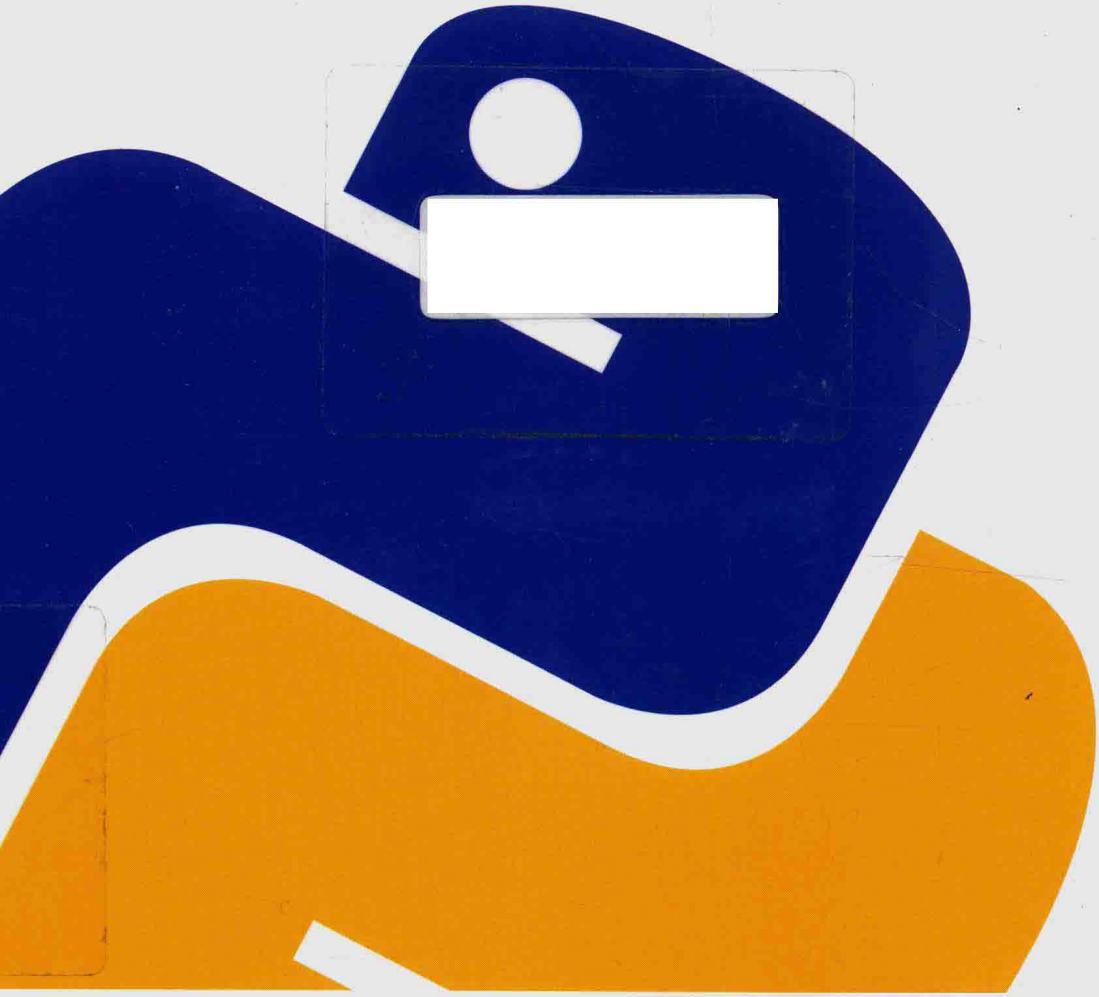
Python 3.5 程序设计教程

涵盖Python 3.0到3.5实用特性
详细介绍函数、模块、类与组件管理
深入探讨常用模块的应用与实践
包含装饰器、meta类的实践等进阶主题

林信良 著



清華大學出版社

Python

程序设计教程

林信良 著

清華大學出版社
北京

内 容 简 介

本书是作者在Python教学中学生在课程上遇到的概念、实战、应用等问题的经验总结。

本书基于Python 3.5编写，介绍了Python 3.0到3.5的实用特性。本书用简短精巧的范例程序贯穿全书，以学习笔记的写作方式进行编写，让读者在Python语言的交互环境中直接动手实战和体验，通过“实战”来掌握Python语言的核心知识和实战用法，并且特意标注了常用范例和重点范例，让读者可以根据自己的时间安排进行取舍。

