一本值得你反复研读的Python佳作，注重内功修炼，而非花拳绣腿
来自于真实教学经验的详略取舍，注重对知识的探寻和思考过程
透过Python的表象，深入程序设计的本质，让你知其然并知其所以然

Python编程从0到1

（视频教学版）

张頔◎著

机械工业出版社
China Machine Press

Python编程从0到1

（视频教学版）

张頔◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Python编程从0到1：视频教学版/张頔著. —北京：机械工业出版社，2019.8

ISBN 978-7-111-63295-5

Ⅰ. P… Ⅱ. 张… Ⅲ. 软件工具－程序设计 Ⅳ. TP311.561

中国版本图书馆CIP数据核字(2019)第152929号

Python 编程从 0 到 1（视频教学版）

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：欧振旭 李华君
责任校对：姚志娟
印 刷：中国电影出版社印刷厂
版 次：2019 年 8 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16
印 张：17.25
书 号：ISBN 978-7-111-63295-5
定 价：79.00 元

客服电话：（010）88361066 88379833 68326294
投稿热线：（010）88379604
华章网站：www.hzbook.com
读者信箱：hzit@hzbook.com

前言

生产力水平的发展是人类历史的主旋律。决定生产力水平的重要因素之一是生产工具。从这个角度讲，人类历史就是从石块到计算机的历史：从石器时代到信息时代。计算机是信息时代的主要生产工具，为了掌握这种工具，就要学习程序设计语言。

程序设计语言很多，相关的书籍更多。初学者常困惑于如何选择入门语言[①]，一旦决定后又困惑于选择什么教材。令人沮丧的是，选择的机会成本并不低：一般初学者完整地学习某门编程语言需要一个月甚至更多时间。大专院校课程改革的机会成本则更高。有人说选择什么语言入门不重要，成为高手后自然兼通各种语言。这种说法忽视了大多数学习者并不会成为高手的事实，而且选择性遗忘了他们自己作为初学者时所走的弯路。**总而言之，学习者和教师在选择适合入门的编程语言和书籍时应当谨慎行事**。因此笔者有必要在前言中将写作意图和内容取舍进行说明，以作为读者选择的依据。

写作背景和目的

笔者在 2018 年秋接到邀约并着手写作，全稿终于 2019 年春。此时 Python 已经跻身主流语言之列。之所以称其为“主流”语言，在笔者看来有以下几点证据：

- 在各类有影响力的语言排行榜上，使用 Python 的比例开始占据较大份额，排名也很靠前[②]；
- 在许多领域，Python 成为主要的编程语言，甚至是首选语言，例如在数据分析、人工智能和服务器开发等领域；
- 教育部将 Python 纳入了国家计算机等级考试的科目[③]。

而在此之前其已经得到了广泛应用：

- Python 逐渐成为许多国际一流高校程序设计课程的教学语言；
- 许多基础软件框架采用 Python 作为设计语言或提供 Python 的编程接口；
- Python 在开源软件界和工业界已广受欢迎。

上述事实是笔者决定基于 Python 编写本书的原因[④]。其他的原因则是在教学中无合适教材可用。数年前笔者领导团队设计 Python 课程体系，那时曾经“遍访”各种 Python 书

---

① 在本书中，“语言”一词默认表示程序设计语言，而非自然语言。

② 在 2019 年 2 月的 TIOBE 排行榜上，Python 以 7.574%的份额占据第三位，超越 C++，仅次于 Java 和 C。

③ 教育部考试中心“全国计算机等级考试简介 2018 版”。

④ 本书首先是“程序设计教科书”，而非“Python 书”。

籍和教程。当时的 Python 书籍或者是简单的“手把手入门”，或者是大部头的“知识大全”，或者是应用于某具体领域的“手册指南”。

随着近年来 Python 教学实践的展开，市面上开始出现各种教科书体裁的 Python 书籍（如罗伯特·塞奇威克等所著的《程序设计导论：Python 语言实践》）。那为何还要再写一本同样主题的教科书呢？因为教学对象不同，教学目标有所差别，教学过程各有侧重。所以教科书的首要价值在于“不同”。这种不同在本书来说至少有两个层面，一是 Python 与其他语言的不同，二是本书与其他 Python 书籍的不同。

学习或讲授新知识时，首先应注意到的往往是其与原有认知的相同之处，但唯有深入理解并能运用其不同后，才算窥得门径。例如 Python 也有循环、分支和函数等各种语言俱有之概念，但学习者如只见共性，并由此得出 Python 很简单的结论，便失去了学习新语言的意义。教师在教学生时也不能仅仅按照讲授旧有语言的经验，把讲义中的例子依次用 Python 重写一遍了事。[①]

