零基础学 Python

V1.0

申龙斌

2017 年 4 月 24 日

金喜擅

读书、写作、锻炼、编程、投资，
一切皆定投……

(如有疑问，请关注我的微信公众号【金喜擅】，给我留言)

目录

1 “零基础学编程”都需要哪些基础？

工作了 20 多年，学了 Basic、C、C++、JAVA、C#、Objective-C、R、Go 等一堆语言，可惜样样都不精。最近好像流行零基础学编程，我努力清空了我的编程知识，仔细搜寻了学习第一门编程语言时的记忆。

大概是在 1987 年，我当时在一个很不起眼的中学读高中，可能当时油田效益还不错，竟然引进了 20 台昂贵的 Apple II 计算机。我当时并不知道这个 Apple 是乔布斯发明的，也没想到 30 年后人人都在用苹果手机。当时不仅仅是零基础学编程，还是零基础学电脑。当时的第一课不是如何学习何用电脑，而是直接学 BASIC 语言，就是比尔盖茨在车库里弄出来的那个 BASIC 编程语言。

BASIC 书发下来后，看了几页，简直就是天书，记得最清楚的是一张用字符组成的熊猫图案要用写好几页的代码！我的记忆中还有点模糊印象的第一条源代码大概是这样的：

```
    10 LET A = 1
```

然后，书中用了好几段的篇幅来讲 BASIC 的语法，行号、关键字、语句、赋值、变量……一大堆概念，直接晕掉。当时好像 1、2 周上机一次，好像就是上机一行一行地试，才慢慢地明白了基础的编程思路。再以后上大学、参加工作，学了一堆 IT 知识，走上了一条程序人生。

学编程有啥用？

不管你身处哪个行业，会点编程，都会使你如虎添翼。不仅扩展了你的思维方式，而且会写点小程序可极大地提高工作的效率。

实际上很多办公人员都在使用电子表格 Excel 程序，大多数人使用的都是其中的 1%的功能，拿它画些表格线，填上一堆文字和数字，只是做一些简单的美化修饰工作。

稍微熟练一些的人士，可以使用一些 Excel 中的公式进行一些求和、取平均等运算。实际上这些操作就是一种简单的编程，而且是如今非常火的**函数式编程**呢！至于什么是函数式编程，则暂时超出了初学者的范围，今天就不介绍了。

但你真的是零基础吗？并不是，想学会编程，你至少得有这些基础：

（1）会一点点基本的英文

几乎所有的编程语言都是用英文来编写的，不会英文的朋友也不要被吓到，在编程语言常用到的英文单词也就是几十个，而且都很超级简单。比如，Python 这种编程语言，常用到的英语单词大概 30 个，下面列出一些。

```
False      class      finally    is
return     None       continue   for
```

```
try         True        from        while
global      not         with        as
if          or          yield       assert
else        import      break       except
```

当然，最新的编程资料都是英文写的，优质的源程序大多数也都在国外网站上。英文阅读过关的话，对你的帮助会更大。

（2）学会搜索

最好用谷歌搜索，大部分你在编程时掉进去过的坑，别人都踩过，所以遇到问题时，第一时间先自己琢磨几分钟，还解决不了时马上搜索。大家都用某度搜索，可它的搜索质量真不敢恭维。

可怜谷歌退出中国，想用它还得学会 vpn 科学上网，国人想学编程还真不容易啊！什么是 vpn？这里不讨论了，可以加我微信 SLOFSLB 私聊。

（3）找个导师

学习编程与学习英语的困难有一点是非常相似的，就是当你遇到一个问题时，就会卡在那里，寝食难安，花上 1 天可能也毫无进展。而如果你问一下有经验的朋友，他很可能在 1 分钟内解决你的问题。

所以这种学习方法称为“**互助式学习法**”，也就是找到一些有经验的人，向他们求助。如果你工作的地方有程序员同事，那你就具有了得天独厚的条件。但这会带来一个问题，你将占用他人的大量时间，所以你得想好用什么来补偿你的朋友。

如果身边没有程序员朋友，还可以在互联网论坛上求助，程序员们通常都很热心，如果你把问题描述得足够清楚，他们是不会吝惜几分钟来解答的。实在不行，也可以在我的文章下面留言，说不定我和大家就能帮你解决问题呢。

（4）早点学会盲打

盲打越早学会越好，即使你不写程序代码，总还是要用电脑打字的。不会盲打的人常用“二指禅”打字，比专业打字员慢上几十倍。而只要刻意练习，一般 1 个月就能学会，只不过一开始速度慢点罢了，以后只需要不停地打下去就行了。1 个月的学习，一辈子受益，早练早受益。如果你是老板，在公司里发现了不会盲打的程序员，直接开掉他吧。

为啥男生更喜欢编程呢？可能是学会编程，有种掌控世界的感觉，你敲上几行指令，计算机就会按照你的意愿来行事。有《超新约全书》这样一部电影，说上帝是个邋遢的程序员……

开始编程之旅前，先看部电影消遣一下吧。

2 用在线编程环境快速上手

编程之路从来都不轻松，一路上你要学习各种知识点，会遇到无数的阻碍，所以你要找到编程的内心驱动力，让学会编程成为你的刚需，才能让你在编程道路上不断前行。

编程虽难，但仍有办法。想起我当时想学编程的动机竟然是缘于游戏，记得有一天的周末，我在 Apple II 微机教室里看到一位同学在玩游戏，他正在与电脑下中国象棋，只见他走了一步之后，计算机经过几十秒的“思考”之后，缓慢地挪动了一枚棋子，当时（80 年代末）也没注意计算机的象棋水平有多高，但感觉太神奇了！计算机竟然会思考、会下棋，这让我产生了强烈的好奇心去探索其中的奥秘。

所以说，如果能够通过完成一步又一步的编程小任务，最终写出一个小游戏，那才是真正的寓教于乐。对于初学者来说，它能快速调动起兴趣，调动起你探索未知世界的欲望，把编码当成一种游戏，这才是最高效的学习方式，学编程应该是一种愉悦的体验。

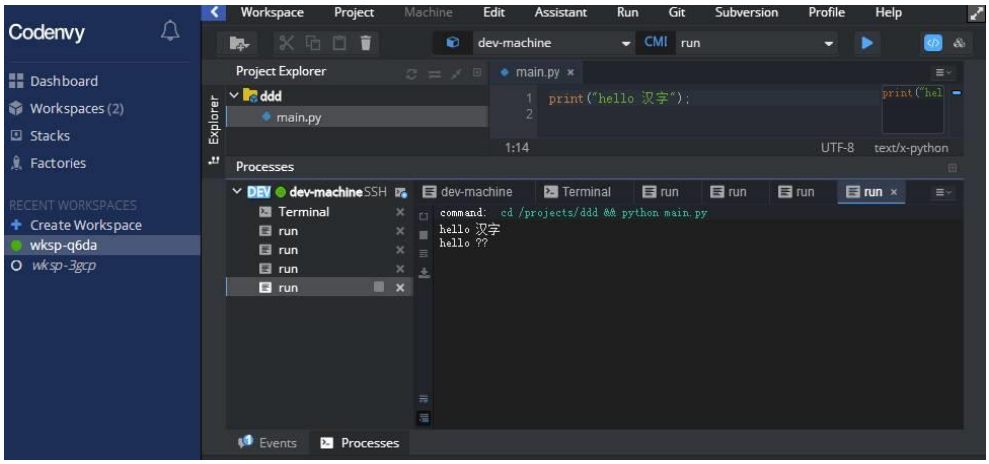
像学英语初期从不讲语法规则一样，我认为学编程也不应该一上来就讲语言的语法，而是应该让初学者快速上手、快速体验、快速试错。像打游戏闯关一样自行学习。我搜索了“通过写游戏学编程”的内容，在 python 语言方面没找到，只能退而求其次，找到了一些在线学编程的网站。

现在已经进入了移动互联网和云的时代，可以随时随地登录这些平台马上动手学习了。第一篇文章里提到了搜索是一项基本技能，这次我先搜索的就是“在线学编程语言”，大概发现了这样几个平台，每个我都简单试用了一番，有些会被**墙**（也就是 blocked 的意思，就是国内把该网站屏蔽了的意思），还有不少是可以用的。

具体网址我就不写了，用什么搜索工具都能找到。

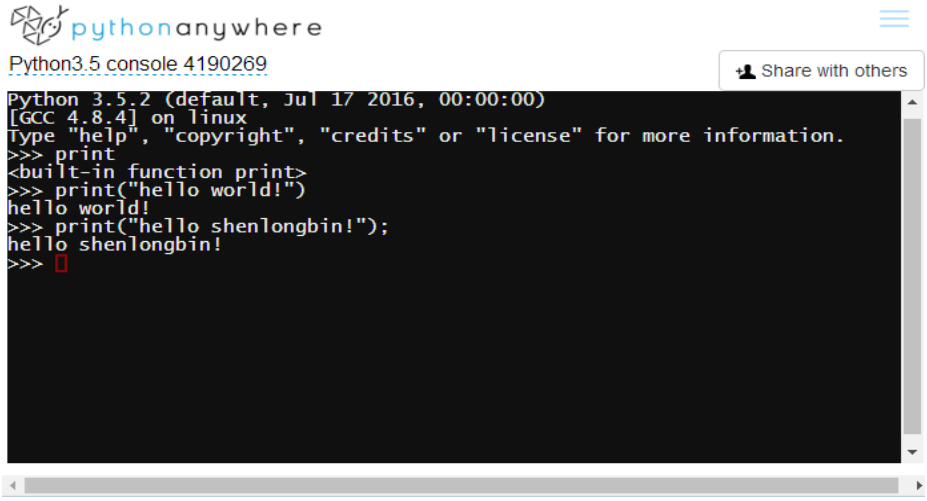
（1）codenvy

要注册用户、建立工作区、建项目，功能太复杂，不适合初学者。



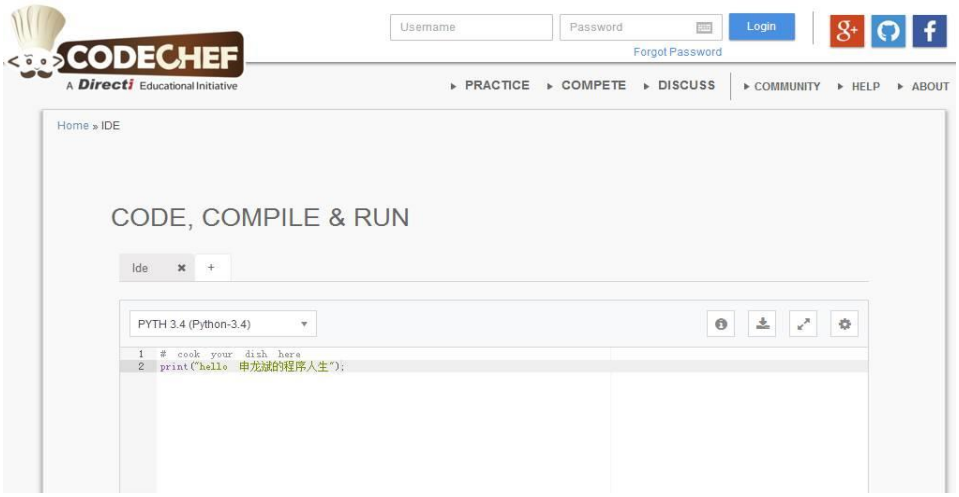
（2）pythonanywhere

登录挺方便，可惜只有一个黑窗口，不能输入汉字，不适合初学者。



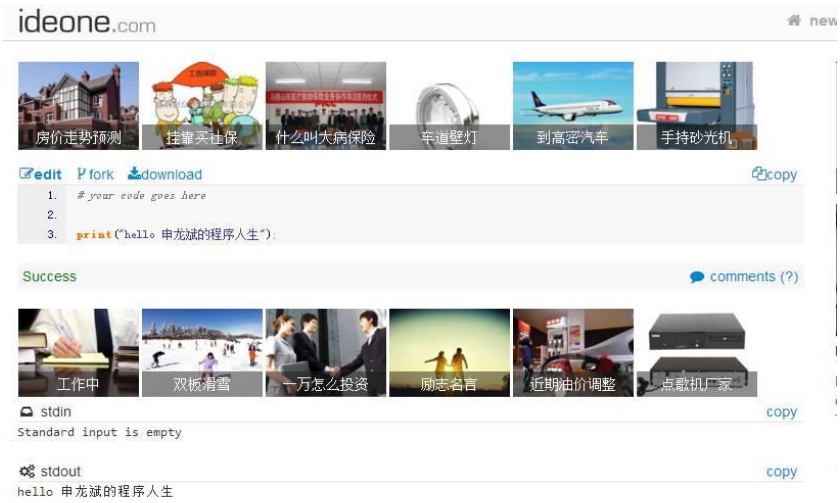
（3）codechef

使用倒是很方便，可惜中文输出是乱码！



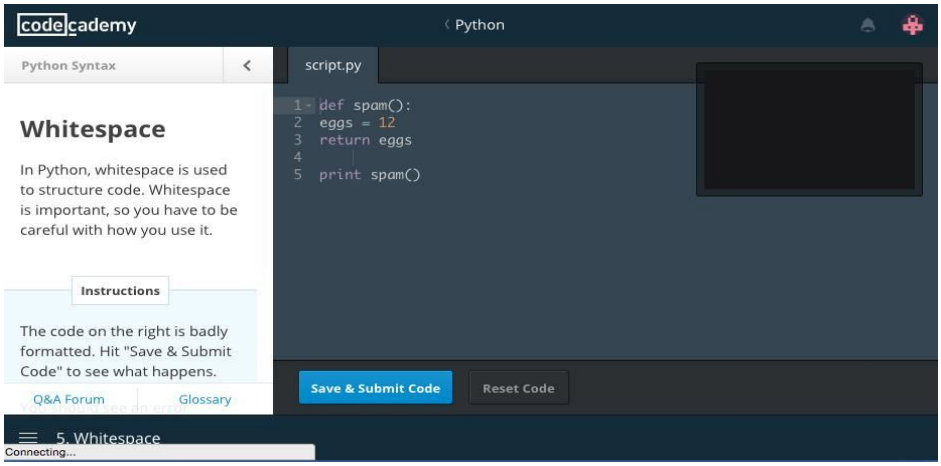
（4）ideone

这个使用非常方便，也支持中文，可惜满屏都是广告！



（5）codecademy

这是我试了几个之后感觉最好的。虽然它布置的任务不是写出一个小游戏，仍是教你语法，但它使用了游戏奖励的机制，你完成一定的任务后会得到奖章。有一些遗憾，整个教程是用英文写的，想起我上一篇文章讲的吧，良好的英文功底，会让学编程更容易。期待将来会出类似的中文教学平台。



该网站支持很多种编程语言，具体的过程先不写了，大家先去试试这个网站，先练上几道非常非常简单的题试试。如有问题，用微信联系我 SLOFSLB。

另外，我还搜索了大名鼎鼎的 stackoverflow 网站，用的关键词是“best way learn programming”，找到了一篇经典的贴子。点赞数最多的那条回答就是游戏化的编程思路，从易到难，给出了多个步骤，还是那句话，需要基本的英文功底。

3 Hello World

前面介绍了 codecademy 在线学编程的网站，不知道大家动手试验了没有？是不是太简单，一下子就完成了许多练习？

第一课的内容只有一条输出语句，点击保存并提交后，在屏幕的右上角输出一行文字“Welcome to Python!”。

实际上，编程世界中的第一个示例程序并不是它，而是 Hello World。

```
print "Welcome to Python!"
```

Python 等编程语言的 Hello World 相当的简单，只需要一行语句，但其它发明的比较早的语言就显得有点麻烦了，比如 C 语言：

```
#include <stdio.h>
int main()
{
    printf("hello, world");
    return 0;
}
```

再比如 JAVA 语言：

```
public class HelloWorld{
    public static void main(string args[]){
        System.out.println( "Hello World! \n" );
```

```
    }
}
```

是不是感觉 Python 更简洁、更人性化一些？

别只会个 hello world，来个数学计算，在代码编辑窗口中输入：

```
print 2**10000
```

保存并提交，看看右上角的小黑窗口出现了什么？见识了 Python 的强大吧？这里的 2**10000 表示 2 的 1 万次方，你能数出来它有多少位吗？

知识点：

- print 表示在屏幕上输出指定的内容
- ** 表示数学中的幂运算，即多少次方
- 注意 print 后面的空格不能省

4 在游戏中学 JAVA 和 C#

前面几篇文章中主要用 Python 当例子，但在这些“在线编程环境”中，还支持其它编程语言，JAVA 和 C#也不例外。CodeCademy 中提供了许多很短小的习题使人能够快速了解关键语法，但还是有点枯燥，如果能够在游戏中学习编程就更好了。实际上，这种事情程序员们也早就想到了，robocode 就是其中之一。

看看百度百科上的介绍：

> Robocode 是一种有趣的竞赛性编程，使用几行简单的代码，就能够让你创建一个活生生的机器人，一个真正的在屏幕上与其他机器人互相对抗的机器人。你可以看到它在屏幕上四处疾驰，碾碎一切挡道的东西。机器人配有雷达与火炮，选手在躲避对手进攻的同时攻击对手，以此来较量得分的多少。Robocode 可以让你在娱乐的同时学习与提高 Java 技术。

准确地说，这个 robocode 平台中创建的并不是机器人，而是机器坦克。这种游戏中，你不能用键盘和鼠标去控制你的坦克，而是用事先写好的代码，让自己的坦克躲避、并攻击敌人。最早这个项目只支持 JAVA 语言，后来才支持了 C#。你需要不断优化你的代码，从而学习基本的编程知识。但玩（或编码）到一定程度后，你需要学习的内容将变为 AI（人工智能）领域了。

有关 robocode 资料，可以访问：http://robowiki.net，全英文说明。该软件最早发布在 ibm 的 web alphaworks 上，现在已经搬到了 sourceforge.net（github 上也有）。很遗憾，如果没有 vpn，sourceforge.net 网站的访问也不太稳定，光下载 robocode 都折腾死人。

国内网站上也可以找到一些 robocode 的教程，但都有点过时了。我今天下载了 1.9.2.5 版本，如果是零基础的话，安装还真不容易。

你需要闯过安装 vpn、安装 java 环境（要 1.6 版本以上）、安装 robocode 程序、启动 robocode 程序等几大关，放上一堆坦克，然后才能见到下面的画面。

如果没人帮助，零基础的朋友真无法完成上面的安装任务，所以，如果你真是零基础，还是先到 codecademy 上把基础教程认真学完再说吧。

如果你已经具备了一定 JAVA 基础，并且成功地安装好了 robocode，就可以开始写自己的机器坦克的代码了，但我看了一下第一个源程序，真不适合零基础的同学。

```
package man;
import robocode.*;

public class MyFirstRobot extends Robot {
    public void run() {
        while (true) {
            ahead(100);
            turnGunRight(360);
            back(100);
            turnGunRight(360);
        }
    }

    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }
}
```

此时你马上遇到的是集成开发环境 IDE 的选择、编译器的设置等一系列障碍，想看到自己写的坦克在屏幕上乱窜并不容易，所以说学编程的最佳办法是**互助式学习**。

5 集成开发环境 IDE

几天前介绍了《用在线编程环境快速上手》学习 Python 等编程语言，这种教学环境中的例子都非常简单，你不需要在自己的电脑中安装任何的软件，就可以马上动手学习 Python 的语法了。不知道大家试了没有？太简单还是太难？

笨办法学 Python

我从用户的反馈中听说有人在用《笨办法学 Python》这本书学编程，我也下载了第 4 版的中文教程翻了几页。这本书写得确实非常简单、啰嗦，挺适合零基础的朋友，但也发现了几点不足：作者推荐用 gedit 文本编辑器来编写代码，再用命令行工具来看运行结果，对初学者又提出了较高的要求。另外，全书的例子全是 Python 2，而不是最新的 3 版本。

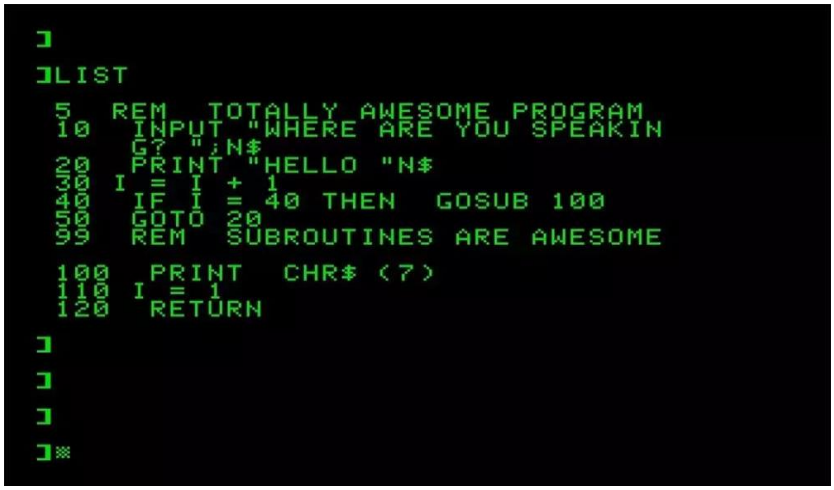
一直**在线**连网学编程总不是个办法，我们早晚需要在自己的机器上安装一套软件开发程序，这样随时随地都可以做练习了。初学者最怕弹出一个黑窗口，从中输入奇怪的命令，再根据奇怪的提示信息去找错误，再回到编辑器中修改代码，不断重复这样的过程。

适合的 IDE 就能让初学者减轻学习的难度。

集成开发环境 IDE

IDE 的全称是「集成开发环境」，英文就是 Integrated Development Environment，是与「非集成开发环境」相对应。

这让我回忆起了最早接触 Apple II 电脑的时候见到的黑屏幕上的一排排绿字符，Windows 进化到现在，仍然还留着这类似的、古老的 cmd 黑窗口。



这种界面，现在通常称为**控制台** Console，你输入一串字符后，按下回车 ENTER 键，计算机给出相应的提示，这种东西现在可不能当作 IDE。

而 IDE 与其他工具的关键区别在于**“集成”**，你编写程序，需要编辑代码、运行程序、查看错误信息、定位错误的位置、查看变量的值、再编辑代码、再运行，整个过程将伴随程序员的一生。

如果没有 IDE，你需要用 Notepad 编辑代码，在 cmd 黑窗口中输入 python 命令运行程序，再根据提示到 Notepad 中编辑代码，再保存再运行。中间只要出现一点点小错误，比如少写空格、文件名写错、忘记保存、当前文件夹不正确……你都需要花费不少的时间去修改。而有了 IDE，则会马上知道错在哪一行，并自动跳转到该位置，修改后马上可以看到运行结果。

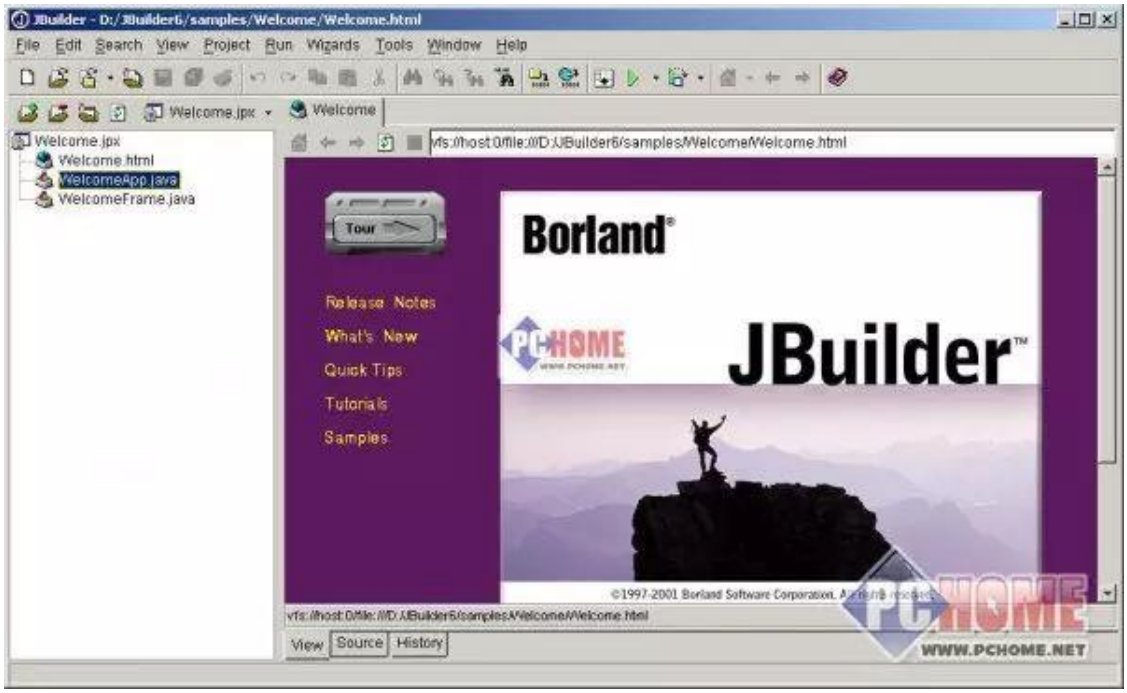
IDE 可以看做一个**“万能工具箱”**，里面装满了扳手、螺丝刀、榔头、电钻、卷尺等各式各样的工具，你在修理一个家电时，想用什么拿起来就用，效率很高。设想你手里只有一把扳手，其它工具需要到各个邻居家里去借，你是发现了问题后再去借相应的工具，还是借来一套完整的工具箱？所以说，强大的 IDE，会让编辑、编译、运行、定位、查错、修改等一气呵成，不需要在多个工具之间来回切换，并且还提供给你许多有用的工具来避免一些错误，极大地提高了效率。

各式各样的 IDE

C 程序员们都使用过经典的 Turbo C，看看它上面的菜单栏就知道它主要集成了哪些功能，文件管理、编辑、运行、编译、工程管理、设置选项、调试、断点查看等，虽然是文本式的 IDE，在当时那个年代，已经非常强大了。当年我就是用 TurboC 自己去写俄罗斯方块，可惜现在一行代码也没留下来。

