第1章 Python 入门

本章导语

在这个快速发展的数字时代，编程已经成为一项核心技能。Python 作为最受欢迎的编程语言之一，以其易学易用和强大的功能脱颖而出。Python 由于从数据科学到网络开发，从自动化到人工智能的广泛应用，已成为技术创新和职业发展的强大工具。

本章作为 Python 学习之旅的第一步，将提供学习 Python 所需的必要工具，从了解 Python 的基本特性到实际安装和运行第一个程序，通过详细的讲解以确保读者在未来的学习中少走弯路。

学习目标

（1）了解 Python 的起源、发展及其在当今世界的重要性。

（2）了解 Python 语言的核心特点和主要应用领域。

（3）能够安装 Python 解释器，并学会基本的程序运行方法。

（4）掌握使用 PyCharm 强大的集成开发环境来编写和测试 Python 程序。

（5）能够进行基础的 Python 编程，解决简单的编程问题。

1.1 Python 概 述

1.1.1 Python 的发展历程

1989 年 12 月，来自荷兰阿姆斯特丹的吉多·范罗苏姆（Guido van Rossum），想起自己曾参与设计的一种优美与强大并存，但最终惨遭失败的程序设计语言 ABC，想着不如开发一个新的脚本解释程序，作为 ABC 语言的继承，于是 Python 诞生了。他用 C 语言写出了 Python 的解释器，由于其非常喜爱一部叫 *Monty Python's Flying Circus* 的生活情景剧，因此将这种全新的程序设计语言命名为 Python。早期世界上 Python 的其他开发爱好者通过邮件列表与吉多进行交流或对其提供建议。不同领域的 Python 使用者根据自身需求对 Python 功能进行了不同的扩展，他们会把自己改进的模块邮件列表发给吉多，由他来决定是否加入该特性或者模块。

随着 Python 的影响力越来越大以及互联网的兴起，信息交流的途径更加方便，于是有了开源这种新的软件开发模式，即将程序代码公布到网络上，由所有研究人员共同开发改进。

2000 年 10 月，Python 2.0 发布。Python 从基于邮件列表的开发方式转为完全开源的开发方式，此时吉多只负责大的框架的制定，至于实现细节，则交给由全世界最优秀的

Python 开发者组成的 Python 社区。Python 有今天的影响力，Python 社区功不可没，但吉多对于 Python 仍然具有绝对的仲裁权。

2008 年 12 月，Python 3.0 发布。Python 3.0 版本在语法和解释器内部都做了重大改进，解释器内部完全采用面向对象的方式实现。Python 3.0 与 Python 2.x 系列不兼容，这使得很多利用 Python 2.x 编写的程序无法在 Python 3.x 的环境下有效运行。

在 2008 年到 2015 年相当长的一段时间里，Python 语言的发展受到了一定的制约。但是到了今天，所有 Python 主流的、最重要的程序都在 Python 3.x 上运行，且国际上最重要的 Python 程序员也在使用 Python 3.x 进行编程，这为 Python 未来的发展提供了非常有力的支持。

1.1.2 了解 Python

Python 的创始人吉多是一位计算机程序员，同时他在数学方面也有着很深的造诣。从设计的哲学上来说，由于吉多经历过数学方面的专业训练，因此他创立的语言具有高度统一性，语法、格式和工具集也都具有一致性。Python 是自由开源软件之一，用户可以自由下载、复制、阅读或修改代码，并可自由发布修改后的代码。这也是相当一部分用户热衷于改进和优化 Python 的原因。

1.1.3 Python 的特点

1. Python 的优点

（1）语言简洁。Python 主要用来精确表达问题逻辑，更接近自然语言，只有 33 个保留字，十分简洁。

（2）语法优美。用一行代码生成一个列表，简洁明了。

（3）易学易用。Python 极易上手，因为 Python 有极其简单的说明文档。

（4）可移植性好。由于开源的本质，Python 已经被移植在许多平台上（经过修改后能够工作在不同平台上），这些平台包括 Linux、Windows、Mac FreeBSD 和 Solaris 等。

（5）扩展性好。Python 提供了丰富的标准库，可以满足各种编程场景的需求，如数据分析与挖掘、图像处理和网络爬虫等。

（6）类库丰富。Python 解释器拥有丰富的内置类和函数库，并且世界各地的程序员通过开源社区贡献了十几万个覆盖各应用领域的第三方库。

（7）模式多样。支持面向对象编程（使用类和对象来组织代码）。

（8）通用灵活。Python 几乎可以用于任何与程序设计相关应用的开发。

（9）良好的中文支持。Python 3 默认字符集和编码方式都是 UTF-8，因此可以直接支持和处理中文字符串。对于 Python 2.x 版本，其默认的编码格式是 ASCII，这意味着如果不进行任何设置，则该版本无法正确处理中文，但只要在文件开头加入“# -*- coding: UTF-8 -*-”或者“# coding=utf-8”，就可实现对中文的支持。

注意：“# coding=utf-8”等号两边不要加空格。Python 还提供了许多内置的汉字处理方法，例如，len() 函数可以计算字符串的长度；str.encode() 函数可以将字符串编码为字节串；str.decode() 函数可以将字节串解码为字符串等。

