图书馆CIP数据核字(2019)第022886号

内 容 提 要

本书是 Python 经典实例解析, 采用基于实例的方法编写, 每个实例都会解决具体的问题和难题。主要内容有: 数字、字符串和元组, 语句与语法, 函数定义, 列表、集、字典, 用户输入和输出等内置数据结构, 类和对象, 函数式和反应式编程, Web 服务, 等等。

本书适合 Python 初中级程序员阅读。

-
- ◆ 著 [美] 史蒂文·F. 洛特
 - 译 闫 兵
 - 责任编辑 张海艳
 - 责任印制 周昇亮
 - ◆ 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>
 - 三河市祥达印刷包装有限公司印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 32.75
 - 字数: 774千字
 - 印数: 1—2 500册
 - 2019年3月第1版
 - 2019年3月河北第1次印刷
 - 著作权合同登记号 图字: 01-2017-8644号

定价: 139.00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: 京东工商广登字 20170147 号

前言

Python 是全球的开发人员、工程师、数据科学家和编程爱好者的首选语言。它是杰出的脚本语言，可以为你的应用程序注入动力，提供出色的速度、安全性和可扩展性。通过一系列简单的实例剖析 Python，你可以在特定的情境中深入了解其具体的语言特性。明确的情境有助于理解语言或标准库的特性。

本书采用基于实例的方法编写，每个实例都会解决具体的问题和难题。

本书内容

第 1 章主要讨论不同类型的数字、字符串、元组和 Python 的基本内置类型，以及如何充分利用 Unicode 字符集的强大功能。

第 2 章首先介绍创建脚本文件的基础知识，然后讨论一些复杂的语句，包括 `if`、`while`、`for`、`try`、`with` 和 `raise`。

第 3 章主要介绍一些定义函数的技巧以及 Python 3.5 的 `typing` 模块，并演示如何利用 `typing` 模块为函数创建更正式的注释。

第 4 章概述 Python 拥有的各种数据结构以及它们解决了哪些问题。这一章将详细介绍列表、字典和集合，以及一些与 Python 处理对象引用方式相关的高级主题。

第 5 章解释如何利用 `print()` 函数的多个功能，并介绍其他用于提供用户输入的函数。

第 6 章将创建一些实现大量统计公式的类。

第 7 章进一步深入探索 Python 的类，并结合一些已经介绍过的功能来创建更复杂的对象。

第 8 章介绍如何编写简洁明了的数据转换函数。你还将了解反应式编程的概念，也就是说，当输入变得可用或者发生改变时，执行处理规则。

第 9 章主要介绍如何处理不同的文件格式，如 JSON、XML 和 HTML。

第 10 章介绍一些可以使用 Python 内置库和数据结构实现的基本统计计算，并讨论相关性、随机性和零假设等话题。

第 11 章详细说明 Python 使用的不同的测试框架。

第 12 章介绍一系列创建 RESTful Web 服务以及提供静态内容或动态内容的实例。

第 13 章针对更大规模、更复杂的复合应用程序介绍一些设计方法，并分析复合应用程序可能出现的复杂性，以及需要集中的一些功能，如命令行解析。

阅读须知

运行本书的示例代码仅需要一台安装有较新 Python 版本的计算机。虽然示例代码都使用 Python 3 编写，但是仅需简单修改就可适用于 Python 2。

读者对象

本书适合 Web 开发人员、程序员、工程师和大数据从业者阅读。如果你是初学者，本书将带领你入门。如果你经验丰富，本书将扩展你的知识储备。了解程序设计的基础知识有助于阅读本书。

排版约定

本书使用多种不同的文本样式来区分不同种类的信息。下面是各类格式的示例及其所表示的含义。

正文中的代码、数据库表名、用户输入等采用如下样式：“我们可以使用 `include` 指令包含其他上下文。”

代码段的样式如下所示：

```
if distance is None:
    distance = rate * time
elif rate is None:
    rate = distance / time
elif time is None:
    time = distance / rate
```