本书既适合初学者学习，又能帮助有一定基础的程序员提升技能，还可作为相关培训的教材。

本书为碁峰资讯股份有限公司授权出版发行的中文简体字版本。
北京市版权局著作权合同登记号 图字：01-2016-8572

图书在版编目（CIP）数据

Python程序设计教程 / 林信良著.—北京：清华大学出版社，2017
ISBN 978-7-302-45786-2

Ⅰ.①P… Ⅱ.①林… Ⅲ.①软件工具－程序设计－教材 Ⅳ.①TP311.56

中国版本图书馆CIP数据核字（2016）第290544号

责任编辑：夏毓彦
封面设计：王 翔
责任校对：闫秀华
责任印制：何 芊

出版发行：清华大学出版社
网 址：http://www.tup.com.cn，http://www.wqbook.com
地 址：北京清华大学学研大厦A座 邮 编：100084
社 总 机：010-62770175 邮 购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市春园印刷有限公司
经 销：全国新华书店
开 本：190mm×260mm **印 张**：22.75 **字 数**：580千字
版 次：2017年1月第1版 **印 次**：2017年1月第1次印刷
印 数：1～3000
定 价：59.00元

产品编号：072376-01

序

你会不会做实验呢？

当然，对于程序设计来说，懂得做实验是件很重要的事。看到一个函数特性，编写个范例程序来实验看看；不懂某条语句及其语法的概念，编写个程序片段来实验看看，虽然这不是在做化学实验或物理实验，不过有时也会想实验看看，程序会不会像化学实验那样“炸”了……

关掉计算机之后，你会不会做实验呢？

愿意从事程序设计的人，按理来说，应该也是乐于做实验的人，那么现在就做个实验吧！关掉计算机、离开桌子，想想除了用计算机之外，还能在生活上做些什么实验？或者尝试看看其他事物，看看会有什么样的结果。

有没有对自己的人生做过实验呢？

这和计算机上做实验不同，对人生做实验需要耐心，没有人能保证何时能有结果，有时人生中看似毫不相关，甚至是失败的几个实验，却在某个时间点获得莫名其妙的成果。

有没有特意对未来的人生进行实验呢？

你回想起过去曾经有过的几次实验，也许算不上实验，只是在随波逐流的过程中，多少尝试过做些努力，若在无意识下曾经对人生做过的实验促成了现在的你，那么现在下意识地对人生做些实验，未来的自己会是什么样子呢？

程序设计很强调 Get your hands dirty（要勤写代码），别忘了，人生也需要 Get your hands dirty（亲力亲为）！

编 者

2016 年 9 月

改编说明

作为具有“胶水”美誉的程序设计语言——Python，它广受欢迎的原因众多，现只列其二：

一、清晰简洁的动态类型语言。让初学者可以把精力集中在编程的对象以及编程的逻辑思维上，不用去担心程序设计语言的语法、数据类型、数据结构等各种烦人的内容。对于没有面向对象程序设计基础的人而言，直接学 C++或 Java 可能困难较大，从 Python 语言入手不失为一种更好的选择。

二、博取众长，兼顾 C++的高效率和 Java 网页设计的灵活性。即便对于已经掌握了 C++或 Java 等语言的人员，在认识了 Python 的超级“粘合剂”的作用之后，都会对它的能力赞叹不已。

因此，Python 不仅在专业人员中流行，越来越多程序设计初学者也开始把 Python 作为自己的第一门程序设计语言来学习。在高等院校采用 Python 作为学生的第一门程序设计语言也成为了最新的趋势。

本书的结构与叙述风格既像传统的程序设计语言教材，又不像。这是因为本书没有通篇“枯燥乏味”的讲解程序设计语言的语法，而且以“学会说一种流利的语言，而不是成为这种语言的语法专家”的原则来展开。学好一门语言的最好方式不是背语法，而是不断地练习怎么“说”——本书用简短精巧的示范和范例程序贯穿全书，让读者在 Python 语言的交互环境中直接动手实践和体验，通过“实战”来掌握 Python 语言的核心知识和实战用法。

为了配合本书作为学习和培训 Python 语言的教材（包括自学），除了各个章节给读者提供实践范例程序和教学范例程序，我们还为每章课后练习提供了解答的参考程序，且提供源代码。

