高等学校计算机应用规划教材

Python程序设计

- 安装PyCharm集成开发环境
- 语法基础和流程控制结构
- 列表、元组、字典和集合
- 字符串和正则表达式
- 函数和函数式编程
- 面向对象编程
- 文件操作和异常处理
- Anaconda虚拟环境
- NumPy、Pandas、matplotlib和SciPy扩展库
- scikit-learn扩展库和机器学习

曹仰杰　段鹏松　陈永霞　杨聪　编著

清华大学出版社

高等学校计算机应用规划教材

Python 程序设计

曹仰杰　段鹏松　陈永霞　杨聪　编著

清华大学出版社

北　京

内 容 简 介

本书全面讲述 Python 的基本知识和开发技术。全书分三部分，共 15 章。第一部分基础篇，介绍 Python 的起源和发展、开发工具、语法基础、控制结构、复合数据结构、字符串与正则表达式、函数、类与对象、文件操作、错误与异常等内容；第二部分进阶篇，深入讲解 Python 的虚拟环境 Anaconda、科学计算库 NumPy、数据分析库 Pandas、绘图工具 matplotlib 和数据分析工具 SciPy；第三部分实践篇，主要介绍 Python 在机器学习领域的应用。

本书内容丰富、难度适中、结构清晰、内容翔实，通过三部分以层次递进方式进行讲解，以引导读者循序渐进地学习、掌握并运用 Python。本书可作为普通高等院校计算机、人工智能、大数据科学、物联网等专业 Python 相关课程的教材，也可作为 Python 爱好者的入门级教程。

本书配套的电子课件、实例源文件、习题答案可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以扫描前言中的二维码下载。

图书在版编目(CIP)数据

Python 程序设计/曹仰杰等编著. —北京：清华大学出版社，2019.12
高等学校计算机应用规划教材
ISBN 978-7-302-53925-4

Ⅰ. ①P… Ⅱ. ①曹… 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 224357 号

责任编辑：胡辰浩
装帧设计：孔祥峰
责任校对：牛艳敏
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：三河市金元印装有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**24.25　　**字　　数：**620 千字
版　　次：2019 年 12 月第 1 版　　**印　　次：**2019 年 12 月第 1 次印刷
印　　数：1~3000
定　　价：69.00 元

产品编号：075914-01

前　言

Python 是一种解释型的、面向对象的、带有动态语义的高级编程语言。它由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年。经过二十多年的发展，Python 已经成为最受欢迎的程序设计语言之一。自从 2004 年以后，Python 的使用率呈线性增长。2011 年 1 月，Python 首次被TIOBE编程语言排行榜评为“2010 年年度编程语言”；2019 年 1 月，时隔 8 年后，Python 再度被TIOBE编程语言排行榜评为“2018 年年度编程语言”。此外，在由著名杂志 *IEEE Spectrum* 发布的 “年度编程语言排行榜”上，Python 更是连续获得 2017 年和 2018 年年度冠军。近年来，Python 在数据分析与处理、Web 应用开发、人工智能应用、桌面软件、网络爬虫开发、云计算开发、自动化运维、金融分析、 科学计算以及游戏开发等领域得以广泛应用。

Python 语言具有简洁性、易读性以及可扩展性等特点，受到广大专业编程人士的青睐，一些知名大学已经采用 Python 来教授程序设计课程。近年来，Python 已经成为目前美国顶尖大学里最受欢迎的计算机编程入门语言之一。目前美国计算机排名前 10 的学校里，有 8 所学校使用 Python 作为编程入门语言。在计算机排名前 39 的学校里，有 27 所学校使用 Python 作为编程入门语言。其中，卡耐基梅隆大学的编程基础、麻省理工学院的计算机科学及编程导论就使用 Python 语言讲授。在国内，高校 Python 课程的开设相对滞后，但近年来一些高校也逐渐将 Python 引进课堂。

Python 是一门开源的编程语言，支持命令式编程、函数式编程以及面向对象编程；众多开源的科学计算软件包都提供了 Python 的调用接口，其中包括著名的计算机视觉库OpenCV、三维可视化库 VTK、医学图像处理库 ITK；同时 Python 拥有大量专用的科学计算扩展库，如当前三个十分经典的科学计算扩展库——NumPy、SciPy 和 matplotlib，它们分别为 Python 提供了快速数组处理、数值运算以及绘图功能。因此，Python 语言及其众多的扩展库所构成的开发环境十分适合工程技术人员、科研人员处理实验数据、制作图表，甚至开发科学计算应用程序。

学习 Python 是一个快乐的过程。相较其他编程语言，Python 语法清晰简洁，代码可读性强，编码方式符合人类思维习惯，易学易用。另外，Python 自带的各种模块加上丰富的第三方模块，免去了很多“重复造轮子”的工作，可以更快地写出东西，非常适合初学者入门以及相关的专业编程人士。

本书分三部分，共 15 章。第一部分基础篇，包含第 1~9 章，介绍 Python 的起源和发展、开发工具、语法基础、控制结构、复合数据结构、字符串与正则表达式、函数、类与对象、文件操作、错误与异常等内容；第二部分进阶篇，包含第 10~14 章，深入讲解 Python 的虚拟环境 Anaconda、科学计算库 NumPy、数据分析库 Pandas、绘图工具 matplotlib 和数据分析工具 SciPy；第三部分实践篇，包含第 15 章，主要介绍 Python 在机器学习领域的应用。

本书内容丰富、难度适中、结构清晰、内容翔实，通过三部分以层次递进方式进行讲解并提供大量实例，引导读者循序渐进地学习、掌握并运用 Python。本书可作为普通高等院校计算机、人工智能、大数据科学、物联网等专业 Python 相关课程的教材，也可作为 Python 爱好者的入门级教程。

除封面署名的作者外，参与本书编写的人员还有周志一、芦扬、刘畅、吕晓阳、王福超、魏婷婷、李昊等。由于作者水平有限，本书难免有不足之处，欢迎广大读者批评指正。我们的信箱是 huchenhao@263.net，电话是 010-62796045。

本书配套的电子课件、实例源文件、习题答案可以到 http://www.tupwk.com.cn/downpage 网站下载，也可以扫描下方的二维码下载。

作者

2019 年 8 月

目　录

第一部分 基础篇

第一部分　基 础 篇

第 1 章

认识Python

Python 是一种解释型的、面向对象的、带有动态语义的高级编程语言，是当今主流编程语言之一。Python 以简洁的语法结构、规范的代码格式、强大且稳定的标准库以及对第三方库的良好兼容而得以广泛应用。本章重点介绍 Python 的基本知识、开发环境的搭建和使用以及 Python 集成开发环境的使用。通过本章的学习，读者可以对 Python 有一个初步的了解，并且学会搭建基本的 Python 开发环境。

本章的学习目标：

- 了解 Python 语言的特点
- 理解 Python 解释器和程序运行过程
- 掌握 Python 在不同操作系统环境的安装方法
- 掌握 Python 代码的两种基本运行方式
- 掌握 Python 集成开发环境的安装和使用
- 了解 Python 的版本差异

1.1 初识 Python

Python 是一种易于学习、功能强大的编程语言。它不仅具有高效的数据结构和简洁的面向对象编程方法，而且具备优雅的语法规范和动态数据类型等特点，这使得它成为许多领域的脚本编写和快速应用程序开发的理想语言。此外，强大且稳定的标准库以及对第三方库的良好兼容能力使得 Python 得到更广泛的应用。特别是近年来随着人工智能技术的快速发展，Python 作为数据分析的强有力工具而大放异彩。系统地学习和掌握 Python 语言不仅适合于初学者，对于有经验的专业技术人员也相当重要。

