省级一流课程、课程思政示范课配套教材
新型工业化·新计算·计算机应用与技术类系列

COMPUTER APPLICATION

Python 程序设计

与人工智能项目教程

梁婷婷　梁肇敏　蒋正锋/主编
刘昊　梁爽　余悦/副主编

扫一扫书中二维码
观看本书配套资源

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书旨在为读者提供全面的 Python 程序设计知识，并紧密结合人工智能领域的实际应用，帮助读者掌握 Python 核心技能及其在人工智能项目中的实践方法。本书围绕 Python 程序设计与人工智能项目开发展开，共 14 章，内容涵盖从基本语法到高级应用的完整知识体系。

第 1 章介绍了 Python 基础知识，包括 Python 的特点、应用领域、发展历史、开发环境配置、基本语法及输入/输出操作等内容。第 2 章介绍了数字类型与运算，包括数字类型、运算符与表达式、数学运算函数、math 模块等内容。第 3 章和第 4 章深入剖析了流程控制与函数编程，通过丰富的实例，帮助读者理解 Python 编程的核心逻辑。第 5 ~ 8 章详细讲解了高级数据类型（字符串、元组、列表、集合与字典）及其操作等内容，结合实例提升读者的数据处理能力。第 9 ~ 12 章详细讲解了文件操作、面向对象程序设计、Matplotlib 数据可视化，以及 Python 生态应用等内容，这是开发人工智能项目十分重要的内容。第 13 章和第 14 章提供了综合实战项目，分别介绍了智慧校园中的用户画像系统构建和基于知识图谱的学习资源推荐系统构建。本书每章都设置了项目实训，引导读者将知识迁移至实际场景中，提升解决复杂问题的能力。

本书配套提供了丰富的在线学习资源，包括微课视频、项目案例、教学课件、习题库及详解、源代码等，支持教师开展线上线下混合式教学，同时便于读者的自主学习。

本书适合作为高等院校各专业的程序设计课程教材，也可供开发者和编程爱好者自学参考。

图书在版编目（CIP）数据

Python 程序设计与人工智能项目教程 / 梁婷婷, 梁肇敏, 蒋正锋主编. -- 北京 : 电子工业出版社, 2025.
7. -- ISBN 978-7-121-50667-3

Ⅰ. TP312.8；TP18

中国国家版本馆 CIP 数据核字第 2025U8V855 号

责任编辑：刘　瑀
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　印张：16.25　字数：416 千字
版　　次：2025 年 7 月第 1 版
印　　次：2025 年 7 月第 1 次印刷
定　　价：59.90 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：liuy01@phei.com.cn。

前　言

在当下数字化与智能化交织的新时代，Python 凭借简洁优雅的语法、丰富强大的生态库和活跃的开发者社区，已成为数据科学与人工智能领域的核心工具和探索科技前沿、实现创新想法的利器。掌握 Python 编程及其在人工智能领域的应用，正逐渐成为科技人才的核心竞争力。

为了响应这一趋势，本书聚焦 Python 程序设计与人工智能项目的系统教学，面向高校学生、Python 编程爱好者，以及人工智能领域初学者，提供从基础到实践的全栈式学习路径。全书内容体系完整，涵盖 Python 基础知识、数字类型与运算、流程控制、函数、高级数据类型及常见操作、字符串、元组和列表、集合与字典、文件操作、面向对象程序设计、Matplotlib 数据可视化，以及 Python 生态应用等核心内容，并通过阶梯式实例设计，帮助读者实现“学用结合”。

本书具有五大特色：

1. 思政教育与技术培养有机融合

创新性地将社会主义核心价值观、民族自豪感、工匠精神等思政元素自然嵌入人工智能项目实例，实现“价值塑造-能力培养-知识传授”三位一体的教学目标。

2. 前沿技术驱动教学内容

紧密跟踪人工智能领域最新发展，系统讲解 Python 在大数据分析、机器学习、智能感知等领域的应用，涵盖数据集构建、模型设计、参数调优、分类预测等关键环节，旨在培养读者解决真实场景问题的能力。

3. 多层次实践体系

构建“基础示例-章节项目-综合实训”三级实践体系，200+精选实例由浅入深，辅以完整代码解析，确保知识可迁移、可落地。

4. 立体化教学资源

为了使读者高效地学习，本书配套提供了丰富的在线学习资源，包括微课视频、项目案例、教学课件、习题库及详解、源代码等，全面支持线上线下混合式教学，同时为广大读者的自主学习提供了极大的便利。

5. 经过验证的教学成效

本书所依托的 Python 程序设计课程自开课以来，已荣获多项省级及以上教育奖项，包括被评为省级线上线下混合式一流本科课程、省级课程思政示范课程等。本书中的部分实例获全国课程思政教学大赛一等奖等 14 项省级以上奖项，教学效果获广泛认可。

本书由南宁学院牵头组织编写，编写分工为：第 1、2、4 章由梁婷婷、刘昊编写；第 3、9 章由梁婷婷编写；第 5、6、7 章由梁婷婷、梁爽编写；第 8、11、14 章由梁肇敏、梁婷婷编写；第 10、12、13 章由蒋正锋（广西民族师范学院）、梁婷婷编写。本书配套提供的在线学习资源由梁婷婷、余悦、梁肇敏开发与制作。另外，本书得到全国高等院校计算机基础教育研究会专项课题（课题编号：2025-AFCEC-121）、广西教育科学“十四五”

规划专项重点课题（课题编号：2023ZJY510）、南宁学院“十四五”发展规划教材资助出版课题（课题编号：2023XJJC02）的支持，在此表示感谢。

我们期待本书能成为读者探索 Python 与人工智能世界的可靠指南，实现个人价值与社会贡献的双重飞跃。

因编者学识水平和能力有限，本书中难免存在疏漏、不妥甚至错误之处，敬请广大读者批评指正。

编　者

2025 年 6 月

目　录

第 1 章　Python 基础知识

1.1　Python 概述

Python 是一种高级、解释型、通用的程序设计语言。其设计理念强调代码的可读性和简洁性，这使得 Python 成为初学者和专业开发者的理想选择。Python 的语法采用缩进式的层次结构，这使得其代码结构清晰，易于维护。此外，Python 支持多种编程范式，包括面向对象程序设计、面向过程程序设计，这为开发者提供了灵活的编程方式。