最后加一点，如果读者按照本书介绍的步骤安装和设置 Python 的命令行或者集成开发的运行环境，那么默认的工作目录是 C:\workspace。如果本书提供的范例程序出现不能运行的情况，请注意设置好 Python 的运行环境和工作目录。

最后祝大家学习顺利，早日对 Python 语言融合贯通，展示自己的聪明才智！

资深架构师 赵军

2016 年 10 月

导 读

这份导读可以让你更了解如何使用本书。

程序范例

本书的范例程序下载地址为：http://pan.baidu.com/s/1kVkCdEJ（注意区分数字和英文字母大小写）。如果下载有问题，请发送电子邮件至 booksaga@126.com，邮件主题设置为“求 Python 程序设计教程下载资源”。

本书许多范例都使用完整的程序实现来展现，当你看到以下程序代码范例：

basicio upper.py

```
import sys

src_path = sys.argv[1]
dest_path = sys.argv[2]
                                        ❶分别以'r'与'w'模式打开
with open(src_path) as src, open(dest_path, 'w') as dest:
    content = src.read()                ←❷使用 read()读取数据
    dest.write(content.upper())         ←❸使用 write()写入数据
```

范例开始的左边名称为 basicio，表示可以在范例文件夹的 **samples** 文件夹内，按各个章节的文件夹找到对应的 basicio 项目，右边名称为 upper.py，表示可以在项目中找到 upper.py 文件。如果程序代码中出现标号与提示文字，表示后续的内容中会有对应于标号和提示的更详细的说明。

建议读者每个项目范例都亲自动手编写，但如果有教学时间或实践时间不足的问题，本书也提供了建议进行的练习，如果在范例开始前发现有个 Lab 图标，例如：

game1 rpg.py

```
class Role:                                    ←❶定义类 Role
    def __init__(self, name, level, blood):
        self.name = name   # 角色名称
        self.level = level # 角色等级
        self.blood = blood # 角色血量

    def __str__(self):
        return "('{name}', {level}, {blood})".format(**vars(self))

    def __repr__(self):
        return self.__str__()

class SwordsMan(Role):    ←❷继承父类 Role
    def fight(self):
        print('挥剑攻击')

class Magician(Role):     ←❸继承父类 Role
    def fight(self):
```

```
        print('魔法攻击')

    def cure(self):
        print('魔法治疗')
```

表示建议动手实现这个范例进行上机实践，而且在范例文件 labs 文件夹中会有练习项目的基础，可以打开项目并完成项目中遗漏或必须补齐的程序代码或设置。

如果文中采用以下的程序代码排版，表示它是个代码段，主要展现程序编写时需要特别注意的片段。

```
class Account:
    ...
    def deposit(self, amount):
        if amount <= 0:
            print('存款金额不得为负')
        else:
            self.balance += amount

    def withdraw(self, amount):
        if amount > self.balance:
            print('余额不足')
        else:
            self.balance -= amount
```

对话框

在本书中会出现以下对话框：

提示 »»

针对课程中所提到的概念，提供一些额外的资源或思考方向，暂时忽略这些提示对课程并没有影响，如果有时间，对这些提示多加阅读、思考和讨论将对本书的学习有帮助。

注意 »»

对课程中所提到的概念，以对话框的方式特别呈现出必须注意的使用方式、陷阱或避开问题的方法，看到这个对话框时请集中精神仔细阅读。

附　录

附录中的内容不是本书编写的主要目的，纯粹是让有兴趣的读者知道有这类主题的存在，因而附录的内容简明扼要。附录 A 简要介绍使用 Python 3.3 之后内建的 venv 模块来建立虚拟环境，附录 B 简要介绍如何使用 Django 编写简单的 Web 应用程序。