书的“不同”则在于体裁形式、内容取舍编排和作者观点的不同。既然是教科书，总不大好写成对话体，也不适宜画成漫画[②]。所以教科书的不同主要体现在取舍、次序、示例和观点等方面。然而写一本“完全不一样”的书是很难的，尤其是在 Python 已经相当流行的当下。[③]

“不同”总是相对的，各种相对性参照中，对读者来说最重要的参照便是读者自身。初来者看处处皆是新奇，见多识广后则觉得不过尔尔。所以下文仅对本书内容的取舍和编排进行说明，至于其中有何“不同”则留待读者自己体会。

取舍

在教科书中全面讲授 Python 的细节是不现实的（这里说的语法细节包括完整的语法模型、各种对象的 API[④]，以及各种标准库），原因有以下几点：

- 过多的细节会喧宾夺主，无法凸显真正重要的核心内容；
- 语言中的艰深部分不适合本书面向的教学阶段，也不常用于多数程序员的日常工作；
- Python 标准库包罗万象，全面介绍是不现实的（无论从篇幅上还是读者的知识背景

① 以回调函数为例，在 C 语言课程中必待讲完指针及函数指针才能引入该概念，而 Java 也须等到讲完继承和接口之后才能展开讨论。因此传统的程序设计课程多在较靠后的课时讲授回调函数。但 Python 语言却可以也应当在讲到函数时就引入回调函数。

② 当然作者也没那么大本事，既没本事用对话的方式教学，也没本事画漫画。不过的确有用对话体或漫画做成的程序设计书籍，大都相当优秀。

③ 笔者在书店购买书籍时有一个原则：在比较熟悉的领域，一本书拿起来随意翻开三处，如果有一处独到，就标记为“待买”，如果处处独到，则立刻买下。如果有一位教师在讲授 Python 时觉得本书有一处有参考价值，那么笔者便心满意足了。

④ Application Programming Interface，应用程序编程接口。

基础而言）；

- Python 在不断发展，其细节仍然在不断变化中。

笔者认为，既然在整体上无法面面俱到，那么在局部也不应有这种负担。例如“异常处理”这一主题，如果完整地讲授异常机制的每个细节（如各种内建异常类型），则需要相当多的篇幅[①]。**但真实的情况是：绝大多数的工程师根本不懂如何处理程序的意外情况，不论是使用 C 语言这样没有异常处理机制的语言，还是使用 Java 或 Python 这样有完整异常处理机制的语言**。究其原因有以下几点：

- 一是初学者没有能力接受太多的异常处理知识，正常程序还写不明白，哪里有精力去整什么异常处理；
- 二是异常处理的核心在于全面、准确地剖析程序的各种意外情况，不具备这个能力，学习再多的异常处理机制也是枉然；
- 三是在学习程序设计的初级阶段，往往用算法进行练习（比如走个迷宫、匹配个字符串），不和复杂的外部世界（如网络）打交道；
- 四是考虑到教学重点和叙述篇幅，也往往假定数据有效，这样就不用考虑处理意外情况。这样一来，教学中费大力气讲授的异常处理机制，也因为暂时无用而被抛诸脑后。

本书则不纠缠于异常处理的细节和内建异常的类型，而是在讲述完基本语法后详细剖析一个程序实例（1.10.2 节），展示如何分析和处理各种意外的输入情况，篇幅上也仅限制为一节。读者若能“吃透”这一部分内容，则当有编写健壮程序的意识，自能够在工作中举一反三，也无须笔者再行赘述。反之，若对健壮性不够重视，则多费篇幅也没有意义。

作为这种取舍思路的补充，本书有时会指出一些自学内容，将其留作练习，并引导读者进行思考或通过网络查阅相关资料。作为编程入门教科书，本书首要讲述程序设计的一般性方法，例如：

- 数据类型、流程控制、输入/输出和函数等基本手段；
- 状态机、递归、函数式编程等高级技巧；
- 数据结构和算法；
- 面向对象的设计思想。

对 Python 自身语法的介绍，也在本书各部分占据了相当篇幅。然而本书终归不是语法手册，所以并不追求语法知识细节的完整性。书中大部分小节后会给出延伸的问题和练习，或是明确的编码练习，或是要求读者进行深入地学习和思考，统称为“思考和扩展练习”。本书在设计和挑选例题时力图做到和传统的程序设计入门书籍在角度和深度上有所差异，目的在于为学生和教师提供更加开阔的思路。