Python 拥有丰富的标准库和第三方库，这些库涵盖了从文件处理、网络通信到数据分析、机器学习等各个领域，极大地扩展了 Python 的应用范围，使 Python 能够胜任 Web 开发、数据科学、人工智能（Artificial Intelligence，AI）、自动化测试等多种任务。

Python 的跨平台特性也是其受欢迎的原因之一。Python 可以在多种操作系统上运行，包括 Windows、macOS 和 Linux，且使用 Python 编写的代码在不同平台之间的移植性极强。此外，Python 的解释器是开源的，开发者可以自由地使用、修改和分发代码，这进一步推动了 Python 社区的繁荣发展。

Python 社区活跃且充满活力，全球范围内的开发者不断贡献新的库、工具和框架。此外，Python 社区还提供丰富的文档和教程，以帮助初学者快速入门，并为专业开发者提供深入的技术支持。

总之，无论是学术研究、工业应用还是个人项目，Python 都能提供高效、灵活的解决方案，是值得广大程序员掌握的语言。

1.1.1　Python 的现状

自 2004 年以来，Python 的使用率持续呈线性增长，展现出强劲的发展势头。2019 年 9 月，IEEE Spectrum 发布的研究报告证实，Python 的影响力和应用范围不断扩大。这一趋势在 2020 年达到新的高度，Python 在 TIOBE 编程语言排行榜中首次超过 Java，跃居第 2 位，仅次于 C 语言。这一变化充分体现了 Python 在数据科学、人工智能、Web 开发等领域的广泛应用，以及 Python 简洁易学的特性对开发者的强大吸引力。

Python 的崛起并未止步于此。2021 年 10 月，Python 在 TIOBE 编程语言排行榜中首次超越 C 语言，登顶。这是 Python 自 2001 年以来在 TIOBE 编程语言排行榜中首次成为全球最受欢迎的程序设计语言，标志着 Python 在开发者社区中处于主导地位。自此之后，Python 的排名一直稳居前列，持续引领程序设计语言的发展潮流。

1.1.2　Python 的应用领域

Python 作为一种功能强大且易于学习的程序设计语言，近年来在多个领域展现了良

好的应用前景。在数据分析领域，Python 凭借 Pandas、NumPy 等库，以及 Matplotlib、Seaborn 等可视化工具，成为数据科学家处理、分析和可视化数据的首选。在 Web 开发领域，Django 和 Flask 等框架为开发者提供了高效构建稳定、可扩展应用的能力。在人工智能和机器学习领域，Python 凭借简洁的语法和丰富的库（TensorFlow、PyTorch 等）支持成为模型训练与部署的核心工具。此外，Python 广泛应用于科学计算、自动化测试、网络爬虫，以及游戏开发等领域。随着技术发展，Python 的独特价值将在更多领域展现。

1.1.3 Python 的发展历史

Python 由 Guido van Rossum 于 1989 年创立，以简洁、优雅的设计理念迅速受到了广泛关注。自 1991 年发布首个版本后，Python 的版本不断迭代更新，1994 年的 Python 1.0 的出现展示了稳定的框架，2000 年的 Python 2.0 引入了列表推导式等关键特性，而 2008 年的 Python 3.0 的出现虽带来了不兼容的挑战，却为未来发展铺平了道路。

1.1.4 Python 快速发展带来的启示

Python 的快速发展不仅是程序设计语言史上的奇迹，更是对学习、创新与适应性的深刻启示。第一，持续学习是应对变化的关键。Python 凭借简洁的语法和丰富的库支持成为初学者与专业人士的共同选择，这展现了在技术领域不断更新知识的重要性。第二，开放与共享是创新的驱动力。Python 的成功得益于其开源社区的繁荣，开发者构建了一个充满活力的生态系统，这体现了开放与共享对技术进步的推动作用。第三，跨领域融合是未来趋势。Python 在数据分析、人工智能、Web 开发等领域的广泛应用，反映了技术融合与学科交叉的必然性，这表明培养跨学科人才成为适应未来需求的关键。

总之，Python 的快速发展不仅是技术进步的象征，更是时代精神的体现。只有持续学习、开放共享、跨领域融合，以及保持适应性，才能更好地迎接未来的机遇与挑战。

1.1.5 Python 官方文档入口

当前，Python 的开发和维护由 Python 软件基金会（Python Software Foundation）负责。可以从 Python 官网获取 Python 的相关信息。在 Python 官网首页，先单击“Documentation”按钮再单击“Python Docs”按钮进入文档列表，在语言菜单中选择中文，即可查看中文版的 Python 入门教程、标准库参考、语言参考和常见问题等文档。此外，若要关注中文版的 Python 的相关信息，也可以访问 Python 中国社区网。

1.2 Python 开发环境配置

1.2.1 解释器的安装和使用

在使用 Python 之前，需要在计算机的操作系统上安装 Python。下面以 Windows 10 为例讲解安装 Python 的步骤。

访问 Python 官网下载 Python 安装文件。在图 1-1 中，先将鼠标指针停留到菜单栏中

的“Downloads”按钮上，会自动弹出下拉菜单，然后将鼠标指针停留到“Windows”选项上，右侧会切换为适合本地计算机操作系统安装的 Python 版本，单击“Python 3.13.1”按钮，开始下载 64 位版本的 Python 安装文件。

