**如何自学 Python(干货合集)**

**Learn Python The Hard Way**  
<http://learnpythonthehardway.org/book/>  
零编程基础学python，浅显易懂，操作性强。中文翻译版本：  
**笨办法学 Python**  
<http://lpthw-cn.ducktypist.com/en/latest/index.html>

其实python非常适合初学者入门，上手很容易。  
我就是完全通过网上资源学了python的。  
最大的是3点经验：  
1.找一本浅显易懂，例程比较好的教程，从头到尾看下去。不要看很多本，专注于一本。把里面的例程都手打一遍，搞懂为什么。  
2.去找一个实际项目练手。我当时是因为要做一个网站，不得已要学python。这种条件下的效果比你平时学一门新语言要好很多。所以最好是要有真实的项目做。可以找几个同学一起做个网站之类。  
3.最好能找到一个已经会python的人。问他一点学习规划的建议，然后在遇到卡壳的地方找他指点。这样会事半功倍  
  
推荐几个网上的资源：  
  
[简明Python教程](http://pan.baidu.com/share/link?shareid=2635764419&uk=204484850)  
[与孩子一起学编程](http://pan.baidu.com/share/link?shareid=2637964168&uk=204484850)  
[Head First Python 中文版](http://pan.baidu.com/share/link?shareid=2640339432&uk=204484850)  
[笨办法学Python](http://pan.baidu.com/share/link?shareid=2641642838&uk=204484850)  
[Dive.Into.Python中文版(附课程源码)](http://pan.baidu.com/share/link?shareid=2645575441&uk=204484850)  
[Python核心编程](http://pan.baidu.com/share/link?shareid=2647476085&uk=204484850)  
[深入理解Python](http://pan.baidu.com/share/link?shareid=2730288035&uk=204484850)  
[Python标准库](http://pan.baidu.com/share/link?shareid=2731946885&uk=204484850)  
[Python编程指南](http://pan.baidu.com/share/link?shareid=2733463562&uk=204484850)  
  
[diango\_book中文版](http://pan.baidu.com/share/link?shareid=2734895267&uk=204484850)  
  
解压密码均为：[http://crossin.me](http://crossin.me/)  
  
======================  
在线教程：  
简明Python教程：[简明 Python 教程](http://woodpecker.org.cn/abyteofpython_cn/chinese/index.html)  
（较新版）：[A Byte of Python](http://zhgdg.gitcafe.com/static/doc/byte_of_python.html)  
  
Dive into Python：[Dive Into Python](http://woodpecker.org.cn/diveintopython/toc/index.html" \t "_blank)

**阶段一：前辈领进门。**  
第一眼是一个前辈给我看了看Python的代码，因为自己最早学习的是Java语言，第一眼看到Python的时候，前辈说，“Python是面向对象的”，然后就 打印了几句代码。可我怎么也看不出到底哪里是“面向对象”的。前辈说“在Python里，一切皆对象”，我才有点领悟的说道：“原来把什么看成对象，就是面向对象”，哈哈。  
  
**阶段二：开始看的是《Dive into Python》、《Leaning python》、《Python Doc》。**  
因为有很多语言学习经历，很快就完成了初步语法和使用学习。太复杂的特性还是在使用中逐步掌握的。  
  
**阶段三：开始使用Python做自己的一些日常工作。**  
比如Python搜索文件，Python批量处理等，使用最多的还是re模块和socket相关模块。写了大量的例子，让自己对Python更加喜爱，也更加熟悉。此时翻阅最多的是《Python Doc》的指南。不管是语言参考、库参考、Demo参考，都有大量可使用的内容，内容质量很高堪比JDK。  
  
**阶段四：生产上马。**  
开始使用Django，Flask，Tornado开发一些web应用，写一些日常使用的工具包等。逐步提升设计能力，和整体代码的管理能力。  
  
**阶段五：更合理的分配好C，Java，Python三门语言各自擅长的部分。**  
把合适的语言用到合适的地方。尽管一门语言有时候可以搞定所有的，但用擅长的语言解决合适的问题才是效率最高的。这也是“Python的大道至简”的理念带给我的帮助和认识。  
  