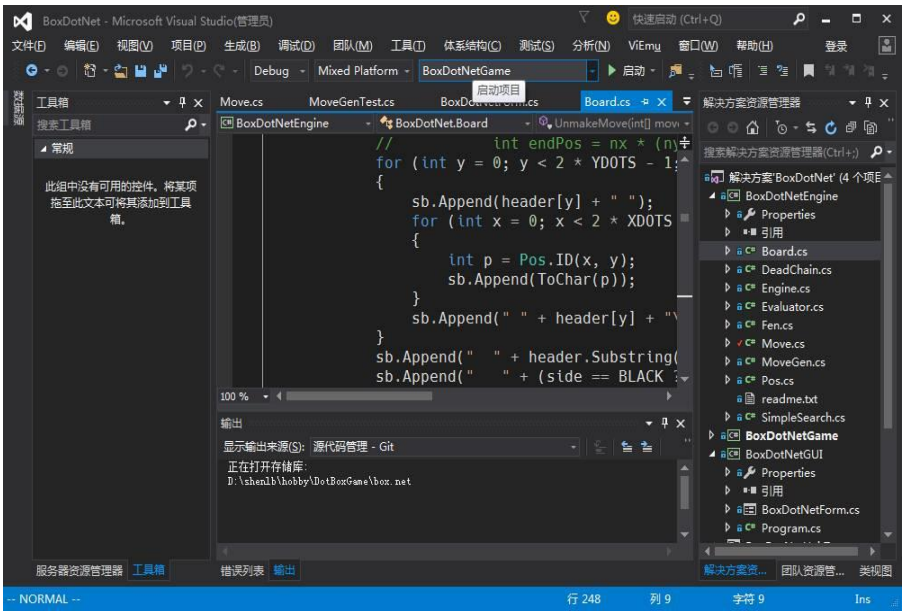
Turbo C IDE

JAVA 的老牌集成环境 Jbuilder，可能很多人都没有听说过了，当年的 Borland 公司也曾经是叱咤风云，开发过无数款功能强大的集成开发环境。



JBuilder IDE

微软的 Visual Studio 也是在不断地更新，功能也是相当地强大。



Visual Studio 2015

Eclipse 通用集成开发环境出世后，几乎通吃了所有语言的 IDE，不管你用什么编程语言，总能找到相关的插件，你只需要把精力放在编程上就行了，剩下的事，Eclipse 基本上都能做。

Eclipse 集成开发环境

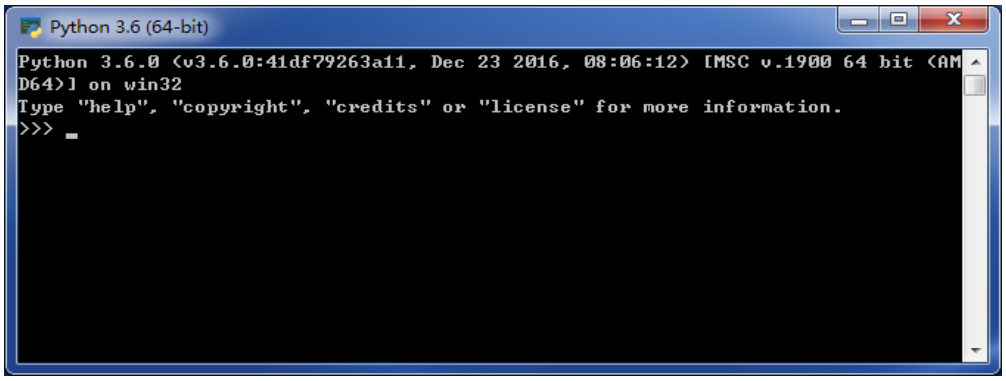
Python 的集成开发环境

别被各式各样的 IDE 的复杂界面吓到，一开始学习 Python 时，可以先从简单的 IDE 入手，以后搞项目开发时，再用功能更全面的 IDE。

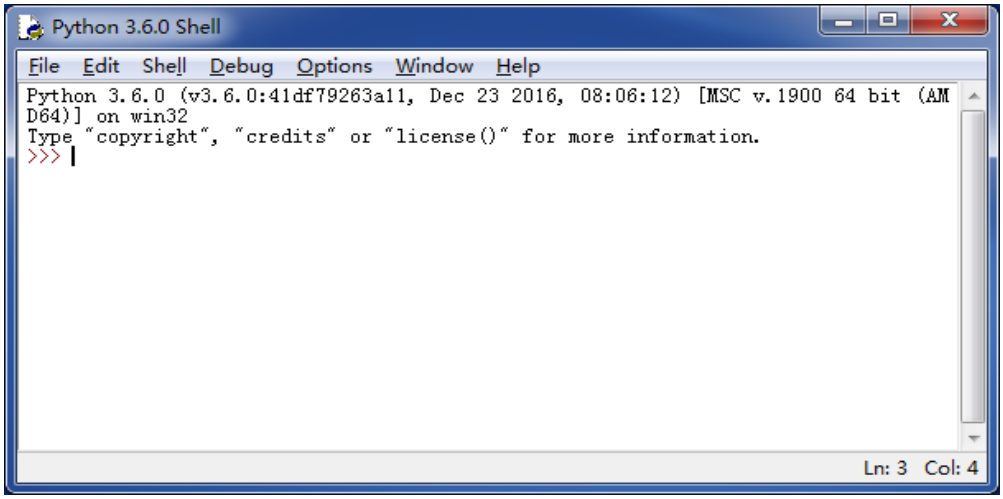
实际上 Python 的安装包里已经包含了一个简单的 IDE，名称叫 Integrated Development and Learning Environment，简称为 **IDLE**，集成开发和学习环境，多了“学习”两个字，也就是你可以用它来学习 Python。

（1）从 Python 官网上下载安装包，注意 Windows 中有 32 位和 64 位版本的区别，我的机器是 64 位的，所以我下载了 python-3.6.0-amd64.exe 这个安装包。安装过程很简单，一直点下一步就可以安装好。可以看到，我安装的是 Python 3 版本。

（2）从开始菜单中找到 Python 的启动程序，可以看到 Python 3.6(64 bit)的菜单项，注意这个不是 IDE 集成开发环境，而是黑窗口控制台。准确地说，应该是 Python **解释器**，这个概念先不介绍了。



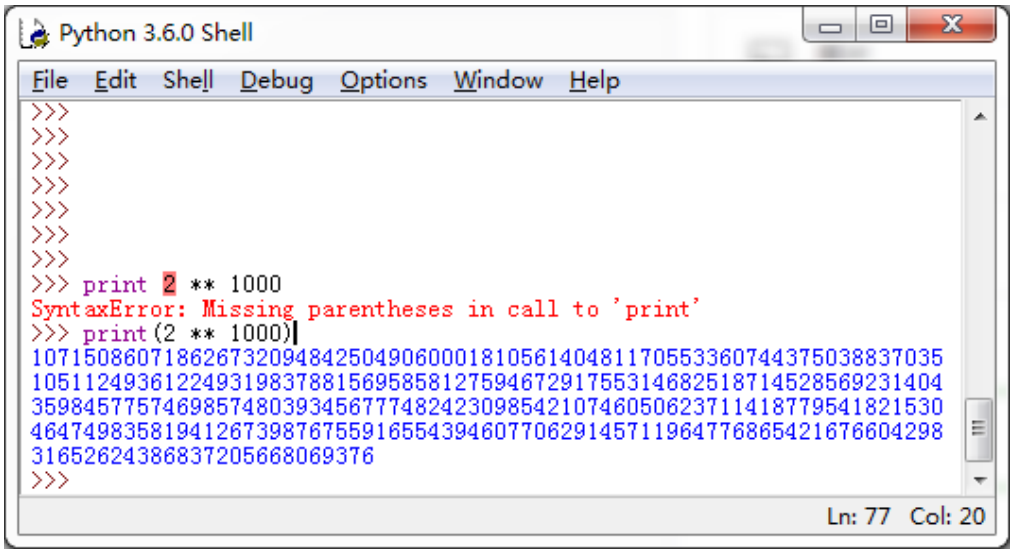
（3）另外一个 IDLE 程序才是集成开发环境（标题栏是 Python 3.6.0 Shell），虽然也比较简单，但暂时够用了。什么是 **Shell**？先别管，以后再说吧。



（4）试试之前在 CodeCademy 学的 print 语句，print 2 ** 1000

出现一行错误信息：SyntaxError: Missing parentheses in call to 'print'

原来 Python 3 的语法已经发生了变化，与版本 2 **不兼容**，最大的一个变化就是这个 print 语句。换成 print(2 ** 1000)就好了。



有了这个 IDLE，你可以把以前在 CodeCademy 上的小练习都拿过来试试，如果出现错误，除了输入错误外，还有可能是 Python 版本的原因。

小练习

留个小练习，我以前发过一篇最枯燥的文章《复利数据表》：

```
(1+0.01) ^ 1 = 1.01
(1+0.01) ^ 2 = 1.02
(1+0.01) ^ 3 = 1.03
... ...
(1+0.01) ^ 364 = 37.41
(1+0.01) ^ 365 = 37.78
```

如何用 Python 如何实现？提示：需要学习**循环语句**。

老程员们可以用其它编程语言试试，能不能用一行语句搞定？

6 打印一行复利数据

上次文章《集成开发环境 IDE》里留了一道练习题：如何用 Python 打印这篇枯燥的《复利数据表》：

```
(1+0.01) ^ 1 = 1.01
(1+0.01) ^ 2 = 1.02
(1+0.01) ^ 3 = 1.03
... ...
(1+0.01) ^ 364 = 37.41
(1+0.01) ^ 365 = 37.78
```

怎样解题

初学者完成这样的任务还是相当有难度的，但不要紧，做练习的过程就是学习并理解编程思维的最有效手段。

记得有一位**黑客**推荐过《怎样解题》这样一本书，写程序与解数学题有相似之处。遇到一个复杂的问题时，首先要将问题分解和简化，然后逐步逼近最终的问题。以后在编程的过程中，还要学习算法和数据结构都是为了掌握这些解题的思路和技巧。当了解的定式越来越多后，编程的思路就越开阔。

简化

比如上面的问题，总共有 365 行数据，我们只需要会打印其中的一行，再按照这种思路打印其它 364 行即可。这里随便取一行，假设是第 3 行。现在的问题就变为：如何打印第 3 行？

```
(1+0.01) ^ 3 = 1.03
```

这样问题是不是简化了许多？

进一步将问题分解

（1）输出等号左侧内容

根据等号，可以拆为两个部分，左边就是一串普通的文本。在最早的《Hello World》里就学过了，这里复习一下：

```
print( "(1+0.01) ^ 3" )
```

注意这里用的是 Python 3 的语法，在 print 后面必须有小括号，引号内的内容会原封不动地输出。

（2）输出等号右侧内容

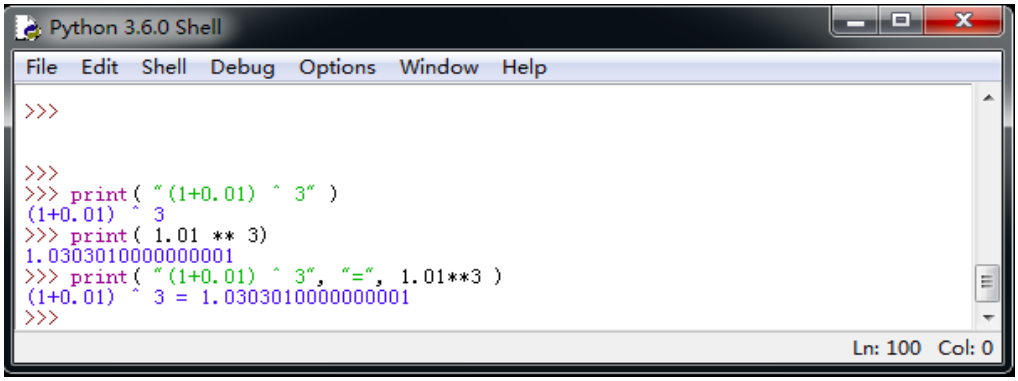
在《Hello World》这一篇文章里介绍过计算 2**10000，即 2 的 1 万次方，而这里我们只是计算 1.01 的 3 次方，简单吧？代码就是这样的：

```
print( 1.01**3 )
```

最后，把左侧、等号、右侧放在一起用 print 输出，即：

```
print( "(1+0.01) ^ 3", "=", 1.01**3 )
```

初学者一定要在集成环境 IDLE 中把这行代码敲上一遍，因为标点符号也不能错！执行的效果：



是不是与我们期望的结果已经很像了？只不过 1.0303010000000001 实在是太太太精确了，而我们只想显示两位小数 1.03 就够用了。这个问题暂时超过了我们当前正在学习的内容，以后再说。

逼近最终的问题

我们现在已经能够输出第 3 行了，我们再照这样把其它 364 行写出来？思路是对的，但计算机擅长做重复的事，可以用**循环语句**来彻底解决这个问题。只需要加一行语句即可，而不用辛苦地抄上 364 遍。

7 赋值语句

在上一章，我们已经能够输出其中的第 3 行数据，源程序只有一行：

```
print( "(1+0.01) ^ 3 ", "=", 1.01**3 )
```

与下面这行代码的结果是一样的，仔细看一下它们的区别：

```
print( "(1+0.01) ^", 3, "=", 1.01**3 )
```

如果我想输出第 100 行数据，则是：

```
print( "(1+0.01) ^", 100, "=", 1.01**100 )
```

《复利数据表》共有 365 行，难道要这样写上 365 次？肯定不会，实际上这个 print 语句中，只有那个指数发生变化，从 1 到 365，如果用引入一个**变量**，则每次的 print 语句就不需变了，源代码变成 2 行：

```
i = 3
print( "(1+0.01) ^", i, "=", 1.01**i )
```

输出第 100 行？就这样：

```
i = 100
print( "(1+0.01) ^", i, "=", 1.01**i )
```

可以看到，print 那一行只字未改。现在我们还没有学**循环**语句，但用笨办法也可以完成任务了。只要写上 365 次赋值语句，分别把 i 的值设置为 1 到 365，再复制、粘贴那行 print 语句 365 次，就可以打印复利数据表了。

更多说明：

```
i = 1
......
i = 365
```

这两行语句就是**赋值**语句，几乎所有的编程语言都类似。等号左侧是**变量名称**，右侧是一个数值（准确地说，应该是**表达式**，这个先不讲）。理解这个赋值语句的时候要从右向左读，把 3 赋给变量 i，再把 100 赋给变量 i。此时，计算机会把**内存**中一个地方放上数值 3，然后再放上 100，把以前的 3 冲掉。

变量的名称由字母和数字构成（下划线也行），要以字母打头，中间不能有空格。例如：i、Money、qq360、ALPHA、a1b2c3 这些都有正确的变量名。在 Python 中甚至可以用汉字当变量名，例如：

```
申龙斌 = 1971
```

但我从来没见过程序员这么用，因为这种程序将来用在其它地方时很可能会有副作用，知道就行但别这样用！

下面这行代码，初学者需要理解一下。

```
i = i + 1
```

从右向左读，表示把当前 i 的值加上 1 之后，赋给变量 i 中，效果就是 i 增一。实际上有些程序员画**流程图**时，写成这样 i + 1 → i，容易理解一些。

总结一下要点：

- 赋值语句是最基本的一种语句
- 等号左侧是**变量名**
- 等号右侧是数值（准确地讲，是**表达式**）
- 变量可以重新赋值
- 变量名由字母、数字、下划线组成，字母打头

8 FOR 循环

在上篇文章《赋值语句》之后，已经可以输出这个枯燥的《复利数据表》的任意一行数据了，例如用这两条语句：

```
i = 100
print( "(1+0.01) ^", i, "=", 1.01**i )
```

可以输出第 100 行数据，像这样：

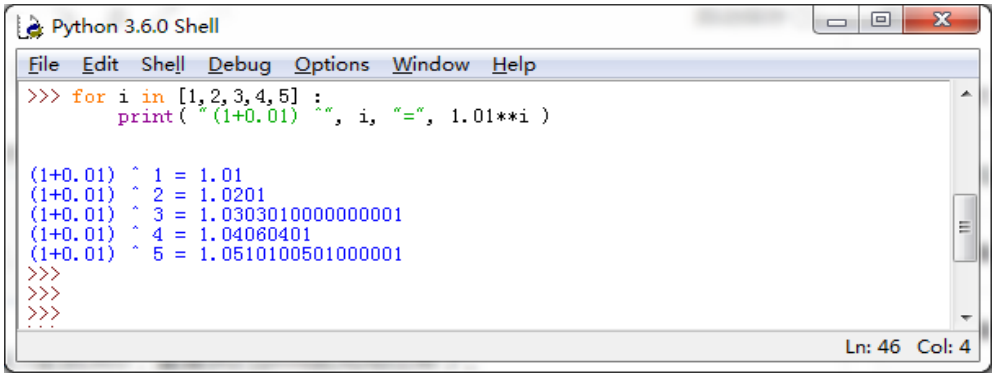
```
(1+0.01) ^ 100 = 2.7048138294215285
```

我们只要让变量 i 从 1 变到 365，再不断重复执行 print 就完成任务了，这里要用到一个重要的 for 语句。

请 Python IDLE 环境中输入这两行代码，看看效果：

```
for i in [1,2,3,4,5] :
    print( "(1+0.01) ^", i, "=", 1.01**i )
```

确保只字不差地输入每一个字符，包括空格、冒号、引号等各种符号，正确结果是这样的：



重点说明：

- for 是循环语句的**关键词**，表示后面要执行循环动作
- [1,2,3,4,5] 是一个**列表**，里面有 5 个元素
- i in [1,2,3,4,5] 表示 i 依次取值为 1，2，3，4，5
- for 语句的末尾有个冒号，别忘了
- 注意 for 下一行的语句，前面有 4 个空格，如果你在 Python IDLE 中，这 4 个空格是系统自动添加好的
- Python 中的空格有缩进的效果，同时也是有语法含义的，这里先理解为上一行（即 for 语句）中要循环执行的语句
- print 语句之后回车，没反应，再回车一次，程序才显示结果

现在我们已经可以输出 5 行复利数据表了，你可以试试如何输出最后 5 行？

```
(1+0.01) ^ 361 = 36.30913774096189
(1+0.01) ^ 362 = 36.672229118371504
(1+0.01) ^ 363 = 37.03895140955522
(1+0.01) ^ 364 = 37.409340923650774
(1+0.01) ^ 365 = 37.78343433288728
```

9 print 语句

在上一章里，我们只写了两行代码，可以输出 5 行《复利数据表》。零基础的朋友对于 print 这条语句的写法会比较迷惑，首先它不是把内容输出到打印机上，而是显示在屏幕上，另外拆解一下：

1）"(1+0.01) ^ " 这里用引号括起来的一串文本，就是一个**字符串**，英文称为 string，以后再细说。

2）语句 print("(1+0.01) ^ ") 与《零基础学编程 002：Hello World》类似，只不过输出的内容换了而已，也就是说放在引号内的文字会原封不动地输出到屏幕上，注意，不含引号。

3）print 语句里，可以输出多个字符串，比如：

```
print( "申龙斌", "的", "程序人生")
```

将输出以下内容：

```
申龙斌 的 程序人生
```

也就是说，在输出多个字符串时，中间会自动用一个空格隔开。

4）print 不仅可以输出字符串，还可以输出数值，比如 1.0201

```
print( 1.0201 )
```

5）循环语句当 i=2 时，与下面的语句等价：

```
print( "(1+0.01) ^", 2, "=", 1.0201 )
```

它会连续输出字符串、数值，中间用空格分隔。

小练习：

请在 IDLE 集成环境中输入：

```
print( "申龙斌", "的", "程序人生", sep='' )
```

10 只显示 2 位小数

我们仍要继续解决这个问题：如何用 Python 打印这篇枯燥的《复利数据表》？

```
(1+0.01) ^ 1 = 1.01
(1+0.01) ^ 2 = 1.02
(1+0.01) ^ 3 = 1.03
... ...
(1+0.01) ^ 364 = 37.41
(1+0.01) ^ 365 = 37.78
```

学完《零基础学编程 007：FOR 循环》之后，我们已经可以输出这样五行数据：

```
(1+0.01) ^ 361 = 36.30913774096189
(1+0.01) ^ 362 = 36.672229118371504
(1+0.01) ^ 363 = 37.03895140955522
(1+0.01) ^ 364 = 37.409340923650774
(1+0.01) ^ 365 = 37.78343433288728
```

但有一个明显的问题，电脑计算得太太太精确了，每个数后面都拖着 10 多位的小数，而我们只要两位小数就够了，怎么办呢？

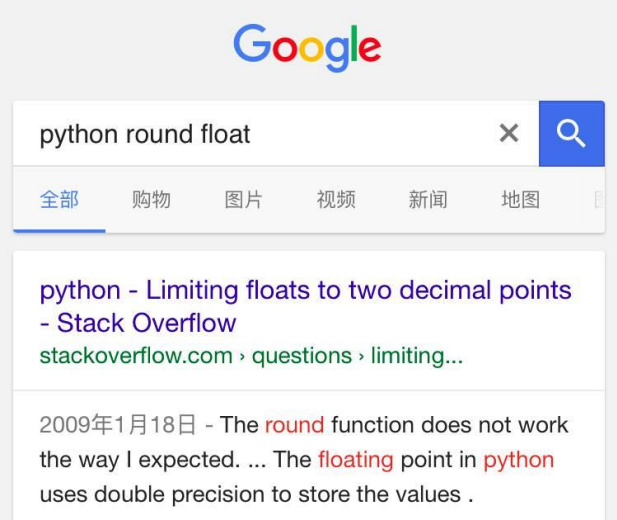
编程新手到这里可能就束手无策了，此时《“零基础学编程”都需要哪些基础？》里提到的 2 项技能就要发挥作用了。

1）英语基础

一种容易想到的办法是对第 3 位的小数部分进行四舍五入的运算，在英语里四舍五入叫 round，小数在计算机的世界里经常被称为浮点数 float。

2）搜索技能

现在需要 google 出场了，输入关键词“python round float”：



如果没有 vpn，只能用某度试试了，你自己对比一下搜索结果吧。点击“这个链接”到 stackoverflow 上看看那篇排在第一位的详细答案有多详细吧，纯英文的。

3）立即到 python 的 IDLE 中试试：

```
>>> round(36.30913774096189, 2)
36.31
>>> round(36.672229118371504, 2)
36.67
```

效果不错，大概意思很容易猜出来，round(x, 2)就是把 x 这个数保留两位小数。

4）把第 7 课《零基础学编程 007：FOR 循环》的代码抄过来，稍加修改，在 python IDLE 环境中执行一下，看看效果：

```
for i in [361,362,363,364,365] :
    print( "(1+0.01) ^", i, "=", round(1.01**i, 2) )
```

小结：

1）英语基础好绝对大有帮助

2）会用搜索，能够快速定位到有用的资源，节省大量的时间

3）立即在 python IDLE 中尝试，帮助快速理解代码的作用

4）知道了 round(x, 2)函数的意思

5）多看看资料，实际上 round(x)就是四舍五入的取整

6）联系到电子表格 EXCEL，你也用过 ROUND(x)函数吧？道理都是一样的，许多编程语言中的取整函数都叫 round

7）在 stackoverflow 的原文中还提到了另一种更为通用的解决办法，是用 format 函数，这里先不介绍了

11 终于可以输出完整的复利数据表了

我为了用 Python 打印这 365 行枯燥的《复利数据表》，竟然啰啰嗦嗦地写了五小章。

```
(1+0.01) ^ 1 = 1.01
(1+0.01) ^ 2 = 1.02
(1+0.01) ^ 3 = 1.03
... ...
(1+0.01) ^ 364 = 37.41
(1+0.01) ^ 365 = 37.78
```

实际上最常用的编程算法之一就是将问题分而治之，一个大的软件**系统**都是由一些**模块**组成的，每个模块又是由一些**函数**（或者**类**）组成的，**函数**是由代码行组成的，代码行要符合严格的编程**语法**规定，各种零件拼装起来后就完成了我们想要的功能。

通过以前的内容，我们可以打印前五行的复利数据表：

```
for i in [1,2,3,4,5] :
    print( "(1+0.01) ^", i, "=", round(1.01**i, 2) )
```

也可以打印后五行的复利数据表：

```
for i in [361,362,363,364,365] :
    print( "(1+0.01) ^", i, "=", round(1.01**i, 2) )
```

当然，我们可以打印出 365 行复利数据表了，可能有人马上想到了这样的办法：

```
for i in  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1
8, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 7
1, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 10
5, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 11
9, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 13
3, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 14
7, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 16
1, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 17
5, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 18
9, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 20
3, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 21
```

```
7, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 23
1, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 24
5, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 25
9, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 27
3, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 28
7, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 30
1, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 31
5, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 32
9, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 34
3, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 35
7, 358, 359, 360, 361, 362, 363, 364, 365] :
    print( "(1+0.01) ^", i, "=", round(1.01**i, 2) )
```

注意这里只有 2 行语句，前面从 1 到 365 都在一行里。当然这样从 1 写到 365 会被计算机和人类所耻笑的，计算机擅长做这种重复性的、有规律的事情，需要找到一个函数能够输出连续的自然数。

这就是 range 函数。试着在 python 里输入：

```
list(range(1,366))
```

这个函数正好能够输出从 1 到 365 的一个**列表**，至于 range 和 list 有何区别，先不用管，现在我们能够只用两行语句输出 365 行的复利数据表了：

```
for i in range(1,366) :
    print( "(1+0.01) ^", i, "=", round(1.01**i, 2) )
```

没错，只需这两行语句就可以完成我们的任务。这两行语句虽然简单，但已经包含了变量、FOR 循环、列表、缩进语法、函数（这里有 3 个：range、print、round）、算术表达式（1.01**i），字符串等一系列的基本概念。

学会了这种编程思路，改用其它编程语言实现也会非常容易，比如用 C#语言实现的版本：

```
for (int i = 1; i < 366; i++)
    Console.Out.WriteLine("(1+0.01) ^ " + i + " = "
          + Math.Round(Math.Pow(1.01, i), 2));
```

这里再把要点总结一下：

1）需要提醒的是我这里的代码全部可以在 Python 3.0 以上版本通过，在 2.7 版本时语法有不一样的地方，需要修改才行。

2）复杂的问题要采用分而治之的办法，把一个大的问题拆解为一些小的问题，逐个击破后，每次进步一点点，不断迭代来逼近最终的结果。

3）你只要明白了《零基础学编程 002：Hello World》，就会知道引号内的内容称为**字符串**，print 可以输出字符串，另外还可以输出**浮点数**。

4）我们非常熟悉“加+、减-、乘*、除/”的**运算符**，Python 中还支持**运算符，这里是两个乘号，表示乘方运算。1.01**3 表示 1.01 的 3 次方。

5）**变量**的运用，解决了第 3 行的输出，只需要把变量的值换一下，就可以输出其它行的内容，而 print 语句并不需要修改。这样程序代码可以重用，还不容易出错。

6）**循环**语句 for 的运用，可以让变量在某个范围内变化，重复执行 N 次。循环是计算机编程中的一种重要结构，实际上最容易理解的就是**顺序**结构，即代码从上到下依次执行。

