# 跟老齐学Python：从入门到精通

**作者：齐伟**

**版权信息**

书名：跟老齐学Python：从入门到精通
作者：齐伟
出版社：电子工业出版社
ISBN：978-7-121-28034-4
定价：69.00

# 前言

这是一本学习材料，是为编程“零基础”的朋友学习Python提供的类似教材的学习材料，所以，内容会有庞杂琐碎之感，但这对于“零基础”的读者来讲是不可缺少的。所以，不要把这本书当作“开发手册”来用。

本书虽然是以“零基础”起步，但是并不打算仅仅涉及一些浅显的入门知识，当然基础知识是必不可少的，还想为“零基础”的朋友多提供一些知识，一些所谓高级的内容，既满足了好奇心，也可以顺势深入研究。当然，真正的深入还需要读者自己努力。

“敬畏上帝是智慧的开端”。在本书的编写过程中，一直惶恐于能否所言无误，但水平有限，错误难免，敬请读者指出，并特别建议，对有异议的地方，请使用Google网站搜索更多的资料进行比较阅读，也可以跟我联系，共同探讨。为了便于进行技术交流，我创建了一个QQ群（群号：26913719），专供本书读者研讨技术问题。

完成本书是一个比较漫长的过程，在这个过程中，得到了很多朋友的帮助，在这里对他们表示感谢，并将他们的名号列在下面：

李航、令狐虫、github641、dongm2ez、wdyggh、codexc、winecat、solarhell、ArtinHuang、吴优。

在本书编辑过程中，电子工业出版社的编辑高洪霞、黄爱萍为本书的面世做出了极大的努力，对她们的工作表示诚挚感谢。

最后，要感谢我的妻子，在本书的写作过程中，她给了我很多鼓励，还协助我检查文本内容。

希望这本书能够为有意学习Python的读者提供帮助。

齐伟

2016年1月

# 第1季　基础

从这里开始，请读者——你已经确信自己是要学习Python的准程序员了——跟我一起，领略一番Python的基础知识，这是学好Python的起步，同时，其内容也和其他的高级编程语言有相通之处。所以，学习Python是一种“性价比”非常高的事情。

在本季中，要向读者介绍Python的基本对象类型、语法规则和函数的相关知识。学习完这些内容，就能够用Python做很多事情了，且在其中还会不断强化一种掌握Python的方法。

# 第0章　预备

从现在开始，本书将带领你——零基础的学习者——进入到Python世界。进入这个世界，你不仅能够体会到Python的魅力，感受到编程的快乐，还顺便可以成为一个程序员，我相信你一定能成为一个伟大的程序员，当然这并不是本书的目的，更不是本书的功劳。当你成为一个技术大牛的时候，最应该感谢的是你的父母，如果你顺便也感谢一下本书，比如多购买一些本书分发给你的弟兄们，那是我的福份，感激不尽。

预备，Let’s go！

## 0.1 关于Python的故事

学习一种编程语言是一件很有意思的事情，从现在开始，我就和你一起来学习一种叫作Python的编程语言。

在编程界，存在着很多某种语言的忠实跟随者，因为忠实，就会如同卫道士一样有了维护那种语言荣誉的义务，所以总见到有人争论哪种语言好、哪种语言不好。当然，好与坏的标准是不一样的，有些人以学了之后能不能挣大钱为标准，有些人以是否容易学为标准，或许还有人以能不能将来和妹子一同工作为标准（也或许没有），甚至有些人就没有什么标准，只是凭感觉或者道听途说而人云亦云罢了。

读者在本书中将看到一个颇为迷恋于Python的人，因为全书看不到一句有关Python的坏话（如果有，则肯定是笔误，是应该删除的部分）。

不管是语言还是其他什么，挑缺点是比较容易的事情，但找优点都是困难的，所以，《圣经》中那句话——为什么你看见弟兄的眼中有刺，却不想自己眼中有梁木呢？——是值得我们牢记的。

在本书开始就废话连篇，显见本书不会有什么“干货”，倒是“水货”颇多，并不是因为“水是生命的源泉”，而是因为作者水平有限，如果不掺“水”，唯恐说不清道不明，还敬请读者谅解。嫌“水”多的，就此可以合上本书去看网上的各种电影吧。也不用在网上喷我，因为那样只能增加更多的“口水”（还是水）。