命令行输入或输出采用如下样式：

```
>>> circumference_diameter_ratio = 355/113
>>> target_color_name = 'FireBrick'
>>> target_color_rgb = (178, 34, 34)
```

新术语和重点强调的文字以黑体字表示。



此图标表示警告或重要的注意事项。



此图标表示提示和小窍门。

读者反馈

欢迎读者提交反馈，内容可以是你对本书的看法，喜欢哪些部分，不喜欢哪些部分。这些反馈至关重要，有助于我们创作出真正对读者有所裨益的内容。

如果反馈一般性信息，可以发送电子邮件到 feedback@packtpub.com，并在邮件标题中注明书名。如果你擅长某个主题，并有兴趣写本书或者为某本书做出贡献，请访问 www.packtpub.com/authors 参阅我们的作者指南。

客户支持

现在你已经拥有了这本由 Packt 出版的图书，为了让你的付出得到最大的回报，本书还为你提供了其他诸多方面的服务。

下载示例代码

你可以使用自己的账户从 <http://www.packtpub.com> 下载本书的示例代码文件。^①如果你是通过其他方式购买本书的，可以访问 <http://www.PacktPub.com/support> 并注册，我们将通过邮件的方式发送给你。

可以通过以下步骤下载代码文件：

- (1) 使用你的电子邮箱和密码登录或者注册我们的网站；
- (2) 把鼠标悬停在网站上方的 SUPPORT 选项卡上；
- (3) 点击 Code Downloads & Errata；
- (4) 在搜索框中输入书名；
- (5) 选择你要下载代码文件的图书；
- (6) 从下拉菜单中选择你购书的途径；
- (7) 点击 Code Download。

文件下载成功后，请确保使用以下最新版本的软件进行解压缩：

- ☐ WinRAR / 7-Zip (Windows)
- ☐ Zipeg / iZip / UnRarX (Mac)
- ☐ 7-Zip / PeaZip (Linux)

本书的代码包也托管在 GitHub 上，网址为 <https://github.com/PacktPublishing/Modern-Python-Cookbook>。Packt 的其他图书和视频的代码包的存放网址为 <https://github.com/PacktPublishing/>。赶快去看看吧！

勘误

虽然我们已经竭尽全力保证本书内容的准确性，但错误仍在所难免。如果你发现了本书的任何错误，无论是文本错误还是代码错误，都请报告给我们，对此我们感激不尽。这样不仅能消除其他读者的疑虑，也有助于改进本书的后续版本。如果需要提交勘误，请访问 <http://www.packtpub.com/support>，选择相应的书名，单击 errata submission form 链接，登记你的勘误详情。^②一旦勘误得到确认，我们将接受你的提交，同时勘误内容也将上传到我们的网站，或者添加到对应书目勘误区的现有勘误表中。

^① 读者也可以到图灵社区本书页面下载代码文件，网址是 <http://ituring.cn/book/1938>。——编者注

^② 本书中文版勘误请到 <http://ituring.cn/book/1938> 查看和提交。——编者注

要查询之前提交的勘误，请访问 <http://www.packtpub.com/support>，并在搜索框中输入书名，所需信息就会显示在勘误区。

侵权

所有正版内容在互联网上都面临的一个问题就是侵权。Packt 非常重视对版权和授权的保护。如果你在网发现 Packt 图书的任何形式的盗版，请立即为我们提供网址或网站名称，以便我们采取补救措施。