7）round 函数，可以把数四舍五入。

8）range 函数，可以产生一个从 1 变到 365 的列表。

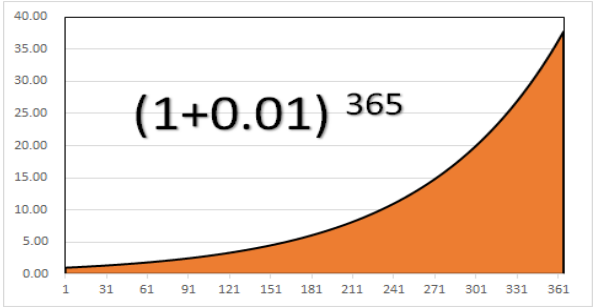
还需要提醒程序员新手要非常注意的地方，就是**严格区分全角、半角标点符号**，只要不是出现在字符串里，其它的地方都是**半角**的标点符号，否则程序就会出错。新手在切换输入法时的疏忽，经常会输入全角的标点，甚至是全角的空格！

12 画出复利曲线图

前面我们输出了 365 行的《复利数据表》：

```
(1+0.01) ^ 1 = 1.01
(1+0.01) ^ 2 = 1.02
(1+0.01) ^ 3 = 1.03
... ...
(1+0.01) ^ 364 = 37.41
(1+0.01) ^ 365 = 37.78
```

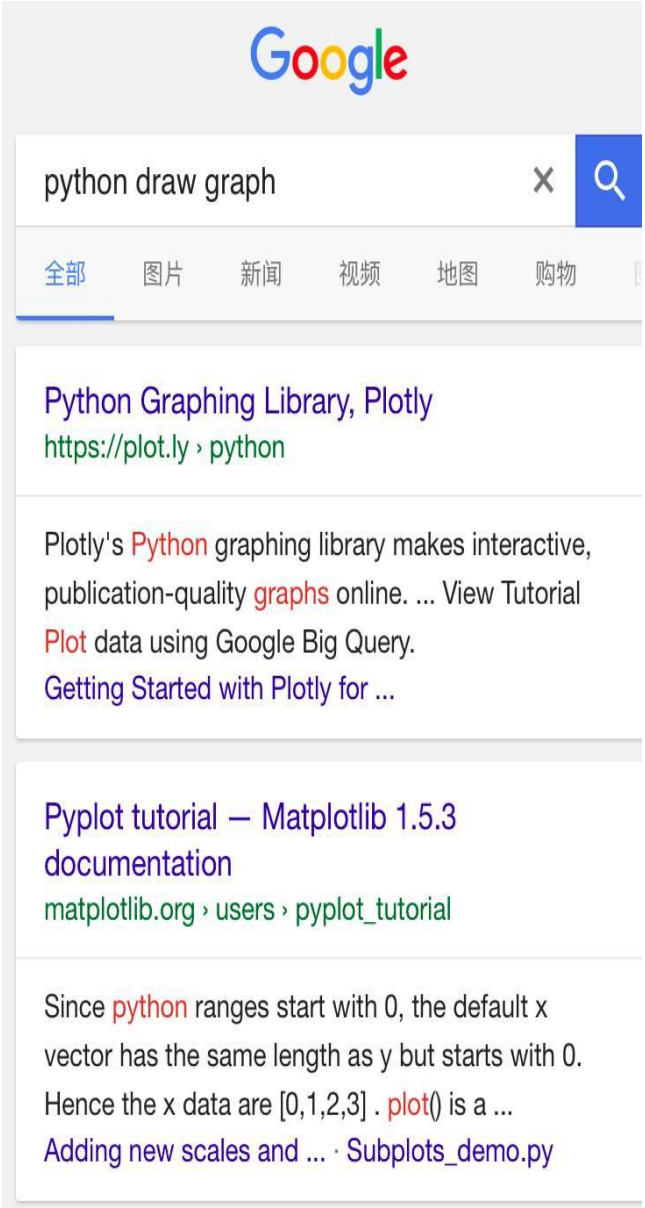
但能否用 Python 画出像下图这样的复利曲线图？



由于我也是从头开始学 Python，不过我不是零基础，而是学过了 N 种其它的编程语言，我只是特意没有去看 Python 入门书籍，想尝试着能否通过搜索和阅读来解决这个问题。

1）用谷歌搜索

因为曲线图、条形图等等在英文里都叫 graph，所以搜索的关键词就选 python draw graph。



看到两种主要答案，点进去快速浏览了一下，发现 plotly 是一个商业包，有免费受限版和商业版；而 matplotlib 也非常有名，相关资料非常丰富，用户群非常大，就选后者 matplotlib 了。

2）安装 matplotlib

搜到 matplotlib 的官网：http://matplotlib.org，由于没有仔细地阅读安装说明，直接运行了下面两行命令：

```
python -m pip install -U pip setuptools
python -m pip install matplotlib
```

在黑窗口 cmd.exe 中给出了大量提示后，以安装失败告终，好像是一些依赖库无法安装。

返回去重新读这一段在 Windows 平台上安装的说明：

If you don't already have Python installed, we recommend using one of the scipy-stack compatible Python distributions such as **WinPython**, **Python(x,y)**, **Enthought Canopy**, or **Continuum Anaconda**, which have matplotlib and many of its dependencies, plus other useful packages, preinstalled.

这里提到的 WinPython 等几种预安装包中已经包含了大量可用的库或模块，直接安装它们就行。可惜在下载 WinPython 的 200 多 M 的安装文件时，网络不争气，还不支持断点续传，结果半个小时也没有下载下来。在缓慢的下载过程中，我又试了一些其它办法，踩过了不少坑，这里全略过。

可行的办法：直接到 http://winpython.github.io 官网去下载 WinPython 即可，有许多版本可用，我最终用的是 3.5.2.3Qt5 这个版本。请根据自己的机器是 32 位还是 64 位，选择相应的版本。比如：32 位的安装包名称是：WinPython-32bit-3.5.2.3Qt5.exe

安装过程非常简单，设定一个安装目录，我设的是 C:\WinPython-32bit-3.5.2.3Qt5，然后一直按下一步即可。

另外还有一个 20M 的 WinPython 3.5.2.3 Zero 精简版本，需要联网运行命令来安装 matplotlib，也可以成功，这里不介绍了。

3）先画数学函数图

通过 matplotlib 官网上的相关链接找到一篇教程：

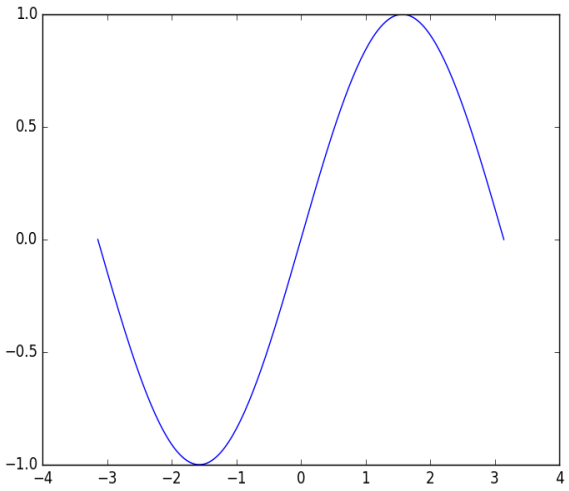
http://www.labri.fr/perso/nrougier/teaching/matplotlib/

运行 C:\WinPython-32bit-3.5.2.3Qt5 目录中的 IDLEX (Python GUI).exe，这个集成环境比以前的 Python IDLE 功能好像要强一点，输入以下代码：

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(-np.pi, np.pi, 200)
```

```
Y = np.sin(X)
plt.plot(X,Y)
plt.show()
```

正弦曲线图出现了，只用 6 行语句，是不是很兴奋？

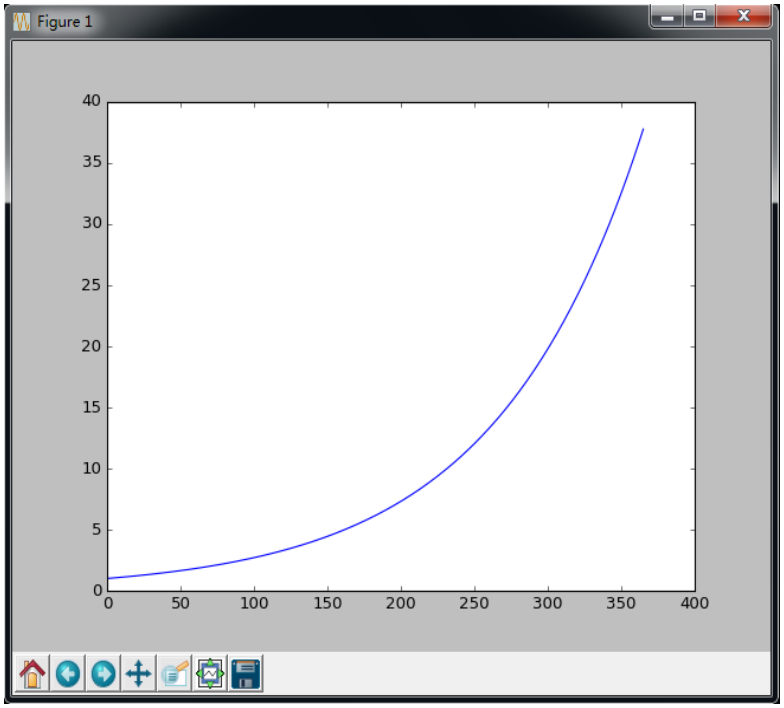


4）修改 2 行画出复利曲线图

由于有 JAVA 语言和 R 语言的基础，所以很容易猜出来 np.linspace 应该是生成出一组 x 坐标，Python 中的**符号在其它语言中都有相应的 power()函数，所以把上面的代码简单修改一下（加粗的地方）：

```
import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(1, 365, 365)
Y = np.power(1.01, X)
plt.plot(X, Y)
plt.show()
```

果然，简洁、清爽的复利曲线图真就画出来了。



具体原理以后再琢磨，先小结一下：

1）下载 WinPython 软件包，找那个 200 多 M 的，安装完成后，matplotlib 及其它一些库就全部装好了，这些是绘图时都要用到的，这种安装方法最简捷。尽管还有其它各种安装包和复杂的安装过程，但新手最好就别试了，先把图画出来再说。

2）在 WinPython 中的集成环境叫 IDLEX，比 IDLE 多了字母 X

3）输入 6 行代码，可以画出正统曲线图

4）照猫画虎，修改 2 行语句，简单的复利曲线图完成！编程是不是很 easy？

13 import 让你飞起来

在《零基础学编程 012：画出复利曲线图》这篇文章中只需 6 行语句就画出复利曲线图，前两行语句全是 import。

```
import numpy as np
import matplotlib.pyplot as plt
```

你可以在 Python IDLE 中输入 import antigravity （反重力）这行语句，回车，就可以在浏览器中看到在 Python 界非常有名的漫画。

```
import antigravity
```

关于这幅漫画的解释：

1）漫画中的对话来自 Python 程序员和 Perl 程序员

2）“飞翔”用来比喻 Python 程序员写代码有种“自由、惬意”的感觉

3）传统的 C 和 JAVA 语言的 Hello World 程序都非常麻烦，需要 N 行才能搞定，而在 Python 中只需一行即可

4）Python 是一种**动态类型**语言，这个一句话解释不清，以后再说

5）**空格**在 Python 中是有语法含义的，使程序更易读，在“FOR 循环”中已经可以看到这种空格的缩进

6）**类** class、**函数** function、**常量** constant 都被封装为**模块** module，Python 中提供了大量直接可用的模块库，只需 import 导入后就可以使用

7）关于这幅漫画的详细英文解释，点击“这个链接”。

再来看看我们用来画图的这行 import 语句：

```
import matplotlib.pyplot as plt
```

意思是：导入 matplotlib 库中的 pyplot 模块，这个模块中提供了许多强大的绘图命令。as 后面的 plt 是个别名，以后只需输入 plt 就可以代表 pyplot，减少输入的字符数。

而 numpy 则是一个科学计算的程序包，编程新手只要知道它非常强大就够了，先把例子抄会了，能运行，以后再慢慢理解。

```
import numpy as np
```

练习：

试着在 Python IDLE 中输入下面这行 import 语句，看看 python 的编程哲学，试试搜搜其中文版：

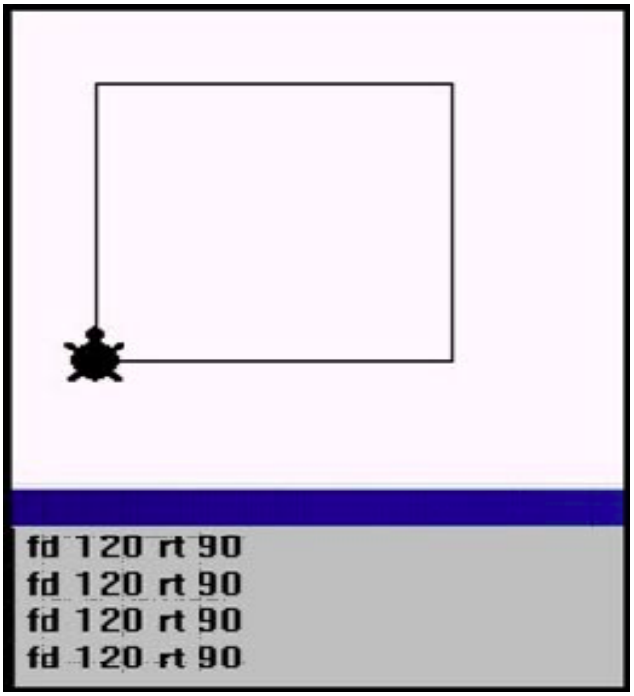
```
import this
```

14 小海龟做画

在《零基础学编程 012：画出复利曲线图》这篇文章中，我们使用了强大的 matplotlib 和 numpy 模块，可以用几行代码画出复杂的图形来。但对于初学者来说，里面的语句理解起来还是非常有难度。既然是零基础，可以看看很久以前孩子们是如何开始学编程的。

1967 年，Daniel G. Bobrow, Wally Feurzeig, Seymour Papert 和 Cynthia Solomon 设计了 LOGO 编程语言，用一种直观的方式教孩子们学习编程。尽管该语言也可以解决复杂问题，但给大家留下最深刻印象的是它里面的**海龟绘图系统**(Turtle Graphics)。

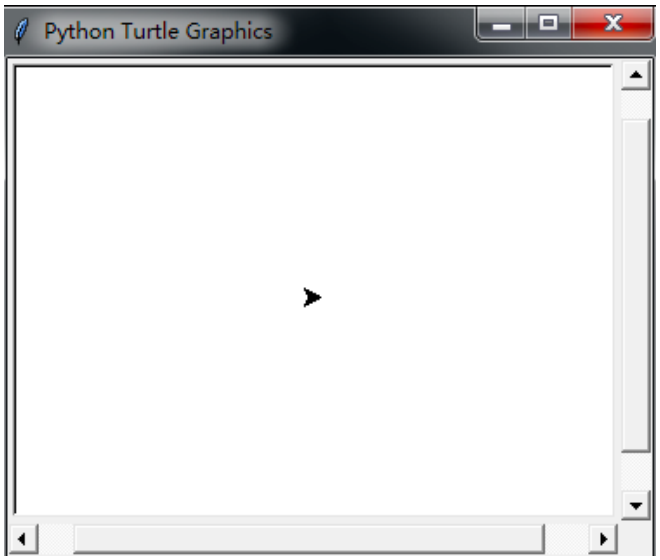
通过向海龟(turtle)发送命令，用户可以直观地学习程序的运行过程，因此它很适合于儿童学习，当然也可以用于几何教学。我以前就在一些学习机上运行过 LOGO 环境，还带着自己的孩子在上面做画，估计她现在早就忘了这段故事了。



强大的 Python 语言肯定也不会遗忘小海龟，在 Python 环境中已经内置好了 turtle 模块，你只需要在 Python IDLE 中输入 2 行语句，就可以让小海龟就位。

```
import turtle
turtle.reset()
```

这时屏幕上会弹出一个图形窗口，那个朝向右方的小箭头就代表小海龟。

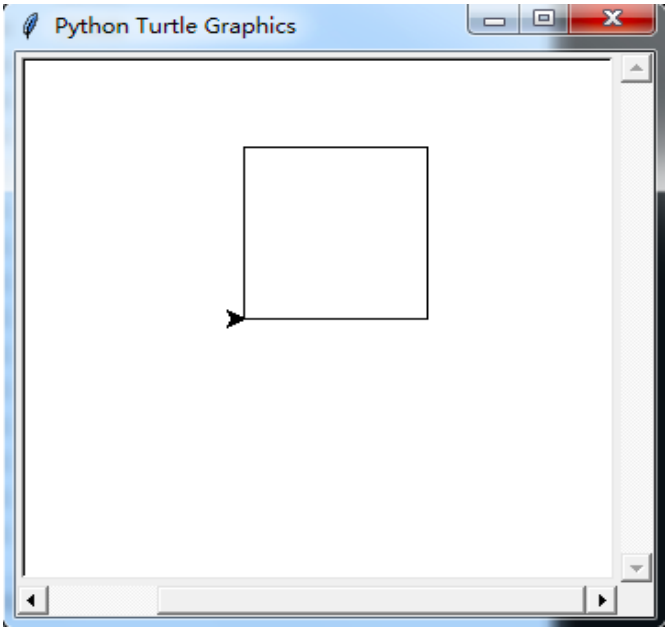


现在试着给小海龟发送命令，forward(100)表示向前方走 100 步，计算机上 1 步就是 1 个像素，left(90)表示向左转 90 度。

咱们来个郭冬临在小品《问路》中的“左转左转左转左转，好像是一个圈呦”：

```
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
```

当然画出来的是一个正方形。



我们已经在《零基础学编程 007：FOR 循环》里学过了循环语句，可以把代码简化为重复 4 次的前进和左转，效果一样：

```
for i in range(4) :
    turtle.forward(100)
    turtle.left(90)
```

这些指令虽然简单，但你可别小瞧它，一段计算机程序也就是**顺序**、**条件**和**循环**这三种程序结构。如果再运用上**递归**算法，能够产生许多奇妙的图案。试着复制并粘贴这段代码到 Python IDLE 中，看看它能画出什么图形？

```
from turtle import *
```

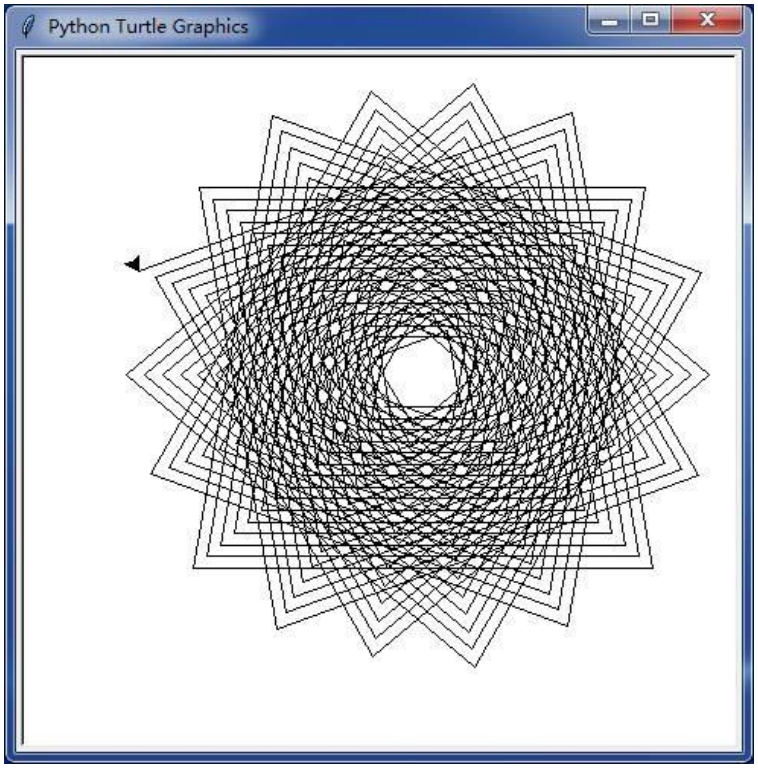
```
def f(length, depth):
   if depth == 0:
     forward(length)
   else:
     f(length/3, depth-1)
     right(60)
     f(length/3, depth-1)
     left(120)
     f(length/3, depth-1)
     right(60)
     f(length/3, depth-1)

f(500, 4)
```

15 画些有趣的图案

从《零基础学编程 014：小海龟做画》中我们学会了基本的做图命令，只需要用上循环语句，就可以画出比较复杂的图案来，比如：

```
from turtle import *

for i in range(255) :
    forward(50 + i)
    left(100)
```

这里总共循环 255 次，每次步子迈得大了一点点，每走一步之后左转 100 度，画出来的图案就是这样：

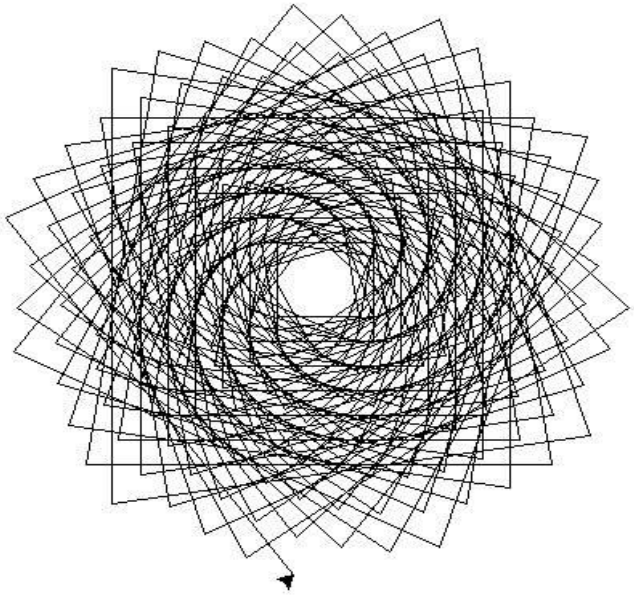


小海龟走得是不是太慢？可以按组合键 CTRL + C 中断程序的运行，在 for 语句前加上 speed(9) 可以让小海龟以最快的速度做画。不过中断之后，屏幕上已经被画乱了，需要用 reset() 清空画布。完整的代码是：

```
from turtle import *

reset()   # 把画布清空，小海龟回到初始点
speed(9)  # 最快速度为 9，最慢速度为 1
for i in range(255) :
    forward(50 + i)
    left(100)
```

这里写了 2 行**注释**，程序员为了让别人或者是几个月之后的自己能够看懂程序，会加上一些解释说明。这是一种良好的编程习惯，黑客只要看你写过的注释，基本就能判断出你的编程水平。Python 中的单行注释非常简单，在#符号之后的全是注释，只是给人类阅读的，计算机会忽略这些字符。

把旋转角度从 100 换成 99，可以得到不同的图案：



还可以加点颜色变化，请自行试验：

```
from turtle import *

reset()
speed(9)

for i in range(255) :
    colormode(255)    # 颜色分量值不超过 255
    pencolor(i, i, i) # 画笔颜色会越来越淡
    forward(50 + i)
    left(99)
```

colormode(255)表示红、绿、蓝三种配色的值不超过 255。

pencolor( )设置画笔的颜色，后面三个参数为 R、G、B 三分量，即红、绿、蓝。

练习：试着运行下面的代码，看看出现什么图案？

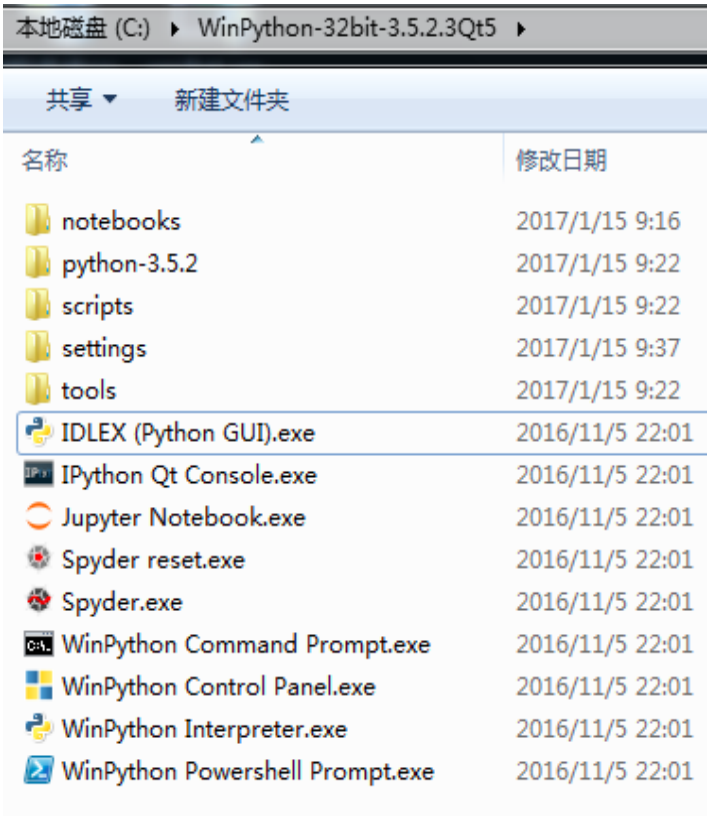
```
import turtle
from turtle import *


def part( total, length, breadth, col ):
    angleInc = 360/total
    width( breadth )
    color( col )
    for i in range(total):
        forward( length )
```

