

零基础入门学习Python

作者：小甲鱼, ePUBw.COM

目录

[前言](#)

[第1章 就这么愉快的开始吧](#)

[1.1 获得Python](#)

[1.2 从IDLE启动Python](#)

[1.3 失败的尝试](#)

[1.4 尝试点儿新的东西](#)

[1.5 为什么会这样](#)

[第2章 用Python设计第一个游戏](#)

[2.1 第一个小游戏](#)

[2.2 缩进](#)

[2.3 BIF](#)

[第3章 成为高手前必须知道的一些基础知识](#)

[3.1 变量](#)

[3.2 字符串](#)

[3.3 原始字符串](#)

[3.4 长字符串](#)

[3.5 改进我们的小游戏](#)

[3.6 条件分支](#)

[3.7 while循环](#)

[3.8 引入外援](#)

[3.9 闲聊数据类型](#)

[3.9.1 整型](#)

[3.9.2 浮点型](#)

[3.9.3 布尔类型](#)

[3.9.4 类型转换](#)

[3.9.5 获得关于类型的信息](#)

[3.10 常用操作符](#)

[3.10.1 算术操作符](#)

[3.10.2 优先级问题](#)

[3.10.3 比较操作符](#)

[3.10.4 逻辑操作符](#)

[第4章 了不起的分支和循环](#)

[4.1 分支和循环](#)

[4.2 课堂小练习](#)

[4.3 结果分析](#)

[4.4 Python可以有效避免“悬挂else”](#)

[4.5 条件表达式（三元操作符）](#)

[4.6 断言](#)

[4.7 while循环语句](#)

[4.8 for循环语句](#)

[4.9 range\(\)](#)

[4.10 break语句](#)

[4.11 continue语句](#)

[第5章 列表、元组字符串](#)

[5.1 列表：一个“打了激素”的数组](#)

[5.1.1 创建列表](#)

[5.1.2 向列表添加元素](#)

[5.1.3 从列表中获取元素](#)

[5.1.4 从列表删除元素](#)

[5.1.5 列表分片](#)

[5.1.6 列表分片的进阶玩法](#)

[5.1.7 一些常用操作符](#)

[5.1.8 列表的小伙伴们](#)

[5.1.9 关于分片“拷贝”概念的补充](#)

- [5.2 元组：戴上了枷锁的列表](#)
- [5.2.1 创建和访问一个元组](#)
- [5.2.2 更新和删除元组](#)
- [5.3 字符串](#)
- [5.3.1 各种内置方法](#)
- [5.3.2 格式化](#)
- [5.4 序列](#)
- [第6章 函数](#)
- [6.1 Python的乐高积木](#)
- [6.1.1 创建和调用函数](#)
- [6.1.2 函数的参数](#)
- [6.1.3 函数的返回值](#)
- [6.2 灵活即强大](#)
- [6.2.1 形参和实参](#)
- [6.2.2 函数文档](#)
- [6.2.3 关键字参数](#)
- [6.2.4 默认参数](#)
- [6.2.5 收集参数](#)
- [6.3 我的地盘听我的](#)
- [6.3.1 函数和过程](#)
- [6.3.2 再谈谈返回值](#)
- [6.3.3 函数变量的作用域](#)
- [6.4 内嵌函数和闭包](#)
- [6.4.1 global关键字](#)
- [6.4.2 内嵌函数](#)
- [6.4.3 闭包 \(closure\)](#)
- [6.5 lambda表达式](#)
- [6.6 递归](#)
- [6.6.1 递归是“神马”](#)
- [6.6.2 写一个求阶乘的函数](#)
- [6.6.3 这帮小兔崽子](#)
- [6.6.4 汉诺塔](#)
- [第7章 字典和集合](#)
- [7.1 字典：当索引不好用时](#)
- [7.1.1 创建和访问字典](#)
- [7.1.2 各种内置方法](#)
- [7.2 集合：在我的世界里，你就是唯一](#)
- [7.2.1 创建集合](#)
- [7.2.2 访问集合](#)
- [7.2.3 不可变集合](#)
- [第8章 永久存储](#)
- [8.1 文件：因为懂你，所以永恒](#)
- [8.1.1 打开文件](#)
- [8.1.2 文件对象的方法](#)
- [8.1.3 文件的关闭](#)
- [8.1.4 文件的读取和定位](#)
- [8.1.5 文件的写入](#)
- [8.1.6 一个任务](#)
- [8.2 文件系统：介绍一个高大上的东西](#)
- [8.3 pickle：腌制一缸美味的泡菜](#)
- [第9章 异常处理](#)
- [9.1 你不可能总是对的](#)
- [9.2 try-except语句](#)
- [9.2.1 针对不同异常设置多个except](#)
- [9.2.2 对多个异常统一处理](#)
- [9.2.3 捕获所有异常](#)
- [9.3 try-finally语句](#)
- [9.4 raise语句](#)
- [9.5 丰富的else语句](#)
- [9.6 简洁的with语句](#)