**PS：其中过程中主要的一些方法：**  
1、看书。学习的基础。  
2、自己本地练习。编程还是要实践出真知。  
3、资料查询。google，stackoverflow等多关注。  
4、交流。各种论坛上的python group，论坛。最早去的CU，JavaEye，不过现在貌似去的少了。google group 必须订阅。  
**学习+实践+总结，掌握语言的法宝。**

学Python，看的第一篇文章来自42区，推荐大家看看[网站开发 . 漫游指南](http://matrix.42qu.com/" \t "_blank)。  
多关注下牛人，多实践。  
  
更新2013-12-17  
前段时间整理了一篇博客，搜集了学习Python入门、数据库和Web开发的一些资料，原文见[http://dylanninin.com/blog/2013/11/23/python\_resource.html](http://dylanninin.com/blog/2013/11/23/python_resource.html" \t "_blank)，这里摘抄一下。

**入门和基础**

Python有Python 2.x 和Python 3.x 之分，争论很多，见[Python2orPython3](https://wiki.python.org/moin/Python2orPython3" \t "_blank)。初学者不用考虑这个问题，可以从Python 2.x入门，之后再讨论会多一些理性。

* [洪强宁：Python于Web 2.0网站的应用](http://www.slideshare.net/hongqn/qcon2010-3881323)：豆瓣网洪强宁在QCon北京2010中的技术分享，若不能访问，请自备梯子；另外，豆瓣的[阿北](http://www.douban.com/people/ahbei/" \t "_blank)很值得关注。
* [A Byte of Python](http://swaroopch.com/notes/python/)：即简明Python教程，边看边练习就对Python有基本的了解，可以轻松存活。
* [Google's Python Lessons](http://blog.hartleybrody.com/google-python/)：Google出品的Python教程，值得信赖。
* [Python Documentation](http://www.python.org/doc/)：Python在线文档，若嫌枯燥，可以直接看[Python Standard Lib](http://docs.python.org/2/library/" \t "_blank)。
* [42区：python入门指引](http://matrix.42qu.com/10757179)：江湖人称[张教主](http://zuroc.42qu.com/)的Python入门指引，除开[《Python核心编程》](http://book.douban.com/subject/3112503/" \t "_blank)。严格来说，这是张教主的一份Python资料索引。
* [CS61A: SICP with Python](http://www-inst.eecs.berkeley.edu/~cs61a/fa11/61a-python/content/www/index.html)：作为计算机相关人士，[SICP](http://en.wikipedia.org/wiki/Structure_and_Interpretation_of_Computer_Programs" \t "_blank)都不了解，实在很惭愧，所以来还债了。这应该是起源于[MIT的SICP](http://mitpress.mit.edu/sicp/" \t "_blank)教程，一个用Python，一个用Scheme。
* [看到一个有趣的python的招聘测试](http://www.douban.com/group/topic/28872729/)：同学发的一个贴，总结了一下应聘中遇到的问题，并给出了相应的资料来学习。如果想检测下自己的Python水平，请自觉移步[北京视讯天下的开发测试](http://www.video-tx.com/devtest/test.html" \l "/home" \t "_blank)。
* MOOC们：在线公开课很多，见[课程图谱](http://coursegraph.com/search_results/python" \t "_blank)；仅在[Codecademy](http://www.codecademy.com/tracks/python)上学了一点；现在希望能够每天跟进一些感兴趣的课程，多多学习。
* 书：去年6月份一同事离职，我买了两本[黑客与画家](http://book.douban.com/subject/6021440/" \t "_blank)，一本送给他(博客[一起去看海](http://www.ooobj.com/" \t "_blank))，另一本给部门老大；他回送了一本[Python源代码剖析](http://book.douban.com/subject/3117898/" \t "_blank)，看书名就知道是讲底层原理的，有机会啃啃。
* 源代码：直接到[Github](https://github.com/search?q=python&ref=cmdform" \t "_blank)上去找吧。

**Python与数据库**

在项目开发中，数据库应用必不可少。这里汇总下目前接触过的数据库和使用教程。

SQL，计算机出身的人应该都学过，若有疑问可以参考[Wikipedia SQL](http://en.wikipedia.org/wiki/SQL" \t "_blank)，并使用SQLite做下练习。

