笨办法学Python（中文第3版）

作者：[美] 泽德 A. 肖（Zed A. Shaw）

译者前言

《笨办法学 Python》(Learn Python The Hard Way，简称 LPTHW)是 Zed Shaw 编写的一本 Python 入门书籍。适合对计算机了解不多，没有学过编程，但对编程感兴趣的朋友学习使用。这本书以习题的方式引导读者一步一步学习编程，从简单的打印一直讲到完整项目的实现。也许读完这本书并不意味着你已经学会了编程，但至少你会对编程语言以及编程这个行业有一个初步的了解。

本书区别于其它入门书籍的特点如下：

- 注重实践。本书提供了足够的练习代码，如果你完成了所有的练习（包括加分习题），那你已经写了上万行的代码。要知道很多职业程序员一年也就写几万行代码而已。
- 注重能力培养。除了原序言提到的“读和写”、“注重细节”、以及“发现不同”这样的基本能力以外，本书还培养了读者自己专研问题和寻求答案的能力。
- 注重好习惯的养成。本书详细地讲解了怎样写出好的代码、好的注释、好的项目。这会让你在后续的学习中少走很多弯路。

本书结构非常简单，其实就是 52 个习题。其中 26 个覆盖了输入输出、变量、以及函数三个课题，另外 26 个覆盖了一些比较高级的话题，如条件判断、循环、类和对象、代码测试、以及项目的实现等。每一章节的格式基本都是一样的，以代码练习题开始，读者照着说明编写代码（不允许复制粘贴），运行并检查结果，然后再做一下加分习题就可以了。当然如果你觉得加分习题对你来说有点难，你也可以暂时跳过，以后再完成也没关系。

另外阅读本书还需要你有一定的英文能力。其实学编程不懂英语是很吃亏的，毕竟编程语言都是基于英语，而编程社群的主要交流方式也是英语。不会英语的人在编程界可能就只好当二等公民了。本书的翻译尽量保留了所有的英文专业词汇（可能会有中文说明），而且遵照 Zed 的建议，代码及答案部分没有翻译成中文，读者看到不懂的地方，请自己查字典解决。

如果你对自己的英文能力比较有信心，译者强烈推荐你直接去下载阅读 英文原版 。这本书代码较多，文字内容较少，因此英文原版的阅读理解也比较容易。

LPTHW 的风格和别的书差异很大。它没有像一般的入门书籍一样通过讨好读者以激发读者兴趣，而是直截了当地告诉你你需要做什么，需要注意什么。这种风格可能会让人觉得枯燥乏味，读者姑且把这也当做 Hard Way 的一部分把。所以如果你觉得实在不能适应这种风格，Zed 推荐你看下面两本书:

- How To Think Like A Computer Scientist
- A Byte Of Python 这本书有 中译版

本书的电子版会随时跟着作者更新。你可以通过 Read The Docs 读到最新的网页版内容，也可以到 bitbucket 代码仓库 下载 PDF 文件。如果你对本书的翻译有任何意见和建议，你可以通过 bitbucket 进行反馈。

你可以访问 lulu.com 购买本书的英文印刷版，这也是对原作者的支持。

前言：笨办法更简单

这本小书的目的是让你起步编程。虽然书名说是“笨办法”,但其实并非如此. 所谓的“笨办法”是指本书教授的方式。在这本书的帮助下，你将通过非常简单的练习学会一门编程语言。做练习 是每个程序员的必经之路：

1. 做每一道习题
2. 一字不差地写出每一个程序
3. 让程序运行起来

就是这样了。刚开始这对你来说会非常难，但你需要坚持下去。如果你通读了这本书，每晚花个一两小时做做习题，你可以为自己读下一本编程书籍打下良好的基础。通过这本书你学到的可能不是真正的编程，但你会学到最基本的学习方法。

这本书的目的是教会你编程新手所需的三种最重要的技能：读和写、注重细节、发现不同。

读和写

很显然，如果你连打字都成问题的话，那你学习编程也会成问题。尤其如果你连程序源代码中的那些奇怪字符都打不出来的话，就根本别提编程了。没有这样基本技能的话，你将连最基本的软件工作原理都难以学会。

