前 言

PREFACE

Python 语言，是当今世界最流行的编程语言之一。随着我国不断加强人工智能（Artificial Intelligence，AI）发展战略，Python 语言已被广泛应用于人工智能产品的研发、行业大数据分析等各个领域。掌握必要的 Python 语言已成为新世纪人才具备的基础素质之一。本书是 Python 语言的入门教材，期望能够为初学者打下良好的基础，为初学者开启一扇探索 Python 语言与行业有效结合的大门。本书具有以下特点。

1. 定位准确

本书主要是为经济金融专业学生进行 Python 程序设计学习而编写的，考虑到这部分学生的程序设计基础比较薄弱，因此，本书的学习目标主要是将程序设计与本专业相结合，通过大量示例讲述程序设计语言中的奥妙。

2. 注重实践

对经济金融专业的学生，在学习了必要的程序设计语法和规则后，更关注的是 Python 语言的实际应用，因此本书着重介绍使用 Python 语言编写程序来解决专业中可能遇到的实际问题。

3. 便于自学

本书由浅入深，通过大量的示例讲解，便于学生根据示例的提示，独立完成 Python 语言程序的编写与调试，有助于培养学生独立解决问题的能力，据此激发学生与本专业知识相结合的创新能力。

本书为校企合作教材，通过资源共享、信息共享和实践与理论结合的方式，培养学生的积极性和综合基础知识。

本书可作为各类院校计算机、大数据和人工智能等相关专业及教育培训机构的专用教材，也可供 Python 编程和大数据技术爱好者及相关从业人员参考使用。

由于水平有限，本书难免会有内容的疏漏，恳请各位同仁和广大读者批评指正，也希望各位能将实践过程中的经验和心得与我们交流。

目　录

CONTENTS

知识入门

0.1 硬件与软件

计算机硬件是计算机中用户能够触及的部分，换句话说，就是用户能够摸到的东西。从信息处理的角度来看，信息由外部世界经由输入设备进入计算机，然后在主机内进行处理，最后通过输出设备把信息处理的结果送出。在这个过程中，这些输入和输出设备根据信息的不同特点而表现不同，比如要输入字符，可以通过键盘进行录入。较常见的输入设备有键盘和鼠标，它是我们向计算机输入信息的主要设备。因此一般市场上所看到的计算机，几乎都带有这两种设备。

输出设备对于计算机来说，也很重要。计算机只有具有了输出设备，人类才能了解它的运行状态和获取信息处理的结果，而不同的输出设备以不同的形式输出信息处理结果。较常用的输出设备就是以图文形式输出信息的显示器和打印机，当然还包括音箱、绘图仪等。

计算机的本质是信息处理机，要处理信息就必须具备处理信息的硬件设备。类似于人的大脑，要处理信息就要有运算能力和存储能力。计算机主机中的中央处理器包括运算器和控制器，就具有运算能力及控制能力；存储器就具有存储能力。

对于计算机来说，其自身特点决定了它的信息处理和信息存储的形式，即使用二进制代码（只有 0 和 1）来表示所有的信息。外界的信息要进入计算机，就必须转换为二进制代码，当然包括文字、图片、声音、视频等各种形式的信息。

计算机在处理信息时，能够按照人们的要求，通过一定的方式、方法或步骤来进行。要实现这个目标，计算机就必须通过软件（程序）来进行控制。因此，计算机输入信息、处理信息和输出信息都必须通过软件来进行控制。

由计算机硬件概念可推知，计算机软件是人们不能触及的一部分，它实际上是对计算机进行控制的方式或步骤的描述。计算机要想完成某个任务，就先要知道完成这个任务所需的人为指定的步骤或方法。对于某个特定的问题，人们通过给定的指令序列来进行方法或步骤的描述，而这里的指令序列本质上是指 CPU 指令的序列，也就是程序。

此外，要解决某一问题需要采取一定的方法，这个方法的具体实现方式我们称之为算法。就像解一道数学应用题，你既可以使用列综合算式的方法来完成，也可以通过列方程的方法来完成，而这些不同的算法最后达成的目标是相同的。

显然，计算机能够运行的前提是必须同时具备硬件和软件。硬件是一切工作的基础，也是计算机赖以存在的物质基础；而软件是硬件工作必不可少的控制者。同时具备硬件系统和软件系统才能构成一个完整的计算机系统。

0.2 编程语言

编程语言从计算机诞生的那天起就存在了，人类用计算机表示信息的方式（二进制代码 0 和 1）来描述对计算机的控制程序，这就是机器语言，它同时也是这台计算机 CPU 的内建命令集。由此也可以看出，机器语言其实就是人类用计算机本身的语言来完成对计算机的控制，要想控制计算机，人类就要学会机器所使用的机器语言（用 0 和 1 描述）。

当然，用一长串 0 和 1 来编写程序控制计算机的话，效率肯定很低，既不容易理解，又很容易出错。此外，不同类型的计算机，其机器语言也不同，也就是说，即使是为了解决同样的问题，不同的计算机，程序也不相同。

要想解决机器语言难学、易错的问题，就不能用 0 和 1 来编写程序，而是用单词来代替命令代码（一串 0、1）。因此，诞生了汇编语言，汇编语言不是采用二进制代码来描述解决问题的步骤，而计算机只能识别二进制代码，所以运行时要先翻译为二进制代码的机器语言（专业术语叫编译），然后计算机才能识别和执行。

汇编语言虽然解决了易错、难记的问题，但还是不同于人类的语言，如果能像对一个人说话一样来书写程序那就最好了，于是人们就使用了一种能够较为准确描述算法步骤的接近于人类语言和数学表示方法的形式来作为书写程序的语言，即现代常用的编程语言——高级语言。常见的高级语言有 Python、C、Java、Perl、Erlang、LISP 等。

当然，高级语言也不能被计算机直接理解，运行前必须进行编译，最终成为机器语言，计算机才能理解和执行。高级语言的执行方式分为两种：一种是编译执行，即程序编写完成后直接将其编译为机器语言后执行；另一种是解释执行，即程序一边解释一边运行。例如，C 语言采取的是编译执行方式，而 Python 语言采取解释执行的方式。

我们要学习一门语言，首要任务就是掌握其语法及语义；其次就是熟练使用它；最后才能够写出高效运行的程序，解决某个具体的问题。

0.3 编程与调试

编程就是根据要解决的某个具体问题，设计出相应的解决步骤（算法），根据所用语言的语法和语义，写出对应的命令序列。由此可见，编程要完成的基本任务如下。

1. 设计科学的算法

解决某一具体的问题往往有多种不同的算法或步骤。所谓科学的算法，既要保证算法能在各种情况下正常地工作并且得到正确的处理结果，又要能够使程序高效率运行。高效率运行程序，即要求它能够使用尽量少的系统资源在较短的时间内得出正确的处理结果。算法是程序的灵魂，只有设计出好的算法，才能写出高效率的程序。

2. 写出正确的程序

写出正确的程序是指运用指定语言的语法和语义写出实现设计算法的程序。一个具有语法错误的程序是不能正常运行的，当然也不会对信息进行处理，更谈不上得到处理结果；程序的逻辑错误会导致程序不会按照预先设计的算法进行处理，当然也不会得到正确的处理结果。

3. 合理的人机交互

编程的最终目标是解决某个具体的问题，这一过程既需要信息的输入，又需要信息的输出。实现

了友好的信息输入和输出，才能更好地解决问题。然而，再高明的程序员写程序也不是一蹴而就的，写出的程序可能存在一些语法错误或逻辑错误，因此编程要完成的第二个任务就是调试。

⑴ 排除语法错误。对于编写的程序中的语法错误，只要尝试运行之后即可发现；一般来说，只要程序能正常运行，程序中就不会有语法错误。

⑵ 排除逻辑错误。程序中若存在逻辑错误，它依然可以正常运行，可能能得到正确的处理结果，也可能得不到正确的处理结果。排除逻辑错误，必须要对程序进行测试或调试。对于一个小型应用项目，可以采取罗列或穷举法来根据处理结果发现是否存在逻辑错误，从而进一步排除错误；而对于大中型应用项目，则需要进行专门测试，才能发现逻辑错误，从而排除逻辑错误。

