第3章 Python数据处理基础

在当今数字化与智能化飞速发展的时代，编程已不再是计算机专业人员的专属技能，而是成为各学科领域进行创新研究、解决复杂问题以及提升工作效率的通用工具。与此同时，人工智能技术的突破有助于跨越语法门槛，使学习者快速聚焦于数据逻辑的理解与应用。本章将介绍如何运用 AI 工具辅助 Python 语言实现数据处理。

3.1 Python 语言概述

Python 是一门易于学习、功能强大的编程语言。Python 拥有丰富的标准库和第三方模块，应用领域广泛，是目前人工智能领域中的常用语言之一。

▶ 3.1.1 Python 语言特点

高级语言翻译成机器语言主要包括两种方式：编译与解释。两者的特性对比如表 3-1 所示。Python 代码由解释器逐行执行，无须像 C++等语言那样先编译成机器码。

表 3-1 解释型与编译型语言的特性对比

特　　性	Python(解释型)	C++/Java(编译型)
执行方式	逐行解释执行	先编译成机器码再执行
开发速度	快(即时反馈)	慢(需要编译)
调试	简单(逐行执行)	复杂(需要重新编译)
性能	较慢	较快
依赖	解释器	编译器＋运行时环境

与 C 等其他语言相比，Python 具有以下优点。

① 简单易学。Python 的关键字较少，结构简单，语法简洁清晰。初学者易于上手，可将注意力更集中于算法和模型设计本身，而不是复杂的语法细节。

② 开源免费。Python 是免费开源软件，用户可以自由地下载、使用和分发其源代码。Python 的开发者社区也提供了丰富的学习资源。

③ 完全面向对象。函数、模块、数字、字符串等一切皆对象。代码复用性好。

④ 支持函数式编程。例如 Lambda 表达式、匿名函数等，可使代码编写更简洁，可读性更强，复用性更好。

⑤ 丰富的库。Python 提供了功能强大的标准库及第三方库，包括各种常用的功能模块，能够快速实现各种任务，极大简化了开发过程。目前在人工智能领域，几乎所有主流框架都支持 Python，学习 Python 后可以直接使用这些强大的工具进行机器学习和深度学习应用。

⑥ 可移植易扩展。Python 具有跨平台特点，可以在多种操作系统中运行。同时，具有很好的扩展性，既可以调用 C 或 C++程序来提升性能，也可以嵌入 C/C++程序中作为脚本接口，非常适合处理大规模数据和复杂计算。

尽管 Python 有诸多优点，但在性能上也面临一些挑战。在处理高速计算、实时性或高并

发的任务时,其运行效率通常不如 C 或 C++等编译型语言。因此,在实际应用中,需要考虑 Python 的这些性能局限,必要时可以结合 C/C++等其他语言来弥补性能短板。

在版本与依赖管理方面,不同版本的 Python(主要是 Python 2 与 Python 3)语法可能不兼容;同时,第三方库的不同版本之间也往往存在差异,例如新版本中废弃了旧版本的部分函数。这可能导致在不同环境中的运行结果不一致。因此,在实际开发中,构建独立、隔离的运行环境至关重要,需要在项目初期做好版本规划,确定合适的 Python 版本及依赖库版本。

▶ 3.1.2 Python 在人工智能领域的支持

Python 语言贯穿于数据处理到智能实现的全过程,为人工智能的开发提供了强有力的支持,主要体现在以下几个方面。

① 科学计算与数据分析。数据的数学表示与运算方法是人工智能算法的基础。引入 NumPy、Pandas 等功能库,Python 能够高效地处理复杂的数学运算和表格数据,并将数据绘制成直观的图表,进而发现数据背后隐藏的规律。

② 人工智能与机器学习。这是目前 Python 应用最热门的领域。借助 TensorFlow、PyTorch 等成熟的人工智能框架,开发者无须编写底层代码,就能直接构建和训练智能模型,实现图像识别、自然语言对话等复杂的 AI 应用。

③ 自动化与数据采集。训练模型往往需要海量数据作为支撑。Python 可以编写网络爬虫脚本,自动从互联网中抓取所需信息,也能自动完成重复性的文件处理任务,极大地提升了数据准备和处理的效率。

④ Web 开发与系统部署。AI 模型训练好后,可以快速开发网站或应用程序接口(API)。通过这种方式,可将封装好的 AI 模型部署到服务器上,进而让用户通过网页或手机终端体验到智能服务的功能。

