高等院校产教融合创新应用系列

Python 程序设计基础与应用

河南打造前程科技有限公司　主编

清華大學出版社
北　京

内 容 简 介

本书是一本 Python 编程语言的入门级教材，旨在系统地介绍 Python 编程语言，从而让读者掌握 Python 编程语言的核心知识和实用技能。全书共 10 章，内容涵盖了 Python 语言的特点、编程环境搭建、Python 基础语法、流程控制和异常处理、高级数据结构、面向对象编程、文件和文件夹操作、数据库编程等多个方面。

本书不仅注重理论，更着眼于实际应用，通过案例动手操作，帮助读者快速掌握 Python 编程的精髓。本书可作为高等院校计算机、信息技术、人工智能及相关专业程序设计语言课程的教材，也可作为 Python 语言初学者的参考书。

图书在版编目(CIP)数据

Python 程序设计基础与应用 / 河南打造前程科技有限公司主编. —北京：清华大学出版社，2024.3
高等院校产教融合创新应用系列
ISBN 978-7-302-65482-7

I. ①P… II. ①河… III. ①软件工具—程序设计—高等学校—教材 IV. ①TP311.561

中国国家版本馆 CIP 数据核字(2024)第 019958 号

责任编辑：王 定
封面设计：周晓亮
版式设计：孔祥峰
责任校对：马遥遥
责任印制：曹婉颖

出版发行：清华大学出版社
网 址：https://www.tup.com.cn，https://www.wqxuetang.com
地 址：北京清华大学学研大厦 A 座 **邮 编：**100084
社 总 机：010-83470000 **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：三河市科茂嘉荣印务有限公司
经 销：全国新华书店
开 本：185mm×260mm **印 张：**16.75 **字 数：**450 千字
版 次：2024 年 3 月第 1 版 **印 次：**2024 年 3 月第 1 次印刷
定 价：69.80 元

产品编号：103258-01

前　言

党的二十大报告明确指出：“教育、科技、人才是全面建设社会主义现代化国家的基础性、战略性支撑。必须坚持科技是第一生产力、人才是第一资源、创新是第一动力，深入实施科教兴国战略、人才强国战略、创新驱动发展战略，开辟发展新领域新赛道，不断塑造发展新动能新优势。”“加快发展数字经济，促进数字经济和实体经济深度融合，打造具有国际竞争力的数字产业集群。”数字经济的崛起与繁荣，为经济社会发展赋予了“新领域、新赛道”和“新动能、新优势”，正在成为引领中国经济增长和社会发展的重要力量。

随着人工智能、物联网、云计算等技术的快速发展与广泛应用，世界正在经历一场数字化变革。当今，人工智能已经渗透到各领域。随着算法和计算机硬件的不断提升，人工智能的应用范围和领域也在不断拓展。学习 Python 编程语言，是适应时代发展的需要，也是占领数字领域的重要一步。作为基础性的编程语言之一，Python 语言因其简单易学、功能强大和生态完整等优势，成了当今时代热门的编程语言之一。而采用 Python 编程的应用领域也越来越广泛，如数据科学、机器学习、深度学习、自然语言处理领域等。

人工智能相关专业的高校教学体系配置过多地偏向理论教学，课程设置与企业实际应用契合度不高，学生很难把理论转化为实践应用技能。为此，我们针对软件开发、网络编程、数据分析、人工智能等领域编写这本《Python 程序设计基础与应用》，以帮助学生将理论能力转化为实践能力。

本书内容由浅入深、适合初学者学习 Python 编程语言。本书旨在系统地介绍 Python 编程语言，从而让读者掌握 Python 编程语言的核心知识和实用技能。全书共分 10 章，内容涵盖 Python 语言概述、基础语法、流程控制、高级数据结构、面向对象编程、文件和文件夹操作、数据库编程等。本书不仅注重理论，更着眼于实际应用，通过设置案例及练习题，帮助读者快速掌握 Python 编程的精髓。

本书内容组织具体如下。

第 1 章介绍了 Python 语言的特点、编程环境搭建，并通过案例实现了一个简单的图形输出。

第 2 章介绍了 Python 语言的基本元素，包括标识符、关键字、变量、各种数据类型、运算符，以及数据的输入与输出。

第 3 章介绍了流程控制和异常处理。其具体包括选择结构设计、循环结构设计、循环跳转及异常处理等内容，通过一个实际案例——猜拳游戏，展示了如何运用这些概念和技巧。

第 4 章详细介绍了 Python 中的高级数据结构，包括列表、元组、字典、集合和切片的使用内容，通过案例——用户管理系统进行了应用演示。其主要内容包括技术点综合运用、程序逻辑思维提升。

第 5 章介绍了 Python 中的正则表达式，讲解了正则表达式语法、re 模块方法的使用，以及正则表达式对象、子模式及 match 对象的使用等，同时也给出了正则表达式在实际应用中的例子。

第 6 章详细介绍了 Python 中函数的各个方面。其主要内容包括函数的定义、调用、参数的默认值、可变参数、命名空间和作用域、高阶函数、匿名函数、生成器和装饰器等。通过一个实际案

例——自动售货机，展示了如何灵活运用这些概念和技巧来构建一个完整的自动售货机系统。学生通过对本章内容的深入学习，可以更好地理解函数的重要性，并提升在 Python 编程中使用函数的能力。

第 7 章介绍了 Python 中的面向对象编程，包括类的定义、对象的创建、成员变量、构造方法、实例方法、类变量、类方法、封装性、继承性和多态性等内容，通过一个实际案例，展示了如何使用 OOP 实现一个点餐系统。

第 8 章介绍了 Python 中对文件和文件夹的操作，包括文本文件、结构化的文本文件的读取和写入、二进制数据的处理。

第 9 章介绍了数据库编程，通过数据库驱动 pymysql 模块实现对 MySQL 数据库连接操作，以及使用数据库连接池提高运行效率。

第 10 章主要介绍了 Python 计算生态，包括内置标准库中的随机库、时间和日期库、绘图库，以及第三方库的使用，同时也通过案例演示了文本处理、图像处理、分词、构造词云等内容。

本书的特色如下。

系统化的学习方式：本书按照基础语法到实际应用的递进方式，介绍了 Python 编程的基础知识和各种应用场景，让读者从简单到复杂，逐步掌握 Python 编程的核心内容。

实用性强的案例：本书重视实际应用，通过编写大量的实例，让读者更好地理解 Python 编程思想和方法，并能在实际应用中灵活运用。

深入浅出的讲解：本书采用通俗易懂的语言，结合丰富的代码实例，让读者更加深入地理解 Python 编程语言中的各个知识点和应用场景。

希望本书能够为初学者和有编程经验的人员提供一份简单明了、易于掌握、实用丰富的 Python 编程学习资料，帮助他们建立扎实的 Python 编程基础，快速掌握 Python 编程技能，逐步成为 Python 编程专家；了解 Python 在人工智能领域的应用，通过实践和项目经验不断提升自己的编程水平，在未来的编程领域中有更大的发展。

本书的适用对象如下。

- Python 零基础的读者。
- 数据分析应用的开发人员。
- 开设有数据分析相关课程的高校教师和学生。
- 开设人工智能专业课程的高校教师和学生。

本书免费提供教学课件、教学大纲、教学视频、案例源代码、习题参考答案等教学资源，读者可扫描下列二维码获取。


教学课件


教学大纲


教学视频


案例源代码


习题参考答案

编者

2023 年 12 月

目 录

第1章 Python 语言概述

Python 是一种高级编程语言，是当前应用较为广泛的计算机语言之一，具有简洁、易读和强大的特点。它于 1991 年由 Guido van Rossum 开发，被广泛应用于 Web 开发、数据科学、人工智能、网络爬虫、自动化脚本、游戏开发等领域。本章将介绍 Python 语言的发展历史、主流版本、语言特点、Python 程序的执行原理、Python 环境的搭建、开发工具的使用，以及体验 Python 的趣味程序等内容，以帮助读者快速地了解 Python。

学习目标

- 了解 Python 的历史和起源
- 理解 Python 的特点和优势
- 认知 Python 的版本
- 掌握 Python 的安装和软件配置方法
- 开发环境的安装与体验

1.1 走近 Python

本节主要介绍 Python 的发展历史、Python 的版本、Python 语言的特点、解释型语言与编译型语言的区别，以及 Python 程序的执行原理等内容。

1.1.1 Python 的发展历史

首先，通过一个真实的故事来了解 Python 语言的诞生。

在计算机的世界里，有一个名叫 Guido van Rossum(以下简称 Guido)的荷兰程序员。当时，Guido 在阿姆斯特丹的荷兰数学与计算机科学研究所(CWI)工作。他是一个富有创造力和激情的人，总是在寻找一种简单而强大的编程语言。

于是，Guido 开始了他的创造之旅。他以 ABC 语言为基础，参考了 Modula-3 和其他一些编程语言的特点，并加入了自己的创新思想，最终设计出了一种新的语言。他给这个语言取名为 Python，以纪念他最喜欢的电视剧《蒙提·派森的飞行马戏团》。

Python 语言的设计理念是“优雅”和“明确”，即使用尽可能简洁的语法表达尽可能多的功能。Guido 希望开发者能够使用最少的代码完成更多的工作，同时保持代码的可读性。

在 1991 年，Guido 发布了 Python 的第一个版本 Python 0.9.0。该版本具备了基本的语法和一些常用的功能。在设计过程中，他注重简洁和可读性，遵循了“一切皆对象”的原则。他还为 Python 引入了缩进风格的代码块表示法，这种风格通过代码的缩进来表示程序的结构，提高了代码的可读性。

随着 Python 逐渐发展壮大，不断吸引更多的开发者加入其中。Guido 也在社区的帮助下不断改进和扩展 Python，融入了模块化、面向对象和动态类型等特性，使 Python 变得更加灵活和强大。随着时间的推移，Python 的应用范围不断扩大，并拥有了庞大的生态系统，包括各种生态库、框架和工具，为开发者提供了丰富的资源和支持。

1.1.2 Python 版本认知

Python 有许多不同的版本，每个版本都有其自身的特点。以下是 Python 的一些主要版本及其特点的简要说明。

(1) Python 1.x 系列：Python 的最早版本，于 1994 年发布。这个系列的版本具有基本的语言功能，如模块、函数、类等，但在功能和性能方面相对较简单。

(2) Python 2.x 系列：Python 的第一个广泛采用的版本，于 2000 年发布。其中，最具影响力的版本是 Python 2.7，在很长一段时间内被广泛使用。Python 2.x 系列存在一些向后不兼容的变化，导致在迁移代码时需要进行一些调整。这个系列的维护于 2020 年 1 月停止，不再更新。