```
        left( angleInc )


def rosette( total, length, width, color, angleInc ):
    for i in range( int(360/angleInc) ):
        part( total, length, width, color )
        left( angleInc )


turtle.setup( 300, 300, 20, 20 )
turtle.speed(9)


rosette(10,40,1,"blue",36)

rosette(5,80,1,"red",36)
turtle.exitonclick()
```

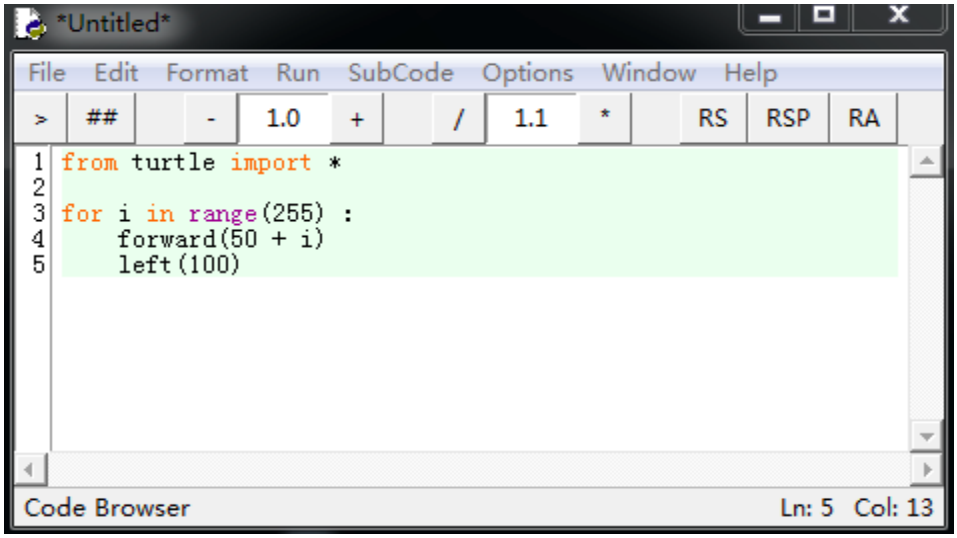
16 Python IDLE 的代码编辑器

Python IDLE 是 Python 的集成开发和学习环境，而 WinPython 集成更多的开发工具包，比如在《零基础学编程 012：画出复利曲线图》提到的 numpy 和 matplotlib 绘图包都包含在内，所以强烈建议在 Windows 环境中的学习者安装 WinPython，省得在安装或运行过程中出现这样那样的错误。

WinPython 中将 IDLE 升级了一下，叫做 IDLEX。



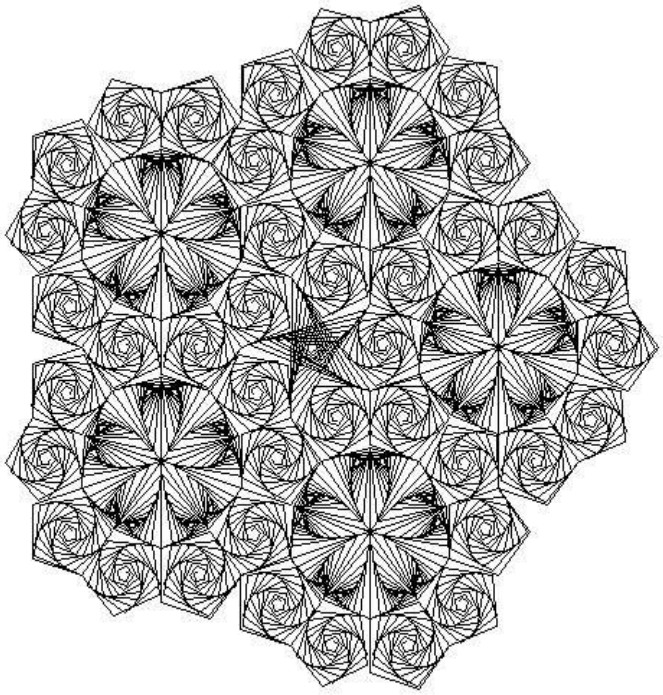
前面我们一直在这个 IDLE 集成环境中输入命令，当代码行较多时，每发生错误后，都要重复输入命令，非常麻烦。实际上 IDLE 中还有一个代码编辑器，从 File 菜单中可以点击 New File 菜单项，就可以调出代码编辑窗口来。



这里面可以直观地看到缩进、行号、列号，可以输入多行的源代码，然后一并运行。大家可以试着把以前写的程序敲进去，点击工具栏右侧的“RS”按钮，就可以运行整个程序，比在 IDLE 窗口中的>>>后面敲入命令更加方便。

相比 Python IDLE，WinPython IDLEX 的菜单里主要多了一项 SubCode，它可以将代码分人为划分为许多块，只执行其中的某一块，非常适合学习和调试。

在 Help 菜单中还有一个 Turtle Demo，里面包含更多的小海龟的演示程序，大家可以依次运行一遍，看看都能产生哪些奇妙的图形。



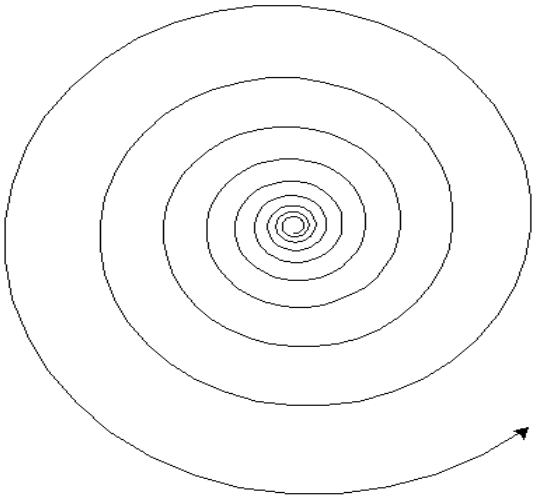
17 画出我的公众号 LOGO

在《零基础学编程 014：小海龟做画》和《零基础学编程 015：画些有趣的图案》里，我们已经可以用编程中的循环结构，加上 turtle 中的前进、转向等指令画些有趣的图案。

有些人已经发现我把公众号图标悄悄地换了，实际上我是用 Python 的 turtle 自己画了一个，以前还请人帮忙做图标，现在发现还不如自己用程序画一个吧。

```
from turtle import *

for i in range(365) :
```

```
    forward(1.01 ** i)
    left(9)
```



forward(1.01 ** i) 表示每天进步一点点，转的圈数有点多，没有颜色、线宽的变化，显得有点单调，再改进一下：

```
from turtle import *

reset()
speed(9)
colormode(255)
bgcolor(136, 177, 221)

for i in range(256) :
    pencolor(255 - i, 0, i) # 逐渐减少红色分量，增加蓝色分量
    pensize(1.01 ** i)
    forward(1.01 ** i)
    left(6)
```

几点说明：

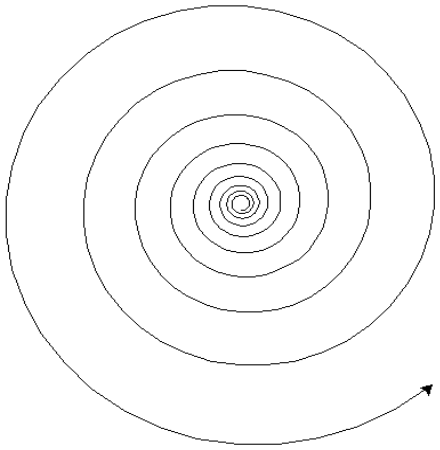
- colormode(255)表示 Red、Green、Blue 三色的分量的值都不超过 255
- bgcolor( )是设置图片的背景色，我从网上找了一种背景，据说这种背景会让公众号显得比较专业
- pencolor( )不断地换画笔的颜色，产生从红渐变到蓝色的效果
- pensize( ) 调节画笔的粗细，里面又用到了复利数据的公式

- 总循环没有用 365，而是 range(256)，一是让颜色控制的代码精练，再是不想转太多圈

18 条件语句

学习了《零基础学编程 017：画出我的公众号 LOGO》之后，可以用几行代码，画出一个螺旋渐开线。

```
from turtle import *
for i in range(365) :
    forward(1.01 ** i)
    left(9)
```



forward(1.01 ** i) 表示每天进步一点点，从画出的圆弧的半径上大致可以看出进步的效果在逐步显现，但如何显示出每周的进步呢？我们可以在每周 7 天的时候画出一个标记来，只需增加 2 行代码即可。

```
from turtle import *
for i in range(365) :
    forward(1.01 ** i)
    left(9)
    if(i % 7 == 0) :
        stamp()
```

程序的三种结构：**顺序**、**循环**、**分支**。

顺序结构非常容易理解，按照代码出现的先后顺序执行，先执行 forward( )再执行 left( )，一步一步地顺序执行，大部分语句都是这样排列的。

循环结构可以让计算机做重复的事，在 Python 中就是 for 语句，在《零基础学编程 007：FOR 循环》里介绍过。

分支结构可以让计算机在满足某种条件时，或者达到某种状态的时候，执行指定的任务。

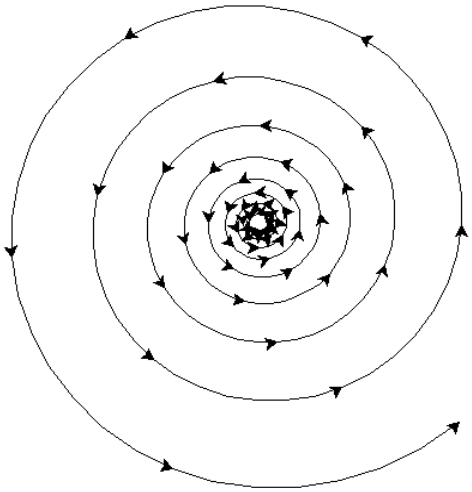
在这里加了 2 行语句：

```
if(i % 7 == 0) :
    stamp()
```

关于 i % 7 == 0 这个表达式暂时不用过多地理解，它表示的意思就是 i 是否能够被 7 整除时，即 i 为 0，7，14，21……时。如果把上面的代码翻译为汉语，可以这样对应：

```
if      (i % 7 == 0)     :
如果     i 能够被 7 整除时      执行下面的语句
```

而 stamp()就是满足条件时要执行的语句，这个函数的作用是画出一个小标记，在 turtle 画图系统中默认是画一个小三角形。整个代码就会每 7 天显示出一个三角形，可以看出每周的进步是不是越来越大？



小结：实际上一个程序主要就这三种结构：顺序、循环和分支。将这些结构组合起来，就会非常强大，可以完成各种各样的任务。

练习：试着在 for 语句之前加上一行语句 shape("turtle")看看会出现什么图形？

19 生成群文章目录

在 2016 年 10 月底，我建立了“分享与成长群”，每人在每月都要输出一篇原创文章，一开始人数不多，汇总成 PDF 的工作量并不大，但现在人数已经超过 70 人了，该写个程序来解决这种重复性的工作了。

最终问题描述：

群分享的文章已经用 Mikecrm 表单工具采集到一个 xls 文件中，包含“姓名、文章标题、文章链接”三列，想生成一份所有文章的合集，用 PDF 格式分享出来。

可以在公众号后台输入“**群分享**”，看看以前几期的群分享 PDF 文档。

姓名	文章标题	文章地址
申龙斌	45岁大叔的5年GTD旅程	http://mp.weixin.qq.com/s/F9pTuAkoDCXAaWvqDGcTFA
王大永	不养儿不知儿女恩 （二）	www.jianshu.com/p/b283e0cd0743
王冰	面对不确定性，该咋办？	http://mp.weixin.qq.com/s/x-ObcGozu2czy3MCywtbQw
孙瑶	"你的爱好是跑步，和我们说一说吧"	http://mp.weixin.qq.com/s/ESWq-rtNJuzYRyJBOFNfyA
周平	我想，生活就应该是这样	http://mp.weixin.qq.com/s/gJuewLntBIuFnm_yZybooQ
于帅	被拒100天挑战	http://www.jianshu.com/p/9ee22c03cff9
贺玲玲	财富自由之面对机会	http://www.jianshu.com/writer#/notebooks/7935838
李娜	自控力的解读	http://mp.weixin.qq.com/s/3IJ4563VV4ZKwlkNL16DTQ
李荣强	互联网总会给你带来惊喜	http://mp.weixin.qq.com/s/7x4oLdc2NNXxmKHWdCYpmQ

XLS 文件样例

问题分解：

直接生成 Doc 或 PDF 相当有难度，根据《怎样解题》的策略，面对相对复杂的问题，首先要办法把问题分解为多个简单的过程，我把该问题分解为两步：

- 第一步：先根据 XLS 生成一份 HTML 文章目录，点击文末左下角的“**阅读原文**”看输出的最终效果
- 第二步：再利用开源的转换工具生成 DOC 或 PDF，我已经有思路，正在试验中，以后再发布

本次先解决第一步的问题。对于编程新手来说，这一步也具有相当的难度，仍需要进一步地分解。**最主要的难点在于编程新手很可能缺少读取文件、CSV、Markdown、HTML 等知识背景。**

1. 把 XLS 手工转换为 CSV 文件
2. 写 Python 程序，把 CSV 转换为 Markdown 格式

3. 把 Markdown 复制在“简书”平台中，即可直接发布，完成任务

什么是 CSV？

就是一种逗号分隔的文本文件（Comma-Separated Values），详细内容请自行百度。

为什么不直接读 XLS？

读文本文件相对容易些，Python 中内置有专门的读取 CSV 的函数库，容易上手。当然也能找到读取 XLS 的函数库，但门槛相对高一些。

什么是 Markdown？为什么不直接用 HTML？

HTML 是网页的描述语言，但它的描述太啰嗦了，而 Markdown 就相当简洁，可以让人专心写文章，减少排版的干扰，详细内容请自行百度吧。

试着读取 csv

假设 201701.csv 文件存放在 D 盘根目录下，百度一下 python 中的 csv 读取教程，原来只需要 4 行，就可以读出其全部内容。

```
import csv
reader = csv.reader(open('d:/201701.csv'))
for line in reader :
    print(line)
```

网上查到的许多文章中写的是下面这行语句，在 Python 3 中运行会报错。

```
reader = csv.reader(open('d:/201701.csv', 'rb'))
```

这里先不介绍 open 函数的具体意思，有经验的 C 程序员看见 'rb' 可以猜出问题的原因。

用 markdown 表示的超链接

我们想要让文章标题显示为一个超链接，点击后跳转到文章的 http 超链接。在 markdown 中是这样表达的，非常简洁。

```
[title](URL)
```

举个例子：

```
[45 岁大叔的 5 年 GTD 旅程](https://mp.weixin.qq.com/s/peTWyaIPO78l577PXvSZVA)
```

把这段文字放在简书中（请开启 markdown 选项），发布之后，就会看到一个超链接：45 岁大叔的 5 年 GTD 旅程。

最终代码：

```
import csv
reader = csv.reader(open('d:/201701.csv'))
for line in reader :
    name = line[0]  #第一列是姓名

    title = line[1]  #第二列是标题
    url = line[2]  #第三列是链接

    print(name, "[" + title + "](" + url + ")" )
```

总共写了 7 行代码，中间三行是为了让新手看明白，实际上用 4 行代码就够了。运行程序后，输出的内容是这样的：

把这些文字复制到简书平台，发布，完成任务。

小结：

- 问题描述：xls -> pdf
- 分步解决：xls -> csv -> markdown -> html -> pdf
- 首先解决：csv -> markdown，其它步骤用手工解决
- csv 是逗号分隔的文本文件，用文本编辑器可以查看
- import csv 用于导入 csv 函数库
- csv.reader 可以直接读入 csv 文件，形成一个列表
- 在 markdown 语法中，[title](URL) 表示超链接
- 用 print 语句拼出想要的文本
- 把 markdown 文本复制到简书中，发布为 html 网页

20 强大的列表推导

问题描述：找出 50 之内的所有勾股数。

所谓勾股数，就是三个正整数，满足 x*x + y*y = z*z。例如：3，4，5 或 5，12，13。

电脑解题只会用笨办法，一个一个地试，x、y、z 都从 1 递增到 49，三重循环搞定。

```
for x in range(1, 50) :
    for y in range(1, 50) :
        for z in range(1, 50) :
            if x*x + y*y == z*z :
                print(x, y, z)
```

输出的结果中有许多重复答案，比如：3 4 5 和 4 3 5 实际上算一种答案。可以加个要求：x<y<z，代码稍微修改即可。

```
for x in range(1, 50) :
    for y in range(x, 50) :
        for z in range(y, 50) :
            if x*x + y*y == z*z :
                print(x, y, z)
```

可能大家感觉这个题目太初级，但这里想提的是 Python 中强大的**列表推导（list comprehension）**。

比如：输出 1 到 9 的平方数，在 Python 中用一行语句：

```
[x*x for x in range(1, 9)]
```

输出：[1, 4, 9, 16, 25, 36, 49, 64]

有了这个武器，前面的问题可以用一行语句搞定：

```
[(x,y,z) for x in range(1,50) for y in range(x,50) for z in range(y,50) if
 x*x + y*y == z*z]
```

输出结果：

```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 1
5), (9, 40, 41), (10, 24, 26), (12, 16, 20), (12, 35, 37), (15, 20, 25),
(15, 36, 39), (16, 30, 34), (18, 24, 30), (20, 21, 29), (21, 28, 35), (24,
 32, 40), (27, 36, 45)]
```

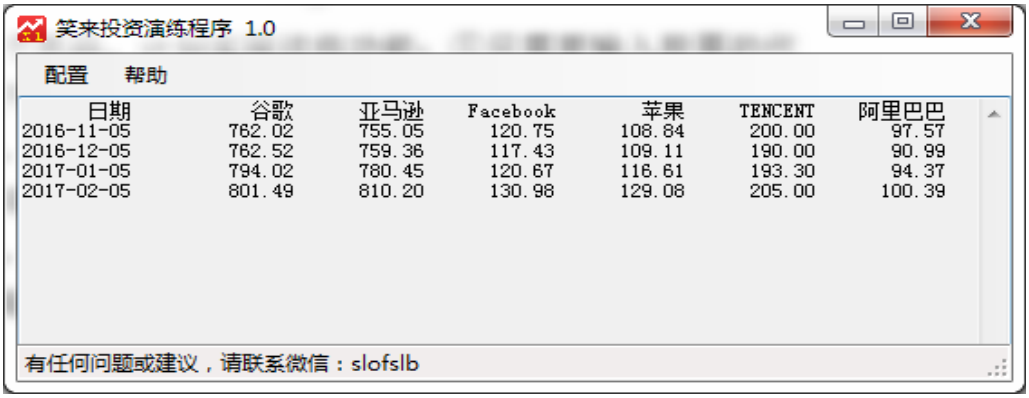
小结：

Python 语言中的列表 List 借鉴了许多现代语言的优点，可以用比较简短的语句写出一些强大的表达式。为了得到一个数组 array 或列表 list，直接用**列表推导**就可以方便地搞定。

知乎上有个贴子：一行 Python 能实现什么丧心病狂的功能？链接在“这里"。

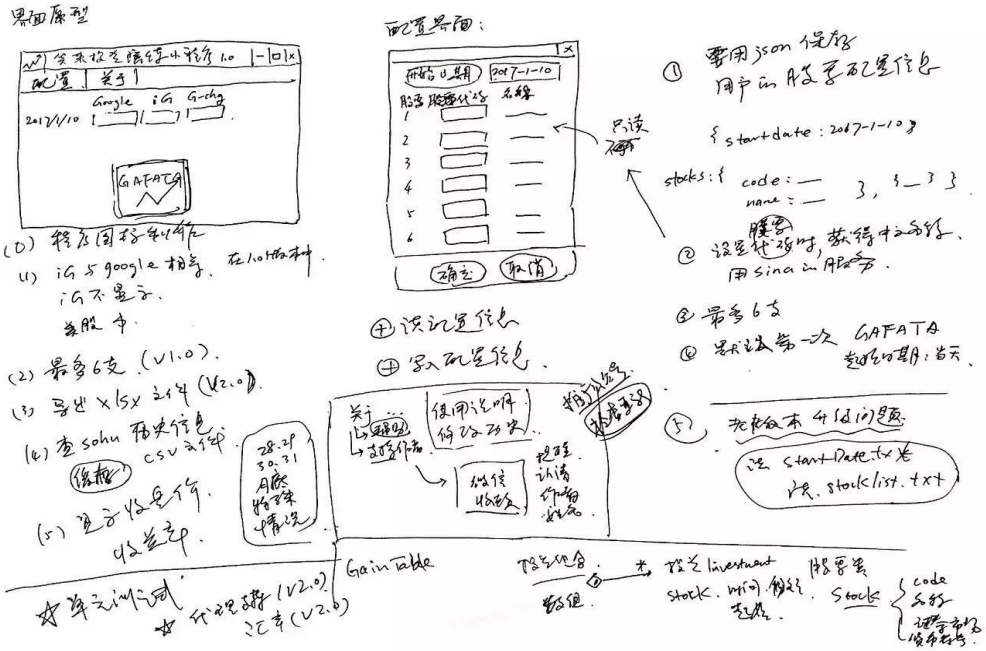
21 获取股票实时行情数据

春节期间重写了“笑来投资演练程序 1.0 版”这个程序，可以每个月自动更新几支股票的行情数据。程序的功能不复杂，但是编程新手想实现它仍有相当的困难。为了短时间内完成主要功能，我使用了最熟悉的 C#编程语言，先看实际完成的效果图。



像学英语一样，只有“**用**”英语才能学好英语，而我们学编程，只有“**用**”编程解决实际问题，才能学好编程。所以将来我准备用 Python 语言把股票小程序再实现一遍，即：**用 Python 写一个程序，可以每月更新几支股票的行情数据**。

单单凭这一句话是无从下手的，得先做**需求分析**，关于需求分析的话题以后再说，我简要地画了一张草图，把想实现的功能和原型记录了下来。



这张图考虑了一些后续的功能，但我只挑选了最基础的功能作为 1.0 版本。面对这样一个程序，利用《怎样解题》中的策略，要将其逐步简化直至我们可以应付：

- 简化：先做一个无界面的程序，可以每月更新几支股票的行情数据
- 再简化：写一个无界面程序，获取几支股票的行情数据
- 再简化：写一个无界面程序，获取一支股票的行情数据
- 再简化：写一个无界面程序，获取“谷歌”股票的行情数据
- 再简化：写一个无界面程序，获取“谷歌”股票的当日开盘价

到了这里，问题就已经相当简化了，我们可以准备动手编程了，再把今天的问题描述清楚。

问题描述：

不要界面，获取“谷歌”股票的当日开盘价。

问题分析：

以前谈过“零基础学编程”都需要哪些基础？先要会用谷歌搜索，我首先用的关键词：python、股票实时行情。

浏览并对比几个搜索结果，发现“Python 获取新浪财经股票数据”这篇文章的原理最简单，不过文章中附带的源代码有问题。

1.新浪股票实时行情

新浪财经可以提供行情数据，但是新浪没有提供API接口，不过我们可以抓包来获取所需要的数据,比如我们可以访问：

http://hq.sinajs.cn/list=sz601688

我们可以获取到证券代码sh601688的当天实时数据：

var hq_str_sh601688="华泰证券,30.76,30.40,31.73,33.18,30.50,31.72,31.74,30361 6031,9697545287,13200,31.72,165800,31.71,119200 ,31.70,18400,31.69,108281,31.68,1100,31.74,347608, 31.75,16600,31.76,14698,31.77,16000,31.78,2015– 05–22,15:04:07,00";

其中的含义如下：

"华泰证券"，股票名字；
”30.76″，今日开盘价；
”30.40″，昨日收盘价；
”31.73″，当前价格；

解决过程：

1）先找到谷歌的股票代码

这个过程略过了，到新浪网上可以找到，直接给出结果：“gb_goog”。

2）试验一下文章中的办法是否可行

在电脑的浏览器中输入“http://hq.sinajs.cn/list=gb_goog”，立刻得到一串行情数据：

```
var hq_str_gb_goog="谷歌,824.16,0.63,2017-02-17 21:25:47,5.18,819.93,82
4.40,818.98,841.95,663.28,1287626,1182462,565835289600,27.88,29.56,0.0
0,1.03,0.00,0.00,686560000,71.00,821.67,-0.30,-2.49,Feb 17 08:25AM EST,
Feb 16 04:00PM EST,818.98,485.00";
```

网上文章的说明完全正确，只是对于美股来说，开盘价之后并不是收盘价。

3）写代码，做试验

```
import urllib.request as req

with req.urlopen('http://hq.sinajs.cn/list=gb_goog') as f :
    hq = f.read()
    print(hq)
```

运行的结果：

```
b'var hq_str_gb_goog="\xb9\xc8\xb8\xe8,824.16,0.63,2017-02-17 21:25:47,
5.18,819.93,824.40,818.98,841.95,663.28,1287626,1182462,565835289600,2
7.88,29.56,0.00,1.03,0.00,0.00,686560000,71.00,821.67,-0.30,-2.49,Feb 1
7 08:25AM EST,Feb 16 04:00PM EST,818.98,485.00";\n'
```

结果出来了，可惜在字符串前面有一个字母 b，尾部多了\n，中间还有一串怪字符。查阅 Python 的语法说明，原来这个 b 表示得到的是**二进制**字符串，中文不能正常显示。有经验的程序员一眼可以猜出，那个神秘的 \xb9\xc8\xb8\xe8 应该是 GBK 或 GB2312 编码的“谷歌”。涉及到字符编码的问题，又是一个大坑，绝对可以讲上 7 天 7 夜，这里先不细说了。

4）修改源代码，正确显示中文

查 urllib 库的帮助文档，里面有一个例子，可以把二进制串转换为指定的编码字符集，我们猜测是 GBK 编码，只需更换这一行代码：

```
    hq = f.read().decode('GBK')
```

这次运行后中文可以正确显示，也没有了首字母 b！

5）获得开盘价

汉字“谷歌”之后的数值 824.16 就是开盘价，而收盘价是哪一个暂不清楚，留在以后再说。我们只需要把 824.16 正确取出来就基本完成任务了。曾经在《生成群文章目录》这里介绍过 CSV，这一串字符也是逗号分隔的，但只有一行，用 Python 自带的字符串函数就行了。

split(',') 函数可以把一个字符串从逗号的位置切开，生成一个列表，而 v[1]就是 824.16，正是我们想要的开盘价。再修改一下代码，任务完成。

```
    v = hq.split(',')
    print( v[1] )
```

小结：

- 复杂的问题先分解

- 学会使用搜索
- 用到了 urllib 库
- 以 b 字符开头的串是二进制串
- 二进制的知识一定要学会，计算机只认二进制
- 字符集转换是个复杂的坑，老程序员都被坑无数
- split 函数可以切分出一个列表

22 函数的世界

通过《零基础学编程 021：获取股票实时行情数据》的学习，我们已经可以取出“谷歌”股票的开盘价，今天我们要取出 GAFATA 共 6 支股票的开盘价。

先回顾上次的代码：

```
import urllib.request as req

with req.urlopen('http://hq.sinajs.cn/list=gb_goog') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])
```

要获取 6 支股票行情，那就先找到 GAFATA 的股票编码（gb_goog，gb_amazn，gb_fb，gb_aapl，gb_hk00700，gb_baba），然后把这段代码复制几遍，把股票编码替换一下就可以完成任务了。

```
import urllib.request as req

with req.urlopen('http://hq.sinajs.cn/list=gb_goog') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])

with req.urlopen('http://hq.sinajs.cn/list=gb_amzn') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])

with req.urlopen('http://hq.sinajs.cn/list=gb_fb') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])
```

```
with req.urlopen('http://hq.sinajs.cn/list=gb_aapl') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])

with req.urlopen('http://hq.sinajs.cn/list=rt_hk00700') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[2])  #注意港股有所不同

with req.urlopen('http://hq.sinajs.cn/list=gb_baba') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])
```

任务虽然完成，但这种代码太重复了，有四行代码重复了 6 次，只是股票编码不同而已。如果以后我们要取出收盘价，又得把所有语句中带 v[1] 的地方都换掉，非常不方便。所有编程语言中都提供了“**函数 function**”这个特性来解决重复代码的问题。

编程新手学习时，可以把编程语言中的**函数**与数学中的**函数**进行类比，数学函数中有函数名、变量、公式、函数值，在编程语言中分别对应着函数名、参数、函数体、返回值。数学中有二元函数，程序中有多个参数。

例如：我们在《零基础学编程 011：复利数据表问题（总结）》里遇到的复利公式就是一个简单的数学函数。

$$f(x) = (1 + 0.01)^x$$

在 Python 中定义一个函数非常方便，刚才的复利公式可以这样写：

