“十三五”普通高等教育规划教材

Python 基础编程与实践

朱旭振　黄赛　编著

提供电子课件、习题答案
http://www.cmpedu.com

机械工业出版社
CHINA MACHINE PRESS

“十三五”普通高等教育规划教材

Python 基础编程与实践

朱旭振　黄　赛　编著

机械工业出版社

本书从教学和工程应用角度出发，首先，介绍了 Python 语言的历史背景、一般编程方法、Python 程序的常见设计方法；其次，介绍了 Python 编程基础知识，包括基本语法、控制结构、输入输出、数据结构以及 Matplotlib 作图，进而分别讨论了面向过程编程的概念及方法和面向对象编程的概念及方法，并引入了 GUI 编程方法，包括简单的图形控件介绍、布局管理器以及 GUI 程序结构；最后，给出了 Python 的重要资源、常用的 ASCII 码表和 Python 关键字表。

本书是数据分析领域 Python 编程的基础教材，可用于高校相关课程教师的教学用书，也可作为高校本科生、研究生的基础学习用书。企业开发人员和数据分析人员，也可以将本书作为“工作参考手册”来阅读。对于需要进行数据分析、算法建模、机器学习等数据科学研究的人员，特别是需要进行大数据学科建设的高校，本教材很适用。

为了教师和工程技术人员教学和培养的需要，本书免费提供电子课件和习题答案。欢迎使用本书作为教材的教师登录 www.cmpedu.com 免费注册、审核后下载，或联系编辑索取（QQ：2446305805，电话 010-88379753）。

图书在版编目（CIP）数据

Python 基础编程与实践 / 朱旭振，黄赛编著. —北京：机械工业出版社，2019.2
“十三五”普通高等教育规划教材

ISBN 978-7-111-62027-3

Ⅰ. ①P… Ⅱ. ①朱… ②黄… Ⅲ. ①软件工具－程序设计－高等学校－教材
Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2019）第 029695 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
责任编辑：时　静　秦　菲　　责任校对：张艳霞
责任印制：郜　敏
北京圣夫亚美印刷有限公司印刷
2019 年 3 月第 1 版 • 第 1 次印刷
184mm×260mm • 19.25 印张 • 470 千字
0001－3000 册
标准书号：ISBN 978-7-111-62027-3
定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：（010）88379833
读者购书热线：（010）88379649

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
教育服务网：www.cmpedu.com
金　书　网：www.golden-book.com

前　言

本书是根据教育部计算机基础课程教学指导委员会发布的计算机基础课程教学基本要求，结合大学理工科教学的特点，立足于编程语言发展趋势并顺应时代潮流的情况下编写的大学生计算机基础新教材。

本书内容主要围绕 Python 编程语言的基础展开介绍，包括但不限于 Python 语言的特点、环境搭建、自顶向下的程序设计思想、Python 基础数据及类型、控制流结构、函数与模块、画图、Python 文本处理、面向对象编程、异常处理以及数据库编程等。

工程技术人员通过对本书的学习可以具备对 Python 编程语言的掌握能力，通过本书学习并结合课后练习，可以熟练使用 Python 语言编码并与计算机进行交流。本书同样适合初入编程领域的编程爱好者，理工、经管类大学生以及需要了解 Python，利用 Python 进行项目设计、数值分析、统计预测等的各领域工程技术人员使用。除此之外，Python 因其“优雅、明确、简单”的设计哲学，非常适合初识编程的新手学习。Python 作为面向对象的编程语言，可作为各类编程语言间的万能胶水，适合作为各类编程语言的“大总管”，极大简化了不同编程语言的兼容性难题。

本书从 Python 语言教学的全局出发，以培养学生使用 Python 语言进行编程的能力为目的，内容介绍力求清楚、明确。从基本概念、基本语法出发，结合大量例题进行概念和语法解析，每章均有实践问题和大量课后习题提供给读者练习使用。