(3) Python 3.x 系列：是当前推荐的版本，是未来发展的重点。Python 3.x 于 2008 年发布，解决了 Python 2.x 系列中的一些设计缺陷和不一致性。它引入了许多新特性和改进，同时移除了一些旧的特性。Python 3.x 系列是向后不兼容的，因此需要对现有代码进行修改以适应新的语法和功能。

在 Python 3.x 系列中，每个小版本(如 Python 3.1、Python 3.2 等)都引入了一些改进和修复。其中，最重要的版本是 Python 3.5、Python 3.6、Python 3.7、Python 3.8、Python 3.9 和最新的 Python 3.11，这些版本都增加了新的语言特性、改进了标准库并且优化了性能。

值得注意的是，Python 的不同版本在语法和功能上可能会有一些差异。因此，开发者应该根据项目的需求选择合适的 Python 版本，并确保代码与目标版本兼容。

除官方版本外，还有一些基于 Python 的替代实现，如 Jython(Python 运行在 Java 虚拟机上)、IronPython(Python 运行在.NET 平台上)和 PyPy(Python 的 Just-In-Time 编译实现)，它们在特定的场景下具有一些优势和特色。

总结起来，Python 的版本演化表明了 Python 社区的不断发展和改进，以提供更好的编程体验和功能。对于新的项目，本书推荐使用最新的 Python 3.x 系列版本。对于已有的项目，可以考虑迁移到 Python 3.x 系列，并根据需求选择合适的 Python 版本。

1.1.3　Python 语言的特点

Python 作为一种解释型、面向对象的高级编程语言，相比其他语言会有许多优点和一些限制。以下是对 Python 语言的优缺点的详细解释。

1. Python 语言的优点

(1) 简洁易读：Python 具有简洁、清晰的语法，使用缩进来表示代码块，使代码易于阅读和理解。这使初学者能够更快地上手，并且提高了代码的可读性和可维护性。

(2) 大量的库和生态系统：Python 拥有广泛且活跃的库和生态系统。这些库涵盖了多个领域，包括网络编程、数据分析、机器学习、Web 开发等，使开发人员能够快速构建应用程序，并且能够借助已有的解决方案和工具提高开发效率。

(3) 跨平台性：Python 可以在多个操作系统上运行，包括 Windows、macOS 和各种 Linux 发行版。这使开发人员可以在不同的平台上开发和部署他们的应用程序，提供了更大的灵活性和可移植性。

(4) 动态类型和自动内存管理：Python 是一种动态类型语言，不需要开发人员在编写代码时指定变量类型，使开发工作变得更加灵活。此外，Python 还具有自动内存管理机制，开发人员无须手动进行内存分配和释放，降低了内存管理的复杂性。

(5) 强大的科学计算和数据分析支持：Python 拥有丰富的科学计算和数据分析库，如 NumPy、Pandas 和 Matplotlib 等。这使得 Python 成为进行数据处理、可视化和机器学习等操作的首选语言。

(6) 社区支持和活跃度：Python 拥有庞大且活跃的社区，开发人员可以获得来自社区的支持、解决方案和学习资源。Python 社区提供了各种教程、文档和开源项目，使开发人员能够更好地学习和分享经验。

2. Python 语言的缺点

(1) 运行速度相对较慢：与一些编译型语言相比，Python 在执行速度上相对较慢。这是因为 Python 是解释型语言，需要在运行时解释代码，而不是提前编译成机器代码。尽管如此，Python 提供了一些工具和方法来优化性能。

(2) 全局解释器锁(GIL)：Python 的标准解释器(CPython)使用 GIL，这限制了在多线程场景中的并行执行能力。由于 GIL 的存在，Python 的多线程并不能实现真正的并行，而只是通过线程之间的切换来模拟并发。这对于某些 CPU 密集型任务来说可能会造成性能瓶颈。

(3) 资源消耗较高：相比一些低级语言，如 C 或 C++，Python 在处理大规模数据和高并发请求时可能消耗较多的系统资源，由于 Python 的动态类型和自动内存管理机制会带来一些开销，故资源消耗较高。

(4) 移动端开发限制：尽管 Python 有一些用于移动应用程序开发的框架(如 Kivy、PyQt 等)，但与一些原生移动开发语言(如 Java 和 Swift)相比，Python 在移动端开发方面的生态系统相对较小，并且性能和访问原生 API 的能力有限。

总体而言，Python 是一种功能强大且易于学习的编程语言，适用于各种应用场景。然而，开发人员在选择 Python 时，应该考虑其性能和资源消耗的限制，并选择适当的优化方法。

1.1.4 解释型语言和编译型语言的区别

前文已经提到，Python 之所以运行速度慢，是因为 Python 属于解释型语言。那什么是解释型语言呢？在介绍解释型语言之前，首先介绍计算机编程中的语言。

在计算机编程中，存在多种不同类型的编程语言。以下是常见的几种类型。

1. 机器语言

机器语言是计算机能够直接理解和执行的二进制代码，它使用二进制位表示指令和数据，并且对于不同的计算机体系结构和处理器，机器语言是特定的。机器语言是最低级别的编程语言，直接操作硬件。

2. 汇编语言

汇编语言使用助记符和符号来代表机器语言指令。它通过将汇编指令转换为对应的机器语言指令来与计算机硬件进行交互。每个汇编语言指令对应一条机器语言指令。

3. 高级编程语言

高级编程语言是相对于机器语言和汇编语言而言的。它使用更接近自然语言的语法和结构，以方便程序员编写和理解代码。高级编程语言提供了丰富的语法和功能，可以通过编译器或解释器将源代码转换为机器语言或直接执行。

常见的高级编程语言如下。

(1) C：一种通用的编程语言，具有高性能和底层硬件访问能力。

(2) C++：在 C 语言基础上扩展的编程语言，支持面向对象编程。

(3) Java：一种跨平台的编程语言，使用 Java 虚拟机(JVM)来执行字节码。

(4) Python：一种简洁、易读易写的高级编程语言，注重代码可读性和开发效率。

(5) Ruby：一种优雅、简洁的动态编程语言，强调简单性和开发人员的幸福感。

这只是列举了一些常见的编程语言，实际上还有许多其他编程语言可供选择，每种语言都有其特定的用途和优势。

从上述可知 Python 是高级编程语言，而高级编程语言可以通过编译器或解释器将源代码转换为机器语言直接执行。此时，就需要先了解什么是编译型语言和解释型语言。

4. 编译型语言

编译型语言是指在代码执行之前需要经过一个称为编译器的特殊程序的处理。编译器将源代码作为输入，对其进行编译和转换，生成与目标计算机体系结构兼容的机器语言代码，也称为目标代码或二进制代码。这个目标代码可以直接在计算机上执行，而不需要再次进行翻译或解释。

(1) 编译：源代码在执行之前需要经过一次编译过程，将其转换为机器码。

(2) 执行效率：由于编译型语言在执行之前已经完成了编译过程，因此通常具有较高的执行效率。

(3) 依赖机器体系结构：生成的目标代码是与特定计算机体系结构相关的，所以在不同的计算机上需要重新编译生成对应的目标代码。

常见的编译型语言有 C、C++、Rust、Go 等。

5. 解释型语言

解释型语言是指不需要显式的编译过程，而是逐行解释和执行源代码。在执行过程中，解释器会逐行读取源代码，并根据其语法规则进行解释和执行。解释型语言通常在运行时动态地解释源代码，而不会生成可执行的目标代码。

(1) 解释：源代码逐行解释执行，不需要进行编译。

(2) 执行效率：相比编译型语言，解释型语言通常具有较低的执行效率，这是因为它们需要在每次执行时解释代码。

(3) 平台无关性：解释型语言的源代码可以在不同的平台上直接执行，因为解释器在每次执行时会动态解释代码。

常见的解释型语言有 Python、JavaScript、Ruby、PHP 等。

需要注意的是，现代的编程语言往往不是纯粹的解释型语言或编译型语言，而是结合了两种方式的特点的语言。例如，Java 是一种编译型语言，但它也使用了一个称为 Java 虚拟机的解释器来执行字节码。这种混合的方式称为即时编译(Just-in-Time Compilation, JIT)。

6. 编译型语言和解释型语言的区别

编译型语言和解释型语言是两种不同的编程语言，它们在代码的执行方式和执行过程方面存在区别，如图 1-1 所示。

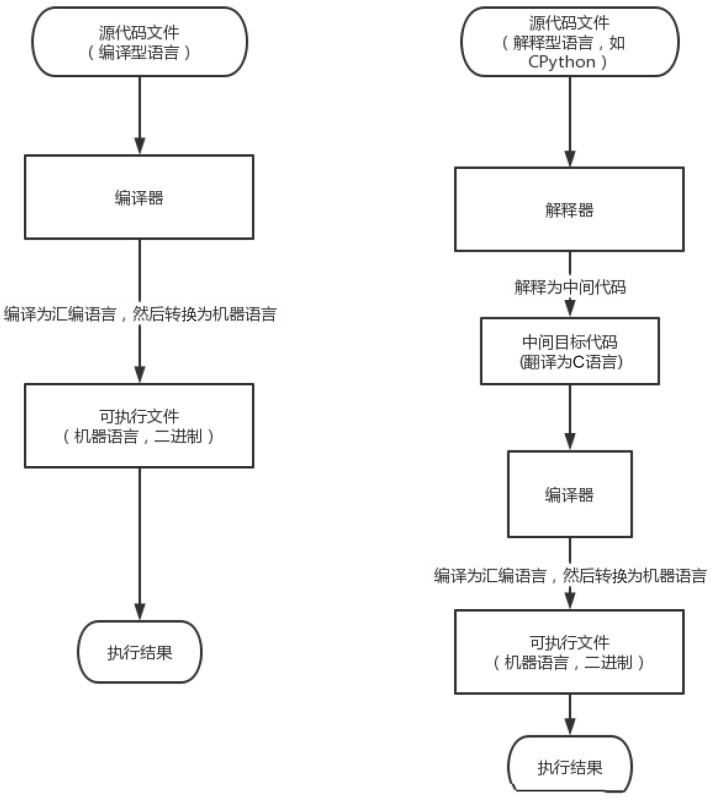

图 1-1　编译型语言和解释型语言的区别

1.1.5 Python 程序的执行原理

前文介绍了解释型语言和编译型语言，接下来根据解释型语言的特性介绍 Python 程序执行的原理。

Python 程序执行的原理可以概括为以下几个步骤。

1. 解析源代码

Python 解释器首先对源代码进行解析。这个过程涉及将源代码分解为语法结构，即识别和理解其中的语句、表达式和标识符等。

2. 编译字节码

解析源代码后，Python 解释器将其转换为字节码。字节码是一种中间形式的低级指令集，类似于机器语言，但是与特定的计算机体系结构无关。字节码的编译过程将源代码翻译成一系列字节码指令，这些指令将在后续的执行阶段被逐条解释和执行。