2. Python 的缺点

（1）执行效率不够高，Python 程序没有 C++、Java 编写的程序高效。

（2）Python 3.x 和 Python 2.x 不兼容。

1.1.4 Python 的主要应用领域

Python 拥有着许多优质的文档和丰富的库，对科学用途的编程任务都是非常有用的，其领域包括但不限于以下几个方面。

（1）Web 开发。在网站开发方面，Python 具有 Django、Flask、Pyramid、Bottle、Tornado 和 Web2py 等框架，使用 Python 开发的网站具有小而精的特点。知乎、豆瓣等应用都是使用 Python 开发的。

（2）网络爬虫。网络爬虫又称网络蜘蛛，是指按照某种规则在网络上抓取所需内容的脚本程序。Python 自带的 urllib 库、第三方的 requests 库和 Scrapy 框架让爬虫开发变得非常容易。

（3）人工智能。虽然可以使用各种不同的编程语言开发人工智能程序，但是 Python 在人工智能领域具有独特的优势。在人工智能领域，有许多基于 Python 的第三方库，如 Scikit-learn、Keras 和 NLTK 等。其中，Scikit-learn 是基于 Python 的机器学习工具，提供了简单高效的数据挖掘和数据分析功能；Keras 是一个基于 Python 的深度学习库，提供了用 Python 编写的高级神经网络应用程序编程接口；NLTK 是 Python 自然语言工具包，用于完成标记化、词形还原、词干化、解析、词性标注等任务。此外，深度学习框架 TensorFlow、Caffe 等的主体都是用 Python 实现的，提供的原生接口也是面向 Python 的。

（4）数据分析。Python 被广泛应用于数据科学领域。在数据采集环节，在 Python 第三方库 Scrapy 的支持下，可以编写网络爬虫程序采集网页数据。在数据清洗环节，第三方库 Pandas 提供了功能强大的类库，可以对数据进行清洗、排序，最后得到清晰的数据。在数据处理分析环节，第三方库 NumPy 和 SciPy 提供了丰富的科学计算和数据分析功能，包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像图例、常微分方程求解、矩阵解析和概率分布等。在数据可视化环节，第三方库 Matplotlib 提供了丰富的数据可视化图表。

（5）自动化运维。随着技术的进步和业务需求的快速增长，一个运维人员通常要管理成百上千台服务器，运维工作也变得重复和繁杂。Python 作为运维工程师首选的编程语言，通过自动化运维，能够将运维人员从复杂的服务器的管理工作中解放出来，使运维工作变得简单、快速和准确。在很多操作系统中，Python 是标准的系统组件。大多数 Linux 发行版和 macOS 都集成了 Python，可以在终端下直接运行 Python。Python 标准库包含了多个调用操作系统功能的库。通过第三方软件包 Pywin32，Python 能够访问 Windows 的 COM 服务及其他 Windows API。使用 IronPython，Python 程序能够直接调用 .NET Framework。一般说来，Python 编写的系统管理脚本在可读性、代码可重用性和扩展性等方面都优于普通的 Shell 脚本。

（6）Python 在科学计算、游戏开发、多媒体应用、图像处理、工业设计和密码学等领域都得到了广泛应用。

1.2 编译的概念和分类

1.2.1 编译器的概念

计算机语言分为高级语言、汇编语言和机器语言。高级语言便于人们编写、阅读交流和维护；汇编语言是一种直接与硬件关联、能够提供对硬件直接控制力的低级语言；机器语言是计算机能直接解读、运行的语言。编译器是将汇编语言或高级语言源程序作为输入，翻译成目标语言、机器代码的等价程序。简单来讲，编译器就是将“一种语言（通常为高级语言）”翻译为“另一种语言（通常为低级语言）”的程序。

1.2.2 计算机语言的编译分类

计算机语言根据编译方式不同，又分为编译型语言和解释型语言。

编译型语言的特征是先将源代码编译成机器语言，再由机器运行机器码，也就是二进制代码。这个过程是在程序运行之前进行，而程序正式运行时，就不需要再次编译，直接使用已经编译好的结果即可。因此编译型语言的代码执行效率非常高，但是其编写效率以及跨平台性能较差。常见的编译型语言有 C、C++、Delphi 等。

解释型语言不需要在程序运行前进行集中编译，而是在程序运行过程中，由专门的解释器负责每个语句的解释并执行代码。因此解释型编程语言代码在执行过程中，每执行一次就需要编译一次，相对编译型语言来说，其效率比较低。解释型语言就是用效率换取开发速度，从而实现相同的功能，解释型语言的代码量和编码速度要优于编译型语言。

1.3 Python 解 释 器

1.3.1 Python 解释器概述

Python 是一门解释型编程语言，在编程过程中生成的 .py 文件需要解释器才能正常执行。目前基于不同的平台，Python 的解释器出现了多种不同的版本，分别使用相应平台的编程语言开发的解释器。目前常见的 Python 解释器包括 CPython、Jython、IPython、PyPy 和 IronPython 五个版本。

CPython 是使用 C 语言开发的 Python 解释器，也是标准的 Python 解释器，目前使用最为广泛。

1.3.2 Python 解释器的安装

登录 Python 官方网站，编者以 Windows 系统为例演示 Python 解释器的下载与安装，选择 Downloads，如图 1.1 所示。单击 Windows 跳转到 Python 下载页面，如图 1.2 所示，该下载页面中包含多个版本的安装包，可根据自身需求下载相应的版本。这里以下载 Python 3.9.13 64 位的离线安装包为例来演示 Python 解释器的安装方法，如图 1.3 和图 1.4 所示。