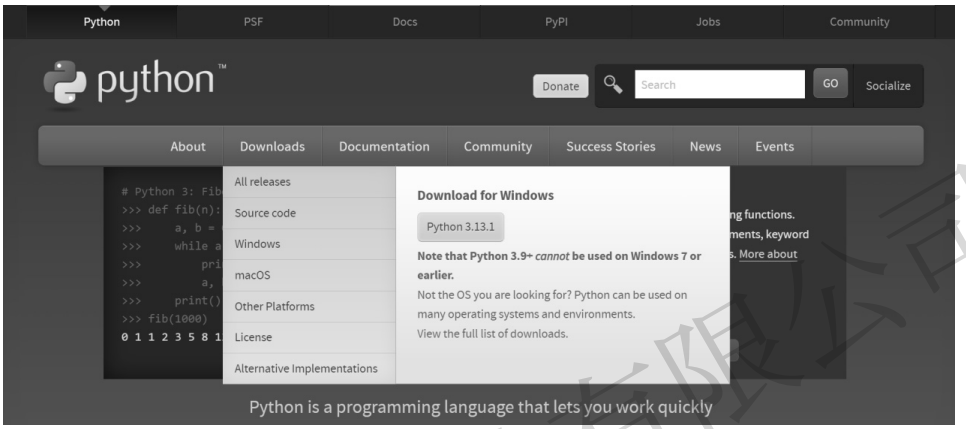

图 1-1　下载 Python 安装文件

下载 Python 安装文件后，以管理员身份运行，显示如图 1-2 所示的“Python 3.13.1（64 bit）Setup”窗口。

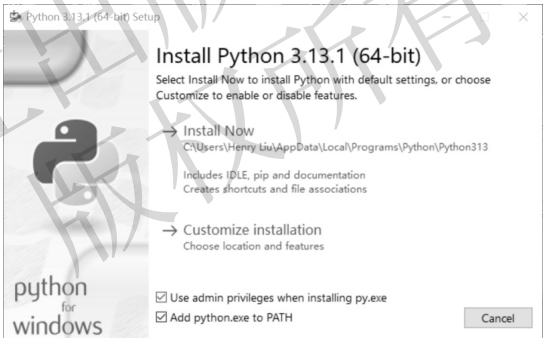

图 1-2　“Python 3.13.1（64 bit）Setup”窗口

默认勾选“Use admin privileges when installing py.exe”复选框，如果想从 PowerShell 中直接执行 Python 命令，那么需要勾选“Add python.exe to PATH”复选框，将 Python 安装目录添加到环境变量 PATH 中。如果想修改 Python 安装目录或自定义安装功能，那么选择“Customize installation”选项，否则选择“Install Now”选项，此时会将 Python 直接安装到“C:\Users\Henry Liu\AppData\Local\Programs\Python\Python313”目录中。成功安装 Python 后，下面通过编程来测试 Python 的开发环境。编程主要有两种：交互式编程与文件式编程。

实例 1-1　交互式编程

按快捷键 Win+R 打开“运行”对话框，输入“cmd”，单击“确定”按钮，即可进入控制台。输入“python”，提示符变成“>>>”，证明已成功进入交互模式，如图 1-3 所示。

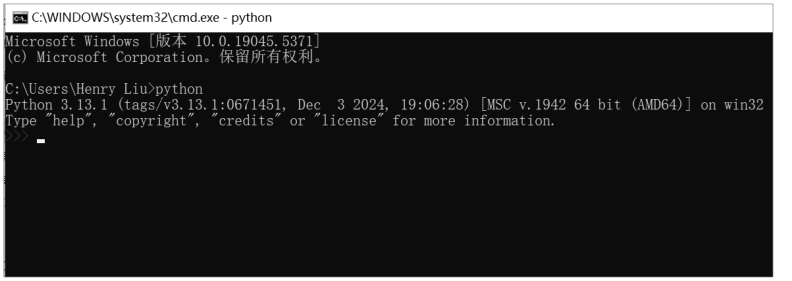

图 1-3　进入交互模式

在提示符“>>>”后输入“print ("This is my first lesson.")”，按 Enter 键，即可输出该字符串。

实例 1-2　　文件式编程

按快捷键 Win+R 打开“运行”对话框，输入“notepad”，单击“确定”按钮，输入“print("This is my first lesson.")”，选择“文件”→“另存为”命令，打开如图 1-4 所示的“另存为”对话框，切换到“C:\Users\Henry Liu”目录中，将“文件名”改为“FirstLesson.py”，并设置“编码”为“UTF-8”。

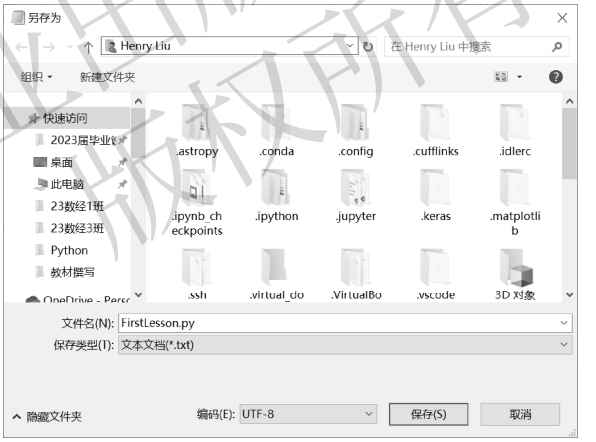

图 1-4　“另存为”对话框

按快捷键 Win+R 打开“运行”对话框，输入“cmd”，单击“确定”按钮，输入“python FirstLesson.py”，按 Enter 键，即可输出与交互式编程同样的结果。

使用文本编辑工具只能编写代码，不能调试代码。要想既能编写代码又能调试代码，可以使用 IDE（Python 集成开发环境）工具。目前用来编写代码的主流 IDE 工具主要有 PyCharm 与 Jupyter Notebook。

1.2.2　IDE 工具——PyCharm

PyCharm 是一款由 JetBrains 开发的 IDE 工具，专为 Python 编程设计。它提供智能代码补全、语法高亮、代码分析与调试、版本控制集成等功能，支持多种框架（Django、Flask 等）和科学计算工具（NumPy、Matplotlib 等）。PyCharm 分为社区版（免费）和专业版