3. 解释执行

一旦字节码编译完成，Python 解释器开始逐条解释和执行字节码指令。解释器按照指令的顺序执行每个指令，将其转换为相应的操作，并更新变量、执行函数、控制流等。这个解释执行的过程是动态的，即在运行时进行。

4. 运行时的环境和内存管理

Python 解释器在执行过程中还负责管理运行时的环境和内存。它维护变量、对象和函数的命名空间，并执行内存管理任务，如垃圾回收，以释放不再使用的内存。

需要注意的是，Python 解释器可以有不同的实现，如 CPython、Jython、IronPython 等，它们在具体实现细节和性能上可能会有所不同，但总体的程序执行原理是相似的。

此外，Python 也支持即时编译技术。一些 Python 解释器在运行时会将热点代码(频繁执行的部分)即时编译成机器码，以提高执行效率。这种混合的执行方式使 Python 在某些情况下可以获得接近编译型语言的执行速度。

1.2 安装 Python 编程环境

前文对 Python 有一个大概的介绍，接下来安装 Python 的编程环境，以下的安装以 Windows 10 操作系统为例。

搭建 Python 编程环境，需要完成以下步骤。

1. 下载 Python 解释器

访问 Python 官方网站下载适合自己计算机操作系统的 Python 解释器。选择最新的稳定版本，并根据操作系统的要求进行安装。在此界面的操作为，将鼠标指针悬浮至“Downloads”选项上，弹出系统版本下拉列表，选择与自己计算机匹配的系统选项，然后单击右侧选项对应的版本，即可快速下载该版本解释器的安装包，如图 1-2 所示。

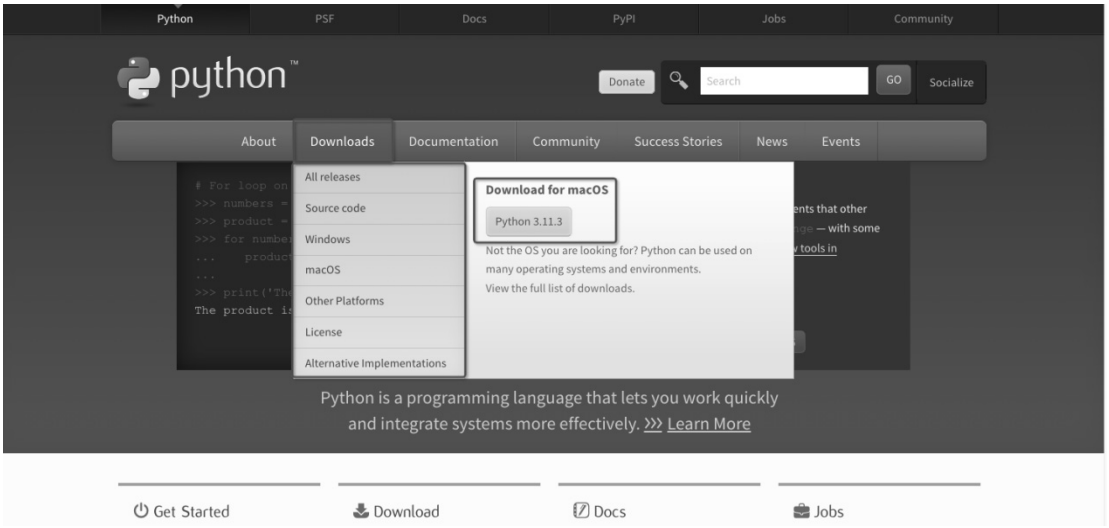

图 1-2　Python 官方网站解释器下载界面

2. 安装 Python 解释器

在官方网站下载完 Python 解释器后，双击打开该程序，打开 Python 解释器安装首页。在该界面中可以选择 Python 解释器的安装位置，如图 1-3 所示。选择“Install Now”选项，将 Python 解释器安装在系统默认的安装位置。也可以通过选择“Customize installation”选项进行自定义安装，注意在自定义安装时需要选择安装的文件位置。

图 1-3　Python 安装首页

(1) “Use admin privileges when installing py.exe”复选框：安装 Python 时使用管理员权限，建议选中该复选框。

(2) “Add python.exe to PATH”复选框：将 Python 添加至环境变量，建议选中该复选框。

选择 Python 附加选项后，选择“Install Now”或“Customize installation”选项，打开下一步的安装界面。

3. 安装 Python 附加功能

如图 1-4 所示，可以设置 Python 解释器的附加功能，根据自己的需求选择相应的附加功能。

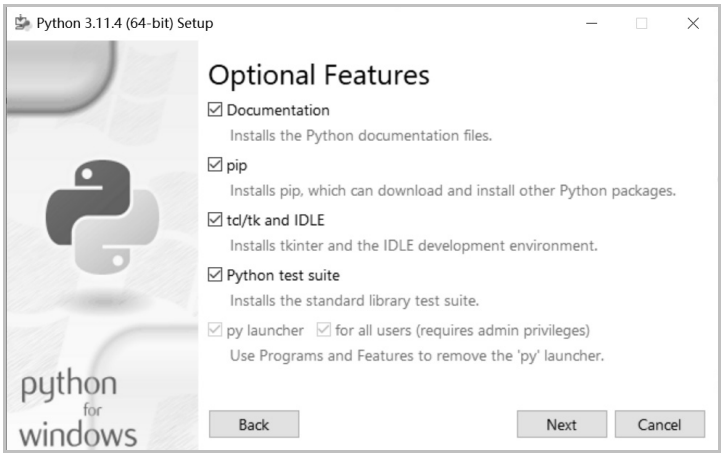

图 1-4　选择附加功能

(1) “Documentation”复选框：安装 Python 文档文件(建议选中)。

(2) “pip”复选框：安装 pip 功能，该功能可以下载并安装其他 Python 包(建议选中)。

(3) “tcl/tk and IDLE”复选框：安装 tkinter 和 IDLE 开发环境(建议选中)。

(4) “Python test suite”复选框：安装 Python 标准库测试套件(建议选中)。

(5) “py launcher”复选框：安装 Python 运行解释器(建议选中)。

(6) “for all users (requires admin privileges)”复选框：会为所有的用户安装 Python 解释器(建议选中)。

选择自己需要的功能后，单击“Next”按钮，打开高级选项界面。

4. 设置 Python 安装高级选项

如图 1-5 所示，在此界面设置 Python 安装的高级选项。

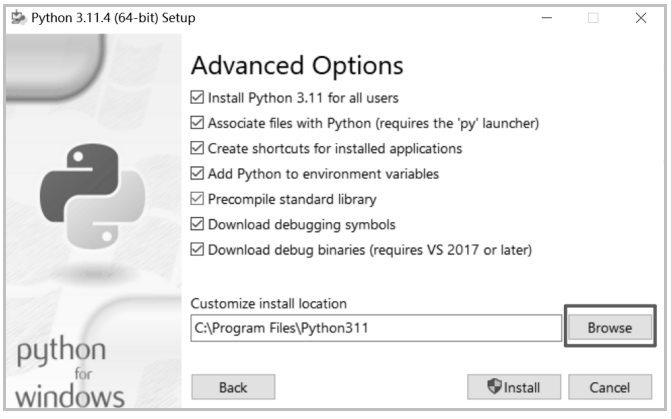

图 1-5　Python 安装高级选项

(1) “Install Python 3.11 for all users”复选框：为所有用户安装 Python 3.11(推荐选中，根据下

载版本不同，出现的安装版本也不同)。

(2) “Associate files with Python (requires the 'py' launcher)”复选框：将文件与 Python 关联(需要“py”启动器，建议选中)。

(3) “Create shortcuts for installed applications”复选框：为已安装的应用程序创建快捷方式(建议选中)。

(4) “Add Python to environment variables”复选框：将 Python 添加到环境变量(建议选中)。

(5) “Precompile standard library”复选框：会预编译标准库(建议选中)。

(6) “Download debugging symbols”复选框：会下载调试符号。

(7) “Download debug binaries (requires VS 2017 or later)”复选框：下载调试二进制文件(需要 VS 2017 或更高版本)。

选择所需的高级选项后，接下来，可以在“Customize install location”文本框中设置安装位置，推荐单击其右侧的“Browse”按钮，在打开的对话框中选择安装的位置即可。最后单击“Install”按钮安装 Python 解释器。

5. 完成安装过程

等待安装 Python 解释器的进度条加载完成，如图 1-6 所示，当出现 Setup was successful 字样界面时，说明安装完成，此时单击“Close”按钮结束解释器的安装，如图 1-7 所示。

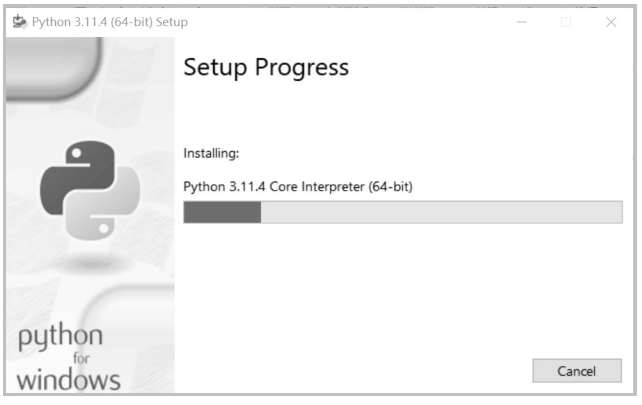

图 1-6　Python 解释器安装进度条

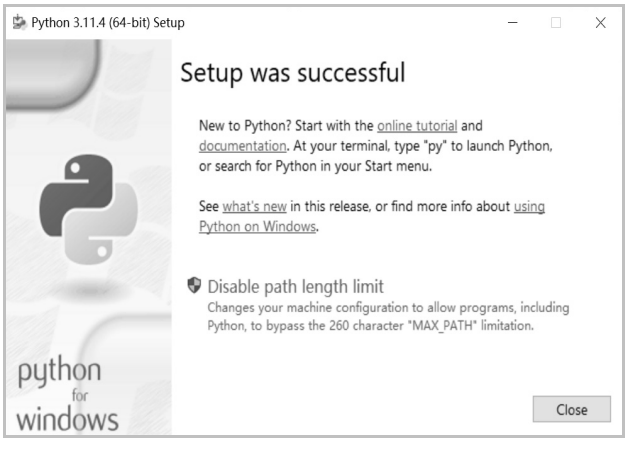

图 1-7　Python 解释器安装完成

6. 测试 Python 安装

打开命令行终端(在 Windows 上是命令提示符，在 macOS 和 Linux 上是终端)，输入以下命令来检查 Python 是否成功安装并显示版本信息。

```
C: python --version
Python 3.11.4     #显示正确的版本信息
```