计算机程序设计的理论基石（如语言基本模型、数据结构和算法）是在半个世纪前奠定的。虽然 Python 诞生的年代稍晚，但也基于这些基石构建而成。Python 的主体部分是

① 以《Python 学习手册（第 4 版）》为例，异常机制总共占据了 60 页之多，还并未一一列举内建异常。

基于朴素平实的想法构建而成，而非天才的灵光一现。本书的一个重要初衷就是将这些朴素呈现给读者。笔者舍去了一些和该初衷无关的内容，如传统教学中往往着重介绍的字符串显示控制标记，在本书中只是一带而过。而那些更为深层次的基本概念则得到了强化，例如状态机模型、递归消除、面向对象动机及性能权衡等。

本书在示例中坚持给出完整可执行的代码或交互界面步骤，然而本书绝非“手把手”的教程。书中有些内容对于学习者来说是具有相当难度的，需要读者反复阅读思考并且动手实践才能够理解或掌握。这种难度是具有现实意义的：较难内容大多数来自于工程实践、后续学习，甚至是求职面试中的重点。读者应当认识到对这些问题的思考能力和思考过程的重要性。

内容编排

本书分为 4 章，取名为：

- 第 1 章　基础；
- 第 2 章　函数；
- 第 3 章　数据结构；
- 第 4 章　面向对象。

上述各章内容并非泾渭分明，原因主要在于 Python 的知识点存在许多“循环依赖”，如图 1 所示。例如 for 循环用到了迭代器的概念，而讲授迭代器需要足够的面向对象知识，绕过分支控制先完整地讲述面向对象也不现实。对于有经验的读者来说，这没有任何问题，但对初学者则会造成很大困扰。基于这些原因及教学实践，本书在第 1 章中简要讲授了函数定义而非拖至第 2 章，将定义类的基本方法提前至第 2 章而不是留待第 4 章。这两节内容的调整是本书的重要关节。

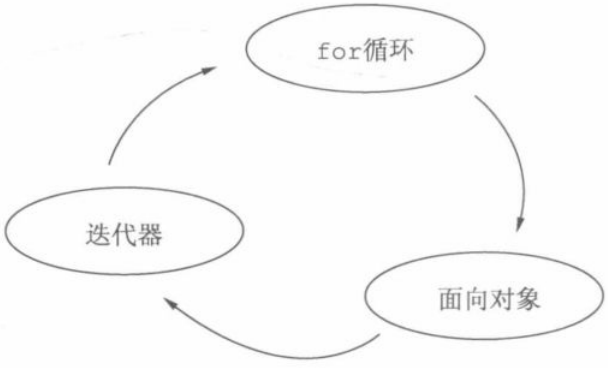

图 1　知识点的循环依赖

这 4 章内容以第 1 章篇幅最长，第 2、3 章次之，第 4 章最短。程序的基本设计方法、异常处理和程序调试等内容归入了第 1 章。模块和迭代器归入了第 2 章。部分内建类型，

如字典的深入讲解归入了第 3 章。在第 4 章面向对象设计部分，考虑到相应的学习阶段课时、Python 的混合风格特性和简洁的面向对象语法，笔者决定精简篇幅而非长篇大论。另外，由于在 Python 中对象模型和面向对象调用风格的无处不在，许多相关知识已经散见于前 3 章，也没必要再次重复讲述。

习题

本书没有单独整理的“习题集”。首先是因为笔者水平所限，并且时间也有限，无法设计大量的原创习题。另一个原因是笔者**反对**采用在语言学习阶段使用“刷题”的方式提升编程能力[①]。读者如能将本书示例及每小节末尾的“思考和扩展练习”完成，在笔者看来就足够了。如果读者在完成这些内容后仍觉不足，不要把精力浪费在所谓的“习题集”或“面试宝典”上，而应该去学习更为深入的主题（如操作系统、网络、算法或编译原理等），或投身于解决实际问题。

参考文献

本书中提到的若干算法、观点、文档和源码的原始出处在书尾以参考文献的形式给出。阅读原始文献是非常有必要的。通过对这些文献的阅读，即便是少量阅读，学习者也能够体会到创造者当时的心境。这看似增加了额外的学习任务，但实为捷径。为了便于读者检索，笔者用[1][2]的编排格式给出了所对应的参考文献序号。