* [PEP249: DB API](http://www.python.org/dev/peps/pep-0249/)：数据库访问接口规范，当时还做了一份[笔记](http://dylanninin.com/blog/2012/11/16/python_pep249.html" \t "_blank)。
* [SQLite](http://www.sqlite.org/)：教程见[SQLite Python Tutorial](http://www.tutorialspoint.com/sqlite/sqlite_python.htm" \t "_blank)；若要深入了解，推荐[The Definitive Guide to SQLite](http://book.douban.com/subject/5392299/" \t "_blank)；看此书时，做了一些好句子、段落的摘要，见[Sentences in SQLite 3](http://dylanninin.com/blog/2013/11/10/sentences_in_sqlite.html" \t "_blank)。
* [MySQL](http://www.mysql.com/)：教程见[Python MySQL Database Access](http://www.tutorialspoint.com/python/python_database_access.htm" \t "_blank)。
* [Oracle Database](http://www.oracle.com/us/products/database/overview/index.html)：有可能这是最庞大的数据库，所以需要专职的DBA。官方教程[The Mastering Oracle+Python Series](http://www.oracle.com/technetwork/articles/dsl/mastering-oracle-python-1391323.html" \t "_blank)，快速入门教程[cx\_Oracle Quick Start](http://dbaportal.eu/sidekicks/sidekick-cx_oracle-code-paterns/" \t "_blank)。
* [SQLAlchemy](http://www.sqlalchemy.org/)：Python的ORM标准和框架，解决面向对象编程和关系数据库模式不匹配的问题。

数据库理论除了SQL，还有另一派NoSQL。

关于常见NoSQL产品的比较见[Main NoSQL Database Comparison](http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis" \t "_blank)；若果需要一些NoSQL的理论知识和基本概念，见[The NoSQL Ecosystem](http://www.aosabook.org/en/nosql.html" \t "_blank)、[Big Data与NoSQL](http://www.slideshare.net/pavlobaron/big-data-nosql-efs11-pavlo-baron" \t "_blank)。目前仅用过[MongoDB](http://www.mongodb.org/" \t "_blank)和[Redis](http://redis.io/" \t "_blank)。

* MongoDB：如果熟悉SQL，MongDB的学习成本会很低；相关资料见[Python Language Center in MongoDB](http://docs.mongodb.org/ecosystem/drivers/python/" \t "_blank)；用过的两个驱动：1）[PyMongo](http://api.mongodb.org/python/current/)，提供了类似Mongo Shell的接口；2）[MongoEngine](http://mongoengine.org/): A Python Object-Document-Mapper for working with MongoDB，即MongoDB的'ORM'框架，此时变成了'ODM'，[MongoEngine on Github](https://github.com/hmarr/mongoengine" \t "_blank)。
* Redis：Redis需要一些学习成本，入门推荐[The Little Redis Book](http://openmymind.net/2012/1/23/The-Little-Redis-Book/" \t "_blank)；用过的Python客户端驱动[Redis-py](https://github.com/andymccurdy/redis-py" \t "_blank)；更多客户端见[Redis Clients](http://redis.io/clients" \t "_blank)。
* NoSQL建模：SQL发展了几十年，有很成熟的建模技术，那么NoSQL呢，见[陈皓：NoSQL数据建模技术](http://coolshell.cn/articles/7270.html" \t "_blank)，原文[NoSQL Data Modeling Techniques](http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/)。
* 更多资料：NoSQL英文站点见[NoSQL Database](http://nosql-database.org/" \t "_blank)；NoSQL中文论坛见[NoSQL Fan：关注NoSQl相关的新闻和技术](http://blog.nosqlfan.com/" \t "_blank)。NoSQL Fan中，MongoDB和Redis资料很多，已经形成了资料专题，包括介绍、内部实现、应用与优化、新闻等，总能发现你想要的东西：1）[NoSQL Fan：Redis资料汇总专题](http://blog.nosqlfan.com/html/3537.html)；2）[NoSQL Fan: MongoDB资料汇总专题](http://blog.nosqlfan.com/html/3548.html)。

**Python与Web开发**

Python的Web框架众多，见[Web Frameworks for Python](https://wiki.python.org/moin/WebFrameworks" \t "_blank)，总有一款适合你或你的项目，实在不行，请动手打造自己的框架；为什么会有这么多框架呢，见[Why so many Python wen frameworks?](http://bitworking.org/news/Why_so_many_Python_web_frameworks" \t "_blank)。