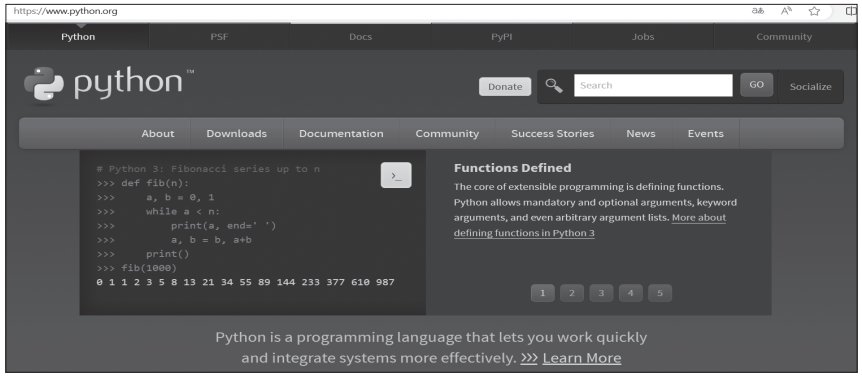

图 1.1 Python 官网界面

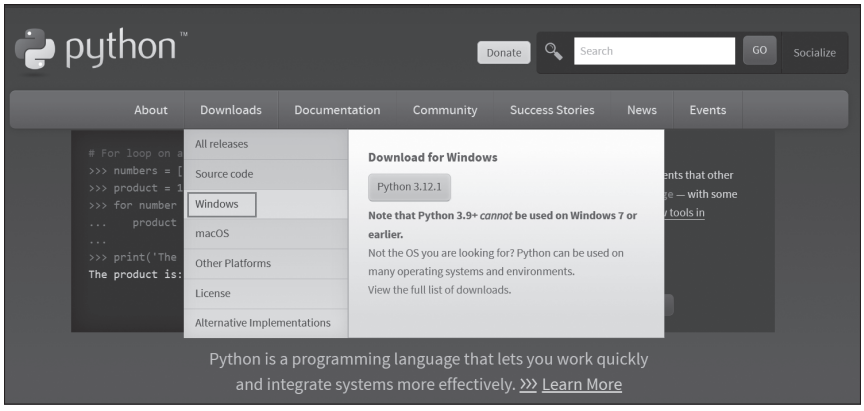

图 1.2 选择 Windows

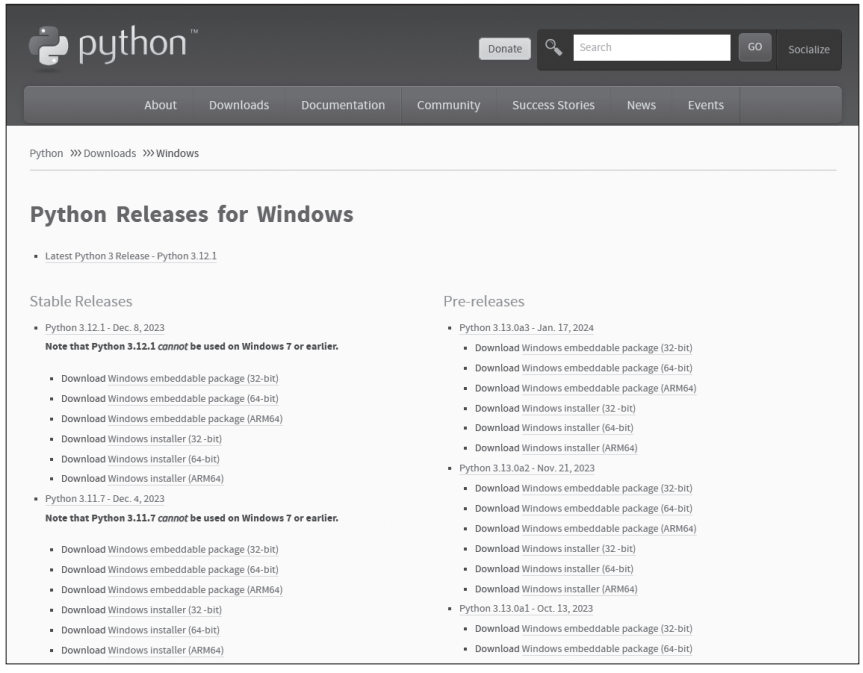

图 1.3 多个版本的安装包(以上只显示其中一部分)

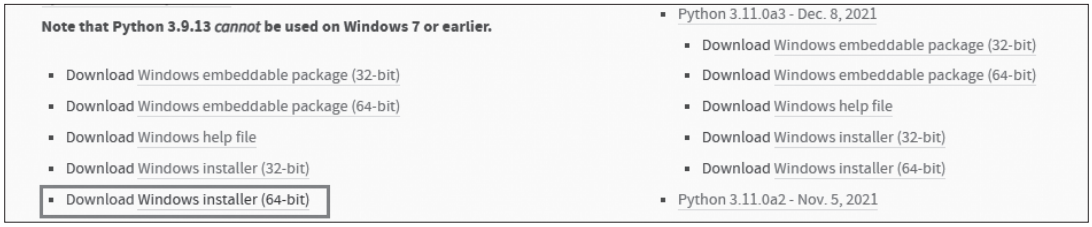

图 1.4　选择 Python 3.9.13 64 位离线安装包

下载完成后双击 Python 安装包进行安装，并保持默认配置，选择 Python 的安装路径后，单击 Install Now 按钮开始安装，如图 1.5 所示。安装成功后显示 Setup was successful，如图 1.6 所示。