版本

笔者在 Mac 计算机上完成了绝大部分的写作工作。写作时的 Python 最新版本是 3.7。当然这绝不意味着读者也需要搞一套这样的软硬件环境。使用 Linux 的读者不用太过担心兼容性问题。使用 Windows 系统的读者除去 1.6.3 节中管道行的相关命令外，应当可以顺利运行本书中的其他程序。笔者平时并不使用 Windows 系统，所以本书示例在 Windows 平台上可能会稍有更多的兼容性问题。但跨平台的兼容性带来的麻烦远不及 Python 本身升级带来的兼容性问题。所以在学习时纠结于应当使用哪种操作系统平台是意义不大的，因为 Python 未来的某次升级可能导致本书代码无法运行的情况更多。好在互联网的便利使得笔者对本书出版后做勘误和代码更新很容易，笔者会将本书的勘误信息和最新的升级信息反馈给出版社，读者可在出版社的官网上找到本书，获取这些资料。

① 习题往往是已有知识和技巧的构造（当然有些书将一些人类尚未解决的猜想也不客气地列为习题，那另当别论）。反复大量地做题会使学习者的思维模式陷入“解题模式”（尤其是那些设计精巧的习题更是害人不浅），导致他们在实际工作或科研中反而成绩平平。

格式约定

为了便于读者学习，本书坚持给出绝大多数示例的完整代码及运行效果。有的代码示例直接运行即可看到期望的结果，而有的代码则需要进行一些样例输入或者将其导入后调用。本书统一将这些运行结果或运行示例称为“**程序运行结果**”。

本书用以下格式表示代码实例。这些实例大部分能够直接运行，少部分需要读者自行补充完整。

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from random import random                    #导入模块
for i in range(10):                          #循环 10 次
print(i, ": ", random())                     #打印随机数
```

本书用以下格式表示交互执行的过程和结果。

```
$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, ...)
......
>>>
```

其中， $ 表示操作系统的终端（shell）提示符，>>>表示 Python 的交互式执行环境提示符。该格式还用来排版批量的文本，例如：

```
False       await       else        import      pass
None        break       except      in          raise
True        class       finally     is          return
and         continue    for         lambda      try
as          def         from        nonlocal    while
assert      del         global      not         with
async       elif        if          or          yield
```

至于在说明文字中用到的解释性代码，则直接用如下格式排版：