[第10章 图形用户界面入门](#)

[10.1 导入EasyGui](#)

[10.2 使用EasyGui](#)

[10.3 修改默认设置](#)

[第11章 类和对象](#)

[11.1 给大家介绍对象](#)

[11.2 对象=属性+方法](#)

[11.3 面向对象编程](#)

[11.3.1 self是什么](#)

[11.3.2 你听说过Python的魔法方法吗](#)

[11.3.3 公有和私有](#)

[11.4 继承](#)

[11.4.1 调用未绑定的父类方法](#)

[11.4.2 使用super函数](#)

[11.5 多重继承](#)

[11.6 组合](#)

[11.7 类、类对象和实例对象](#)

[11.8 到底什么是绑定](#)

[11.9 一些相关的BIF](#)

[第12章 魔法方法](#)

[12.1 构造和析构](#)

[12.1.1 __init__\(self, ...\)](#)

[12.1.2 __new__\(cls, ...\)](#)

[12.1.3 __del__\(self\)](#)

[12.2 算术运算](#)

[12.2.1 算术操作符](#)

[12.2.2 反运算](#)

[12.2.3 增量赋值运算](#)

[12.2.4 一元操作符](#)

[12.3 简单定制](#)

[12.4 属性访问](#)

[12.5 描述符（property的原理）](#)

[12.6 定制序列](#)

[12.7 迭代器](#)

[12.8 生成器（乱入）](#)

[第13章 模块](#)

[13.1 模块就是程序](#)

[13.2 命名空间](#)

[13.3 导入模块](#)

[13.4 `__name__ = '__main__'`](#)

[13.5 搜索路径](#)

[13.6 包](#)

[13.7 像个极客一样去思考](#)

[第14章 论一只爬虫的自我修养](#)

[14.1 入门](#)

[什么是编码](#)

[14.2 实战](#)

[14.2.1 下载一只猫](#)

[14.2.2 翻译文本](#)

[14.3 隐藏](#)

[14.3.1 修改User-Agent](#)

[14.3.2 延迟提交数据](#)

[14.3.3 使用代理](#)

[14.4 Beautiful Soup](#)

[14.5 正则表达式](#)

[14.5.1 re模块](#)

[14.5.2 通配符](#)

[14.5.3 反斜杠](#)

[14.5.4 字符类](#)

[14.5.5 重复匹配](#)