联系作者

若有勘误和反馈等和本书有关的问题，可通过网站与作者联系：

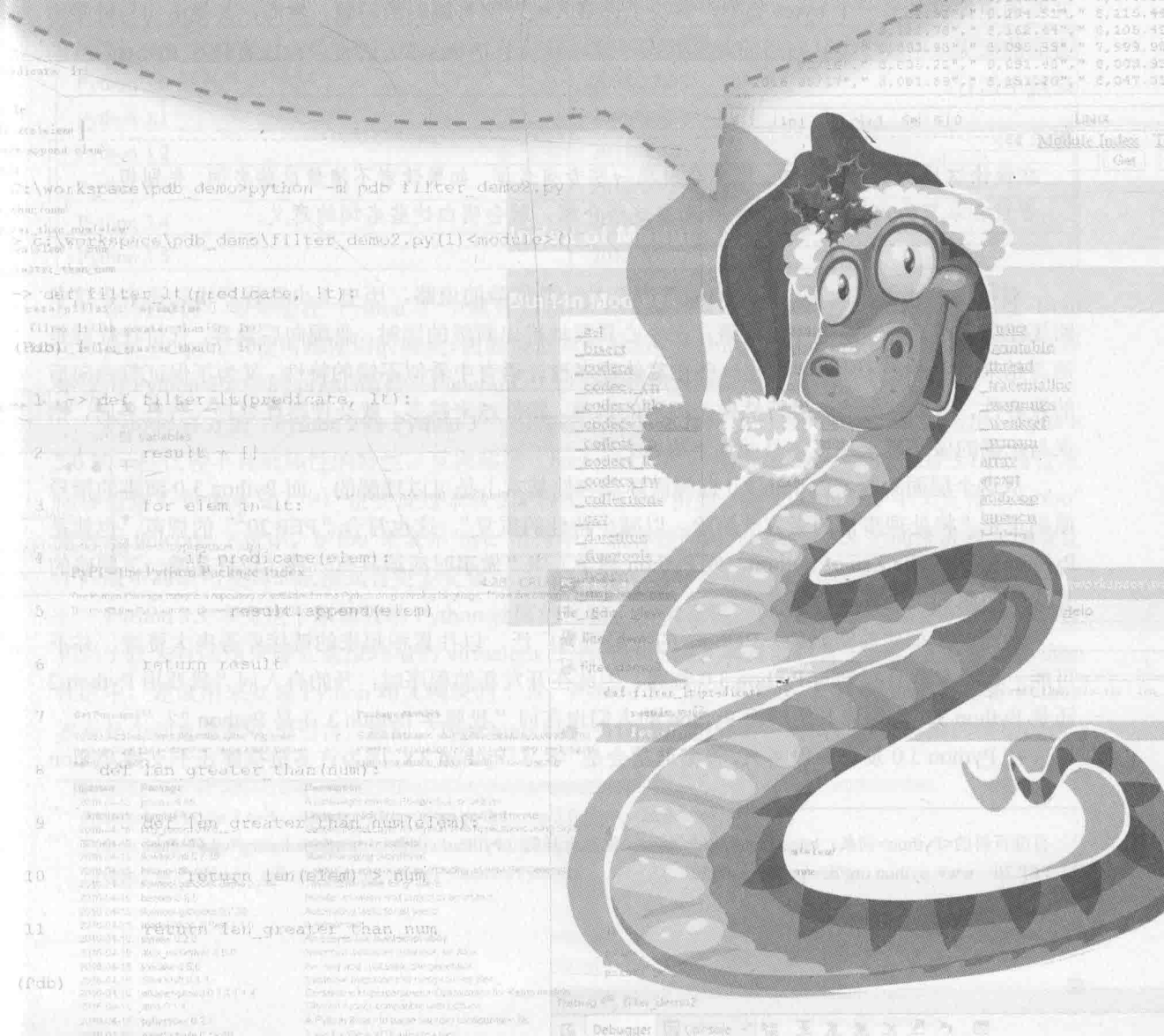
http://openhome.cc

目 录

第 1 章 Python 起步走

学习目标

- 选择 2.x 还是 3.x
- 初识 Python 资源
- 认识 Python 的实现
- 建立 Python 环境



1.1 认识 Python

Python 诞生于 1991 年，至今足足有二十几个年头了，算是一门“古老的”程序设计语言。Python 经过这么长时间的发展，现在要学习 Python，究竟要先认识些什么，接下来将为读者详细介绍。

1.1.1 Python 3 的诞生

正因为 Python 是一门古老的程序设计语言，它的应用极为广泛，在系统管理、科学计算、Web 应用程序、嵌入式系统等各个领域都可以看到 Python 的踪迹。然而，本书并不打算从它的诞生开始谈起，如果读者有兴趣，可以看看百度百科或者维基百科上的“Python[1]”词条。不过，就这个时间点来说，读者关心的应该不是 Python 的诞生，而是 Python 3 的发布。