```
def f(x) :
   return (1 + 0.01) ** x
```

关键词 **def** 表示定义一个函数块，可以用英语单词 **def**ine function（定义函数）来助记，定义完成后，以后只需**调用函数（call function）**就可以得到相应的结果。

```
print( f(100) )
print( f(365) )
```

函数的一个主要功能就是减少重复的代码，便于将来的维护。回到我们的股票程序上，我们定义一个函数，给出股票编码，返回开盘价。

```
def price(stock) :
    url = 'http://hq.sinajs.cn/list=' + stock
    with req.urlopen(url) as f :
```

```
        hq = f.read().decode('GBK')
        v = hq.split(',')
        return v[1]
```

有了函数定义之后，只需要调用 6 次，就可以得到 6 支股票的行情了。

```
print( price('gb_goog') )
print( price('gb_amzn') )
print( price('gb_fb') )
print( price('gb_aapl') )
print( price('rt_hk00700') )
print( price('gb_baba') )
```

其实 print 也是函数，不过它是 python 系统自带的，也叫**内置函数**（built-in function）。而我们用 def 定义的函数，称为**自定义函数**。

小结：

- 函数 function 可以减少重复性代码，便于将来的维护
- 有内置函数，我们自己写的叫自定义函数
- python 中大量模块中包括了大量函数
- def 关键字用于定义一个函数块
- 冒号之后的各行是函数体，要注意缩进
- 用 return 返回想要的结果
- 调用函数的基本形式：函数名(函数参数)

上面的程序中有一个小问题，你能否发现？你能自己解决它吗？

23 用 with 实现优雅地释放资源

在《零基础学编程 022：函数的世界》中我们写了一个函数，通过访问新浪的实时行情服务，得到股票的开盘价。

```
import urllib.request as req

def price(stock) :
    url = 'http://hq.sinajs.cn/list=' + stock
    with req.urlopen(url) as f :
        hq = f.read().decode('GBK')
        v = hq.split(',')
        return v[1]
```

新手对 **with** 的用法不太理解，如果以前学过 C#语言，这个 **with** 类似于 **using** 关键词。先来看看不太好的写法吧：

```
import urllib.request as req

def price(stock) :
    url = 'http://hq.sinajs.cn/list=' + stock
    f = req.urlopen(url)   # 尽量不要用这种写法！
    hq = f.read().decode('GBK')
    v = hq.split(',')
    f.close() # 当前面发生异常时，不一定能够执行到这条语句
    return v[1]
```

上面这段代码没有用 **with** ... **as** ... 的写法，而是用赋值语句把 req.urlopen(url) 赋给了 f，在返回开盘价 return v[1] 之前调用了 f.close() 把网络连接关闭。

在绝大多数情况下，这种代码不会有什么问题。但这里的代码访问了网络，而访问网络会有各种异常情况，比如网卡被禁用、WIFI 未连接、无法连接互联网、网络地址无效、代理设置不正确、网络服务器故障、防火墙阻挡等等，这些异常都是编程之前无法完全预料的。

我们调用 urlopen() 打开了一个网络连接，在最后务必要保证把它关闭，即调用 close() 函数。但当网络已经发生异常了，此时还未执行到 close() 函数，程序就已经异常退出了，所以网络连接可能仍处于打开状态。

一般的小程序，这少量的未关闭的网络连接并不会造成什么麻烦，有时操作系统还会在**进程**关闭时自动释放这些连接，但如果编写服务端程序时，几秒钟之内就可能产生数千个并发连接，当这种问题积累到一定程度后，程序就会出现莫名其妙的错误，而且这种错误特别难定位。

我在 2002 年用 java 写过一个网络信息发布系统，当时有人的代码里没有正确地释放 Oracle 数据库连接，当正式上线时，几分钟之内产生了数百个未释放的数据库连接，Oralce 主数据库差点宕掉，幸好我们及时地把程序摘掉，才避免了一次事故的发生。

所以学习编程时，一定要参考别人的例子代码，尤其是参考官方的例子代码。网上流传的一些核心代码只是为了说明具体的用法，写法上并不规范，也没有加入异常处理的相关代码，而真正产品级的代码，会加上许多边界条件检查、异常判断的语句，从而让产品更加健壮。

小结：

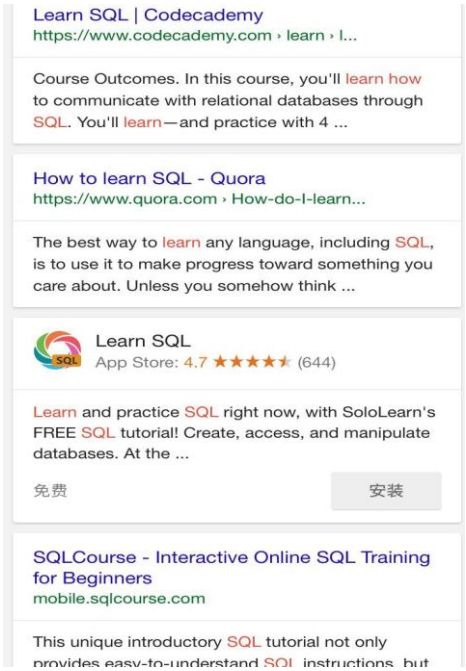
- **with** 语句用于保证一些资源（文件、网络连接、数据库等）在发生异常时能够正常地关闭或释放
- 编程初期就养成良好地编程习惯，将错误扼杀在摇篮里

- **with** 语句内部会自动调用 close()语句释放网络连接，其背后还有比较复杂的实现机制，以后再说

24 如何快速学会 SQL？

一位朋友问我如何能够较快地学会 SQL，我一时还真不知道如何回答。想学会 SQL（结构化查询语言），大概需要理解这些术语：数据库、关系型数据库、面向对象的数据库、键值型的数据库、数据表、数据记录、数据列、数据表的关系运算等等，但对于一个对数据库一无所知的人来说，该如何一步一步地让他建立起这个知识体系？

我尝试着先把大脑中的相关知识清空，利用《“零基础学编程”都需要哪些基础？》中提到的搜索技巧，先试着 google 一下，关键词用“how to learn sql"，第一行的广告被我去掉了，后面的几条结果如下图：



1、CodeCademy

在开始学习 python 的时候，我写过《零基础学编程 001：用在线编程环境快速上手》，学习一门语言需要快速上手来建立信心，一步一步地输入代码并马上看到反馈结果，时刻体会到自己的进步，从而进入程序的世界，而不要在安装编程环境方面就花上 2 天时间，使刚建立的一点点学习热情严重受挫。

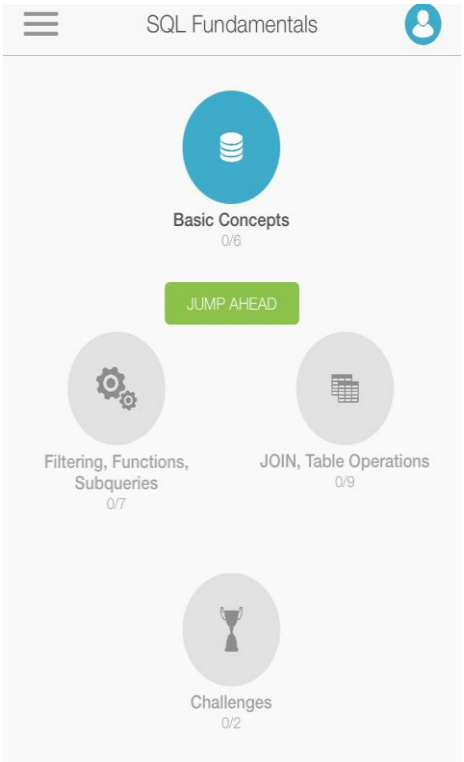
这个 Codecademy 也支持 SQL 的学习，我进去试了一下，第一课就是输入 SELECT 语句，直接看到查询的结果。可惜课程界面和讲解都是英文的，英语基础好的朋友首选用这个网站学 SQL 或其它语言。

2、Quora

这个网站相当于英文版的知乎，点进去之后，列出来一堆的学习资源和教程，因为是用户自己评价排名，所以比较可信，自己去看吧。

3、Learn SQL

这是一款手机 APP，我简单地安装试用了一下，仍是纯英文的，设计成关卡模式，不完成一关无法进入下一关。一上来介绍数据库的基本概念，让你回答问题，有一定难度，需要注册之后才能使用更多功能。



4、SQLCourse.com

一个专门学 SQL 的网站，大段大段的英文内容，介绍得非常详细。从 google 上搜索到的前四条结果相当不错，如果英文不好，你就只能在知乎上搜索“如何自学 SQL”，也有不错的回答。

5、通过 python 学 SQL

单单学 SQL 效果不好，需要与编程语言配合学习，才能明白 SQL 在编程中的强大用处。既然已经学了一些基本 Python 知识，能否通过 python 的环境快速上

手 SQL？我尝试着搜索了几个相关模块库，最后锁定了这个 db.py，能够达到上述目的。

1）安装

如果你之前安装过 WinPython，则直接跳到下一步，看清楚了，是 **WinPython**。

之前介绍过 **WinPython** 这个安装程序包含许多常用的模块库，如果你只是安装纯净的 Python 3.6，后面的 import 语句会出现问题，新手还是尽量别被安装问题折磨得死去活来吧。重复一遍，在 Windows 上推荐安装 **WinPython** 软件包，有人试验小海龟画图出错，很可能也是这个原因。

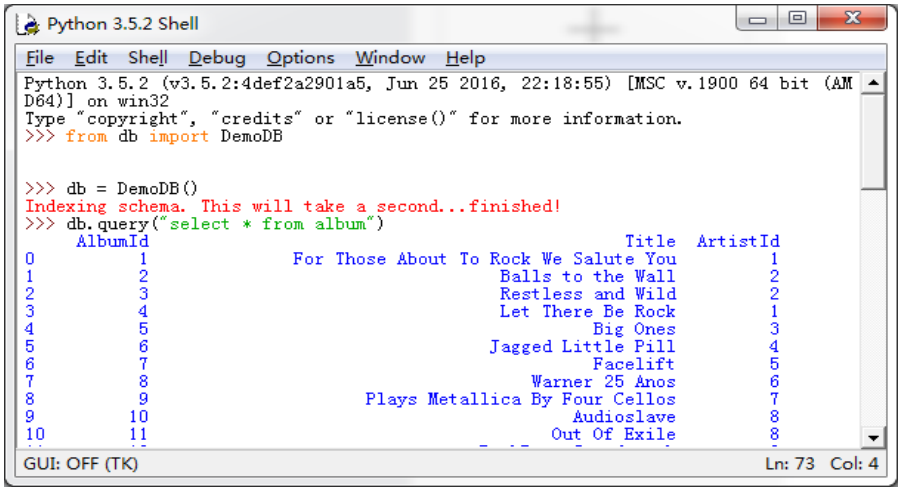
2）快速试验 SELECT 语句

SELECT 语句相当于编程语言中的 Hello World，如果将来你有幸成为了一名程序员，并经常与数据库打交道，那么你写的 80%的 **SQL** 都是 **SELECT** 语句！

直接在 python 的集成环境 IDLE 中输入下面三行语句：

```
from db import DemoDB
db = DemoDB()
db.query("select * from album")
```

查询结果：



这个样例是一个歌曲的**数据库(Database)**，album 是一个**数据表(Table)**，就像 EXCEL 电子表格一样，它由许多**列(Column 或 Field)**组成，这里有三列：AlbumId（歌曲 ID）、Title（曲名）、ArtistId（艺术家 ID），这个数据表还有许多**行(Row)**，一行称为一条**记录(Record)**。

再看看我们以前在《零基础学编程 019：生成群文章目录》学过的 **CSV**，是不是挺类似？其实 **CSV** 就是一个文本格式的**数据表**。

我们刚才写的 select * from ALBUM 就是一条最简单的 SQL 语句，意思是查询 ALBUM 中的所有**记录(Record)**，SQL 语句一般不区分大小写，这种语言有点像自然语言，只说明目的，而不关心背后的实现逻辑，复杂的处理逻辑由**数据库管理系统(DBMS)**来完成。

今天先介绍一个开头，大家是否对 SQL 感兴趣？欢迎留言，如果留言人数超过 20，我就专门写上几篇有关数据库和 SQL 的文章。

小结：

- SQL 是数据库的结构化查询语言
- 想快速学 SQL，首先得会搜索
- 英文基础好，学编程会容易许多
- 快速上手反馈建立学习的信心
- CodeCademy 可以快速学习，值得一试
- python 中可以用 db 模块学习 SQL
- 数据库由数据表构成，表由行和列组成
- select 是基础的 sql 语句，以后操作数据库时 80%以上都是写这条语句

25 学什么编程语言最有前途？

想学习编程的朋友可能一直纠结于到底学哪一种编程语言最有前途，我 google 了一下，在维基百科的下面这个页面里大概有 500 多种编程语言，这些相对来说还是比较知名的编程语言，不包括一些语言的方言以及一些标记性的语言。

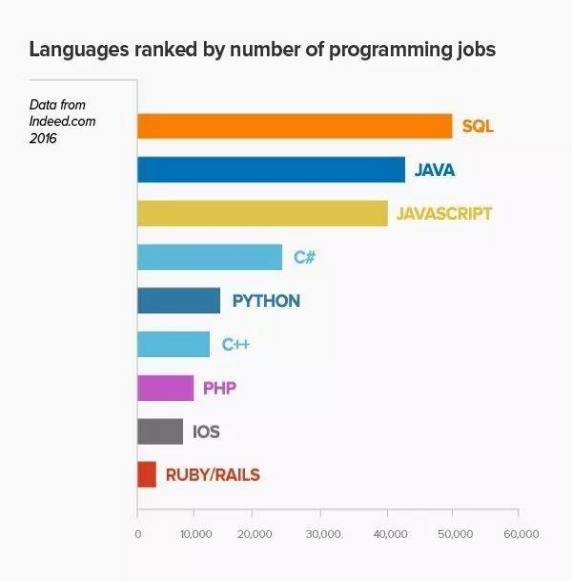
https://en.wikipedia.org/wiki/List_of_programming_languages

市场上哪种程序员最抢手？我也 google 了一下，发现了 2016 年和 2017 年的两篇贴子：

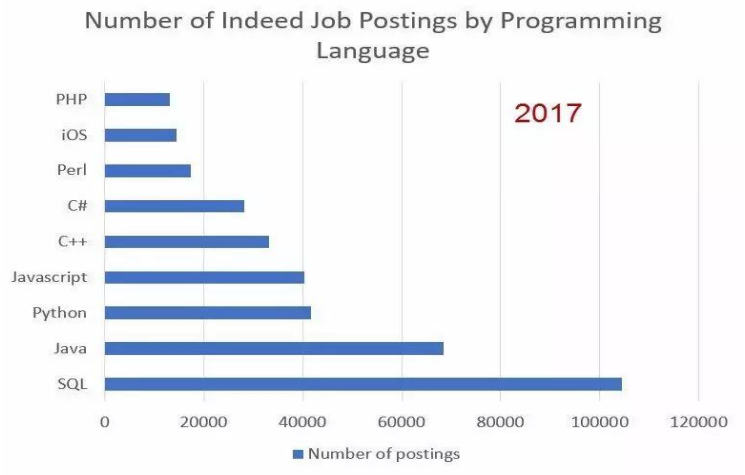
http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2016

http://www.codingdojo.com/blog/9-most-in-demand-programming-languages-of-2017

先看 2016 年的语言热度排名，SQL、JAVA、JavaScript 占据前三位，后面分别是 C#、Python、C++、PHP、iOS、Ruby。



再来看看 2017 年的情况，SQL、Java、Python 占据前三位，实际上 JavaScript 与 Python 差不太多。



另外有一个网站几乎每个季度都会公布一次编程语言排名，它是著名的 TIOBE，它的排名算法大概也是根据网上相关文章多少、搜索关键词的频度等计算出来的。2017 年 2 月公布的最新结果是 Java、C、C++、C#、Python 位居前五。(文章写于 2017 年，2019 年 3 月的最新排名中，Python 位居第三)

Feb 2017	Feb 2016	Change	Programming Language	Ratings	Change
1	1		Java	16.676%	-4.47%
2	2		C	8.445%	-7.15%
3	3		C++	5.429%	-1.48%
4	4		C#	4.902%	+0.50%
5	5		Python	4.043%	-0.14%
6	6		PHP	3.072%	+0.30%
7	9	^	JavaScript	2.872%	+0.67%
8	7	v	Visual Basic .NET	2.824%	+0.37%
9	10	^	Delphi/Object Pascal	2.479%	+0.32%
10	8	v	Perl	2.171%	-0.08%
11	11		Ruby	2.153%	+0.10%
12	16	^^	Swift	2.125%	+0.75%
13	13		Assembly language	2.107%	+0.28%
14	38	^^	Go	2.105%	+1.81%
15	17	^	R	1.922%	+0.73%
16	12	vv	Visual Basic	1.875%	+0.02%
17	18	^	MATLAB	1.723%	+0.63%
18	19	^	PL/SQL	1.549%	+0.49%
19	14	vv	Objective-C	1.536%	+0.13%
20	23	^	Scratch	1.500%	+0.71%

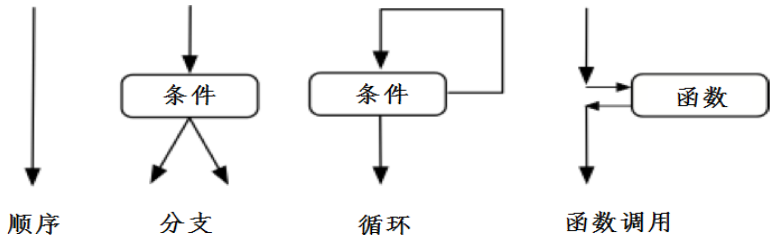
而关于“哪种语言最好？”这种问题一直在各种论坛上争吵不休，这种问题实际上是一个伪命题，何为最好？写的代码行最少？执行效率最高？可以并行执行？维护最方便？可以处理海量数据？最接近于自然语言？用不同的评价指标会得出完全不同的结果。

停止纠结

与其在多种编程语言中纠结和争吵，不如从排名前 10 的语言中挑一种快速入手，所有的主流编程语言基本上都是相通的。编程的基本原理是相同的，背后的**算法**没有变，**数据结构**也类似，只是**语法**稍有不同而已。

明白了编程的基本原理，换成另外一种语言相对来说比较容易，只是换一种语法结构去重写而已（当然背后还有庞大的类库要熟悉，这些都需要时间去熟悉）。对于某种特定的功能，有些语言写起来啰嗦一些，有些语言写起来简捷许多；有些语言执行效率高，有些语言运行稍微慢一点；有些语言可直接适用于多核 CPU 并行，有些语言的并行化需要做大量工作。

我学过 N 种编程语言，现在看来语法是最容易学的部分，最麻烦的部分在于要去熟悉大量的模块或类的使用方法，还要学会正确的使用方法。我们以四种基本的流程结构来看看几种语言的语法有何不同吧，一个程序的执行流程大致可以分为这四种：**顺序**、**分支**、**循环**、**函数调用**。



1）顺序

这种结构太简单了，所有的语言都是从上至下的执行，没什么可说的。

2）分支

Python 代码：

```
if(i % 7 == 0) :
    stamp()
else :
    forward(1)
```

JAVA、C#、C 代码：

```
if(i % 7 == 0) {
    stamp();
} else {
    forward(1);
}
```

3）循环

Python 代码：

```
for i in range(1,366) :
    print( "hello world" )
```

JAVA、C#代码：

```
for(int i=1; i<=365; i++) {
    打印语句;
}
```

C 代码：

```
int i;
for(i=1; i<=365; i++) {
    打印语句;
}
```

4）函数调用（实际上函数定义的语法差别稍大些）

Python 代码：

```
func( para1, para2 )
```

JAVA、C#、C 代码：

```
func( para1, para2 );
```

可以看出，这些不同语言的基本语法是非常相似的。相比语法，更重要的是理解计算机原理、数据结构和算法。如果你真是零基础，那就选 Python 吧，这语言写起来比较简练，上手相对容易些。

26 站在巨人的肩膀上

在《零基础学编程 021：获取股票实时行情数据》这一节里，我们利用 urllib 抓取新浪财经中的股票数据，可以取出谷歌股票的开盘价，回顾一下代码：

```
import urllib.request as req

with req.urlopen('http://hq.sinajs.cn/list=gb_goog') as f :
    hq = f.read().decode('GBK')
    v = hq.split(',')
    print(v[1])
```

但我们很多时候并不需要也不应该从零开始构建一个程序，大量的程序员已经构建了丰富的而且免费的模块供我们使用。编程领域中流行着一句非常有名的话，叫做“**不要重复发明轮子**(Don't Reinvent the Wheel)”，意思是说不要重新去写别人已经写好的、甚至已经优化过的基本函数。

程序员通常会看不起别人写的代码，所以重复发明了一个又一个的轮子，但他写的代码也好不到哪里去。因为一串代码从表面上看比较简单，但实际动手时会遇到许多复杂的情况，有时需要花费大量的时间才能让程序不出错，并且性能稳定。所以，如果有比较可信的代码库，又不侵犯版权的情况下，尽量还是用别人写好的代码库更能节省时间。

Python 中已经建立了一个庞大的代码库社区，称作 Python Package Index，简称 Pypi，网址：https://pypi.python.org 。世界各地的程序员们已经贡献了无数的优秀的模块，在动手写一个程序时，记得到这个网站上搜一搜有没有可以直接拿来就用的函数，站在巨人们的肩膀上，写起程序来又快又好。

对于我们文章开头提到的功能，在 Pypi 中就有一个现成的提到股票行情的模块库，叫 yahoo-finance，看看该模块库的简单帮助说明，几行代码就可以取出开盘价。这个代码库的优点在于还提供了几十个其它功能，包括取出历史行情数据，这个功能我们将来会用到。

```
from yahoo_finance import Share
openPrice = Share('goog').get_open()
print(openPrice)
```

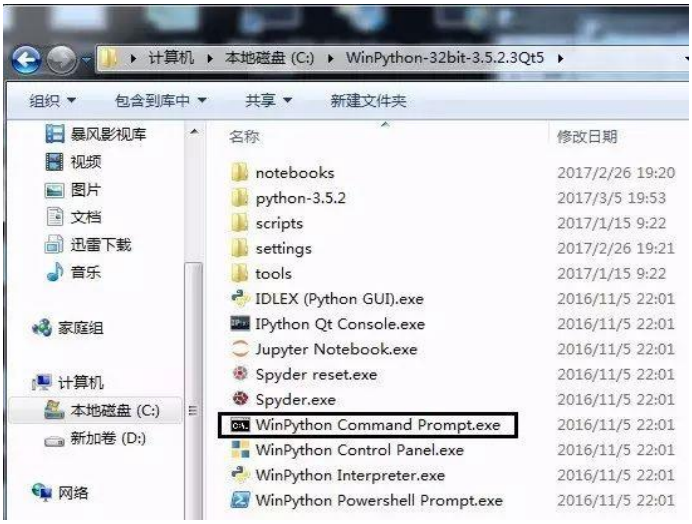
对于新手来说，上面的代码一运行就会报错，是 import 语句执行的时候提示找不到 yahoo_finance 包。所以，最值得一提的是安装这个 yahoo-finance 模块库的过程，将来安装其它任何模块库也如法炮制即可。

如今的 Python 已经极大地简化了模块安装的过程，它就是 **pip**，在 python2.7.9 和 python 3.4 之后的安装版本中都已经内置了 pip，只要有网络，安装任何模块库，一般只需要一个命令就可以搞定。

pip 是“Pip Installs Packages”的缩写，是一个专门用于管理 Python 软件包的程序，运行这条命令：

```
python -m pip install yahoo-finance
```

新手不知道在哪里运行上面那条命令，如果环境不正确（PATH 环境变量问题），在 cmd.exe 黑窗口中运行也会出错。如果你安装了 WinPython，则运行那个 WinPython Command Prompt.exe，在这个黑窗口中执行 pip 命令即可。安装过程是全自动的，成功后会给出相应的提示。



小结：

- 不要重复发明轮子
- pypi 中资源非常丰富
- pip 用于安装其它模块库

27 面向对象编程 OOP

在《零基础学编程 021：获取股票实时行情数据》一节中，我们想获取 6 支股票的行情数据，在《零基础学编程 022：函数的世界》里我们能够把重复性的代码封装为一个函数 price()，以后获得不同的股票行情只需**调用函数**即可，回顾一下这个函数的代码：

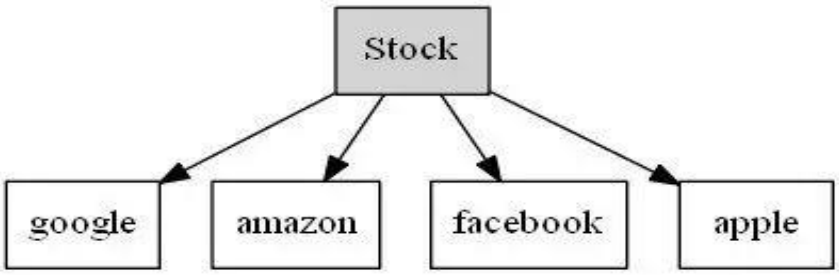
```
def price(stockCode) :
    url = 'http://hq.sinajs.cn/list=' + stockCode
    with req.urlopen(url) as f :
```

```
        hq = f.read().decode('GBK')
        v = hq.split(',')
        return v[1]
```

以后，我们还会写许多函数，比如获取股票的名称、取收盘价，取某日的行情等，当函数写得越来越多时，代码就会变得难以维护，这时需要用一种办法把同类的东西封装在一起，就要用到**面向对象编程(OOP: Object Oriented Programming)**的概念。

现代化的编程语言几乎都支持面向对象的概念，所以要尽快掌握面向对象编程的思路，程序员的世界里几乎全是**对象**和**函数**。

以股票程序为例，GAFATA 中的 google 是股票，amazon 是股票，facebook 是股票，apple 是股票，在计算机的世界里就把这些单一的个体称为**对象(object)**，把这些对象的共性的抽象为**类(class)**，在这个例子里，google、amazon、facebook，apple 等都是一个一个的**对象**，而股票 Stock 就是**类**。



先来看股票 Stock 类的代码：

