# 第 1 章

# Python 概述

本章主要介绍了计算机及程序设计语言发展史、Python 程序设计的背景知识和基础语法。

本章要求了解计算机发展历史和体系结构，了解程序设计语言的发展过程，掌握 Python 运行环境的安装过程，学会编写简单的 Python 程序。

学习目标：

- 计算机及程序设计语言概述
- Python 语言特点和应用领域
- Python 版本和开发环境
- 集成开发环境 Spyder 的使用
- Python 语法基础和输入输出函数

## 1.1 计算机及程序设计语言概述

### 1.1.1 计算机发展史

计算机（Computer）的全称为电子计算机，也称电脑，是一种能够按照程序运行，自动、高速处理海量数据的现代化智能电子设备。计算机的应用领域从最初的军事科研应用扩展到社会的各个领域，已形成了规模巨大的计算机产业，带动了全球范围的技术进步，由此引发了深刻的社会变革。目前，计算机已经成为信息社会中必不可少的工具。

电子计算机的奠基人是英国科学家艾伦·麦席森·图灵（Alan M. Turing）和美籍匈牙利科学家冯·诺依曼（John Von·Neumann）。图灵建立了图灵机的理论模型，奠定了人工智能的基础，冯·诺依曼率先提出了计算机体系结构的设想。

**第一代电子管计算机（1946—1958 年）**

1946 年 2 月 15 日，标志现代计算机诞生的电子数字积分计算机（Electronic Numerical Integrator and Calculator，ENIAC）在美国费城公诸于世。为了满足美国军方计算弹道的需

要，美国政府和宾夕法尼亚大学合作研发 ENIAC，共使用了 17840 支电子管，70000 多个电阻器，有 5 百万个焊接点，重达 28 吨，耗电 170 千瓦，造价约为 487000 美元，其运算速度为每秒 5000 次。

1950 年，世界上第一台并行计算机离散变量自动电子计算机（Electronic Discrete Variable Automatic Computer，EDVAC）诞生，采用了“计算机之父”冯·诺依曼的两个设想：二进制和存储程序。

**第二代晶体管计算机（1954—1964 年）**

1948 年，晶体管的应用代替了体积庞大的电子管，电子设备的体积不断减小。1954 年，IBM 公司制造了第一台使用晶体管的计算机 TRADIC（TRAnsistor Digital Computer），增加了浮点运算。第二代计算机体积小、速度快（一般为每秒数 10 万次，可高达 300 万次）、功耗低、性能更稳定。

这一时期出现了更高级的 COBOL 和 FORTRAN 等语言，使计算机编程更容易。新的职业（程序员、分析员和计算机系统专家）和整个软件产业也由此诞生。应用领域以科学计算和事务处理为主，并开始进入工业控制领域。

**第三代集成电路计算机（1964—1970 年）**

1958 年德州仪器公司发明了集成电路（Integrated Circuit，IC），将三种电子元件结合到一片小小的硅片上。更多的元件集成到单一的半导体芯片上，使得计算机变得更小，功耗更低，速度更快。第三代计算机的基本电子元件是每个基片上集成几个到十几个电子元件（逻辑门）的小规模集成电路和每片上几十个元件的中规模集成电路。运算速度可达每秒数百万次至数千万次基本运算。

这一时期的发展还包括使用了操作系统，使得计算机在中心程序的控制协调下可以同时运行许多不同的程序。软件方面出现了分时操作系统以及结构化、规模化程序设计方法。机器可靠性有了显著提高，价格进一步下降，产品走向了通用化、系列化和标准化之路。应用领域开始进入文字处理和图形图像处理领域。

**第四代大规模集成电路计算机（1970 年至今）**

1967 年和 1977 年分别出现了大规模和超大规模集成电路。由大规模和超大规模集成电路组装成的计算机，被称为第四代电子计算机。美国伊利诺伊自动计算机（ILLIAC-IV）是第一台全面使用大规模集成电路作为逻辑元件和存储器的计算机，它标志着计算机的发展到了第四代。到了 20 世纪 80 年代，超大规模集成电路（Very Large Scale Integration，VLSI）在芯片上容纳了几十万个元件，后来的甚大规模集成电路（Ultra Large Scale Integration，ULSI）上将数量扩充到百万级，可以在硬币大小的芯片上容纳如此数量的元件使得计算机的体积和价格不断下降，而功能和可靠性不断增强，运算速度可达每秒一亿甚至几十亿次。

这一时期在软件方面出现了数据库管理系统、网络管理系统和面向对象语言等。1971 年世界上第一台微处理器（INTEL4004）在美国硅谷诞生，开创了微型计算机的新时代。应用领域从科学计算、事务管理、过程控制逐步走向家庭。

**第五代新型计算机**

基于集成电路的计算机短期内还不会退出历史舞台，同时新的计算机正在跃跃欲试地加紧研究，下一代计算机包括能识别自然语言的计算机、高速超导计算机、纳米计算机、

激光计算机、DNA 计算机、量子计算机和神经网络计算机等，具有体积更小、运算速度更快、更加智能化和耗电量更少等特点。

### 1.1.2 计算机体系结构

冯·诺依曼于 1946 年提出存储程序原理，把程序本身也当作数据来对待，程序和该程序处理的数据用同样的方式储存，计算机的数制采用二进制，按照程序顺序执行。冯·诺依曼的这个理论被称为冯·诺依曼体系结构。

冯·诺依曼体系结构提出计算机由控制器、运算器、存储器、输入设备、输出设备五部分组成，如图 1.1 所示。

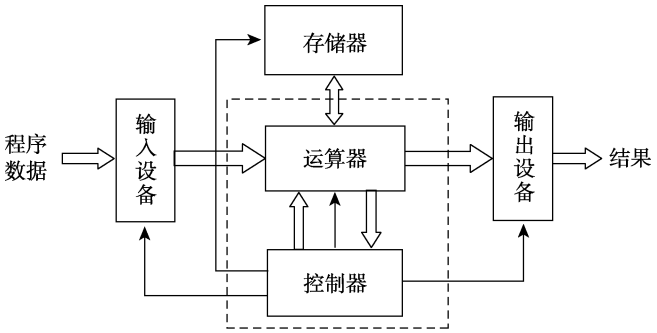


图 1.1 冯·诺依曼体系结构图

运算器：运行算术运算和逻辑运算，并将中间结果暂存到运算器中。

控制器：用来控制和指挥程序和数据的输入运行，以及处理运算结果。

存储器：用来存放数据和程序。

输入设备：用来将人们熟悉的信息形式转换为机器能够识别的信息形式，如键盘、鼠标等。

输出设备：将机器运算结果转换为人们熟悉的信息形式，如打印机、显示器等。

冯·诺依曼体系结构的指令和数据均采用二进制码表示；指令和数据以同等地位存放于存储器中，均可按地址寻访；指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数所在存储器中的位置；指令在存储器中按顺序存放，通常指令是按顺序执行的，特定条件下，可以根据运算结果或者设定的条件改变执行顺序；机器以运算器为中心，输入输出设备和存储器的数据传送通过运算器。