- [14.5.6 特殊符号及用法](#)
- [14.5.7 元字符](#)
- [14.5.8 贪婪和非贪婪](#)
- [14.5.9 反斜杠+普通字母=特殊含义](#)
- [14.5.10 编译正则表达式](#)
- [14.5.11 编译标志](#)
- [14.5.12 实用的方法](#)
- [14.6 异常处理](#)
 - [14.6.1 URLError](#)
 - [14.6.2 HTTPError](#)
 - [14.6.3 处理异常](#)
- [14.7 安装Scrapy](#)
- [14.8 Scrapy爬虫之初窥门径](#)
 - [14.8.1 Scrapy框架](#)
 - [14.8.2 创建一个Scrapy项目](#)
 - [14.8.3 定义Item容器](#)
 - [14.8.4 编写爬虫](#)
 - [14.8.5 爬](#)
 - [14.8.6 取](#)
 - [14.8.7 在Shell中尝试Selector选择器](#)
 - [14.8.8 使用XPath](#)
 - [14.8.9 提取数据](#)
 - [14.8.10 使用item](#)
 - [14.8.11 存储内容](#)
- [第15章 GUI的最终选择: Tkinter](#)
 - [15.1 Tkinter之初体验](#)
 - [15.2 Label组件](#)
 - [15.3 Button组件](#)
 - [15.4 Checkbutton组件](#)
 - [15.5 Radiobutton组件](#)
 - [15.6 LabelFrame组件](#)
 - [15.7 Entry组件](#)
 - [15.8 Listbox组件](#)
 - [15.9 Scrollbar组件](#)
 - [15.10 Scale组件](#)
 - [15.11 Text组件](#)
 - [15.11.1 Indexes用法](#)
 - [15.11.2 Marks用法](#)
 - [15.11.3 Tags用法](#)
 - [15.12 Canvas组件](#)
 - [15.13 Menu组件](#)
 - [15.14 Menubutton组件](#)
 - [15.15 OptionMenu组件](#)
 - [15.16 Message组件](#)
 - [15.17 Spinbox组件](#)
 - [15.18 PanedWindow组件](#)
 - [15.19 Toplevel组件](#)
 - [15.20 事件绑定](#)
 - [15.21 事件序列](#)
 - [15.21.1 type](#)
 - [15.21.2 modifier](#)
 - [15.22 Event对象](#)
 - [15.23 布局管理器](#)
 - [15.23.1 pack](#)
 - [15.23.2 grid](#)
 - [15.23.3 place](#)
 - [15.24 标准对话框](#)
 - [15.24.1 messagebox \(消息对话框\)](#)
 - [15.24.2 filedialog \(文件对话框\)](#)
 - [15.24.3 colorchooser \(颜色选择对话框\)](#)

[第16章 Pygame: 游戏开始](#)

[16.1 安装Pygame](#)

[16.2 初步尝试](#)

[16.3 解惑](#)

[16.3.1 什么是Surface对象](#)

[16.3.2 将一个图像绘制到另一个图像上是怎么一回事](#)

[16.3.3 移动图像是怎么一回事](#)

[16.3.4 如何控制游戏的速度](#)

[16.3.5 Pygame的效率高不高](#)

[16.3.6 我应该从哪里获得帮助](#)

[16.4 事件](#)

[16.5 提高游戏的颜值](#)

[16.5.1 显示模式](#)

[16.5.2 全屏才是王道](#)

[16.5.3 使窗口尺寸可变](#)

[16.5.4 图像的变换](#)

[16.5.5 裁剪图像](#)

[16.5.6 转换图片](#)

[16.5.7 透明度分析](#)

[16.6 绘制基本图形](#)

[16.6.1 绘制矩形](#)

[16.6.2 绘制多边形](#)

[16.6.3 绘制圆形](#)

[16.6.4 绘制椭圆形](#)

[16.6.5 绘制弧线](#)

[16.6.6 绘制线段](#)

[16.7 动画精灵](#)

[16.7.1 创建精灵](#)

[16.7.2 移动精灵](#)

[16.8 碰撞检测](#)

[16.8.1 尝试自己写碰撞检测函数](#)

[16.8.2 sprite模块提供的碰撞检测函数](#)

[16.8.3 实现完美碰撞检测](#)

[16.9 播放声音和音效](#)

[16.10 响应鼠标](#)

[16.10.1 设置鼠标的位置](#)

[16.10.2 自定义鼠标光标](#)

[16.10.3 让小球响应光标的移动频率](#)

[16.11 响应键盘](#)

[16.12 结束游戏](#)

[16.12.1 发生碰撞后获得随机速度](#)

[16.12.2 减少“抖动”现象的发生](#)

[16.12.3 游戏胜利](#)

[16.12.4 更好地结束游戏](#)

[16.13 经典飞机大战](#)

[16.13.1 游戏设定](#)