下面说点儿正经的。

### 0.1.1 Python的昨天、今天和明天

这个题目有点大了，似乎回顾过去、考察现在、张望未来都是那些掌握方向的大人物做的。那就让我们每个人都成为大人物吧。因为如果不回顾一下历史，似乎无法满足好奇心；如果不考察一下现在，也不放心（担心学了之后没有什么用途）；如果不张望一下未来，怎么能吸引（也算是一种忽悠吧）你呢？

1.Python的历史

历史向来是成功者的传记，现在流传的关于Python的历史也是如此。

Python的创始人为吉多·范罗苏姆（Guido van Rossum），关于他开发Python的过程，很多资料里面都要记录下面的故事：

1989年的圣诞节期间，吉多·范罗苏姆为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序，作为ABC语言的一种继承。之所以选中Python作为程序的名字，是因为他是一个蒙提·派森的飞行马戏团的爱好者。ABC是由吉多参加设计的一种教学语言，在吉多本人看来，ABC这种语言非常优美和强大，是专门为非专业程序员设计的。但是ABC语言并没有成功，究其原因，吉多认为是非开放造成的。吉多决心在Python中避免这一错误，并取得了非常好的效果，完美结合了C和其他一些语言。

这个故事是从维基百科里面直接复制过来的，很多讲Python历史的资料里面，也都转载了这一段文字。但是，在我来看，吉多是为了“打发时间”而决定开发Python，源自他的这样一段自述：

Over six years ago，in December 1989，I was looking for a"hobby"programming project that would keep me occupied during the week around Christmas.My office（a government-run research lab in Amsterdam）would be closed，but I had a home computer，and not much else on my hands.I decided to write an interpreter for the new scripting language I had been thinking about lately:a descendant of ABC that would appeal to Unix/C hackers.I chose Python as a working title for the project，being in a slightly irreverent mood（and a big fan of Monty Python's Flying Circus）.（原文地址：https://www.python.org/doc/essays/foreword/）

首先，必须承认，这个哥们儿是一个非常牛的人，此处献上恭敬的崇拜。

其次，刚刚开始学习Python的朋友，可千万别认为Python是一个可以随随便便鼓捣的东西，人家也是站在巨人的肩膀上的。

第三，牛人在成功之后，往往把奋斗的过程描绘得比较简单。或者是出于谦虚，或者是为了让人听起来他更牛。反正，我们看最后结果的时候，很难感受过程中的酸甜苦辣。

不管怎样，吉多·范罗苏姆在那时刻创立了Python，而且，更牛的在于他具有现代化的思维——开放，并通过Python社区，吸引来自世界各地的开发者，参与Python的建设。在这里，请读者一定要联想到Linux和它的创始人林纳斯·托瓦兹。两者都秉承“开放”思想，得到了来自世界各地开发者和应用者的欢呼和尊敬。

请读者向所有倡导“开放”的牛人们表示敬意，是他们让这个世界变得更美好，他们以行动诠释了热力学第二定律——“熵增原理”。

2.Python的现在

Python现在越来越火了，因为它搭上了“大数据”、“云计算”、“自然语言处理”等这些时髦名词的便车。

网上时常会有一些编程语言排行榜之类的东西，有的初学者常常被排行榜所迷惑，总想要学习排列在第一位的，认为排在第一位的编程语言需求量大。不管排行榜是怎么编制的，Python虽然没有登上状元、榜眼、探花之位，但也不太靠后呀。

另外一个信息，更能激动一下初学者那颗脆弱的小心脏。

Dice.com网上对20000名IT专业人士进行调查的结果显示：Java类程序员平均工资91060美元；Python类程序员平均工资90208美元。

Python程序员比Java程序员的平均工资低，但看看差距，再看看两者的学习难度，学习Python绝对是一个性价比非常高的投资。

这么合算的编程语言不学等待何时？

3.Python的未来

Python的未来要靠读者了，你学好了、用好了，未来它就光明了，它的未来在你手里。如图0-1所示为Python创始人吉多·范罗苏姆。

### 0.1.2 Python的特点

很多高级语言都宣称自己是简单的、入门容易的，并且具有普适性，但真正能做到这些的，只有Python。有朋友做了一件衬衫，上面写着“生命有限，我用Python”，这说明什么？说明Python有着简单、开发速度快、节省时间和精力的特点。因为它是开放的，有很多可爱的开发者（为开放社区做贡献的开发者是最可爱的人），将常用的功能做好了放在网上，谁都可以拿过来使用。这就是Python，这就是开放。