⑶ 做好后期维护。随着一个应用项目的实施，用户可能会提出一些改进意见或修改建议，程序员要进行及时修正与维护。

第 1 章

Python 的简介与安装

素养目标

◎培养学生对 Python 编程的基础知识的理解和掌握能力。

◎培养学生养成一个良好的 Python 编程习惯。

◎培养学生用 Python 解决简单问题的能力。

知识目标

◎掌握 Python 的基本知识。

◎掌握 Python 的安装和环境的搭建。

◎掌握 Anaconda 的安装。

技能目标

◎能够安装 Python 并搭建好 Python 环境。

◎能够根据需要选择并使用适当的 Python 数据类型。

◎能够运用 Python 工具进行基本的数据操作。

1.1 什么是编码

在计算机中，所有的数据在存储和运算时都是用二进制数表示的。因此计算机默认只能处理数字，而不能处理文本。为了能够处理这些文本数据，需要将文本对应的每个字符转换为数字，而具体用哪些二进制数字表示哪个字符，就需要有一个统一的转换规则，这就是编码。

为了方便计算机处理文本数据，美国有关标准化组织出台了 ASC Ⅱ编码。它使用 1 个字节表示 1 个字符，即使用 8 位二进制，最多可以处理 256 个字符。ASC Ⅱ编码如今已被国际标准化组织 (ISO) 定为国际标准，称为 ISO 646 标准。该编码对大小写字母、数字、特殊符号、控制字符等指定了编码规则，将其转换为对应的二进制数字。常见的单字符与 ASC Ⅱ编码二进制对应关系如表 1–1 所示。

表 1–1　单字符与 ASC Ⅱ编码二进制对应关系

单字符	二进制	单字符	二进制
数字 0	00110000	大写字母 S	01010011
数字 1	00110001	大写字母 T	01010100
数字 2	00110010	大写字母 U	01010101
数字 3	00110011	大写字母 V	01010110
数字 4	00110100	大写字母 W	01010111
数字 5	00110101	大写字母 X	01011000
数字 6	00110110	大写字母 Y	01011001
数字 7	00110111	大写字母 Z	01011010
数字 8	00111000	小写字母 a	01100001
数字 9	00111001	小写字母 b	01100010
冒号 :	00111010	小写字母 c	01100011
分号 ;	00111011	小写字母 d	01100100
小于 <	00111100	小写字母 e	01100101
等号 =	00111101	小写字母 f	01100110
大于 >	00111110	小写字母 g	01100111
问号 ?	00111111	小写字母 h	01101000
大写字母 A	01000001	小写字母 i	01101001
大写字母 B	01000010	小写字母 j	01101010
大写字母 C	01000011	小写字母 k	01101011
大写字母 D	01000100	小写字母 l	01101100
大写字母 E	01000101	小写字母 m	01101101
大写字母 F	01000110	小写字母 n	01101110
大写字母 G	01000111	小写字母 o	01101111
大写字母 H	01001000	小写字母 p	01110000

续表

单字符	二进制	单字符	二进制
大写字母 I	01001001	小写字母 q	01110001
大写字母 J	01001010	小写字母 r	01110010
大写字母 K	01001011	小写字母 s	01110011
大写字母 L	01001100	小写字母 t	01110100
大写字母 M	01001101	小写字母 u	01110101
大写字母 N	01001110	小写字母 v	01110110
大写字母 O	01001111	小写字母 w	01110111
大写字母 P	01010000	小写字母 x	01111000
大写字母 Q	01010001	小写字母 y	01111001
大写字母 R	01010010	小写字母 z	01111010

1.2　初识 Python

Python 是一种面向对象、直译式的计算机程序设计语言，也是一种功能强大而完善的通用型语言，已经具有二十多年的发展历史，成熟且稳定。Python 可以完成从文本处理到网络通信等各种工作。Python 自身已经提供了大量的模块来实现各种功能，除此以外还可以使用 C/C++ 来扩展 Python，甚至还可以将 Python 嵌入其他语言中。

1.2.1　什么是 Python

Python 是一种面向对象的解释型计算机程序设计语言，是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。Python 是免费的解释型语言，具有面向对象的特性，可以运行在多种操作系统之上。Python 具有清晰的结构、简洁的语法及强大的功能。

1.2.2　Python 的优点

Python 不仅能在 Windows、Linux 系统中运行，还能运行在各类蓬勃发展的移动系统中。虽然 Python 不是程序语言唯一的选择，但却是一种不错的选择。面对简单易学而且功能强大的 Python，越来越多的人选择使用它来完成各种各样的任务。Python 的主要优点有以下几点。

1. Python 是免费的自由软件

Python 遵循通用公共授权协议，是自由软件，这是 Python 流行的原因之一。用户使用 Python 进行开发和发布自己编写的程序不需要支付任何费用，不用担心版权问题。即使作为商业用途，使用 Python 也是免费的。开源的自由软件正在成为软件行业的一种发展趋势，很多商业软件公司也开始将自己的产品变为开源的，如 Java。作为开源软件的 Python 将具有更强的生命力。

作为自由软件，用户可以很方便地获取源代码，通过阅读其源代码，发现其中的神奇之处。当深入使用 Python 以后，Python 的某些特性就显露出来了，而这些特性并没有详细的文档去说明，此时用

户可以通过 Python 的源代码去详细地了解它们。

2. Python 是跨平台的

跨平台、良好的可移植性是 C 语言成为经典编程语言的关键。Python 正是用可移植的 ANSIC 编写的。这意味着 Python 也具有良好的跨平台特性，也就是说，在 Windows 系统下编写的 Python 脚本可以轻易地运行在 Linux 系统下。当然如果在 Python 脚本中使用了 Windows 的某些特性，如 COM，那就另当别论了。

3. Python 功能强大

Python 强大的功能是很多用户支持 Python 的最重要原因。从字符串处理到复杂的 3D 图形编程，Python 借助扩展模块就可以轻易完成。实际上，Python 的核心模块已经提供了足够强大的功能。使用 Python 精心设计的内置对象可以完成许多功能强大的操作。

基于此类强大的功能，Python 可以应用在如下领域中。

（1）系统编程。通过 Python 编程可帮助用户完成烦琐的日常工作。

（2）科学计算。Python 简洁的语法可以使用户像使用计算器一样来完成科学计算。

（3）快速原型。Python 省去了编译调试的过程，用户可以快速地实现系统原型。

（4） Web 编程。使用 Python 可以编写公共网关接口 (Common Gateway Interface, CGI)，而现在流行的 Web 框架也可以使用 Python 来实现。

4. Python 是可扩展的

Python 提供了扩展接口，通过使用 C/C++ 可以为 Python 编写扩展。另外，Python 还可以嵌入 C/C++ 编写的程序之中。在 C/C++ 编写的程序之中可以使用 Python 完成一些对于 C/C++ 实现起来较复杂的任务。在某些情况下，Python 还可以作为动态链接库的替代品在 C/C++ 中使用。

Python 可以很容易地被修改、调试，而不需要重新编译。使用 Python 也不必担心版本的问题。

5. Python 易学易用

Python 的语法十分简单，而且在 Python 中数据类型的概念十分模糊。在使用变量时无须事先声明变量的类型。

使用 Python 不必关心内存的使用和管理，Python 会自动分配、回收内存。Python 提供了功能强大的内置对象和方法。使用 Python 可以减少其他编程语言的复杂性，如在 C 语言中使用数十行代码实现的排序，在 Python 中可以通过列表的排序函数轻易完成。通过一段时间的学习，加上对 Python 模块函数的不断了解，用户很快就可以使用 Python 编写出强大的脚本。

1.2.3 Python 的应用

Python 作为一种功能强大、简单易学的编程语言而广受好评，它可以应用在以下几个方面。

1. Web 应用开发

服务器端编程，具有丰富的 Web 开发框架。使用 Python 可以快速完成一个网站的开发和 Web 服务。国内的豆瓣、搜狐、知乎等，国外的 Google、Dropbox 等都应用到了 Python。

2. 系统网络运维

在网络运维工作中，很多方面有大量重复性的工作，如管理系统、监控系统等需要将工作自动化

起来，这些都离不开 Python 的功能。

3. 科学与数字计算

Python 被广泛地运用于科学和数字计算中，如生物信息学、物理、建筑、地理信息系统、图像可视化分析、生命科学等。

4. 图形界面开发

Python 可编写桌面图形用户界面，还可以扩展微软的 Windows 系统，如 Tk 库、PyQt 库等。

5. 3D 游戏开发

Python 有很好的 3D 渲染库和游戏开发框架，还有很多使用 Python 开发的游戏，如 Sid Meier’s Civilization(文明)。

6. 爬虫

随着大数据的兴起，爬虫应用被提升到了前所未有的高度，多数数据分析挖掘公司以网络爬虫的方式获取不同来源的数据，并为其所用。Python 提供了大量的库，可以很容易做到随机获取想要的信息。