[16.13.2 主模块](#)

[16.13.3 我方飞机](#)

[16.13.4 响应键盘](#)

[16.13.5 飞行效果](#)

[16.13.6 敌方飞机](#)

[16.13.7 提升敌机速度](#)

[16.13.8 碰撞检测](#)

[16.13.9 完美碰撞检测](#)

[16.13.10 一个BUG](#)

[16.13.11 发射子弹](#)

[16.13.12 设置敌机“血槽”](#)

[16.13.13 中弹效果](#)

[16.13.14 绘制得分](#)

[16.13.15 暂停游戏](#)

- [16.13.16 控制难度](#)
- [16.13.17 全屏炸弹](#)
- [16.13.18 发放补给包](#)
- [16.13.19 超级子弹](#)
- [16.13.20 三次机会](#)
- [16.13.21 结束画面](#)
- [参考文献](#)

前言

Life is short. You need Python.

——Bruce Eckel

上边这句话是Python社区的名言，翻译过来就是“人生苦短，我用Python”。

我和Python结缘于一次服务器的调试，从此便一发不可收拾。我从来没有遇到一门编程语言可以如此干净、简洁，如果你有处女座情节，你一定会爱上这门语言。使用Python，可以说是很难写出丑陋的代码。我从来没想过一门编程语言可以如此简单，它太适合零基础的朋友踏入编程的大门了，如果我有一个八岁的孩子，我一定会毫不犹豫地使用Python引导他学习编程，因为面对它，永远不缺乏乐趣。

Python虽然简单，其设计却十分严谨。尽管Python可能没有C或C++这类编译型语言运行速度那么快，但是C和C++需要你无时无刻地关注数据类型、内存溢出、边界检查等问题。而Python，它就像一个贴心的仆人，私底下为你都一一处理好，从来不用你操心这些，这让你可以将全部心思放在程序的设计逻辑之上。

有人说，完成相同的一个任务，使用汇编语言需要1000行代码，使用C语言需要500行，使用Java只需要100行，而使用Python，可能只要20行就可以了。这就是Python，使用它来编程，你可以节约大量编写代码的时间。

既然Python如此简单，会不会学了之后没什么实际作用呢？事实上你并不担心这个问题，因为Python可以说是一门“万金油”语言，在Web应用开发、系统网络运维、科学与数字计算、3D游戏开发、图形界面开发、网络编程中都有它的身影。目前越来越多的IT企业，在招聘栏中都有“精通Python语言优先考虑”的字样。另外，就连Google都在大规模使用Python。

好了，我知道过多的溢美之词反而会使大家反感，所以我必须就此打住，剩下的就留给大家自己体验吧。

接下来简单地介绍一下这本书。一年前，出版社的编辑老师无意间看到了我的一个同名的教学视频，建议我以类似的风格撰写一本书。当时我是受宠若惊的，也很兴奋。刚开始写作就遇到了不小的困难——如何将视频中口语化的描述转变为文字。当然，我希望尽可能地保留原有的幽默和风趣——毕竟学习是要快乐的。这确实需要花不少时间去修改，但我觉得这是值得的。

本书不假设你拥有任何一方面的编程基础，所以本书不但适合有一定编程基础，想学习Python3的读者，也适合此前对编程一无所知，但渴望用编程改变世界的朋友！本书提倡理解为主，应用为王。因此，只要有可能，我都会通过生动的实例来让大家理解概念。虽然这是一本入门书籍，但本书的“野心”可并不止于“初级水平”的教学。本书前半部分是基础的语法特性讲解，后半部分围绕着Python3在爬虫、Tkinter和游戏开发等实例上的应用。编程知识深似海，没办法仅通过一本书将所有的知识都灌输给你，但我能够做到的是培养你对编程的兴趣，提高你编写代码的水平，以及锻炼你的自学能力。最后，本书贯彻的核心理念是：实用、好玩，还有参与。