本书由朱旭振、黄赛编写，同时还有卢德鹏、姜南、柴録、于慧、郑丹旸等对于本书的完成给予了帮助。此外，陆高锋、戴蕊、严正行等参与了本书的校验工作。

由于时间、人员等问题，本书在总体结构、内容、叙述、实例、题目等方面难免有偏颇与疏漏之处，欢迎广大读者提出宝贵意见，敬请批评指正。

编者

目　录

前言
第 1 章　Python 概述……1
1.1　了解 Python……1
1.1.1　Python 的由来……1
1.1.2　Python 的版本……2
1.1.3　Python 的应用领域……2
1.2　Python 语言的特点……3
1.3　一个简单的 Python 程序……5
1.4　搭建 Python 开发环境……6
1.4.1　Python 下载与安装……6
1.4.2　在 Linux 和 UNIX 系统中安装 Python……10
1.4.3　在 Mac OS 中安装 Python……11
1.4.4　交互式 IDLE 的使用……12
1.5　Python 开发工具……12
1.5.1　Anaconda 介绍……12
1.5.2　PyCharm 的使用……16
1.5.3　Eclipse IDE 的使用……17
1.6　Python 跨平台……18
1.6.1　Jython……19
1.6.2　IronPython……19
1.7　小结……19
实践问题 1……19
习题 1……19
参考文献……20
第 2 章　程序设计算法……21
2.1　算法的概念……21
2.1.1　算法的定义……21
2.1.2　算法的特征……21
2.1.3　算法的评价……22
2.2　算法的表示……22
2.2.1　用自然语言表示算法……23
2.2.2　用流程图表示算法……23
2.2.3　用 N-S 图表示算法……25
2.2.4　用伪代码表示算法……26
2.2.5　用计算机语言表示算法……27
2.3　结构化程序设计方法……27
2.3.1　自顶向下的程序设计……27
2.3.2　结构化编程……29
2.4　程序错误……30
2.4.1　语法错误……30
2.4.2　运行错误……32
2.4.3　逻辑错误……32
2.5　简单算法举例……33
2.6　小结……36
实践问题 2……36
习题 2……36
参考文献……38
第 3 章　Python 基础数据……39
3.1　Python 编码规范……40
3.1.1　命名规则……40
3.1.2　代码缩进与冒号……41
3.1.3　模块导入语法……43
3.1.4　空行分割代码……43
3.1.5　注释和续行……44
3.1.6　语句分割……45
3.2　变量和常量……46
3.2.1　变量命名……46
3.2.2　变量赋值……47
3.2.3　局部变量……48
3.2.4　全局变量……49
3.2.5　常量……50
3.2.6　关键字……53
3.3　基本输入输出……54
3.3.1　获取用户输入……54
3.3.2　基本输出……55
3.4　数值……56
3.4.1　基本数值：整型和浮点型……56
3.4.2　算术运算符……57

3.4.3 数值变量……59
3.4.4 括号与优先级……59
3.4.5 内存中的数字对象……60
3.4.6 常见的数值函数……61
3.5 字符串……62
3.5.1 字符和字符串……62
3.5.2 字符串字面值……63
3.5.3 索引和切片……65
3.5.4 反向索引……67
3.5.5 切片的默认边界……68
3.5.6 索引和切片越界……69
3.5.7 字符串拼接……69
3.5.8 常见字符串函数……70
3.5.9 格式化数字和字符串……77
3.5.10 正则表达式……82
3.5.11 使用 re 模块实现正则表达式……85
3.6 列表和元组……91
3.6.1 通用序列操作……92
3.6.2 列表……94
3.6.3 元组……100
3.6.4 列表与元组的区别……104
3.7 集合……104
3.7.1 集合的创建……105
3.7.2 集合的添加和删除……105
3.7.3 集合推导……106
3.7.4 集合运算……106
3.8 字典……107
3.8.1 字典的创建……107
3.8.2 字典的访问……109
3.8.3 字典的方法……110
3.9 小结……111
实践问题 3……114
习题 3……120
参考文献……126
第 4 章 Python 控制流结构……127
4.1 关系和逻辑运算……127
4.1.1 关系运算符……127
4.1.2 逻辑运算符……129
4.1.3 短路求值……130
4.2 布尔数据类型……130
4.3 简化条件……132
4.4 条件判断语句……132
4.4.1 if 语句……132
4.4.2 if-else 语句……133
4.4.3 多重条件判断 elif 语句……135
4.4.4 条件判断嵌套……137
4.4.5 绝对真和假……138
4.5 循环控制语句……139
4.5.1 while 循环控制语句……139
4.5.2 for 循环控制语句……142
4.5.3 range 函数……145
4.5.4 循环嵌套……146
4.5.5 break 和 continue 语句……147
4.5.6 pass 语句……149
4.5.7 无限循环……150
4.5.8 字符串中字符的循环遍历……151
4.6 小结……151
实践问题 4……151
习题 4……152
参考文献……156
第 5 章 Python 函数与模块……157
5.1 函数定义……157
5.1.1 内建函数……158
5.1.2 用户自定义函数……161
5.1.3 向函数传值……162
5.1.4 函数返回值……162
5.1.5 变量作用域……164
5.2 函数调用……165
5.2.1 调用其他函数……165
5.2.2 函数返回多值……167
5.2.3 基于函数的列表解析……168
5.2.4 函数调用中的默认参数……169
5.2.5 按参数名向函数传值……171
5.3 特殊函数……172
5.3.1 函数嵌套……172
5.3.2 递归函数……173
5.3.3 Sorted 函数……175
5.3.4 Lambda 函数……176