如果发现可疑盗版材料，请通过 copyright@packtpub.com 联系我们。

非常感谢你帮助我们保护作者权益，我们将竭诚为读者提供有价值的内容。

其他问题

如果你对本书某方面存在疑问，请通过 questions@packtpub.com 联系我们，我们将尽力解决。

电子书

如需购买本书电子版，请扫描以下二维码。



目 录

第 1 章 数字、字符串和元组..... 1	2.10 避免 <code>except</code> :子句带来的潜在问题..... 64
1.1 引言..... 1	2.11 使用 <code>raise from</code> 语句链接异常..... 65
1.2 创建有意义的名称和使用变量..... 2	2.12 使用 <code>with</code> 语句管理上下文..... 67
1.3 使用大整数和小整数..... 5	第 3 章 函数定义..... 70
1.4 在浮点数、小数和分数之间选择..... 8	3.1 引言..... 70
1.5 在真除法和 <code>floor</code> 除法之间选择..... 13	3.2 使用可选参数设计函数..... 70
1.6 重写不可变的字符串..... 15	3.3 使用灵活的关键字参数..... 75
1.7 使用正则表达式解析字符串..... 19	3.4 使用 <code>*</code> 分隔符强制使用关键字参数..... 77
1.8 使用 <code>"template".format()</code> 创建复杂的字符串..... 22	3.5 编写显式的函数参数类型..... 80
1.9 通过字符列表创建复杂的字符串..... 25	3.6 基于偏函数选择参数顺序..... 84
1.10 使用键盘上没有的 Unicode 字符..... 27	3.7 使用 RST 标记编写清晰的文档字符串..... 87
1.11 编码字符串——创建 ASCII 和 UTF-8 字节..... 29	3.8 围绕 Python 栈限制设计递归函数..... 91
1.12 解码字节——如何根据字节获得正确的字符..... 31	3.9 根据脚本/库转换规则编写可重用脚本..... 94
1.13 使用元组..... 33	第 4 章 内置数据结构——列表、集、字典..... 98
第 2 章 语句与语法..... 36	4.1 引言..... 98
2.1 引言..... 36	4.2 选择数据结构..... 99
2.2 编写 Python 脚本和模块文件——语法基础..... 37	4.3 构建列表——字面量、 <code>append()</code> 和解析式..... 102
2.3 编写长行代码..... 40	4.4 切片和分割列表..... 106
2.4 添加描述和文档..... 44	4.5 从列表中删除元素—— <code>del</code> 语句、 <code>remove()</code> 、 <code>pop()</code> 和 <code>filter()</code> 109
2.5 在文档字符串中编写 RST 标记..... 48	4.6 反转列表的副本..... 114
2.6 设计复杂的 <code>if...elif</code> 链..... 51	4.7 使用 <code>set</code> 方法和运算符..... 116
2.7 设计正确终止的 <code>while</code> 语句..... 54	4.8 从集中移除元素—— <code>remove()</code> 、 <code>pop()</code> 和差集..... 120
2.8 避免 <code>break</code> 语句带来的潜在问题..... 58	4.9 创建字典——插入和更新..... 122
2.9 利用异常匹配规则..... 61	