### 1.1.3 计算机系统组成

计算机系统是由硬件系统（Hardware System）和软件系统（Software System）两部分组成的，如图 1.2 所示。

1. 硬件系统

硬件系统是指组成计算机的各种物理设备，也就是人们看得见、摸得着的实际物理设备，包括计算机的主机和外部设备。

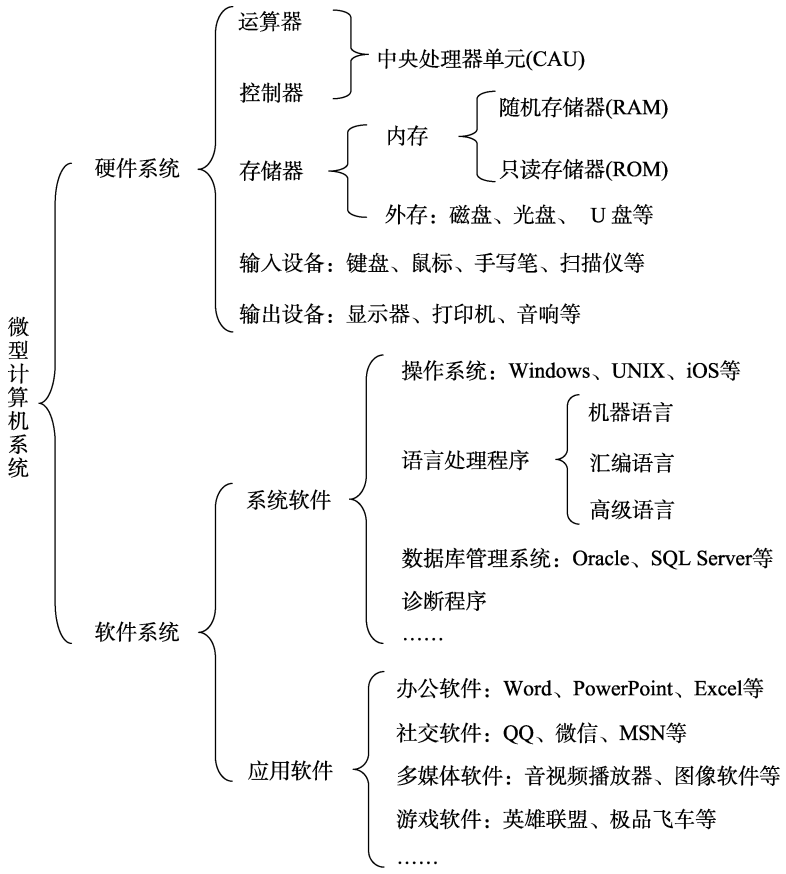


图 1.2　计算机系统组成图

自第一台计算机 ENIAC 发明以来，计算机系统的技术已经得到了很大的发展，但计算机硬件系统的基本结构没有发生变化，仍然属于冯・诺依曼体系计算机。计算机硬件系统仍然由运算器、控制器、存储器、输入设备和输出设备 5 部分组成。

（1）运算器和控制器集成在中央处理器单元（Central Processing Unit，CPU）上。CPU 作为计算机系统的运算和控制核心，被称为计算机的大脑，是信息处理、程序运行的最终执行单元。

运算器：又称算术逻辑单元，它是完成计算机对各种算术运算和逻辑运算的装置，能进行加、减、乘、除等数学运算，也能作比较、判断、查找、逻辑运算等。

控制器：由程序计数器、指令寄存器、指令译码器、时序产生器和操作控制器组成，它是发布命令的“决策机构”，即完成协调和指挥整个计算机系统的操作。控制器是计算机指挥和控制其他各部分工作的中心，其工作过程和人的大脑指挥和控制人的各器官一样。

（2）存储器：将输入设备接收到的信息以二进制的数据形式存到存储器中。存储器有两种，分别叫作内存储器和外存储器。

内存储器：微型计算机的内存储器是由半导体器件构成的。从使用功能上可分为随机存储器（Random Access Memory，RAM）和只读存储器（Read Only Memory，ROM）。

外存储器的种类很多，又称辅助存储器。外存通常是磁性介质或光盘，像硬盘、软盘、

磁带、CD 等，能长期保存信息，但是其速度与内存相比慢，优势是价格相对较低。

（3）输入设备：将数据、程序、文字符号、图像、声音等信息输送到计算机中。常用的输入设备有键盘、鼠标、触摸屏、数字转换器等。

（4）输出设备：将计算机的运算结果或者中间结果打印或显示出来。常用的输出设备有：显示器、打印机、绘图仪和传真机等。

2. 软件系统

软件系统是指由系统软件和应用软件组成的计算机软件系统，它是计算机系统中由软件组成的部分。

计算机软件分为系统软件和应用软件两大类：

（1）系统软件：负责管理计算机系统中各种硬件，使得它们可以协调工作。系统软件使得计算机使用者和其他软件将计算机当作一个整体而不需要顾及到底层每个硬件是如何工作的。

系统软件是指各类操作系统，如 Windows、Linux、UNIX、MacOS 等，包括操作系统的补丁程序及硬件驱动程序，都可归属于系统软件类。它也可以是一个由众多独立程序组成的庞大的软件系统，比如数据库管理系统。

（2）应用软件：为了某种特定的用途而被开发的软件。它可以是一个特定的程序，比如网络浏览器，也可以是一组功能联系紧密互相协作的程序集合，比如微软的 Office 软件。

应用软件包括办公软件、社交软件、多媒体软件、游戏软件等。

### 1.1.4 程序设计语言

程序设计语言（Programming Language）可以简单理解为一种计算机和人都能识别的语言，能够实现人与机器之间的交流和沟通。语言由一组记号和一组规则组成，是根据规则由记号构成的记号串的总体。计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下应当采取的行动。

编程语言处在不断的发展和变化中，从最初的机器语言发展到如今的 2500 种以上的高级语言，每种语言都有其特定的用途和发展轨迹。编程语言并不像人类自然语言发展变化一样的缓慢而又持久，其发展是相当快速的，这主要是因为计算机硬件、互联网和 IT 产业的发展促进了编程语言的发展。

计算机程序设计语言一般分为：低级语言、高级语言和面向对象时代。具体内容如下：

1. 低级语言时代（1946—1953）

包括机器语言以及汇编语言。

1）机器语言

计算机工作基于二进制，计算机只能识别和接受由 0 和 1 组成的指令。这些指令的集合就是该计算机的机器语言。机器语言的缺点有：难学、难写、难记、难检查、难修改，难以推广使用。因此初期只有极少数的计算机专业人员会编写计算机程序。

2）汇编语言

由于机器语言难以理解，莫奇莱等人开始想到用助记符来代替 0、1 代码，于是汇编语言出现了。该语言主要是以缩写英文作为标记符进行编写，运用汇编语言进行编写的一般都是较为简练的小程序，其在执行方面较为便利，但汇编语言程序较为冗长，出错率较高。

2. 高级语言时代（1954—至今）

随着世界上第一个高级语言 FORTRAN 的出现，新的编程语言开始不断涌现出来。

所谓的高级语言，是由多种编程语言结合之后的总称，可以对多条指令进行整合，将其变为单条指令完成输送，在操作细节指令以及中间过程等方面都得到了适当的简化，整个程序更为简便，具有较强的操作性。而这种编码方式的简化，使得计算机编程对于相关工作人员的专业水平要求不断放宽。

1）第一个高级语言——FORTRAN

为了克服低级语言的缺点，20 世纪 50 年代由美国约翰·贝克斯（John Backus）创造出了第一个计算机高级语言——FORTRAN 语言。它很接近人们习惯使用的自然语言和数学语言。程序中所用运算符和运算表达式，很容易理解，使用也十方便，并且 FORTRAN 以其特有的功能在数值、科学和工程计算领域发挥着重要作用。