1.1.1 编程语言概述

除 Python 编程语言外，目前流行的编程语言非常多，如 C/C++、Java、C#等。为了更深入地理解 Python 编程语言及其优势，有必要对计算机编程语言的基本知识做些了解。编程语言是用来定义计算机程序的形式语言，通常包括机器语言、汇编语言、高级语言。简单来讲，编程语言是一种被标准化的交流方式，用来向计算机发出指令并指导计算机按照指定要求进行工作。因此，编程语言是让程序员能够准确地定义计算机所要使用的数据，并精确地定义在不同情况

下应当采取什么行动的一种工具。

1. 机器语言

机器语言是一种指令集，属于低级语言。这种指令集通常被称为机器码(Machine Code)，是计算机中央处理器(CPU)可直接解读的数据和指令。通俗来讲，机器语言是用二进制代码表示计算机能直接识别和执行的机器指令系统，是计算机的设计者通过计算机的硬件结构赋予计算机的基本操作功能。机器语言具有灵活、直接执行和速度快等特点。但是使用机器语言编写的程序无明显特征，难以记忆，不便阅读和书写，且依赖于具体机器硬件，局限性很大。

2. 汇编语言

汇编语言是面向机器的程序设计语言。在汇编语言中，采用助记符代替操作码，用地址符号(Symbol)或标号(Label)代替地址码，因此汇编语言又称为符号语言。使用汇编语言编写的程序，机器不能直接识别，需要用一种系统程序(即汇编程序)将汇编语言翻译成机器语言。在实际编程中，通常使用汇编语言来完成一般高级语言所不能够实现的低层功能。由于经汇编生成的可执行文件不仅比较小，而且执行速度很快，因此汇编语言多适用于编写与底层硬件密切相关的系统程序。然而，汇编程序的源代码一般比较冗长、结构较复杂，因此不方便阅读且容易出错。使用汇编语言编程通常需要更多的计算机专业知识，因此实际使用汇编语言编写的应用程序相对较少。

3. 高级语言

高级语言相对于汇编语言而言，编程方式与人们思考问题的思维方式更加接近，语法结构更加符合自然语言。高级语言通常并不特指某种具体语言，而是包括多种编程语言，如目前流行的 Java、C/C++、C#、Pascal、Python 等。使用高级语言编写的程序也不能直接被计算机识别，必须经过转换才能执行，按转换方式的不同通常将它们分为两类：编译型语言和解释型语言。

编译型语言：编译型语言是指在程序执行之前，需要将程序的源代码“翻译”成目标程序(机器语言)，目标程序可以脱离语言环境独立执行。编译型语言通常使用比较方便、执行效率较高。但应用程序一旦需要修改，就必须重新编译生成新的目标文件才能执行。以 C 语言为例，编译型语言从编辑到输出结果的整个过程如图 1-1 所示。

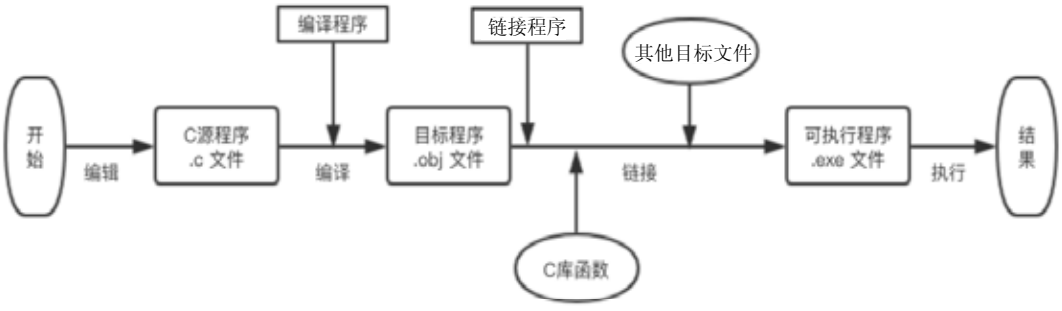

图 1-1　C 语言程序的执行过程

C 语言程序从开始编写到输出需要经过编辑、编译、链接和运行四个步骤。其他编译型语言与此类类似。首先按照语言规定的语法格式编写程序代码，然后通过编译程序将程序代码翻译成二进制表示的目标文件，接着进行程序链接，即使用系统提供的链接程序将目标程序、库函数或其他目标程序链接装配成可执行的目标程序，最后将可执行的目标程序调入实际机器的内存，执行并输出结果。编译型语言通常能够依据目标机器的硬件特性进行编译时优化，因此执行效率较高。

解释型语言：执行方式类似于日常生活中的“同声翻译”，程序的源代码一边由相应语言的解释器“翻译”成目标代码(机器语言)，一边执行，因此执行效率相比编译型语言要低一些。解释型语言通常不生成可独立执行的可执行文件，因此程序不能脱离解释器独立运行。但是这种方式提供了比编译型语言更高的灵活性，可以更容易实现程序的交互执行和动态调整优化等。比较流行的解释型编程语言有 Python、Perl 等。

解释型语言的关键部分是解释器和虚拟机。解释器负责将源代码翻译为与平台无关的字节码文件，而虚拟机是解释型语言的运行引擎，负责实际代码指令的执行，如图 1-2 所示。解释型语言在程序运行时需要通过解释器对程序进行动态解释和执行，因此使用更加灵活、可移植性也非常好。

图 1-2　解释型语言的执行过程

1.1.2　Python 常用解释器

Python 是一种解释型高级编程语言，因此 Python 代码的成功运行依赖于设计良好的解释器。在具体实现上，Python 语言的解释器有多种不同版本，其中包括 CPython、IPython、PyPy、Jython 和 IronPython 等，如表 1-1 所示。

表 1-1　Python 常用解释器

名称	特点
CPython	官方版本的解释器。使用 C 语言开发，是使用最广泛的 Python 解释器
IPython	基于 CPython，但在交互方式上有所增强，执行 Python 代码的功能和 CPython 完全一样
PyPy	一种追求执行速度的 Python 解释器。采用 JIT 技术，对 Python 代码进行动态编译，能提升 Python 代码的执行速度
Jython	运行在 Java 平台上的 Python 解释器，可以直接把 Python 代码编译成 Java 字节码
IronPython	运行在微软.NET 平台上的 Python 解释器，可以直接把 Python 代码编译成.NET 的字节码

1.1.3　Python 语言特点

Python 本身是由诸多其他语言发展而来的，因此不仅具有其他语言许多共同的特性，而且具备很多其他语言没有的特点。Python 除了具有作为解释型语言和面向对象编程语言的固有优势外，还具有许多其他鲜明的特点，例如互动模式、可嵌入性、动态语言等，如表 1-2 所示。

表 1-2　Python 语言特点

特点	说明
易于学习	Python 语言的关键字较少、结构相对简单、语法定义简洁
易于阅读	Python 代码更清晰。和其他语言不同，Python 不需要定义变量类型、代码块和命令式符号等
丰富的标准库	Python 标准库足够丰富，例如 NumPy、matplotlib、SciPy 等用于数值计算与可视化的 Python 库
互动模式	互动模式支持用户直接从终端输入代码执行，并即时返回运行结果
跨平台性	Python 使用 C 语言编写，因此 Python 可以运行在任何带有 ANSIC 编译器的平台上。Python 程序无须修改即可在主流平台上运行，如 Linux、Windows 等
数据库支持	Python 提供了大量主流数据库的接口，例如 MySQL、SQLServer 和 Oracle 等
GUI 编程	Python 支持 GUI(图形用户界面)编程
可扩展性	如果需要一段关键代码运行得更快或者希望某些算法不公开，可以部分程序用 C 或 C++编写，然后在 Python 程序中使用它们
可嵌入性	Python 语言可嵌入 C/C++程序，让程序具有“脚本化”能力
解释性	Python 解释器把源代码转换成称为字节码的中间形式，然后再把它们翻译成计算机使用的机器语言并运行
动态语言	Python 能够在运行时修改程序结构，相比编译型语言提供了更为灵活的运行方式
面向对象	Python 支持以对象作为基本程序结构单位，因此 Python 程序是由对象(对象由数据和功能组合而成)构建起来的

