O'REILLY®

TURING 图灵程序设计丛书

流畅的 Python

Fluent Python

PSF研究员、知名PyCon演讲者心血之作

全面深入，对Python语言关键特性剖析到位

[巴西] Luciano Ramalho 著

安道 吴珂 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

流畅的Python

Fluent Python

Clear, Concise, and Effective Programming

[巴西] Luciano Ramalho 著
安道 吴珂 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北 京

图书在版编目（CIP）数据

流畅的Python / （巴西）卢西亚诺·拉马略（Luciano Ramalho）著 ; 安道, 吴珂译. -- 北京 : 人民邮电出版社, 2017.5
（图灵程序设计丛书）
ISBN 978-7-115-45415-7

Ⅰ. ①流… Ⅱ. ①卢… ②安… ③吴… 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(2017)第061279号

内 容 提 要

本书致力于帮助 Python 开发人员挖掘这门语言及相关程序库的优秀特性，避免重复劳动，同时写出简洁、流畅、易读、易维护，并且具有地道 Python 风格的代码。本书尤其深入探讨了 Python 语言的高级用法，涵盖数据结构、Python 风格的对象、并行与并发，以及元编程等不同的方面。

本书适合中高级 Python 软件开发人员阅读参考。

◆ 著 [巴西] Luciano Ramalho
译 安 道 吴 珂
责任编辑 朱 巍
执行编辑 温 雪 黄志斌
责任印制 彭志环

◆ 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
北京市昌平百善印刷厂印刷

◆ 开本：800×1000 1/16
印张：39.25
字数：927千字 2017年5月第1版
印数：1－5 000册 2017年5月北京第1次印刷
著作权合同登记号 图字：01-2015-7561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：京东工商广字第 8052 号

版权声明

O'Reilly Media, Inc.介绍

O'Reilly Media通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自1978年开始，O'Reilly一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了*Make*杂志，从而成为DIY革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly现在还将先锋专家的知识传递给普通的计算机用户。无论是通过图书出版、在线服务或者面授课程，每一项O'Reilly的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar博客有口皆碑。”

——*Wired*

“O'Reilly凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本O'Reilly的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照Yogi Berra的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

致 Marta，用我全心全意的爱。

前言

> 要不这样吧，如果编程语言里有个地方你弄不明白，而正好又有个人用了这个功能，那就开枪把他打死。这比学习新特性要容易些，然后过不了多久，那些活下来的程序员就会开始用 0.9.6 版的 Python，而且他们只需要使用这个版本中易于理解的那一小部分就好了（眨眼）。[1]
>
> —— Tim Peters
> 传奇的核心开发者，“Python 之禅”作者

Python 官方教程（https://docs.python.org/3/tutorial/）的开头是这样写的：“Python 是一门既容易上手又强大的编程语言。”这句话本身并无大碍，但需要注意的是，正因为它既好学又好用，所以很多 Python 程序员只用到了其强大功能的一小部分。

只需要几个小时，经验丰富的程序员就能学会用 Python 写出实用的程序。然而随着这最初高产的几个小时变成数周甚至数月，在那些先入为主的编程语言的影响下，开发者们会慢慢地写出带着“口音”的 Python 代码。即便 Python 是你的初恋，也难逃此命运。因为在学校里，抑或是那些入门书上，教授者往往会有意避免只跟语言本身相关的特性。

另外，向那些已在其他语言领域里有了丰富经验的程序员介绍 Python 的时候，我还发现了一个问题：人们总是倾向于寻求自己熟悉的东西。受到其他语言的影响，你大概能猜到 Python 会支持正则表达式，然后就会去查阅文档。但是如果你从来没见过元组拆包（tuple unpacking），也没听过描述符（descriptor）这个概念，那么估计你也不会特地去搜索它们，然后就永远失去了使用这些 Python 独有的特性的机会。这也是本书试图解决的一个问题。

这本书并不是一本完备的 Python 使用手册，而是会强调 Python 作为编程语言独有的特性，这些特性或者是只有 Python 才具备的，或者是在其他大众语言里很少见的。Python 语言核心以及它的一些库会是本书的重点。尽管 Python 的包索引现在已经有 6 万多个库了，而且其中很多都异常实用，但是我几乎不会提到 Python 标准库以外的包。

注 1：给 comp.lang.python Usenet 小组的留言，2002 年 12 月 23 日，“Acrimony in c.l.p”（https://mail.python.org/pipermail/python-list/2002-December/147293.html）。