图 1.5　开始安装

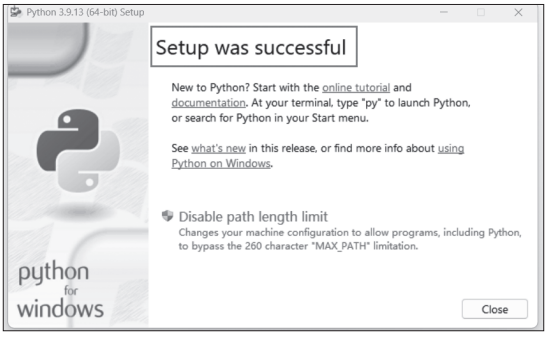

图 1.6　安装成功

按 Win+R 组合键进入运行界面，输入 cmd，按回车键，进入 Windows 命令提示符窗口。在命令提示符窗口输入 python 可以进入 Python 3.9.13 的命令行模式，如果出现如图 1.7 所示安装的 Python 版本，就说明 Python 3.9.13 安装成功。

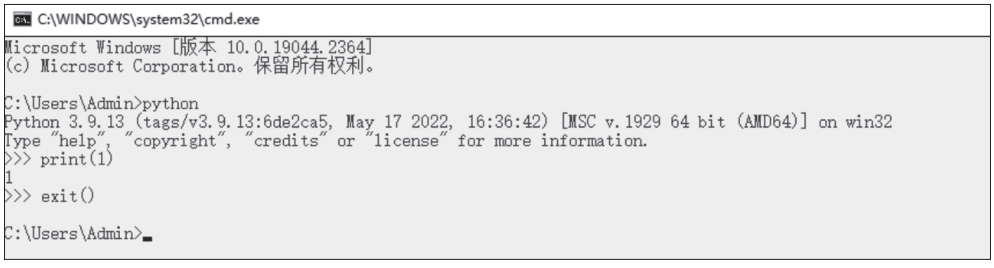

图 1.7　测试是否安装成功

在 Windows 系统中打开 Python 开发环境 IDLE(Python 3.9 64-bit),如图 1.8 所示。演示输出“hello world”,初步了解程序的运行方式,如图 1.9 和图 1.10 所示。

图 1.8 在 Windows 系统中打开 Python 开发环境 IDLE

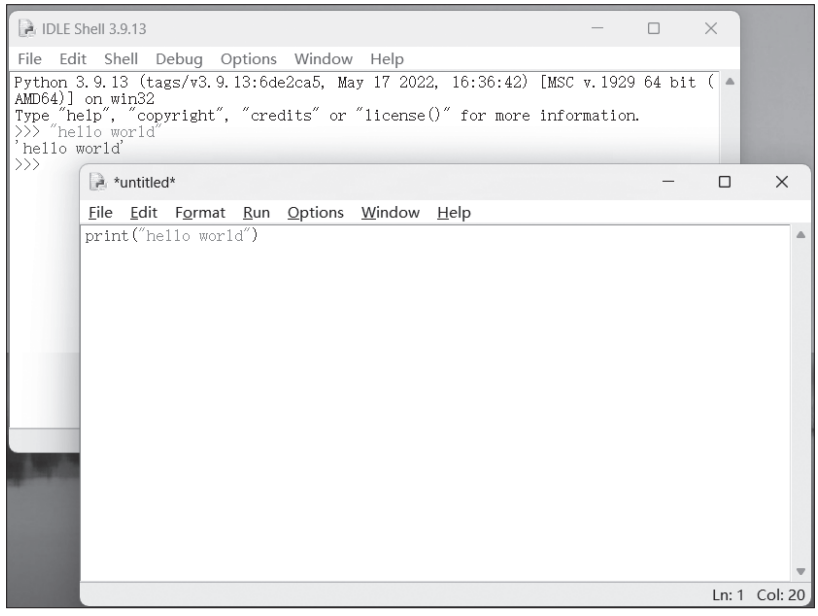

图 1.9 输出“hello world”

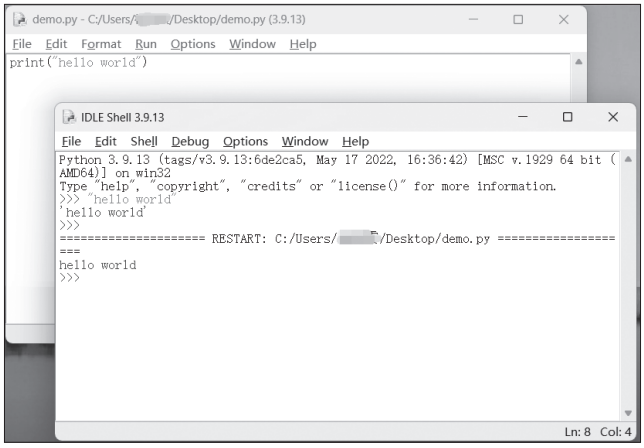

图 1.10　运行程序“hello world”