（付费），适合从初学者到专业开发者十分广泛的用户群体，是开发 Python 的高效工具。

实例 1-3　使用 PyCharm 的基本流程

通过浏览器访问 PyCharm 官网，下载与安装免费的 PyCharm Community Edition。首次启动 PyCharm 后，先单击左侧的“自定义”按钮，设置 IDE 语言和区域，主题由个人使用习惯或喜好而定。返回主窗口后，先单击左侧的“项目”，然后单击“新建项目”按钮，若已经安装 Python，则解释器会自动识别检测出 Python 的版本，设置好项目的目录位置，单击“创建”按钮，即创建项目的虚拟环境。进入项目的虚拟环境后，选择“文件”→“新建”命令，设置“文件名”为“myfirstlesson”，单击“确定”按钮，在文件编辑区中输入“print("This is my first lesson")”，单击▷按钮，即可输出结果。

1.2.3　IDE 工具——Jupyter Notebook

Jupyter Notebook 是基于网页的交互式 IDE 工具，专为数据科学和机器学习设计。Jupyter Notebook 支持多种程序设计语言（Python、R、Julia 等），允许用户创建和共享包含代码、文本和公式等的笔记本。此外，Jupyter Notebook 提供灵活的界面，支持多文档布局、终端集成和丰富的插件扩展，是数据分析和科学计算的强大工具。

实例 1-4　使用 Jupyter Notebook 的基本流程

通过浏览器访问 Anaconda 官网，下载与安装免费的版本。安装成功后，即可看到“开始”菜单中出现“Anaconda”文件夹。打开该文件夹后，即可看到 Jupyter Notebook 快捷方式图标。启动 Jupyter Notebook 后，会自动打开默认浏览器，进入 Jupyter Notebook 界面，如图 1-5 所示。单击“New”按钮，新建文件后，即可在该文件中运行代码。

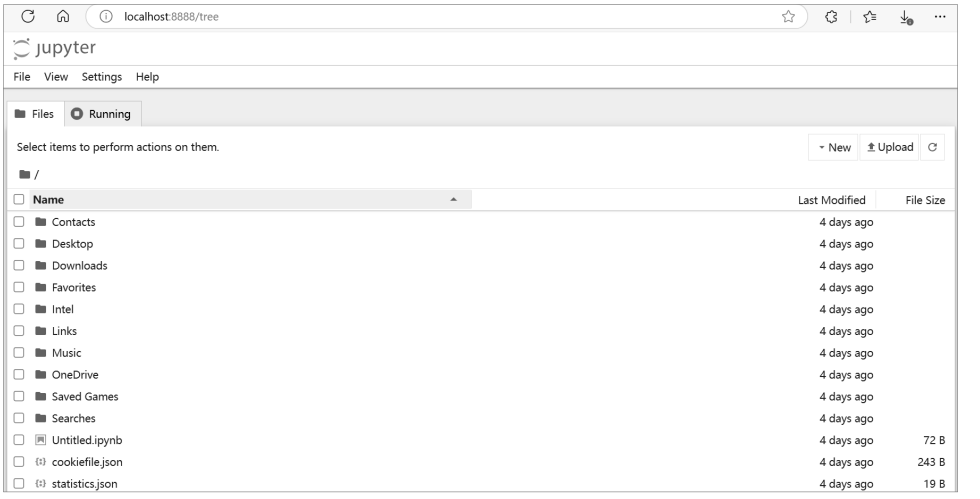

图 1-5　Jupyter Notebook 界面

1.3 Python 基本语法

1.3.1 标识符

在 Python 中，标识符用于为变量、函数、类、模块等元素命名。它为代码中的各元素进行唯一标识，以便引用和操作。

1. 标识符的命名规则

在 Python 中标识符需要遵循的命名规则如下。

（1）组成字符：只能包含字母（a ~ z、A ~ Z）、数字（0 ~ 9）和下划线（_）。

（2）首字符：必须以字母或下划线开头，不能以数字开头。

（3）区分大小写：如 myVar 和 myvar 是两个不同的标识符。

（4）关键字：不能使用 Python 的关键字（if、else、for、while 等）。

（5）长度：理论上没有长度限制，但建议遵循简洁、易懂的规则。

2. 标识符的命名规范

为了提高代码的可读性和可维护性，在 Python 中标识符需要遵循的命名规范如下。

（1）变量名：使用小写字母，单词之间用下划线分隔，如 student_name。

（2）函数名：使用小写字母，单词之间用下划线分隔，如 calculate_average。

（3）类名：使用单词首字母大写的驼峰命名法，如 StudentInfo。

（4）常量名：使用大写字母，单词之间用下划线分隔，如 MAX_VALUE。

标识符是 Python 编程的基础。良好的标识符命名习惯不仅能够提升代码的可读性，还能够增强代码的可理解性和可维护性。因此，在进行 Python 编程时，应严格遵守标识符的命名规则与命令规范，以确保代码结构清晰、逻辑明确，从而编写出高质量、易于协作和维护的 Python 程序。

1.3.2 关键字

关键字（keywords）是 Python 中具有特殊含义和功能的保留字。关键字被 Python 的解释器用于定义语法结构和程序逻辑，不能被用作标识符，如变量名、函数名等。

用于构建程序基本结构的关键字如下。

（1）流程控制的关键字有 if、else、elif、for、while、break、continue、pass。

（2）函数定义的关键字有 def、return、lambda。

（3）类和对象定义的关键字有 class、self、init。

（4）模块和包导入的关键字有 import、from、as。

（5）异常处理的关键字有 try、except、finally、raise。

（6）逻辑运算的关键字有 and、or、not。

（7）变量作用域定义的关键字有 global、nonlocal。

（8）其他功能的关键字有 True、False、None、in、is、with、yield、del、assert。

关键字是 Python 的基础组成部分。理解并正确使用关键字是进行 Python 编程的前提

条件。在编写代码时，应避免使用关键字作为标识符，以免引发语法错误。注意，关键字区分大小写，如 if 是关键字，但 IF 不是关键字。此外，关键字会随着版本的更新而发生变化，建议查阅官方文档获取最新信息。