* [Web.py](http://webpy.org/)：已故[Aaron Swartz](http://www.aaronsw.com/" \t "_blank)的框架。一句话介绍"web.py is a web framework for Python that is as simple as it is powerful. web.py is in the public domain; you can use it for whatever purpose with absolutely no restrictions."。接触的第一个Web框架，后来模仿MovableType，写了一个简单的博客，见[Blog on Github](https://github.com/dylanninin/blog/" \t "_blank)，在线demo [Pastime Paradise](http://ec2-54-251-227-120.ap-southeast-1.compute.amazonaws.com/)；碰到的坑点是模板中嵌套Python代码一直有缩进问题。
* [Flask](http://flask.pocoo.org/)：一句话介绍"Flask is a lightweight web application framework written in Python and based on the Werkzeug WSGI toolkit and Jinja2 template engine. It is BSD licensed. Flask is called a microframework because it keeps the core simple but extensible"。因为工作变动，练习过一段时间的Flask，以便熟悉开发的工具链；如果没有Rails，我想这才是自己首选的Web开发框架: simple but extensible and for fun。实战教程[The Flask Mega-Tutorial](http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world" \t "_blank)。
* [Tornado](http://www.tornadoweb.org/en/stable/):来自[FriendFeed](http://friendfeed.com/" \t "_blank)的异步框架，FriendFeed被Facebook收购后开放了源代码，见[Tornado on Github](https://github.com/facebook/tornado" \t "_blank)。一句话介绍"Tornado is a Python web framework and asynchronous networking library, originally developed at FriendFeed"。Tornado主要特点是[non-blocking](http://en.wikipedia.org/wiki/Non-blocking_I/O" \t "_blank)，如果你想开发[real-time](http://en.wikipedia.org/wiki/Real-time_web" \t "_blank)的Web应用，Tornado是一个不错的选择。
* [Django](https://www.djangoproject.com/)：一句话介绍"Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design."，接触不多，作为全栈式框架，听说它的组件都是Made in Django。更多资料见[Django资料](http://haoluobo.com/trac/wiki/Django" \t "_blank)。

**IDE**

* [Vim](http://www.vim.org/) + [Python Mode](https://github.com/klen/python-mode)。目前就用这个方案，很方便，直接引用同学的[效果图](http://www.douban.com/photos/photo/1975213817/" \t "_blank)。关于Vim资料和讨论，请移步[Vim资料大全](http://wiki.hotoo.me/Vim.html" \t "_blank)；关于Vim更多插件和演示，请移步[k-vim on Github](https://github.com/wklken/k-vim" \t "_blank)；关于Git，Github，请移步本站[Free Blog with Github Pages](http://dylanninin.com/blog/2013/11/02/free_blogs.html" \t "_blank)。

**社区**

* [啄木鸟社区](http://wiki.woodpecker.org.cn/moin/)
* [42区：网站开发.漫游指南](http://matrix.42qu.com/)
* [Python on V2EX](https://www.v2ex.com/go/python)
* [Python4cn](http://www.simple-is-better.com/)
* [豆瓣Python小组](http://www.douban.com/group/python/)
* [哲思社区](http://www.zeuux.com/)

**周刊**

* [Python Weekly](http://www.pythonweekly.com/)：每周更新，包括Python相关的文章、教程、演讲、书籍、项目、工作等。
* [Pycoder's Weekly](http://www.pycoders.com/)：与Python Weekly类似，两者可以互为补充，了解过去一周动态。
* [码农周刊](http://weekly.manong.io/): [developerWorks](http://weibo.com/developerworks" \t "_blank)出品的周刊，来自国人的分享。可以先看[为什么要做《码农周刊》？](http://blog.manong.io/why-create-a-weekly-newsletter-for-programmers/" \t "_blank)；接着[《码农周刊》用到的一些技术](http://blog.manong.io/technologies-we-use/)； 再接着[周刊回顾](http://weekly.manong.io/issues/)。这里不仅仅是Python。