1.4　两种运行 Python 程序的方式

Python 程序有两种运行方式：交互式和文件式。交互式是指 Python 解释器逐行接收 Python 代码并即时响应；文件式也称批量式，指先将 Python 代码保存在文件中，再启动 Python 解释器批量解释代码。

1.4.1　交互式

打开 Python 开发环境，进入 IDLE 界面，如图 1.11 所示。

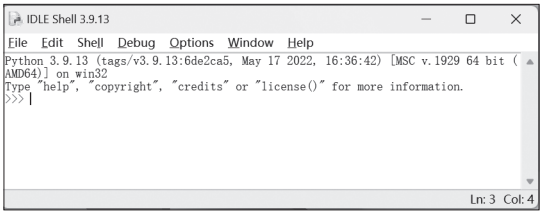

图 1.11　进入 IDLE 界面

下面以实践示例“天天向上的力量”为例演示交互式运行方式。Python 解释器逐行接收 Python 代码并即时响应。假设每天的变化量是 0.005，如图 1.12 所示。

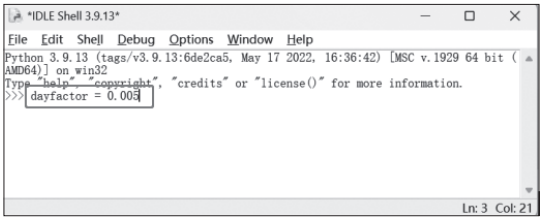

图 1.12　每天的变化量

如果每天进步 0.005，那么一年 365 天的累计进步如图 1.13 所示。

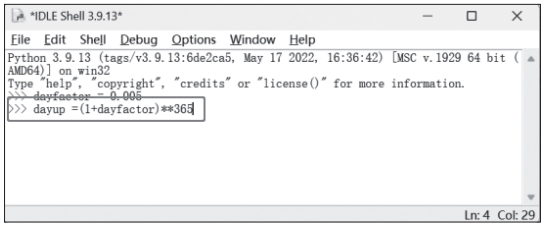

图 1.13 365 天的累计进步

如果每天退步 0.005,那么一年 365 天的累计退步如图 1.14 所示。

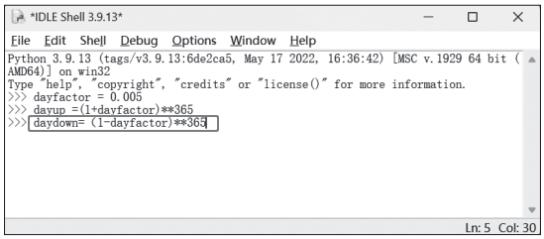

图 1.14 365 天的累计退步

通过 print 函数输出结果,如图 1.15 所示。

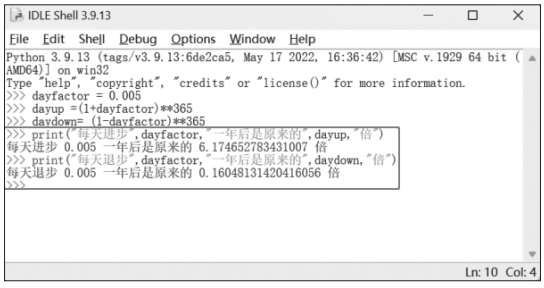

图 1.15 函数输出结果

1.4.2 文件式

如图 1.16 所示,选择 File 菜单中的 New File 下拉选项,新建文件,在其中写入刚才逐行输入的代码,如图 1.17 所示。

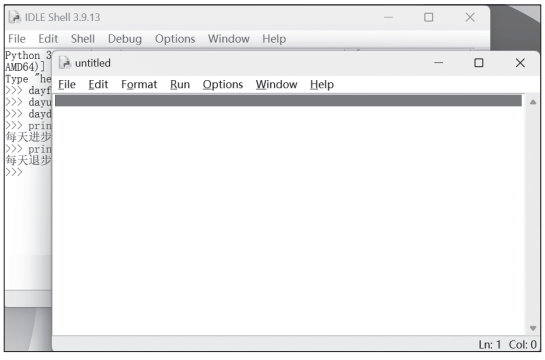

图 1.16 新建文件

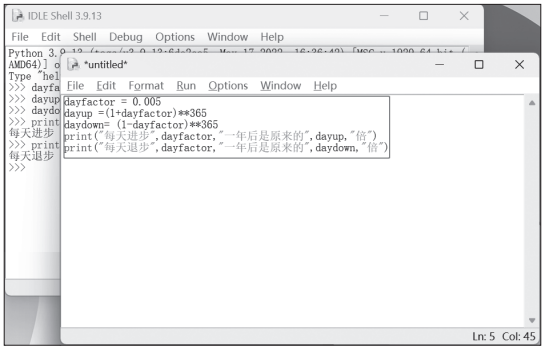

图 1.17　输入开源代码

如图 1.18 所示，完成代码输入后保存该文件，文件名为 demo1，然后运行输出结果，单击 Run 菜单下方的 Run Module 选项，如图 1.19 所示。