5.3.5 Generator 函数 ····· 177
5.3.6 随机数函数 ····· 179
5.4 模块 ····· 180
5.4.1 模块的创建 ····· 181
5.4.2 模块的导入 ····· 181
5.4.3 模块的属性 ····· 183
5.4.4 模块的内置函数 ····· 185
5.4.5 自定义包 ····· 186
5.5 小结 ····· 188
实践问题 5 ····· 189
习题 5 ····· 189
参考文献 ····· 193
第 6 章 Python 画图 ····· 194
6.1 科学画图 Matplotlib 模块 ····· 194
6.1.1 Matplotlib 画图 ····· 194
6.1.2 修改图属性 ····· 198
6.2 海龟图 ····· 205
6.2.1 坐标 ····· 205
6.2.2 turtle 模块中的基本方法 ····· 206
6.2.3 简单图形 ····· 207
6.2.4 折线图 ····· 208
6.2.5 柱状图 ····· 210
6.3 小结 ····· 212
实践问题 6 ····· 212
习题 6 ····· 213
参考文献 ····· 215
第 7 章 Python 文件处理 ····· 216
7.1 文本文件处理 ····· 216
7.1.1 读取文本文件 ····· 216
7.1.2 创建文本文件 ····· 220
7.1.3 向旧文本中添加新文本 ····· 221
7.1.4 修改文本文件内容 ····· 222
7.1.5 使用基本文件方法 ····· 223
7.2 数据处理 ····· 224
7.2.1 CSV 文件 ····· 225
7.2.2 访问 CSV 文件的数据 ····· 225
7.2.3 使用列表分析 CSV 文件中的数据 ····· 227
7.2.4 Excel 和 CSV 文件 ····· 228
7.3 小结 ····· 229
实践问题 7 ····· 229
习题 7 ····· 230
参考文献 ····· 231
第 8 章 面向对象编程 ····· 232
8.1 面向对象简介 ····· 232
8.2 类与对象 ····· 233
8.2.1 类与对象的关系 ····· 233
8.2.2 类的定义 ····· 233
8.2.3 对象的创建 ····· 234
8.3 属性与方法 ····· 235
8.3.1 类的属性 ····· 235
8.3.2 类的方法 ····· 237
8.3.3 构造函数 ····· 239
8.3.4 析构函数 ····· 240
8.3.5 垃圾回收 ····· 240
8.3.6 类的内置方法 ····· 241
8.3.7 类方法的动态绑定 ····· 241
8.4 继承 ····· 243
8.4.1 继承的使用 ····· 243
8.4.2 基类的抽象 ····· 244
8.4.3 多态 ····· 247
8.4.4 多重继承 ····· 250
8.4.5 混合继承 ····· 252
8.5 操作符重载 ····· 253
8.6 小结 ····· 257
实践问题 8 ····· 258
习题 8 ····· 258
参考文献 ····· 259
第 9 章 Python 异常处理 ····· 260
9.1 Python 中的异常 ····· 260
9.2 try-except 结构 ····· 261
9.3 finally 结构 ····· 264
9.4 异常抛出 ····· 265
9.5 自定义异常 ····· 267
9.6 断言语句 ····· 268
9.7 小结 ····· 269
实践问题 9 ····· 270
习题 9 ····· 271

参考文献 ……272
第 10 章　Python 用户图形界面编程 ……273
10.1　控件 ……274
10.1.1　图形用户界面简介 ……274
10.1.2　按钮控件 ……275
10.1.3　标签控件 ……277
10.1.4　输入控件 ……278
10.1.5　列表框控件 ……280
10.1.6　滚动条控件 ……283
10.2　网格布局管理器 ……284
10.2.1　网格 ……284
10.2.2　粘属性 ……286
10.2.3　向列表框添加滚动条 ……288
10.2.4　设计窗口布 ……289
10.3　编写 GUI 程序 ……290
10.3.1　简单 GUI 程序 ……290
10.3.2　将文件加载到列表框 ……292
10.3.3　面向对象编写 GUI 程序 ……294
10.4　小结 ……295
实践问题 10 ……296
习题 10 ……296
参考文献 ……296
附录 ……297
附录 A　ASCII 码表 ……297
附录 B　Python 保留字 ……298
附录 C　Python 学习资源 ……298

