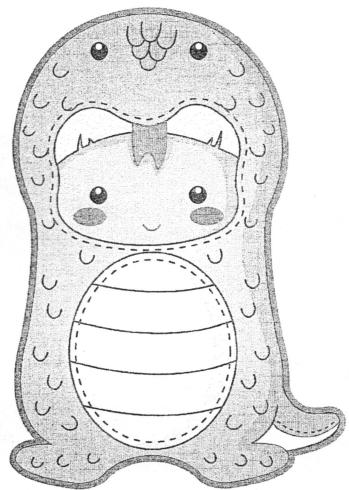


# Python 3

## 学习笔记

上卷

雨痕 / 著



电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

## 内 容 简 介

经过 9 年的发展，Python 3 生态已相当成熟。无论是语言进化、解释器性能提升，还是第三方支持，都是如此。随着 Python 2.7 EOF 日趋临近，迁移到 Python 3 的各种障碍也被逐一剔除。是时候在新环境下学习或工作了。

人们常说 Python 简单易学，但这是以封装和隐藏复杂体系为代价的。仅阅读语言规范很难深入，亦无从发挥其应有能力，易学难精才是常态。本书尝试通过分析解释器的工作机制来解析 Python 3.6 语言理论，以期帮助读者加深理解。

本书着重于剖析语言的相关背景和实现方式，适合有一定 Python 编程基础的读者阅读、参考。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

### 图书在版编目（CIP）数据

Python 3 学习笔记. 上卷 / 雨痕著. —北京：电子工业出版社，2018.1

ISBN 978-7-121-33274-6

I. ①P… II. ①雨… 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 308906 号

策划编辑：许 艳

责任编辑：李云静

印 刷：北京京科印刷有限公司

装 订：北京京科印刷有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×980 1/16 印张：26.75 字数：500 千字

版 次：2018 年 1 月第 1 版

印 次：2018 年 1 月第 1 次印刷

定 价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 [zlts@phei.com.cn](mailto:zlts@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

本书咨询联系方式：010-51260888-819, [faq@phei.com.cn](mailto:faq@phei.com.cn)。

# 前言

写作本书时，我已然摆脱“萌新”身份，算是稍有经验的作者。可即便如此，我依然无法保证本书的内容完全正确，且满足所有人的胃口。显然，这不可能做到。

在我看来，图书大抵分两类：学习和研究。学习类书籍满足日常学习和提升需要，用简练的语言把问题说清楚。最关键的是有清晰的线索，把散乱的知识串联起来，学习者可据此了解前因后果。至于研究类图书或论文，则应摆脱基础，摆脱语法，重点关注算法、架构、性能，乃至内部实现。所有这些，均以思想为支撑，超脱语言窠臼，构建并完善体系。

不同于写散文或小说，技术类图书的文字不好组织。自然语言易阅读，但不便描述有复杂流程分支的逻辑，易导致歧义。更何况，这其中还有各种转译带来的麻烦。故技术类图书应以自然语言开宗明义，阐述理论与规则，随后用代码对这段文字进行解释，毕竟代码先天有描述逻辑的优势。

很多书，尤其是英文版的图书，习惯于用大量篇幅对代码示例做各种讲解。我感觉这有些啰唆，想必很少有人去读第二遍，大家最多也就是用记号笔画出重点而已。既然如此，我们为何不信读者能阅读并理解这些代码呢？这本来就是程序员吃饭的本钱，最多在关键位置辅以注释便可。当然，阅读前提怕是要设定为非入门读者。好在我一再强调自己写的是第二本书，或曰“闲书”。

在本书中，对于理论层面，我会尝试说得明白些。当然，书中还会引入一些类比，这些类比或许不是非常合适，但却可以加深读者对相关问题的理解，毕竟不是所有人都能明白那些云里雾里的抽象理念。一如上面所言，文字与代码相辅相成，我们应静下心来用代码去验证文字背后的含义。在我眼里，代码也是一种自然语言，缩排跳转仿若图形，本就是最好的笔记注释。起码它离机器语言上有些距离，是为了便于人类阅读而发明的。

无论我说得多悦耳动听，这终归只是一本学习笔记，算不上专业，仅适合读者闲暇时翻阅一二。

## 关于本书

全套书分为上下两卷。上卷以语言为主，基本涵盖语言相关内容，包括语法、测试、调试，乃至解释器等层面的基本知识。下卷计划以标准库、优秀扩展库、并发编程，以及架构设计展开，算是对上卷“闲书”稍加修正。

书中示例运行环境：macOS 10.12，CPython 3.6，IPython 6.2

---

鉴于不同运行环境的差异性，示例输出结果（尤其是 id、内存地址等信息）会有所不同。另外，为阅读方便，本书对输出结果做了裁剪处理，请以实际运行结果为准。

---

## 读者定位

本书着重于剖析语言的相关背景和实现方式，适合有一定 Python 编程基础的读者（比如准备从 Python 2.7 升级到 Python 3.6 环境的读者）阅读。至于初学者，建议寻找从零开始、循序渐进地介绍如何编写代码的其他图书为佳。