2）第一个结构化程序设计语言——ALGOL

这是在计算机发展史上首批清晰定义的高级语言，由欧美计算机学家合力在 20 世纪 50 年代所开发。国际计算机学会将 ALGOL 模式列为算法描述的标准，启发了 ALGOL 类现代语言 Pascal、Ada、C 语言等的出现。

3）最简单的语言——BASIC

1964 年 BASIC 语言正式发布。它是由达特茅斯学院院长、匈牙利人约翰·凯梅尼（John G. Kemeny）与数学系教师托马斯·库尔茨（Thomas E. Kurtz）共同研制的。该语言只有 26 个变量名，17 条语句，12 个函数和 3 个命令，这门语言被称为“初学者通用符号指令代码”。

4）编程语言重要的里程碑——Pascal

这是基于 ALGOL 的编程语言，为纪念法国数学家、哲学家、电脑先驱布莱兹·帕斯卡而命名。它由瑞士 Niklaus Wirth 教授于 20 世纪 60 年代末设计。Pascal 具有语法严谨、层次分明等特点，是第一个结构化编程语言，被称为“编程语言重要的里程碑”。

5）现代程序语言革命的起点——C 语言

C 语言的祖先是 BCPL（Basic Combined Programming Language）语言，1970 年，美国贝尔实验室的 Ken Thompson 在 BCPL 语言的基础上，设计出了 B 语言。接着在 1972 到 1973 年间，美国贝尔实验室的 Dennis M.Ritchie 在 Ken Thompson B 语言的基础上设计出了 C 语言。

3. 面向对象时代（90 年代初—至今）

面向对象程序设计最突出的特点为封装性、继承性和多态性。

1）Java

Java 是由 Sun Microsystem 于 1995 年推出的高级编程语言。近几年来，Java 企业级应

用飞速发展，主要被运用于电信、金融、交通等行业的信息化平台建设。Java 是一个普遍适用的软件平台，其具有易学易用、平台独立、可移植、多线程、健壮、动态、安全等主要特性。

2）Python

近几年来，Python 语言上升势头比较迅速，其主要原因在于大数据和人工智能领域的发展，随着产业互联网的推进，Python 语言未来的发展空间将进一步得到扩大。Python 是一种高层次的脚本语言，目前应用于 Web 和 Internet 开发、科学计算和统计、教育、软件开发和后端开发等领域，且有着简单易学、运行速度快、可移植、可扩展、可嵌入等优点。

## 1.2 Python 语言特点及应用领域

Python 是一种简单易学、功能强大的编程语言，具有高效率的高级数据结构和简洁有效的面向对象编程方法。Python 语言具有简练的语法、动态的编程方法和解释执行的属性，已经成为很多领域和平台上的脚本撰写和快速应用开发的理想语言。

### 1.2.1 Python 语言特点

Python 语言具有以下特点。

（1）Python 是结合了解释型、交互型和面向对象的高级语言。

（2）Python 语言风格简洁优雅。基于“对于一个特定问题，只提供一种最好的解决方法”的思路，Python 语言具有简洁明确的语法风格，易读懂、易维护。

（3）Python 语言具有强大的处理能力，集成了模块、异常处理和类的概念，内置支持灵活的数组和字典等高级数据结构类型，完全支持继承、重载、派生、多继承，支持重载运算符和动态类型，源代码易于复用。

（4）Python 语言结构清晰，关键字相对较少，容易学习。

（5）Python 语言对代码行的缩进等编程习惯有严格要求。

（6）Python 语言可扩展性好，提供了丰富的 API 和工具，可作为扩展语言为各种应用开发接口、可编程接口方便对接当前主要的系统、函数库和应用程序，可在 C 或 C++ 中扩展。

（7）Python 语言开发的应用具有很好的可移植性和兼容性，可以在 UNIX、Mac、OS/2、MS-DOS、Windows 等各个版本的操作系统上运行。

（8）Python 语言提供了丰富的标准库和扩展库。Python 标准库功能齐全，提供了系统管理、网络通信、文本处理、文件处理、网页浏览器、数据库接口、电子邮件、密码系统、图形用户界面等操作。除了标准库，Python 还提供了操作系统管理、科学计算、自然语言处理、Web 开发、图形用户界面开发和多媒体应用等多个领域的高质量第三方扩展包。

### 1.2.2 应用领域

Python 语言目前已经广泛应用多个领域，主要包括：

1. 操作系统管理和服务器运维

Python 语言具有易读性好、效率高、代码重用性好、扩展性好等优势，适合用于编写操作系统管理脚本。Python 提供了操作系统管理扩展包 Ansible、Salt、OpenStack 等。

2. 科学计算

Python 科学计算扩展库包括了快速数组处理模块 NumPy、数值运算模块 SciPy、数据分析和建模库 Pandas、可视化和交互式并行计算模块 IPython 和绘图模块 matplotlib 等，其他开源科学计算软件包也为 Python 提供了调用接口，例如计算机视觉库 OpenCV、医学图像处理库 ITK、三维可视化库 VTK 等。因此 Python 开发环境很适合用于处理实验数据、制作图表或者开发科学计算应用程序。

3. 自然语言处理

Python 语言本身可以完成文本处理任务，同时还拥有功能强大的第三方自然语言处理工具库，包括 NLTK、spaCy、Pattern、TextBlob、Gensim、PyNLPI、Polyglot、MontyLingua、BLLIP Parser、Quepy 等，提供了分词、词干提取、词性标注、语法分析、情感分析、语义推理、机器翻译等类库，以及机器学习的向量空间模型、分类算法和聚类算法等丰富的功能。

4. Web 应用开发

Python 提供了多种 Web 应用开发解决方案和模块，可以方便地定制服务器软件，提供了 Web 应用开发框架，如 Django 和 Pyramid 等。微型 Python Web 框架有 Flask 和 Bottle 等，提供的高级内容管理系统有 Plone 和 django CMS 等。提供的工具集包括：Soket 编程、CGI、Freeform、Zope、CMF、Plone、Silva、Nuxeo CPS、WebWare、Twisted Python、CherryPy、SkunkWeb、Quixote、Suite Server、Spyce、Albatross、Cheetah、mod_python 等，Python 标准库支持的 Internet 协议包括：HTML、XML、JSON、E-mail processing、FTP、IMAP 等。

5. 图形用户界面开发

Python 提供的 GUI 编程模块包括 Tkinter、wxPython、PyGObject、PyQt、PySide、Kivy 等，用户可以根据需要编写出强大的跨平台用户界面程序。

6. 多媒体应用

Python 提供了丰富的多媒体应用模块，包括能进行二维和三维图像处理的 PyOpenGL 模块，以及可用于编写游戏软件的 PyGame 模块等。

7. 人工智能

Python 大大简化了人工智能领域的机器学习、神经网络、深度学习等技术的构建和实验运行的难度，人工智能技术和主流算法在 Python 平台上得到了广泛的支持和应用。

## 1.3 Python 版本和开发环境

### 1.3.1 Python 版本

1989 年，Guido van Rossum 与荷兰国家数学和计算机科学研究所共同设计了 Python 语言的雏形。Python 的设计基于多种计算机语言，包括 ABC、Modula-3、C、C++、Algol-68、SmallTalk、UNIX shell 和其他的脚本语言。