如果你安装了多个 Python 版本，可能需要使用特定的命令来指定要使用的版本。例如，在某些系统中，可能需要使用以下命令来检查 Python 3 的版本：

```
xxx: python3 --version
Python 3.11.4     #显示正确的版本信息
```

如果能够正确显示 Python 的版本信息，则表示 Python 环境已成功搭建。

现在，你已经成功搭建了 Python 编程环境，接下来快速体验一下 Python 程序吧！

1.3 Python 开发工具介绍

在 Python 开发过程中，使用一个好的开发工具可以大大提高编程开发的效率、简化任务，并为开发者提供更好的编码体验。目前常用的 Python 编程开发工具包括 Python 解释器自带的 IDLE、集成开发环境 Eclipse、PyCharm、Vscode、IPython、Eric5、PythonWin 等。

1.3.1 IDLE 的使用方法

Python 自带的 IDLE 是指 Python 的集成开发环境，是 Python 语言官方提供的一个简易开发环境。

IDLE 是基于 Python 内置的 Tkinter 模块制作的。它提供了一个交互式的 Shell(解释器)和一个简单的代码编辑器。它可以让你在一个窗口中输入和执行 Python 代码，并提供了基本的代码编辑功能，如语法高亮显示、代码缩进和自动补全等。

IDLE 特别适合初级开发者学习使用。通过使用 IDLE，你可以快速编写、测试和调试 Python 代码。然而，对于更复杂的项目或专业开发人员，会选择使用其他高级的集成开发环境，如 PyCharm、Visual Studio Code 等，这些工具提供了更高级的调试和项目管理功能。