```
import urllib.request as req

class Stock :
    code = ""

    def __init__(self, code1) :
        self.code = code1

    def getPrice(self) :
        url = 'http://hq.sinajs.cn/list=' + self.code
        with req.urlopen(url) as f :
            hq = f.read().decode('GBK')
            v = hq.split(',')
            return v[1]
```

解释一下：

1）class Stock :

这一行语句，表示声明一个类，类的名称是 Stock，后面一定要有冒号，以后的代码行要缩进，因为它们要描述类中的**成员变量**和**成员函数**。

2）code = ""

股票代码不同，则代表不同的股票对象，这里的 code 用来表示股票代码，例如：gb_goog，gb_amzn 等。这个 code 称为**成员变量**。

3）def __init__(self, code1) 函数

这个函数的写法有点特别，称为**构造函数**，用于完成一些初始化的任务，这里是把一个股票代码记录在 self.code 中。**self** 是一个 python 中一个非常重要的关键字，表示对象本身，你要使用类中的变量时，就要用 self 后面加一个小点。

4）def getPrice(self) 函数

这个函数称为**成员函数**，这里把以前的 price 函数修改了一下，主要的变化就在于 self.code。

上面的类的定义就完成了，类名叫 Stock，有一个**成员变量** code，有一个**成员函数** getPrice()，在面向对象的概念里，这些**函数**也称为**方法**，英文是 function 或 method。还有一个名称比较奇怪的函数，前后都有两个下划线，表示这个函数一般不给外界使用，只是给系统调用的。

这样就完成了一个股票类的封装，我们使用它时，只需构造一个对象，并调用相应的方法即可。

```
google = Stock("gb_goog") # 构造一个对象，谷歌股票，系统会自动调用__init__函数
print(google.getPrice()) # 取开盘价

amazon = Stock("gb_amzn") # 构造另一个对象，亚马逊
print(amazon.getPrice()) # 取开盘价
```

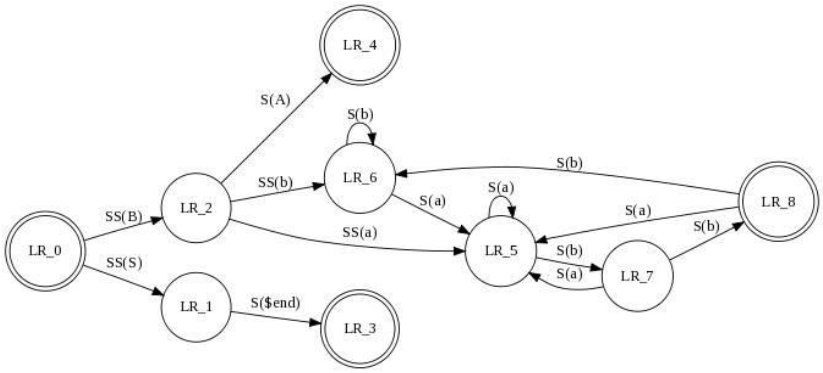
小结：

- 现代语言都支持**面向对象编程** OOP，需尽早掌握
- 相同的对象 **object** 抽象为类 **class**
- class 关键字用于声明一个类
- 成员变量用 **self** 访问
- __init__是构造函数，完成一些初始化的任务

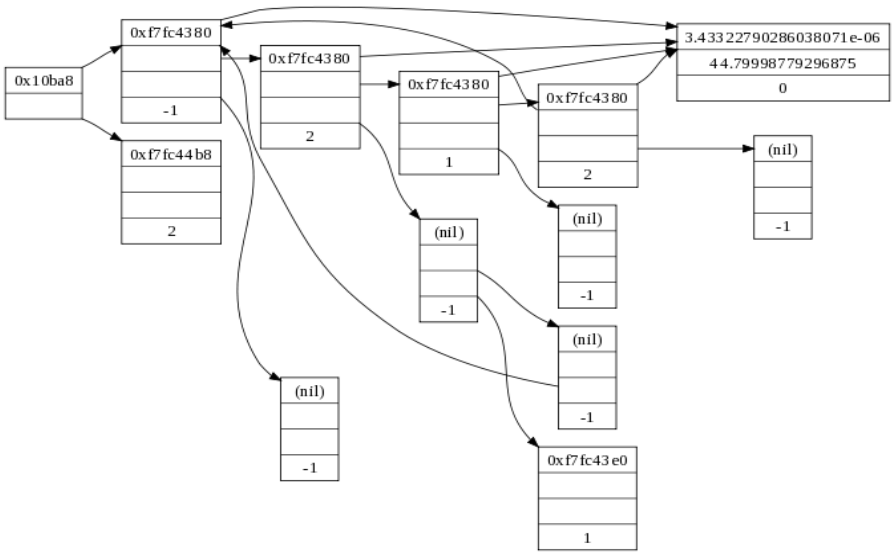
- 面向对象里的**函数**，也称为**方法**，即 function
- Stock()就可以构造一个对象
- 小圆点用来访问**成员变量 code** 或**对象的方法 getPrice**

28 程序员作图不用笔

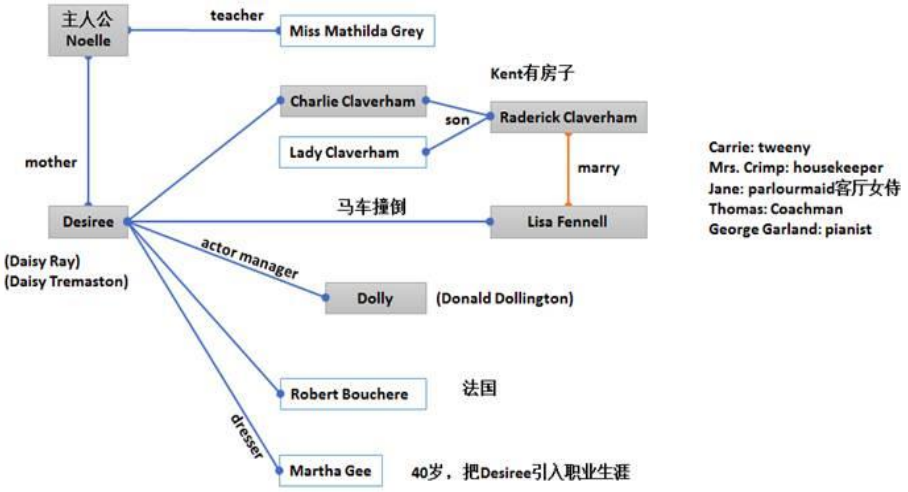
现在写专业文章离不开图，有些图非常复杂但非常有规律，用 PowerPoint 或 Visio 画都很吃力，这时候会编程就轻松多了，比如下面这张状态转换图：



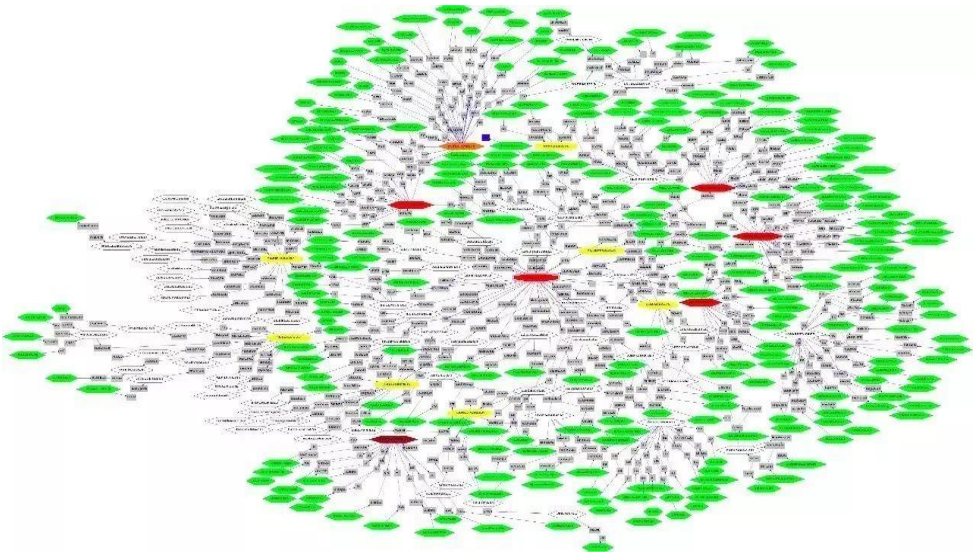
再比如这张数据结构图：



再比如英文小说《欺骗的女儿》中的人物关系图：



再比如这张超复杂的网络结点图：



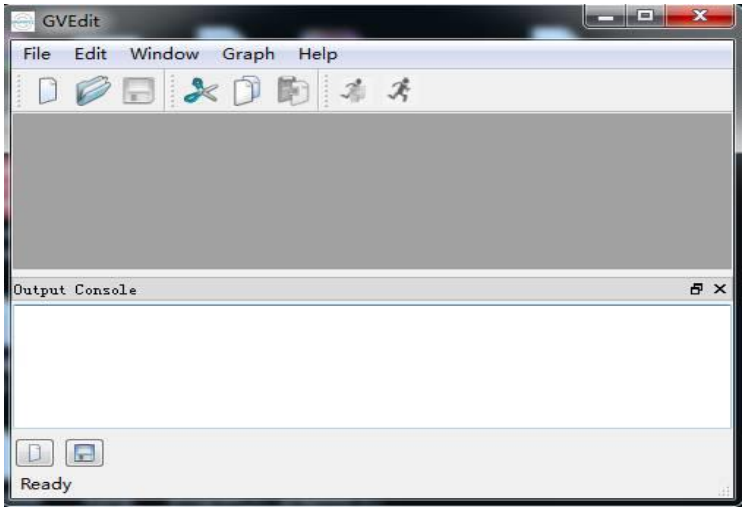
有些图看起来简单，可能用 PowerPoint 画也费不了多少时间，但如果这种图需要频繁调整，那工作量可就大了去了。比如程序员经常画的流程图、类图、数据结构图等，公司里经常画的组织结构图、工作流图等。

对于这类非常有规律的图，还有一个强大的工具，它就是 GraphViz。上面举的几个例图都是摘自它的官网：http://www.graphviz.org。这个 GraphViz 不仅仅是一个工具，而且还对应了一种画图的语言，称为 **DOT Language**。

安装与运行

到官网上点击 Download 链接，可以看到各种平台的下载安装包，支持 Linux、Windows、Mac，我下载的是 Windows 平台的 graphviz-2.38.msi 安装包，安

装过程一路默认下一步即可。完成之后，从开始菜单中找到 gvedit.exe，运行它出现主界面。



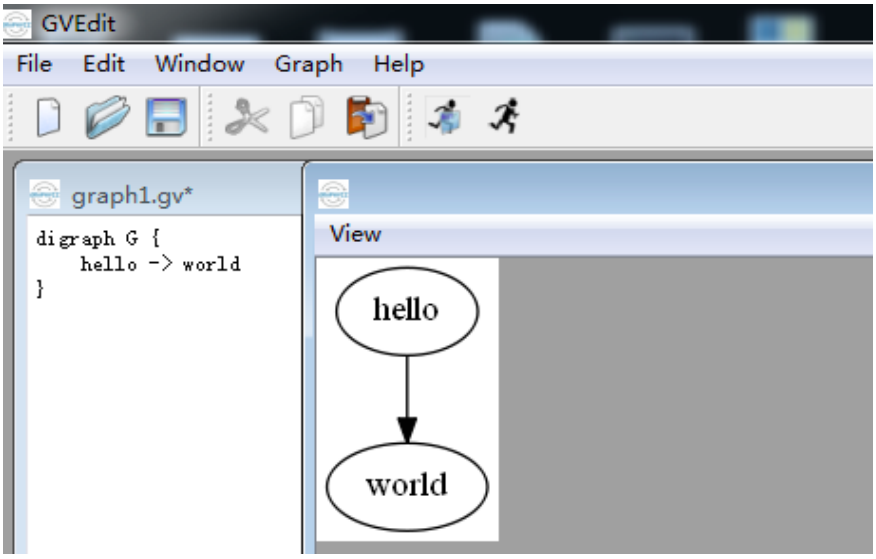
Hello World

任何语言都有个 Hello World，DOT Language 也不例外。从 File 菜单中点击 New，会新建一个子窗口，名称为 graph1.gv，所有 GraphViz 的文件的扩展名都为.gv，刚才的 gvEdit.exe 的意思也明白了吧？

在文本编辑窗口中输入以下代码：

```
digraph G {
    hello -> world
}
```

再点击 Graph 菜单中的 Layout，或者直接按 F5 键，弹出一个 View 窗口，一张简单的图就画出来了。

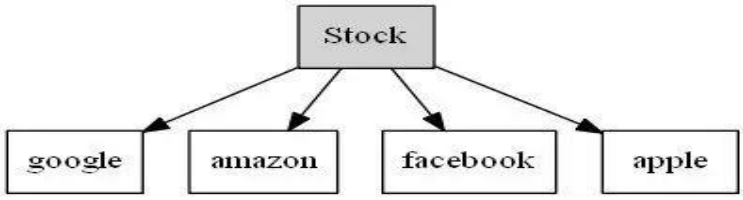


简单解释一下：

- digraph 表示**有向图**，是 Directed Graph 的缩写形式，什么是有向图？请参考《图论》
- G 是图的名称
- 花括号{}内是图形的描述语句
- hello 和 world 是两个**节点 node**
- -> 表示左边指向右边的一个**边 edge**

类与对象图

在《零基础学编程 028：面向对象编程 OOP》里我画了一张图，实际上就是用 GraphViz 生成的。



图并不复杂，直接看代码：

```
digraph G {
    node[shape=box]
    Stock[style=filled]
```

```
    Stock -> {google; amazon; facebook; apple}
}
```

花括号内的代码就三行，记得按 F5 看看运行效果，解释一下：

- node[...]表示对图中的所有结点统一进行设置
- [ ] 中设置一些属性，称为 attr
- shape=box 把结点设置为矩形
- Stock 是结点的名称，由于前面已经设置了 shape=box，所以也为矩形
- stype=filled，填充的矩形
- Stock -> {google; amazon; facebook; apple}相当于以下四句
  Stock -> google
  Stock -> amazon
  Stock -> facebook
  Stock -> apple

小结：

- GraphViz 的语法挺简单，里面主要是**结点 node** 和**边 edge**。
- -> 表示一条有向边
- 最复杂的是 Attr，里面可以设置填充、排列、颜色、链接等等，详细内容以后再说，也可以参考官网的 Documentation 链接，长达 N 页的全英文详细说明，慢慢看吧

29 像黑客般玩玩字符艺术

经常安装盗版软件的朋友可能会看到一些由字母组成的奇怪图案，比如下面这张图：

在 Python 里，我们无法用字母拼出如此复杂的图案，但也可以搞点简单的**字符艺术**，这就要用到 pyfiglet 模块库了。看看 FIGlet 官网（www.figlet.org）上的简介，figlet 就是用一些普通符号拼出一个比较大的字符图案，比如这样：

安装 pyfiglet

figlet 本身支持多种平台，有人用 python 实现了 figlet，就是 pyfiglet 模块。在《零基础学编程 027：站在巨人的肩膀上》中我们已经会使用 pip 安装第三方模块库，这里复习一下安装命令。

```
python -m pip install pyfiglet
```

源代码

程序很简单，只需三行：

```
from pyfiglet import Figlet
fig = Figlet(font='standard')
print(fig.renderText('Shen Longbin'))
```

第一行是 import 语句，不多解释，用到了类 Figlet。

第二行构造一个对象 fig，指定一种字体。

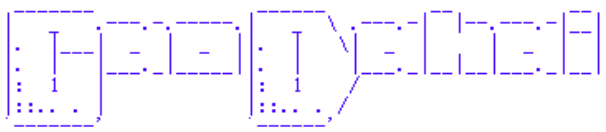
第三行 fig.renderText()可以产生出字符串拼出的图案。

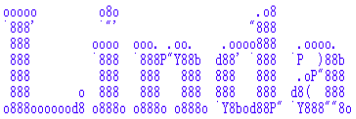
换换字体和文字

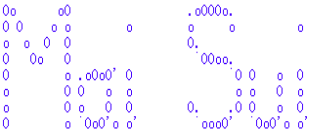
实际上 pyfiglet 的主要用法用这么多，还有一些选项并不常用，你只需要换换不同的字体（http://www.figlet.org/fontdb.cgi）就可以产生不同的效果了。官方的 figlet 也支持点阵的宋体和仿宋汉字，但 pyfiglet 好像并不支持汉字。

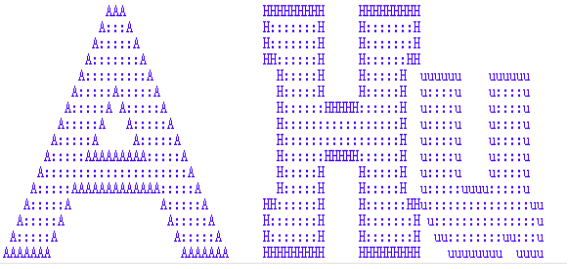
我经过试验，比较喜欢这几种字体：'letters', 'banner', 'doh', 'cricket', 'slant', 'univers', 'starwars', 'rounded', 'roman', 'puffy', 'pebbles', 'larry3d', 'epic'。喜欢玩的就自己去试吧。

```
 SSSSS                    SSSSS  iii
SS         aa aa nn nnn   SS
 SSSSS   aa aaa nnn  nn    SSSSS  iii
     SS aa  aaa nn   nn        SS iii
 SSSSS   aaa aa nn   nn    SSSSS  iii
```









30 Python 与其它语言最不同的一条语法规则

有 C 或 JAVA 其它编程语言基础的人可能对 Python 中的这条语法规则最不适应：Python 中的**缩进**是有语法含义的，它用来表示一个**代码块**(code block)。这里说的代码块是指函数定义、条件语句、循环语句等等。缩进就是指每行代码最前面的几个空格或 TAB 制表符，通常是与上一行的**冒号**一起使用的，例如：

```
# 为了清楚地表示缩进，我把空格都用.表示
def price(stock) :
....url = 'http://hq.sinajs.cn/list=' + stock
....with req.urlopen(url) as f :
........hq = f.read().decode('GBK')
........v = hq.split(',')
........return v[1]
```

假设用 C#语言来写，大概是这样：

```
// 这段代码只是一个示例，无法通过编译
double price(string stock) {
    string url = 'http://hq.sinajs.cn/list=' + stock;
    using(WebRequest f = new WebRequest(url)) {
        string hq = f.read().decode('GBK');
        string []v = hq.split(',');
        return v[1];
    }
}
```

大部分编程语言都是通过花括号"{"、"}"这类符号来标记块的开始和结束，花括号内部的代码并不需要缩进，缩进只是为了让程序员更容易读，更容易看懂代码的逻辑结构。在 C#里，刚才的代码也可以这样写：

```
// 这段代码只是一个示例，没有缩进的代码难以阅读
double price(string stock) {
string url = 'http://hq.sinajs.cn/list=' + stock;
using(WebRequest f = new WebRequest(url)) {
string hq = f.read().decode('GBK');
string []v = hq.split(','); return v[1];
}  }
```

Indent 缩进、Deindent 反缩进

而 Python 把这些花括号也给省了，**缩进**是一条强制性的语法规则，如果缩进不正确，则代码会报错！

来看看《18 零基础学编程 018：条件语句》中的那段代码，如果写成这样：

```
from turtle import *
for i in range(365) :
    forward(1.01 ** i)
    left(9)
    if(i%7==0) :
    stamp()
```

则会提示：

```
There is an error (expected an indented block) at line 6, column 9.
```

indent 就是向右缩进的意思，而 **deindent**(或 **dedent**)是向左缩进的意思，向左缩进是我编的名字，我也不知道 deindent 的正规中文翻译是什么。

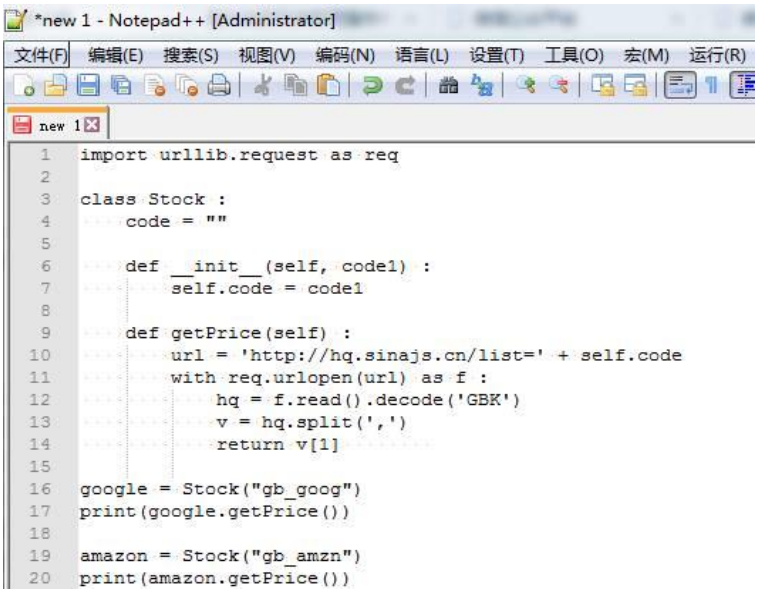
在 IDLE 集成环境中，还有专门的快捷键 ctrl +] 用于缩进，而 ctrl + [则是相反的操作。

不要使用制表符 TAB？

按 TAB 键也可以产生缩进的效果，这一点在 Word 排版时也会用到，但在 Python 中要小心了，不同的文本编辑器对 TAB 的解释可能不一致，有些默认是 8 个字符，有些是 4 个字符，有些则是对齐到 8 的倍数列上，如果你的代码中混用 TAB 和空格，则会出现一些奇怪的错误。

现在的许多编辑器中都增加了一项设置，可以自动将 TAB 保存为多个空格，比如 IDLE、Notepad++(注意不是 Windows 中自带的记事本，多个两个加号，功能强大多了)，当你按 TAB 键时，自动产生的是 4 个空格。

在这种编辑器中写代码就方便多了，你按一次 TAB 可以，按四次空格也行，效果一样。像 Notepad++中还提供了**细细的对齐线**等功能，还可以突出显示空格和制表符，让你看得更清楚。



Python 编码规范

Python 语言本身对于缩进的空格数没有规定，但为了让程序员们互相之间容易沟通，Python 社区对代码的规范性提出了许多建议，如果你遵守这些建议，则与全世界的程序员们基本上保持了一致的习惯。网址：https://www.python.org/dev/peps/pep-0008/

关于缩进这一部分，规范中规定：

缩进用 4 个空格

一条比较长的语句也可以用缩进分成多行，详细的规定请阅读英文原文

Python 3 中已经禁止 TAB 和空格混用

悬挂 else 问题

在其它编程语言中，有可能会遇到这类的悬挂 else 问题，即末尾的 else 语句与 2 个 if 语句中哪一个配对？有经验的程序员会要求所有的语句都要有花括号。

在 Python 中的这样一段代码：

```
if (条件 1) :
    if (条件 2) :
        fun1();
else :
    fun2();
```

在 C#或 JAVA 中会写成这样：

```
if (条件 1) {
    if (条件 2) {
        fun1();
    }
}
else {
    fun2();
}
```

Python 的缩进规则让代码显得更简洁，根本不会出现悬挂 else 的问题，你的缩进表明了你的 else 与哪个 if 相匹配。

当然 Python 中的缩进也有一个缺点，如果你从网页上复制 python 代码，有些网页上的代码排版本来就很乱，如果原始的 python 代码缩进乱了，则无法重新格式化。

小结：

- 选用支持将 TAB 自动转换为 4 个空格的编辑器或集成开发环境
- 保持 TAB 为 4 个空格的默认编辑器设置
- 从其它地方粘贴的代码如果出现错误，可以用对齐线辅助检查
- 遵守 Python 的编码规范，方便与全世界的其他程序员沟通

31 生成二维码

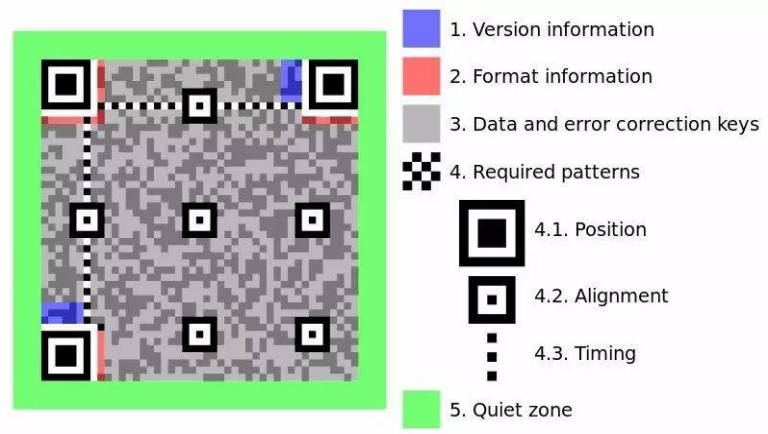
现在生成二维码的工具遍地都是，既然手里已有强大的 python，那么这种小事也不需求人了，只需三行代码搞定：

```
import qrcode
img = qrcode.make("http://www.cnblogs.com/speeding")
img.save("slb-blog-qrcode.png")
```

二维码

二维码的标准英文全称是 Quick Response Code，简称 QR code，直译“快速响应矩阵码”估计没人能听懂，这种东西是在以前的条行码的基础上发展起来的，由于二维码存储的信息量可以更大，并且手机摄像头的普及和更多 APP 的支持，二维码才变得更加流行。

详细的技术原理请查阅维基百科中的 qrcode 词条，明白了技术原理，把你的二维码做得像朵花一样也不是不可能。



安装

二维码应用分为**生成器**和**识别器**两类，Python 中也有生成二维码的模块包，名称就叫 qrcode，在《零基础学编程 027：站在巨人的肩膀上》里已经学会了安装各种模块包，再复习一遍。

```
python -m pip install qrcode
```

运行

运行文章开头的三行代码，会生成一个 PNG 格式的图片。编程小白在运行这段代码后不知道到哪里找到那张图片。如果 python 的设置没有改过，你只需在 python 的安装目录的 notebooks 子文件夹下就能找到那张图片。还可以利用《方便得令人发指的 everything 软件》，根据文件名快速找到你的文件。

qrcode.make()函数中的参数是一个字符串，二维码规范本身并不对存储的内容进行限制，但通常大家都用网站的 URL 链接，这样微信扫一扫之后，直接就跳转到相关网页，给用户带来极大的便利，比如本例子中存的就是我的博客网址。

更多

qrcode 中还可以进行更精细的设置，比如指定图片的大小、精度、版本、图片格式等，我就不重复了，详见官网链接：https://pypi.python.org/pypi/qrcode/#downloads

有能力的人，还可以试着在二维码的中心加上自己喜爱的 LOGO。

手机上此类 APP 很多，试着搜“qrcode"。我在手机上安装了一个 workflow 的软件，可以快速把剪贴板中的链接地址生成二维码，自动保存在相册中，这样发朋友圈或写文章就方便多了。

小结：

- 二维码就是 qrcode
- python 中有个模块库就叫 qrcode
- 三行代码就可以生成一张二维码图片

32 字符串的 split 拆分与 join 连接

在《零基础学编程 021：获取股票实时行情数据》这一节里，我们学了 split()函数，可以将一个字符串切开。假设有一个历史行情字符串，信息包括：股票名称、开盘价、最高价、最低价、收盘价、交易量等，用 split() 之后可以方便地取出任何一个价格，例如：v[1]就是开盘价。