编程语言从接近于机器语言的汇编语言起步，到经典的 C/C++语言，再到以 Java 和 C#为代表的完全面向对象编程语言，在此发展过程中，大量工程实践证明，完全的面向对象编程方式虽然可以达到相当高程度的抽象和封装，但是对于公共模块或功能的调用在一定程度上过于刻板，容易造成代码的大量冗余。Python 语言融合了面向过程和面向对象的优点，以功能强大、跨平台、友好、易学等特点完美诠释了人们追求的理念。

1.2 Python 的安装

在不同的操作系统环境中，Python 的安装也不尽相同。本节主要介绍 Python 在 Windows、Linux、macOS 环境中的安装过程及注意事项。

1.2.1 Windows 环境中 Python 的安装

1. Python 安装包的下载

访问 Python 官网(https://www.python.org/downloads/)，如图 1-3 所示，页面默认提供 Python 3.7.x 版本供下载，单击 Download Python 3.7.2 按钮可以直接下载 Python 3.7.2 的安装文件。当然也可以通过访问页面下方的 Looking for a specific release 列表以下载特定版本，例如 Python

2.7.6 的 Python 安装文件，如图 1-4 所示。

图 1-3　Python 官网下载界面

图 1-4　Python 安装版本列表

选择合适的 Python 版本后，单击 Download 进入下载详情页面。选择 Windows 平台常用的 Python 安装文件 Windows x86-64 executable installer 进行下载，如图 1-5 所示。

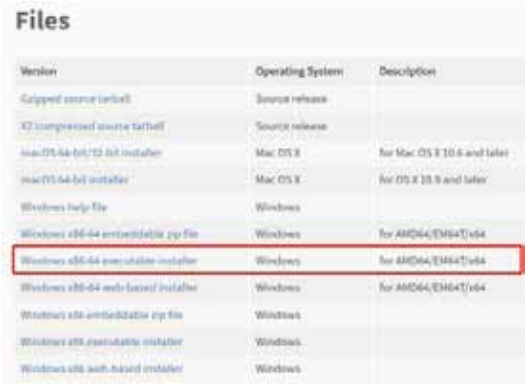

图 1-5　Python 安装文件列表

在选择 Python 版本时，需要注意以下事项：

(1) Python 软件包名中的 x86 指 32 位机，x86-64 指 64 位机，注意 64 位机既可以安装 64 位也可以安装 32 位的软件包，但 32 位机只能安装 32 位的软件包。

(2) 首选下载 Python x.x.x 版本，Python x.x.x rc 版本属于候选版本。

(3) 注意区分图 1-5 中软件安装包的类别。

- web-based install 表示通过网页完成安装。
- executable install 表示以可执行文件(*.exe)方式安装，下载后直接在本地安装。
- embeddable zip file 表示嵌入式版本，可以集成到其他应用中。

(4) 在 Python 2.x 和某些 Python 3.x 版本中，安装包是以 msi 数据包的形式提供的。但是在最新的 Python 3.6 和 Python 3.7 版本中，安装包主要以.exe 文件的形式提供。

2. Python 的安装与配置

具体步骤如下：

(1) 双击已下载的安装包，例如 python-3.7.2-amd64.exe，进入 Python 安装向导，如图 1-6 所示。

(2) 选中 Add Python 3.7 to PATH 以方便后面直接运行 python 命令，然后选择 Install Now 按照系统默认配置进行安装。默认安装路径通常是 C:\User\<user name>\AppData\Local\Programs\Python\Python37。为了环境配置方便，此处建议选择 Customize installation，进行自定义安装。

(3) 建议不改变默认设置，安装所有可选特性库，如图 1-7 所示，然后单击 Next，进入 Python 安装高级选项界面。

图 1-6　进入 Python 安装向导

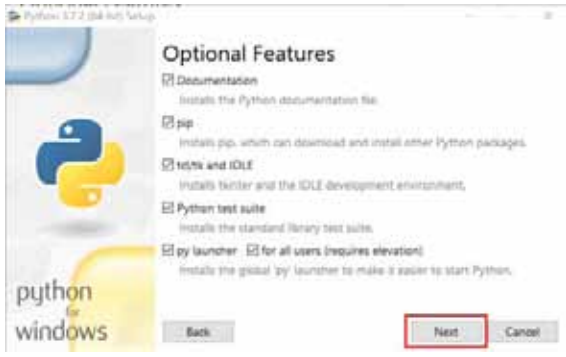

图 1-7　Python 安装可选功能界面

(4) 在 Python 安装高级选项界面中，建议选中所有选项，并更改安装路径，建议设置为 D:\Python37，如图 1-8 所示。单击 Install 进入 Python 加载安装界面。

Install for all users 选项选中后，表示为所有用户安装 Python，未选中则表示仅为当前用户安装 Python。

(5) 在安装阶段，安装器会将软件包复制到指定文件夹，并进行相关配置，如图 1-9 所示。

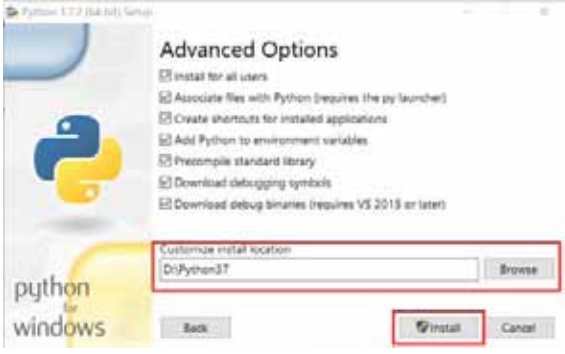

图 1-8　Python 安装高级选项界面

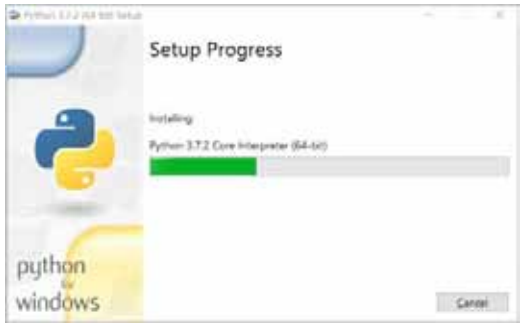

图 1-9　Python 正在安装

(6) 安装完成后会提示安装成功，如图 1-10 所示。

(7) 安装完之后，“开始”菜单中会出现 Python 3.7 图标，如图 1-11 所示。单击该图标可进入 Python 交互式界面，如图 1-12 所示。此时，Python 在 Windows 环境中的安装已经完成，可以愉快地开始 Python 之旅了。

图 1-10　安装完成

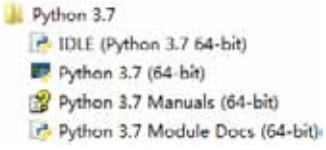

图 1-11　Python 启动菜单

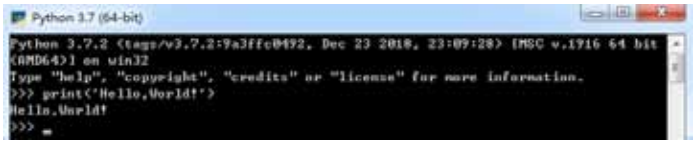

图 1-12　Python 交互式界面

1.2.2　Linux 环境中 Python 的安装

大多数 Linux 系统中已经预装了 Python，但如果对版本有特殊需求，可以自行下载不同版本的 Python 进行安装。下面介绍两种基本安装方式：源码编译安装方式和包管理器安装方式。

1. 包管理器安装方式

Linux 操作系统存在多种软件包管理系统和安装机制，可以通过 Linux 系统自带的包管理器安装软件。受限于 Linux 的安全机制，通过 Linux 包管理器安装软件需要具备系统管理员(root 或 sudo 账号)权限。利用 Linux 包管理器安装 Python 非常简便，以在 Ubuntu 系统中安装 Python 3.5 版本为例，如图 1-13 所示，只需要执行如下命令：

```
$ sudo apt-get install python3.5
```