1.3.3　变量

在 Python 中，变量是用来存储数据的容器。每个变量都有一个标识符，用于引用存储在内存中的数据。变量可以存储不同类型的数据，如数字、字符串、列表、字典等。

1. 变量的赋值

使用赋值运算符“=”可以将数据存储到变量中。赋值语句的基本语法是“变量名 = 值”。

```
a = 123                    # 将数字 123 赋给变量 a
b = "Hello, World!"        # 将字符串"Hello, World!"赋给变量 b
```

Python 除支持为单个变量赋值外，还支持同时为多个变量赋值。

```
x,y,z = 1,2,3              # 将数字 1 赋给变量 x，将数字 2 赋给变量 y，将数字 3 赋给变量 z
x,y = y,x                  # 交换两个变量的值
```

2. 变量的类型

Python 是一种动态语言，变量的类型由其所存储的数据决定，且可以在程序运行过程中改变。使用 type()函数可以查看变量的类型。

```
x,y = 1,'1'
print(type(x),type(y))     # 输出不同的类型
x,y = 2.888,[50,60,70]     # 在程序运行过程中改变变量的类型
print(type(x),type(y))     # 输出不同的类型
```

1.3.4　行和缩进

在 Python 中，行是代码的基本组成单位。在通常情况下，一行代码对应一条语句。如果一条语句过长，那么可以使用换行符（\）将一条语句拆分成多行。

```
print("Hello, World!")     # 一行代码对应一条语句
allsum = item_1 + \        # 将一条语句拆分成两行
         item_2
```

缩进是 Python 语法的重要组成部分，用于定义代码的层次结构。Python 使用缩进表示代码的开始和结束。

```
if x > 0:
print("x is greater than zero")
```

对于多行语句，后续行需要与第 1 行保持相同的缩进量。

```
Allsum = item_1 + \        # 多行语句的缩进
         item_2 + \
         item_3
```

在 Python 中，缩进规则是定义代码层次结构的关键规则。一段代码中的所有语句必须保持相同的缩进量，以确保代码的逻辑结构清晰。虽然 Python 允许使用空格或制表符进行缩进，但为了代码的一致性和可读性，建议使用 4 个空格作为标准的缩进量。如果代码中的缩进量不一致，那么解释器会抛出 IndentationError 异常，导致程序无法正常运

行。因此，严格遵守缩进规则是实现 Python 编程的基本要求。

1.3.5 引号和转义字符

在 Python 中，引号用于定义字符串。字符串是 Python 中表示文本的基本数据类型，包含字母、数字、符号等字符。

Python 支持以下 3 种类型的引号。

（1）单引号（'）：如'Hello, World!'。

（2）双引号（"）：如"Hello, World!"。

（3）三引号（'''或"""）：如'''Hello, World!'''或"""Hello, World!"""。

在 Python 中，引号的使用应遵循以下规则：单引号和双引号在功能上是完全等价的，可以根据需要互换使用。如果字符串中包含单引号，那么可以使用双引号定义字符串，以免冲突发生；反之，如果字符串中包含双引号，那么可以使用单引号定义字符串。此外，三引号通常用于定义多行字符串，或在字符串中包含换行符的情况，它能够保留字符串中的换行和缩进格式，尤其适用于多行文本或文档字符串的定义。

```
a = 'Hello, World!'                          # 使用单引号定义字符串
b = "Hello, World!"                          # 使用双引号定义字符串
c = "It's a beautiful day!"                  # 字符串中包含单引号
d = 'He said, "Hello!"'                      # 字符串中包含双引号
e =    """This is a multi-line string.
It spans across multiple lines. """          # 使用三引号定义多行字符串
```

如果需要在字符串中使用特殊字符（引号、换行符等），那么可以使用“\”作为转义字符进行转义。

```
a = 'It\'s a beautiful day!'                 # 使用转义字符表示单引号
b = "He said, \"Hello!\""                    # 使用转义字符表示双引号
c = "This is a line. \nThis is another line."  # 使用转义字符表示换行符
```

理解并正确使用单引号、双引号和三引号，以及转义字符，是实现 Python 编程的基础。

1.3.6 注释

注释是 Python 中用于解释代码的文本。注释不会被解释器执行。其主要作用是提高代码的可读性和可维护性，帮助开发者理解代码的功能和逻辑。

Python 支持以下两种类型的注释。

（1）单行注释：以“#”开头，“#”后的内容会被视为注释，直到行尾。

（2）多行注释：使用三引号引用，可以跨越多行。

```
X = 100 # 这是单行注释
"""
这是一个多行注释，
可以跨越多行
"""
```

注释是 Python 中不可或缺的一部分，良好的注释习惯可以提高代码的可读性、可维护性和可协作性。在编写代码时，需遵循注释的规范，养成添加注释的习惯，不要过度注

释，只对必要的代码进行注释即可。

1.3.7　空行

在 Python 中，空行是不包含任何代码（包括注释）的行。空行不会影响程序的执行。其主要作用如下。

（1）在不同的函数或类之间添加空行，可以使代码结构更加清晰。

（2）在逻辑相关的语句块之间添加空行，可以提高代码的可读性。

（3）在导入语句和其他代码之间添加空行，可以使代码更加整洁。

空行是代码中用于分隔语句块、提高代码可读性的重要元素。合理使用空行，会使代码结构更加清晰，代码更易于理解和维护，而过度使用空行则会使代码显得松散。

1.3.8　编码规范

编码规范是编写代码时应遵循的一系列规则和约定，旨在提高代码的可读性、可维护性和一致性。Python 社区广泛采用 PEP 8，遵循 PEP 8 可以使代码更易于阅读和理解。