图0-1 Python创始人：吉多·范罗苏姆

恭敬地抄录来自《维基百科》的描述：

Python是完全面向对象的语言，函数、模块、数字、字符串都是对象，并且完全支持继承、重载、派生、多继承，有益于增强源代码的复用性。Python支持重载运算符，因此也支持泛型设计。相对于Lisp这种传统的函数式编程语言，Python对函数式设计只提供了有限的支持。有两个标准库（functools和itertools）提供了Haskell和Standard ML中久经考验的函数式程序设计工具。

虽然Python可能被粗略地分类为“脚本语言”（Script Language），但实际上一些大规模软件开发项目（例如Zope、Mnet、BitTorrent及Google）也都广泛地使用它。Python的支持者较喜欢称它为一种高级动态编程语言，原因是“脚本语言”泛指仅做简单程序设计任务的语言，如shell script、VBScript等，但其只能处理简单任务的编程语言，并不能与

Python相提并论。

Python本身被设计为可扩充的，并非所有的特性和功能都集成到语言核心。Python提供了丰富的API和工具，以便程序员能够轻松地使用C、C++、Cython来编写扩充模块。Python编译器本身也可以被集成到其他需要脚本语言的程序内。因此，很多人还把Python作为一种“胶水语言”（glue language）使用，使用Python将其他语言编写的程序进行集成和封装。在Google内部的很多项目，例如Google Engine使用C++编写性能要求极高的部分，然后用Python或Java/Go调用相应的模块。《Python技术手册》的作者马特利（Alex Martelli）说：“2004年，Python已在Google内部使用，Google招募许多Python高手，但在这之前就已决定使用Python。他们的目的是尽量使用Python，在不得已时改用C++；在操控硬件的场合使用C++，在快速开发时使用Python。”