4.10	从字典中移除元素——pop() 方法和 del 语句	126
4.11	控制字典键的顺序	128
4.12	处理 doctest 示例中的字典和集	130
4.13	理解变量、引用和赋值	132
4.14	制作对象的浅副本和深副本	134
4.15	避免可变默认值作为函数参数	137
第 5 章	用户输入和输出	141
5.1	引言	141
5.2	使用 print() 函数的功能	141
5.3	使用 input() 和 getpass() 收集用户输入	145
5.4	使用 "format".format_map(vars()) 进行调试	150
5.5	使用 argparse 模块获取命令行输入	151
5.6	使用 cmd 模块创建命令行应用程序	156
5.7	使用操作系统环境设置	161
第 6 章	类和对象的基础知识	165
6.1	引言	165
6.2	使用类封装数据和操作	166
6.3	设计操作类	169
6.4	设计数据类	174
6.5	使用 __slots__ 优化对象	177
6.6	使用更复杂的集合	180
6.7	扩展集合——统计数据的列表	183
6.8	使用特性计算惰性属性	186
6.9	使用可设置的特性更新及早属性	190
第 7 章	高级类设计	195
7.1	引言	195
7.2	在继承和扩展之间选择——is-a 问题	195
7.3	通过多重继承分离关注点	201
7.4	利用 Python 的鸭子类型	205
7.5	管理全局单例对象	208
7.6	使用更复杂的结构——列表映射	212
7.7	创建具有可排序对象的类	214
7.8	定义有序集合	218
7.9	从映射列表中删除元素	223
第 8 章	函数式编程和反应式编程	228
8.1	引言	228
8.2	使用 yield 语句编写生成器函数	229
8.3	使用生成器表达式栈	234
8.4	将转换应用于集合	241
8.5	选择子集——三种过滤方式	244
8.6	汇总集合——如何归约	248
8.7	组合映射和归约转换	252
8.8	实现 there exists 处理	257
8.9	创建偏函数	260
8.10	使用不可变数据结构简化复杂算法	265
8.11	使用 yield from 语句编写递归生成器函数	269
第 9 章	输入/输出、物理格式和逻辑布局	274
9.1	引言	274
9.2	使用 pathlib 模块处理文件名	275
9.3	使用上下文管理器读取和写入文件	281
9.4	替换文件，同时保留以前的版本	284
9.5	使用 CSV 模块读取带分隔符的文件	287
9.6	使用正则表达式读取复杂格式	291
9.7	读取 JSON 文档	295
9.8	读取 XML 文档	301
9.9	读取 HTML 文档	305
9.10	将 CSV 模块的 DictReader 更新为 namedtuple 读取器	310
9.11	将 CSV 模块的 DictReader 更新为 namespace 读取器	314
9.12	使用多个上下文读取和写入文件	317

第 10 章 统计编程和线性回归	322	12.3 使用 Flask 框架实现 RESTful API	408
10.1 引言	322	12.4 解析请求中的查询字符串	414
10.2 使用内置统计库	322	12.5 使用 urllib 发送 REST 请求	418
10.3 计算 Counter 对象中值的平均值	329	12.6 解析 URL 路径	424
10.4 计算相关系数	332	12.7 解析 JSON 请求	433
10.5 计算回归参数	336	12.8 实施 Web 服务认证	441
10.6 计算自相关	339	第 13 章 应用程序集成	455
10.7 确认数据是随机的——零假设	344	13.1 引言	455
10.8 查找异常值	348	13.2 查找配置文件	456
10.9 通过一次遍历分析多个变量	353	13.3 使用 YAML 编写配置文件	462
第 11 章 测试	359	13.4 使用 Python 赋值语句编写配置 文件	468
11.1 引言	359	13.5 使用 Python 类定义编写配置文件	470
11.2 使用文档字符串进行测试	360	13.6 设计可组合的脚本	475
11.3 测试抛出异常的函数	365	13.7 使用 logging 模块监控和审计 输出	481
11.4 处理常见的 doctest 问题	368	13.8 将两个应用程序组合为一个复合 应用程序	488
11.5 创建单独的测试模块和包	372	13.9 使用命令设计模式组合多个应用 程序	494
11.6 组合 unittest 测试和 doctest 测试	378	13.10 管理复合应用程序中的参数和 配置	497
11.7 涉及日期或时间的测试	381	13.11 包装和组合 CLI 应用程序	501
11.8 涉及随机性的测试	385	13.12 包装程序并检查输出	506
11.9 模拟外部资源	388	13.13 控制复杂的步骤序列	509
第 12 章 Web 服务	398		
12.1 引言	398		
12.2 使用 WSGI 实现 Web 服务	400		

第1章

数字、字符串和元组



本章将通过以下实例介绍 Python 的基本数据类型。

- ☐ 创建有意义的名称和使用变量
- ☐ 使用大整数和小整数
- ☐ 在浮点数、小数和分数之间选择
- ☐ 在真除法和 floor 除法^①之间选择
- ☐ 重写不可变的字符串
- ☐ 使用正则表达式解析字符串
- ☐ 使用 "template".format() 创建复杂的字符串
- ☐ 通过字符列表创建复杂的字符串
- ☐ 使用键盘上没有的 Unicode 字符
- ☐ 编码字符串——创建 ASCII 和 UTF-8 字节
- ☐ 解码字节——如何根据字节获得正确的字符
- ☐ 使用元组