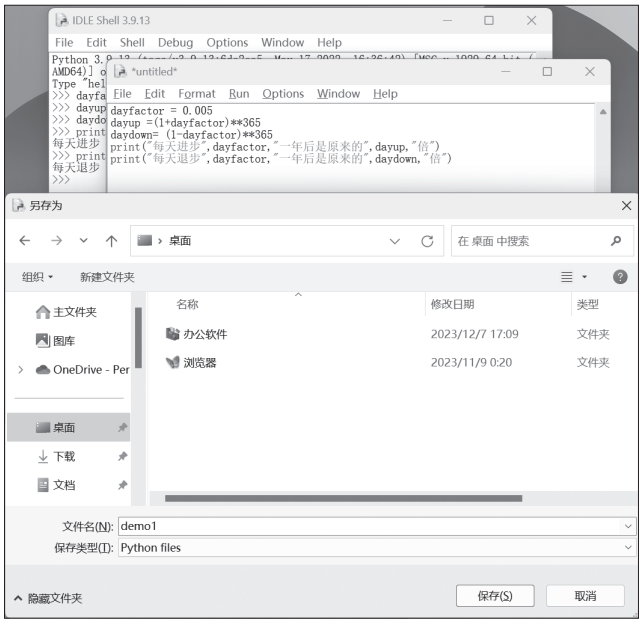

图 1.18　保存文件

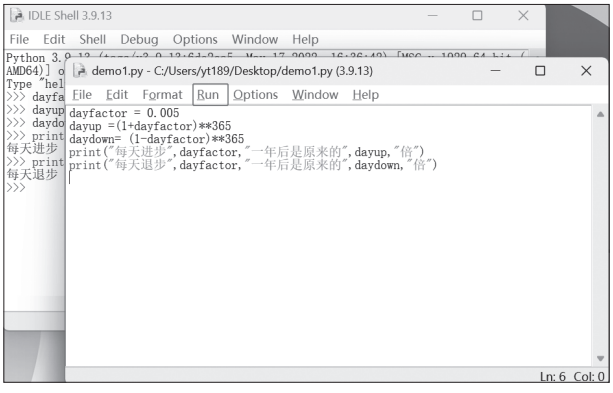

图 1.19　运行输出结果

通过刚才的运行结果，看到了天天向上的力量和坚持不懈的价值，如图 1.20 所示。

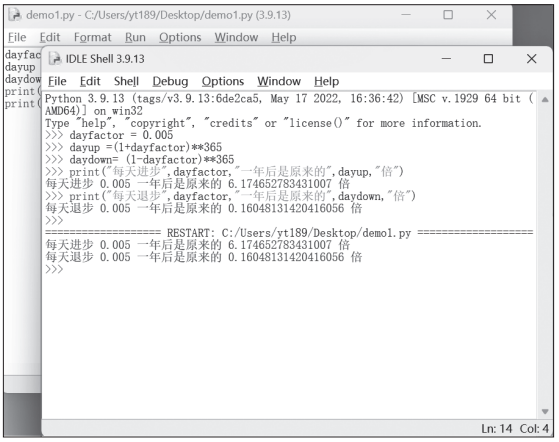

图 1.20 天天向上的力量

1.5 了解和安装 PyCharm

1.5.1 PyCharm 概述

PyCharm 是 JetBrains 公司开发的一款 Python 集成开发环境，由于其具有智能代码编辑器、智能提示和自动导入等功能，目前已经成为 Python 专业开发人员和初学者广泛使用的 Python 开发工具。

1.5.2 PyCharm 安装

访问 JetBrains 官网，下载 PyCharm 工具，如图 1.21 所示。

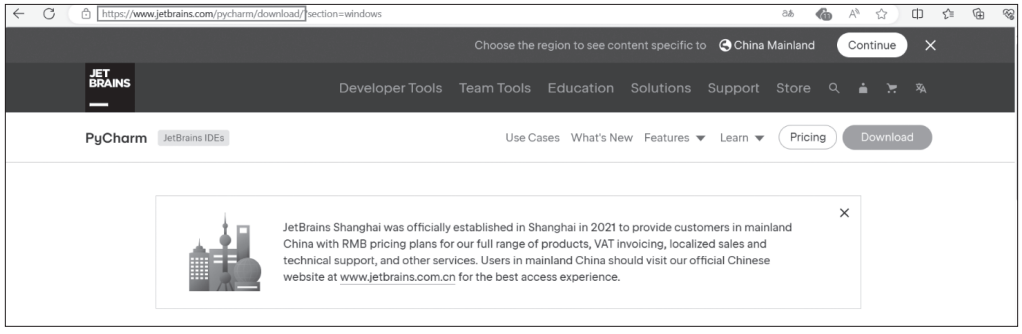

图 1.21 JetBrains 官网

PyCharm 分为 Professional（专业版）和 Community Edition（社区版）两个版本，分别如图 1.22 和图 1.23 所示。

单击相应版本下的 Download 按钮，下载 PyCharm 安装包，这里下载 Community Edition 版本。

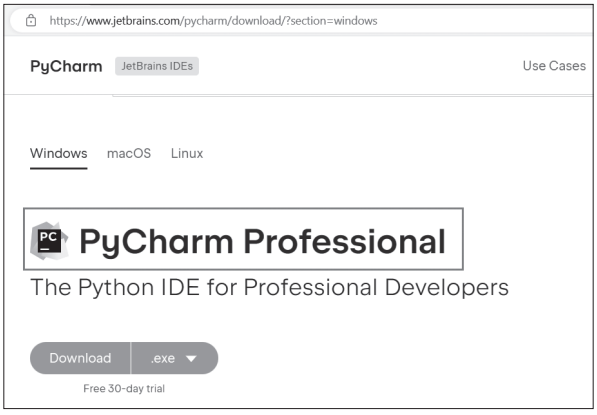

图 1.22　PyCharmProfessional(专业版)