Python 语言的第一个公开发行版发行于 1991 年，目前主要使用的版本是 Python 3。Python 开发团队同时维护 Python 2.x 和 Python 3.x 两个系列，Python3.x 的发行时间并不一定晚于 Python 2.x。

本书编写完成时最高版本为 Python 3.8.0，更多版本的更新可以关注 Python 官方网站 https://www.python.org/。

由于 Python 3 在设计时不考虑向下兼容，还有很多 Python 2.x 的代码、第三方扩展库不支持 Python 3，因此作为 Python 的初学者，选择合适的版本是首要问题。选择 Python 版本的参考原则如下：

首先，如果开发的应用对 Python 的版本有特殊要求，应该按此要求选择 Python 的版本。

其次，如果在开发中需要使用特定的第三方扩展库，要注意确定是否与选定版本兼容。

**说明：**

☑ 如果没有特别声明，本书后续内容的讲解均基于 Python 3.7 版本。

### 1.3.2 集成开发环境

Python 是一门跨平台的语言，集成开发环境可以提供 Python 程序开发环境的各种应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具，同时集成代码编写功能、分析功能、编译功能、调试功能等于一体。使用 Python 集成开发环境，可以帮助开发者提高开发的速度和效率，减少失误，也方便管理开发工作和组织资源。

常用的 Python 集成开发环境主要有：

1. IDLE

IDLE 是 Python 内置的集成开发环境。当安装好 Python 以后，IDLE 就自动安装好了。其基本功能包括：语法加亮、段落缩进、基本文本编辑、TABLE 键控制、调试程序等。

2. Anaconda

Anaconda 是 Python 的一个集成安装，完全开源和免费。其中默认安装 Python、IPython、集成开发环境 Spyder 和众多流行的科学、数学、工程、数据分析的 Python 包，支持 Linux、Windows、Mac 等操作系统平台，支持 Python 2.x 和 3.x，可在多版本 Python 之间自由切换。Anaconda 额外的加速、优化是收费的，对于学术用途可以申请免费许可。

Spyder 是一个强大的开放源代码的交互式跨平台 Python 语言科学运算开发环境，集成

了 NumPy、SciPy、Matplotlib 与 IPython 等开源软件。提供高级的代码编辑、交互测试、调试等特性，支持 Windows、Linux 和 OS X 操作系统。Spyder 官方下载网址为 https://pypi.python.org/pypi/spyder，也可以使用 Anaconda 中集成的 Spyder。

3. Eclipse with PyDev

PyDev 是 Eclipse 开发的 Python 集成开发环境，支持 Python、Jython 和 IronPython 的开发。Eclipse+PyDev 插件适合开发 Python Web 应用，功能包括：自动代码完成、语法高亮、代码分析、调试器以及内置的交互浏览器。

PyDev 官方下载网址 http://pydev.org/。

4. PyCharm

PyCharm 是 JetBrains 开发的 Python 集成开发环境，功能包括：调试器、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制等，并支持 Google App Engine 和 IronPython。

Pycharm 专业版是商业软件，提供部分功能受限制的免费简装版本，官方下载地址为 http://www.jetbrains.com/pycharm/。

5. Wing

Wing 是一个功能强大的 Python 集成开发环境，兼容 Python 2.x 和 3.x，可以结合 Tkinter、mod_wsgi、Django、matplotlib、Zope、Plone、App Engine、PyQt、PySide、wxPython 等 Python 框架使用，支持测试驱动开发，集成了单元测试和 Django 框架的执行和调试功能，支持 Windows、Linux、OS X 等操作系统。Wing 的专业版是商用软件，也提供了免费的简装版本，官方下载地址为 https://wingware.com/。

**说明：**

☑ 本书后续内容的讲解将基于 Anaconda Spyder 集成开发环境。

### 1.3.3 Anaconda 安装

Anaconda 是 Python 的免费科学计算集成安装环境，编译集成了 Python 基本环境、交互式解释器 IPython、集成开发环境 Spyder、Python 科学计算和数据挖掘的第三方库 numpy、scipy、matplotlib 等模块。

Anaconda 的安装简单，在 Anaconda 中多版本 Python 和第三方库的安装和维护也简单易行。Anaconda 跨平台性好，可以在 Windows、MacOS 或 Linux 平台上使用。因此，Anaconda 非常适合需要协调不同版本 Python 及其扩展库的开发者和初学者使用。

Python 在不同操作系统平台上的安装和配置过程基本一致，本书基于 Windows 10 和 Python 3.7 搭建 Anaconda Spyder 集成开发环境。