1.1 引言

本章将介绍 Python 对象的一些核心类型。我们将讨论不同类型的数字以及字符串和元组的使用方法。它们是最简单的 Python 数据类型，因此会首先介绍。后面的章节将探讨数据集合。

本章大部分实例假定你对 Python 3 有基本的了解，主要介绍如何使用 Python 提供的数字、字符串和元组等基本内置类型。Python 拥有丰富的数字类型和两个除法运算符，因此需要仔细研究它们之间的区别。

使用字符串时，有几个常见的操作非常重要。本章将探讨操作系统文件所使用的字节和 Python 所使用的字符串之间的区别，以及如何充分利用 Unicode 字符集的强大功能。

本章将在 Python 的交互式解释器中演示实例，这种模式有时也称为 REPL (read-eval-print loop, “读取-求值-输出”循环)。后面的章节将仔细研究脚本文件的编写。这样做的目的是鼓励交互式探索，因为它是学习语言的极佳方式。

^① floor 除法就是向下取整除法。向上取整除法是 ceiling。——译者注

1.2 创建有意义的名称和使用变量

如何确保程序易于理解呢？要编写富有表现力的代码，一个核心要素就是使用有意义的名称。但什么是有意义的呢？在本实例中，我们将回顾一些创建有意义的 Python 名称的通用规则。

我们还将介绍 Python 的一些不同形式的赋值语句，如用同一条语句为多个变量赋值。

1.2.1 准备工作

创建名称的核心问题是：被命名的是什么？对于软件，我们需要一个描述被命名对象的名称。显然，像 `x` 这样的名称不是很有描述性，它似乎并不指向实际的事物。

模糊的非描述性名称在一些程序设计中很常见，令人十分痛苦。当使用它们时，无助于其他人理解我们的程序。描述性名称则一目了然。

在命名时，区分解决方案域和问题域（真正想要解决的问题）也很重要。解决方案域包括 Python、操作系统和互联网的技术细节。不需要深入的解释，任何人在阅读代码时都可以看到解决方案。然而，问题域可能因技术细节而变得模糊。我们的任务是使问题清晰可见，而精心挑选的名称将对此有所帮助。

1.2.2 实战演练

首先看看如何命名，然后再学习如何为对象分配名称。

1. 明智地选择名称

在纯技术层面上，Python 名称必须以字母开头。它们可以包括任意数量的字母、数字和下划线。因为 Python 3 基于 Unicode，所以字母不限于拉丁字母。虽然通常使用拉丁字母 A~Z，但这不是必须遵循的规定。

当创建一个描述性变量时，我们需要创建既具体又能表达程序中事物之间关系的名称。一种广泛使用的命名技巧就是创建“从特殊到一般”这种风格的长名称。

选择名称的步骤如下。

(1) 名称的最后一部分是事物的广义概要。有时候，仅此一部分就能满足命名的需要，靠上下文提供其余的信息。稍后将介绍一些典型的广义概要的类别。

(2) 在应用程序或问题域周围使用前缀限定名称。

(3) 如有必要，使用更精确和专用的前缀，以阐明它与其他类、模块、包、函数和其他对象的区别。对前缀有疑问时，回想一下域名的工作原理。例如，`mail.google.com` 这个名称表明了从特殊到一般的三个级别。三个级别的命名并不罕见，我们经常采用这种命名方法。

(4) 根据在 Python 中的使用方法来命名。需要命名的事物有三大类，如下所示。

□ 类：类的名称能够概述类中的所有对象。这些名称通常使用大驼峰命名法（Capitalized-CamelCase）。类名的第一个字母大写，强调它是一个类，而不是类的实例。类通常是一个通用的概念，很少用于描述有形的事物。