1.3　搭建 Python 开发环境

Python 语言是解释型高级程序设计语言，要使用它就必须搭建 Python 开发环境，即安装 Python 的编译系统 (解释器)。因为 Python 是解释型编程语言，所以需要一个解释器才能运行代码。

1.3.1 Python 的下载和安装

Python 的安装程序及源代码可以从其官方网站 https://www.python.org 获取。下面以 Windows 7、Python 3.8 为例，介绍在 Windows 系统下安装 Python 的过程，具体操作过程如下。

(1) 进入 Python 官网，如图 1–1 所示，点击 Download，点击 All release 找到所有发行版本。

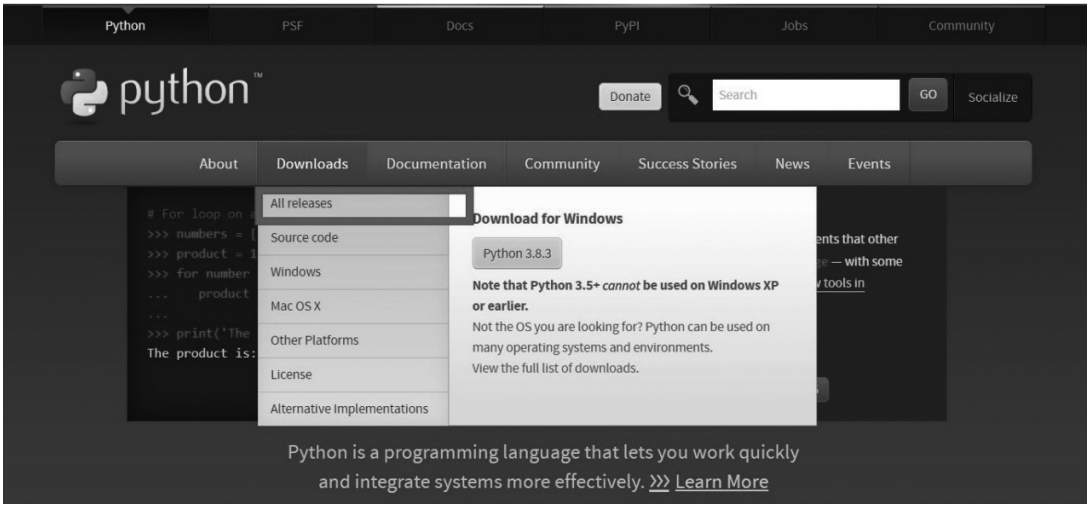

图 1–1　python 官网下载软件

（2）点击后页面向下滑到如图 1–2 所示界面，找到要下载的 Python 版本，点击进入新的页面后，再次向下滑动到页面底部。选择文件下载，.zip 后缀是压缩文件。

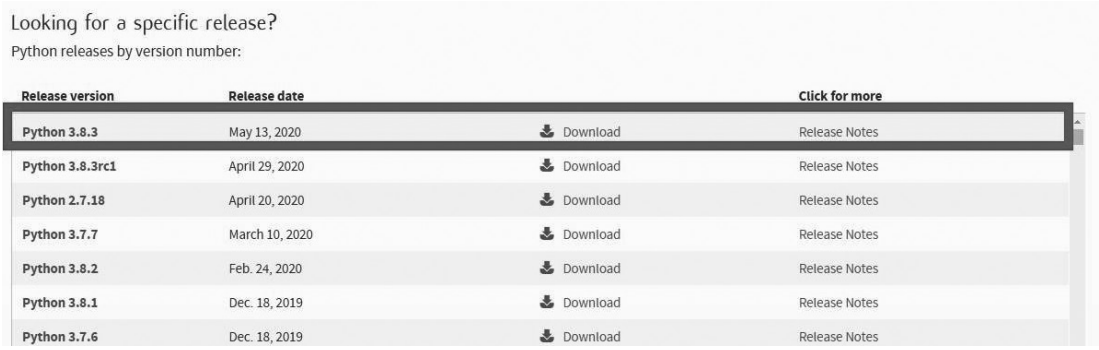

图 1–2　选择 python 版本

（3）下载完成后，如图 1–3 所示，直接进入文件夹打开运行 .exe 文件，勾选“Add Python 3.8 to PATH”，方便直接在命令行中直接使用 Python 命令。

图 1–3　运行 python 安装程序

（4）安装完成后，显示如图 1–4 所示的安装成功界面，单击 Close 按钮关闭对话框即可。

图 1–4　安装成功画面

（5）检查安装情况，打开 cmd 命令框，输入 python，显示成功安装 Python 3.8.3，如图 1–5 所示。

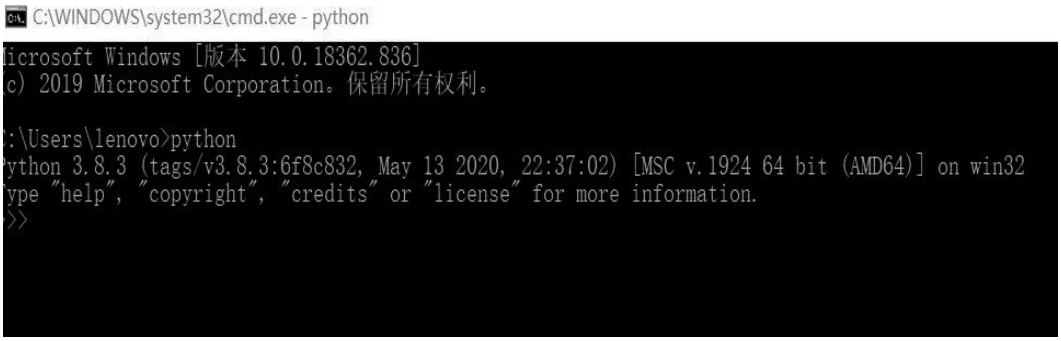

图 1–5　检查安装情况

1.3.2 Anaconda 软件安装

Anaconda 是一个方便 Python 的 python 包管理和环境管理软件，支持 Linux、Mac、Windows，可以很方便地解决版本 Python 并存、切换以及各种第三方包安装问题，使用方便、环境部署步骤简单，具体操作步骤如下。

（1）点击进入 Anaconda 官网，如图 1–6 所示，点击 Distribution（分配）。

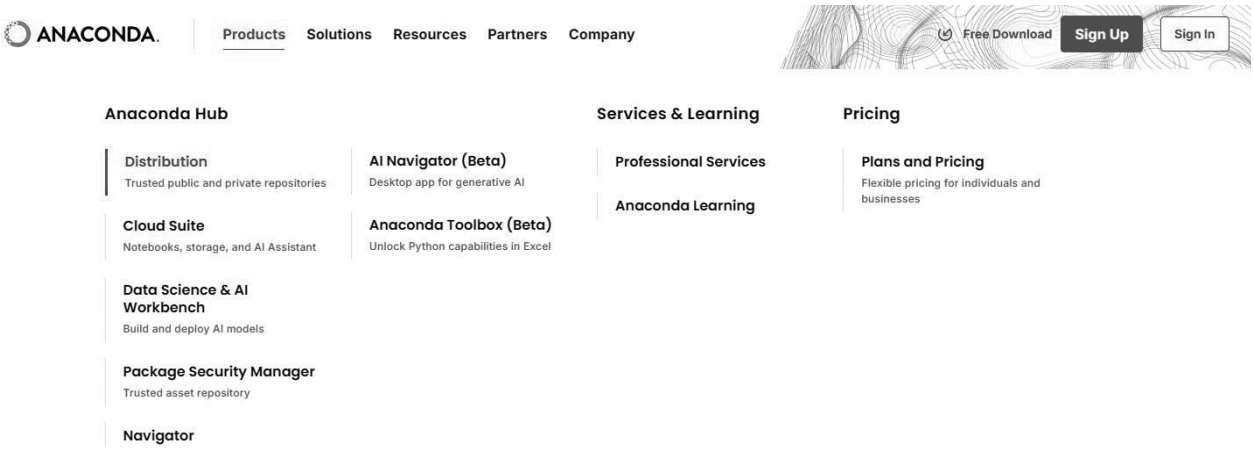

图 1–6　Anaconda 官网页面

（2）进入新页面后，可以输入电子邮件下载或跳过注册直接下载最新版本，如图 1–7 所示。本书以已下载好的 Python 3.8 为例进行说明，新版本安装运行方法与之类似。