第 1 章　Python 概述

本章主要介绍 Python 语言的由来、版本、应用以及一些简单的 Python 语言程序。此外，还将介绍 Python 在不同操作系统的环境搭建，演示如何安装和使用一些常用的 Python 开发工具，如 Anaconda、PyCharm 等，为后续学习本书做好准备。

本章知识点：

- Python 的由来、版本及应用领域
- Python 语言的特点
- 一个简单的 Python 程序
- Python 在 Windows、Linux 及 Mac OS 系统的安装
- Anaconda、PyCharm 等 Python 开发工具的使用
- Python 跨平台

1.1　了解 Python

1.1.1　Python 的由来

Python 的创始人为 Guido van Rossum，曾获得阿姆斯特丹大学数学和计算机双料硕士学位。Guido 接触并使用过 Pascal、C、Fortran 等语言，但这些语言都不能让他感到满意。这些语言的基本设计原则是让机器运行得更快，这就要求程序员需要模仿计算机的思考模式来写出更符合机器口味的程序。这种编程方式让 Guido 感到苦恼，所以 Guido 尝试选择 Shell，C 语言中许多上百行的程序在 Shell 中只用几行就可以完成。然而，Shell 的本质是调用命令，它不是一个真正的语言，无法全面调动计算机的功能。

随后，Guido 在荷兰的数学和计算机研究所参与了 ABC 语言的开发，希望开发一种既能够像 C 语言一样全面调用计算机的功能接口，又可以像 Shell 一样轻松编程的语言。与当时大部分语言不同，ABC 语言以教学为目的，目标是“让用户感觉更好”，希望通过 ABC 语言让语言变得更容易阅读、容易使用、容易记忆、容易学习，并以此激发人们学习编程的兴趣。ABC 语言尽管已经具备了良好的可读性和易用性，但是由于其对计算机配置的高要求，导致其始终没有流行起来。除此之外，ABC 语言不能直接操作文件系统，无法直接读写文件，输入输出的困难对于计算机语言来说是致命的。

1989 年的圣诞假期，Guido 开始写 Python 语言的编译器。Python 这个名字来自于 Guido 所挚爱的电视剧——Monty Python's Flying Circus。他希望创造一种介于 C 和 Shell 之间，功能全面、易学易用、可拓展的语言。1991 年，第一个 Python 编译器诞生。该编译器

是用 C 语言实现的，并且能够调用 C 语言的库文件。Python 诞生时便具有类、函数、异常处理、包含表和词典在内的核心数据类型以及模块为基础的拓展系统[1]。

最初，Python 完全由 Guido 本人开发，后来逐渐受到 Guido 同事的欢迎，它们迅速反馈使用意见，并参与 Python 的改进。Guido 和一些同事构成了 Python 的核心团队，他们将自己大部分业余时间用于 hack Python，Python 逐渐拓展到了研究所外。Python 将许多机器层面的细节隐藏交给编译器处理，并凸显逻辑层面的编程思考，因此，程序员使用 Python 时可以将更多时间用于程序逻辑的思考，而不是具体细节的实现。这一特征吸引了广大程序员，使得 Python 越来越受到人们的关注。

1.1.2 Python 的版本

Python 自发布以来，主要有三个版本：1994 年发布的 Python1.0 版本、2000 年发布的 Python2.0 版本和 2008 年发布的 Python3.0 版本。

虽然目前使用 Python2.x 的开发者略多于使用 Python3.x 的开发者，但使用 Python3.x 在未来将是大势所趋。Python3.x 在 Python2.x 的基础上做了功能升级，对 Python2.x 的标准库进行了一定程度的重新拆分和整合，比 Python2.x 更容易理解。特别是在字符编码方面，Python2.x 对于中文字符串的支持性能不够好，需要编写单独的代码对中文进行处理，否则不能正确显示中文，而 Python3.x 成功解决了该问题。

此外，Python3.x 和 Python2.x 的思想基本是共通的，只有少量的语法差别。学会了 Python3.x，只要稍微花一点时间学习 Python2.x 的语法，就可以灵活运用这两个不同版本了。