目标读者

本书的目标读者是那些正在使用 Python，又想熟悉 Python 3 的程序员。如果你懂 Python 2，但是想迁移到 Python 3.4 或者更新的版本，也没问题。在写这本书的时候，大多数专业 Python 程序员用的还是 Python 2，因此如果书中出现来自 Python 3 的特性，读者可能会感到陌生，我也会特别地做出解释。

然而，本书的主要目的是为了充分地展现 Python 3.4 的魅力，因此我不会一字一句地说明如何让本书的代码在旧版本里正常运行。本书中的大多数例子稍做修改（甚至不用修改）就可以在 Python 2.7 里面跑起来，但是有些例子，如果追求向下兼容，就会需要大量的重写。

话虽如此，我还是认为，即便你无法从 Python 2.7 里脱身，这本书也会对你很有帮助，因为 Pyhon 语言的核心概念是不会变的。Python 3 也不是一门全新的语言，大多数的改动花一下午大概就能适应，官方文档里“Python 3.0 的新特性”一节（https://docs.python.org/3.0/whatsnew/3.0.html）就是很好的切入点。固然，自 2009 年发布以来，Python 3.0 也在变化，但是这些变化比起 Python 3.0 和 Python 2.0 之间的区别，并没有那么重要。

如果你尚不清楚自己对 Python 的熟悉程度能否跟得上本书的内容，建议你回头看看 Python 的官方教程。注意，除非是跟 Python 3 的新特性有关，教程里的其他内容本书不会重复。

非目标读者

如果你才刚刚开始学 Python，本书的内容可能会显得有些“超纲”。比难懂更糟的是，如果在学习 Python 的过程中过早接触本书的内容，你可能会误以为所有的 Python 代码都应该利用特殊方法和元编程（metaprogramming）技巧。我们知道，不成熟的抽象和过早的优化一样，都会坏事。

本书的结构

如果你是本书的目标读者，那你应该可以从本书的任意一章开始阅读，但是如果按照我写作时的构思来的话，本书一共分为六个独立的部分，每个部分内的章节最好按照顺序来读。

在介绍让你自己实现某些功能的方法之前，我通常会先把现成可用的工具讲清楚。比如说第二部分的第 2 章覆盖了序列类型（sequence type），但是像 `collections.deque` 这种类可能就会一带而过。一直到第四部分，我们才会看看如何从抽象基类（abstract base class，ABC）中获利，抽象基类则被封装在 `collections.abc` 这个包里。如果想创建自己的 ABC，你可能得看到第四部分的最后一些内容才行，因为我一直觉得，如果没有熟练使用 ABC 的经验，贸然去实现一套自己的东西是不合适的。

这样做有几个好处。第一，知道有什么现成的工具可用，能避免重新发明轮子。毕竟我们使用现有集合类型（collection type）的概率要远大于自己动手写一套新的。第二，这样一来，在讨论如何写新类型之前，我们能够有更多的机会来了解这些现成类的高级用法。第三，比起从零开始构建一个 ABC，继承已有的 ABC 库应该会简单一些。最后，我认为在

看过一些实际的案例之后，理解抽象会更轻松。

当然，这样也会带来一些不便之处，比如书里的向前引用就会分散在各个不同的章节里面。但是经过上述这番梳理，我想这一点不便之处也是可以容忍的。

下面是本书每一部分的主题。

第一部分

第一部分只有单独的一章，讲解的是 Python 的数据模型（data model），以及如何为了保证行为一致性而使用特殊方法（比如 `__repr__`），毕竟 Python 的一致性是出了名的。其实整本书几乎都是在讲解 Python 的数据模型，第 1 章算是一个概览。

第二部分

第二部分包含了各种集合类型：序列（sequence）、映射（mapping）和集合（set），另外还提及了字符串（`str`）和字节序列（`bytes`）的区分。说起来，最后这一点也是让亲者（Python 3 用户）快，仇者（Python 2 用户）痛的一个关键，因为这个区分致使 Python 2 代码迁移到 Python 3 的难度陡增。第二部分的目标是帮助读者回忆起 Python 内置的类库，顺带解释这些类库的一些不太直观的地方。具体的例子有 Python 3 如何在我们观察不到的地方对 `dict` 的键重新排序，或者是排序有区域（locale）依赖的字符串时的注意事项。为了达到本部分的目标，有些地方的讲解会比较大而全，像序列类型和映射类型的变种就是这样；有时则会写得很深入，比方说我会对 `dict` 和 `set` 底层的散列表进行深层次的讨论。