可能这里面有一些术语还不是很理解，没关系，只要明白：Python是一种很牛的语言，应用简单，功能强大，Google都在使用，这就足够了，足够让你下决心学习了。

### 0.1.3 Python哲学

Python之所以与众不同，还在于它强调一种哲学理念：优雅、明确、简单。有一段诗歌读起来似乎很玄，但真实反映了Python开发者的开发理念：

```
The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than “right” now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

网上能够看到这段文字的中文译本，读者可以去搜索阅读。

## 0.2　从小工到专家

这个标题，我借用了一本书的名字——《程序员修炼之道：从小工到专家》，并在此特别推荐阅读。

“从小工到专家”也是很多刚学习编程的朋友的愿望。如何能实现呢？《程序员修炼之道：从小工到专家》这本书中，给出了非常好的建议，值得借鉴。

有一个学习Python的朋友曾问我：“书已经看了，书上的代码也运行过了，习题也能解答了，但是还不知如何开发一个真正的应用程序，不知从何处下手，怎么办？”

另外，也遇到过一些刚刚毕业的大学生，从简历上看，相关专业的考试分数是不错的（我一般相信那些成绩是真的），但是，一讨论到专业问题，常常不知所云，特别是当让他面对真实的工作对象时，表现出来的比成绩单差太多了。

对于上述情况，我一般会武断地下一个结论：练得少。

要从小工成长为专家，必经之路是要多阅读代码，多调试程序。古言“拳不离手，曲不离口”，多练习是成为专家的唯一途径。

### 0.2.1　零基础

有一些初学者，特别是非计算机专业的人，担心自己基础差，不能学好Python。诚然，在计算机方面的基础越好，对学习任何一门新的编程语言越有利。但如果是“绝对零基础”也不用担心，本书就是从这个角度切入来满足你的需要的。凡事总得有一个开始，那么就让本书成为你学习编程语言的开始吧。

就我个人来看，Python是比较适合作为学习编程的入门语言的。

美国有不少高校也这么认为，他们纷纷用Python作为编程专业甚至是非编程专业的大学生入门语言，如图0-2所示为美国各高校设立的编程语言专业。

图0-2　美国高校设立的编程语言专业

总而言之，学习Python，你不用担心基础问题。

### 0.2.2　阅读代码

有句话说得好：“读书破万卷，下笔如有神”，这也适用于编程。通过阅读别人的代码，“站在巨人的肩膀上”，让自己眼界开阔，思维充实。

阅读代码的最好地方就是：www.github.com。

如果你还没有账号，请尽快注册，它将是你成为一个优秀程序员的起点。当然，不要忘记来follow我，我的账号是：qiwsir。

阅读代码最好的方法是一边阅读、一边进行必要的注释，这样可以梳理对别人代码的认识。然后可以run一下，看看效果。当然，还可以按照自己的设想进行必要修改，然后再run。经过几轮，就可以将别人的代码消化吸收了。

### 0.2.3　调试程序

阅读是信息的吸收过程，写作则是信息的加工输出过程。

要自己动手写程序。“一万小时定律”在编程领域也是成立的，除非你天生就是天才，否则，只有通过“一万小时定律”才能成为天才。

在调试程序的时候，要善于应用网络，看看类似的问题别人是如何解决的，不要仅局限于自己的思维范围。利用网络就少不了使用搜索引擎，在此特别向那些要想成为专家的小工们说：Google能够帮助你成为专家。

我不相信“三天掌握Python”、“三周成为高手”之类的让人听起来热血沸腾、憧憬无限的骗人宣传。如果你通过本书跟我对话，至少说明你我都是普通人，普通人要做好一件事情，除了“机缘巧合”遇到贵人之外，就要靠自己勤学苦练了，没有捷径，凡是宣传捷径的，大多都是骗子。

## 0.3 安装Python

任何高级语言都需要一个自己的编程环境，这就好比写字一样，需要有纸和笔，在计算机上写东西，也需要有文字处理软件，比如各种名称的Office类软件。笔和纸以及Office软件，就是写东西的硬件或软件，总之，那些文字只能写在相应的硬件或软件上，才能最后成为一篇文章。编程也要有个程序之类的东西，把代码写在上面，才能形成类似文章那样的文件——自己编的程序。

阅读本书的零基础朋友，乃至于非零基础的朋友，不要希望在这里学到很多高深的Python语言技巧。重要的是学会一些方法，比如刚才给大家推荐的“上网Google一下”，就是非常好的学习方法。互联网的伟大之处，不仅仅在于打游戏、看看养眼的照片或者各种视频之类的，我衷心希望本书的读者不仅仅把互联网当作娱乐网，还要当作知识网和创造网。

扯远了，拉回来。在学习过程中，遇到一点点疑问都不要放过，思考、尝试之后，不管有没有结果，都要Google一下，当然，要做到这一点，需要有点技术，这点技术对于立志于成为编程专家的读者来说是必须要掌握的。如果阅读到这里，你还没有理解，那说明的确是我的读者：零基础。

《辟邪剑谱》和《葵花宝典》这两本盖世武功秘籍，对练功者提出了一个较高的要求：欲练神功，挥刀自宫。

学Python，如果要达到比较高的水平，其实不用自宫（如果读者真的要以此明志，该行为结果与本书的作者和出版机构无关，纯属个人行为。若读者尚未成年，请在法定监护人的监护下阅读本书，特此声明）。但是，需要在你的计算机中安装一些东西，并且这个过程有时比较耗费时间。

所需要安装的东西，都在这个页面里面：www.python.org/downloads/。

www.python.org是Python的官方网站，如果你的英语非常好，那么阅读该网站，可以获得非常多的收获。

在下载页面里面，显示出Python目前有两大版本：Python 3.x.x和Python 2.7.x。可以说，Python 3是未来，它比Python 2.7有进步。但是，现在还有一些东西没有完全兼容Python 3，所以，本书在讲解中以Python 2.7为主，兼顾Python 3.x。一般而言，如果学了Python 2.7，再学习Python 3就很容易，因为两者只是某些地方的变化。请读者不要纠结于是学习Python 2还是学习Python 3这种无聊的问题，如果两个差别达到让你学了而这个无法使用那个，这将是Python的灾难。绝顶聪明的Python创造者和维护者们，绝对不会允许这样的事情出现。

### 0.3.1 Ubuntu系统

你的计算机是什么操作系统的？如果是Linux的某个发行版，比如Ubuntu，那么就跟我同道了。如果是iOS，也一样，因为都是UNIX麾下的，与在Ubuntu中的安装过程大同小异。只是Widows有点另类了。

但没关系，Python就是跨平台的，它可以在任何系统下进行编程，用它写出来的程序也可以运行在任何操作系统中。

但是，我个人推荐使用Linux/UNIX系列的操作系统。

正常情况下，只要安装了Ubuntu这个操作系统，就已经安装好了Python的编程环境。你只需要打开Shell，然后输入Python，单击“回车”之后就会看到如下内容：