时光暂且回到 2008 年 12 月 3 日，新出炉的 Python 3.0（也被称为 Python 3000 或 Py3K）中增添了许多人引颈期盼的新功能，其中最引人注目的是 Unicode 的支持，将 str/unicode 进行了整合，并明确地提供了另一个 bytes 类型，解决了许多人处理字符编码的问题。然而，其他语句与链接库方面的变更，也破坏了其向后的兼容性，导致许多基于 Python 2.x 的程序无法直接在 Python 3.0 的环境中运行。

提示 »»

在谈论这段历史的过程中，难免会出现一些专有名词，如果读者不清楚这些名词，先别担心，看过各章节的内容之后，回头再来看这些介绍，就会明白这些名词的意义。

对程序设计语言而言，破坏向后兼容性是一条危险的道路，历史上少有程序设计语言走这条路还能获得成功。许多程序设计语言在小心翼翼地推出新版的同时，兼顾向后兼容，代价往往就是语言越来越臃肿，有时想要吸收一些在其他程序设计语言中看似不错的特性，又为了保证符合向后的兼容性，结果总会将这类特性做些畸形的调整。特性越来越多，就会使得在处理一件任务时，错误与正确的做法越来越多，且并存于语言之中。

从这个层面来看，Python 3.0 选择破坏兼容性基本上是可以理解的。而 Python 3.0 演进的指导原则正是“将处理事情的老方法删除，以减少特性的重复”，这也符合“PEP 20[2]”的规范，也就是 Python 哲学（The Zen of Python，或 Python 禅学）中“做事时应该只有一种（也许是唯一）明确的方式”的原则。

然而，正如先前谈到的，Python 的应用极为广泛，以往累积起来的链接库等庞大资源，并不可能一朝一夕就升级到与 Python 3.0 兼容，因此在开发新的程序时，开始有人问“我要用 Python 2 还是 Python 3？”打算开始学习 Python 的人们也在问“我要学 Python 3 还是 Python 2？”

在 Python 3.0 发布后没多久，答案通常会是“学习 Python 2.x，因为许多链接库还不支持 Python

---

1 百度百科的<Python>词条：http://baike.baidu.com/；维基百科的〈Python〉词条：zh.wikipedia.org/wiki/Python

2 PEP 20：www.python.org/dev/peps/pep-0020/

3.0!”然而，随着时间的推移，答案渐渐地变得难以选择，许多介绍 Python 的入门文档或书籍，也不得不同时介绍 Python 2 与 Python 3 两个版本；尽管有“2to3[1]”这个工具声称可以将 Python 2 的程序代码转换为 Python 3，但是它也不能发现所有问题；渐渐地，甚至有了“Python 3 is killing Python[2]”这类文章出现，预测着 Python 社区将会分裂，甚至预测现有的拥护者也可能会离开 Python。

1.1.2　从 Python 3.0 到 3.5

尽管破坏了向后兼容性的程序设计语言多半不会有什么好结果，然而，就这几年来从 Python 3.x 与 2.x 的发展来看，过程与那些失败了的程序设计语言不太一样。

官方的推动

首先，Python 本身以每隔一年左右推出一个 3.x 版本推进着，过程也并不是官方一厢情愿地推进，而是不断地倾听着 Python 社区的声音，不断地为兼容转换做出努力。表 1.1 为 Python 3.0 到 3.5 发布的日期。

表 1.1　Python 3.0 到 3.5 发布的日期

版本	发布的日期
Python 3.0	2008/12/03
Python 3.1	2009/06/27
Python 3.2	2011/02/20
Python 3.3	2012/09/29
Python 3.4	2014/03/16
Python 3.5	2015/09/13

举例来说，如果想在 Python 2 中就开始使用 Python 3.x 的一些特性，可以试着通过 from __future__import 来使用想使用的模块，例如最基本的 from __future__import print_function 就可以开始使用 Python 3.x 中的 print()函数（Function），以兼容方式来编写输出语句。

Python 官方的“Python 2 or Python 3[3]”也整理了许多兼容转换的相关资源。其中指出，Python 3.0 的一些比较不具破坏性的特性，反向移植（backport）到了 Python 2.6 中，Python 3.1 的特性反向移植到了 Python 2.7 中；也会反过来从 2.x 移植至 3.x。例如，在 Python 3.3 中又支持了 u"foo" 来表示 unicode 字符串，b"foo"来表示 byte 字符串，兼容性同时在 2.x 与 3.x 之间推进着，试着让不同版本的语句及其语法格式有更多交集。