我最先看的是《[简明 Python 教程](http://woodpecker.org.cn/abyteofpython_cn/chinese/)》来熟悉一下语法，自己敲了一下代码，发现python很容易上手，渐渐喜欢上它。接着买了一本《python基础教程》，跟着书中的例子深入学习。这本书讲得很详细，浅显易懂。  
当开始熟悉python的时候，尝试自己写一些脚本或软件来干一些有趣的事情。后来图像处理课程作业我也用python来完成。  
与此同时，我坚持写[博客](http://liamchzh.github.io/)来记录一下自己的学习心得。  
现在我开始学习用python的web应用。这才是python擅长的领域。它有很多好用的框架例如django,tornado等，可以轻松地做一个网站，用python做网站后端也是一个越来越流行的趋势。  
  
学习python，最重要的是要写出pythonic的代码，这不是一朝一夕的事情。我喜欢python，就是因为它的简洁和强大。不要重复造轮子，学会优雅地使用python。[Pythonic到底是什么玩意儿？](http://blog.csdn.net/lanphaday/article/details/2762251" \t "_blank)  
  
附我看过的觉得不错资料  
[PyCoder’s Weekly 中文翻译](http://pycoders-weekly-chinese.readthedocs.org/en/latest/index.html)  
[学好Python必读的几篇文章](http://developer.51cto.com/art/201203/321041.htm)  
也应该订阅一些python大牛的博客[有哪些 Python 大牛的 blog 推荐？](http://www.zhihu.com/question/20474582/answer/15227114)  
和邮件列表<http://groups.google.com/group/python-cn/topics>

我是自学的Python。从对Python一无所知，到在博客上写Python相关的系列文章（[Python快速教程](http://www.cnblogs.com/vamei/archive/2012/09/13/2682778.html)），前后有将近三年的时间。期间有不少门槛，但也充满乐趣。乐趣是自学的最大动力。Python是一个容易编写，又功能强大的动态语言。使用Python，可以在短短几行内实现相当强大的功能。通过自己写一些小程序，迅速的看到效果，发现问题，这是学习Python最便利的地方。  
  
在学习Python之前，可以了解一下Python的特点和设计理念([Python简史](http://www.cnblogs.com/vamei/archive/2013/02/06/2892628.html))。在设计之初，Python就试图在复杂、强大的C和方便、功能有限的bash之间，找到一个平衡点。Python的语法比较简单，用起来很方便，因此有些人把它当作脚本语言使用。但Python要比普通的脚本语言功能强大很多。通过良好的可拓展性，Python的功能相当全面，应用面很广：web服务器，网络爬虫，科学运算，机器学习，游戏开发…… 当然，天下没有免费的午餐，也没有完美的语言，Python为了达到上述两点，有意的牺牲了Python的运行速度。如果你是在编写高业务量、运算量的程序，可能Python并不是最好的选择。  
  
-----  
  
Python的主体内容大致可以分为以下几个部分：

1. 面向过程。包括基本的表达式，if语句，循环，函数等。如果你有任何一个语言的基础，特别是C语言的基础，这一部分就是分分钟了解下Python规定的事。如果你没有语言基础，建议从[Python Programming](http://book.douban.com/subject/5407267/" \t "_blank)为参考书。这本书是计算机导论性质的教材，不需要编程基础。
2. 面向对象，包括面向对象的基本概念，类，方法，属性，继承等。Python是面向对象的语言，“一切皆对象”。面向对象是很难回避的。Python的面向对象机制是相对比较松散的，不像Java和C++那么严格。好处是容易学，容易维护，坏处是容易犯错。
3. 应用功能，包括IO，数据容器如表和词典，内置函数，模块，格式化字符串等。这些在其它语言中也经常出现，有比较强的实用性。
4. 高级语法，上下文管理器，列表推导，函数式编程，装饰器，特殊方法等。这些语法并不是必须的，你可以用前面比较基础的语法实现。学这些高级语法的主要原因是：它们太方便了。比如列表推导一行可以做到的事情，用循环结构要好几行才行。

学习Python主体最好的参考书是[Learning Python](http://book.douban.com/subject/3988517/" \t "_blank)，它非常全面，满满的都是干货。虽然很厚，读起来并不难读。另一个是参考官网的教程[Python.org](http://python.org/" \t "_blank)  
  
-----  
  
Python号称“Battery Included"，也就是说，功能都已经包含在了语言中。这一自信，主要来自Python功能全面的标准库。标准库提供了许多功能模块，每个模块是某一方面功能的接口，比如文件管理，操作系统互动，字符处理，网络接口，编码加密等等。  
  