为了让你记住各种符号的名字并对它们熟悉起来，你需要将代码写下来并且运行起来。这个过程也会让你对编程语言更加熟悉。

注重细节

区分好程序员和差程序员的最重要的一个技能就是对于细节的注重程度。事实上这是任何行业区分好坏的标准。如果缺乏对于工作的每一个微小细节的注意，你的工作成果将缺乏重要的元素。以编程来讲，这样你得到的结果只能是毛病多多难以使用的软件。

通过将本书里的每一个例子一字不差地打出来，你将通过实践训练自己，让自己集中精力到你作品的细节上面。

发现不同

程序员长年累月的工作会培养出一个重要技能，那就是对于不同点的区分能力。有经验的程序员拿着两份仅有细微不同的程序，可以立即指出里边的不同点来。程序员甚至造出工具来让这件事更加容易，不过我们不会用到这些工具。你要先用笨办法训练自己，等你具备一些相关能力的时候才可以使用这些工具。

在你做这些练习并且打字进去的时候，你一定会写错东西。这是不可避免的，即使有经验的程序员也会偶尔写错。你的任务是把自己写的东西和要求的正确答案对比，把所有的不同点都修正过来。这样的过程可以让你对于程序里的错误和 bug 更加敏感。

不要复制粘贴

你必须手动将每个练习打出来。复制粘贴会让这些练习变得毫无意义。这些习题的目的是训练你的双手和大脑思维，让你有能力读代码、写代码、观察代码。如果你复制粘贴的话，那你就是在欺骗自己，而且这些练习的效果也将大打折扣。

对于坚持练习的一点提示

在你通过这本书学习编程时，我正在学习弹吉他。我每天至少训练 2 小时，至少花一个小时练习音阶、和声、和琶音，剩下的时间用来学习音乐理论和歌曲演奏以及训练听力等。有时我一天会花 8 个小时来练习，因为我觉得这是一件有趣的事情。对我来说，要学好一样东西，每天的练习是必不可少的。就算这天个人状态很差，或者说学习的课题实在太难，你也不必介意，只要坚持尝试，总有一天困难会变得容易，枯燥也会变得有趣了。

在你通过这本书学习编程的过程中要记住一点，就是所谓的“万事开头难”，对于有价值的事情尤其如此。也许你是一个害怕失败的人，一碰到困难就想放弃。也许你是一个缺乏自律的人，一碰到“无聊”的事情就不想上手。也许因为有人夸你“有天分”而让你自视甚高，不愿意做这些看上去很笨拙的事情，怕有负你”神童”的称号。也许你太过激进，把自己跟有20多年经验的编程老手相比，让自己失去了信心。

不管是什么原因，你一定要坚持下去。如果你碰到做不出来的加分习题，或者碰到一节看不懂的习题，你可以暂时跳过去，过一阵子回来再看。只要坚持下去，你总会弄懂的。

一开始你可能什么都看不懂。这会让你感觉很不舒服，就像学习人类的自然语言一样。你会发现很难记住一些单词和特殊符号的用法，而且会经常感到很迷茫，直到有一天，忽然一下子你会觉得豁然开朗，以前不明白的东西忽然就明白了。如果你坚持练习下去，坚持去上下求索，你最终会学会这些东西的。也许你不会成为一个编程大师，但你至少会明白程序是怎么工作的。

如果你放弃的话，你会失去达到这个程度的机会。你会在第一次碰到不明白的东西时(几乎是所有的东西)放弃。如果你坚持尝试，坚持写习题，坚持尝试弄懂习题的话，你最终一定会明白里边的内容的。

如果你通读了这本书，却还是不知道编程是怎么回事。那也没关系，至少你尝试过了。你可以说你已经尽过力但成效不佳，但至少你尝试过了。这也是一件值得你骄傲的事情。

给“小聪明”们的警告

有的学过编程的人读到这本书，可能会有一种被侮辱的感觉。其实本书中没有任何要居高临下地贬低任何人的意思。只不过是我比我面向的读者群知道的更多而已。如果你觉得自己比我聪明，然后觉得我在居高临下，那我也没办法，因为你根本就不属于我的目的读者群。

如果你觉得这本书里到处都在侮辱你的智商，那我对你有三个建议:

1. 别读这本书了。我不是写给你的，我是写给需要学习的人的。
2. 放下架子好好学。如果你认为你什么都知道，那你就很难从比你强的人身上学到什么了。
3. 学 Lisp 去。我听说什么都知道的人可喜爱 Lisp 了。