```
$ python
Python 2.7.6 (default, Nov 13 2013, 19:24:16)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

如果没有该编程环境，就需要安装了，一种最简单的方法：

```
$ sudo apt-get install python
```

还有另外一种比较麻烦的方法，但似乎能炫耀一下自己的水平，方法如下。

（1）到官方网站下载源码。比如：

```
wget http://www.python.org/ftp/python/2.7.8/Python-2.7.8.tgz
```

（2）解压源码包

```
tar -zxvf Python-2.7.8.tgz
```

（3）编译

```
cd Python-2.7.8 ./configure --prefix=/usr/local
make&&sudo make install
```

这里指定了安装目录/usr/local。如果不指定，可以使用默认的，直接运行./configure即可。

安装好之后，进入Shell，输入Python，就会看到结果。我们将那个带有“>>>”的界面称之为“Python的交互模式”。

### 0.3.2 Windows系统

到下载页面里找到Windows安装包下载，比如下载了这个文件：python-2.7.8.msi。然后完成安装。

特别注意，安装完之后，需要检查系统环境变量是否有Python，如果没有，就设置一下。

以上搞定后，在cmd中，输入“python”，得到与前面类似的结果，就说明已经安装好了。

### 0.3.3 Mac OS X系统

其实根本就不用再写怎么安装Mac OS X系统了，因为用Mac OS X的朋友，肯定是高手中的高高手了，至少我一直很敬佩那些用Mac OS X并坚持没有更换为Windows的人。

所幸的是用苹果电脑的就不用安装了，因为里面一定预装好了，拿过来就可以直接用。

如果按照以上方法顺利安装成功，只能说明幸运。如果没有安装成功，则是提高自己的绝佳机会，因为只有遇到问题才能解决问题，才能知道更深刻的道理，不要怕，使用Google，它能帮助你解决问题。当然，也可以加入QQ群或者通过微博问我。

## 0.4 集成开发环境（IDE）

安装好Python之后，就可以进行开发了。按照惯例，第一行代码总是：Hello World。

### 0.4.1 值得纪念的时刻：Hello world

不管你使用的是什么操作系统，总之肯定能够找到一个地方运行Python，进入到交互模式，像下面一样：

```
Python 2.7.6 (default, Nov 13 2013, 19:24:16)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

在“>>>”后面输入print"Hello，World"，并按回车键，下面是见证奇迹的时刻。

```
>>> print "Hello, World"
Hello, World
```

如果你从来不懂编程，那么从这一刻起，就跨入了程序员行列；如果已经是程序员，那么就温习一下当初的惊喜吧！

“Hello，World”是你用代码向这个世界打招呼了。

每个程序员，都曾经经历过这个伟大时刻，不经历这个伟大时刻的程序员不是伟大的程序员。为了纪念这个伟大时刻，理解其伟大之所在，下面执行分解动作：

说明：在下面的分解动作中，用到了符号“#”，这个符号在Python编程中表示注释。所谓注释，就是在计算机中不执行那句话，只是为了说明某行语句表达什么意思，是给计算机前面的人看的。特别提醒，在编程实践中，注释是必须的，尽管很多人要求代码要具有可读性，但必要的注释也是少不了的。请牢记：程序在大多数情况下是给人看的，只是偶尔让计算机执行一下。

#看到“>>>”符号，表示Python做好了准备，等待你向它发出指令，让它做事情。

```
>>>
```

#print，意思是打印。在这里也是这个意思，是要求Python打印东西。

```
>>> print
```

#"Hello,World"是打印的内容，注意，双引号是英文状态下的。引号不是打印内容，它相当于一个包裹，把打印的内容包起来，统一交给Python。

```
>>> print "Hello, World"
```

#上面命令执行的结果。Python接收到你要求它所做的事情：打印Hello,World，于是它就老老实实地执行这个命令，丝毫不走样。