第三部分

如何把函数作为一等对象（first-order object）来使用。第三部分首先会解释前面这句话是什么意思，然后话题延伸到这个概念对那些被广泛使用的设计模型的影响，最后读者会看到如何利用闭包（closure）的概念来实现函数装饰器（function decorator）。这一部分的话题还包括 Python 的这些基本概念：可调用（callable）、函数属性（function attribute）、内省（introspection）、参数注解（parameter annotation）和 Python 3 里新出现的 `nonlocal` 声明。

第四部分

到了这里，书的重点转移到了类的构建上面。虽然在第二部分里的例子里就有类声明（class declaration）的出现，但是第四部分会呈现更多的类。和任何面向对象语言一样，Python 还有些自己的特性，这些特性可能并不会出现在你我学习基于类的编程的语言中。这一部分的章节解释了引用（reference）的原理、“可变性”的概念、实例的生命周期、如何构建自定义的集合类型和 ABC、多重继承该怎么理顺、什么时候应该使用操作符重载及其方法。

第五部分

Python 中有些结构和库不再满足于诸如条件判断、循环和子程序（subroutine）之类的顺序控制流程，第五部分的笔墨会集中在这些构造和库上。我们会从生成器（generator）起步，然后话题会转移到上下文管理器（context manager）和协程

(coroutine)，其中会涵盖新增的功能强大但又不容易理解的 `yield from` 语法。这一部分以并发性和面向事件的 I/O 来结尾，其中跟并发性相关的是 `collections.futures` 这个很新的包，它借助 `futures` 包把线程和进程的概念给封装了起来；而跟面向事件 I/O 相关的则是 `asyncio`，它的背后是基于协程和 `yield from` 的 `futures` 包。

第六部分

第六部分的开头会讲到如何动态创建带属性的类，用以处理诸如 JSON 这类半结构化的数据。然后会从大家已经熟悉的特性（property）机制入手，用描述符从底层来解释 Python 对象属性的存取。同时，函数、方法和描述符的关系也会被梳理一遍。第六部分会从头至尾地实现一个字段验证器，在这个过程中我们会遇到一些微妙的问题，然后在最后一章中就自然引出像类装饰器（class decorator）和元类（metaclass）这些高级的概念。

以实践为基础

一般情况下，我们会用 Python 的交互式控制台来探索各种库和语言本身。有些读者可能对静态的需要编译的语言更熟悉，但是这些语言可能不会提供 REPL（read-eval-print loop，读取、求值、输出的循环）。在这里我想强调一下 Python 交互式控制台，也就是 REPL，作为一个学习工具的重要性。

doctest（https://docs.python.org/3/library/doctest.html）是 Python 的一个标准库，做测试用的。这个库通过模拟控制台对话来检验表达式求值是否正确，而本书中几乎所有代码的测试，包括那些在控制台里的输出，都是通过这个库来进行的。doctest 看起来就像是 Python 交互式控制台的剧本，你甚至都不需要了解它背后的运行机制就可以直接用它来试验书里的例子。

我有时为了事先说明一段代码的目的，会在展示代码之前先摆出相应的 doctest 文本。这是因为我认为，在考虑如何实现一个功能之前，先严格地列出这个功能能做什么，这能帮助我们在编程时把精力花在该花的地方。测试驱动开发（TDD）的精髓就是先写测试，我后来发现这种精神在教学中也是大有益处的。如果你对 doctest 还不熟悉，花点时间阅读它的文档（https://docs.python.org/3/library/doctest.html）。结合本书的源码（https://github.com/fluentpython/example-code），你可以在操作系统的控制台里键入 `python3 -m doctest example_script.py` 来验证书中几乎所有代码的正确性。

硬件

书中有一些简单的时间和基准测试，跑这些测试的时候我用的是写书时的两台笔记本电脑。一台是产于 2011 年的 MacBook Pro 13 英寸笔记本，配置是 2.7 GHz 的英特尔 Core i7 处理器、8GB 的内存和机械硬盘；另一台是产于 2014 年的 MacBook Air 13 英寸笔记本，配置是 1.4 GHz 的英特尔 Core i5 处理器、4GB 内存和一个固态硬盘。MacBook Air 的处理

器虽然慢一些，内存也没有另一台多，但是它的内存快一些（1600 MHz，MacBook Pro 13英寸则是 1333 MHz），另外它的硬盘也更快，因此在日常使用中我并没有感觉到两台笔记本有速度上的差异。