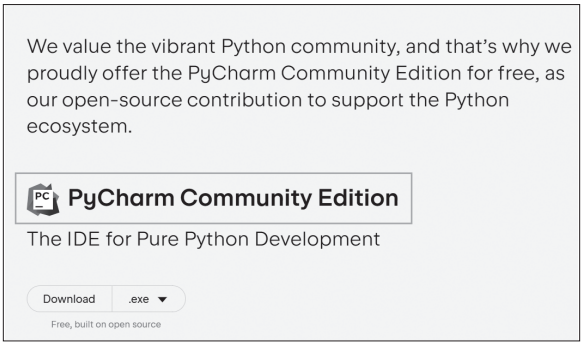

图 1.23　PyCharm Community Edition(社区版)

下载成功后双击 PyCharm 安装包，弹出欢迎界面，单击“下一步”进入 PyCharm 选择安装路径的界面，如图 1.24 所示。确定安装路径，单击“下一步”，根据需求勾选安装选项。再单击“下一步”，保持默认配置，然后单击“安装”，PyCharm 安装完成后弹出提示信息，最后单击“完成”关闭页面，如图 1.25 ～图 1.28 所示。

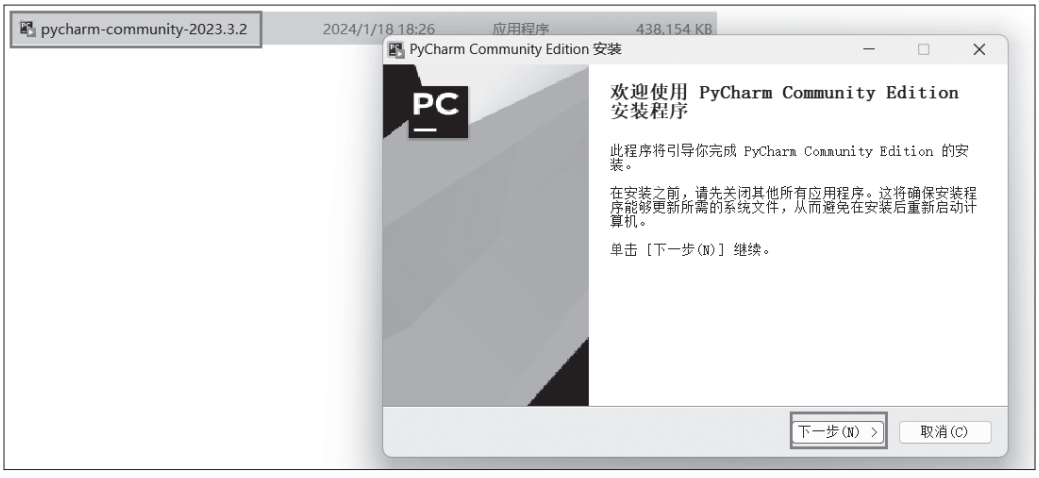

图 1.24　PyCharm 安装界面

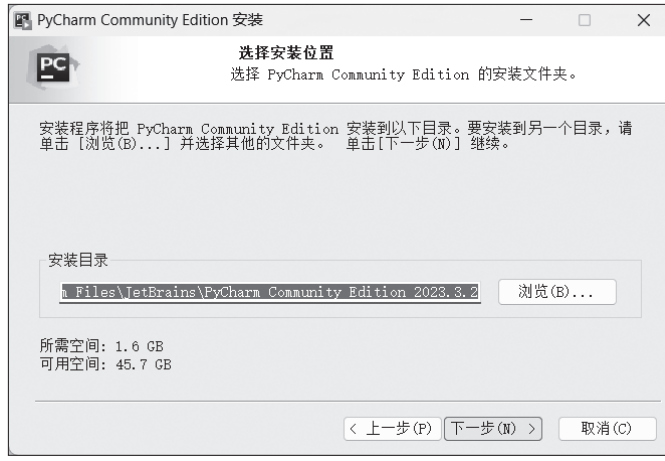

图 1.25 选择安装路径

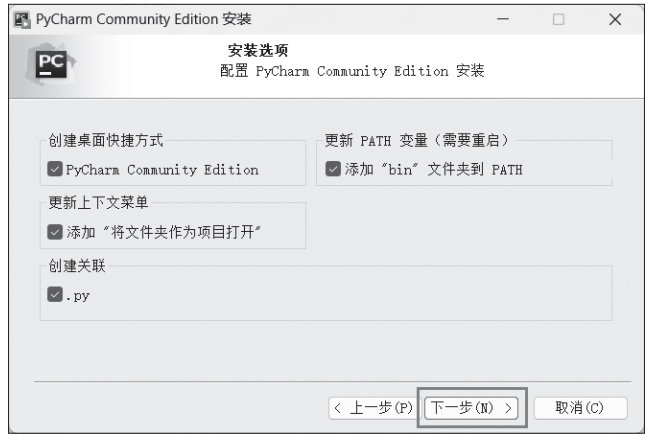

图 1.26 勾选安装选项

图 1.27 保持默认配置

图 1.28　安装完成界面

安装完成后，双击 PyCharm 快捷图标打开 PyCharm 进入导入配置文件的界面。这里选择 Do not import settings 选项，单击 OK 按钮，如图 1.29 所示。