❑ **对象**：对象的名称通常使用蛇底命名法（snake_case）。名称全部小写，单词之间使用多个下划线连接。在 Python 中，一切皆是对象，包括变量、函数、模块、包、参数、对象的属性、类的方法等。

❑ **脚本和模块文件**：这些文件是 Python 看到的真正的操作系统资源。因此，文件名应遵循 Python 对象的约定，使用字母、下划线并以 .py 扩展名结尾。单从技术上说，你可天马行空地设置文件名。但是，不遵循 Python 规则的文件名可能难以用作模块或包的名称。

如何选择名称中广义类别的那一部分呢？通用类别取决于讨论的是事物还是事物的属性。虽然世界上有很多事物，但我们仍然可以创建一些有用的广义分类，例如文档、企业、地点、程序、产品、过程、人、资产、规则、条件、植物、动物、矿物等。

然后可以用修饰语来限定这些名称：

```
FinalStatusDocument
ReceivedInventoryItemName
```

第一个示例是 Document 类，我们通过添加一个前缀对其进行了略微的限定，即 StatusDocument，又通过将其命名为 FinalStatusDocument 来进一步限定。第二个示例是 Name 类，我们通过详细说明它是一个 ReceivedInventoryItemName 来对其进行限定。该示例需要一个 4 个级别的名称来阐明。

对象通常具有特性（property）或者属性（attribute）。它们应当是完整名称的一部分，可以根据其表示的信息类型进行分解，如数量、代码、标识符、名称、文本、日期、时间、日期时间、图片、视频、声音、图形、值、速率、百分比、尺寸等。

命名的思路就是把狭义、详细的描述放在最前面，把宽泛的信息放在最后：

```
measured_height_value
estimated_weight_value
scheduled_delivery_date
location_code
```

在第一个示例中，height 限定了更一般的表示术语 value，而 measured_height_value 做了进一步限定。通过这个名称，可以思考一下其他与 height 相关的变体。类似的思想也适用于 weight_value、delivery_date 和 location_code。这些名称都有一个或者两个限定前缀。

需要避免的情况

切勿使用经过编码的前缀或后缀去描述详细的技术信息。不要使用 f_measured_height_value 这样的名称，其中 f 可能指的是浮点数。这种命名方法通常被称为匈牙利命名法（Hungarian Notation）。像 measured_height_value 这样的变量可以是任意数字类型，Python 会做所有必要的转换。技术性修饰对于代码阅读者并没有多大帮助，因为类型说明可能造成误导甚至错误。

不要浪费太多的精力使名称看起来属于哪个类别。不要使用 SpadesCardSuit、ClubsCardSuit 这样的名称。Python 有许多种命名空间，包括包、模块和类，命名空间对象会把相关的名称收集起来。如果将这些名称添加到 CardSuit 类中，就可以使用 CardSuit.Spades，以类作为命名空间来区分其他相似的名称。



2. 为对象分配名称

Python 没有使用静态变量定义。当把名称分配给对象时, 就会创建变量。把对象看作处理过程的核心非常重要, 变量有点像标识对象的标签。使用基本赋值语句的方法如下。

(1) 创建对象。在许多示例中, 对象以字面量的形式创建。我们使用 355 或 113 作为 Python 中整数对象的字面量表示, 也可以使用 FireBrick 表示字符串, 或使用 (178, 34, 34) 表示元组。

(2) 编写如下类型的语句: 变量 = 对象。例如:

```
>>> circumference_diameter_ratio = 355/113
>>> target_color_name = 'FireBrick'
>>> target_color_rgb = (178, 34, 34)
```

我们创建了一些对象并把它们赋值给了变量。第一个对象是数值计算的结果, 接下来的两个对象是简单的字面量。对象通常由包含函数或类的表达式创建。

上面的基本赋值语句并不是唯一的赋值方式, 还可以使用链式赋值的方式, 将一个对象分配给多个变量, 例如:

```
>>> target_color_name = first_color_name = 'FireBrick'
```