本书使用的版本为 Python3.6。

1.1.3 Python 的应用领域

Python 作为一种功能强大且通用的编程语言广受好评。它具有非常清晰的语法特点，适用于多种操作系统，目前在国际上非常流行，正在得到越来越多的应用。

1．数据分析与处理

通常情况下，Python 被用来做数据分析。用 C 语言设计一些底层算法进而封装，然后用 Python 进行调用。因为算法模块较为固定，所以用 Python 直接进行调用，方便且灵活，可以根据数据分析与统计需要灵活使用。Python 也是一个比较完善的数据分析生态系统，其中 Matplotlib 经常会被用来绘制数据图表，它是一个 2D 绘图工具，有着良好的跨平台交互特性。日常做描述统计用到的直方图、散点图、条形图等都会用到它，几行代码即可出图。人们日常看到的 K 线图、月线图也可用 Matplotlib 绘制。如果在证券行业做数据分析，Python 是必不可少的。

再如 Pandas 也是 Python 在做数据分析时常用的数据分析包，也是很好用的开源工具。Pandas 可对较为复杂的二维或三维数组进行计算，同时还可以处理关系型数据库中的数据，和 R 语言相比，data.frame 计算的范围要远远小于 Pandas 中的 DataFrame 的范围，这也从另一个侧面说明 Python 的数据分析功能要强于 R 语言。

除以上两点之外，SciPy 还可以解决很多科学计算的问题，比如微分方程、矩阵解析、概率分布等数学问题。

2．Web 开发应用

Python 是 Web 开发的主流语言，但不能说是最好的语言。同样是解释型语言的 JavaScript，在 Web 开发中应用得已经较为广泛，原因是其有一套成熟的框架。但 Python 也具有独特的优势，比如 Python 相比于 JS、PHP 在语言层面较为完备，而且对于同一个开发需求能够提供多种方案。库的内容丰富，使用方便。Python 在 Web 方面也有自己的框架，如 django 和 flask 等。可以说用 Python 开发的 Web 项目小而精，支持最新的 XML 技术，而且数据处理的功能较为强大。

3．人工智能应用

在人工智能的应用方面，笔者认为还是得益于 Python 强大而丰富的库以及数据分析能力。比如说在神经网络、深度学习方面，Python 都能够找到比较成熟的库包来加以调用。而且 Python 是面向对象的动态语言，且适用于科学计算，这就使得 Python 在人工智能方面备受青睐。虽然人工智能程序不限于 Python，但依旧为 Python 提供了大量的 API，这也正是因为 Python 当中包含着较多的适用于人工智能的模块，比如 sklearn 模块等。调用方便、科学计算功能强大依旧是 Python 在 AI 领域最强大的竞争力。

4．游戏编程

Python 在很早的时候就是一种游戏编程的辅助工具，在《星球大战》中扮演了重要的角色。在《深渊（The Abyss）》《星际迷航（Star Trek）》《夺宝奇兵（Indiana Jones）》等大片中担当特技和动画制作的工业光魔（Industrial Light）公司就采用 Python 制作商业动画。目前，通过 Python 完全可以编写出非常棒的游戏程序。

5．企业与政务应用

目前，Python 已经成功实现企业级应用，全球已有很多公司采用 Python 进行企业级软件的开发和应用，比如：Enterprise Resource Planning（ERP，企业资源计划）和 Customer Relationship Management（CRM，客户关系管理）这样的应用。同时，通过 Python 技术，人们成功实现了许多政务应用。

1.2 Python 语言的特点

1．面向对象

Python 既支持面向过程的函数编程也支持面向对象的抽象编程。在面向过程的语言中，程序是由过程或可重用代码的函数构建起来的。在面向对象的语言中，程序是由数据和功能组合而成的对象构建起来的。与其他主要语言如 C++和 Java 相比，Python 以一种非常强大又简单的方式实现面向对象编程，使得编程更加灵活[2]。

2．内置数据结构

数据结构由相互之间存在一种或多种关系的数据元素以及元素之间的关系组成。Python 本身自带的数据结构包括列表、元组、字符串、字节、字节数组、集合、字典 7 种内置数据结构，且它们都是可迭代对象。

3．简单易学

Python 的语法简单优雅，甚至没有像其他语言的大括号、分号等特殊符号，代表了一种极简主义的设计思想。同时，Python 内置多种高级数据结构，实现了列表、元组、字典

和集合等高级数据结构，这些结构在传统 C、Java 等语言中需要用户自定义结构。Python 非常适合阅读，并且容易理解。此外，Python 虽然基于 C 语言编写，但是摒弃了 C 中非常复杂的指针，简化了 Python 语法。

4．语言健壮

Python 提供了异常处理机制，能捕获程序的异常情况。此外 Python 的堆栈跟踪对象能够指出程序出错的位置和出错的原因。异常机制能够避免不安全退出的情况，同时能够帮助程序员调试程序。