安装完成后，可以在命令行终端打开 Python 3.5 的交互式环境。在 Linux 终端输入命令 python3.5 后回车执行，如图 1-14 所示。

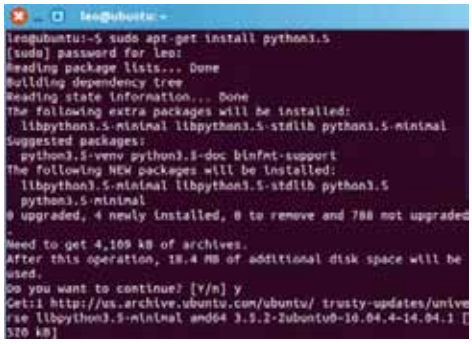

图 1-13　安装 Python 3.5

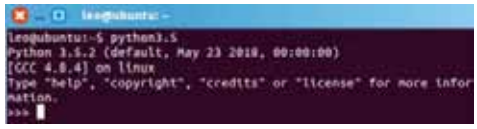

图 1-14　Python 3.5 的交互式环境

2. 源码编译安装方式

使用包管理器安装 Python 相当方便，但是在某些情况下不能使用包管理器进行安装，例如当前用户没有 root 权限、需要特殊 Python 版本等，此时可以通过源码编译安装方式进行安装。通过源码安装 Python 虽然过程稍有点复杂，但是更加灵活，安装步骤如下：

(1) 访问 https://www.python.org/downloads/source/，选择下载 Python 源码文件。以 Python 3.7.2 版本为例，选择 Gzipped source tarball 进行下载，如图 1-15 所示，获得 Python-3.7.2.tgz 软件压缩包。

(2) 对 Python-3.7.2.tgz 软件压缩包进行解压，解压指令为：

```
$ tar  -zvxf  Python-3.7.2.tgz
```

注意，Python-3.7.2.tgz 可替换为实际下载的文件名。解压后生成新的目录 Python-3.7.2，如图 1-16 所示。

图 1-15　Python 下载页面

图 1-16　解压文件并生成新的目录

通过 cd 命令切换到 Python 源码目录进行编译安装。在 Linux 环境中通过源码安装软件通常需要执行 3 个基本步骤：配置(configure)、编译(make)、安装(make install)。进入 Python 源码目录后，顺序执行如下命令即可完成安装。

```
$ ./configure
$ make
$ make install
```

详细安装过程参考图 1-17~图 1-21。

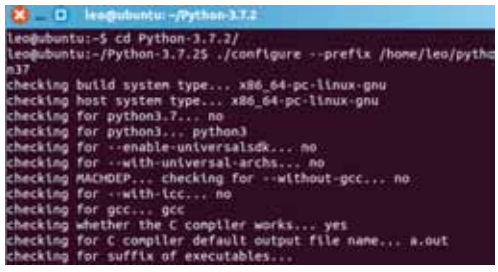

图 1-17　配置 Python 安装环境

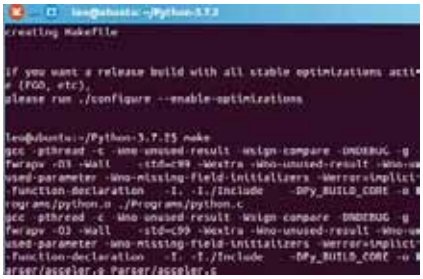

图 1-18　编译 Python 源码

图 1-19　安装 Python 到指定目录

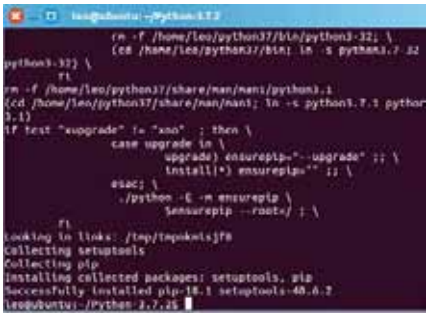

图 1-20　Python 安装成功

图 1-21　执行 Python 解释器

通过源码安装 Python 可以非常方便地指定最终执行文件的安装路径。可通过配置指令 configure 的--prefix 参数指定安装路径，例如：

```
$ ./configure --prefix /home/leo/python37
```

通过--prefix 参数，可将 Python 安装到用户 leo 自己的主目录的 python37 目录中。由于在 Linux 系统中，普通用户对自己的主目录有完全的读写权限，因此在没有管理员权限时，这种安装方式更为灵活。注意，如果使用 configure 指令时未指定--prefix 参数，Python 可执行文件默认将放置在/usr/local/bin 目录中，库文件默认放置在/usr/local/lib 目录中，配置文件默认放置在/usr/local/etc 目录中，其他相关的资源文件放置在/usr/local/share 目录中。

运行命令./python37/bin/python3 以启动 Python 交互式环境，图 1-21 表明 Python 在 Linux 系统中已安装完成。

注意，如果使用 configure 指令为 Python 指定了安装路径，为了能够方便使用新版 Python，在 Linux 系统中还需要手动修改 PATH 环境变量。利用 Vim 等文本编辑器编辑 Linux 用户主目录中的.bashrc 文件，在文件的最后增加一行并输入如下内容：

```
export PATH= /home/leo/python37/bin:$PATH
```

1.2.3　Mac OS 环境中 Python 的安装

同 Linux 系统一样，Mac OS 系统一般也预装了 Python，当然也可以使用和 Linux 相似的方式，用源码编译安装特定版本的 Python，这里不再赘述。下面介绍使用图形界面的方法安装 Python。

1. Python 安装包的下载

访问 Python 官网(https://www.python.org/downloads/)，选择下载 Python Releases for Mac OS

X。以 Python 3.7.2 版本为例，选择 macOS 64-bit installer 进行下载，如图 1-22 所示，获得 python-3.7.2-macosx10.9.pkg 文件。

图 1-22　下载 Python 安装文件

2. 安装 Python

具体步骤如下：

(1) 双击下载的 python-3.7.2-macosx10.9.pkg 文件，进入安装介绍界面，如图 1-23 所示。单击“继续”，进入安装前需要阅读的重要信息界面。

(2) 阅读“请先阅读”和“许可”相关内容，并同意相关要求，如图 1-24 和图 1-25 所示，单击“继续”。

图 1-23　安装介绍界面

图 1-24　安装前需要阅读的重要信息界面

(4) 选择目的卷宗，可以安装在不同的磁盘上，如图 1-26 所示，单击“继续”进入“安装类型”界面。

图 1-25　许可信息

图 1-26　选择安装的目标存储位置

(5) 单击“更改安装位置”按钮可选择安装位置，如图 1-27 所示，单击“安装”进入安装界面。

(6) 安装完成后会提示安装成功，如图 1-28 所示。

图 1-27　设置完成后的安装确认界面

图 1-28　安装成功界面

(7) 安装成功后会自动打开 Python 应用程序文件夹，如图 1-29 所示。

(8) 双击图 1-29 中的 IDLE 图标，打开 Python Shell，显示 Python 解释器的相关信息，如图 1-30 所示。

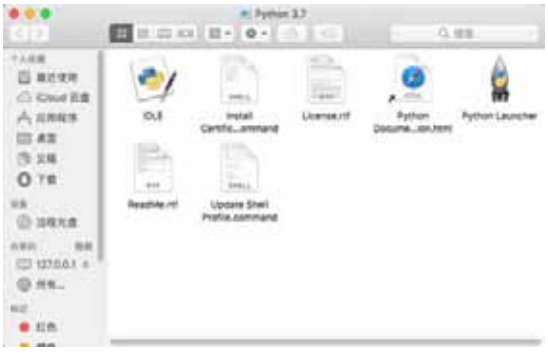

图 1-29　Python 应用程序文件夹

图 1-30　Python Shell

1.3　Python 代码的执行

Python 代码的编辑和运行方式主要分为两种：交互模式和脚本模式。在交互模式下，用户输入 Python 代码并按回车键后，Python 解释器将立即解释执行代码并返回结果；在脚本模式下，用户将已经编写好的 Python 代码文件作为 Python 解释器命令的参数，由解释器解释批量执行并返回结果。下面通过具体实例介绍这两种方式。