```
hq = "谷歌,843.64,847.24,840.8,845.62,779900"
v = hq.split(',')
print(v)

# 输出结果：['谷歌', '843.64', '847.24', '840.8', '845.62', '779900']
```

可以看到 print(v)的输出内容是一个**列表 list**，python 将其输出时，会在前后加上中括号[]，里面的每一项内容仍是字符串，显示出来不直观。如果你想把这些字符串再重新拼起来，以前有编程基础的朋友马上会这样做：

```
v = ['谷歌', '843.64', '847.24', '840.8', '845.62', '779900']
hq = ""
for i in v :
    hq += str(i) + ","
hq = hq[:-1]  #去掉尾部多余的一个逗号
print(hq)
```

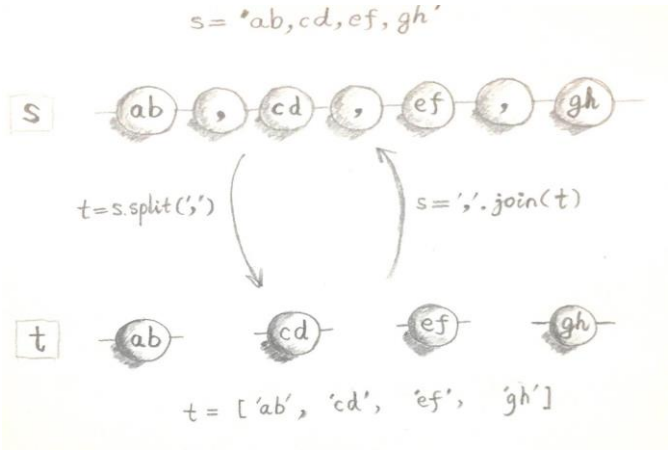
但实际上应该用 join() 函数，简洁、直观、性能好：

```
v = ['谷歌', '843.64', '847.24', '840.8', '845.62', '779900']
hq = ','.join(v)
print(hq)
```

这个 join()函数与 split()函数的功能几乎正好相反，但初学者在理解','.join()这条语句时感觉会很不习惯，似乎 v.join(',')更符合思维习惯。关于这个问题，感兴趣的朋友可以读下面这篇帖子，里面介绍了这种设计的主要考虑和经过：http://stackoverflow.com/questions/493819/python-join-why-is-it-string-joinlist-instead-of-list-joinstring

最近正在零基础学画画，尝试着把这两个函数的意思画了下来。

```
s = 'ab,cd,ef,gh'
t = s.split(',')
s = ','.join(t)
```



这里需要注意的是：s 是一个字符串，而 t 是 4 个字符串。

还需要特别注意这样一种用法：

```
','.join('abcde')
```

join()函数里的参数是一个列表 list（准确地说，应该是一个**可遍历的**对象，这里先不介绍它），python 的字符串也是可遍历的，可以拆为一个个的单个字符，所以结果就是：

```
'a,b,c,d,e'
```

我把以前的 365 行复利数据表再利用 join 函数写一遍，你还能看懂吗？

```
def fuli(i) :
    y = round(1.01**i, 2)
    return "(1+0.01) ^ " + str(i) + " = " + str(y)

fuli365 = [fuli(i) for i in range(1,366)]
print('\n'.join(fuli365))
```

33 解决一个 pandas 问题

昨天一位朋友问了一个程序问题：一个 csv 电子表格文件，里面有不规范数据，如何用 pandas 的 dataframe，将某一列是空值的记录行删掉。

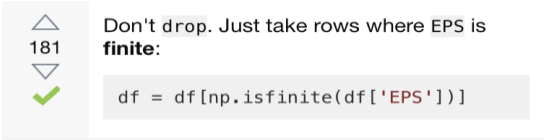
收到了 CSV 文件，如果 RPROC_DMS_ID 没有内容，则该行剔除。

RPROC_PK	RDOC_FK	RPROC_DMS_ID	RPROC_Current_User	RPROC_Cur
22433627	20140102W30001300044	7397672	BPO-MCCAIN\svc-cdc	CDC Servi
22433628	20140102W30001300044	7399087	sathish.genji@mcca	Sathish
22433629	20140102W30001400001	7397686	BPO-MCCAIN\svc-cdc	CDC Servi
22433630	20140102W30001400001	7399788	upasna.chaturvedi@	Upasna Ch
22434017	20130903J10000900006	6524910	BPO-MCCAIN\svc-cdc	CDC Servi
22434018	20130903J10000900006	6533496	suganya.rajendran@	Suganya R
22434019	20130903J10000900006	6623002	rvanwyk@mccain.co.	rachel va
22434020	20130903J10000900006	6631477	suganya.rajendran@	Suganya R
	29:58:00			
22434021	20130903J10000900006	7000722	rvanwyk@mccain.co.	rachel va
22434022	20130903J10000900006	7008886	suganya.rajendran@	Suganya R
22434023	20130903J10000900006	7047424	rvanwyk@mccain.co.	rachel va
22434024	20130903J10000900006	7059804	suganya.rajendran@	Suganya R
22434025	20130903J10000900006	7129366	rvanwyk@mccain.co.	rachel va
22434026	20130903J10000900006	7137833	suganya.rajendran@	Suganya R
	29:58:00			
22434027	20130903J10000900006	7395798	rvanwyk@mccain.co.	rachel va
22434028	20130903J10000900006	7396992	meenakshi.subramar	Meenakshi

该问题的最终答案并不太重要，更关键的是问题的解决思路和过程。我听说过 pandas，但并没有用它写过一行相关代码，但这并不妨碍我解决这个问题。

运用《零基础都需要哪些基础》里提到的搜索技巧，第一种直接的办法是谷歌搜索。我以前学过 R 语言，知道这个 dataframe 的大概功能，这种问题在大数据分析里称为**数据清洗**，需要将不规范的数据（例如空值 null）剔除掉。我马上想到的搜索关键字是 pandas dataframe filter null。

第三条搜索结果的 drop rows 与我的问题描述太吻合了，直接点开这个网页，里面有一行简短的说明和代码。



第一步：安装 pandas

在《站在巨人的肩膀上》里已经学会了安装程序包，重复一次那个过程：

```
python -m pip install pandas
```

第二步：读入 csv 文件

由于我以前没学过 pandas，所以仍是搜索 pandas read csv，发现了这行代码：

```
import pandas
df = pandas.read_csv('data.csv')
```

运行出错，错误信息：

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa8 in position 3:
 invalid start byte
```

看到 utf-8，再根据以前的编程经验，感觉是字符集不正确。翻阅 read_csv() 函数的帮助，发现了 encoding 选项，又因为 csv 文件中并没有汉字，看来也不可能是 GBK 等字符集，先试试 iso-8859-1 吧，竟然直接通过！

```
df = pandas.read_csv('data.csv', encoding='iso-8859-1')
```

第三步：筛选数据

把搜索到的代码直接录入，字段名换换。

```
df2 = df[np.isfinite(df['RPROC_DMS_ID'])]
```

又报错：NameError: name 'np' is not defined

在《零基础学编程 012：画出复利曲线图》里，我见过 numpy 被缩写为 np，看来就是它的问题。

```
import numpy
df2 = df[numpy.isfinite(df['RPROC_DMS_ID'])]
```

运行正常，看看记录数变化了吗？

```
print(len(df), len(df2))
```

看到记录数从 10683 变成了 10000 行，看来好像是完成任务了。检查的办法还需要其它函数，这里不展开介绍了。

小结：

- 学会搜索，多试试不同的关键字
- 以前的 R 语言经验对理解 dataframe 有帮助
- 数据挖掘的知识也有帮助
- utf-8、iso-8859-1、GBK 字符集的知识
- 以前用过 numpy 程序包，解决了 np 出错的问题

- 解决具体的问题不难，但学习 pandas 还得一步一步地来

最终代码：

```
import numpy
import pandas
df = pandas.read_csv('data.csv', encoding='iso-8859-1')
df2 = df[numpy.isfinite(df['RPROC_DMS_ID'])]
print(len(df), len(df2))
```

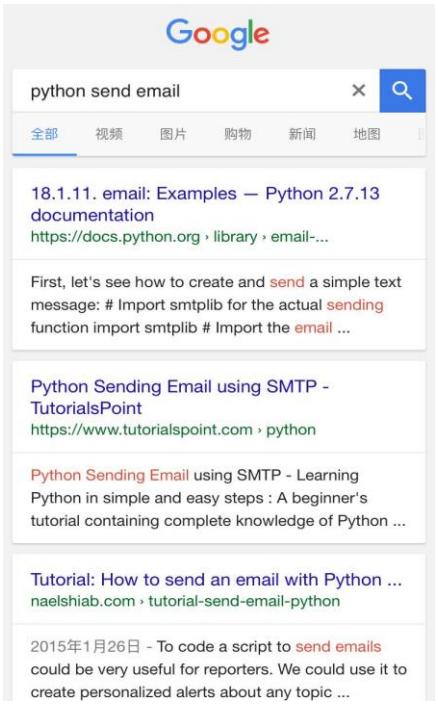
34 群发邮件并不难

我是 GTD 的重度用户，GTD 中讲究将所有事情先收集起来再说，所以收集操作越快越好，这样才不至于把手边的工作打断。很多老牌的 GTD 工具软件支持发邮件实现快速收集，比如 OmniFocus，所以我的 Windows 桌面上放着一个 script 脚本，用于把待办事项一键发送到我的 OmniFocus 服务器，它会自动同步到我的手机上的 OmniFocus，省去了我在手机上繁重的手指录入操作。

永远不要学编程，而是用编程，来和我一起探索如何用 Python 发邮件吧。

第一步：搜索大法

运用《零基础都需要哪些基础》里提到的第一条技巧：谷歌搜索。我以前学过收发邮件的基本原理，第一组关键词就用 python send email，打开 VPN 和谷歌，看看搜索的结果。



第二步：官方文档

搜索的第一条结果就指向 python 的官方类库参考手册，以后如果知道某个模块包 packages 的名字后，这个网站 https://docs.python.org/3 是首选打开的，注意里面的 3 表示 Python 3 版本，换成 2 就是 Python 2 的帮助手册。

打开搜索的第一个网页，第一段样例代码：

```
# Import smtplib for the actual sending function
import smtplib
# Import the email modules we'll need
from email.message import EmailMessage

# Open the plain text file whose name is in textfile for reading.
with open(textfile) as fp:
    # Create a text/plain message
    msg = EmailMessage()
    msg.set_content(fp.read())

# me == the sender's email address
# you == the recipient's email address
msg['Subject'] = 'The contents of %s' % textfile
msg['From'] = me
msg['To'] = you

# Send the message via our own SMTP server.
s = smtplib.SMTP('localhost')
```

```
s.send_message(msg)
s.quit()
```

读懂这段例子代码就需要用到以前学过的知识了，这段例子代码中把一个文本文件中的内容作为邮件正文发出去。我们稍微改一下：

```
import smtplib
from email.message import EmailMessage

msg = EmailMessage()
msg.set_content("这是申龙斌发出的一封测试邮件")

msg['Subject'] = '邮件标题：零基础学编程'
msg['From'] = 'shenlongbin@qq.com'  #请换成你的邮箱
msg['To'] = 'slb-omnifocus@sync.omnigroup.com' #请换成你的邮箱

s = smtplib.SMTP('smtp.qq.com') #请查询你的邮箱服务商的 SMTP 域名
s.send_message(msg)
s.quit()
```

在 Python 中发邮件还是非常简单的，需要用到 2 个模块包，即 email 和 smtplib。上面代码中的 From 和 To 分别对应着发送邮箱和接收邮箱的地址，请换成你自己的邮箱试试，而 SMTP 就需要补充一点邮件收发的基础知识了。

第三步：设置 SMTP

打开维基百科或者百度百科，搜索 SMTP。

SMTP（Simple Mail Transfer Protocol）即**简单邮件传输协议**，它是一组用于由源地址到目的地址传送邮件的规则，由它来控制信件的中转方式。SMTP 协议属于 TCP/IP 协议簇，它帮助每台计算机在发送或中转信件时找到下一个目的地。通过 SMTP 协议所指定的服务器，就可以把 E-mail 寄到收信人的服务器上了，整个过程只要几分钟。SMTP 服务器则是遵循 SMTP 协议的**发送邮件服务器**，用来发送或中转发出的电子邮件。

详细的原理对于初学者来说比较难懂，但要明白一个 SMTP Server 概念，你要配置这个**发送邮件服务器**，通过它你就可以顺利地发出邮件了。你使用谁提供的服务，服务商会给你该服务器的域名，例如：QQ 邮箱的发送服务器就是 smtp.qq.com。

如果公司有独立的邮件服务器，系统管理员肯定会告诉你这个邮件服务器的地址。

好了，上面代码中的 from，to，smtp server 这几个关键信息都填好了，运行代码。

第四步：邮箱需要认证

如果你没有收到错误信息，那么就去打开邮箱查收一下，看看邮件是否已经到达。希望你能够一次运行代码成功，但我的环境并不顺利，出现了如下错误：

```
raise SMTPSenderRefused(code, resp, from_addr)

smtplib.SMTPSenderRefused: (530, b'5.7.1 Client was not authenticated', 'shenlongbin@my-mail-server.com')
```

最早的邮件服务器支持任何人发送邮件，但随着垃圾邮件的泛滥，绝大多数邮件服务器都需要通过用户名和密码的认证后才能发送邮件，上面的出错信息表明我与邮件服务器的连接没有通过认证。

再次利用谷歌搜索，加个关键字 authentication：

第二个搜索结果来自于著名的 stackoverflow 网站，里面的 s.login()让人眼前一亮，马上打开该网页，原来需要在 send_message()之前增加一行代码：

```
s.login('shenlongbin', 'Password')
```

再测试运行一次，顺利通过，我可以用 Python 向我的 OmniFocus 发送邮件了！我用这个功能经常把大段的文字发送到 iPhone 手机上，省去了我在手机上打字的痛苦。

当然上面只是最简单的示例，其它功能还需要额外的代码，比如：

- 发送文件附件 attachments
- 接收邮件：这个与发送邮件是两个不同的服务

小结：

- 学会搜索，多试试不同的关键字
- python 的官方文档是第一手参考资料，记住网址：https://docs.python.org/
- smtp 是简单邮件传输协议的缩写
- 邮件服务器的主机地址是必填项
- from 是发送邮箱的地址
- to 是接收邮箱的地址
- email 和 smtplib 是发送邮件的两个模块包
- stackoverflow 是一个查找编程问题的重要网站，里面回答的质量比国内的 csdn 强百倍
- 为了减少垃圾邮件，发送邮件都需要通过认证

最终代码：

```
import smtplib
from email.message import EmailMessage

msg = EmailMessage()
msg.set_content("这是申龙斌发出的一封测试邮件")

msg['Subject'] = '邮件标题：零基础学编程'
msg['From'] = 'shenlongbin@qq.com'  #请换成你的邮箱
msg['To'] = 'slb-omnifocus@sync.omnigroup.com' #请换成你的邮箱

s = smtplib.SMTP('smtp.qq.com') #请查询你的邮箱服务商的 SMTP 主机域名
```

```
s.login('shenlongbin', 'Password') #请换成你的邮箱用户名和密码
s.send_message(msg)
s.quit()
```

35 快速编写一个 GUI 程序

在《零基础学编程 035：群发邮件并不难》里，我们学会了发邮件，我用于向 shenlongbin@sync.omnigroup.com 发送一封邮件，就可以实现 GTD 的快速收集。

写程序没有 GUI 界面，好像显得不专业。什么是 **GUI**？当然是指**图形用户界面**了，黑客们都用黑窗口 Console+键盘，很少用鼠标，而给普通用户们还得用方便友好的窗口界面。

话说 HTML 浏览器的盛行，让桌面端的应用越来越少，以前编写复杂的用户界面程序的本领，当今则变得无用武之地。Python 中的用户界面程序体系也是相当的复杂，今天来个简单可上手的，它就是 guidata。

安装

如果你安装的是 WinPython，则系统中已经包含了 guidata 模块包。如果没有安装也不要紧，正好可以复习《零基础学编程 027：站在巨人的肩膀上》这一节，一行命令搞定：

```
python -m pip install guidata
```

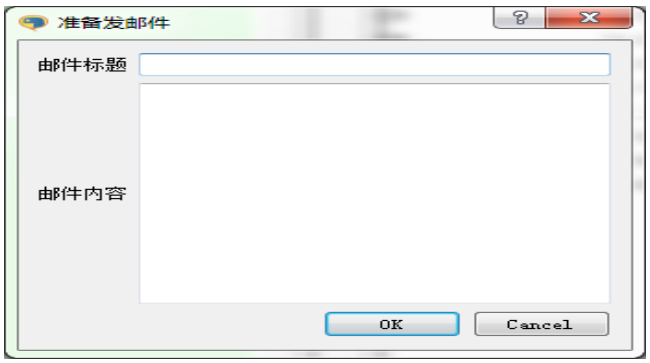
快速上手

先从 guidata 的官方说明手册中抄来一段代码，修改一下：

```
import guidata
_app = guidata.qapplication()

import guidata.dataset.datatypes as dt
import guidata.dataset.dataitems as di

class MailData(dt.DataSet) :
    """ 准备发邮件 """
    subject = di.StringItem("邮件标题")
    content = di.TextItem("邮件内容")

mail = MailData()
mail.edit()
```

```
print(mail.subject)
print(mail.content)
```

运行后，应该能够出现一个 Windows 程序，是不是很神奇？



在单行文本框、多行文本框中随便输入些文字，点击 OK 按钮后，看看 Python 的 IDLE 控制台输出什么？

详细说明

```
_app = guidata.qapplication()
```

这行代码建立一个 GUI 应用的实例，搞不清说的啥？实际内部的细节也不用管了，guidata 背后依赖 Qt，Qt 又是啥？Qt 是一款跨平台的图形界面开发框架。框架又是啥？先不谈框架了，程序员们的术语就是多。图形界面程序背后要完成许多许多的初始化事情，这行代码能把这些事情都准备好。

```
class MailData(dt.DataSet) :
```

在《零基础学编程 028：面向对象编程 OOP》里介绍过**类 class**，这里利用 dt.DataSet 建立了一个**子类** MailData，子类会继承父类的所有特性，很多内部如何实现的细节也不用考虑了。

```
""" 准备发邮件 """
```

这是 Python 中的多行字符串，用来说明类的用途，在**类**定义或**函数**定义之后写上这条语句是个良好的习惯，将来这行语句还可以生成代码的使用说明文档。

```
subject = di.StringItem("邮件标题")
content = di.TextItem("邮件内容")
```

这两行负责获取一个单行的字符串到 subject 中，并把多行的文本保存在 content 变量中，内部细节也不用管。复习一下面向对象编程 OOP 的概念，这两个变量应该叫做**成员变量**。

```
mail = MailData()
```

声明一个对象实例 mail，一个**类 class** 可以产生多个**实例 instance**。

```
mail.edit()
```

此时会出现 Windows GUI 界面窗口，你输入的任何内容，在点击 OK 按钮后，会自动赋值给 subject 和 content 那两个变量中。

最后的程序

合并《零基础学编程 035：群发邮件并不难》上一节的代码，我们就可以做一个带用户界面的向指定邮箱发送邮件的小程序了。

```
import guidata
_app = guidata.qapplication()

import guidata.dataset.datatypes as dt
import guidata.dataset.dataitems as di

class MailData(dt.DataSet) :
    """ 准备发邮件 """
    subject = di.StringItem("邮件标题")
    content = di.TextItem("邮件内容")

mail = MailData()
mail.edit()
import smtplib
from email.message import EmailMessage

msg = EmailMessage()
msg.set_content(mail.content)

msg['Subject'] = mail.subject
msg['From'] = 'shenlongbin@qq.com'  #请换成你的邮箱
msg['To'] = 'slb-omnifocus@sync.omnigroup.com' #请换成你的邮箱

s = smtplib.SMTP('smtp.qq.com') #请查询你的邮箱服务商的 SMTP 主机域名
s.login('shenlongbin', 'Password') #请换成你的邮箱用户名和密码
s.send_message(msg)
s.quit()
```

小结：

- GUI 是图形用户界面

- Qt 是个跨平台的图形用户界面开发框架
- guidata 可以自动生成一个简单的用户界面，收集到一个类的成员变量中
- 搞明白类 class 和实例 instance 的关系
- 三个引号是多行字符串，在类、函数之后写上描述性的文字是个好习惯
- 子类继承父类的所有特性，不用操心内部的实现细节

36 小数据分析

R 语言内置强大的向量运算，是搞数据分析的强大的编程语言，而 Python 也毫不逊色。今天就试着分析一下考试成绩表中两门科目的相关性。

问题描述：

有一个 CSV 文件，包含着 600 名学生在一次考试后的几门课程的考试成绩，想分析一下数学和物理成绩的相关关系。CSV 数据样例：

```
num,class,chinese,math,english,physical,chemical,politics,biology,history,
geo,pe
158,3,99,120,114,70,49.5,50,49,48.5,49.5,60
442,7,107,120,118.5,68.6,43,49,48.5,48.5,49,56
249,4,98,120,116,70,47.5,47,49,47.5,49,60
......
```

各列含义为：学号、班级、语文、数学、英语、物理、化学、政治、生物、历史、地理、体育。

读出 CSV 数据

CSV 是一种逗号分隔的文本文件（Comma-Separated Values），在《零基础学编程 019：生成群文章目录》介绍过如何读取 CSV 文件。这里换一种方法，因为 CSV 本身是一种文本文件，而 numpy 模块包中也可以方便地读入这种数据，请搜索“numpy read csv”可以找出相关文章。

```
import numpy as np
scores = np.genfromtxt('scores.csv', delimiter=',', names=True)
x = scores['math']
y = scores['physical']
```

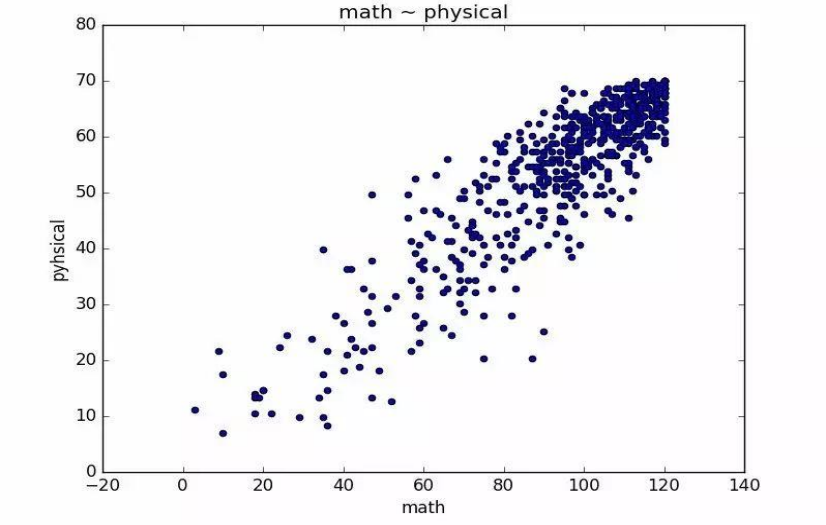
这里需要解释的是第 2 行，delimiter 指明分隔符为逗号；names=True，表示文件中包含标题行，后面就可以使用列名来访问各列数据了。

画散点图

在《零基础学编程 012：画出复利曲线图》这一课里，我们用 matplotlib 画曲线图，同样我们可以用它画出散点图。

```
import matplotlib.pyplot as plt
plt.scatter(x, y)
plt.xlabel('math')
plt.ylabel('pyhsical')
plt.title('math ~ physical')
plt.show()
```

核心代码就一行：plt.scatter(x, y)，后面几行分别设置 x 轴名称、y 轴名称、图名。



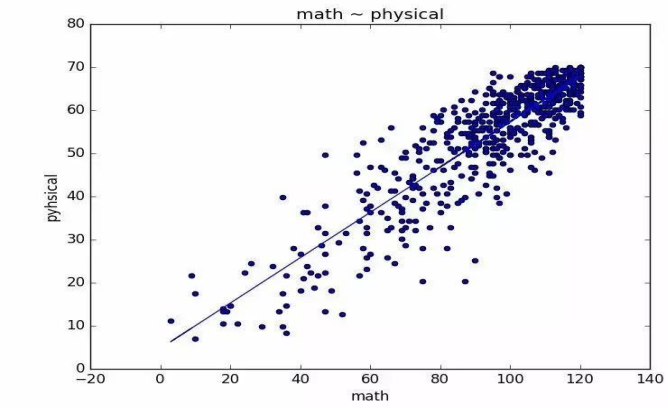
加上线性拟合线

我在谷歌中准备搜索 matplotlib linear 时，它自动弹出了搜索建议“matplotlib linear regression line”