5．可移植性

Python 的开源本质使得它已经被移植在许多平台上（经过改动使它能够在不同平台上工作）。如果在编写程序时避免使用依赖于系统的特性，那么这些 Python 程序无需修改就可以在下述任何平台上面运行。这些平台包括 Linux、Windows、FreeBSD、Macintosh、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE 甚至还有 PocketPC、Symbian 以及 Google 基于 Linux 开发的 Android 平台。

6．易扩展性

Python 出于一种自由的设计思想，没有抽象类，也没有其他语言里 private、public、protect 这些设定，但在 Python 中同样也可以通过封装实现私有、公有、抽象这些设定。假如让所有默认接口 raise 异常，那么这个类就在一定意义上成为了抽象类。虽然抽象类的适用范围很广，但是并不是任何情况下都优于非抽象类，于是 Python 让使用者自己选择是否使用抽象类。

当然，这只是 Python 内在的一个细节，实际上 Python 的可扩展性不仅仅表现在对内的设计思想上，还表现在对外不同语言之间的配合使用效果上。例如在游戏开发中，游戏的服务端可以用 C 作为底层游戏引擎，Python 作为逻辑脚本，这样可以非常方便地调用 C 编写的引擎接口，仿佛 C 语言的底层不存在一样。

7．动态性

Python 的动态性和多态性是 Python 语言简洁灵活的基础。在 Python 中，类型是在运行过程中自动决定的，而不是通过代码声明。这意味着在 Python 中没有必要事先声明变量，变量名没有类型，类型属于对象而不是变量名。从另一方面讲，对象知道自己的类型，即每个对象都包含了一个头部信息，这一头部信息标记了这个对象的类型。Python 语言的动态性优化了人的时间而不是机器的时间，可以大幅提高程序员的生产力。

8．解释型

大多数计算机编程语言都是编译型语言，在运行之前需要将源代码编译为操作系统可以执行的二进制格式（0110 格式的），这样大型项目编译过程非常消耗时间。而 Python 程序不需要编译成二进制代码，可以直接从源代码运行程序。在计算机内部，Python 解释器把源代码转换成字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。事实上，由于不再需要担心如何编译程序，如何确保连接转载正确的库等，这使得 Python 使用变得更加简单。

9．应用广泛

Python 目前广泛应用在多个领域，具体如下。

（1）游戏编程：可以在 Pygame 系统中使用 Python 对图形和游戏进行编程。

（2）串口通信：PySerial 扩展在 Windows、Linux 及更多系统上进行串口通信，系统是由相互联系、相互作用的若干要素按一定的规则组成并具有一定功能的整体。

（3）图像处理：用 PIL、PyOpenGL、Blender、Maya 和一些其他工具进行图像处理管理。

（4）机器人控制：用 PyRo 工具包进行机器人控制编程。

（5）人工智能：使用神经网络仿真器和专业的系统 Shell 进行 AI 编程图像处理；用 PIL、PyOpenGL、Blender、Maya 和一些其他工具进行图像处理。

（6）自然语言分析：使用 NLTK 包进行自然语言分析图像处理。

1.3 一个简单的 Python 程序

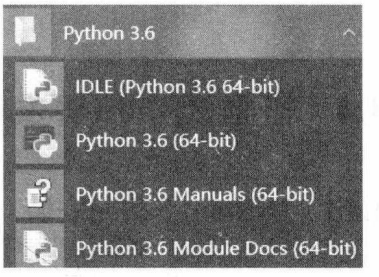

图 1-1 Python 菜单

安装好 Python 后，在“开始”菜单栏中会自动添加一个 Python3.6 文件夹，单击该文件夹会出现图 1-1 所示的子目录。Python 菜单如图 1-1 所示。

Python 目录下有 4 个子目录，从上到下依次是 IDLE、Python3.6、Python3.6 Manuals 和 Python3.6 Module Docs，分别具有如下功能。

- IDLE 是 Python 集成开发环境，也称交互模式，具备基本的 IDE 功能，是非商业 Python 开发的不错选择。
- Python3.6 是 Python 的命令控制台，窗口跟 Windows 下的命令窗口一样，不过只能执行 Python 命令。
- Python3.6 Manuals 是帮助文档，单击后会弹出全英文的帮助文档。
- Python3.6 Module Docs 是模块文档，单击后会跳转到一个可以查看目前集成模块的网址。