1.3.1　在交互模式下执行 Python 代码

一般来说，成功安装 Python 之后，有两种方式可以进入 Python 交互模式。一种是通过 Python 自带的非常简洁的集成开发环境 IDLE，如图 1-31 所示。另一种是在系统命令行终端直接运行

python 命令，如图 1-32 所示。在成功进入 Python 交互模式后，控制台的最后一行会显示 Python 命令提示符>>>，此时可以键入 Python 语句进行交互式执行。

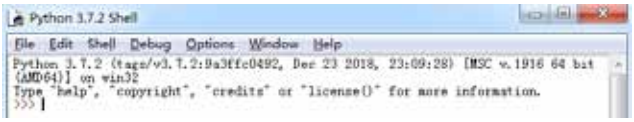

图 1-31　Python 自带集成开发环境 IDLE

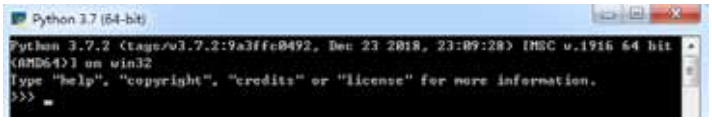

图 1-32　Python 命令终端的交互模式

在交互模式下，输入代码并回车后会立即执行并打印执行结果，当输入不合法的 Python 语句时，控制台将立即显示相关错误信息，如图 1-33 所示。

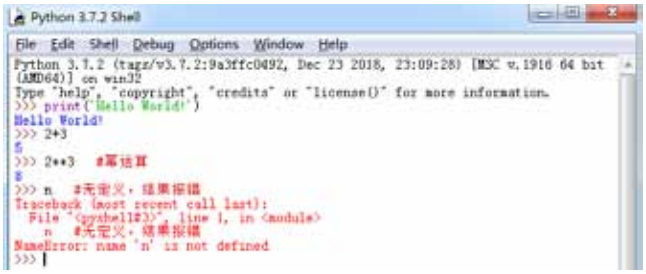

图 1-33　Python 交互式实例

在 IDLE 环境下使用 Python 交互模式时，可以通过 Alt+P 和 Alt+N 组合键查询之前输入的历史命令。在系统命令行终端直接运行 python 命令，进入交互模式，可使用向上和向下的箭头查询之前输入的命令，实现 Python 程序语句的重复使用和执行。但时，在交互式环境中输入的代码不会被自动保存，所以在关闭或退出 Python 交互式环境后，之前输入的代码将不复存在。

在 IDLE 环境下，其他较常用的快捷键如表 1-3 所示。

表 1-3　IDLE 常用快捷键

名称	功能说明
Alt+3	多行注释
Alt+4	取消多行注释
Tab	命令自动补全
Alt+P	浏览上一条命令
Alt+N	浏览下一条命令
Ctrl+]	多行代码缩进
Ctrl+[	取消多行代码缩进
F5	进入 Python Shell 调试界面

1.3.2　在脚本模式下执行 Python 代码

Python 脚本通常是扩展名为.py 的文本文件，Python 脚本文件可以使用常用的任何文本编译器进行编辑修改。例如，以 Windows 平台为例，可以将代码实例 1-1 保存到 D:\Python_src\Exam01.py 文件中。

代码实例 1-1

```
print('Hello World!')        # 打印字符串
print('2 + 3 = ', 2 + 3)     # 进行简单计算
print('2**3 = ', 2**3)       # 幂计算
print('a = ', a)             # 无定义，结果会报错
```

进入 Windows 命令行终端，执行 python D:\Python_src\Exam01.py 命令，结果如图 1-34 所示。Python 脚本在执行过程中，如果遇到错误，将会终止脚本的执行并在系统终端控制台打印错误信息。

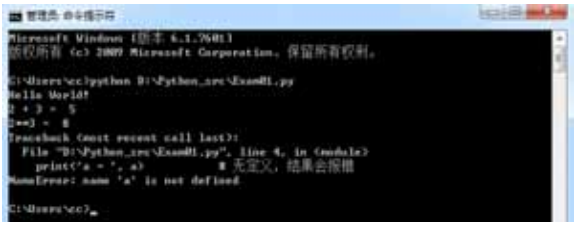

图 1-34　运行 Python 脚本

在脚本模式下，在 python 命令的后面可以附加一些参数来扩展功能。例如，可以使用 python -V 查看 Python 版本信息，python 命令的常用参数如表 1-4 所示。

表 1-4　Python 常用的命令行参数

选项	描述
-d	在解析时显示调试信息
-O	生成优化代码(.pyo 文件)
-S	启动时不引入查找 Python 路径的位置
-V	输出 Python 版本信息
-x	忽略脚本的第一行，以更好兼容非 UNIX 平台的脚本
-c cmd	执行 Python 脚本，并将运行结果作为 cmd 字符串
-h	打印 python 命令的帮助信息

1.4　Python 集成开发环境

集成开发环境(IDE，Integrated Development Environment)是专用于软件开发的软件程序。顾

名思义，IDE 集成了为软件开发而设计的工具，通常包括专门为了处理代码的编辑器(包含语法高亮和自动补全等功能)，以及构建、执行、调试工具和某种形式的源代码控制。大部分的集成开发环境兼容多种编程语言，并且提供方便灵活的扩展机制以便支持更多功能。

目前支持 Python 语言开发的 IDE 非常多，如 PyCharm、VS Code、Eclipse + PyDev、Spyder、Thonny 和 Komodo 等。不同 IDE 的使用方法大同小异，本书以 PyCharm 为例进行简要介绍。PyCharm 由 JetBrains 公司推出，是目前较为常用的 Python IDE，带有一系列可以提升 Python 开发效率的工具，如调试、语法高亮、代码跳转、智能提示、自动完成、单元测试、版本控制、项目管理等。在使用 PyCharm 时，可以根据需要在线或离线安装所需第三方库，非常便捷。

1.4.1 PyCharm 的安装

JetBrains 公司的官网分别提供了支持 Windows、Linux 和 Mac OS 平台的 PyCharm 版本，开发者可以根据需要选择下载。针对各种操作系统的 PyCharm 版本的安装过程大同小异，本书以 Windows 平台上的安装为例进行介绍。

1. 下载 PyCharm

访问 http://www.jetbrains.com/pycharm/download/index.html#section=windows 并下载软件包，下载页面如图 1-35 所示。

图 1-35　PyCharm 下载页面

针对每种操作系统，PyCharm 均分为 Professional 版本和 Community 版本。其中，Professional 版本提供 Python 开发的全部功能，可免费试用，商用时需要付费。Community 版本完全免费，但是不能开发 Web 项目。本书以 Professional 版本的 pycharm-professional-2018.3.4.exe 安装为例进行介绍。