[The Python Standard Library](http://docs.python.org/2/library/)中，你可以看到标准库模块的列表。这里也是标准库最好的学习资料。如果想找书，我只看到过两本关于标准库的：  
[Python Essential Reference](http://book.douban.com/subject/3273420/)  
[The Python Standard Library by Example](http://book.douban.com/subject/6540551/)  
  
说实话，这两本都不算很好的标准库教材，而标准库的参考书也确实很难写。因为标准库只是调用功能的接口，最终实现的是Python和系统的互动。这需要很强的系统知识，比如文件系统知识，进程管理，http原理，socket编程，数据库原理…… 如果这些知识都已经准备充分，那么标准库学起来完全没有难度。然而，这些背景知识的学习并非一朝一夕的事情。  
  
更深入的Python学习也是如此，需要大量的背景知识，而不是Python自身。如果你对Python的编译和运行机制感兴趣，你可以往Python底层这一深度挖。如果你对应用感兴趣，你可以多学习几个自己用的上的第三方包。学到这个时候，就是要自己探索的广阔空间了。  
  
-----  
  
基本上，学过主体内容之后，Python还是要靠做项目来练习。有不少小练习题类型的资料，比如[Python Cookbook](http://book.douban.com/subject/1418172/" \t "_blank)。但更好的方式是自己去想一些应用场景，用Python来解决。Python功能全面，所以不要担心自己想的问题Python解决不了 (基本上Python解决不了的问题，别的语言也没戏)。比如我学习多线程的动力，就因为要并行的下载大量的文件。基本上一个项目下来，会用到Python好几块的内容，知识会特别巩固。  
  
最后，和其它任何知识的学习一样，笔记和总结很重要。在看参考书和看网页时，可以做一些笔记。等到学了一段时间后，可以把笔记整理成更有条理的参考卡片(reference card)，或者写博客。这也是我写“[Python快速教程](http://www.cnblogs.com/vamei/archive/2012/09/13/2682778.html)”的主要原因。另外这个教程内容的顺序，也是我认为的比较合理的学习顺序，仅供参考。

从快毕业的时候在图书馆里借来第一本有关python的书算起，接触python的时间也不过半年有余。时间真的很短，很难有什么经验之谈，自己至今也仍有许多需要学习的地方。不过对于怎么入门这一块，倒是颇有感触。在这里记录下来，也许能对后人有所帮助吧~

我是怎么开始了解python

快毕业的时候，在中南的图书馆里瞎逛，偶然之间看到这么一本书，《可爱的python》。第一眼看上去，只是觉得书名还挺新颖的，反正也是闲着，抽出来看看吧。“人生苦短，我用python”，这是我在封面上看到的第一句话，这感叹句实在太吸引眼球，以至于这么长时间后，我早忘了书中讲的什么内容。留在脑海中的就只有封面上的这句话和作者的前言。

当时看完前言部分，我就感慨良多。一本好的编程入门书，不应该是一上来就告诉你怎么写Hello World，给你介绍变量、函数、控制流 blablabla...，而是作者站在一个朋友的角度来和你谈心，告诉你他自己学习这门编程语言的经历，他自己所体会到的这门编程语言的魅力在哪里，有哪些优点和不足之处，怎样能够更快更好地熟悉这门语言。这感觉就和当初学C++时候读的第一本书《Thinkng in C++》一样。作者提到，由于python这门语言的特殊性，对它的学习并不必拘泥于传统的教科书式的学习方式，而是重点在“使用”中学习，其基本思想就是用最短的时间掌握python最基础最核心的语法，然后在使用中碰到具体的问题时候，再去主动学习相关知识。这个观念对我的影响很深，可以说，回顾自己的历程，基本就是按照这个原则来的，而且收获确实很多。

下面就结合我自己的学习经历，谈谈刚入门时候的基本原则。

1. **你只需要掌握最基础的**

刚开始学习python的时候，可能会查看许多书，这些书为了能够涵盖得尽量全面，往往会涉及语言方方面面的细节。但是，*并不是每一个知识点都是你所需要的*。一开始你只需要掌握最基础的那部分知识。你可能会问，“我哪知道哪些是最基础的东西呢？” 我觉得，一个很简单的判断方法就是，拿起书都第一遍的时候，如果你能硬著头皮看下去并且能够理解里面所讲的内容，那很好，这就是最基础的。如果看了第一遍后云里雾里，鬼才知道哪天会用得上这些东西。OK，专门找个小笔记本，记下这部分内容方便以后查阅，然后，跳过这部分。我在第一次看decorator装饰器这个部分的时候实在看不下去，也不知道可能会有啥用，果断跳过，最近上高性能计算的课，学习下cuda的python接口时，里面都是装饰器修饰的函数，才又好好学习来一下，结合来自己的实际问题，这样理解起来也就更深入。