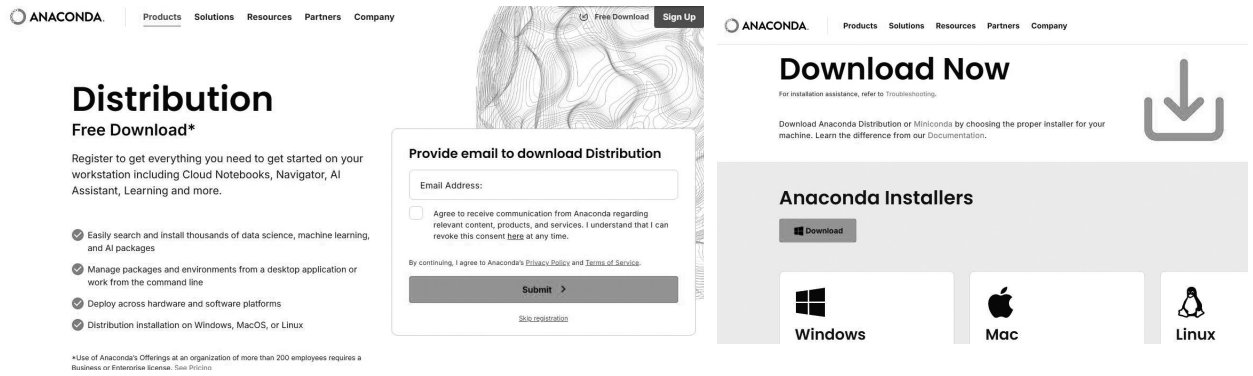

图 1–7　选择版本

（3）运行下载好的 .exe 文件，一直点击下一步到修改安装路径，修改安装路径，安装路径的所有文件名不能有空格，如图 1–8 所示。

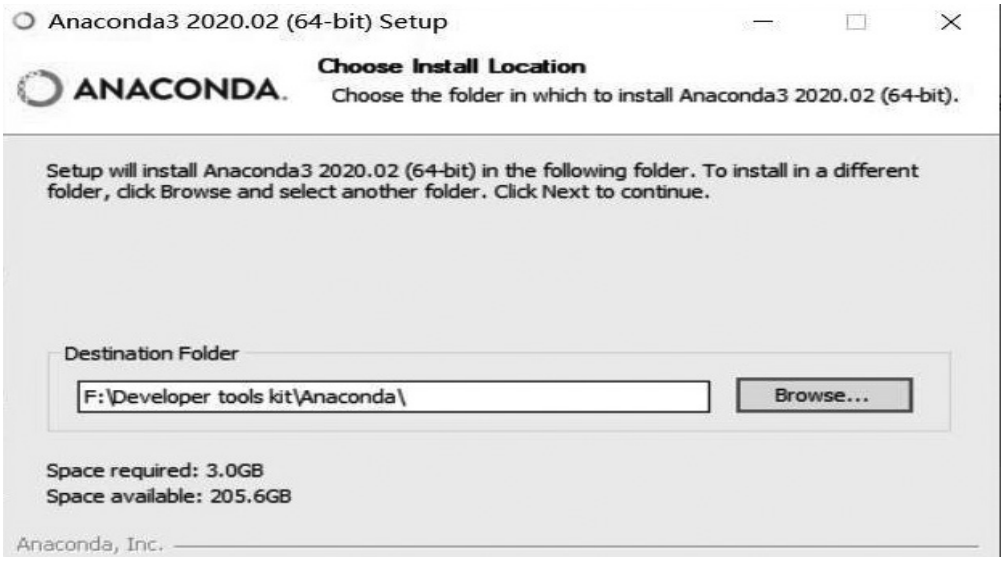

图 1–8　进入安装界面

（4）完成安装后在开始栏里找到并点击 Anaconda Navigator，期间会弹出一系列命令框，等待几分钟成功打开 Anaconda，完成安装，如图 1–9 所示。

图 1–9　完成 Anaconda 安装

1.3.3 Jupter–Notebook 软件安装

Jupter Noteabook 是一个开源的 Web 应用程序，允许开发者方便地创建和共享代码文档，可以实时写代码、运行代码、查看结果并可视化数据，允许把代码写入独立的 cell 中，然后单独执行。用户可以在测试项目时单独测试特定代码块，无须从头开始执行代码，基于 Web 框架进行交互开发，非常方便。具体操作步骤如下。

（1）进入到 Anaconda，点击 Environment，如图 1–10 所示。

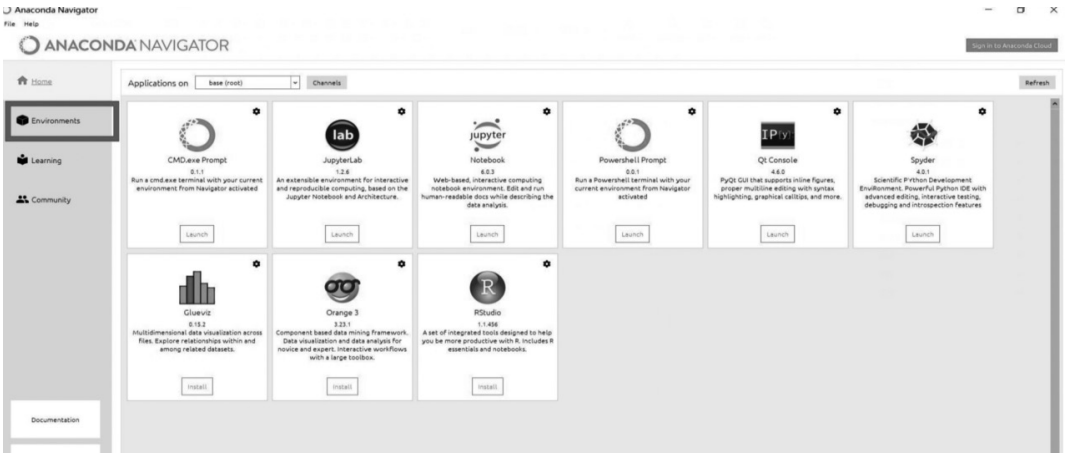

图 1–10　点击 Environment

（2）点击 Open Terminal 运行终端，如图 1–11 所示。

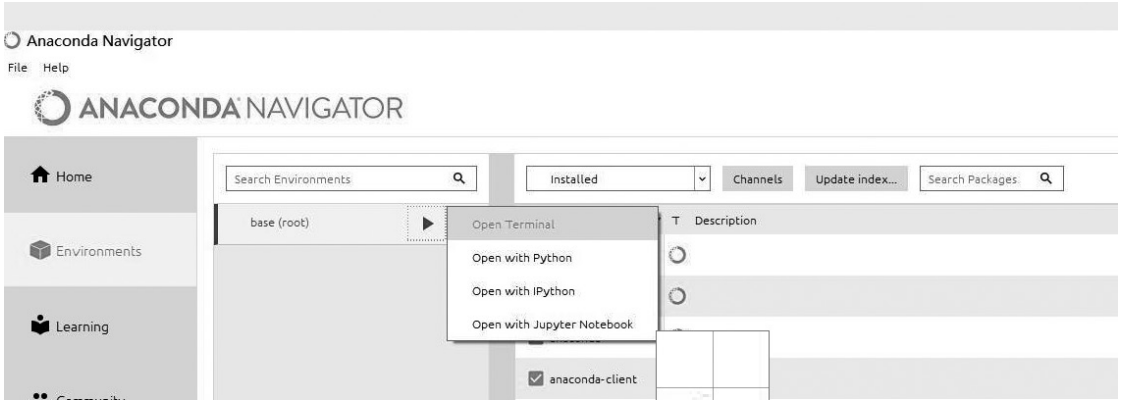

图 1–11　运行终端

（3）弹出命令行窗口，输入下面配置语句新建环境，name 是自定义名字，继续输入配置语句，激活环境，括号内从 base 换成自定义的 name 之后成功完成激活，如图 1–12 所示。

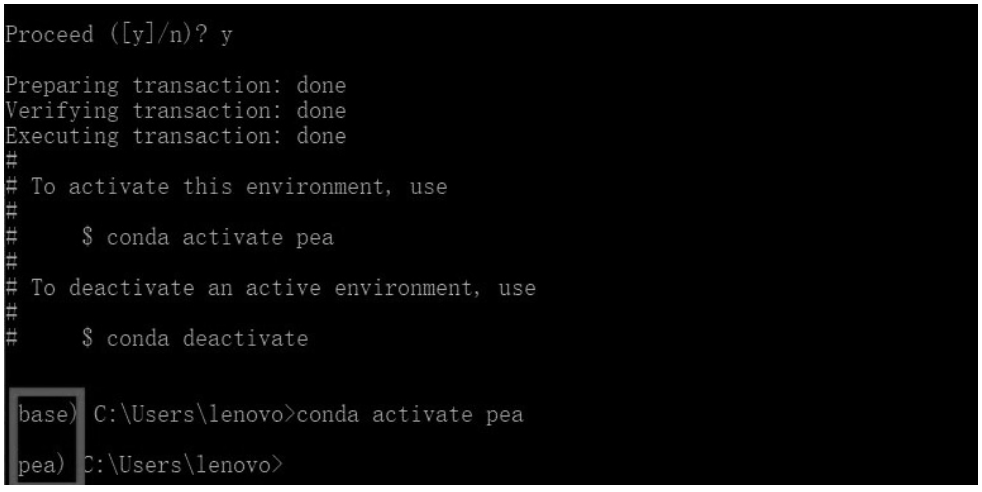

图 1–12　输入配置语句

（4）回到 Anaconda 里已经可以找到新建的安装环境。原本的 base 里有 Jupyter–Notebook，但要在新的环境开发，切换之后安装 Jupyter–Notebook，如图 1–13 所示。