对于其他在这里学习的人，你们读的时候就想着我在微笑就可以了，虽然我的眼睛里还带着恶作剧的闪光。

许可协议

特别感谢

首先我要感谢帮助我完成这版书的人。首先是 Pretty Girl Editing Services 可爱的编辑所做的编辑工作。然后是 Greg Newman，他提供了美工图并帮我设计了封面，而且还帮忙复审了本书。是他让这本书看上去像本真正的书籍，而且就算我没在第一版里提到他的辛劳，他也没跟我计较。我还要感谢 Brian Shumate 在网站设计方面的帮助，这方面的帮助也是我非常需要的。

最后，我还要感谢成千上万读过本书第一版而且提出 bug 报告和改进建议的读者。你们的贡献让这本书的内容更为扎实，没有你们我是做不到的。谢谢你们。

习题 0: 准备工作

这道习题并没有代码内容，它的主要目的是让你在计算机上安装好 Python。你应该尽量照着说明进行操作，例如 Mac OSX 默认已经安装了 Python 2，所以就不要在上面安装Python 3 或者别的 Python 版本了。

Warning

如果你不知道怎样使用 Windows 下的 PowerShell，或者 OSX 下的 Terminal，或者 Linux 下的"bash"，那你就需要学习了。我有一个免费的快速入门教程放在 http://cli.learncodethehardway.org/ ，你可以快速学到 PowerShell 和 Terminal 的基本用法。学完后再回来看这本书吧。

Mac OSX

你需要做下列任务来完成这个练习：

1. 用浏览器打开 http://www.barebones.com/products/textwrangler/ 找到并安装 `TextWrangler` 文本编辑器。
2. 把 TextWrangler (也就是你的编辑器) 放到 Dock 中，以方便日后使用。
3. 找到系统中的 "命令行终端(Terminal)" 程序。到处找找，你会找到的。
4. 把 Terminal 也放到 Dock 里面。
5. 运行 Terminal 程序，这个程序看上去不怎么地。
6. 在 Terminal 程序里边运行 `python` 。运行的方法是输入程序的名字再敲一下回车。
7. 敲击 CTRL-D (^D) 退出 python。
8. 这样你就应该退回到敲 `python` 前的提示界面了。如果没有的话自己研究一下为什么。
9. 学着使用 Terminal 创建一个目录，你可以上网搜索怎样做。
10. 学着使用 Terminal 进入一个目录，同样你可以上网搜索。
11. 使用你的编辑器在你进入的目录下建立一个文件。你将建立一个文件。使用 "Save" 或者 "Save As..." 选项，然后选择这个目录。
12. 使用键盘切换回到 Terminal 窗口，如果不知道怎样使用键盘切换，你一样可以上网搜索。
13. 回到 Terminal，看看你能不能使用命令看到你新建的文件，上网搜索如何将文件夹中的内容列出来。

OSX: 你应该看到的结果

以下是我在自己电脑的 Terminal 中执行上述练习时看到的内容。和你做的结果会有一些不同，所以看看你能不能找出两者不同点来。

```
Last login: Sat Apr 24 00:56:54 on ttys001

~ $ python

Python 2.5.1 (r251:54863, Feb  6 2009, 19:02:12)

[GCC 4.0.1 (Apple Inc. build 5465)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

> >> ^D

~ $ mkdir mystuff

~ $ cd mystuff

mystuff $ ls

#  ... 使用 TextWrangler 编辑 test.txt ...

mystuff $ ls

test.txt

mystuff $
```

Windows

Note

感谢 zhmark 的贡献。

1. 用浏览器打开 http://notepad-plus-plus.org/ 下载并安装 `Notepad++` 文本编辑器。这个操作无需管理员权限。

2. 把 Notepad++ 放到桌面或者快速启动栏，这样你就可以方便地访问到该程序了。这两条在安装选项中可以看到。
3. 从开始菜单运行“PowerShell”程序。你可以使用开始菜单的搜索功能，输入名称后敲回车即可打开。
4. 为它创建一个快捷方式，放到桌面或者快速启动栏中以方便使用。
5. 运行命令行终端程序(也就是 PowerShell)，这个程序看上去不怎么地。
6. ***在命令行终端里边运行 `python` 。运行的方法是输入程序的名字再敲一下回车。***
    1. 如果你运行 python 发现它不存在(python 不是可执行命令，或者系统找不到python云云)。你需要访问 http://python.org/download 并且安装 Python。
    2. 确认你安装的是 **Python 2** 而不是 Python 3。
    3. 你也可以试试 ActiveState Python，尤其是你没有管理员权限的时候。
    4. 如果你安装好了但是 python 还是不能被识别，那你需要在 powershell 下输入并执行以下命令：