Python 3.x 本身也不断地吸纳 Python 社区的经验，举例来说，Python 3.3 中包含了 venv 模块，相当于过去社区用来建立虚拟环境的 virtualenv 工具；Python 3.4 本身就包含了 pip，这是过去 Python 社区中，建议用来安装 Python 相关模块的工具；Python 3.5 进一步纳入了 Type hints，尽管 Python 本身是个动态数据类型语言，然而，Type hints 特性有助于静态分析、重构、运行时刻的类型检查，

1　Automated Python 2 to 3 code translation：docs.python.org/3.0/library/2to3.html

2　Python 3 is killing Python：blog.thezerobit.com/2014/05/25/python-3-is-killing-python.html

3　Python 2 or Python3：wiki.python.org/moin/Python2orPython3

这对大型项目开发有显著帮助，而且对现有程序代码不会有影响。

社区的接纳

尽管 Python 3.x 本身不断在兼容性与新特性上发布好消息，但是 Python 社区不买账也是徒劳无功，所幸实际情况并非如此。

在 Python 官方的持续推动之下，许多基于 Python 2.x 的链接库或框架在这段期间不断地往 Python 3.x 移植，例如 Web 快速开发框架 Django，在笔者著书期间，新版本已经支持至 Python 3.4，官方 Tutorial 也是基于此版本而改写的，一些科学运算软件包，像 Numpy、SciPy 等也有了支持 Python 3.x 的版本。

想要知道其他链接库是否支持 Python 3.x，有个“PYTHON 3 WALL OF SUPERPOWERS[1]”可以作为不错的参考资料，其中列出了 200 个链接库，174 个标示为绿色，表示支持 Python 3.x，在 2014 年左右，仍有 35 个被标示为不支持 Python 3.x 的红色，当前只剩下 26 个链接库是红色状态。

一个现实的问题是，新系统开发时究竟要基于 Python 2.x 还是 Python 3.x？最好的方式是编写出能同时兼容 2.x 与 3.x 版本的程序代码，除了要建立良好的程序代码编写习惯之外，Python 社区中还有着 six[2]这类的软件包，可以作为编写出兼容 Python 2.7 和 Python 3.x 版本的程序代码的基础。

另一方面，由于 Python 2.7 是 Python 2 的最后一个版本（不会再有 Python 2.8），且预计只支持到 2020 年，在停止支持之前的这段日子，也不再加入新特性，只会针对程序中的错误（bug）或安全问题进行修正，因此除许多链接库已经在进行移植之外，有些操作系统也开始进行相对应的移植工作。

像 Linux 系统，目前多半同时默认加载了 Python 2.x 与 Python 3.x。以 Ubuntu 为例，从 13.04 之后的版本开始就默认加载了 Python 3.x，Ubuntu 也在持续去除系统中对 Python 2.x 的依赖，在其未来的计划中，希望有朝一日能够全面采用 Python 3.x，而不再默认加载 Python 2.x。

当然，一定还会有人死守着 Python 2.x，宣称未来绝不会支持 Python 3.x 的软件包、链接库或应用程序。身为 Python 开发者，将来也可能遇到必须面对这些链接库的时候，然而更重要的是，无论现阶段个人偏好如何，在遇到类似“我要学习 Python 2.x 还是 Python 3.x？”的问题时，答案不应只是简单的“学习 Python 2.x，因为许多链接库还不支持 Python 3.x”，而是应该针对自己或客户的需求，进行全面性的调查，就像在选择一门程序设计语言，或者是调查某个链接库是否可以采用时，必须进行多方面的考虑，像了解其更新（Update）的时间、更新日志（Changelog）、修正问题（Issue）的频度、作者身份等。

当然，从 Python 2.7 终究会在 2020 年停止支持，以及官方在 Python 3.x 上的推动和 Python 社区支持这两个方面来看，未来 Python 的生态圈，应会持续接纳 Python 3.x，因而就学习 Python 来说，可以先学 Python 3.x，因为有了 Python 3.x 的基础，将来若有必要面对或学习 Python 2.x，就不会是件难事。

---

1 PYTHON 3 WALL OF SUPERPOWERS：python3wos.appspot.com/

2 six：pypi.python.org/pypi/six