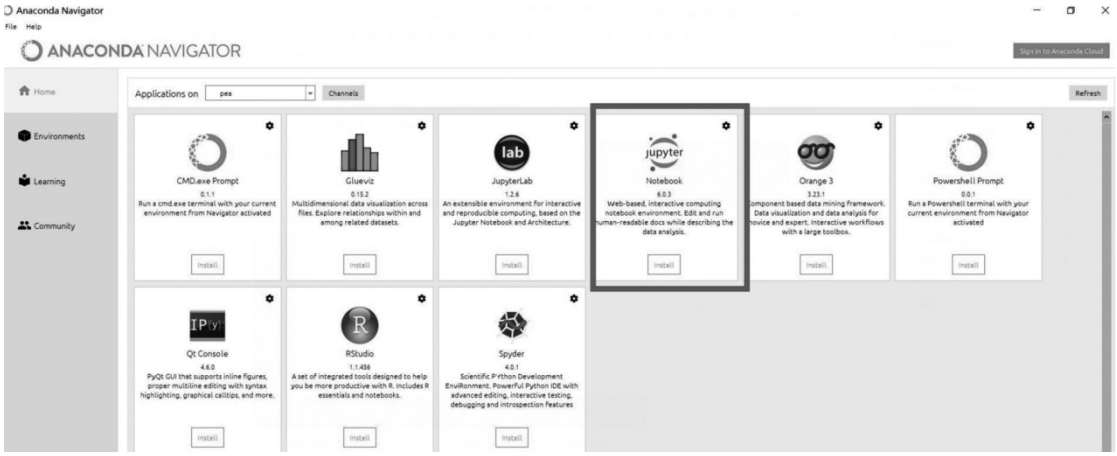

图 1–13　安装 Jupyter–Notebook

（5）安装之后 luanch，回到之前的命令行窗口下载工具包进行页面优化，输入以下配置命令，如图 1–14 所示。

```
(base) C:\Users\lenovo>conda activate pea
(pea) C:\Users\lenovo>pip install jupyterthemes -i https://pypi.tuna.tsinghua.edu.cn/simple/
```

图 1–14　输入配置命令

（6）完成环境安装，打开 Python 文件可以开始使用，如图 1–15 所示。

File　Edit　View　Insert　Cell　Kernel　Help

▶运行　代码

```
In [2]:
import os
print(os.path.abspath('.'))

C:\Users\lenovo

In [3]:
print("Hello, World!")

Hello, World!

In [6]:
#!/usr/bin/python

list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

图 1–15　完成环境安装

本章习题

一、选择题

1. 一个完整的计算机系统应包括（　　）。

A. 主机和外设

B.CPU 和内存

C. 硬件和软件

D. 软件和主机

2. Python 语言源程序的执行方式是（　　）。

A. 编译执行

B. 解释执行

C. 直接执行

D. 边编译边执行

3. 下面选项中，不属于 Python 语言特点的是（　　）。

A. 简单易学

B. 开源

C. 面向过程

D. 可移植性

二、填空题

1. Python 是一种_____类型的编程语言。

2. Python 自带的开发工具是_____

三、判断题

1. Python 是开源的，它可以被移植到其他平台上。（　　）

2. Python 程序被解释器转换后的文件格式后缀名为 .pyc。（　　）

3. Python 是开源免费的，不需要付费，也不存在商业风险。（　　）

四、程序题

编写一个 Python 程序，输出以下语句。

```
学 Python，从现在开始。
学 Python，从现在开始。
```

第 2 章

Python 起步必备

素养目标

◎培养学生对 Python 编程的基础知识的理解和掌握能力。

◎培养学生养成一个良好的 Python 编程习惯。

◎培养学生用 Python 解决简单问题的能力。

知识目标

◎ 掌握基本的 Python 语法，包括缩进分层、代码注释和语句断行。

◎ 掌握 Python 的基本数据类型，包括简单的数据类型和复杂的数据结构等。

◎ 掌握 Python 的数据类型转换和基础的数据操作。

技能目标

◎能够运用 Python 语法编写简单的 Python 程序。

◎能够根据需要选择并使用适当的 Python 数据类型。

◎ 能够运用 Python 数据类型进行基本的数据操作，如创建、查询、修改删除和运算等。

任何一门语言 (自然语言和编程语言) 都有一定的语法，以及最基础的知识。要学好语言，就必须从基础知识学习。牢牢掌握基础知识，才能灵活地运用这些基础知识，来解决更复杂的问题。Python 的语法非常简单，学习和掌握起来也很容易。在开始探讨 Python 编程的深层次应用之前，我们需要先理解 Python 的基础知识，在这一章里，我们将介绍 Python 语法基础和 Python 数据类型，这些都是学习 Python 编程的基础。本章的学习将为后续章节中更复杂的 Python 编程知识打下坚实的基础，希望同学们能够通过实践和练习，熟练掌握这些基础知识，为自己的经济金融学习和研究工作打下坚实的基础。

2.1 Python 语法基础

在本节中，我们将介绍 Python 的基础语法规则。掌握这些规则对于编写清晰、高效的 Python 代码至关重要。本节内容将涉及缩进分层、代码注释和语句断行三个方面。在开始本节内容之前对 Python 的基础知识进行入门介绍。

（1）数据。人类是通过语言将自己要表达的信息传达给别人的。计算机不能识别人类语言，它是通过数据来传达信息的。不同的数据需要使用不同类型的方式进行表示和存储。

（2）数据类型。在 Python 中，数据类型包括数字类型 (整数、浮点数、复数)、文本类型 (字符串)、状态类型 (布尔类型)、其他类型 (集合、列表、元组、字典)。

（3）表达式。表达式是由运算符和参与运算的数据组成的式子。参与运算的数据可以是数值本身，也可以是变量。表达式运算后得出的结果类型由参与的数据和运算符共同决定。其中，运算符不是必需的，单独的一个值也可以是一个表达式，单独的一个变量也可以是一个表达式。例如，下面的代码都是表达式。

```
50
a=3+10
25>10
c=2* b
```

2.1.1 缩进分层

在 Python 中，缩进不仅仅是代码美观的问题，它是语法的一部分。正确的缩进表示代码块的层次结构，使得程序的逻辑关系一目了然。

例如，一个简单的条件判断语句：

```
if condition:
# 如果 condition 为 True，执行这里的代码
    perform_action()
```

在这里，perform_action() 函数调用是属于 if 条件语句内的操作，因为它被缩进了。

缩进错误是新手常犯的错误之一，例如：

```
if condition:
perform_action() # 缩进错误，这会导致运行时错误
```

这会导致运行结果出现 IndentationError，因为 perform_action() 没有正确缩进来表示它是 if 条件语句的一部分。

Python 官方推荐使用四个空格作为标准缩进量，而不是使用制表符 Tab，但大多数 Python IDE 和文本编辑器通常有将制表符自动转换为 4 个空格的功能。

2.1.2 代码注释

注释是解释代码的文本，它不会被 Python 解释器执行。注释可以帮助其他人或者自己理解代码的意图和工作方式。Python 中有两种类型的注释：单行注释和多行注释。单行注释以 # 开头，例如：

```
# 这是一个单行注释
balance = 1000  # 注释也可以出现在代码行的尾部
```

多行注释可以用三个引号来表示，虽然实际上这是多行字符串，但如果没有赋值给变量，也常被用作多行注释：