2. 安装 PyCharm

(1) 双击下载好的文件 pycharm-professional-2018.3.4.exe，进入安装向导，如图 1-36 所示，单击 Next>按钮。

(2) 进入安装路径选择界面，如图 1-37 所示，单击 Next>按钮。

(3) 进入安装参数设置界面，如图 1-38 所示。开发者可以根据需要勾选所需的选项，单击 Next 按钮继续安装。

(4) 设置开始菜单文件夹的名字，一般使用默认设置即可，如图 1-39 所示。单击 Install 按钮，进入正式安装过程。

图 1-36　安装向导

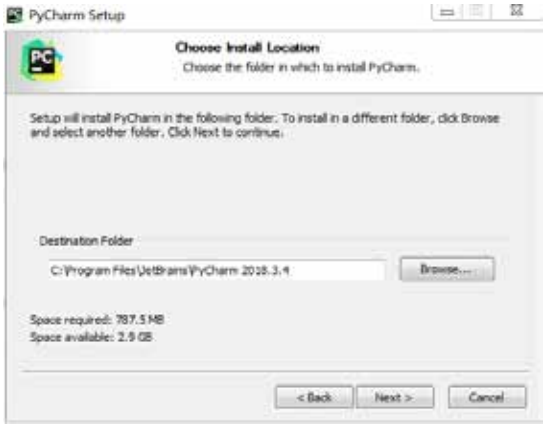

图 1-37　安装路径选择界面

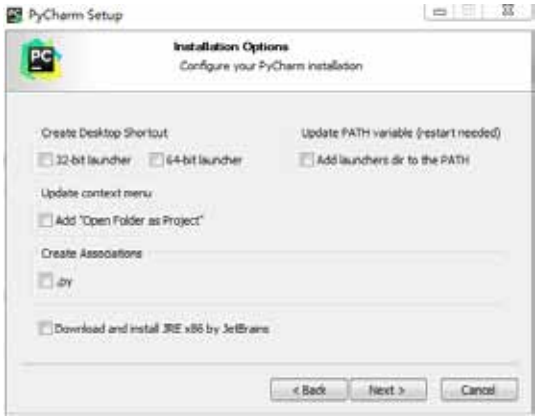

图 1-38　安装参数设置界面

图 1-39　命名开始菜单文件夹

(5) 开始安装后，会出现安装进度提示，如图 1-40 所示。安装完成后的界面如图 1-41 所示。

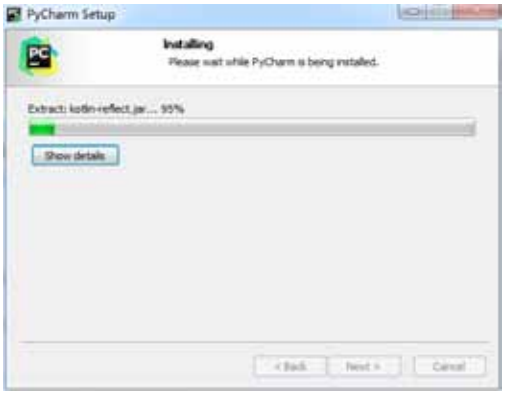

图 1-40　安装过程

图 1-41　安装完成

至此，PyCharm 已经安装完成。开发者可以选中图 1-41 中的 Run PyCharm 复选框运行 PyCharm 软件了。需要注意的是，在运行 PyCharm 软件之前，一定要确保操作系统中已经搭建好相应的 Python 环境了。

1.4.2 PyCharm 的使用

首次打开 PyCharm IDE 时，会提示选择界面的 UI 主题，如图 1-42 所示。

对于初学者，可以单击 Skip Remaining and Set Defaults 按钮，使用系统默认的 UI 风格，如图 1-42 所示。单击该按钮后，进入项目引导界面，如图 1-43 所示。

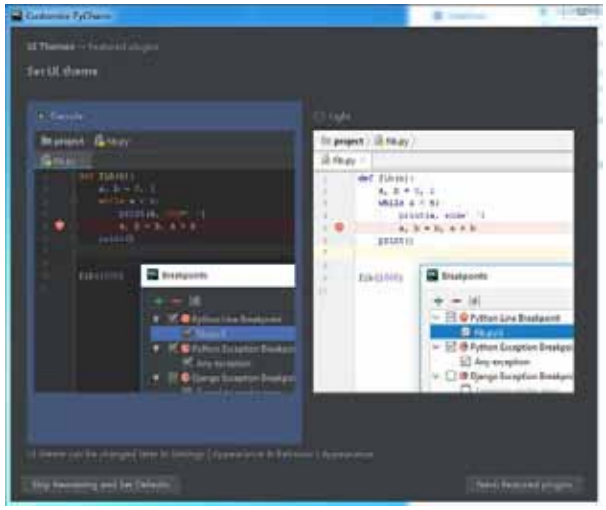

图 1-42　选择 UI 主题

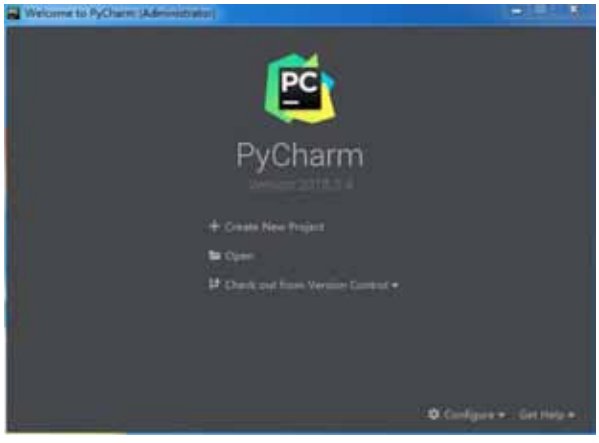

图 1-43　项目引导界面

在项目引导界面中，可以选择创建新项目或打开已有项目。对于 PyCharm，代码的运行是以项目为单位组织的，一个项目下可以有多个 Python 文件。因为是首次运行 PyCharm，所以需要新建一个项目。单击 Create New Project 按钮，进入项目创建界面，如图 1-44 所示。

在项目创建界面中，可以根据需要在左边栏选择创建不同类型的项目。在此创建一个 Pure Python 类型的项目，项目名为 MyProject，然后单击 Create 按钮后，进入 PyCharm 主界面，如