下面开始在 IDLE 中编辑第一个 Python 程序，首先打开 IDLE，如图 1-2 所示。

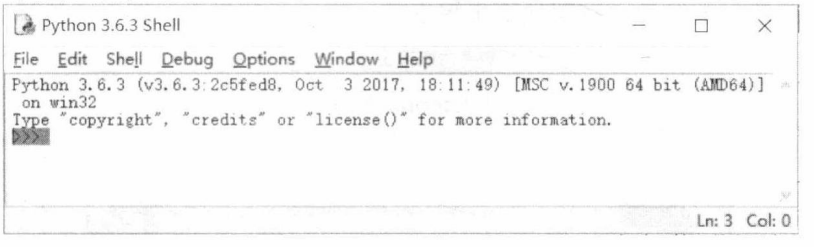

图 1-2 PythonIDLE

“>>>”表示此时在 IDLE 中输入 Python 代码只能立刻执行。

首先输入：

```
print('Hello,world!')
```

按〈Enter〉键后，就可以看到输出了“Hello,world!”，如图 1-3 所示，成功完成了一

个简单的 Python 语言程序。此处 print 后面带了括号，表示 print 是一个函数，单引号里面的叫字符串。如果要让 Python 打印指定的文字，就可以用 print()函数。把待打印的文字用单引号或双引号括起来，但注意单引号和双引号不能混用。

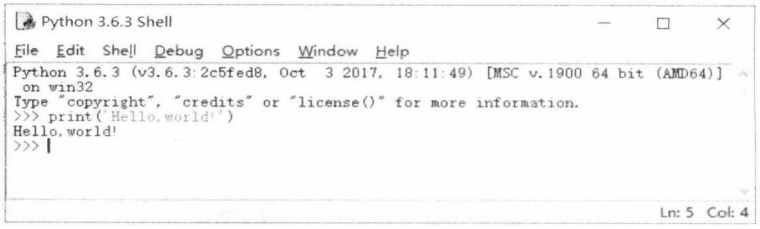

图 1-3　Python 输入输出

1.4　搭建 Python 开发环境

1.4.1　Python 下载与安装

（1）在网站 https://www.python.org/downloads/windows/中找到所需要的 Python 版本进行下载，本次下载版本为 3.6.3。版本信息如图 1-4 所示。