```
"""
这是一个多行注释的示例。
可以在这里写多行的注释内容，
包括说明函数的功能、输入输出等信息。
"""
```

以上是一个多行注释，多行注释它可以跨越多行，用来描述复杂的操作和逻辑。注释应该简洁明了，直接说明代码的目的和功能。注释不应该用来描述代码的具体实现，因为这可以直接从代码本身读取。注释最重要的是解释代码为什么这样写，以及这段代码在上下文中是如何工作的。对于复杂的算法或者关键的业务逻辑，适当的注释是必要的。另外，注释还应该保持更新，避免因为代码变动后，注释和代码不匹配的情况。

2.1.3 语句断行

有时候，为了提高代码的可读性，我们需要将长代码行分割成几行。Python 使用反斜杠 (\) 作为续行符，来表示下一行是当前语句的延续。

```
total = item_one_price + \
item_two_price + \
item_three_price
```

但是在括号、方括号和花括号中，我们不需要使用反斜杠。在下面的例子中，Python 解释器能够理解这些括号内的表达式是一个整体，因此不需要使用反斜杠。

```
total = (
item_one_price +
```

```
    item_two_price +
    item_three_price)
```

Python 官方建议每行代码的长度不超过 79 个字符。这个规则的目的是为了提高代码的可读性，因为在大多数编辑器和终端中，过长的代码行可能需要滚动才能阅读全行。当然，这只是一个建议，有些项目可能会选择更大的长度限制，比如 99 个字符。

掌握 Python 的基础语法对于编写有效的 Python 代码是至关重要的。缩进分层帮助我们定义程序结构，代码注释让我们的代码更加易懂，而语句断行则提高了代码的可读性。在接下来的章节中，我们将利用这些基础知识，探讨 Python 在经济金融领域中的应用。

2.1.4 综合应用

输出带有注释的《望庐山瀑布》，主要任务：

（1）输出《望庐山瀑布》这首诗；

（2）显示注释信息。

打开 Jupyter Notebook，定义若干变量，分别用于表示《望庐山瀑布》的诗名、每条诗句、注释信息、作者，并保存代码。编写的代码如下所示：

```
Poem_Name='望庐山瀑布'
Name='李白'
Poem_1='日照香炉生紫烟，'
Poem_2='遥看瀑布挂前川。'
Poem_3='飞流直下三千尺，'
Poem_4='疑是银河落九天。'
Note_1='香炉：指香炉峰'                    #注释 1
Note_2='川：河流，这里指瀑布'              #注释 2
Note_3='三千尺：形容山高'                  #注释 3
Note_4='九天： 一作“半天”'                #注释 4
print('《',Poem_Name,'》')
print('作者：',Name)
print(Poem_1)                              #香炉：指香炉峰
print(Poem_2)                              #川：河流，这里指瀑布
print(Poem_3)                              #三千尺：形容山高
print(Poem_4)                              #九天： 一作“半天”
print('********** 注释 **********')
print(Note_1)
print(Note_2)
print(Note_3)
print(Note_4)
```

点击 Jupyter Notebook 左上角的三角形按钮，执行代码，运行结果如图 2–1 所示。

```
《 望庐山瀑布 》
作者：  李白
日照香炉生紫烟，
遥看瀑布挂前川。
飞流直下三千尺，
疑是银河落九天。
**********注释**********
香炉：指香炉峰
川：河流，这里指瀑布
三千尺：形容山高
九天：  一作“半天”
```

图 2-1　综合应用运行结果

2.2　变量与常量

在 Python 编程中，变量和常量是最基本也是最重要的概念之一，它们是我们处理和存储数据的方式。对于经济金融学生来说，理解变量和常量的概念和使用方法，将有助于我们更好地利用 Python 进行数据处理和分析，从而更准确、更有效地解决实际问题。

2.2.1　变量

在 Python 中，变量通常被定义成一个指向值或数据的引用，但这并不好理解，有一种好理解的定义是，变量是可以赋值的标签。可以将变量指向一个数，也可以将其指向一个列表，或者将其指向一个文件，等等，总之，变量本身只是一个标签，指向了特定的值。

下面尝试在 hello_world.py 中使用一个变量，在这个文件开头添加一行代码，并对第二行代码进行修改，如下所示：

```
message =  “Hello World!”
print(message)
```

运行这个程序，运行结果如下：

```
   Hello World!
```

从结果发现该程序运行结果和以前输出的结果完全相同。

在上述的示例中添加了一个名为 message 的变量，指向的值为文本“Hello World!”，因此输出 message 的结果即是“Hello World!”。

为了提高程序的可读性，可以在一行代码中给多个变量赋值，只要变量和值的个数相同，Python 就能正确地将它们关联起来，如：

```
x,y,z = 0, 0, 0
```

上述代码表示将变量 x、y 和 z 都初始化为 0。

在使用变量的过程中，需要遵守一些规则，有关变量的规则如下。

（1）变量名只能包含字母、数字和下划线。变量名可以是字母或下划线开头，但不能以数字开头。例如，变量名可以为 message_1，但不能为 1_ message。

（2）变量名不能包含空格，但可以使用下划线来分隔其中的单词。例如 message_ex 是有效的，但 message ex 则会引发错误。

（3）不要将 Python 关键字和函数名用作变量名，否则将引发错误。

（4）变量名应既简短又具有描述性。例如，name 比 n 更好，student_name 比 s_n 更好。

（5）变量名是区分大小写的，这意味着 Name 和 name 是两个不同的变量。

（6）慎用小写字母 l 和大写字母 O，因为它们可能会被错看数字 1 和 0。

2.2.2 常量

常量类似于变量，但其值在程序的整个生命周期内保持不变。Python 中没有内置的常量类型，但一般我们使用全大写来指出将某个变量视为常量，其值始终保持不变，例如设置 PI 为一个常量。

```
PI = 3.1415926
```

但这只是一种约定，Python 本身并不会阻止我们去修改这些“常量”的值，因此本质上常量也是一种变量，只是我们约定用全大写方式表达的变量为常量，并不去修改其值。

2.3 简单的数据类型

数据类型是程序的基础。程序设计的本质就是对数据进行处理。要想准确地处理数据，必须了解要处理数据对象的数据类型，只有这样才能在处理时根据其类型进行正确运算。否则，不仅不能正常地处理数据，而且连程序也不能正常运行。

Python 语言中的数据类型很多，主要有简单数据类型和结构数据类型。简单数据类型就是我们日常生活中经常使用的数据，本节主要介绍这些简单的数据类型。在金融领域，经常使用数来记录排名、表示可视化数据、存储企业信息等等，Python 能根据数的用法以不同的方式处理它们。数是一个重要的概念。Python 支持多种数据类型，包括整数、浮点数、字符串、列表、元组、集合和字典等。

2.3.1 整数类型

在 Python 中，整数类型 (Integer) 是数值数据类型的一种，用于表示整数。在 Python 中，只有一种整数类型：int，它可以表示正数、负数和零。创建整数对象非常简单，只需要直接给变量赋值即可，例如：

```
age = 18
```

在这个例子中，我们创建了一个名为 age 的变量，并赋值为 18，这个 18 就是一个整数对象。在

Python 中，可对整数进行加 (+) 减 (-) 乘 (*) 除 (/) 运算，并支持“先乘除、后加减、有括号先算括号”的运算次序。

```
>>> 2 + 3
5
>>> 3 – 2
1
>>> 2 * 3
6
>>> 3 / 2
1.5
>>> 2 + 3*4
14
>>> (2+3) * 4
20
```

在上述示例中，空格不影响计算表达式的结果，只是为了阅读方便，另外，Python 中可使用两个乘号表示乘方运算，如：

```
>>> 3 ** 3
27
```

在 Python 中，整数长度受限于可用内存的大小，没有固定的位数限制，这意味着在 Python 中处理非常大的整数时，不需要担心整数溢出的问题。在经济金融领域，整数类型常常用于表示数量、计数、排名等。例如，一个公司的员工数量就是一个整数，股票的排名也是一个整数。

总的来说，理解和掌握 Python 的整数类型是编程的基础，它是我们进行数据处理和分析的基础工具。在后续的章节中，我们将看到更多使用整数的实例。

2.3.2 浮点数类型

在 Python 中，浮点数类型 (Float) 是用来表示有小数点的数。它对于经济金融来说非常重要，因为在这个领域中，许多计算涉及到利率、价格和其他需要小数精度的数值。创建浮点数和创建整数一样简单，只需将数值赋给一个变量即可，浮点数可以直接用小数点表示。例如：

```
interest_rate = 0.05            # 表示 5% 的利率
market_price = 123.45           # 表示市场价格
```

和整数一样，浮点数也支持基本的数值运算，但是需要注意的是，Python 中的除法运算总是返回一个浮点数，即使是可以整除的两个整数相除也是如此，如果想进行整除 (即不考虑余数)，可以使用 // 运算符。如：