PEP 8 涵盖了代码多方面的规范，下面列举一些关键内容。

（1）缩进：使用 4 个空格作为缩进量，避免使用制表符。

（2）行长度：每行代码均不超过 79 个字符，当代码过长时使用换行符进行换行。

（3）空行：使用空行分隔函数、类等，提高代码的可读性。

（4）导入：将导入语句放在文件顶部，按标准库、第三方库、本地库的顺序分组，每两组之间用空行分隔。

（5）命名规范如下。

① 变量名、函数名使用小写字母，单词之间用下划线分隔，如 snake_case。

② 类名使用驼峰命名法，如 CamelCase。

③ 常量名使用大写字母，单词之间用下划线分隔，如 UPPER_CASE。

（6）使用注释解释代码的功能和逻辑，注释与代码同步更新。

（7）在运算符、逗号等符号周围添加空格，提高代码的可读性。

遵循 Python 的编码规范是编写高质量代码的重要保障。PEP 8 提供了详细的编码风格指南，建议读者在编写代码时严格遵守。此外，还可以使用 Flake8、Black 等工具检查和格式化代码，以更高效地保持代码风格的一致性。

1.3.9　Python 语法综合辨析

Python 作为一种简洁、易读、功能强大的程序设计语言，在语法设计上体现了“优雅”“明确”“简单”的理念。为了编写出高效、易维护的程序，需要深入理解并灵活运用 Python 语法的相关知识。下面以一个构建人工智能模型的实例对 Python 语法进行综合辨析。

实例 1-5　基于文本的情感分析

情感分析是自然语言处理（Natural Language Processing，NLP）任务中的一个常见任务，用于判断一段文本的情感倾向（正向、负向或中性）。本实例通过 Python 的 TextBlob 对象实现一个简单的情感分析程序，适合初学者学习和理解 Python 的基本语法，以及初步应用人工智能。

```
# 导入必要的库
from textblob import TextBlob

# 定义情感分析函数，便于复用
def analyze_sentiment(text):                    # 使用 def 定义函数
    """
    分析文本的情感
    :param text: 输入的文本
    :return: 情感极性（-1 到 1，负数为负向，正数为正向）
    """
    blob = TextBlob(text)                       # 创建 TextBlob 对象
    sentiment = blob.sentiment.polarity         # 获取情感极性
    return sentiment

# 定义 main()函数
def main():
    # 输入文本
    text = "Python is an amazing language for beginners!"

    # 调用情感分析函数
    sentiment = analyze_sentiment(text)

    # 使用条件语句判断情感极性，注意缩进
    if sentiment > 0:
        print("这段文本的情感是正向的！")  # 输出结果
    elif sentiment < 0:
        print("这段文本的情感是负向的！")
    else:
        print("这段文本的情感是中性的。")

# 程序入口
if __name__ == "__main__":
    main()    # 确保程序在直接运行时执行 main()函数
```


1.4　人机交互（基本输入/输出）

1.4.1　程序设计的基本流程


微课视频

程序设计并非一蹴而就的，而是一个循序渐进、不断迭代的过程。为了

编写出结构清晰、功能完善、易于维护的程序，需要遵循一定的程序设计流程。下面是程序设计的基本流程。

1. 问题分析与需求定义

在程序设计之初，首先需要明确目标，即需要解决的具体问题，以及实现的核心功能，如数据分析、网站构建、游戏开发等；其次需要对问题进行细致的分析，将其拆解为多个可管理的子部分，并明确程序的输入、输出及各项功能需求；最后需要综合考虑程序在性能、资源和时间等方面的约束条件，以确保程序设计的可行性和高效性。

2. 算法设计与流程规划

在明确需求后，需要设计算法，即规划解决问题的具体步骤和方法，可以通过自然语言、流程图、伪代码等形式清晰地描述算法的逻辑。同时，需要根据算法的需求选择合适的数据结构来高效地存储和操作数据，如列表、字典、集合等。此外，需要将算法转换为程序运行流程，合理划分模块并确定函数之间的调用关系，从而构建出结构清晰、逻辑严谨的程序框架。

3. 代码编写与调试

在设计好算法和规划好流程后，即可开始编写代码，利用 Python 的语法规则和丰富的库函数将逻辑转换为实际代码。在编写过程中，应严格遵循 PEP 8，确保代码风格一致、清晰易读，以便后续维护和协作。编写完成后，需通过调试工具或 print()函数对代码进行调试，定位并修复潜在的错误，以确保程序的正确性和稳定性。

4. 测试与优化

在编写与调试好代码后，需要通过单元测试对程序的各模块进行验证，编写有针对性的测试用例，以确保每个模块的功能都符合预期。同时，需要进行集成测试，将各模块组合起来，测试它们之间的交互是否正常，以确保整个系统的协调运行。此外，需要针对程序的性能瓶颈对程序进行优化，如采用更高效的算法或数据结构，以提升程序执行效率和资源利用率。

5. 文档编写与维护

为了提升代码的可读性和可维护性，需要在代码中添加必要的注释，清晰地解释代码的功能和逻辑，以便他人理解和后续修改。同时，需要编写详细的用户文档，说明程序的使用方法、功能特性及注意事项，以帮助用户快速上手并解决常见问题。此外，在发布程序后需要根据用户反馈和需求变化对程序进行持续的维护，修复潜在问题并优化功能，以确保程序长期稳定运行和用户体验的不断提升。

1.4.2　标准输出函数 print()


微课视频

在程序设计的基本流程中进行代码编写与调试时，需要将程序运行结果展示给用户，这就涉及数据的输出。Python 提供了多种输出方式，其中常用的是使用标准输出函数 print()。print()函数用于将指定的内容输出到控制台中，是程序与用户交互的重要桥梁。以下是 print()函数的语法。