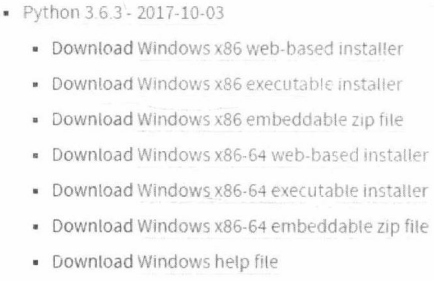

图 1-4　Python3.6.3 版本

（2）双击下载好的软件，按照如下步骤安装。初始安装界面如图 1-5 所示。

图 1-5　Python 安装界面

选择 Customize installation，将 Python 安装到指定目录下。具体安装过程如图 1-6、图 1-7、图 1-8 所示。

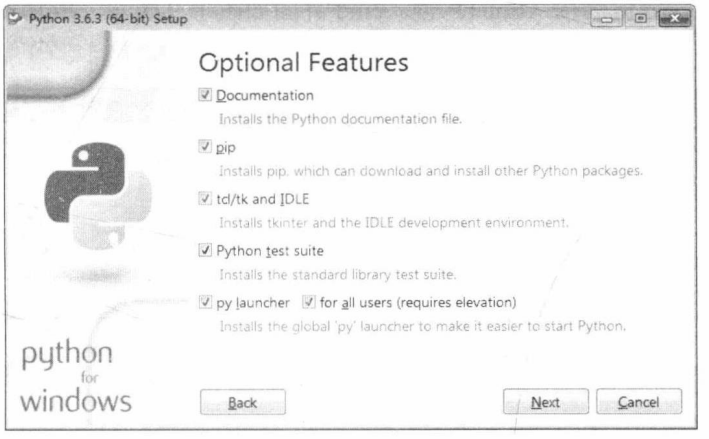

图 1-6　单击“Next”按钮

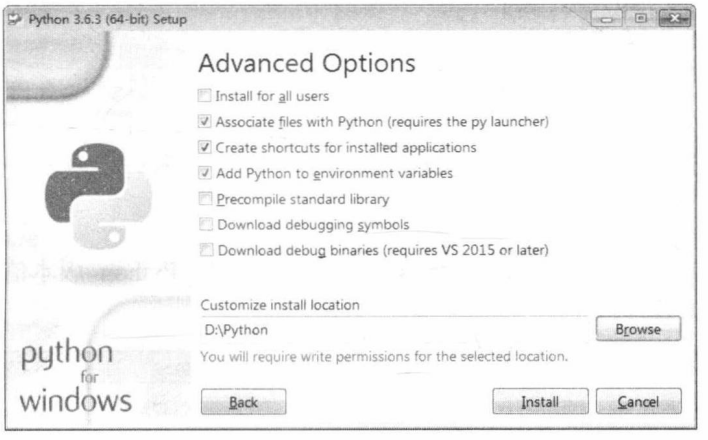

图 1-7　单击“Install”按钮

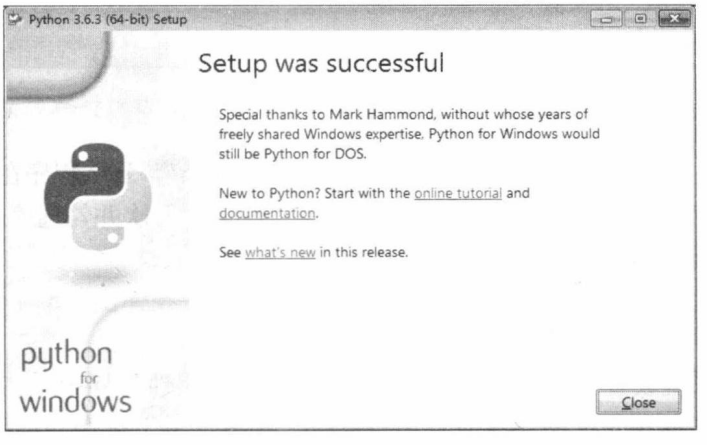

图 1-8　Python 安装完成

（3）查看软件是否能成功运行。

单击“开始”按钮，在输入框中输入“cmd”后按〈Enter〉键，在得到的 cmd 命令界面中输入“python”，完成后按〈Enter〉键。cmd 界面如图 1-9 所示，成功安装界面如图 1-10 所示。

图 1-9　cmd 界面

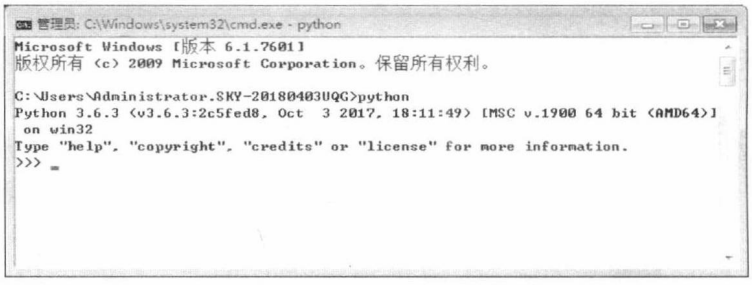

图 1-10　检验 Python 安装

若只想查看版本信息，只输入“python--version”即可。Python 版本信息查询如图 1-11 所示。

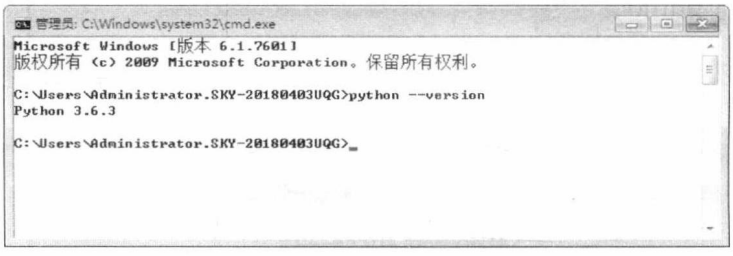

图 1-11　查看版本信息

（4）如果在安装过程中没有勾选“Add Python3.6 to Path”，那么程序在 cmd 命令框中就无法正常运行。解决方法一是把 Python 安装程序重新运行一次并勾选“Add Python3.6 to Path”，二是手动配置环境变量。

（5）手动配置环境变量方法。找到“计算机”并右击，在弹出菜单中单击“属性”，弹出计算机属性界面，如图 1-12 所示。

单击“高级系统设置”，弹出图 1-13 所示的“系统属性”界面。在“系统属性”界面选择“高级”菜单界面，在该界面右下角单击“环境变量”按钮。环境变量如图 1-14 所示。

图 1-12　计算机属性

图 1-13　选择环境变量

图 1-14　环境变量

双击系统变量中的“Path”，弹出“编辑系统变量”界面，在界面的“变量值”输入框中输入 Python 的安装路径（如 D:\Python）。环境变量的配置如图 1-15 所示。

图 1-15　配置环境变量

注意，变量值中的内容可以以英文分号“;”开始，前面有一个分号，并以“\”结尾，如; D:\Python\。单击“确定”保存即添加环境变量成功。接下来重复第（3）步验证即可。