```
>>> 4 / 2
2.0
>>> 3 // 2
1
```

浮点数在计算机中的表示和存储并不是完全精确的，这可能导致一些意外的结果。比如我们都知道 0.1+0.2 = 0.3，那么将 0.1+0.2 和 0.3 比较大小是不是一定相等呢，看下结果：

```
>>> print(0.1 + 0.2 == 0.3)
False
```

从输出结果可以看到，两个看似相等的浮点数进行比较时并不相等，那这是为什么呢？现在我们尝试用 Python 计算 0.1+0.2 的值，看看输出结果是多少：

```
>>> print(0.1 + 0.2)
0.30000000000000004
```

可以看到在 Python 中 0.1+0.2 并不完全等于 0.3，这是因为在计算机中表示十进制的小数时可能会出现精度的损失，这种情况在处理金融数据时必须特别小心，因为它可能导致重大的金融计算错误。对于这种精度问题，Python 提供了 decimal 模块来处理，它允许设置小数的精度，并进行精确的小数计算，这在需要高精度计算的金融领域中非常有用。例如：

```
from decimal import Decimal, getcontext
getcontext().prec = 6                    # 设置全局精度为 6 位
d1 = Decimal('0.1')
d2 = Decimal('0.2')
print(d1 + d2 == Decimal('0.3'))         # 这会输出 True
```

在经济金融学中，浮点数可以用来表示各种连续的数据，如货币金额、利率、价格指数等。掌握浮点数的使用和它们的特性对于经济金融领域的学生来说非常重要。

总之，浮点数是 Python 中处理数值数据的重要类型，特别是在需要进行小数点计算的场合。然而，由于其内在的不精确性，使用时需要小心，尤其是在经济金融等对精度要求较高的领域。了解如何使用浮点数以及如何通过 decimal 模块进行精确控制，将有助于在将来的学习和工作中进行有效的数值计算。

2.3.3 大数表示方法

在经济金融领域，大数据和精确计算成为决策的重要工具，因此，理解和熟练使用 Python 中的大数表示方法，有助于在处理大数据和进行复杂计算时，编写出更清晰、更准确的代码。在这里，主要介绍 Python 中的两种大数表示方法：使用下划线和科学计数法。

1. 使用下划线表示大数

在 Python 中，我们可以使用下划线“_”来表示较大的数字，以提高代码的可读性。这种表示方法并不会改变数字的值，而只是作为一种可选的格式，帮助更清楚地识别和理解大数。例如：

```
one_million = 1_000_000
print(one_million)                       # 输出：1000000
```

在上述代码中，我们使用下划线“_”将数字 1_000_000 分隔开，使其更容易读取和理解。但这并

不影响其值，打印 one_million 仍然输出 1000000。

2. 科学计数法表示大数

科学计数法是一种表示非常大或非常小的数的有效方式。在 Python 中，可以使用字母 e 或 E 来表示 10 的幂。例如：

```
one_billion = 1e9
print(one_billion)  # 输出：1000000000.0
```

在上述代码中，1e9 表示的是 1*10^9，即十亿。使用科学计数法，可以方便、精确地表示非常大或非常小的数。

知识拓展

在大多数程序设计语言中，保存整数的变量都有一个值的区域，当超过该区域后，就无法保存更大的数。例如，在 C 语言中，int 型的变量使用 4 个字节来保存值，保存数据的范围为 −2147483648~2147483647。如果计算结果为更大的值，则无法保存，即便使用 long 型，仍然有一个范围限制。这样，就无法进行如阶乘、大的幂运算等，要保存这些结果值很大的数据，就需要另外编写大整数处理程序。

幸运的是，Python 直接提供了对大整数的支持，用户可以直接调用。例如，在交互命令状态下执行以下的幂运算，可得到一个很大的整数。

```
>>>   99**99
369729637649726772657187905628805440595668764281741102430259972423552570455277523421410650010128232727940978889548326540119429996769494359451621570193644014418071060667659301384999779999159200499899
```

在上面的运算中，两个星号表示进行幂运算。从上面的结果可看出，计算结果有 198 位，远远超出了普通整型变量的表示范围，但是，Python 处理起来比较简单和便捷。

2.3.4 布尔类型

布尔类型主要用来判断数据条件是否成立，它只包含两个值 True(逻辑为真) 和 False(逻辑为假)，这两个值称为布尔值。Python 中的布尔值可以转化为数值。其中，True 表示 1，而 False 表示 0。

```
>>>2**3>2*3
True
```

如上代码，执行结果为 True，表示 2**3>2*3 成立，即 2 的立方（8）大于 2 乘以 3 的值（6），这是正确的，因此输出结果为 True。其中“>”是 Python 中的运算符，用来比较数值。

提示：布尔类型的布尔值是区分大小写的。例如，这里是 True 而不是 true 和 TRUE。

2.3.5 类型转换

在 Python 中，常用的数据类型之间是可以相互转换的，可通过以下函数进行转换。

（1）bin( )：可以将整数类型转换为二进制整数的字符串类型。

（2）oct( )：可以将整数类型转换为八进制整数的字符串类型。

（3）hex( )：可以将整数类型转换为十六进制整数的字符串类型。

（4）int( )：可以将整数、浮点数、字符串类型转换为十进制整数类型。

（5）float( )：可以将整数、字符串类型转换为浮点数类型。

（6）str( )：可以将整数、浮点数类型转换为字符串类型。

此外，str( ) 可以创建一个空字符串，int( ) 也可以建立一个默认值为 0 的整数，float( ) 可以建立一个默认值为 0.0 的浮点数。

下列是一些进行类型转换的代码程序：

```
>>> bin(60)                # 整数类型转换为二进制整数的字符串类型
'0b111100'
>>> bin(0o123)             # 八进制整数类型转换为二进制整数的字符串类型
'0b1010011'
>>> oct(60)                # 整数类型转换为八进制整数的字符串类型
'0o74'
>>> oct(0b1010)            # 二进制整数类型转换为八进制整数的字符串类型
'0012'
>>> hex(60)                # 整数类型转换为十六进制整数的字符串类型
'0x3c'
>>> hex(0o123)             # 八进制整数类型转换为十六进制整数的字符串类型
'0x53'
>>> int(0b1010)            # 二进制整数转换为十进制整数
10
>>> int(0o123)             # 八进制整数转换为十进制整数
83
>>> int(0x la2b)           # 十六进制整数转换为十进制整数
6699
>>> int(6.8)               # 浮点数转换为十进制整数
6
>>> int('60')              # 字符串转换为十进制整数
60
>>> float(60)              # 整数转换为浮点数
60.0
>>>float(0b1010)          # 二进制整数转换为浮点数
10.0
>>> float(0o123)           # 八进制整数转换为浮点数
83.0
>>>float(0x la2b)         # 十六进制整数转换为浮点数
6699.0
>>>float('60')            # 字符串转换为浮点数
```

```
60.0
>>> str(60)                # 整数转换为字符串
'60'
>>> str(0b1010)            # 二进制整数转换为字符串
'10'
>>> str(0o123)             # 八进制整数转换为字符串
'83'
>>> str(0x 1a2b)           # 十六进制整数转换为字符串
'6699'
>>> str(6.8)               # 浮点数转换为字符串
'6.8'
```

2.4　常用的运算符

处理数据就需要用到运算符，与其他高级程序设计语言一样，Python 也提供了丰富的运算符，下面介绍常用的运算符。

2.4.1 算术运算符

算术运算符可以实现数学运算，如加、减、乘、除、求余等运算。Python 的算术运算符及其作用如表 2–1 所示。

表 2–1　算术运算符及其作用

运算符	作用
+	加法运算
–	减法运算
*	乘法运算
/	除法运算
%	求余数运算
//	整除运算
**	幂运算

以下演示的是利用算术运算符进行运算的代码语言：

```
>>>   20+30              # 加法运算
50
>>>   20-30              # 减法运算
-10
>>>  20*30               # 乘法运算
600
```

```
>>>15/2                    # 除法运算
7.5
>>>15//2                   # 整除运算
7
>>>15%82                   # 求余运算
15
>>>2**3                    # 幂运算
8
```

2.4.2 赋值运算符

赋值运算符主要用来为变量赋值，将运算符右侧的常量或变量的值赋值到运算符左侧的变量中。Python 的赋值运算符及其作用如表 2–2 所示。

表 2–2　赋值运算符及其作用

赋值运算符	作用
=	直接赋值
+=	增强加法赋值
–=	增强减法赋值
*=	增强乘法赋值
/=	增强除法赋值
%=	增强求余赋值
//=	增强整除赋值
**=	增强幂赋值

以下演示的是利用赋值运算符进行运算的代码语言：