## 联系方式

鄙人能力有限，书中难免存在错漏之处。读者如在阅读过程中发现任何问题，请与我联系，以便更正。谢谢！

- 邮件：qyuhen@hotmail.com
- 微博：weibo.com/qyuhen

雨痕

二〇一七年，仲秋

# 读者服务

轻松注册成为博文视点社区用户（[www.broadview.com.cn](http://www.broadview.com.cn)），扫码直达本书页面。

- **提交勘误：**您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/33274>



---

# 上卷 语言详解

---

基于 Python 3.6



# 目录

第 1 章 概述 .....	1
第 2 章 类型 .....	5
2.1 基本环境 .....	5
2.1.1 印象 .....	5
2.1.2 名字 .....	9
2.1.3 内存 .....	16
2.1.4 编译 .....	26
2.1.5 执行 .....	30
2.2 内置类型 .....	34
2.2.1 整数 .....	35
2.2.2 浮点数 .....	44
2.2.3 字符串 .....	50
2.2.4 字节数组 .....	61
2.2.5 列表 .....	65
2.2.6 字典 .....	76
2.2.7 集合 .....	85

<b>第 3 章 表达式</b>	91
3.1 词法	91
3.1.1 源文件	92
3.1.2 代码	95
3.2 赋值	100
3.2.1 增量赋值	101
3.2.2 序列解包	102
3.2.3 作用域	107
3.3 运算符	110
3.3.1 链式比较	113
3.3.2 切片	114
3.3.3 逻辑运算	117
3.4 控制流	121
3.4.1 选择	121
3.4.2 循环	123
3.5 推导式	128
<b>第 4 章 函数</b>	133
4.1 定义	133
4.2 参数	141
4.3 返回值	149
4.4 作用域	151
4.5 闭包	157
4.6 调用	165
<b>第 5 章 迭代器</b>	174
5.1 迭代器概述	174

5.2 生成器 .....	179
5.3 模式 .....	186
5.4 函数式编程 .....	190
<b>第6章 模块 .....</b>	<b>195</b>
6.1 定义 .....	195
6.2 导入 .....	199
6.2.1 搜索 .....	200
6.2.2 编译 .....	202
6.2.3 引用 .....	204
6.3 包 .....	213
6.3.1 初始化 .....	214
6.3.2 相对导入 .....	218
6.3.3 拆分 .....	221
<b>第7章 类 .....</b>	<b>223</b>
7.1 定义 .....	223
7.2 字段 .....	230
7.3 属性 .....	235
7.4 方法 .....	238
7.5 继承 .....	243
7.5.1 统一类型 .....	244
7.5.2 初始化 .....	246
7.5.3 覆盖 .....	247
7.5.4 多继承 .....	248
7.5.5 抽象类 .....	254
7.6 开放类 .....	256

7.7 运算符重载 .....	263
<b>第 8 章 异常 .....</b>	<b>269</b>
8.1 异常概述 .....	269
8.2 断言 .....	284
8.3 上下文 .....	288
<b>第 9 章 元编程 .....</b>	<b>294</b>
9.1 装饰器 .....	294
9.1.1 实现 .....	295
9.1.2 应用 .....	301
9.2 描述符 .....	304
9.3 元类 .....	308
9.3.1 自定义 .....	309
9.3.2 应用 .....	314
9.4 注解 .....	315
<b>第 10 章 进阶 .....</b>	<b>318</b>
10.1 解释器 .....	318
10.1.1 字节码 .....	318
10.1.2 全局锁 .....	321
10.1.3 执行过程 .....	326
10.1.4 内存分配 .....	334
10.1.5 垃圾回收 .....	343
10.2 扩展 .....	349
10.2.1 ctypes .....	349
10.2.2 Cython .....	356

---

第 11 章 测试 .....	364
11.1 单元测试 .....	364
11.1.1 unittest .....	365
11.1.2 unittest.mock .....	374
11.1.3 coverage .....	383
11.2 性能测试 .....	383
11.2.1 timeit .....	383
11.2.2 profile .....	385
11.2.3 line profiler .....	388
11.2.4 memory profiler .....	390
11.2.5 pympler .....	391
第 12 章 工具 .....	396
12.1 调试器 .....	396
12.2 包管理 .....	410



# 第 1 章 概述

Python 是一门相当有趣的编程语言。

其始于 1989 年末，约莫而立之年，比许多程序员的年龄还要大。在这段漫长的时光里，它见证了 C++ 的兴盛和群雄大战，看到了 Java 的异军突起和如日中天，更有同类 Ruby 凭借 RoR 领一时风骚，还有习惯丢三落四却每每笑到最后的 VC 被同族怼得灰头土脸。