第二条就是我们想要的答案，在 plt.show()之前加入两行代码：

```
a, b = np.polyfit(x, y, 1)
plt.plot(x, a*x+b, '-')
```

polyfit 表示最小二乘法的多项式拟合，1 表示拟合为一阶线性函数。

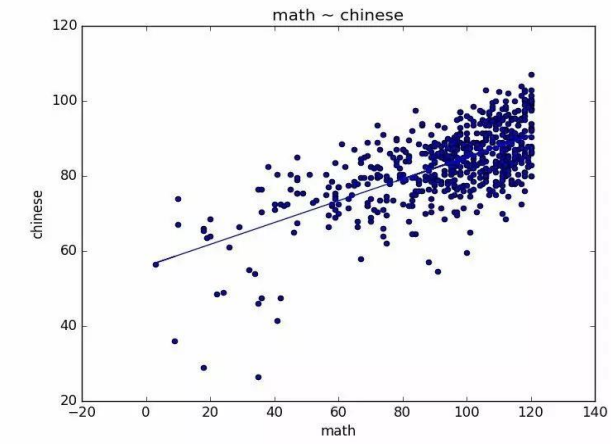


写个函数，画出任意两个科目的散点分布图

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def plot_scatter(scores, subject1, subject2) :
    ''' 画出两门科目的散点分布图和拟合直线 '''
    x = scores[subject1]
    y = scores[subject2]
    plt.scatter(x, y)
    plt.xlabel(subject1)
    plt.ylabel(subject2)
    plt.title(subject1 + ' ~ ' + subject2)
    a, b = np.polyfit(x, y, 1)
    plt.plot(x, a*x+b, '-')
    plt.show()

scores = np.genfromtxt('scores.csv', delimiter=',', names=True)
plot_scatter(scores, 'math', 'chinese')
```

数学和语文的相关情况：



可以看出数理不分家，数学成绩好的一般物理也好，但数学和语文的相关性就不太明显了。

小结：

- CSV 是一种逗号分隔的文本文件
- csv 模块包中的 reader()函数可以读 CSV 文件
- numpy 中的 genfromtxt()函数也可以读 CSV 文件
- matplotlib 中的 scatter()函数可以画散点分布图
- numpy 中的 polyfit()可以计算多项式拟合的系数
- 公众号发消息：**py037**，得 CSV 样例文件的下载链接

37 送你一份编程知识小抄

前几天读完了《世界观》这本书，它把人的世界观类比成各种信念的拼图，感觉人生之旅也是一种拼图，心灵成长的拼图。“零基础学编程”这个系列的文章已经写完 37 篇了，学编程也像是一幅庞大的拼图，需要在学习过程中不断地完善。

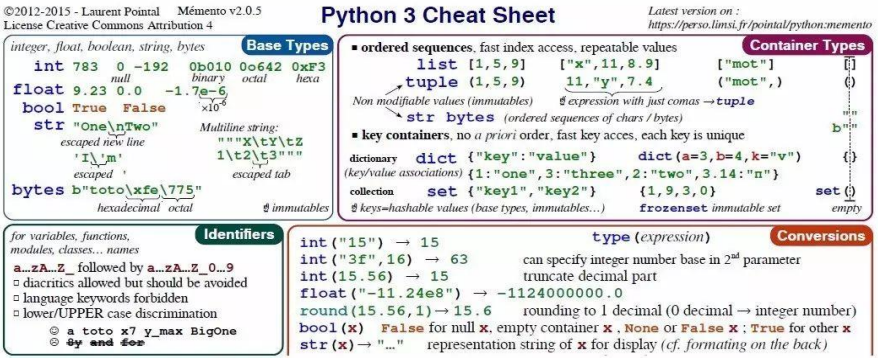
一开始面对一个未知的世界，可能感觉无处下手，但只要起步了，你就在不断地探索这块庞大的知识拼图，核心的学习方法就是英文、搜索、实践、教练反馈、总结等，随着学习的深入，不断地加入程序语法、编程算法、数据结构、数据库、硬件知识等拼图，再在解决实际问题的过程中不断地创建各个拼图之间的连接，才能学会编程。

我建议刚学编程的朋友，要找张大纸把这些知识点记录整理下来，等你积累到 100 张、500 张、1000 张拼图时，也就变成编程高手了。

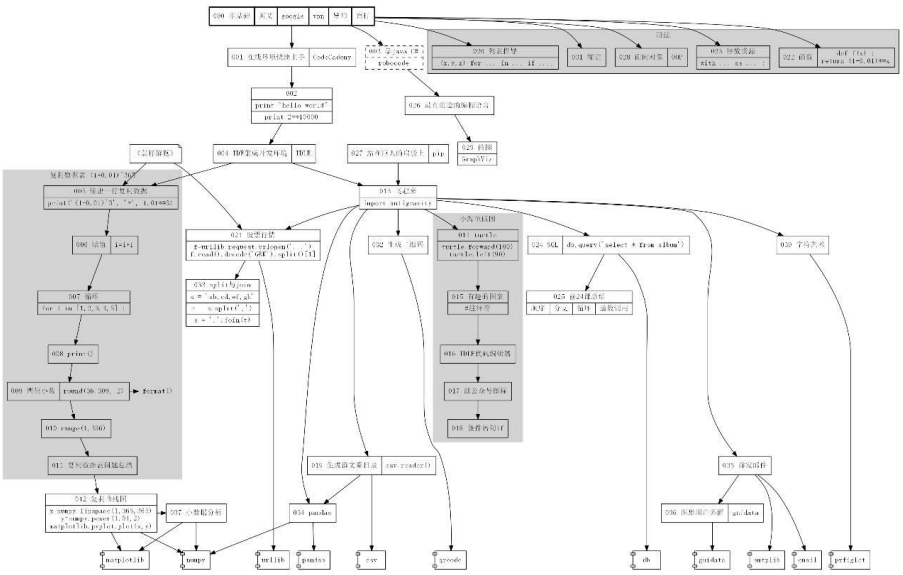
在写这个系列文章时，开始几篇让你快速上手，对编程不再恐惧，建立信心；有几篇文章针对复利数据表问题展开；有几篇讲述了有趣的小海龟做图让你有直观的感受；有些是发邮件、生成二维码、数据分析等实用的小程序；中间则穿插着讲解略显枯燥的语法知识点。

同一个编程问题，解决的手段很多，随着掌握的知识点、模块库越来越多，会出现更方便、更有效率的解决方案，但所有这些知识点都是连接在一起的，互相之间充满了连接。在《如何高效学习》这本书中也提到了创建这些连接的重要性，所以在学习的过程中自己亲手整理一张知识地图是别人无法代劳的。

因为整理这种知识要点对于任何编程语言都非常重要，所以互联网上有一个专门的网站（http://cheat-sheets.org）存放这种笔记或知识图。国外把这种东西称为 Cheat Sheet，你把它用于考试现场就叫做“小抄”，而我们用它主要是强化记忆、建立各个知识的连接。



为了大家方便，我先把前面 37 篇文章的知识点整理出了一张图片，当然我生成这张图也用到了《程序员作图不用笔》的技巧，程序员圈里流行着这样一句话“**吃自己的狗粮**”，自己写的程序自己先要用起来。



微信公众号中不方便嵌入大图片，公众号后台发消息：**cheatsheet** 或 **小抄**，得到这张高分辨率的图片，另外再送出我从互联网上辛苦收集到有关 Python 的 9 份 PDF 格式的 Cheatsheet，一并拿去不谢。

名称	修改日期	类型	大小
beginners_python_cheat_sheet_pcc_all.pdf	2017/3/14 9:50	Adobe Acrobat ...	1,745 KB
mementopython3-english.pdf	2017/3/14 8:36	Adobe Acrobat ...	244 KB
NumPy_SciPy_Pandas_Quandl_Cheat_Sheet.pdf	2017/3/14 8:34	Adobe Acrobat ...	135 KB
PQRC-2.4-A4-latest.pdf	2017/3/14 8:33	Adobe Acrobat ...	600 KB
python.pdf	2017/3/14 8:30	Adobe Acrobat ...	53 KB
python_cheat_sheet.pdf	2017/3/14 8:34	Adobe Acrobat ...	59 KB
Python_qr.pdf	2017/3/14 8:37	Adobe Acrobat ...	145 KB
python_refcard.pdf	2017/3/14 8:34	Adobe Acrobat ...	170 KB
python-cheat-sheet-v1.pdf	2017/3/14 8:33	Adobe Acrobat ...	389 KB
零基础学编程知识拼图v1.png	2017/3/30 10:37	PNG 图像	752 KB

再强调一句，别人的小抄只是参考，自己亲手整理的才更有效。

38 生成群文章目录(2)

每个月的月底，“分享与成长群”要汇总所有成员的原创文章，这次我改用了水滴微信平台把数据采集到一个电子表格文件中。在《零基础学编程 019：生成群文

章目录》这一节里，我已经可以用读 csv 文本文件的办法，配合 markdown 语法自动生成所有文章的目录。

但这次情况发生了几点变化：

- 直接读取 xlsx 的电子表格会更方便
- 有些人想用笔名来署名，不显示真实姓名
- 有些文章暂时不方便对外公开，不显示超链接
- 有些人会多次提交，以最后一次的文章为准。比如下图中的第 120、127 行是同一人的，只保留第 127 行

序号	姓名	笔名	文章标题	文章超链接	是否公开文章的链接？
129	吴强		越简单越快乐	u.com/p/163f008c	不公开
128	王斌		关于信息系统高项考试总结	u.com/p/3e21d57	公开
127	葛勇		微习惯的力量	u.com/p/b32d2bd	公开
126	于迪彬		试错成本<犹豫成本	q.com/s/3qFBH9zI	公开
125	[illegible]	鱼事我想	选错了跑道，你还指望能赢？	qq.com/s?__biz=	公开
123	[illegible]	江湖中人	你要开始，而不是准备开始	u.com/p/9400e32	公开
122	于帅		阅读整理	u.com/p/cf961c19	公开
121	朱海		3月“洗脑”	u.com/p/41f72b3c	公开
120	葛勇		微习惯的力量	u.com/writer#/not	公开
119	郑少榕		离线数据模拟实时数据	et/Nanphonfy/arti	公开
118	黎嘉文		如何鉴别突破	q.com/s/ZHkq4W1	公开
117	王婧	王婧	读后感	u.com/p/d4e04b5	公开
116	[illegible]	如水心境	浅谈-复盘	u.com/p/11f7e26b	公开

这次程序想直接读取电子表格，省掉转换 csv 这一步，查了一下相关资料，python 中读 xls 或 xlsx 的模块库非常多，主要可选的是 xlrd 和 pyexcel 等，最后我选定了 pandas，因为 pandas 也是依赖 xlrd 来读取电子表格，并且将来还可以做更为强大的数据分析，学 pandas 绝对用得上。

读电子表格很方便，用 read_excel()函数。

```
import pandas as pd
df = pd.read_excel("d:/分享与成长群/201703.xlsx")
```

xlsx 原始文件中文章是按提交日期反序排列的，我想让先提交的文章排在前面，因此需要将数据集按“序号”从小到大排序。

```
df = df.sort("序号")
```

删除重复数据，我使用了谷歌，找到了 drop_duplicates()函数，一行代码搞定。意思是：如果“姓名”这一列相同，表示是重复记录，keep='last'表示只保留最后出现一条记录。

```
df = df.drop_duplicates('姓名', keep='last')
```

这个 pandas 采用了与 R 语言类似的 DataFrame 设计，功能非常强大，可以根据设定的条件快速地选出所需的行和列。因为我已经学过 R 语言，看了一下 pandas 的快速入门，就找到了这条语句：

```
df = df.loc[:, ["姓名", "文章标题", "文章超链接", "是否公开文章的链接？", "笔名"]]
```

原表格中还包括 openid、填写时间、IP 地址、备注等列，对于我的文章汇总没有用处，而真正有用的就是"姓名"、"文章标题"、"文章超链接"、"是否公开文章的链接？"、"笔名"这五列。

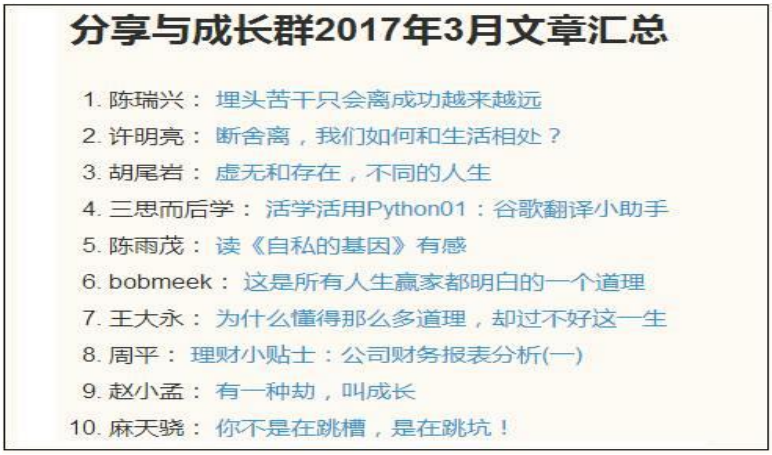
再下来就是逐行循环处理了，pandas 应该有更理想的处理办法，但我现在还没学到。

```
str = ""
for line in df.values :
    name = line[0]
    title = line[1]
    url = line[2]
    public = line[3]
    penName = line[4]

    if(pd.notnull(penName)) :
        name = penName

    str += "1. " + name + ":  "
    if(public == '不公开') :
        str += title + "\n";
    else :
        str += "[" + title + "](" + url + ")\n"
```

生成的文本是 Markdown 格式，还可以更懒一些，把生成的文本直接复制到剪贴板中，从 stackoverflow 上抄来代码：

```
from tkinter import Tk
r = Tk()
r.withdraw()
r.clipboard_clear()
r.clipboard_append(str)
r.destroy()
```

现在只需要到简书上粘贴，并发布就 OK 了。



小结：

- 软件需求永远在变，程序也要不断迭代
- pandas 的 read_excel()可直接读取 xls 和 xlsx 的电子表格
- DataFrame 很强大，可以选行或选列，用.loc[ ]
- sort()排序
- drop_duplicates()去掉重复的行

39 欧拉公式的几何意义

欧拉公式号称是最美的出自上帝之手的数学公式，即 $e^{i\pi}+1=0$，这个公式里 ***e*** 和 ***π*** 都是无理数，***i*** 是 -1 的平方根，是一个虚数，0 和 1 是最简单的整数，欧拉公式把它们联系在一起。

下面我们来理解一下它的几何含义，并用 Python 中的小海龟把它画出来。

利用级数展开的公式可以有下面的推导过程：

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \cdots$$

令： $x = i\pi$

$$e^{i\pi} = 1 + i\pi + \frac{(i\pi)^2}{2!} + \frac{(i\pi)^3}{3!} + \frac{(i\pi)^4}{4!} + \frac{(i\pi)^5}{5!} + \frac{(i\pi)^6}{6!} + \cdots$$

$$= 1 + i\pi + \frac{i^2\pi^2}{2!} + \frac{i^3\pi^3}{3!} + \frac{i^4\pi^4}{4!} + \frac{i^5\pi^5}{5!} + \frac{i^6\pi^6}{6!} + \cdots$$

因为这个数列中含有虚数 *i*，所以可以把上面的每一项看作是**复平面**上的一个向量。

第 0 项：1，表示从(0, 0)点出发沿 x 轴前进 1 个单位。

第 1 项：***iπ***，把其中的 ***i*** 理解为逆时针旋转 90 度，这样就是在垂直方向上前进 ***π*** 个单位。

第 2 项，再旋转 90 度，前进 (***π*******π*** / 2) 个单位。

……

最后这个无数级数的和为(-1)，表示最后逼近(-1, 0)这个点。神秘的欧拉公式的几何含义就是这么简单！

下面用小海龟画出欧拉公式的几何含义。

先复习一下《零基础学编程 014：小海龟做画》这一课中的画图基本命令：

```
import turtle
turtle.forward(100) # 前进 100 个单位
turtle.left(90) # 左转 90 度
```

第 0 项：

```
turtle.forward(1)
```

第 1 项，这里用到了数学包 math 中的 pi，你不用写 3.1415926 了：

```
turtle.left(90)
turtle.forward(math.pi)
```

第 2 项，还记得运算符 ** 是什么意思吧？

```
turtle.left(90)
turtle.forward((math.pi ** 2) / 2)
```

第 3 项，分母是 3 的阶乘，需要用到数学函数 math.factorial()：

```
turtle.left(90)
turtle.forward((math.pi ** 3) / math.factorial(3))
```

现在我们已经找到规律了，假设我们循环 16 次，就是 range(1,17)，每一层的循环只需要执行这两步就可以了：

```
for i in range(1,17) :
    turtle.left(90)
    turtle.forward((math.pi ** i) / math.factorial(i))
```

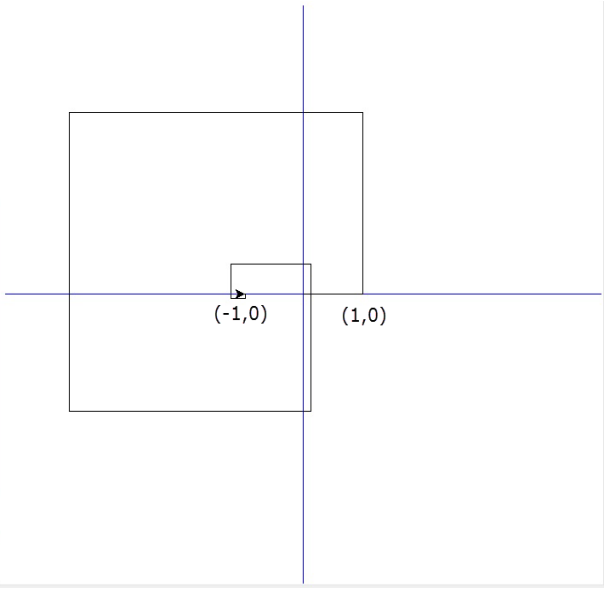
这里小海龟的默认画布是以像素为单位的，前进 1、2 个像素看不出效果，需要把画布的坐标范围设置一下，在图形世界里称为**世界坐标系**。下面的语句表示画布的左下角坐标是(-5,-5)，右上角坐标是(5,5)：

```
setworldcoordinates(-5, -5, 5, 5)
```

最后的代码是这样的：

```
import turtle
import math

setworldcoordinates(-5, -5, 5, 5)
turtle.forward(1)
for i in range(1,17) :
    turtle.left(90)
    turtle.forward((math.pi ** i) / math.factorial(i))
```

我加上了坐标系和两个参考点，最后的图形是这样的：

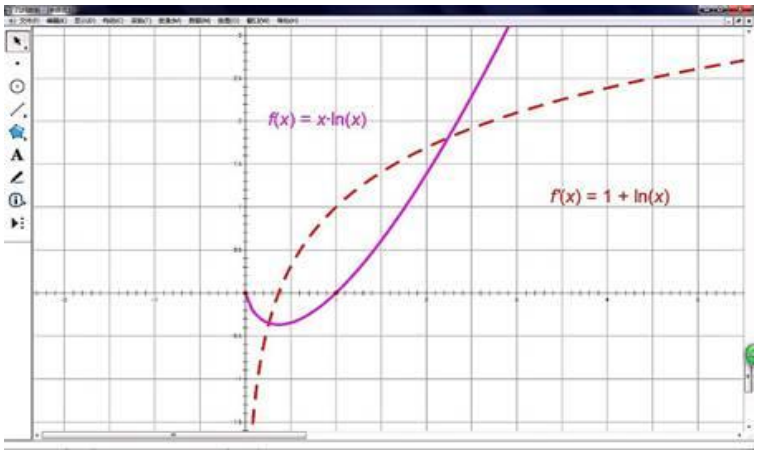


小海龟从原点出发，每走一次，左转 90 度，很快就收敛到(-1, 0)这一点。

40 画函数图像

孩子马上就要参加高考了，我以前还能帮着辅导一下数学功课，现在就不行了，一来她很忙，晚上很晚才到家，二来高中的数学题太变态，琢磨一个小时可能也解不出一道。

前几天她让我帮着打印几张函数及导函数的图像，我发现这些图像都是用一款软件制作的，例如第一幅图像是这样的：

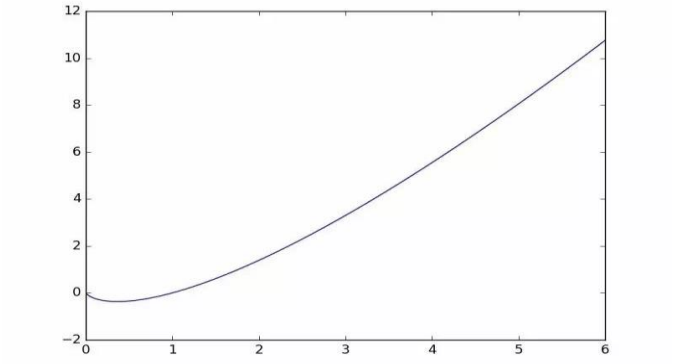


我以前用 Python 画过复利曲线图，这种图像只不过稍微复杂了一点，应该难不倒我，下面就跟着我来一步一步把这个图做出来。

第一步：画出 f(x)=x*ln(x)

复习一下以前学过的内容，换一下函数，马上就完成了这一步。

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 6, 200)
y = x * np.log(x)
plt.plot(x, y)
plt.show()
```

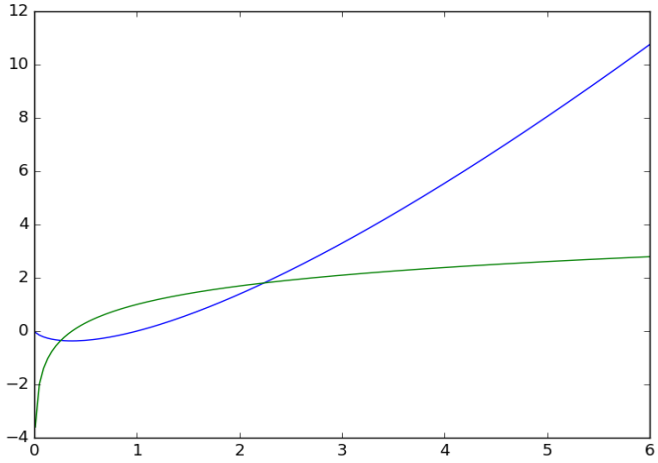


需要说明一下 linspace(-2, 6, 200)相当于在 x 轴从-2 到 6 之间采样 200 个点，形成一个数组。np.log()就是自然对数函数。

第二步：再加上导数图像 f'(x) = 1 + ln(x)

在 plot()函数之前加上两条语句就可以了。

```
dy = 1 + np.log(x)
plt.plot(x, dy)
```



可以看到多了一条曲线，而且自动用了不同的颜色。

第三步：把第二条曲线用红虚线表示

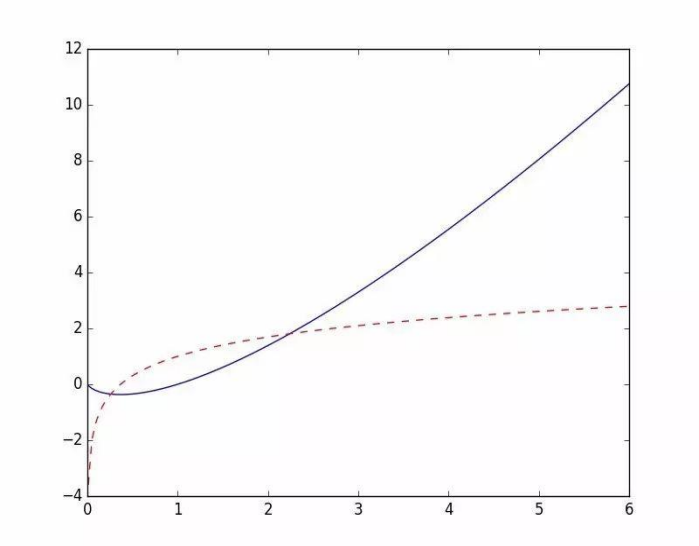
把这一行：

```
plt.plot(x, dy)
```

换成：

```
plt.plot(x, dy, 'r--')
```

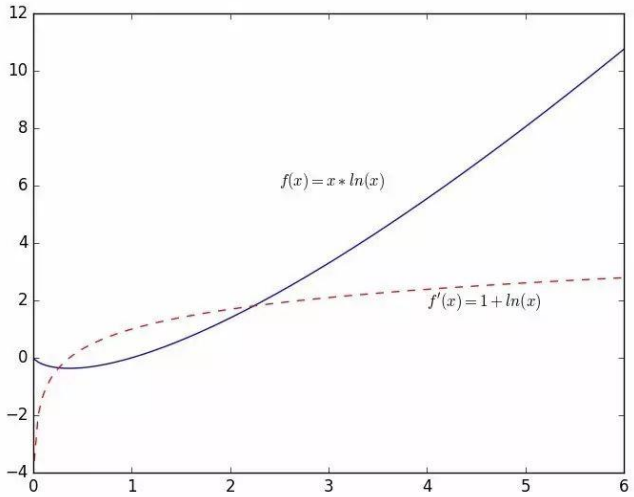
这里的 **r** 表示红色 red，**--** 表示虚线，**r--** 就是红虚线。



第四步：在函数曲线旁边标上函数名称

在 plt.show()函数之前加上两条语句：

```
plt.text(2.5, 6, r'$f(x) = x * ln(x)$')
plt.text(4, 1.8, r"$f'(x) = 1 + ln(x)$")
```



解释一下：

```
r'$f(x) = x * ln(x)$'
```

这个也是字符串，r 的意思是 raw，原始的、未经过加工的，因为'\n'，'\\'等这类特殊字符需要进行转义处理，而用上这个 r 字母开头之后，里面的反斜杠就不当转义符处理了，关于字符串以后还得专门细说一下。

在《零基础学编程 021：获取股票实时行情数据》这一课里我们见过以字母 b 开头的字符串，表示二进制串，例如：**b'**hq="**\xb9\xc8\xb8\xe8**,824.16.....";**\n'**

至于 r'.....'里面的两个$符号，则与 latex 有关，专门用来表示数学公式的，你可以把两个$符号去掉，看一看文字样式有何区别。

第五步：加上网格线和坐标轴

把图的 x 坐标范围设置为[-2, 6]：

```
plt.xlim(-2, 6)
```

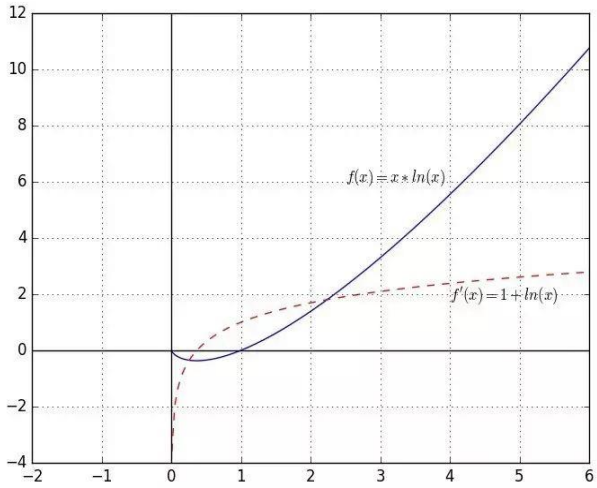
加上网格线：

```
plt.grid(True)
```

加上水平坐标轴和垂直坐标轴：

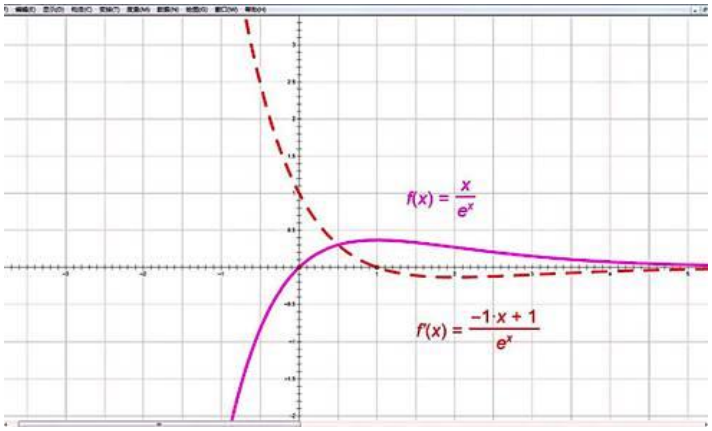
```
plt.axhline(color='black')
plt.axvline(color='black')
```

上面这些语句都放在 plt.show()之前，最后的效果是这样的：



作业：

试着画出下图中的两个函数图像。



后记：

后面有关学 Python 的文章没有继续写下去，喜欢上了 Rust。

由于原来的公众号《申龙斌的程序人生》被封，又启用了新的公众号。

如果在学习过程中有疑问，请关注我的微信公众号【金喜擅】，给我留言。

个人独立博客上还有其它内容，欢迎访问：http://shenlb.me

金喜擅

读书、写作、锻炼、编程、投资，
一切皆定投……