```
>>>a=8                    # 直接赋值
>>>a+= 2                  # 增强加法赋值
>>> print(a)
10
>>> a=8
>>>a-=6
>>> print(a)
2
>>> a=8
>>>a*=2                   # 增强乘法赋值
>>> print(a)
16
>>> a=8
>>> a/=2                  # 增强除法赋值
>>> print(a)
4.0
>>>a=8
```

```
>>>a//=2                    # 增强整除赋值
>>> print(a)
4
>>> a = 8
>>> a%=2                    # 增强求余赋值
>>> print(a)
0
>>> a=8
>>>a**=2                    # 增强幂赋值
>>> print(a)
64
```

2.4.3 比较运算符

比较运算符是对数值变量或表达式的结果进行比较。如果比较结果为真，返回 True；如果为假，返回 False。Python 的比较运算符及其作用如表 2–3 所示。

表 2–3　比较运算符及其作用

比较运算符	作用
==	等于，比较两个对象是否相等
!=	不等于，比较两个对象是否不相等
>	大于
<	小于
>=	大于等于
<=	小于等于

以下演示的是利用比较运算符进行运算的代码语言：

```
>>>100=99          # 等于比较
False
>>>100!=99         # 不等于比较
True
>>>100>99          # 大于比较
True
>>>100<99          # 小于比较
False
>>>100>=99         # 大于等于比较
False
>>>100<=99         # 小于等于比较
False
```

2.4.4 位运算符

在计算机中，程序中无论是数字、文字，还是图像，都是二进制形式存储的。为了方便计算，

Python 还提供了位运算符。位运算符是直接按照数值的二进制形式进行运算。Python 的位运算符及其作用如表 2–4 所示。

表 2–4　位运算符及其作用

位运算符	作用
&	位与运算 (相同位都为 1，则为 1；若有一个不为 1，则为 0)
|	位或运算 (相同位有一位为 1，则为 1；反之，则为 0)
^	异或运算 (相同位的数值不同，则为 1；相同位的数值相同，则为 0)
~	取反运算 (将二进制数加 1，然后取其负值)
<<	左移位运算 (二进制后面以 0 补充相应的位数)
>>	右移位运算 (二进制后面移除相应的位数)

以下演示的是利用位运算符进行运算的代码语言：

```
>>>0b01001000 == 0b01001000              # 位与运算
True
>>>bin(0b01001000|0b01001000)            # 进行位或运算并转换成二进制
'0b1001000'
>>>bin(0b01001000^0b01001000)            # 异或运算并转换成二进制
'0b0'
>>>bin(~0b01001000)                      # 取反运算并转换成二进制
'-0b1001001'
>>>bin(0b01001000<<3)                    # 左移位运算并转换成二进制
'0b1001000000'
>>>bin(0b01001000>>3)                    # 右移位运算并转换成二进制
'0b1001'
```

2.4.5 逻辑运算符

逻辑运算符是对真和假两种布尔值进行运算，运算结果仍然是一个布尔值。Python 的逻辑运算符及其作用如表 2–5 所示。

表 2–5　运算符及其作用

逻辑运算符	作用
and	逻辑与运算
or	逻辑或运算
not	逻辑非运算

为了更方便进行逻辑运算，下面依次给出各逻辑运算的规则技巧。逻辑与 (and) 的运算规则及结果如表 2–6 所示。

表 2–6　逻辑与 (and) 运算的规则及结果

x(逻辑运算符左侧)	y(逻辑运算符右侧)	结果
真 (1)	真 (1)	真 (1)

续表

x(逻辑运算符左侧)	y(逻辑运算符右侧)	结果
真 (1)	假 (0)	假 (0)
假 (0)	假 (0)	假 (0)
假 (0)	真 (1)	假 (0)

巧记：同为真时为真，有一个为假就为假。

逻辑或 (or) 的运算规则及结果如表 2–7 所示。

表 2–7　逻辑或 (or) 运算的规则及结果

x(逻辑运算符左侧)	y(逻辑运算符右侧)	结果
真 (1)	真 (1)	真 (1)
真 (1)	假 (0)	真 (1)
假 (0)	假 (0)	假 (0)
假 (0)	真 (1)	真 (1)

巧记：同为假时为假，有一个为真就为真。

逻辑非 (not) 的运算规则及结果如表 2–8 所示。

表 2–8　逻辑非 (not) 运算的规则及结果

y(逻辑运算符右侧)	结果
真	假
假	真

巧记：原为真，现为假；原为假，现为真。

以下演示的是利用逻辑运算符进行运算的代码语言：

```
>>>12>7  and  9>6                # 逻辑与运算
True                             # 同为真，结果为真
>>>12>17  and   9>6              # 逻辑与运算
False                            #12>17 为假，结果为假
>>>12>7  or  9>6                 # 逻辑或运算
True                             #2>7 为真，9>6 为真，结果为真（2 个都为真，就为真）
>>>12>17  or   9>6               # 逻辑或运算
True                             #9>6 为真，结果为真（有一个为真，就为真）
>>>12>17  or   9>16              # 逻辑或运算
False                            #12>17 为假，9>16 为假，结果为假（同为假，结果为假）
>>>not 12>7                      # 逻辑非运算
False                            #12>7 为真，结果为假
>>>not  12>17                    # 逻辑非运算
True                             #12>17 位假，结果为真
```

2.4.6 成员运算符

除了上述运算符之外，Python 还支持成员运算符，用于检查某个数据是否存在于某包含多个成员的数据类型 (字符串、列表、元组、字典等) 之中。Python 的成员运算符及其作用如表 2–9 所示。

表 2–9　成员运算符及其作用

成员运算符	作用
in	如果在指定的序列中找到值就返回 True，否则返回 False
not in	如果在指定的序列中没有找到值就返回 True，否则返回 False

2.4.7 运算符优先级

一个运算中可能会包含多种运算符，在运算时会根据运算符的优先级依次进行运算，先运算优先级高的，再运算优先级低的。Python 中各种运算符的优先级从高到低顺序如表 2–10 所示。

表 2–10　运算符优先级

运算符	描述
**	指数 (最高优先级)
~	相反运算
+、—	正数、负数运算符
*、/、%、//	乘、除、取余、取整 (整除)
+、—	加减法运算
<<	左移位运算
>>	右移位运算
&	位与运算
^	异或运算
|	位或运算
<=	小于等于运算
<	小于运算
>=	大于等于运算
>	大于运算
==、!=	等于和不等于运算
**=	指数增强赋值运算
*=、/=、%=、//=	乘除增强赋值运算
+=、 −=	加减增强赋值运算
not	逻辑非运算
and	逻辑与运算
or	逻辑或运算

2.4.8 综合应用

实现运算结果比较。

任务描述：

（1）要求用户依次输入要进行运算的 2 个表达式；

（2）编写代码进行运算，并比较运算结果，得出结论。

打开 Jupyter Notebook，使用变量 A 接收用户输入的第 1 个运算，使用变量 B 接收用户输入的第 2 个运算。当用户输入完成以后，比较这两个运算给出结果。编写的代码如下：

```
A = eval(input('请输入第一个数A：'))
B = eval(input('请输入第二个数B：'))
if A > B:
    print('比较结果：A > B')
if A == B:
    print('比较结果：A = B')
if A < B:
    print('比较结果：A < B')
```

在代码的书写过程中需要注意 if 语句的缩紧，具体代码执行结果如图 2–2 所示。

```
请输入第一个数A:  5*61-50
请输入第二个数B:  56**2
比较结果：A < B
```

图 2–2　运算结果图

本章习题

一、选择题

1. Python 语言的缩进不可以使用（　）。

A. 空格　　B.Tab

C.\　　D. 空格和 Tab

2. 以下关于 Python 语言中的注释说法错误的是（　）。

A. 注释语句可以被执行

B. 注释语句以符号“#”开头

C. 多行注释可以用“”将其包围起来

D. 单行注释可以和非注释语句在同一行，并出现在非注释语句之后

3. 下面符合 Python 命名规范的标识符是（　）。

A.if

B.name

C.3n ame

D.na@me

二、填空题

1. 布尔类型的值包括_____和_____。
2. 在 Python 中，int 表示的数据类型是_____。
3. 使用_____判断数据类型。

三、判断题

1. Python 的单行注释使用的符号是“#”。（　　）
2. Pythpn 的标识符可以以数字开头。（　　）
3. Python 的成员运算符用于判断指定序列中是否包含某个值。（　　）

四、程序题

输入矩形的长和宽的长度 a、b，编写程序求矩形的周长和面积。