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

（1）objects：接收任意多个参数，将其转换为字符串后输出。

（2）sep：指定多个输出对象之间的分隔符，默认值为空格。

（3）end：指定输出结尾的字符，默认值为换行符。

（4）file：指定输出流，默认值为 sys.stdout。

（5）flush：指定是否强制刷新输出缓冲区，默认值为 False。

```
print("Hello, world!")              # 输出单个对象。输出“Hello, world!”
print("The answer is",42)           # 以空格分隔多个对象。输出“The answer is 42”
print("2023","10","27",sep="-")     # 指定分隔符。输出“2023-10-27”
print("Loading",end="...")          # 指定结尾的字符，不用换行。输出“Loading…”
```

1.4.3 print()函数的格式控制

为了提升输出结果的可读性和美观性，需要对格式进行控制，如对齐文本、设置小数点位数、填充字符等。Python 提供了多种方式来实现 print()函数的格式控制，包括使用 f-string(格式化字符串字面值)和使用 format()方法等。

1. f-string(格式化字符串字面值)

f-string(格式化字符串字面值)是 Python 3.6 引入的新特性，语法简洁，可读性强。通过在字符串前添加字母 f，并在字符串中使用花括号直接嵌入表达式，可以将表达式的值插入字符串。

```
print(f"我的名字是 {name}，今年 {age} 岁了。")
print(f"{age} 岁是法定成年的年龄")
```

2. format()方法

format()方法提供了更灵活、更强大的字符串格式化功能。通过在字符串中使用花括号作为占位符，并在 format()方法中传入参数，可以将参数的值插入字符串。

```
print("我的名字是 {}，今年 {} 岁了。".format(name, age))
print("我的名字是 {1}，今年 {0} 岁了。".format(age, name))
print("pi 的值是 {:.2f}".format(pi))
```

3. 字符串拼接的方法

用“+”可以将多个字符串拼接为一个字符串输出。参与拼接的可以是字符串，也可以是字符串变量。当参与拼接的变量为整数等其他数据类型时，需要先使用 str()函数将其转换为字符串再参与拼接。

```
print('我叫' + name + '，今年' + str(age) + '岁')  # 输出“我叫王伟，今年 18 岁”
```

4. 格式控制

格式控制在实际应用中非常广泛。

（1）对齐文本：可以通过“<”“>”“^”分别实现左对齐、右对齐和居中对齐。

（2）设置小数位数：可以通过“:.nf”控制浮点数的小数位数。

（3）填充字符：可以通过在“:”后添加填充字符填充文本。

```
print(f"{'Name':<10}|{'Age':>5}|")      # 输出“Name      |  Age|”
print(f"{'Alice':<10}|{25:>5}|")        # 输出“Alice     |   25|”
print(f"{'Bob':<10}|{30:>5}|")          # 输出“Bob       |   30|”
pi = 3.1415926
print(f"Pi 的值为: {pi:.2f}")           # 输出“Pi 的值为: 3.14”
print(f"{'Hello':*^10}")                # 输出“**Hello***”
```

实例 1-6　　**格式化模型预测结果**

假设有一个机器学习模型，要求根据房屋特征预测房屋价格。本实例需要将模型预测结果格式化为易于理解的报告。

```
area = 120.5                          # 特征 1，房屋面积（平方米）
rooms = 3                             # 特征 2，房间数量
location = "市中心"                    # 特征 3，房屋位置
predicted_price = 1234567.89          # 假设模型预测结果，总价格（元）
predicted_price_wan = predicted_price / 10000  # 转换价格
print(f"特征：面积 {area:.1f} 平方米，{rooms} 个房间，位于 {location}。")
print(f"预测的房屋价格为: {predicted_price:,.2f} 元")
print(f"预测的房屋价格为: {predicted_price_wan:.2f} 万元")
```

程序运行结果如下。

```
特征：面积 120.5 平方米，3 个房间，位于市中心。
预测的房屋价格为: 1,234,567.89 元
预测的房屋价格为: 123.46 万元
```

通过 print()函数的格式控制可以将程序运行结果、提示信息等内容清晰地展示给用户。然而，许多程序需要根据用户的输入动态调整其行为，如计算用户的输入数据、根据用户的选择执行不同的操作等。为了实现这种交互，Python 提供了 input()函数，用于从用户处获取输入数据。

1.4.4　标准输入函数 input()


微课视频

一个功能完备的程序不仅需要向用户输出信息，还需要与用户进行交互，以获取用户的输入数据。input()函数通过标准输入流（stdin）读取用户输入的一行内容（默认关联键盘输入），并返回字符串。以下是 input()函数的语法。

```
input([prompt])              # 仅用于语法展示
```

prompt：可选参数，用于在输入前向用户显示提示信息。

```
name = input("Please enter your name:")      # 获取用户的输入数据，如获取“小李”
print(f"你好，{name}!")    # 输出用户的输入数据，如输出“你好，小李”
```

input()函数返回的是字符串，如果需要其他类型的数据，如整数、浮点数等，那么需要进行类型转换。此外，在实际应用中，常常需要将格式控制与输入/输出结合起来，以提升程序的交互性和可读性。

实例 1-7　　**设计系统运行状态问卷调查程序**

设计一个系统运行状态问卷调查程序，收集并输出用户反馈。

```
print("系统运行状态问卷调查")# 输出问卷标题
print("=" * 30)
# 输出问题并收集用户对各项指标的评分
print("1. 您对系统整体运行速度的满意度如何？（1～5 分）")
speed_score = int(input("请输入您的评分: "))
```

```
print("2. 您对系统稳定性的满意度如何？（1～5 分）")
stability_score = int(input("请输入您的评分: "))
print("3. 您对系统界面设计的满意度如何？（1～5 分）")
ui_score = int(input("请输入您的评分: "))
print("感谢您参与问卷调查！您的反馈对我们来说非常重要。")# 输出感谢信息
# 输出用户反馈
print("=" * 30)
print(f"运行速度评分: {speed_score}")
print(f"稳定性评分: {stability_score}")
print(f"界面设计评分: {ui_score}")
```