上例为同一个字符串对象创建了两个名称。可以通过检查 Python 使用的内部 ID 值来确认这两个对象是否为同一个对象:

```
>>> id(target_color_name) == id(first_color_name)
True
```

结果表明, 这两个对象的内部 ID 值是相同的。



相等测试使用 ==, 简单赋值使用 =。

随后介绍数字和集合时将会说明结合运算符进行赋值的方法。例如:

```
>>> total_count = 0
>>> total_count += 5
>>> total_count += 6
>>> total_count
11
```

我们通过运算符进行了增量赋值。total_count += 5 与 total_count = total_count + 5 是等价的。增量赋值的优点在于简化了表达式。

1.2.3 工作原理

本实例创建名称的方法遵循如下模式: 狭义的、更具体的限定符放在前面, 更宽泛的、不太特定的类别放在最后。这种方法遵守用于域名和电子邮件地址的通用约定。

例如, 域名 mail.google.com 包括特定的服务、更通用的企业和最后的非常通用的域, 这遵循了从窄到宽的原则。

又如, service@packtpub.com 以具体的用户名开始, 然后是更通用的企业, 最后是非常通用的域。

甚至用户名（PacktPub）也是一个具有两部分的名称，包括限定的企业名称（Packt），以及更广泛的行业 [Pub, “Publishing”（出版）的简写，而不是“Public House”（酒吧）的简写]。

赋值语句是为对象命名的唯一途径。通过前面的实例，我们注意到，同一个底层对象可以有两个名称，但现在还不清楚这种特性有什么用处。第 4 章将介绍为一个对象分配多个名称的一些有趣后果。

1.2.4 补充内容

我们将在所有实例中使用描述性名称。



没有遵循这种模式的现有软件应当保持现状。一般而言，最好与遗留软件保持一致，而不是强加新规则，即使新规则更好。

几乎每个示例都涉及变量赋值。变量赋值是有状态的面向对象编程的核心。

第 6 章将讨论类和类名，第 13 章将讨论模块。

1.2.5 延伸阅读

描述性命名是一个正在研讨的主题，涉及两个方面——语法和语义。Python 语法的设想起始于著名的 PEP-8（Python Enhancement Proposal number 8）。PEP-8 建议使用 CamelCase 和 snake_case 命名风格。

此外，务必进行以下操作：

```
>>> import this
```

这有助于领悟 Python 的设计哲学。



有关语义的信息，请参阅遗留的 UDEF 和 NIEM 命名和设计规则标准（<http://www.opengroup.org/udefinfo/AboutTheUDEF.pdf>）。有关元数据和命名的详细信息，请参阅 ISO11179（https://en.wikipedia.org/wiki/ISO/IEC_11179）。

1.3 使用大整数和小整数

许多编程语言区分整数、字节和长整数，有些编程语言还存在有符号整数和无符号整数的区别。如何将这些概念与 Python 联系起来呢？

答案是“不需要”。Python 以统一的方式处理所有类型的整数。对于 Python，从几个字节到数百位的巨大数字，都是整数。

1.3.1 准备工作

假设我们需要计算一些非常大的数字，例如，计算一副 52 张的扑克牌的排列数。 $52! = 52 \times 51 \times 50 \times \dots \times 2 \times 1$ ，这是一个非常大的数字。可以在 Python 中实现这个运算吗？

1.3.2 实战演练

别担心！Python 表现得好像有一个通用的整数类型，涵盖了所有整数，从几个字节到填满所有内存的整数。正确使用整数的步骤如下。

- (1) 写下你需要的数字，比如一些小数字：355，113。实际上，数字的大小没有上限。
- (2) 创建一个非常小的值——单个字节，如下所示：

```
>>> 2
2
```