        ```
        [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Python27", "User")
        ```

    5. 关闭并重启 powershell，确认 python 现在可以运行。如果不行的话你可能需要重启电脑。
7. 键入 CTRL-Z (^Z)，再敲回车以退出 python 。
8. 这样你就应该退回到敲 python 前的提示界面了。如果没有的话自己研究一下为什么。
9. 学着使用 Terminal 创建一个目录，你可以上网搜索怎样做。
10. 学着使用 Terminal 进入一个目录。同样你可以上网搜索。
11. 使用你的编辑器在你进入的目录下建立一个文件。你将建立一个文件，使用 “Save” 或者 “Save As...” 选项，然后选择这个目录。
12. 使用键盘切换回到 Terminal 窗口，如果不知道怎样使用键盘切换，你一样可以上网搜索。
13. 回到 Terminal，看看你能不能使用命令看到你新建的文件，上网搜索如何将文件夹中的内容列出来。

Warning

有时这一步你会漏掉：Windows 下装了 Python 但是没有正确配置路径。 确认你在 powershell 下输入了 [Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Python27", "User") 。你也许需要重启 powershell 或者计算机来让路径设置生效。

Windows: 你应该看到的结果

```
>  python
ActivePython 2.6.5.12 (ActiveState Software Inc.) based on
Python 2.6.5 (r265:79063, Mar 20 2010, 14:22:52) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
> >> ^Z



>  mkdir mystuff


>   cd  mystuff


... 使用 Notepad++ 编辑 mystuff 目录下的 test.txt ...
```

```
>
   <如果你没有使用管理员权限安装，你会看到一堆错误。忽略它们，按回车即可。>
>  dir
 Volume in drive C is
 Volume Serial Number is 085C-7E02


 Directory of C:\Documents and Settings\you\mystuff


04.05.2010  23:32    <DIR>          .
04.05.2010  23:32    <DIR>          ..
04.05.2010  23:32                 6 test.txt
               1 File(s)              6 bytes
               2 Dir(s)  14 804 623 360 bytes free


>
```

你看到的命令行信息，Python 信息，以及其它一些东西可能会非常不一样，不过应该大致不差。你可以通过 http://learnpythonthehardway.org 把你找到的错处告诉我们，我们会修正过来。

Linux

Linux 系统可谓五花八门，安装软件的方式也各有不同。我们假设作为 Linux 用户的你已经知道如何安装软件包了，以下是给你的操作说明：

1.
    1. 用浏览器打开 http://learnpythonthehardway.org/exercise0.html 下载并安装 `gedit` 文本编辑器。
    2. ***把 gedit (也就是你的编辑器) 放到窗口管理器显见的位置，以方便日后使用。***
    3. 运行 gedit，我们要先改掉一些愚蠢的默认设定。
    4. 从 `gedit menu` 中打开 `Preferences` ，选择 `Editor` 页面。
    5. 将 `Tab width:` 改为 4。
    6. 选择 (确认有勾选到该选项) `Insert spaces instead of tabs` 。
    7. 然后打开 “Automatic indentation” 选项。
    8. 转到 `View` 页面，打开 “Display line numbers” 选项。
3. 找到 “Terminal” 程序。它的名字可能是 `GNOME Terminal` 、 `Konsole` 、或者 `xterm` 。
4. 把 Terminal 也放到 Dock 里面。
5. 运行 Terminal 程序，这个程序看上去不怎么地。
6. 在 Terminal 程序里边运行 `python` 。运行的方法是输入程序的名字再敲一下回车。 a. 如果你运行 `python` 发现它不存在的话，你需要安装它，而且要确认你安装的是 Python 2 而非 Python 3。
7. 敲击 CTRL-D (^D) 以退出 `python` 。
8. 这样你就应该退回到敲 `python` 前的提示界面了。如果没有的话自己研究一下为什么。
9. 学着使用 Terminal 创建一个目录。你可以上网搜索怎样做。
10. 学着使用 Terminal 进入一个目录。同样你可以上网搜索。
11. 使用你的编辑器在你进入的目录下建立一个文件。你将建立一个文件，使用 “Save” 或者 “Save As...” 选项，然后选择这个目录。
12. 使用键盘切换回到 Terminal 窗口，如果不知道怎样使用键盘切换，你一样可以上网搜索。