杂谈：个人的一点看法

从 1998 年起，我一直在使用 Python，也做 Python 教学，另外还一直在为它辩护。我一直都很享受这个过程，尤其是喜欢研究 Python 同其他语言在设计和理论上的不同。因此在有些章节的最后，我会加上一点自己对 Python 以及其他语言的看法，我把这部分叫作“杂谈”。如果你对这些东西不感兴趣，跳过即可，因为这些并不是必读的。

Python术语表

我希望这本书不仅仅是关于 Python 的，也是关于 Python 的文化的。在过去 20 多年的交流中，Python 社区形成了它独有的行话和缩写。本书的最后有一部分叫“Python 术语表”，里面列出了在 Python 爱好者中具有特别意义的词句。

Python版本表

本书所有的代码都在 Python 3.4 里测试过，而且是应用最广的用 C 实现的 CPython 3.4。只有一个例外，在 13.4 节中的“Python 3.5 新引入的中缀运算符 `@`”附注栏里，我提到了新的 `@` 运算符，它只在 Python 3.5 里被支持。

凡是支持 Python 3.*x* 的解释器——包括 PyPy3 2.4.0——都可以运行书里的代码（PyPy3 2.4.0 其实已经支持 Python 3.2.5）。有一点需要注意的是，`yield from` 和 `asyncio` 只在 Python 3.3 或者更新的版本里才有。

几乎所有的代码稍做修改后都能在 Python 2.7 里运行，除了第 4 章中那些跟 Unicode 相关的例子，这是从 Python 3 出现以来就有的问题。

排版约定

本书使用了下列排版约定。

- *楷体*
  表示新术语。
- `等宽字体（constant width）`
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。
- **`加粗等宽字体（constant width bold）`**
  表示应该由用户输入的命令或其他文本。

- 等宽斜体（*Constant width italic*）
  表示应该由用户输入的值或根据上下文确定的值替换的文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

使用代码示例

书中的所有完整代码和大多数程序片段都可以从本书的 GitHub 代码库中获取（https://github.com/fluentpython/example-code）。

我们很希望但并不强制要求你在引用本书内容时加上引用说明。引用说明一般包括书名、作者、出版社和 ISBN。比如："*Fluent Python* by Luciano Ramalho (O'Reilly). Copyright 2015 Luciano Ramalho, 978-1-491-94600-8."

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应运而生的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。

技术专家、软件开发人员、Web 设计师、商务人士和创意专家等，在开展调研、解决问题、学习和认证培训时，都将 Safari Books Online 视作获取资料的首选渠道。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那里找到本书的相关信息，包括勘误表、示例以及其他信息。本书的网站地址是：http://shop.oreilly.com/product/0636920032519.do

对于本书的评论和技术性问题，请发送电子邮件到：bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

Josef Hartwig 设计的包豪斯国际象棋套装体现了最佳的设计理念：美观、简洁而清晰。有一位建筑师父亲，以及一位字体设计师弟弟，Guido van Rossum 设计出了一门经典的编程语言。我之所以热衷于教授 Python，也正是因为它的美观、简洁和清晰。

Alex Martelli 和 Anna Ravenscroft 是最先看到本书大纲的人，也是他们鼓励我把大纲交给 O'Reilly 出版社的。他们的书不但向我展示了地道的 Python 代码，还让我见识了什么才称得上是清晰、准确和有深度的技术写作。Alex 在 Stack Overflow 上的 5000 多个回答（http://stackoverflow.com/users/95810/alex-martelli）也体现了他对 Python 语言基础和正确用法的深入理解。

Martelli 和 Ravenscroft 同时也是本书的技术审稿人。除了他们之外，技术审稿人还有两位：Lennart Regebro 和 Leonardo Rochael。技术审稿团队里的每个人都至少有 15 年的 Python 经验，为许许多多具有广泛影响力的 Python 项目贡献过代码，并且跟社区里的其他开发者走得很近。审稿人一共提出了数百个修订、建议、问题和观点，为这本书做出了巨大贡献。另外，Victor Stinner 帮我审阅了第 18 章，他同时也是该章里提到的 `asyncio` 的维护者之一。在过去的几个月里能够跟他们合作，我感到很荣幸。

本书编辑 Meghan Blanchette 是一位出色的导师。她不但帮助我梳理整本书的结构、增强内容的连贯性，还为我指出哪里写得不够有趣，并且督促我及时交稿。Brian MacDonald 在

Meghan 休假的时候帮忙编辑了第三部分。跟他们以及 O'Reilly 的所有人打交道的过程都十分愉快。另外 Atlas 系统的开发和支持团队也很棒（Atlas 是 O'Reilly 的图书出版平台，我就是在这个平台上写作的）。