图 1-45 所示。

在 Python 主界面中，可以在所创建项目的名称上右击，选择创建 Python 文件，如图 1-46 所示。

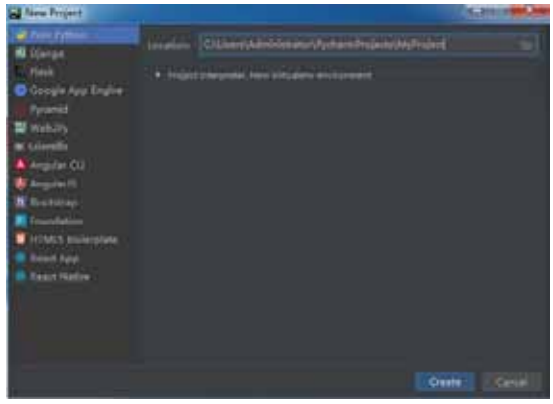

图 1-44　项目创建界面

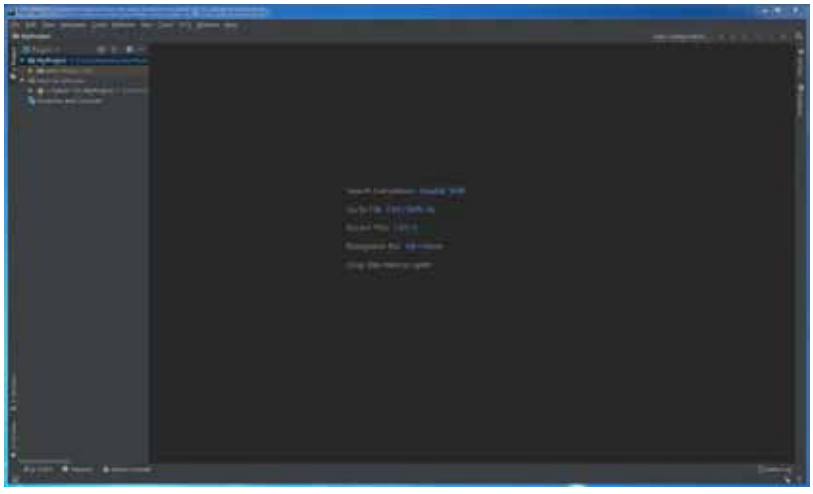

图 1-45　PyCharm 主界面

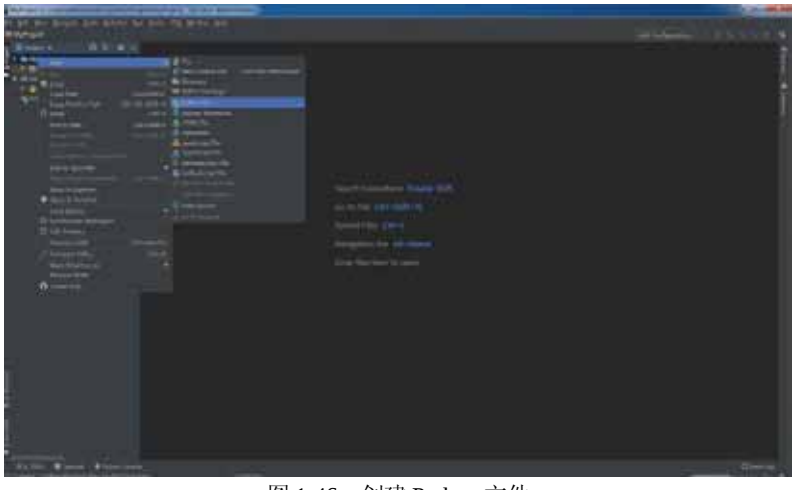

图 1-46　创建 Python 文件

在此，创建一个名为 MyTest 的 Python 文件，并完成输出"Hello World"字符串的功能，如图 1-47 所示。

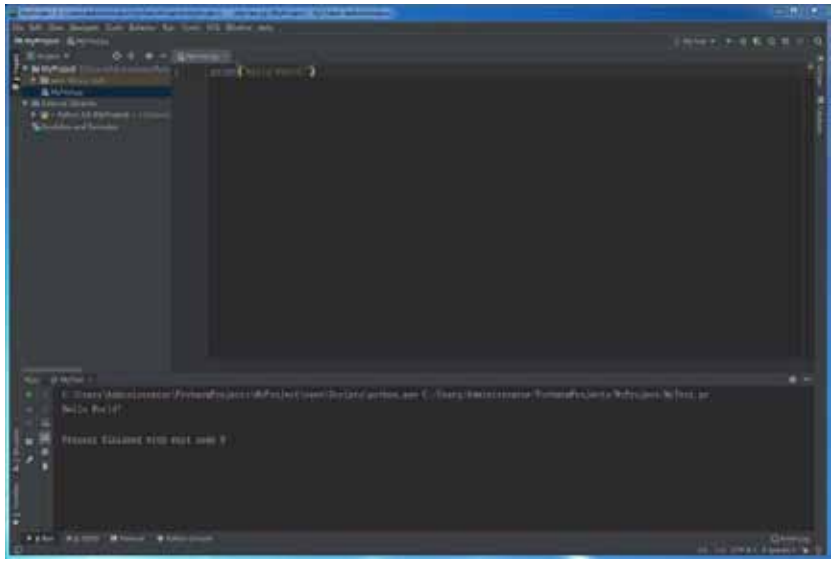

图 1-47　运行 Python 代码

1.4.3　PyCharm 的插件

PyCharm 在按照默认设置安装完成后，实际上已经集成了很多常用插件，对于初学者来说已经足够。如果要开发较为复杂的项目，需要第三方插件，也可以在 PyCharm 中进行添加。下面以 Python 中常用的绘图库 matplotlib 为例介绍 PyCharm 中安装插件的流程，其他插件的安装流程与此类似。

(1) 单击菜单项 File | Settings，进入插件添加界面，如图 1-48 所示。单击+后，进入插件搜索界面。

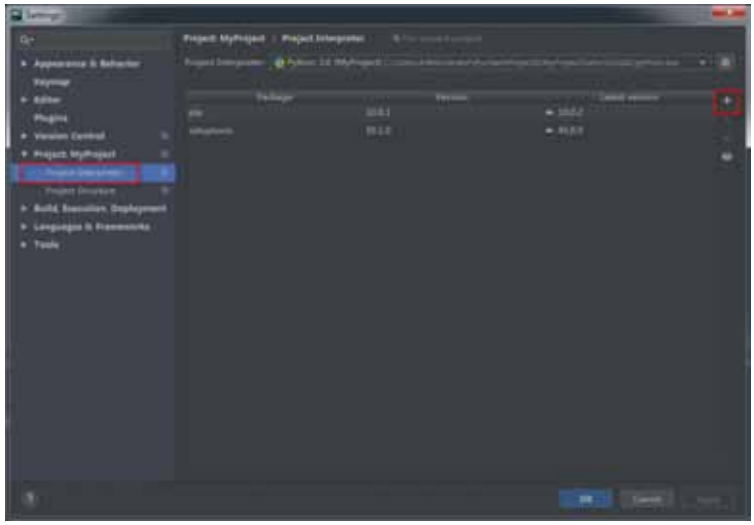

图 1-48　插件添加界面

(2) 在搜索栏中输入要搜索的插件名称 matplotlib，找到相应的插件后，单击 Install Package 按钮安装插件，如图 1-49 所示。

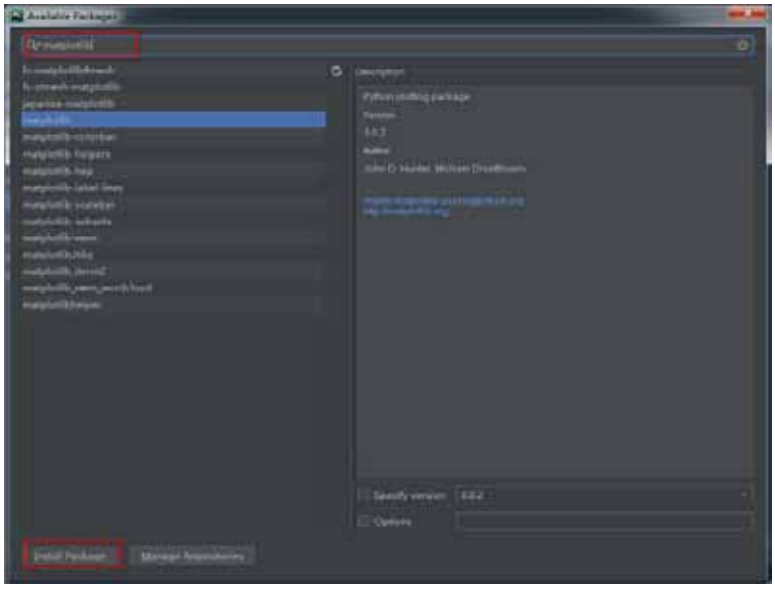

图 1-49　搜索插件

(3) 安装过程通常是在线安装，所以在安装时要保证网络畅通。根据网络连接速度的快慢以及插件包的大小，安装时长不定。安装成功后，出现 Package 'matplotlib' installed successfully 提示，如图 1-50 所示。