13. 回到 Terminal，看看你能不能使用命令看到你新建的文件，上网搜索如何将文件夹中的内容列出来。

Linux: 你应该看到的结果

```
[~]$ python
Python 2.6.5 (r265:79063, Apr  1 2010, 05:28:39)
[GCC 4.4.3 20100316 (prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
> >>
[~]$ mkdir mystuff
[~]$ cd mystuff
#  ... 使用gedit编辑text.txt ...
[mystuff]$ ls
test.txt
[mystuff]$
```

你看到的命令行信息，Python 信息，以及其它一些东西可能会非常不一样。不过应该大致不差就是了。

给新手的告诫

你已经完成了这节练习，取决于你对计算机的熟悉程度，这个练习对你而言可能会有些难。如果你觉得有难度的话，你要自己克服困难，多花点时间学习一下。因为如果你不会这些基础操作的话，编程对你来说将会更难学习。

如果有程序员告诉你让你使用 `vim` 或者 `emacs` ，那你应该拒绝他们。当你成为一个更好的程序员的时候，这些编辑器才会适合你使用。你现在需要的只是一个可以编辑文字的编辑器。我们使用 `gedit` 是因为它很简单，而且在不同的系统上面使用起来是一样的。就连专业程序员也会使用 `gedit` ，所以对于初学而言它已经足够了。

也许有程序员会告诉你让你安装和学习 Python 3。你应该告诉他们“等你电脑里的所有python 代码都支持 Python 3 了，我再试着学学吧。”你这句话足够他们忙活个十来年的了。

总有一天你会听到有程序员建议你使用 Mac OSX 或者 Linux。如果他喜欢字体美观，他会告诉你让你弄台 Mac OSX 计算机，如果他们喜欢操作控制而且留了一部大胡子，他会让你安装 Linux。这里再次向你说明，只要是一台手上能用的电脑就可以了。你需要的只有三样东西: `gedit` 、一个命令行终端、还有 `python` 。

最后要说的是这节练习的准备工作的目的，也就是让你可以在以后的练习中顺利地做到下面的这些事情：

1. **写出** 习题的代码，在 Linux 下用 `gedit` ，OSX 下用 `TextWrangler` ，Windows 下用 `Notepad++` 。 2. **运行** 你写的习题。 3. **修改** 错误的地方。 4. 重复上述步骤。

其他的事情只会让你更困惑，所以还是坚持按计划进行吧。

习题 1: 第一个程序

你应该在练习 0 中花了不少的时间，学会了如何安装文本编辑器、运行文本编辑器、以及如何运行命令行终端，而且你已经花时间熟悉了这些工具。请不要跳过前一个练习的内容直接进行下面的内容，这也是本书唯一的一次这样的警示。

将上面的内容写到一个文件中，取名为 `ex1.py` 。这个命名方式很重要，Python 文件最好以 `.py` 结尾。

```
1          print   "Hello World!"
2          print   "Hello Again"
3          print   "I like typing this."
4          print   "This is fun."
5          print   'Yay! Printing.'
6          print   "I'd much rather you 'not'."
7          print   'I "said" do not touch this.'
```

如果你使用的是 Mac OSX 下的 TextWrangler，那你的文本编辑器大致是这个样子:

如果你在 Windows 下使用 Notepad++，那你看到的应该是这个:

别担心编辑器长得是不是一样，关键是以下几点：

1. 注意我没有输入左边的行号（1-7）。这些是额外打印到书里边的，以方便对代码具体的某一行进行讨论。例如“参见第 5 行……”你无需将这些也写进 python 脚本中去。
2. 注意我截图中开始的 `print` 语句，它和代码范例中是完全一样的，而且是精确的完全相同，不仅仅是表面相似而已。要让这段脚本正常工作，代码中的每个字符都必须完全匹配。当然，显示的颜色可能是不同的，颜色并不重要，只有字符才是重要的。