Mario Domenech Goulart 在看过本书第一次提前发行的版本后，提供了海量的详细建议。另外我还从 Dave Pawson、Elias Dorneles、Leonardo Alexandre Ferreira Leite、Bruce Eckel、J. S. Bueno、Rafael Gonçalves、Alex Chiaranda、Guto Maia、Lucas Vido 和 Lucas Brunialti 那里获得了宝贵的反馈。

几年来有很多人都在劝我写书，Rubens Prates、Aurelio Jargas、Rudá Moura 和 Rubens Altimari 这几位算是最有说服力的了。Mauricio Bussab 算得上带我入门的人，并且他让我有了第一次写书的尝试。Renzo Nuccitelli 毫不在乎这本书的写作可能会影响到我们合作的 python.pro.br 项目的进度，他从一开始就大力支持。

Python 巴西社区是一个集思广益、乐于分享且充满乐趣的地方。Python 巴西小组（https://groups.google.com/group/python-brasil）中有数千个人，每次的全国范围的会议都会把成百上千人聚集在一起。在我的 Python 爱好者成长之路上，对我影响最大的人有：Leonardo Rochael、Adriano Petrich、Daniel Vainsencher、Rodrigo RBP Pimentel、Bruno Gola、Leonardo Santagada、Jean Ferri、Rodrigo Senra、J. S. Bueno、David Kwast、Luiz Irber、Osvaldo Santana、Fernando Masanori、Henrique Bastos、Gustavo Niemayer、Pedro Werneck、Gustavo Barbieri、Lalo Martins、Danilo Bellini 和 Pedro Kroger。

Dorneles Tremea 是个非常棒的朋友（他很愿意花时间分享他的知识），他不但是很厉害的开发者，还是巴西 Python 协会中最鼓舞人心的领导人。可惜他过早离开了我们。

我的学生们同时也是我的老师，他们的问题、见解、反馈和那些富有创造性的回答教会了我很多。Érico Andrei 和 Simples Consultoria 让我头一次有机会集中精力做一名 Python 教师。

Martijn Faassen 是我的 Grok 导师，他同我分享了很多关于 Python 和尼安德特人的想法。Martijn 所做的事情，还有来自 Zope、Plone 和 Pyramid planets 的 Paul Everitt、Chris McDonough、Tres Seaver、Jim Fulton、Shane Hathaway、Lennart Regebro、Alan Runyan、Alexander Limi、Martijn Pieters 和 Godefroid Chapelle 等人所做的事情，在我事业发展的过程中起到了决定性的作用。多亏了 Zope 和第一波互联网浪潮，让我在 1998 年就开始从事 Python 相关的工作并以此为生。José Octavio Castro Neves 是我的搭档，我们在巴西开了第一家以 Python 业务为主的软件公司。

在更广阔的 Python 社区当中高手如云，我实在是没办法一一列出他们的名字。但是除了之前提到的之外，我还要感谢 Steve Holden、Raymond Hettinger、A.M. Kuchling、David Beazley、Fredrik Lundh、Doug Hellmann、Nick Coghlan、Mark Pilgrim、Martijn Pieters、Bruce Eckel、Michele Simionato、Wesley Chun、Brandon Craig Rhodes、Philip Guo、Daniel Greenfeld、Audrey Roy 和 Brett Slatkin，感谢他们让我见识到更新更好的教授 Python 的方法。

我基本上是在家里的办公室和两个公共空间完成这本书的写作的。两个公共空间分别是 CoffeeLab 和 Garoa Hacker Clube。CoffeeLab（http://coffeelab.com.br）是位于巴西圣保罗 Vila Madalena 区的一个咖啡极客大本营。Garoa Hacker Clube（https://garoa.net.br）则是一个开放的黑客空间，任何人都可以在这里实验他们的新点子。

Garoa 社区还为我提供了灵感、基础设施和放松的环境，我想 Aleph 会喜欢这本书的。

我的母亲 Maria Lucia 和父亲 Jairo 一直都以各种方式支持我。我真希望我的父亲还健在并看到本书的出版，同时也为能与我的母亲分享这本书而感到开心。

在写这本书的 15 个月里，身为丈夫的我几乎一直在工作，我的妻子 Marta Mello 陪我一起熬过了这段日子。在这如同跑马拉松的写作过程中，她不但一直支持我，而且在我想要放弃的时候陪我一起渡过难关。

谢谢你们每一个人，谢谢你们做的每一件事。

电子书

扫描如下二维码，即可购买本书电子版。

目录