1. Anaconda 的下载和安装

（1）Anaconda 的官方下载地址是 https://www.continuum.io/downloads，如果在国内访

问，建议到清华大学开源软件镜像站下载，地址如下：

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/

本书下载的版本是 Anaconda3-2019.10-Windows-x86_64.exe，是 Windows 操作系统下的 64 位安装包。

（2）下载后双击安装程序，安装界面如图 1.3 所示。


图 1.3　Anaconda 安装界面

（3）单击 Next 按钮，在如图 1.4 所示的 License Agreement 界面选择 I Agree。

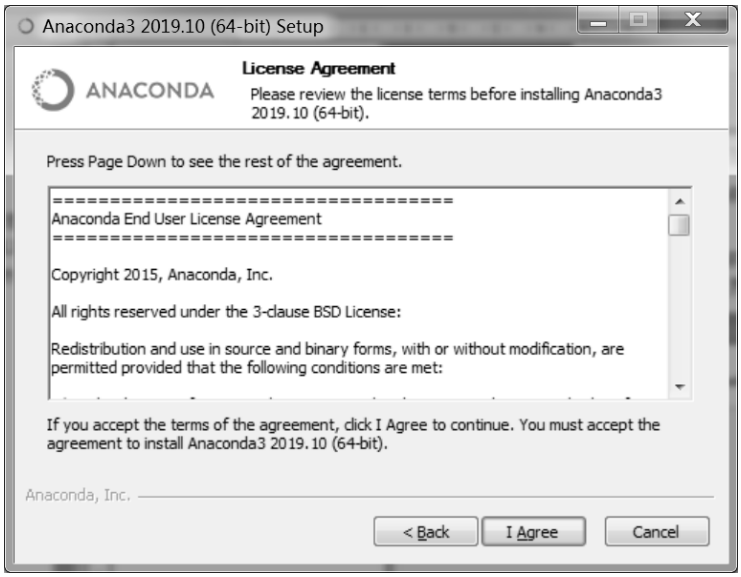


图 1.4　Anaconda License Agreement 界面

（4）在如图 1.5 所示的 Select Installation Type 界面选择使用此应用的用户。

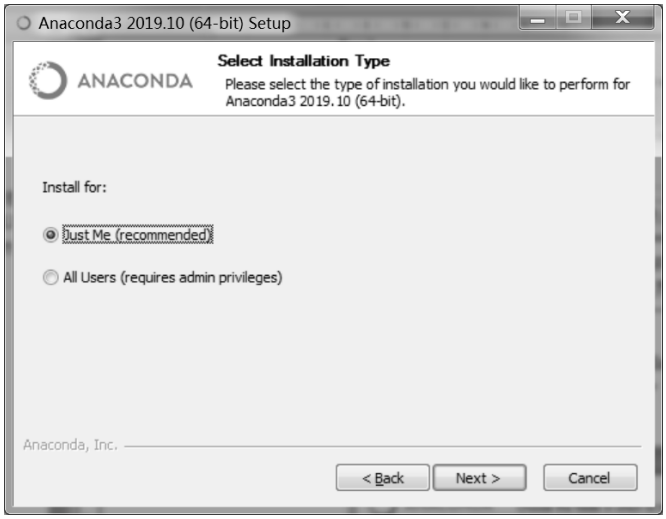


图 1.5　Anaconda Select Installation Type 界面

（5）单击 Next 按钮，进入如图 1.6 所示的 Choose Install Location 界面，在这里为 Anaconda 选择安装路径。

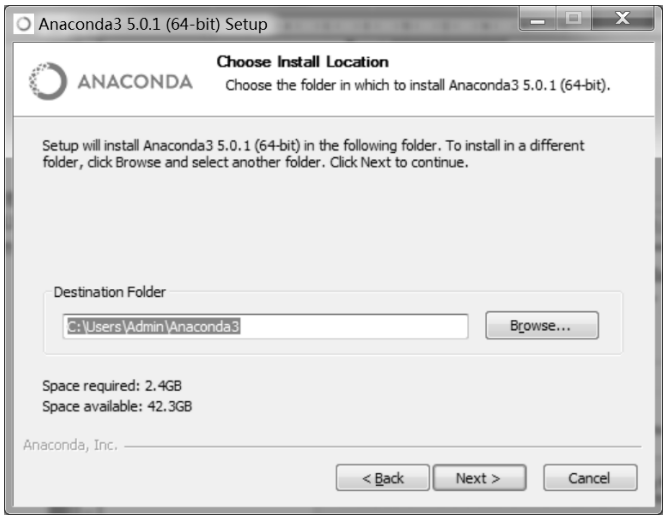


图 1.6　Anaconda Choose Install Location 界面

（6）单击 Next 按钮，进入 Advanced Installation Options 界面，如图 1.7 所示。

如果是首次安装 Anaconda，建议选中 Add Anaconda to my PATH environment variable 复选框，将 Anaconda 加入 PATH 环境变量，这样当使用 Python、IPython、conda 和其他 Anaconda 应用时，程序在系统中的路径可以被找到，从而顺利启动。

**说明：**

☑ 选中复选框 Register Anaconda as my default Python 3.7，则其他应用程序会自动检

测并将 Python3.7 作为 Anaconda 的首要版本。

☑ 如果安装了多个版本的 Anaconda，选择此项会引起版本冲突，这种情况下不建议选择，可以通过 Windows“开始”菜单| Anoconda3（64-bit）顺利启动程序。

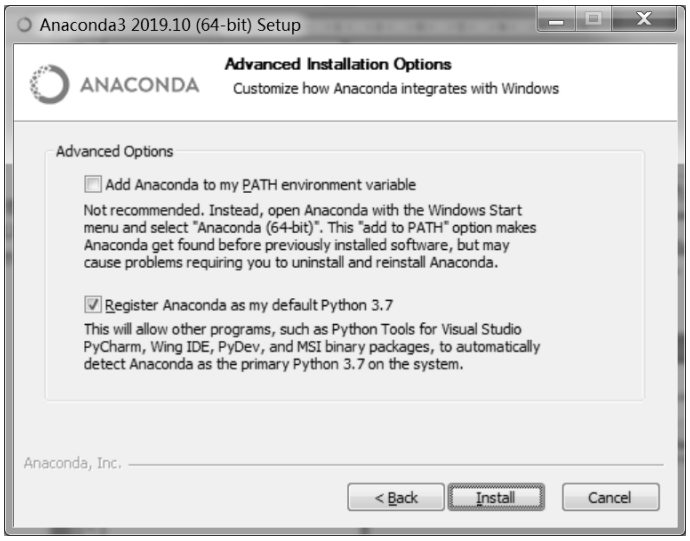


图 1.7　Anaconda Advanced Installation Options 界面

（7）单击 Install 按钮，进入 Installing 界面，可以单击 Show details 按钮查看安装详情，如图 1.8 所示。

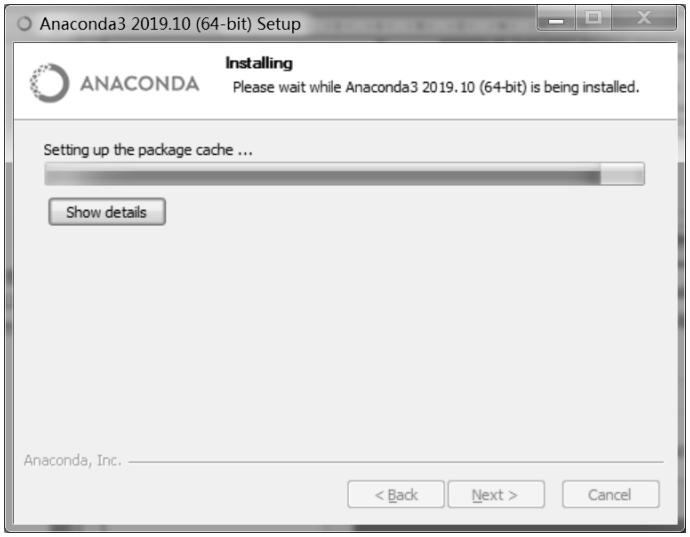


图 1.8　Anaconda Installing 界面

（8）单击 Next 按钮，显示如图 1.9 所示的 Anaconda+JetBrains 说明界面。

（9）单击 Next 按钮，进入 Thanks for installing Anaconda3 界面，如图 1.10 所示，单击 Finish 按钮，完成安装。


图 1.9　Anaconda+JetBrains 说明界面

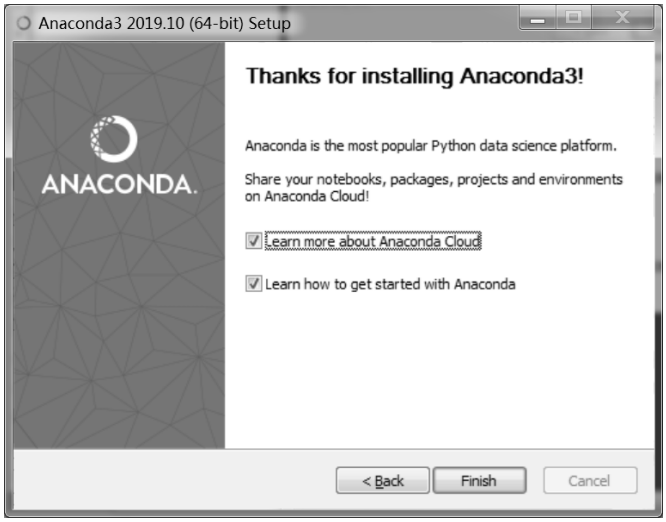


图 1.10　Thanks for installing Anaconda3 界面

### 1.3.4　Anaconda 组件

在 Windows “开始” 菜单中选择 Anaconda3 ( 64-bit )，可以显示 Anaconda 的主要组件。

1. Anaconda Cloud

Anaconda Cloud 是一种连续分析包管理服务，云服务使得各种服务便于查询、访问、储存和共享。对于使用者，在 Anaconda Cloud 中不需要注册账号就可以搜索、下载和安装公共扩展包。对于开发者，Anaconda Cloud 可以为软件开发、发布和维护提供方便。