本书对应的系列视频教程，可以在<http://blog.fishc.com/category/python>下载得到，也可扫描以下二维码关注微信号进行观看。

□

微信扫描书中对应二维码，亦可观看相关视频。

编者

2016年7月

第1章 就这么愉快的开始吧

1.1 获得Python

我观察到这么一个现象：很多初学的朋友都会在学习论坛上问什么语言才是最好的？他们的目的很明确，就是要找一门“最好”的编程语言，然后持之以恒地学习下去。没错，这种“执子之手，与子偕老”的专一精神是我们现实社会所推崇的。但在编程的世界里，我们并不提倡这样。我们更提倡“存在即合理”，当前热门的编程语言都有其存在的道理，它们都有各自擅长的领域和适用性。因此我们没办法去衡量哪一门语言才是最好的。

Python的语法是非常精简的，对于一位完美主义者来说，Python将是他爱不释手的伙伴。Python社区的目标就是构造完美的Python语言！本书将使用Python3来进行讲解，而Python3不完全兼容Python2的语法，这样做无疑会让大多数程序员心生怨愤且喋喋不休，因为他们用Python2写的大量代码经过层层调试已经趋近完美，并已部署到服务器或应用上了。Python3对Python2的语法不兼容，意味着他们的这些应用需要进行转换和重新调试……但是，Python社区仍旧坚持推出全新的Python3。只有勇敢地割掉与时代发展不相符的瑕疵部分，才能缔造出真正的完美体验！

工欲善其事，必先利其器。我们要成为“大牛”，要用Python去拯救世界，要做的第一件事就是要下载一个Python的安装程序并成功地将它安装到你的计算机上。

安装Python非常容易，你可以在它的官网找到最新的版本并下载（注：本书所需要的程序、例子均附带在本书配套资源中），地址是<http://www.python.org>。

如图1-1所示，进入Python官网后找到Download字样，下载最新版本的Python即可。

如果是其他操作系统（例如，Mac OS X），在页面下方可以找到对应的下载地址，如图1-2所示。

此处演示的是本书截稿前的最新版本Python 3.4.3（32位）（注：这里建议大家安装32位版，因为本书第16章安装Pygame时需要32位版本的Python），一般大家下载最新版本即可。安装Python3非常简单，打开下载好的安装包，按照默认选项安装即可。

□

图1-1 下载Python3

□

图1-2 下载Python3

1.2 从IDLE启动Python

IDLE是一个Python Shell，shell的意思就是“外壳”，从基本上说，就是一个通过输入文本与程序交互的途径。像Windows的cmd窗口，像Linux那个黑乎乎的命令窗口，它们都是shell，利用它们，就可以给操作系统下达命令。同样，可以利用IDLE这个shell与Python进行互动。

>>>这个提示符含义是：Python已经准备好了，在等着输入Python指令呢。如图1-3所示，可以看到Python已经按照我们的要求去做了，在屏幕上打印（注：这里打印的意思是“打印”到屏幕上）I love fishc.com这个充满浓浓爱意的字符串，这说明什么？没错，这说明我们是“爱鱼C”的，也说明了 we 跟Python的第一次亲密接触是有感觉的，她完全能够理解我的想法。

□

图1-3 在Python的IDLE中输入命令

1.3 失败的尝试

像下面这样输入，Python就会“笨笨地”出错：

□

其实Python3哪里是“笨”，她只是小气，所以显得蠢萌蠢萌的。我们仿佛听到她在说：为什么此时此刻你跟我在一起还想着前任？为什么你跟我在一起还想着其他女人，小C她哪儿比我好，她还要加分号呢，我可不用！