1. **脚踏实地，出来混，迟早是要还的**

记住，前面你跳过的那些问题，迟早是会冒出来的。你自己得清醒地意识到，这种”刻舟求剑“式的做法，是存在一些弊端的，虽然大多数时候，这些弊端不过是自己动手来实现一些别人已经实现来的东西，多花点时间精力罢了，但还有的时候，你可能会付出沉重的代价。类似的教训实在太多，比如看书的时候觉得itertools这个包没有太大用就跳过了，后来有一天要实现个排列组合的算法时花了很长时间来实现，结果偶然一天看到这货居然内置在iterrools里了；还有迭代器和生成器那部分，一开始以为自己可能用不到，后来要对一堆很大的文本做分析时候才发现内存不够了......所以说，出来混，迟早是要还的，那些跳过了东西，迟早某一天要出来坑你一把。那肿么办咧，”跳还是不跳“，这是个问题，个人觉得，刚入门的时候，还是能跳就跳吧。等自己对这门语言产生兴趣了，再来深入了解其语言的细节，也不算太晚。

1. **多读书，都好书**

关于python的书虽不如C++，Java之类的那么多，但好书却不少了，这半年看了有十多本书了吧，整体感觉质量都挺不错。以下按照由浅入深的顺序来推荐给大家。

* + 相信我，你看的第一份文档，应该是[The Python Tutorial](http://docs.python.org/3/tutorial/index.html" \t "_blank)。什么？英语的看不懂！我去，你都还没开始看！！！
  + 看完上面的教程后，你可能会有种意犹未尽的感觉，难道，只需要这么点知识我就算入门了吗？如果你看完毫无压力，我只能说真的，这样就算入门。不过除此之外还有另外一些讲解python基础书，也值得一看。你应该把大多数时间花在上面这份tutorial上，下面(1)中基础点的书应该是当作补充。看这几本书的时候，牢记上面的两条原则！
    1. 基础点的：[A Byte of Python](http://files.swaroopch.com/python/byte_of_python.pdf), [learn python the hard way](http://learnpythonthehardway.org/book/)
    2. 稍稍进阶点的：[dive into python 3](http://www.diveintopython3.net/" \t "_blank),
    3. 需要当工具书一样看的：[The Python Standard Library by Example](http://it-ebooks.info/book/1506/" \t "_blank)
    4. 骨灰级的：[Python Cookbook, 3rd Edition](http://it-ebooks.info/book/2334/" \t "_blank)

(我是不会告诉你上面的这些书大多都有中文版的:~)

1. **好用才是王道**

看完上面这些书，你应该对python的基本语法特性，内部的标准库有了很深的了解。但是，我最想说的是，并不一定要等的你把这些书都读完了才开始做些事，（事实上，读完那份tutorial你就可以动手做很多事了）。你应该很清楚的知道自己要用python来做什么！！！想当初大一学c语言时候，学了也不知道为什么而学，所以啊，最后学完了那些语法知识后全都丢到一边，我那时候哪还知道c可以用来干那么多事。就我自己而言，学习python的目的是为了在一定程度上代替matlab作为科学计算工具，利用其丰富的包来实现许多功能，另外，用python写的代码可读性很高，不管是自己写还是读别人的代码，都是一种享受。

我想，你也一定有自己使用python目的，比如想用python爬网络上的资源，比如要用python建个网站，又或者是要和服务器上的后台打交道...你总可以找到自己要学习的那个部分，记住，把重点花在这里！。然后，等你对python有一些感性认识了，某一天自然会想起来要了解下python的底层是怎么实现的，为什么这样做比那样做更好等等问题。

编程语言说到底也只是工具罢了，工具固然是越好用越好，但更重要的是你要知道拿这些工具去解决什么样问题，以及怎样去解决！