下面是详细的操作过程。

1. 启动 IDLE

在 Windows 操作系统中，通过“开始”菜单找到安装的 Python 版本文件夹，打开后找到 IDLE 的程序并双击启动，如图 1-8 所示。

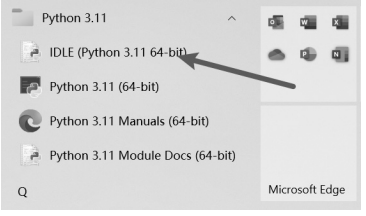

图 1-8　IDLE 程序

2. IDLE 的 Shell 界面

启动后，首先出现的是 IDLE 的 Shell 界面，也就是运行界面。该运行界面如图 1-9 所示，界面中的菜单栏含义如下。

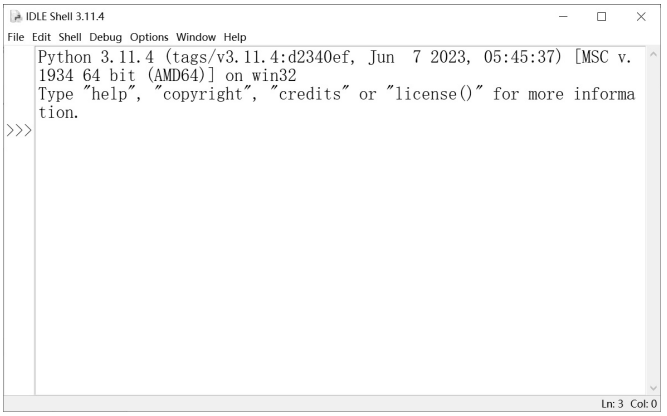

图 1-9　IDLE 的 Shell 界面

(1) File(文件)：创建文件、打开文件和保存文件等。

(2) Edit(编辑)：复制、粘贴程序内部的编码，查找字符等。

(3) Shell(命令解析器)：对 Shell 界面的控制。

(4) Debug(调试)：调试代码。

(5) Options(选项)：用于控制 IDLE 的窗口布局和显示选项。

(6) Window(窗口)：模式切换，可以在此处实现 Edit 模式和 Shell 模式来回切换。

(7) Help(帮助)：IDLE 的帮助菜单，可以在此寻找所需的功能并进行疑难解惑。

3. 创建文件

创建文件时，可以选择 IDLE 菜单栏中的“File”→“New File”选项来新建，然后就可以在该文件中编写程序了，如图 1-10 所示。

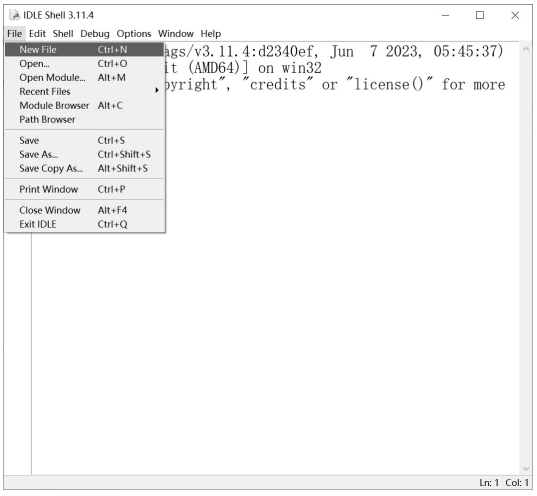

图 1-10　创建文件

4. 编写 Python 程序

创建成功后，出现如图 1-11 所示的编程界面，可以在该界面的编辑区编写 Python 程序。

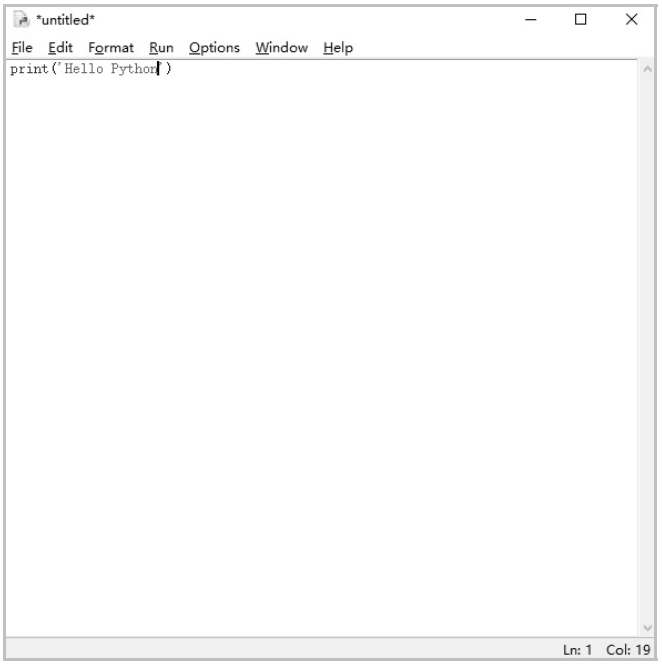

图 1-11 IDLE 的编辑界面

该界面中的菜单栏与 Shell 界面中的菜单栏略微不同，其含义如下。

(1) File(文件)：包含文件相关的选项，如新建、打开、保存、退出等。

(2) Edit(编辑)：包含编辑相关的选项，如剪切、复制、粘贴、查找、替换等。

(3) Format(格式)：包含代码格式化相关的选项，如自动缩进、块缩进、行尾空格等。

(4) Run(运行)：包含运行代码的选项，如运行模块、运行自定义的 Python Shell、重新启动等。

(5) Options(选项)：包含 IDLE 环境的各种选项设置，如字体、颜色、缩进等。

(6) Windows(窗口)：用于管理 IDLE 环境中的窗口，如打开新的 Shell、关闭窗口等。

(7) Help(帮助)：提供有关 IDLE 的帮助和文档，包括 IDLE 快捷键、Python 文档等。

注意››› 根据不同版本的 IDLE 和操作系统，菜单栏的具体内容可能会有所不同，通常会包含上述功能选项。

5. 保存和运行程序

在 IDLE 编辑界面选择“Run”→“Run Module”选项，保存并运行程序，如图 1-12 所示。

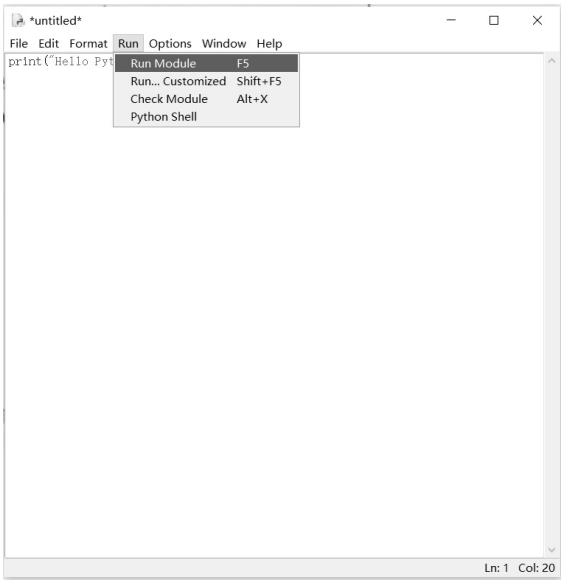

图 1-12　IDLE 运行选项界面

在运行程序时，需要先将文件保存，然后运行，如图 1-13 所示。如果是新建文件，则需要先选择程序的存放位置，如图 1-14 所示，然后才可以运行。

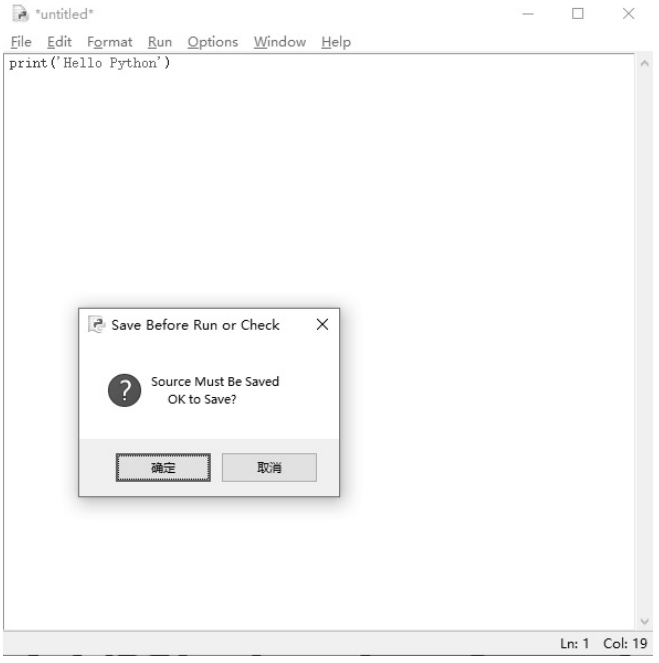

图 1-13　保存文件界面

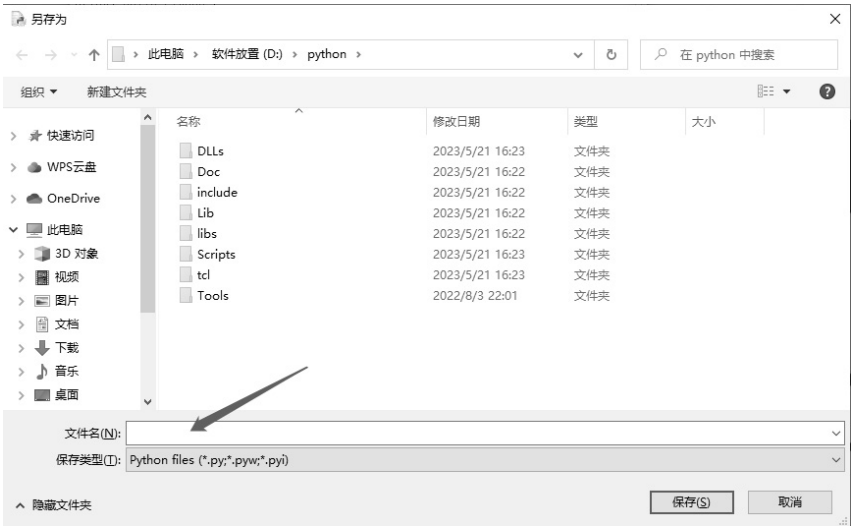

图 1-14　选择程序存放的位置

程序保存并运行后，就可以在 IDLE 的 Shell 界面中显示运行效果了，如图 1-15 所示。

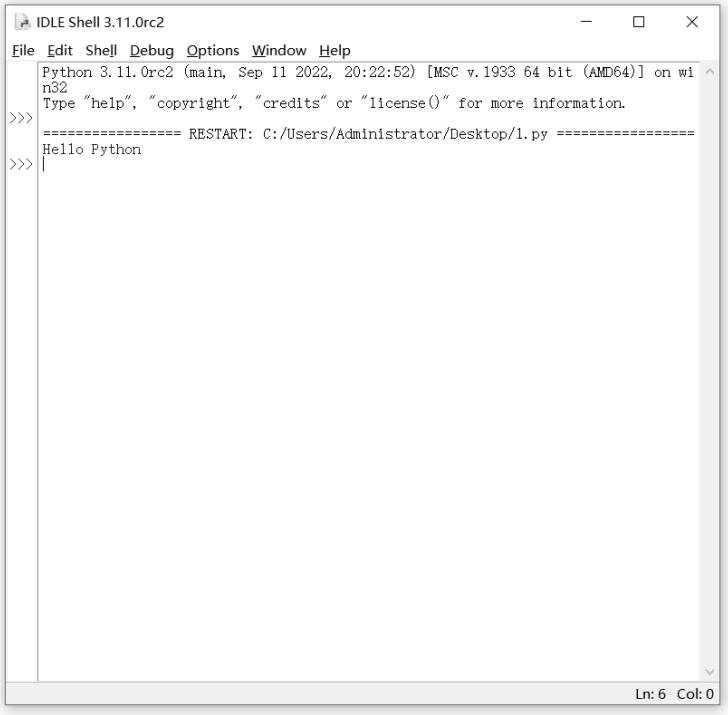

图 1-15　程序的运行效果

1.3.2　PyCharm 的安装与使用

在介绍了 Python 自带的 IDLE 后，下面来认识一款专业编写 Python 程序的软件——PyCharm。它是一种由 JetBrains 公司开发的集成开发环境(IDE)，专门用于 Python 编程语言。它提供了许多功能和工具，旨在提高开发人员的生产力和代码质量。

1. PyCharm 的特点和功能

(1) 代码编辑器：PyCharm 提供了一个强大的代码编辑器，具有自动完成、语法高亮、代码导航和代码重构等功能。它支持 Python 的各种版本，并具有智能代码补全功能，能够根据上下文提供准确的代码建议。

(2) 调试器：PyCharm 内置了强大的调试器，允许开发人员逐行执行代码并检查变量的值。它提供了断点、条件断点、表达式求值和堆栈跟踪等功能，帮助开发人员快速定位和修复错误。

(3) 代码版本控制：PyCharm 集成了流行的代码版本控制系统，如 Git、Mercurial 和 Subversion。它提供了一套工具，用于管理代码仓库、提交和更新代码、解决代码冲突，以及查看版本历史记录。

(4) 测试工具：PyCharm 支持各种测试框架，如 unittest、pytest 和 doctest。它提供了方便的测试运行器和测试报告，帮助开发人员编写和运行单元测试、集成测试和功能测试。

(5) 代码分析：PyCharm 具有强大的静态代码分析功能，可以检测潜在的错误、代码重复和不一致之处。它还提供了代码重构工具，可以自动重命名变量、提取方法和优化导入等。

(6) 虚拟环境支持：PyCharm 可以轻松管理 Python 的虚拟环境。它允许创建、配置和切换虚拟环境，以便在不同项目之间隔离和管理依赖关系。

(7) 插件生态系统：PyCharm 具有丰富的插件生态系统，允许开发人员根据自己的需求扩展和定制 IDE 的功能。用户可以安装各种插件，如代码检查工具、自动化工具和主题等。

2. PyCharm 的下载

在安装 PyCharm 之前，先打开 PyCharm 的官方网址。单击网页中间的“Download”按钮，进入下载界面，如图 1-16 所示。

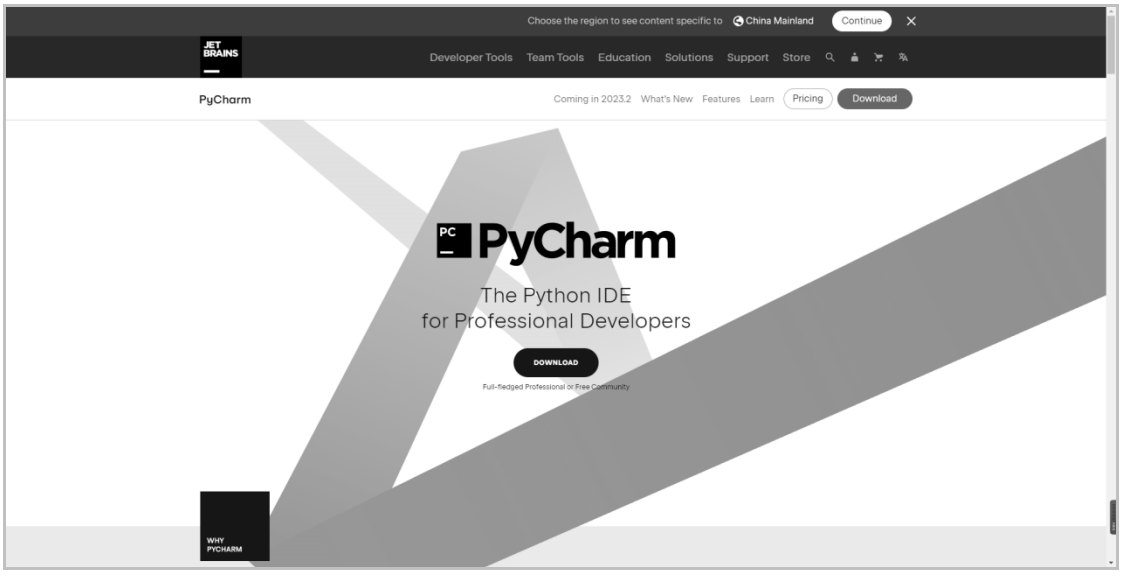

图 1-16　PyCharm 的下载界面

在 PyCharm 的下载界面中，官方提供了两个版本，分别是专业版(Professional)和社区版(Community)。它们在功能和许可证方面有一些区别，以下是它们之间的主要区别。

(1) 功能：PyCharm 专业版具有比社区版更多的功能和工具。专业版包含了一些高级特性，如数据库工具、远程开发、科学计算、Web 开发和框架支持等。它还提供了更多的插件和集成支持，

能够满足更广泛的开发需求。

(2) 支持的框架：PyCharm 专业版给许多流行的 Python 框架提供了更好的支持，如 Django、Flask、Pyramid、web2py 等。它提供了更多的模板和代码片段，简化了框架开发的流程。

(3) 数据库工具：专业版集成了各种数据库工具，如 SQLAlchemy、SQL 和数据库导航器等。这些工具使其与数据库的交互更加方便，可以直接在 IDE 中执行和调试 SQL 查询。

(4) 远程开发：专业版支持远程开发，可以与远程服务器进行连接，并在远程环境中进行代码编辑、调试和测试。这对于开发和调试位于远程服务器上的应用程序非常有用。

(5) 科学计算支持：专业版提供了一些用于科学计算和数据分析的高级工具和集成，如 NumPy、Pandas 和 Matplotlib。这些工具使在 PyCharm 中进行数据处理和可视化更加方便。

(6) 许可证：社区版可以免费使用，适用于个人项目。而专业版是商业软件，使用时需要购买许可证。

综上所述，专业版相对于社区版具有更多的高级功能和工具，适用于专业开发人员和团队。社区版则适合个人开发者或小型项目，提供了基本的 Python 开发功能。可以根据自己的需求和预算选择适合的版本。