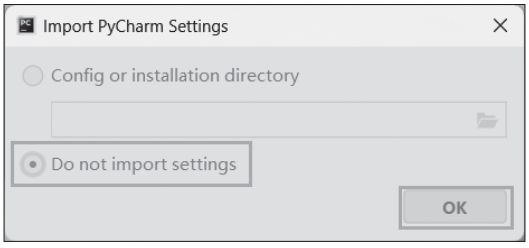

图 1.29　导入配置文件界面

进入 IDLE 运行环境后，单击 New Project 按钮创建一个 Python 项目如图 1.30 和图 1.31 所示。单击 Create 按钮即可完成一个新项目的创建。

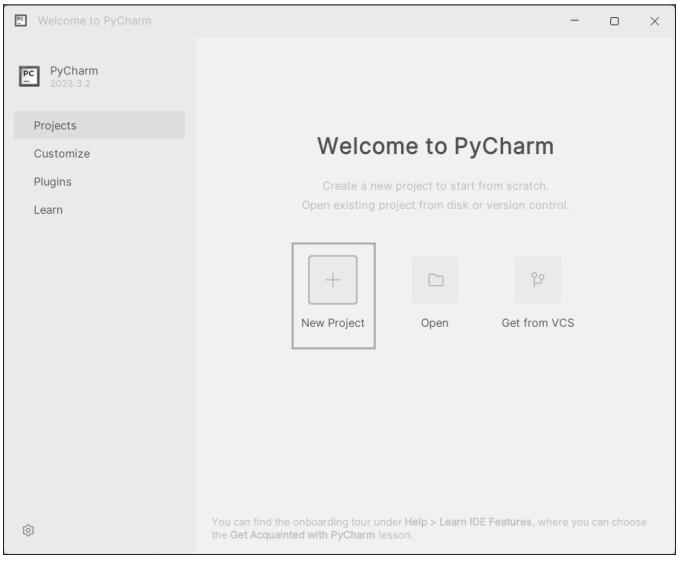

图 1.30　单击 New Project 按钮

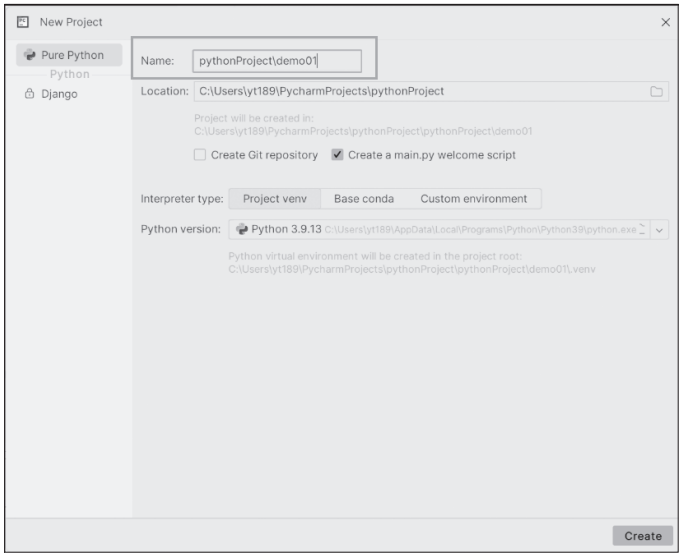

图 1.31 创建一个 Python 项目

习 题

1. 下列选项中不是 Python 语言特点的是(　　)。

A. 简洁　　B. 开源

C. 面向过程　　D. 可移植

2. 什么是编译型语言？什么是解释型语言？两者有什么区别？

3. 请简述交互式和文件式的特点。

4. 在 Python 官网上下载 Python 3.7 以上版本文件并正确安装，在交互式解释器中输入一些命令，体会其运行效果。

5. 进入 JetBrains 官方下载页面，下载 PyCharm 社区版（Community）并正确安装，新建项目后再新建一个 Python 文件，体会其运行效果。

6. 双击安装的 IDLE 启动 Python 运行环境，Python 3.x 环境中通过交互方式运行下列 Python 命令，观察运行结果。

（1）print("Hello, 欢迎来到 Python 的世界！ ")

（2）name = input(" 请输入你的姓名：")

（3）hometown = input(" 请输入你的家乡：")

（4）print(" 大家好！ 我是来自 {} 的 {}。".format(college,name))

7. 同样在 IDLE 运行环境下，新建文件 choname.py，文件内容如下：

```
name = input(" 请输入你的姓名： ")
age = input(" 请输入你的年龄： ")
Print("{} 岁的 {} 同学， 人生苦短， 学好 Python ！ ".format(age,name))
```