然后你需要在命令行终端通过输入以下内容来运行这段代码：

```
python ex1.py
```

如果你写对了的话，你应该看到和下面一样的内容。如果不一样，那就是你弄错了什么东西。不是计算机出错了，计算机没错。

你应该看到的结果

在 Mac OSX 的 Terminal 下面你应该看到以下内容:

在 Windows 的 PowerShell 下你应该看到这些:

你也许会看到 `python ex1.py` 前面显示了不同的用户名，计算机名，以及其他一些信息，这不是问题，重要的是你输入了命令，而且看到了相同的输出。

如果你看到类似如下的错误信息：

```
1          $  python ex/ex1.py
2            File "ex/ex1.py", line 3
3              print "I like typing this.
4                                      ^
5          SyntaxError: EOL while scanning string literal
```

这些内容你应该学会看懂的，这是很重要的一点，因为你以后还会犯类似的错误。就是我现在也会犯这样的错误。让我们一行一行来看。

1. 首先我们在命令行终端输入命令来运行 `ex1.py` 脚本。
2. Python 告诉我们 `ex1.py` 文件的第 3 行有一个错误。
3. 然后这一行的内容被打印了出来。
4. 然后 Python 打印出一个 `^` (井号，caret) 符号，用来指示出错的位置。 注意到少了一个 `"` (双引号，double-quote) 符号了吗？
5. 最后，它打印出了一个“语法错误(SyntaxError)”告诉你究竟是什么样的错误。通常这些错误信息都非常难懂，不过你可以把错误信息的内容复制到搜索引擎里，然后你就能看到别人也遇到过这样的错误， 而且你也许能找到如何解决这个问题。

Warning

如果你来自另外一个国家，而且你看到关于 ASCII 编码的错误，那就在你的 python 脚本的最上面加入这一行：

```
# -*- coding: utf-8 -*-
```

这样你就在脚本中使用了 unicode UTF-8 编码，这些错误就不会出现了。

加分习题

你还会有 `加分习题` 需要完成。加分习题里边的内容是供你尝试的。如果你觉得做不出来，你可以暂时跳过，过段时间再回来做。

在这个练习中，试试这些东西：

1. 让你的脚本再多打印一行。
2. 让你的脚本只打印一行。
3. 在一行的起始位置放一个 ‘#’ (octothorpe) 符号。它的作用是什么？自己研究一下。

从现在开始，除非特别情况，我将不再解释每个习题的工作原理了。

Note

井号有很多的英文名字，例如：’octothorpe(八角帽)’，’pound(英镑符)’, ‘hash(电话的#键)’, ‘mesh(网)’ 等。

习题 2: 注释和井号

程序里的注释是很重要的。它们可以用自然语言告诉你某段代码的功能是什么。在你想要临时移除一段代码时，你还可以用注解的方式将这段代码临时禁用。接下来的练习将让你学会注释:

```
1          # A comment, this is so you can read your program later.
2          # Anything after the # is ignored by python.
3
4          print   "I could have code like this."   # and the comment after is ignored
5
6          # You can also use a comment to "disable" or comment out a piece of code:
7          # print "This won't run."
8
9          print   "This will run."
```

从现在开始，我将用这样的方式来写代码。我一直在强调“完全相同”，不过你也不必按照字面意思理解。你的程序在屏幕上的显示可能会有些不同，不过重要的是你在文本编辑器中输入的文本的正确性。事实上，我可以用任何编辑器写出这段程序，而且内容是完全一样的。

你应该看到的结果

```
$  python ex2.py
I could have code like this.
This will run.
$
```

再次说明，我不会再贴各种屏幕截图了。你应该明白上面的内容是输出内容的字面翻译，而 `$ python ...` 和最后的 `$` 之间才是你应该关心的内容。

加分习题

1. 弄清楚’#’符号的作用。而且记住它的名字。(中文为井号，英文为 octothorpe 或者 pound character)。
2. 打开你的 `ex2.py` 文件，从后往前逐行检查。从最后一行开始，倒着逐个单词单词检查回去。
3. 有没有发现什么错误呢？有的话就改正过来.
4. 朗读你写的习题，把每个字符都读出来。有没有发现更多的错误呢？有的话也一样改正过来。