```
Hello, World
```

在Python中，如果进入了上面的样式，就是进入了“交互模式”。这是非常有用而且简单的模式，便于我们进行各种学习和探索，随着学习的深入，你将更加觉得它魅力四射。

笑一笑：有一个程序员，感觉自己书法太烂了，于是立志继承光荣文化传统，购买了笔墨纸砚。在某天开始练字，将纸铺好，拿起笔蘸足墨水，挥毫在纸上写下了两个大字：Hello World。

虽然进入了程序员序列，但是，如果程序员用这个工具仅仅是打印“Hello，World”，又怎能用“伟大”来形容呢？况且这个工具也太简陋了。你看美工妹妹用的Photoshop，行政妹妹用的Word，出纳妹妹用的Excel，就连坐在老板桌后面的那个家伙也在用PPT播放自己都不相信的新理念呢，难道我们伟大的程序员，就用这么简陋的工具来写旷世代码吗？

当然不是。软件是谁开发的？程序员。程序员肯定会先为自己打造好用的工具，这也叫作“近水楼台先得月”。

IDE就是程序员的工具。

### 0.4.2 集成开发环境概述

IDE的全称是：Integrated Development Environment，简称IDE，也称为Integration Design Environment或Integration Debugging Environment，翻译成中文叫作“集成开发环境”，它是一种辅助程序员开发用的应用软件。

维基百科这样对IDE定义：

IDE通常包括程序语言编辑器、自动建立工具和除错器。有些IDE包含编译程序和直译器，如微软的Microsoft Visual Studio，有些则不包含，如Eclipse、SharpDevelop等，这些IDE是通过调用第三方编译器来实现代码的编译工作的。有时IDE还会包含版本控制系统和一些可以设计图形用户界面的工具。许多支持面向对象的现代化IDE还包括类别浏览器、对象查看器、对象结构图。虽然目前有一些IDE支持多种程序语言（例如Eclipse、NetBeans、Microsoft Visual Studio），但是一般而言，主要还是针对特定的程序语言而量身打造（例如Visual Basic）。

如图0-3所示是微软提供的名字叫作Microsoft Visual Studio的IDE，用C#进行编程的程序员都用它。

图0-3 名叫Microsoft Visual Studio的IDE

如图0-4所示是在苹果电脑中出现的名叫XCode的IDE。

要想了解更多IDE的信息，推荐阅读维基百科中的词条。

- 英文词条：Integrated development environment
- 中文词条：集成开发环境

图0-4 名叫XCode的IDE

### 0.4.3 Python的IDE

用Google搜索一下：Python IDE，会发现能够进行Python编程的IDE还真不少。东西一多就容易无所适从，所以有不少人都问用哪个IDE好。可以看看下面链接里的内容：http://stackoverflow.com/questions/81584/what-ide-to-use-for-python。

顺便推荐一个非常好的与开发相关的网站：stackoverflow.com。在这里可以提问，可以查看答案。如果有问题，一般先在这里查找，大多数情况都能找到非常满意的结果，且有很大启发。

那么作为零基础的学习者，用哪个IDE好呢？既然是零基础，就别瞎折腾了，就用Python自带的IDLE，原因就是：简单，虽然它比较简陋。

在Windows中，通过“开始”菜单→“所有程序”→“Python 2.x”→“IDLE（Python GUI）”来启动IDLE。启动之后，看到如图0-5所示的界面。



图0-5　IDLE界面

注意：若看到的界面显示版本与这个图不同，则说明安装的版本不同，但大致模样差不多。

其他操作系统的用户，也都能在启动IDLE这个程序之后，出现与上面一样的图。

这里其实是Python的交互模式编程环境。

当然还有一个文本环境，读者可以在File菜单中新建一个文件，即进入文本编辑环境。

除了这个自带的IDE，还有很多其他的IDE，列出来，供喜欢折腾的朋友参考。

- PythonWin:是Python Win32 Extensions（半官方性质的Python for win32增强包）的一部分，也包含在ActivePython的Windows发行版中。如其名字所言，只针对Win32平台。
- MacPython IDE：MacPythonIDE是Python的Mac OS发行版内置的IDE，可以看作是PythonWin的Mac对应版本，由Guido的哥哥Just van Rossum编写。
- Emacs和Vim：Emacs和Vim号称是这个星球上最强大的文本编辑器，对于许多程序员来说是万能IDE的不二选择。
- Eclipse+PyDev：Eclipse是新一代的优秀泛用型IDE，虽然是基于Java技术开发的，但出色的架构使其具有不逊于Emacs和Vim的可扩展性，现在已经成为了许多程序员最爱的瑞士军刀。

磨刀不误砍柴工，IDE已经有了，伟大程序员就要开始从事伟大的编程工作了。