2. Anaconda Navigator

Anaconda Navigator 是基于 Web 的图形用户界面交互式计算环境，用于启动应用、管

理扩展包和管理应用环境。可以设置 Anaconda Navigator 在 Anaconda Clouds 中或在本地搜索扩展包。

3. Anaconda Prompt

Anaconda Prompt 是 Anaconda 的命令行界面，在这里可以通过输入命令管理应用环境和扩展包。

4. IPython

IPython 是一个改进的 Python 交互式运行环境，为交互式计算提供了丰富的架构。支持变量自动补全、自动缩进、交互式的数据可视化工具、Jupyter 内核、可嵌入的解释器和高性能并行计算工具等功能。

5. Jupyter Notebook

Jupyter Notebook 是一个交互式笔记本，支持运行 40 多种编程语言，可以用来编写漂亮的交互式文档，便于创建和共享程序文档，支持实时编码、数学方程和可视化，可以用于数据清理和转换、数值模拟、统计建模、机器学习、展示数据分析过程等任务。

6. Jupyter QTConsole

Jupyter QTConsole 是一个轻量级可执行 IPython 的仿终端图形界面程序，可以直接显示代码生成的图形、实现多行代码输入执行，并提供内联数据、图形提示和图形显示等功能和函数。

7. Spyder

Spyder 是使用 Python 语言、跨平台的、科学运算集成开发环境，集成在 Anaconda 中，功能包括代码编辑、交互测试、程序调试和自检功能等。

8. Reset Spyder Settings

恢复 Spyder 的默认设置。

### 1.3.5 Anaconda Navigator 环境配置

Anaconda Navigator 是管理环境和扩展包的图形用户界面，在这里无须掌握相关命令就可以通过菜单就可以完成搜索、安装、运行和升级扩展包等功能。

在 Windows“开始”菜单中选择 Anaconda3（64-bit）| Anaconda Navigator，启动 Anaconda Navigator，如图 1.11 所示。

1. 创建一个新的环境

Anaconda 安装好后，Anaconda Navigator 中可以看到默认的基本环境 base（root），也称为“根环境”。用户也可以为特定的任务定制一个新的环境。单击如图 1.11 中的 Environment 标签，在如图 1.12 所示的 Environment 窗口中单击 Create 按钮。

在图 1.12 的对话框中，输入 Environment name，如 NLP3.7，在 Pakage 中选择 Python，在 Python version 中选择 3.7，单击 Create 按钮。

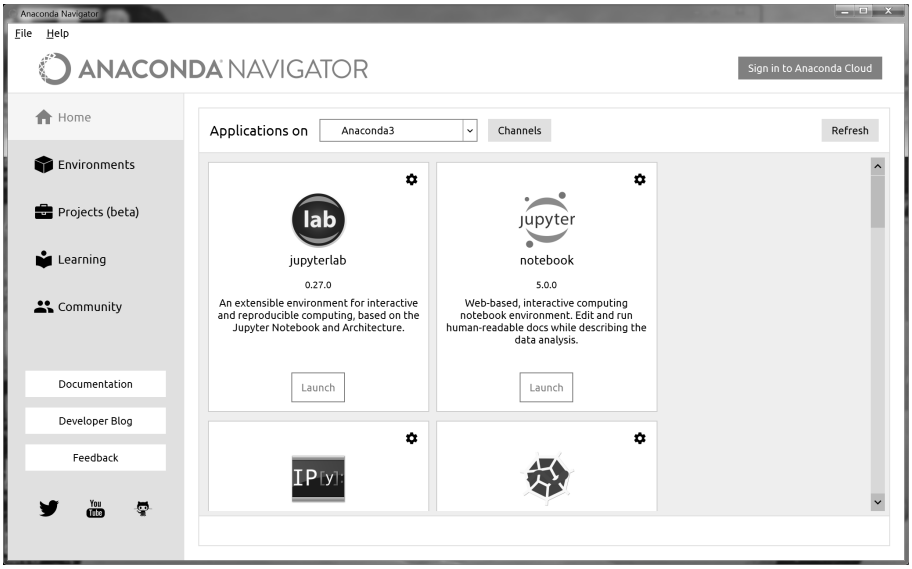


图 1.11　Anaconda Navigator 的 Home 界面

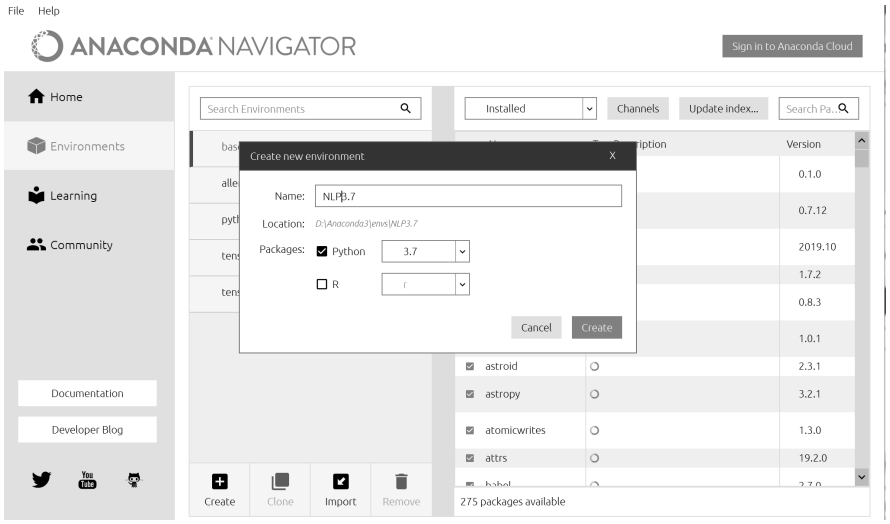


图 1.12　Anaconda Navigator 的 Environment 界面

这样，Anaconda Navigator 就创建了一个新的环境并激活了这个环境 NLP3.7，除非更改当前环境，之后所有的操作就在此环境中进行。

2. 扩展包管理

在 Environment 窗口中，右边显示的就是当前环境中的扩展包，可以在窗口上方选择 Installed 查看已经安装的扩展包，选择 Not Installed 查看未安装的扩展包，选择 Upgradable 查看可升级的扩展包，或者选择 All 查看全部扩展包。

例如，选择 NLP3.7 环境和 Installed 选项，可以查看当前 NLP3.7 环境中已经安装的扩展包及其版本。

3. 安装新的扩展包

有些扩展包需要用户自行安装，这时候选择 All 显示所有扩展包，在右边的搜索框中输入要安装的扩展包名称，如 Gensim，下面的窗口中即出现可以安装的新扩展包，如图 1.13 所示。