对于 Python 的初学者，本书推荐下载社区版。单击社区版的“Download”按钮进行下载，如图 1-17 所示。

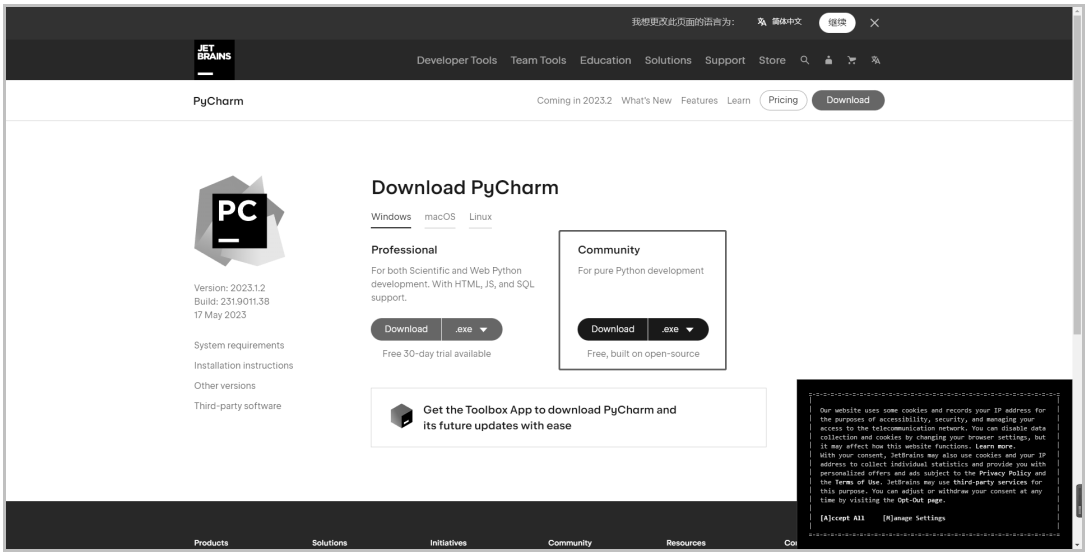

图 1-17　下载 PyCharm 社区版

注意 PyCharm 下载界面会自动下载对应计算机操作系统的安装包。

3. PyCharm 的安装

下载好安装包后，接下来就需要安装程序了。在对应下载文件夹中找到安装包，双击运行该安装包。

(1) 打开“PyCharm Community Edition Setup”窗口，首先出现 PyCharm 的欢迎安装界面等，如图 1-18 所示，直接单击“Next”按钮。

图 1-18　PyCharm 的安装欢迎界面

(2) 选择安装目录的位置。在打开的如图 1-19 所示的界面中，选择 PyCharm 的安装位置，单击“Browser”按钮，在打开的对话框中选择安装的位置即可。选择完成后，单击“Next”按钮。

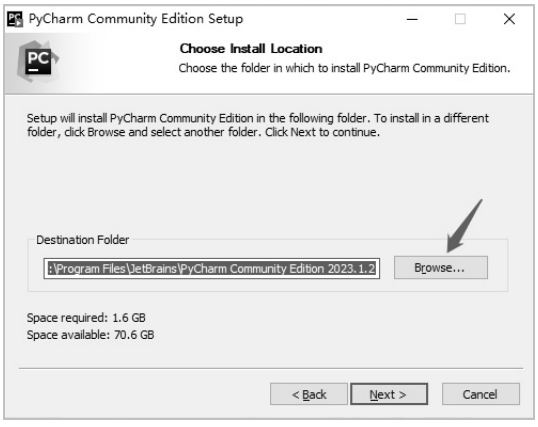

图 1-19　选择安装目录的位置

(3) 选择 PyCharm 内置的配置功能。在打开的如图 1-20 所示的 PyCharm 安装界面中，进行相应的设置，相关选项的含义如下。

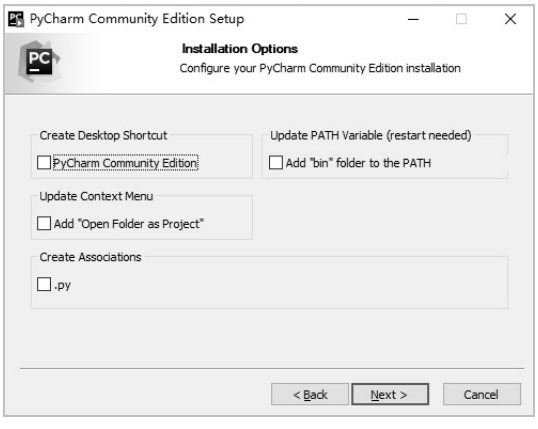

图 1-20　PyCharm 安装设置

- Create Desktop Shortcut：是否创建桌面快捷方式。
- Update Context Menu：更新上下级菜单。是否添加“将文件夹作为项目打开”等功能。
- Update PATH Variable (restart needed)：更新环境变量(需要重启)。是否将“bin”文件夹添加到环境变量中。
- Create Associations：是否与文件格式为“.py”的文件创建关联。

(4) 创建“开始”菜单文件夹。在“开始”菜单中创建以下列选项名称命名的文件夹，也可以输入名称来创建新的文件夹，默认为“JetBrains”文件夹。选择后，单击“Install”按钮进行安装，如图 1-21 所示。

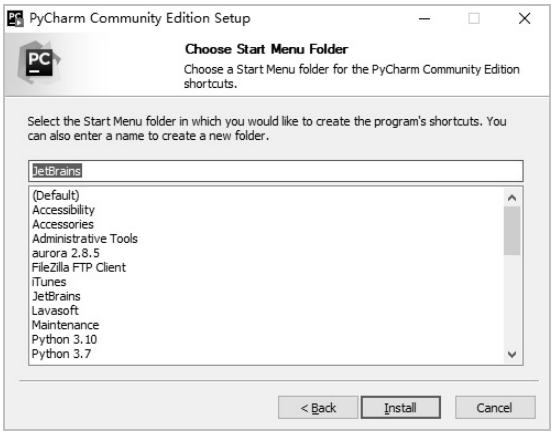

图 1-21　创建 PyCharm 的“开始”菜单文件夹

(5) 等待安装过程，如图 1-22 所示。

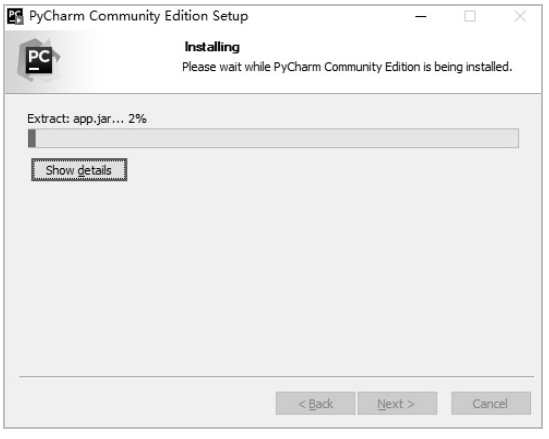

图 1-22　PyCharm 安装过程界面

(6) PyCharm 社区版安装完成后需要重启计算机，可以选择“Reboot now”选项立刻重启计算机，也可以选择“I want to manually reboot later”选项稍后手动重启计算机，根据个人需求进行选择，如图 1-23 所示。

图 1-23　PyCharm 安装完成界面

4. PyCharm 的使用

完成安装后，下面学习社区版 PyCharm 软件的使用方式，使用步骤如下。

(1) 安装后，打开“PyCharm User Agreement”对话框，选中“I confirm that I have read and accept the terms of this User Agreement”复选框，表示同意该用户协议条款，然后单击“Continue”按钮，如图 1-24 所示。

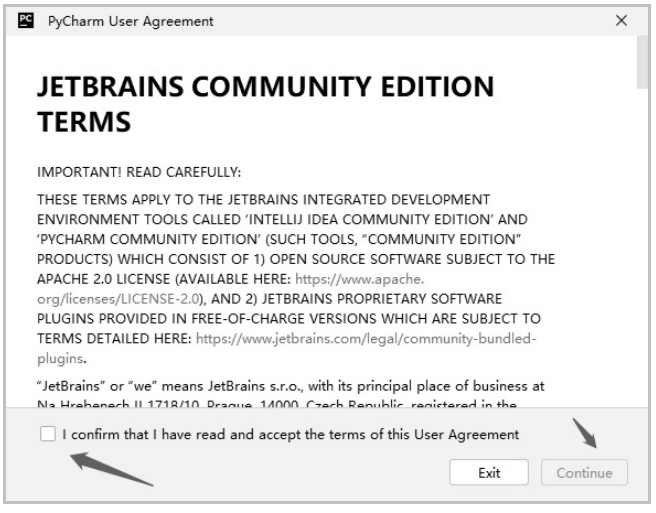

图 1-24　“PyCharm User Agreement”对话框

(2) 创建 Python 项目目录。在打开的如图 1-25 所示的界面中，可以创建 Python 程序项目，也可以打开一个 Python 程序项目，单击“+”号图标或选择“New Project”选项，创建一个新的 Python 项目。

(3) 新建 Python 项目配置。在 PyCharm 中，创建新项目时可以选择真实环境或虚拟环境。

真实环境指的是在你的计算机上全局安装的 Python 解释器。当选择真实环境时，PyCharm 将使用计算机上的全局 Python 解释器来运行和开发项目。这意味着你使用的是系统中已经安装的 Python 版本和库。

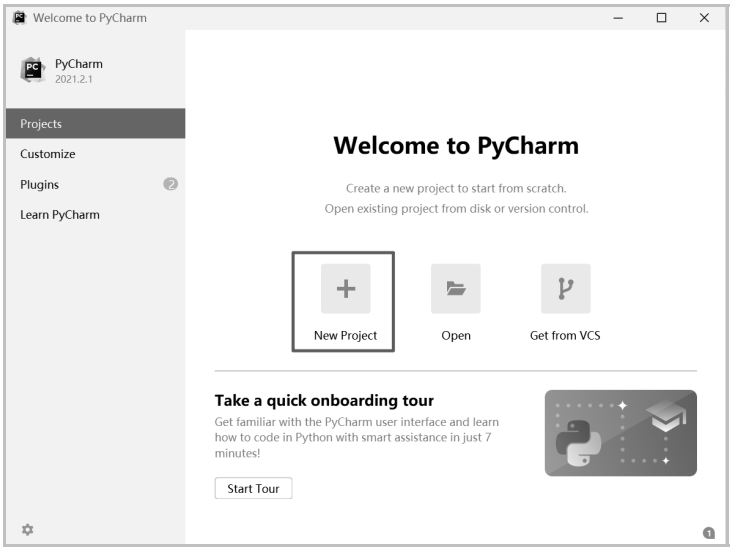

图 1-25　创建新项目

虚拟环境是一个独立的 Python 运行环境，它与全局环境隔离开来。使用虚拟环境可以为每个项目创建独立的 Python 环境，并可以安装特定版本的 Python 和所需的库。这对于不同项目使用不同 Python 版本或库的情况非常有用，同时也有助于避免项目之间的冲突。

在此推荐使用虚拟环境。如图 1-26 所示，界面中的相关选项含义如下。

- Location(位置)：创建新项目存放的文件目录。
- New environment using Virtualenv：使用新环境为虚拟环境(推荐选择此选项)，下方的“Location”选项为虚拟环境的存放位置，也可以使用 PyCharm 提供的默认位置。
- 在“Base interpreter”下拉列表中选择本机安装 Python 解释器，也可以使用 PyCharm 提供的默认解释器。
- Create a main.py welcome script：是否提供一个“main.py”的欢迎脚本(推荐选中该复选框)。