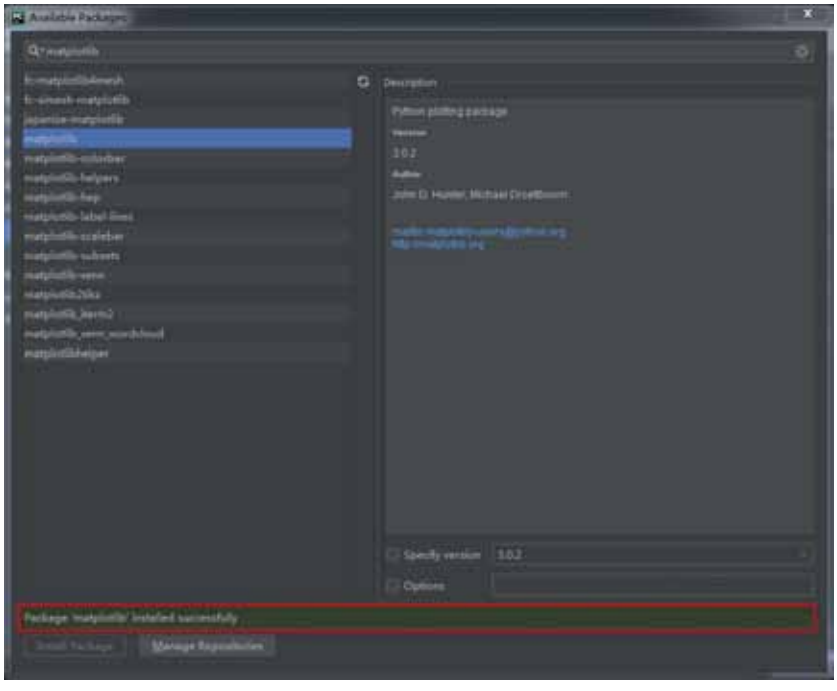

图 1-50　成功安装 matplotlib 插件

为确保插件安装成功，可以使用 matplotlib 插件绘制一张折线图，如图 1-51 所示。

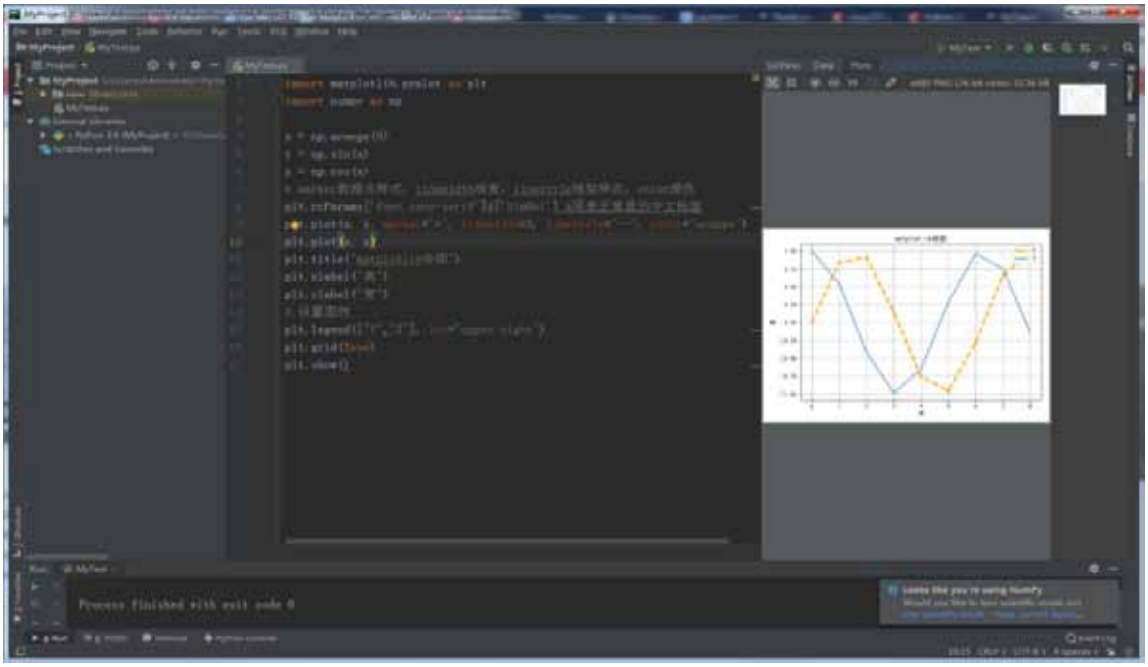

图 1-51　使用 matplotlib 插件绘制折线图

1.5 Python 2.x 与 Python 3.x 的区别

Python 主要有 Python 2.x 和 Python 3.x 两个不同版本系列，它们互不兼容。2008 年 Python 3.0 发布，两年后，Python 2.x 系列的最后一个版本 Python 2.7 发布。在这之后，Python 2.x 系列没有任何主要的新属性发布。Python 3.x 一直处于持续开发中，这意味着新开发的标准库只在 Python 3.x 中展现。Python 3.x 对 Python 2.x 的语法进行了适当的废弃、修改，并新增了一些特性，所有在 Python 3.x 中增加的新特性在 Python 2.x 中均不支持。通过 pip 官方下载源 PyPI 搜索两个版本的第三方工具包可以发现，两个版本在第三方工具包的支持数量上差距也相当大。Python 3.x 引入了一些与 Python 2.x 不兼容的关键字和特性，在 Python 2.x 中，可以通过内置的__future__模块导入这些新内容。如果希望在 Python 2.x 环境下编写的代码也可以在 Python 3.x 环境下运行，那么建议使用__future__模块。例如，如果希望在 Python 2.x 中拥有 Python 3.x 的整数除法行为，可以通过下面的语句导入相应的模块。

```
from __future__ import division
```

两个 Python 系列版本之间的部分区别如表 1-5 所示， Python 版本的每次更新都可能会增加新的特性，想要了解更多的新特性，可以查看官方文档。

表 1-5　Python 2.x 和 Python 3.x 的主要区别

	Python 2.x	Python 3.x
字符编码	默认编码方式是 ASCII，所以默认编码方式下不支持中文，解决方式：在文件的首行加上#-*- encoding:utf-8 -*-	默认编码方式是 UTF-8
基础数据类型	对于长整型，type()得到的数据类型是 long	对于长整型，type()得到的数据类型是 int
整数除法	整数除法的取值结果是向下取整数	除法/的结果包含小数，实现真除；除法//实现整除，结果是向下取整
print	print 可以加括号也可以不加，如 print 'abc' 或 print('abc')	print 只有一种用法，必须加括号，如 print('abc')
range	分为 range()和 xrange()，用法基本相同，但原理不同，range()直接生成一个列表，而 xrange()是一个生成器，类似于迭代器	range() 就 是 Python 2.x 中 的 xrange()
input	分为 raw_input()和 input()，区别在于，raw_input()读取的输入流默认转换为字符串，而 input()读取的输入流要按照基本格式输入，比如输入字符串需要加引号，数字则不需要	只有 input()，相当于 Python 2.x 中的 raw_input()，默认得到的输入都是字符串

1.6 本章小结

本章首先介绍了 Python 语言的运行机制与特点，并详细讲述了 Python 在主流操作系统平台上的安装和使用；其次，介绍了 Python 代码的编辑与运行以及 Python 集成开发环境；最后就 Python 2.x 与 Python 3.x 两个系列版本的区别做了对比。

第 2 章

Python语法基础

Python 语法与 C/C++、Java 和 C#等语言有相似之处，但是也存在很大差异。Python 具有更为简洁的语法结构，Python 程序更具有可读性且易于维护。本章主要介绍 Python 语法基础。通过本章的学习，读者可以掌握数据类型、标识符和变量的基本概念及用法，掌握语法规则和 I/O 操作相关函数的使用，了解 Python 对象的含义及用法，为使用 Python 编程打下基础。

本章的学习目标：

- 掌握 Python 数据类型的含义及用法
- 掌握 Python 标识符的含义及用法
- 掌握 Python 变量的定义及使用
- 掌握 Python 编程的语法规则
- 掌握 I/O 操作相关函数的使用
- 了解 Python 对象的含义及用法

2.1 数据类型

计算机最终执行的都是二进制形式的机器指令，但为了提高编程效率并同时降低编程难度，绝大多数编程语言都在逻辑层面设置了不同的数据类型，Python 语言也是如此。和其他编程语言不同，Python 语言中的变量在使用前并不需要进行显式数据类型声明。但是，这并不代表它们没有类型，因为每个变量在使用前都必须赋值，变量只有赋值以后才会被真正创建。

通常，变量的类型实质上是指变量在内存中对应对象的存储类型，Python 变量的类型是依据变量的赋值内容而自动生成的。Python 变量可以是不同的数据类型，不仅可以是数字，还可以是任意合法的数据类型。

Python 的六大数据类型如下所示。

- 数字(Number)：包含 int(整型)、long(长整型)、complex(复数)、float(浮点型)、bool(布尔型)。
- 字符串(String)：例如"Python"、'Python'。
- 列表(List)：例如[1,2,3,4]、[5,6,7,[8,9],10]。
- 字典(Dictionary)：例如{1:"study",2:"Python"}。