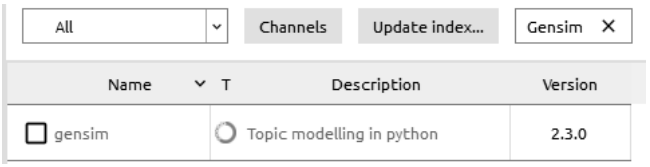


图 1.13　Anaconda Navigator 安装新的扩展包

选中 gensim 前面的复选框，框中出现一个绿色的下载箭头，单击下方的 Apply 按钮，弹出如图 1.14 所示对话框，选择 OK，即开始安装。

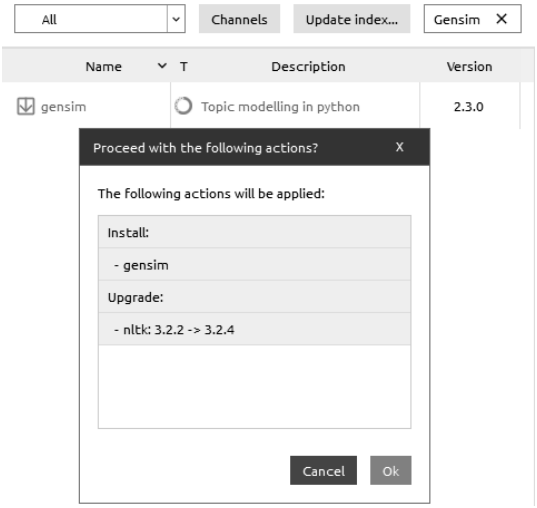


图 1.14　开始安装扩展包

## 1.4　在 Spyder 中运行 Python 程序

Spyder 是 Python 的科学计算集成开发环境，支持 Python 解释器和 IPython 解释器，支持常用的 Python 类库，例如线性代数包 NumPy、信号和图像处理包 SciPy、交互式 2D/3D 绘图包 matplotlib 等。

在 Windows“开始”菜单中选择 Anaconda3（64-bit）| Spyder，启动 Spyder，打开的界面如图 1.15 所示。

1. Spyder 编辑器（Editer）

图 1.15 中间的窗口是 Spyder 编辑器，Spyder 的编辑器是一个多语言编辑环境，支持语法彩色标记、实时代码分析、高级代码分析、代码自检功能和类浏览器等功能。

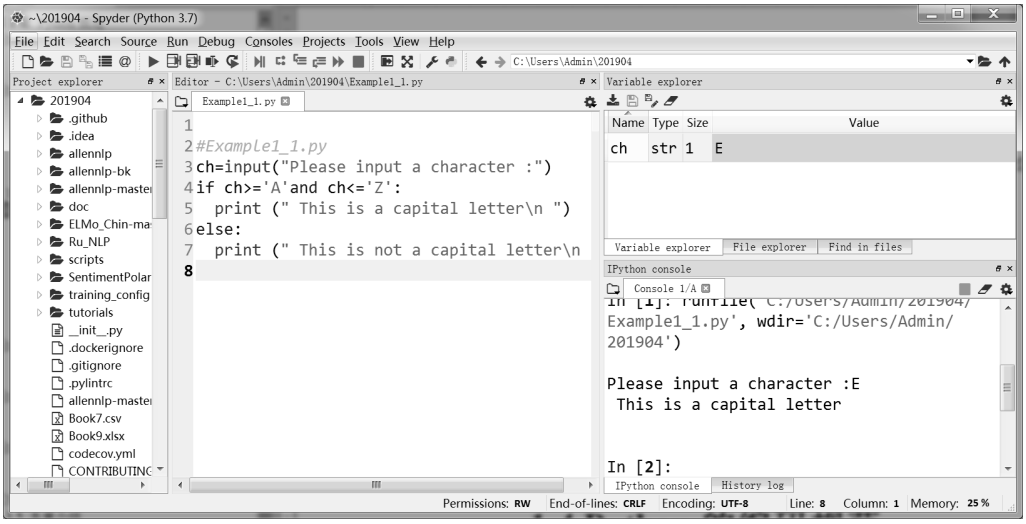


图 1.15　Spyder 界面

2. Spyder“运行”按钮

要运行上述编辑的程序代码，可单击工具栏中的“运行”按钮，运行刚才输入的代码，如图 1.16 所示。

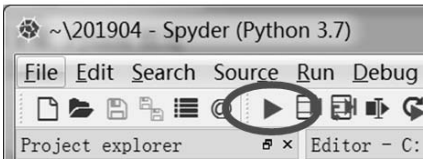


图 1.16　Spyder 工具栏中的“运行”按钮

3. Spyder 控制台（console）

Spyder 控制台中，可以在浏览和显示程序运行过程和结果。在控制台中输入的各个命令在各自独立的进程中执行时，自动在 Spyder 窗口右下角的 IPython console 中显示运行过程和结果，如图 1.17 所示。

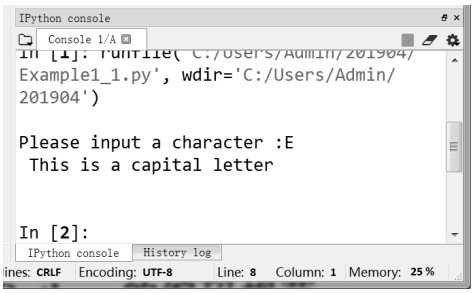


图 1.17　IPython console

如果不小心关闭了 IPython console 窗口，可以选择菜单 Consoles | Open an IPython console 打开新的 IPython 控制台。

### 1.4.1 第一个 Python 程序

**【实例 1.1】由键盘输入的一个字符，判断是否是大写字母。**

**说明：**

- ☑ 可以通过 input()函数由键盘输入一个字符。
- ☑ 可以通过此字符的 ASCII 码判断这个字符是否为大写字母。
- ☑ 判断的结果可以通过 print()函数输出。
- ☑ print()函数中的"\n"表示换行。

```
#Example1_1.py

ch=input("Please input a character :")
if ch>='A'and ch<='Z':
   print (" This is a capital letter\n ")
else:
   print (" This is not a capital letter\n ")
```

**【操作步骤】**

（1）在 Spyder 编辑器中输入上述代码，注意缩进的代码行；

（2）检查无误后，选择菜单 File | Save as 将上述代码以 Example1_1.py 为文件名保存；

（3）单击 Spyder 窗口上方的“运行”按钮；

（4）由于有 input()函数，此时 Spyder 控制台中显示 Please input a character：并等待输入。此时输入一个字符，如字符“E”，按回车键；

（5）程序根据输入的字符判断，并在 Spyder 控制台中输出结果 This is a capital letter，如图 1-15 运行结果二所示。

**【运行结果一】**

```
Please input a character : e↙
This is not a capital letter
```

**【运行结果二】**

```
Please input a character : E↙
This is a capital letter
```

**Python 的文件类型说明：**

- ☑ 源代码文件.py

  python 源代码文件以.py 为扩展名，由 python 程序解释，不需要编译。

- ☑ 字节代码文件.pyc

  Python 源代码文件经过编译后生成扩展名为.pyc 的字节代码文件。在 Spyder 集成开发环境下，运行过的.py 文件在同一文件夹下自动生成一个同名的.pyc 字节代码文件。

### 1.4.2 Python 语法基础

1. 缩进分层与冒号

Python 程序代码通过缩进结构来分层次，使得代码结构清晰、易于阅读。缩进结构一般用在条件语句、循环语句、函数定义等控制语句的语句块中。一般代码行末尾的冒号“:”标志着下一行应该有代码缩进，缩进的空格个数没有明确规定，一般每次缩进 4 个空格，同一语句块的代码缩进空格个数应该相同。

例如，实例 1.1 中的“if ch>='A'and ch<='Z':”这一行末尾的冒号标志着条件语句的条件后面需要代码缩进，“else:”这一行末尾的冒号标志着后面需要代码缩进。

```
#Example1_1.py
ch=input("Please input a character :")
if ch>='A'and ch<='Z':                          #冒号“:”下一行开始缩进
   print (" This is a capital letter\n ")       #缩进 4 个空格，此语句块有 1 行
else:                                           #else 与 if 对齐，冒号“:”
                                                 下一行开始缩进
   print (" This is not a capital letter\n ") #缩进 4 个空格，此语句块有 1 行
```