设置完成后，单击“Create”按钮，就可以创建并应用相应环境了。

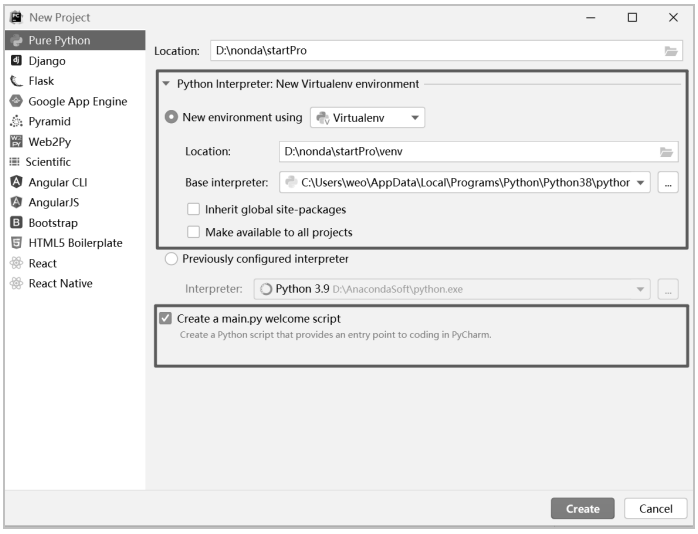

图 1-26　新建 Python 项目配置

(4) 认识 PyCharm 编辑界面。创建 Python 项目后，打开 PyCharm 的编辑界面，分为 3 个功能区域，即项目文件结构区、菜单功能区、代码编辑区，如图 1-27 所示。

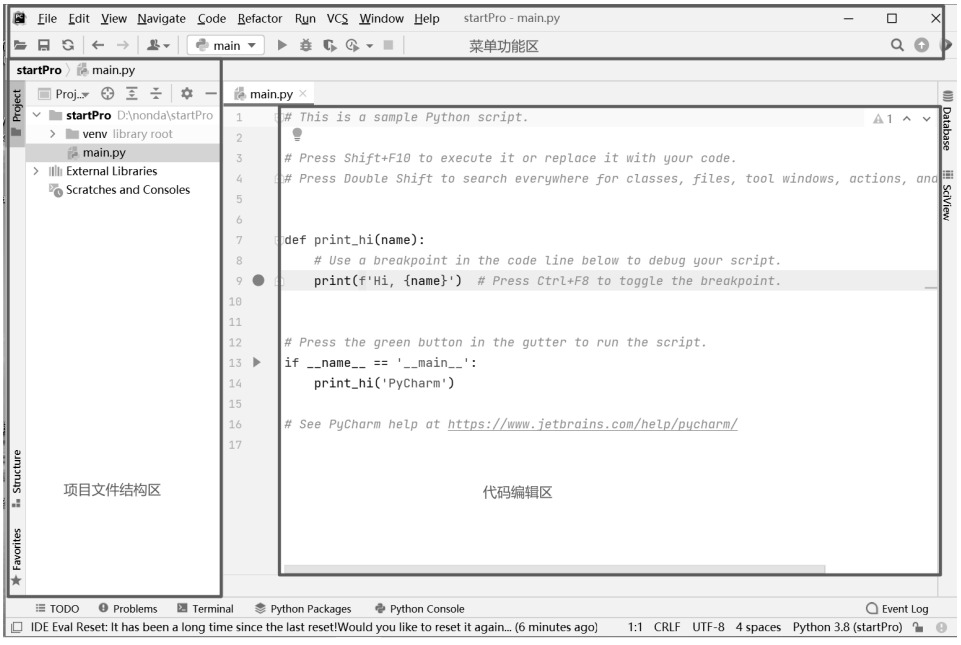

图 1-27　PyCharm 编辑界面

(5) 新建 Python 文件。选中左侧文件目录栏中的项目文件夹，右击，在弹出的快捷菜单中选择“New”→“Python File”选项新建 Python 文件，如图 1-28 所示。

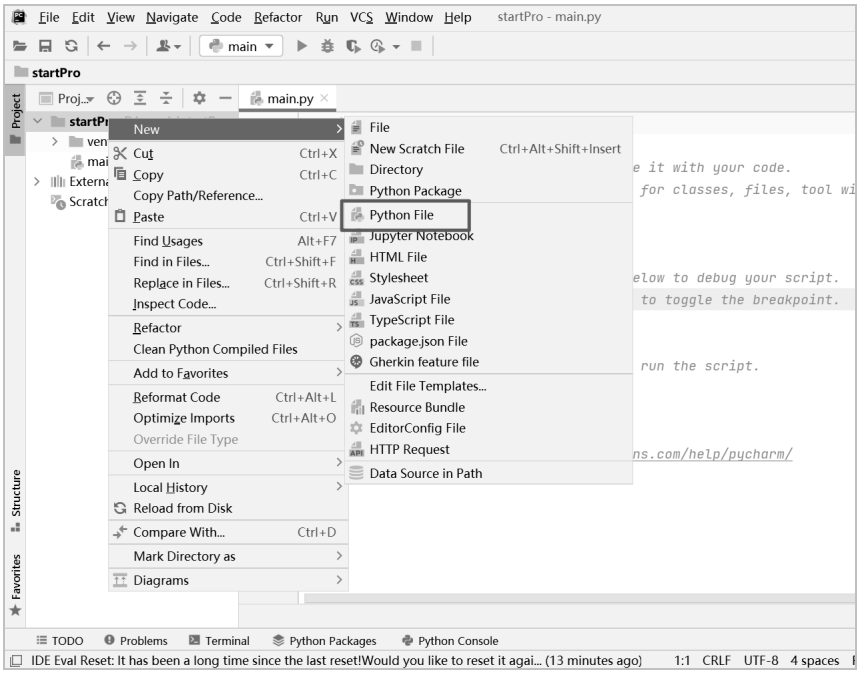

图 1-28　新建 Python 文件

(6) 创建第一个 Python 程序。在右侧代码编辑区中编辑“print("Hello world")”的程序。“print("Hello world")”程序的含义为在控制栏输出“Hello world”，如图 1- 29 所示。

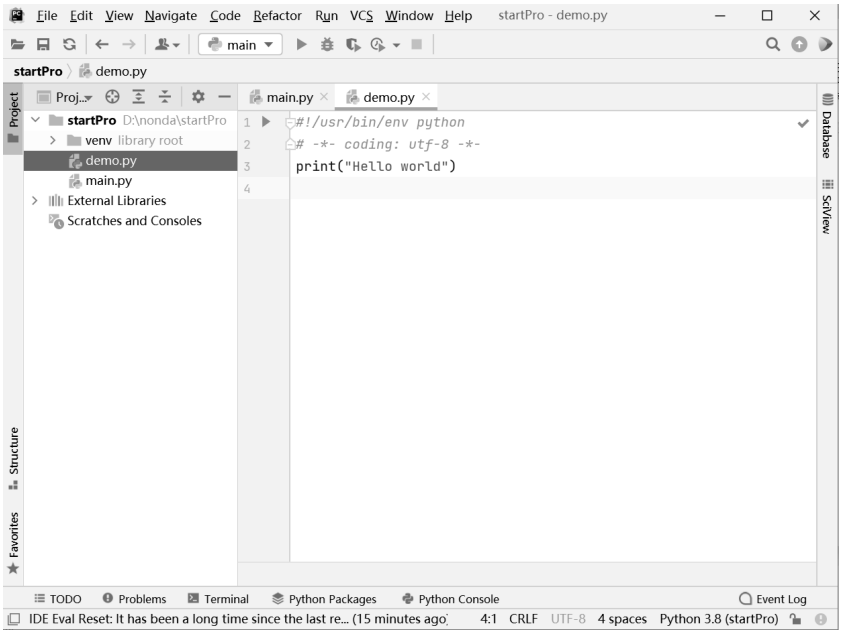

图 1-29　编写 Python 程序

(7) 运行 Python 程序。在代码编辑区的空白处右击，弹出该程序的功能选项快捷菜单，运行程序时需要选择“Run 'demo'”选项，如图 1-30 所示。其中，demo 为 Python 的程序名称，创建的文件名称不同，显示也不同。