大家看到上边的代码中井号（#）后边加了段中文，井号起到的作用是注释，也就是说，井号后边的内容是给人们看的，并不会被当作代码运行。

1.4 尝试点儿新的东西

尝试点儿新的东西，在IDLE中输入`print(5+3)`或者直接输入`5+3`：

□

看起来Python还会做加法！这并不奇怪，因为计算机最开始的时候就是用来计算的，任何编程语言都具备计算能力，那接下来看看Python在计算方面有何神奇。

不妨再试试计算`1234567890987654321 * 9876543210123456789`：

□

怎么样？如果C语言实现起来费劲，要九曲十八弯地利用数组做大数运算，在这里Python轻而易举就完成了！

还有呢，大家试试输入`print(" Well water " + " River ")`：

□

可以看到，井水和河水又友好地在一起生活了，祝它们幸福吧！

1.5 为什么会这样

再试试`print(" I love python\n " * 3)`：

□

哇，字符串和数字还可以做乘法，结果是重复显示N个字符串。既然乘法可以，那不妨试试加法。

`print(" I love python\n " +3)`：

□

失败了！这是为什么呢？大家不妨课后自己思考一下。

第2章 用Python设计第一个游戏

2.1 第一个小游戏

有读者可能会说：“哇，小甲鱼（注：作者）！你开玩笑呢？这么快就教我们开发游戏啦？难道你不打算先讲讲变量、分支、循环、条件、函数等常规的内容？”

没错的，大家如果继续学下去就会发现，本书的教学会围绕着一个个性鲜明的实例来展开，跟着本书完成这些实例的编写，你会发觉不知不觉中那些该掌握的知识，已经化作你身体的一部分了！这样的学习方式才能充满快乐并让你一直期待下一章节的到来。

好，今天来讲一下“植物大战僵尸”这款游戏的编写……但这是不可能的，因为虽然说Python容易入门，但像“植物大战僵尸”这类游戏要涉及碰撞检测、边缘检查、画面刷新和音效等知识点比较多，需要将这些基础知识累积完成才能开始讲。

目前对于我们所掌握的基础.....貌似只有print()这个BIF，哦，BIF的概念甚至还没讲解.....不过请淡定，这一点儿也不影响我们今天的节奏！

那么今天是一个什么样的节奏呢？今天打算讲一个文字游戏.....

先来看下这段代码，并试图猜测一下每条语句的作用：

□

在这里要求大家都动动手，亲自输入这些代码，你需要做的是：

打开IDLE。

选择File->New Window命令（或者你可以直接按Ctrl+N键，在很多地方这个快捷键都是新建一个文件的意思）。

按照上边的格式填入代码。

按快捷键Ctrl+S，将源代码保存为名为p2_1.py的文件。

输完代码一起来体验一下，F5走起（也可以选择Run->Run Module命令）！

程序执行结果如下：

□

提示

Tab按键的使用：

（1）缩进。

（2）IDLE会提供一些建议，例如输入pr TAB会显示所有可能的命令供你参考。

OK，我们是看到程序成功跑起来了，但坦白说，这玩意儿配叫游戏吗？呃.....没事啦，咱慢慢改进，好，我们说下语法。

有C-like语言（一切语法类似C语言的编程语言称为C-like语言）编程经验的朋友可能会受不了，变量呢？声明呢？怎么直接就给变量定义了呢！有些真正零基础的读者可能还不知道什么是变量，不怕，随着本书内容的展开，大家很快就能掌握相关的知识。有些读者可能发现这个小程序没有任何大括号，好多编程语言都用大括号来表示循环、条件等的作用域，而在Python这里是没有的。在Python中，只需要用适当缩进来表示即可。

2.2 缩进

缩进是Python的灵魂，缩进的严格要求使得Python的代码显得非常精简并且有层次。但是，在Python里对待代码的缩进要十分小心，因为如果没有正确地使用缩进，代码所做的事情可能和你的期望相差甚远（就像在C语言里括号打错了位置）。

如果在正确的位置输入冒号（:），IDLE会在下一行自动进行缩进。正如方才的代码，在if和else语句后边加上冒号（:），然后按下回车，第二行开始的代码会自动进行缩进。if条件下边有两个语句分别有缩进，那么说明这两个语句是属于if条件成立后所需要执行的语句，换句话说，如果if条件不成立，那么两个缩进的语句就不会被执行。

提示

if...else...是一个条件分支，if后边跟的是条件，如果条件成立，就执行以下缩进的所有内容；如果条件

欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：《零基础入门学习Python》小甲鱼 著.epub

请登录 <https://shgis.cn/post/330.html> 下载完整文档。

手机端请扫码查看：