**说明：**

- ☑ 可以通过 input()函数由键盘输入一个字符。
- ☑ 可以通过此字符的 ASCII 码判断这个字符是否为大写字母。
- ☑ 判断的结果可以通过 print()函数输出。

2. 代码注释

规范的代码注释是代码阅读和代码维护的重要前提。Python 中的代码注释有两种形式：

● 单行注释：以字符“#”开始，“#”之后同一行的所有内容都被看作是注释，不作为代码执行。

● 多行注释：如果需要注释的内容有多行，可以采用三个英文状态下的单引号或者双引号，扩在注释内容的两端。

在 Python 集成开发环境中，注释部分会自动改变颜色方便用户查看。例如，在 Spyder 中注释部分会默认变为灰色。

3. 断行

Python 代码通常一条语句单独写在一行中。如果一条语句过长，可以使用“\”符号表示下一行是同一条语句。例如下面代码与实例 1.1 效果相同：

```
ch=input("Please input a character :")
if ch>='A'and ch<='Z':
   print \
       (" This is a capital letter\n ")
else:
   print (" This is not a capital letter\n)
```

### 说明：

☑ 使用“\”表示下一行是同一条语句时，“\”应该在行的末尾，此行后面不能有任何内容。

## 1.4.3 输入与输出

输入与输出是用户与 Python 程序进行交互的主要途径。通过输入语句，程序能获取运行过程中所需要的原始数据；通过输出语句，用户能够了解程序运行的中间结果和最终结果。

1. 输入函数 input( )

函数 input( )用于向用户生成一个提示，然后获取用户输入的内容，Python3.x 中 input( )函数将用户输入的内容放入字符串中，因此用户输入任何内容，input()函数总是返回一个字符串。input( )基本形式如下：

```
input([输入提示信息])
```

其中[输入提示信息]为可选信息，在运行到此条语句时显示输入提示信息，提示用户输入，用户输入后按回车键，输入的内容以字符串的形式输入到程序中。

2. 输出函数 print( )

Python 通过输出函数 print( )显示输出程序运行的中间结果和最终结果，基本形式如下：

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

### 参数说明：

☑ objects：表示输出的对象，*号表示一次可以输出多个对象，各个对象在 print 函数中用“,”逗号分隔。

☑ sep：表示输出显示时，各个对象之间的间隔，默认间隔是一个空格。

☑ end：表示输出显示时，用来结尾的符号，默认值是换行符\n，可以换成其他字符串。

☑ file：表示输入信息写入的文件对象。

☑ flush：如果设置此参数值为 True，输出信息缓存流会被强制刷新，默认值为 False。

**【实例 1.2】由键盘输入 3 个小写字母，转换为大写字母输出。**

### 说明：

☑ 可以通过 input( )函数由键盘输入字母。

☑ 输入的小写字母分别保存在 3 个变量中。

☑ 通过 upper( )函数将字母转换为大写字母。

☑ 转换的结果可以通过 print( )函数输出。

```
#Example1_2.py

ch1=input("Please input the first lowercase letter:")
ch2=input("Please input the second lowercase letter:")
ch3=input("Please input the third lowercase letters:")
ch1=ch1.upper()             #ch1 的类型是字符串，可以调用.upper( )函数转换为
                             大写
ch2=ch2.upper()
ch3=ch3.upper()
print (ch1,ch2,ch3)         #用默认参数输出
print (ch1,ch2,ch3,sep=',') #sep 分隔符号修改为逗号
print (ch1,ch2,ch3,end=';') #end 结束符号修改为分号
```

【运行结果一】

```
Please input the first lowercase letter:y↙
Please input the second lowercase letter:o↙
Please input the third lowercase letters:u↙
Y O U
Y,O,U
Y O U;
```

三个 print 函数分别以空格作为分隔符、逗号作为分隔符、分号作为结束符号显示输出结果。

【运行结果二】

```
Please input the first lowercase letter:Y↙
Please input the second lowercase letter:o↙
Please input the third lowercase letters:u↙
Y O U
Y,O,U
Y O U;
```

当输入的字母中有大写字母，upper( )函数的返回结果仍然为大写字母。

## 1.5 本 章 小 结

计算机发展经历了电子管计算机、晶体管计算机、集成电路计算机、大规模集成电路计算机、第五代新型计算机。

计算机遵循冯·诺依曼体系结构，由控制器、运算器、存储器、输入设备、输出设备五部分组成。

程序设计语言发展经历了低级语言、高级语言、面向对象语言三个阶段。

Python 是一种高层次的结合了解释型、交互型和面向对象的脚本语言，语言风格简洁，集成了模块、异常处理和类的概念，提供丰富的标准库、扩展库、API 和工具，开发的应用具有很好的可移植性和兼容性。

Python 语言目前已经广泛应用于操作系统管理、科学计算、自然语言处理、Web 编程、图形用户界面开发、多媒体应用和人工智能等领域。

本章以 Windows 10 和 Python 3.7 为基础，介绍了 Python 的安装和开发环境设置，在不同操作系统平台上的安装和配置过程基本一致。本章介绍的集成开发环境 Anaconda 可以提供 Python 程序开发环境的各种应用程序和配置功能，能够有效帮助开发者提高效率和减少失误，本章还介绍在集成开发环境 Spyder 中编辑和运行 Python 程序的过程。

最后，本章通过两个 Python 程序实例介绍了 Python 的基础语法和基本输入输出函数。

## 1.6 习　　题

### 一、单项选择题

1. 以下关于 Python 语言中注释说法错误的是（　　）。

   A. 单行注释以“#”符号开头

   B. 多行注释可以用三个单引号或双引号，包括在注释部分两端

   C. 单行注释可以放在正常语句同一行的后面

   D. 注释语句也会被执行

2. Python3.7 环境下，执行下列语句后的显示结果是（　　）。

```
s = "Python"
print(s.upper( ), end=';')
```

   A. P　Y　T　H　O　N　　B. P,Y,T,H,O,N

   C. PYTHON;　　D. 程序出错

3. 以下语句能够输出如下图形的是（　　）。

```
*
**
***
```

   A. print ('*','**','***')　　B. print ('*','**','***',end ='\n')

   C. print ('*','**','***',sep='\n')　　D. 以上都不对

4. 以下语句的输出结果是（　　）。

```
ch='q'
if ch>='A'and ch<='Z':
     print (" This is a capital letter\n ")
else:
     print (" This is not a capital letter\n ")
```

   A. This is a capital letter　　B. This is not a capital letter

   C. This is not a capital letter:q　　D. 程序出错

5. 在 Spyder 集成开发环境下，以下（　　）是断行错误的提示信息。

   A. SyntaxError: invalid syntax

   B. IndentationError: expected an indented block

   C. IndentationError: unexpected indent

   D. SyntaxError: unexpected character after line continuation character

6. Python 的源代码文件的扩展名是（　　）。

   A. pya　　B. pyc　　C. py　　D.pyo

### 二、编程题

1. 参照实例 1.1 编写程序，由键盘输入一个字符，判断是否是小写字母并输出判断

结果。

2. 参照实例 1.2 和单项选择题第 2 题，编写程序，由键盘输入一个单词，转换为小写字母输出。提示：可以通过 lower( )函数将字母转换为小写字母。



3. 编写程序，输出如下图形。

```
  *
 ***
*****
 ***
  *
```