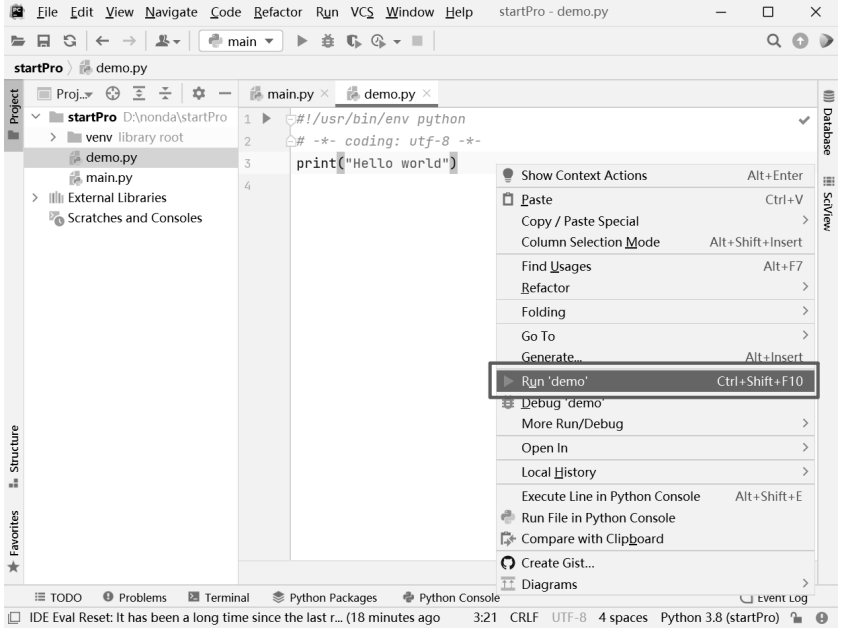

图 1-30　运行 Python 程序

(8) 查看运行结果。运行程序后，程序下方会出现运行结果窗口，即可查看运行结果，如图 1-31 所示。

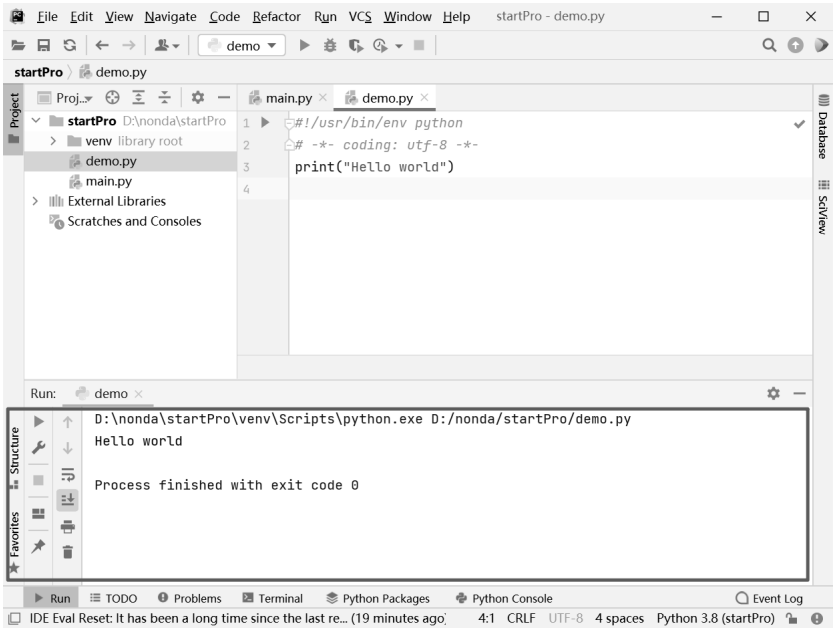

图 1-31　查看运行结果

1.4　绘制菱形图案

前面介绍了 Python 内置的 IDLE 和专业级 PyCharm 开发工具的安装和使用，接下来使用相应的工具绘制菱形图案，如图 1-32 所示。

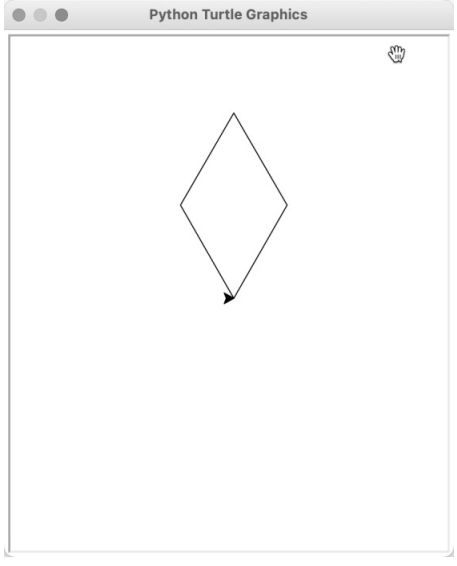

图 1-32　菱形图案

示例代码如下。

```
import turtle
#创建一个画笔对象
my_pen = turtle.Pen()
#绘制菱形
my_pen.left(60)          #向左旋转 60°
my_pen.forward(100)      #向前移动 100 个单位
my_pen.left(60)          #向左旋转 60°
my_pen.forward(100)      #向前移动 100 个单位
my_pen.left(120)         #向左旋转 120°
my_pen.forward(100)      #向前移动 100 个单位
my_pen.left(60)          #向左旋转 60°
my_pen.forward(100)      #向前移动 100 个单位
my_pen.left(60)          #向左旋转 60°
#关闭绘图窗口
turtle.done()
```

上述代码使用 turtle.Pen()创建了一个 Pen 对象，并命名为 my_pen。接下来，交替使用向左旋转和向前移动的方法，绘制 4 条长度相等的边，并设置边与边之间的夹角，它们组合在一起形成一个菱形。最后，使用 turtle.done()来关闭绘图窗口。

1.5 绘制雪人图案

在绘制菱形后，接下来绘制卡通图案——雪人，如图 1-33 所示。

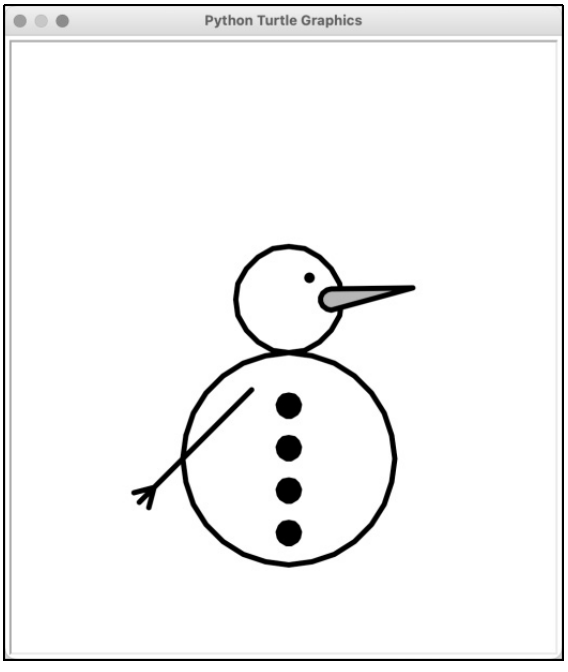

图 1-33 雪人图案

示例代码如下。

```
#导入 turtle 模块并重命名为 t
import turtle as t
#初始化画笔
t.speed(0)                      #设置画笔的速度为最快
t.pensize(5)                    #设置画笔的宽度为 5 像素

#绘制雪人的头和身体
t.circle(50)                    #绘制雪人的头部，半径为 50
t.circle(-100)                  #绘制雪人的身体，半径为-100，表示向相反方向绘制圆形

#绘制雪人的纽扣
for i in range(4):
    t.right(90)                 #右转 90°，调整画笔方向
    t.pu()                      #抬起画笔
    t.fd(40)                    #向前移动 40 像素
    t.pd()                      #放下画笔
    t.left(90)                  #左转 90°，调整画笔方向
    t.begin_fill()              #开始填充形状
    t.circle(-10)               #绘制纽扣，半径为-10,表示向相反方向绘制圆形
t.end_fill()                    #结束填充形状

#绘制雪人的眼睛
t.pu()                          #抬起画笔
t.goto(20,70)                   #将画笔移动到指定坐标位置
t.pd()                          #放下画笔
t.dot(10,"black")               #绘制一个直径为 10 像素的黑色点，表示眼睛

#绘制雪人的鼻子
t.pu()                          #抬起画笔
t.goto(40,60)                   #将画笔移动到指定坐标位置
t.pd()                          #放下画笔
t.fillcolor("orange")           #设置填充的颜色为橙色
t.begin_fill()                  #开始填充形状
t.left(180)                     #左转 180°，调整画笔的方向
t.circle(10,180)                #绘制半径为 10 的半圆
t.left(15)                      #左转 15°，调整画笔的方向
t.fd(80)                        #向前移动 80 像素
t.goto(40,60)                   #回到起始点
t.end_fill()                    #结束填充形状

#绘制雪人的手
t.pu()                          #抬起画笔
t.goto(-35,-35)                 #将画笔移动到指定坐标位置
t.setheading(225)               #设置画笔的方向为 225°，即左下方
t.pd()                          #放下画笔
t.fd(130)                       #向前移动 130 像素
t.right(30)                     #右转 30°，调整画笔的方向
for i in range(3):
    t.fd(20)                    #向前移动 20 像素
```

```
    t.bk(20)                    #向后移动 20 像素
    t.left(30)                  #左转 30°，调整画笔的方向
t.hideturtle()                  #隐藏画笔
t.done()                        #完成绘制
```

运行上述程序，即可绘制雪人图案。

本章小结

本章介绍了 Python 语言的发展历史，以及语言的优缺点、Python 环境的搭建、IDE 开发工具的安装使用等。然后使用两个案例来快速体验 Python 语言的魅力。

(1) Python 是一种简洁优雅、易读易学的解释型编程语言，拥有强大的标准库和广泛的应用领域，如 Web 开发、数据科学、人工智能、自动化测试、游戏开发、系统管理等领域。

(2) Python 是一种高级编程语言，由于其简洁易读的语法和强大的库支持，已经成为当前较受欢迎的编程语言之一，Python 有两个主要版本系列，推荐使用 Python 3.x。

(3) PyCharm 作为一款功能丰富、集成开发环境完整、开箱即用、社区支持强大的 Python 开发工具，可以帮助开发者更高效地编写和调试 Python 代码。PyCharm 适合专业开发人员使用，有以下优点。

① 功能丰富：PyCharm 提供了许多强大的功能，如代码自动补全、代码错误提示、智能代码重构、版本控制、调试等，可帮助开发者更高效地编写和调试 Python 代码。

② 多语言支持：PyCharm 不仅支持 Python，还支持其他很多种语言，如 JavaScript、HTML、CSS 等，可以为开发者提供一个统一的开发环境。

③ 集成开发环境：PyCharm 是一款完整的集成开发环境，它包含了多种工具和功能，并可以通过插件进行扩展。同时，PyCharm 还提供了自动化部署、远程调试等功能，为开发者提供了全面的开发和调试环境。

④ 开箱即用：PyCharm 安装后就可以直接使用，无须进行额外的配置。同时，PyCharm 可以与其他工具和框架无缝衔接，如 Django、Flask 等，方便开发者快速构建 Web 应用。

思考与练习

一、选择题

1. Python 的发展史中，(　　)是 Python 的创始人。

A. Larry Page　　B. Bill Gates

C. Mark Zuckerberg　　D. Guido van Rossum

2. Python 的两个主要版本系列是(　　)。

A. 1.x 和 2.x　　B. 2.x 和 3.x

C. 3.x 和 4.x　　D. 4.x 和 5.x

3. Python 语言的特点之一是(　　)。

A. 静态类型　　B. 复杂烦琐　　C. 面向过程　　D. 简洁易读

4. 下列准确描述了解释型语言的是(　　)。
   A. 源代码需要事先编译为机器码
   B. 源代码在运行时逐行解释执行
   C. 运行速度较快
   D. 不支持面向对象编程
5. IDLE 是(　　)。
   A. Python 的主要集成开发环境　　B. 一种编译器
   C. 一种数据库管理系统　　D. Python 的虚拟环境管理工具

二、填空题

1. Python 2.x 系列在 2020 年停止了_____。
2. Python 是一种_____语言，无须显式声明变量类型。
3. PyCharm 提供了更多高级特性，如代码_____、调试器、版本控制等。
4. 解释型语言的代码在运行时逐行_____并执行。
5. IDLE 是 Python 官方提供的轻量级集成_____环境。

三、简答题

1. 请简要描述 Python 的发展历史。
2. 请简要描述解释型语言和编译型语言的区别。
3. 请简要说明 Python 语言的特点。
4. 什么是 IDLE 和 PyCharm?
5. 请简要说明 Python 的两个主要版本系列的区别。