在这期间还发生过什么？面向对象、设计模式、多层架构、面向服务等数不清的概念和名词。以程序界的划代标准，这已然是一个早该供在 WiKi 里的老古董，偶尔被某个年纪大的前辈拉出来讲古。然而，历经世事变幻，一朵朵“白莲花”最终都免不了闹个“腹黑”收场。就连当年那些无敌论的吹鼓手，如今都成了大肆指摘的异见人士。

且不管风云如何，Python 活得很好，依旧占据排行榜前列，可见其不全然是一部遗存的程序设计编年史。Python 不仅能当“胶水”写工具脚本，还借着大数据、深度学习、机器学习和人工智能的风潮，一跃成为当红之选。

当然，事物总有两面性。一方面，我们能从雨后春笋般出现的新技术支持名单里找到它的身影；而另一面，身边似乎并无多少人去使用或关注它。它被推荐给孩子作为编程入门语言，也被专业人士用于特定场合，可恰恰在靠编程吃饭的程序员主战场上位置尴尬。其虽然有浩如烟海的第三方支持，但说起来似乎只有系统维护和网站应用。所有这些，让我们对这门语言既熟悉又陌生。

喜欢它的各有因由，批评的则火力集中。在各大社区里，不乏有人对其性能、语法，乃至千年话题全局锁大加指责，以烘托某种语言才是更好的选择。这源自部分开发人员，尤其是新人追求大而全的心理，缺乏理性定位。历史上，还从没有一种语言能包办所有应用，更不曾讨好过所有人。

作为应用语言，脑门还刻着简单和优雅字眼，自然要支持各时期的主流编程范式，竭力涵盖各类应用范畴，还需大费周章将复杂封装并隐藏起来，以期换取惯常喜新厌旧的程序员垂青。所以，从中你能看到命令式、函数式、面向对象、面向切面等程序设计方式。这造就了其广泛的支持，也带来易学难精的后果。历史包袱出现在所有步入中年的技术身上，其中有操作系统，有浏览器，自然也有编程语言。呼吁某某减肥和变革的声音不绝于耳，而后是新生代迈着轻盈的步伐后来居上。

可换个角度看，正因为与时俱进和兼容并蓄，方能存活至今。那些特立独行的，反倒未必笑到最后。君不见，方兴未艾的各路 NoSQL 不但因功能单一而横遭嫌弃，还遭看似老朽的“革命对象” RDBMS 反戈一击。要么纯粹享受寂寞，要么广博得汇溪成海。以“色”娱人，能讨巧于一时，终难长久。人们总有个错觉，似乎新技术是凭空出现，是年轻人的主场。但实际上，其依托早已存世，或埋于地下，或束之高阁，且待时机。

Python 的简单和周全，降低了非专业人士的使用门槛，毕竟他们的精力不会放在语言身上。此时，广泛支持就成为优点，即便不了解面向对象，也可用面向过程写点什么。更何况，那令人咋舌的生态系统里，总能找到你需要却又无法实现的东西。相当有趣的是，很有些专业扩展库，恰恰是用程序员看不上眼的代码完成的。兴许，作者只是个数学家，或图形学方面的天才。是以，任何一种设计都有其出现的原因和存在的理由。

在国内，将 Python 当作主力编程语言的人群很有限，其影响力和热度甚至不如某些后来者。这固然有其自身的种种原因，可社区疲软也显而易见。不管境况如何，难得有这样一门能长久陪伴，且行事周全的语言用于工作和学习，大家须珍惜。

## Python 3

如果你对 Python 3 的了解尚停留在数年以前，那是时候更新一下认知了。下面这样一段文字，或许可代表生态圈的主流态度。

---

What Python version should I use with Django ?

Python 3 is recommended. Django 1.11 is the last version to support Python 2.7. Support for Python 2.7 and Django 1.11 ends in 2020.

Since newer versions of Python are often faster, have more features, and are better supported, the latest version of Python 3 is recommended.

You don't lose anything in Django by using an older release, but you don't take advantage of the improvements and optimizations in newer Python releases.

—— Django FAQ

---

最初，迁入 Python 3 的阻碍可能是某个扩展库不支持，这也是很多人的主要理由。但到了今天，在 Python 3 Readiness 所统计的最流行的 360 个包里，有超过 95% 支持 Python 3。起码对于新项目，这已不是问题。

另一个理由，应该是 Python 3 早期那让人失望的性能。可自 2008 年发布，至今 9 年，期间经多个版本的优化改进，其性能改善良多，早已不再是“弱鸡”的形象。

尽管在多年前，官方将 Python 2.7 EOL (end of life) 推迟到 2020 年。可晃悠至今，所余时间已不足三年。即便因某些原因再度推迟，那也不是新项目继续使用 2.7 的理由，因为不会再有 Python 2.8。

还有，Python 3 的 asyncio 已成为主流异步框架。众多 Web Framework、Database Driver 等都已提供支持，并获得更好的执行性能。至于那些新增的、改进的，被摒弃且不合时宜的，等等，都值得我们去了解和尝试一下。