或者使用十六进制：

```
>>> 0xff
255
```

后面的实例中将讨论只含有一个值的字节序列：

```
>>> b'\xfe'
b'\xfe'
```

严格说来，这不是一个整数。它有一个前缀 `b'`，这表明它是一个一字节序列（1-byte sequence）。

- (3) 通过计算创建一个很大的数字。例如：

```
>>> 2 ** 2048
323...656
```

该数字有 617 个数位，这里并没有完全显示。

1.3.3 工作原理

Python 内部使用两种数字，两者之间的转换是无缝且自动的。

对于较小的数字，Python 通常使用 4 字节或 8 字节的整数值。细节隐藏在 CPython 的内核中，并依赖于构建 Python 的 C 编译器。

对于超出 `sys.maxsize` 的较大数字，Python 将其切换到大整数——数字（digit）序列。在这种情况下，一位数字通常意味着 30 位（bit）的值。

一副 52 张的扑克牌有多少种排列方法？答案是 $52! \approx 8 \times 10^{67}$ 。我们将使用 `math` 模块的 `factorial` 函数计算这个大整数，如下所示：

```
>>> import math
>>> math.factorial(52)
80658175170943878571660636856403766975289505440883277824000000000000
```

这些巨大的数字工作得非常完美！

计算 $52!$ 的第一部分（从 $52 \times 51 \times 50 \times \dots$ 一直到约 42）可以完全使用较小的整数来执行。在此之后，其余的计算必须切换到大整数。我们看不到切换过程，只能看到结果。

通过下面的示例可以了解整数内部的一些细节。

```
>>> import sys
>>> import math
>>> math.log(sys.maxsize, 2)
```



```
63.0
>>> sys.int_info
sys.int_info(bits_per_digit = 30, sizeof_digit = 4)
```

`sys.maxsize` 的值是小整数中的最大值。我们通过计算以 2 为底的对数来说明这个数字需要多少位。

通过计算可知, Python 使用 63 位值来表示小整数。小整数的范围是从 -2^{63} 到 $2^{63}-1$ 。在此范围之外, 使用大整数。

通过 `sys.int_info` 的值可知, 大整数是使用 30 位的数字序列, 每个数字占用 4 字节。

像 52! 这样比较大的值, 由 8 个上述 30 位的数字组成。一个数字需要 30 位来表示可能有些令人困惑。以用 10 个符号表示十进制 (base 10) 的数字为例, 我们需要 $2^{*}30$ 个不同的符号来表示这些大数字的每位数。

涉及多个大整数值的计算可能会消耗相当大的内存空间。小数字怎么办呢? Python 如何跟踪大量的小数字, 如 1 和 0?

对于常用的数字 (-5 到 256), Python 实际上创建了一个私有的对象池来优化内存管理。你可以在检查整数对象的 `id()` 值时得到验证。

```
>>> id(1)
4297537952
>>> id(2)
4297537984
>>> a = 1 + 1
>>> id(a)
4297537984
```

我们显示了整数 1 和整数 2 的内部 `id`。当计算 `a` 的值时, 结果对象与对象池中的整数 2 对象是同一个对象。

当你练习这个示例时, `id()` 值可能跟示例不同。但是, 在每次使用值 2 时, 将使用相同的对象。在我的笔记本电脑上, 对象 2 的 `id` 等于 4297537984。这种机制避免了在内存里大量存放对象 2 的副本。

这里有个小技巧, 可以看出一个数字到底有多大。

```
>>> len(str(2 ** 2048))
617
```

通过一个计算得到的数字创建一个字符串, 然后查询字符串的长度。结果表明, 这个数字有 617 个数位。

1.3.4 补充知识

Python 提供了一组丰富的算术运算符: `+`、`-`、`*`、`/`、`//`、`%` 和 `**`。`/` 和 `//` 用于除法, 1.5 节将讨论这些运算符。`**` 将执行幂运算。

对于位处理, 还有其他一些运算符, 比如 `&`、`^`、`|`、`<<` 和 `>>`。这些运算符在整数的内部二进制表示上逐位操作。它们分别计算二进制与、二进制异或、二进制或、左移和右移。