上述实例先使用 print() 函数依次输出问卷标题、问题及感谢信息，引导用户完成问卷的填写，然后使用 input()函数实时收集用户对各项指标的评分，最后输出用户反馈。当然，也可以使用写文件的方式，将用户反馈以结构化形式保存，以便后续分析和处理。

实例 1-8　设计模拟简单的心情对话机器人的程序

设计一个模拟简单的心情对话机器人的程序，要求用户输入自己的心情，计算机会根据心情给出相应的回应。这个实例用到的 if 语句将在第 3 章中详细介绍。

```
print("您好！我是心情对话机器人。请告诉我您今天的心情吧！")
mood = input("您今天的心情如何？（开心/难过/生气/平静）: ")# 获取输入数据
# 心情对话机器人的回应逻辑
if mood == "开心":
    print("心情对话机器人: 太棒了！希望您一直保持好心情！")
elif mood == "难过":
    print("心情对话机器人: 别难过，一切都会好起来的！")
elif mood == "生气":
    print("心情对话机器人: 深呼吸，冷静一下，事情会解决的。")
elif mood == "平静":
    print("心情对话机器人: 平静是一种智慧，享受当下的宁静吧。")
else:
    print("心情对话机器人: 我不太明白您的心情，但无论如何，我都在这里陪您。")
```

如果用户输入了“开心”，那么程序运行结果如下。

```
您好！我是心情对话机器人。请告诉我您今天的心情吧！
您今天的心情如何？（开心/难过/生气/平静）: 开心
心情对话机器人: 太棒了！希望您一直保持好心情！
```

1.5　项目实训：个人名片设计——美好愿景

1.5.1　项目背景

在当今社会，个人名片是展示个人形象、传递联系方式的重要工具。一个设计精美、信息清晰的个人名片，不仅能够给人留下深刻的印象，还能够有效地提升个人品牌价值。

1.5.2　项目任务

本项目实训旨在通过 Python 编程，设计一个交互式的个人名片生成程序，用户可以根据提示输入个人信息，并选择自己喜欢样式，程序将自动生成美观的个人名片。

1.5.3　项目分析

（1）使用 input()函数获取用户的个人信息，如姓名、职位、电话、邮箱等。
（2）使用 print()函数和格式控制，设计名片样式。
（3）将生成的名片输出到控制台或保存到文件中。

1.5.4　项目实现

（1）信息输入模块。

```
name = input("请输入您的姓名: ")
title = input("请输入您的职位: ")
phone = input("请输入您的电话: ")
email = input("请输入您的邮箱: ")
```

（2）名片输出模块。

```
print("=" * 30)
print(f"{'姓名':<10}{user_info['name']}")
print(f"{'职位':<10}{user_info['title']}")
print(f"{'电话':<10}{user_info['phone']}")
print(f"{'邮箱':<10}{user_info['email']}")
print("=" * 30)
```

1.5.5　项目优化

为了进一步提升个人名片生成程序的功能和用户体验，可以从以下方面进行优化。首先，丰富名片模板库，设计多种样式的名片模板，以满足用户多样化的需求；其次，引入 GUI，使用 Tkinter 等开发直观的操作界面，以提升程序的交互性和易用性；再次，增加名片预览功能，允许用户在生成名片前实时查看效果，并支持对名片样式进行调整；最后，扩展批量生成功能，支持同时为多个用户生成名片，以应对团队或公司等需要批量处理的场景，从而提升程序的实用性和效率。

1.5.6　总结和启示

本项目实训介绍了如何使用 Python 的标准输入/输出函数和格式控制，设计并实现一个交互式的个人名片生成程序。读者通过学习本项目实训，不仅可以加深对 Python 基础知识的理解，还可以锻炼 Python 编程实践能力。在实际开发中，应根据需求不断优化和扩展程序功能，以提升用户体验和程序的实用性。本项目实训通过合理的项目分析和项目实现，将复杂的问题分解为简单的模块。掌握这种思维方式对 Python 编程的学习和实

践都具有重要的指导意义。

1.6 巩固练习

（1）编写一个 Python 程序，要求使用 input()函数输入姓名、年龄和职业，并使用 print()函数和格式控制输出以下结果。

```
姓名: [姓名]
年龄: [年龄]
职业: [职业]
```

（2）编写一个 Python 程序，要求使用 input()函数输入圆的半径，并使用 print()函数和格式控制输出以下结果。

```
圆的面积为: [面积] # 保留两位小数
```

（3）编写一个 Python 程序，要求使用 input()函数输入一个整数，判断该整数是奇数还是偶数，并使用 print()函数输出以下结果。

```
[数字] 是奇数/偶数
```

（4）编写一个 Python 程序，要求使用 input()函数输入摄氏温度，将其转换为华氏温度，转换公式为“华氏温度 = 摄氏温度 × 9/5 + 32”，并使用 print()函数输出以下结果。

```
摄氏温度: [摄氏温度] → 华氏温度: [华氏温度] # 保留一位小数
```

（5）编写一个 Python 程序，要求使用 input()函数输入两个数字和一个运算符（“+”“–”“*”“/”），并使用 print()函数输出以下结果。

```
[数字 1] [运算符] [数字 2] = [结果]
```

（6）编写一个 Python 程序，要求使用 input()函数输入身高（米）和体重（千克），公式为“BMI = 体重 / (身高 × 身高)”，使用 print()函数输出以下结果。

```
BMI: [BMI] # 保留两位小数
```

（7）编写一个 Python 程序，要求使用 input()函数输入一个字符串，将其反转，并使用 print()函数输出以下结果。

```
反转后的字符串为: [反转后的字符串]
```

（8）编写一个 Python 程序，要求使用 input()函数输入一个整数，判断该整数是否为回文数，并使用 print()函数输出以下结果。

```
[数字] 是回文数/不是回文数
```

（9）编写一个 Python 程序，要求使用 input()函数输入用户名和密码，判断输入的用户名和密码是否正确（假设用户名为 admin、密码为 123456），并使用 print()函数输出以下结果。

```
登录成功！/登录失败！
```