```
if expA and expB :
    ...
```

本书的代码参考 Python 的推荐风格 PEP-8，但有时为了行文紧凑也会缩减空格及空行。

获取配书资料

本书提供以下配书资料：

- 配套教学视频；
- 实例源代码文件；
- 教学 PPT。

这些资料需要读者自行下载。请登录华章公司网站 www.hzbook.com，在该网站上搜索到本书，然后单击“资料下载”按钮，即可在本书页面上找到下载链接。

读者对象

本书的最佳读者设定是有教师指导的程序设计初学者，笔者称之为“教科书”设定。这意味着本书不会教读者诸如安装执行环境和使用代码编辑器这类准备工作。这是教师的职责。如果读者是自学者，那么你应当首先确保自己正确地安装了 Python 运行环境，并尝试编辑一两个 Python 小程序运行一下。这在今天很容易，读者只需要随便找一个在线视频，看上二三十分钟即可。对于自学者来说，找到几位 Python 的使用者去获取一些初始建议，并且能够在学习遇到困惑时求助也是非常重要的。很多人在学习过程中因为某些不经意的障碍而放弃。比如原本代码写对了，但某个执行路径不对，在反复检查代码后依然失败而信心全无。

本书还特别适合一类读者，即学过一些程序设计语言的粗浅知识，马马虎虎地写过一些程序，处于某种目的又希望认真、正式地学习程序设计的学生。他们有可能是打算大学毕业希望找一份 IT 工程师工作的学生，也可能是中学阶段学过一些编程后希望在这条路上走得更远的学生。对这类读者来说，本书将带给他们更加深厚的基础能力。

致谢

本书的写作得到了好友姜寒先生的大力协助。姜先生有丰富的实践经验，又兼通各种主流程序设计语言。笔者在写作本书时多有疑难困阻，或关于 Python 本身，或关于其他广泛主题；又有时在详略取舍上难以决断，或是对一些观点信心不足。每每及此，与姜先生讨论后总能破除心中所障。除此之外，姜先生还审阅了本书的部分章节，从内容到行文，均提出了宝贵意见。

张颢

于北京

目录

第1章 基 础

本章将介绍程序设计的入门方法，主要分为以下 3 个阶段进行介绍。

第1阶段：1.1～1.6节

这部分介绍最基本的知识，如历史（1.1 节）、表达式（1.2 节）、运行程序（1.3 节）、内建类型（1.4 节）、流程控制结构（1.5 节）和输入输出（1.6 节）。这部分内容力图精简（尤其是内建类型一节），其目的在于让学习者尽快进入编程训练中。

本阶段还会简要介绍函数的定义方法（1.5.6 节），以便于在示例和练习中使用。完整论述函数这一主题则是第 2 章将要介绍的内容。本节介绍的内建类型（列表、字典等）则还会在第 3 章中深入讨论。

第2阶段：1.7～1.9节

本阶段通过一组编码实例展示前述程序设计基本方法的应用。学习者，尤其是初学者，在本阶段应进行一定数量的编码练习。本阶段的示例又分为两个层次：

- “直来直去”的程序（1.7 节）：这部分程序的核心是将实际问题（以绘图和数学计算为主）转换成代码进行解决。解决这类问题需要具备基本的编码知识和对问题的准确理解。本节的目的在于展示基本方法（如循环分支）的应用。
- “有技巧”的程序（1.8 节）：这部分程序用到了编程中最基本的模型和算法，即状态机、栈、队列和搜索等。这些技巧虽然基础但不简单。在初学阶段了解甚至掌握这些技术带来的好处是巨大的。

本阶段的最后引入了算法复杂度的初步概念及标记方法（1.9 节）。

第3阶段：1.10～1.11节

本阶段主要讲述了异常处理机制（1.10 节）。这是写出 Python 完整代码必不可少的知识。优秀的工程师能够全面地考虑各种意外情况并加以处理，从而设计出健壮的系统。在实际工作中，异常处理的重要性足以比肩本章其他全部内容，然而在初学阶段不便展开，故本部分篇幅短小，并设置为“节”归入第 1 章。

此外，本阶段还简要介绍了 PDB 调试工具（1.11 节）。

1.1 历　史

在莱布尼茨[①]（1646～1716）生活的时代，哲学家们开始研究计算问题。人们希望能够“发现一种普遍化的数学，能用来以计算代替思考”[②]。1847 年，英国数学家布尔建立了“布尔代数”，他创造了一套符号系统和运算法则，利用代数方法研究逻辑问题，奠定了数理逻辑的基础。19 世纪 70 年代，人类进入电气时代。从此人类文明开始不断寻找能够完成布尔运算的单元器件，从继电器到电子管、晶体管，再到量子比特。1936 年，英国数学家、逻辑学家艾伦·图灵提出了著名的图灵机模型，一台具有以下性质的机器：

- 无限线性存储器；
- 一个可移动的读写头；
- 内部具有有限个状态寄存器；
- 接受一系列有限指令进行动作。

图灵模型是计算机时代开启的标志。第二次世界大战则加速了人类在这条道路上的探索进程。战后，数学家从具体的计算问题（如密码破译、原子弹爆炸）中解放出来，思考计算机器的统一结构。1946 年，美籍匈牙利数学家冯·诺依曼提出了奠定现代电子计算机基础的结构：

- 拥有算术逻辑单元和寄存器的中央处理单元；
- 指令寄存器和程序计数器；
- 存储数据和指令的内存；
- 用于存储大量数据的外部存储器；
- 输入/输出系统。

这个结构被命名为冯·诺依曼结构（或普林斯顿结构）。从此科学家和工程师们大展身手。从微观上，科学家们不断寻找更好的开关器件，以实现布尔逻辑的级联运算。从宏观上，工程师们不断地构建出更加庞大的软/硬件体系。在 20 世纪 50～70 年代，以 Dijkstra 为代表的计算机科学家们系统性地发展和建设了这门学科。程序设计和操作系统等领域开始出现完整的理论体系。

随着计算机技术的发展，各种各样的程序设计语言被创造出来。同其他世界性人

① 戈特弗里德·威廉·莱布尼茨（德语：Gottfried Wilhelm Leibniz，1646 年 7 月 1 日－1716 年 11 月 14 日）：德国哲学家、数学家，历史上少见的通才，获誉为 17 世纪的亚里士多德。莱布尼茨有个显著的信仰，大量的人类推理可以被归约为某类运算，而这种运算可以解决看法上的差异：“精炼我们的推理的唯一方式是使它们同数学一样切实，这样我们能一眼就找出我们的错误，并且在人们有争议的时候，我们可以简单地说，让我们计算 calculemus，而无须进一步的忙乱，就能看出谁是正确的。”他曾断言：“二进制乃是具有世界普遍性的、最完美的逻辑语言”。以上内容来自维基百科“戈特弗里德·莱布尼茨”条目。

② 《西方哲学史》中译本 [38]第 11 章莱布尼兹。