3.2 Python 环境与 AI 开发平台

对于初学者而言,编程环境的配置往往是入门的第一道关卡。编程环境主要分为本地部署和云端在线两种形式,二者各有其独特的优势与适用场景。本地部署是指在个人计算机上安装解释器与编辑器,适合需要深度定制和离线开发的场景;云端在线编程则是通过浏览器直接访问远程服务器,无须安装配置,即开即用。

▶ 3.2.1 Python 本地开发环境

本地部署高级语言环境,主要涉及安装解释器、选择合适的代码编辑器。Python 解释器是负责将编写的代码翻译成机器能够执行指令的核心程序,而代码编辑器则是供开发者编写和修改代码的工具。

Python 的核心优势在于其拥有海量且功能强大的第三方库,这些库涵盖了数据分析、人工智能等应用。因此,搭建一个能够方便管理这些库的开发环境至关重要。对于初学者而言,Python 环境的搭建主要有两条路径:一是专注于语言基础的轻量化方案,二是面向工程应用的一站式解决方案。

1. Python 官网下载包

如果学习的初衷仅仅是掌握 Python 的内置语法、理解编程逻辑,而不涉及复杂的第三方

库，那么直接访问 Python 官方网站下载安装包是最直接的方式。安装包中自带的 IDLE (Integrated Development and Learning Environment，集成开发环境)是一个轻量级的开发工具，它包含了 Python 解释器和基本的代码编辑功能，完全满足运行基础代码的需求。然而，一旦进入实际应用阶段，就需要频繁下载和安装各种第三方库，手动配置往往容易出错。

2. Anaconda

对于希望系统学习 Python 并能够进行实际操作的初学者，推荐直接安装 Anaconda。Anaconda 是一个开源的 Python 发行版，虽然安装体积较大，但是它集成了 Python 解释器、Conda 开源包管理工具、常用的第三方库和一些工具，极大地降低了环境配置的难度，非常适合初学者快速上手。

在安装好 Anaconda 后，使用其自带的 Jupyter Notebook 即可开始练习。图 3-1(a)显示了 Jupyter Notebook 界面，它以网页形式打开。单击 New 按钮，在下拉菜单中选择 Python3，新建一个 Untitled 开头的 Python 文件。单击文件名可以对其进行修改，保存后的文件扩展名为.ipynb。这种文件称为交互式笔记本，可以保存代码、运行结果和文本，非常适合做学习笔记。

编辑代码非常简单。如图 3-1(b)所示，在 In[]右边的单元格中输入代码，单击顶部“Run”按钮，运行结果会直接显示在代码块下。单击顶部工具栏可以方便地上下移动单元格、增加或删除单元格。

(a) 在Jupyter Notebook中新建一个Python 3文件　　(b) 在In[]: 文本框中输入代码

图 3-1　在 Jupyter Notebook 中新建一个 Python 源程序

3. Visual Studio Code

Visual Studio Code(简称 VS Code)是由微软开发的一款免费、开源且跨平台的代码编辑器。通过扩展丰富的插件，VS Code 支持大多数编程语言，其强大的代码调试、代码重构、智能补全等功能，能够帮助用户更高效地编写代码。此外，它具有轻量级和启动速度快等特点，因此在全球开发者中得到了广泛欢迎。

在官网下载后，单击左边的扩展图标，在搜索框中输入“Python”，安装 Microsoft 公司的 Python 插件。然后按 Ctrl+Shift+P，输入 Python 解释器(Python: Select Interpreter)，在下拉项中选择本地已安装的 Python 解释器，如图 3-2 所示。

图 3-2　VS Code 中选择解释器

4. PyCharm

PyCharm 是一款功能强大的 Python 集成开发环境(IDE),由 JetBrains 公司开发,社区版(Community)免费,具有多平台安装包,适合完成较复杂的工程项目。它的智能代码补全、语法高亮和调试功能,能提高编程效率。

官网下载后,按照提示完成安装。启动 PyCharm 后,它会自动检测本地已安装的 Python 环境,例如可以从已安装的 VS Code 设置中导入。设置好解释器后即可创建项目,然后创建 Python 文件,扩展名为.py,如图 3-3 所示。在文件中输入代码后,单击 run 按钮运行,在底部窗口中观察输入结果。此外,也可以新建 Jupyter Notebook 文件(扩展名为.ipynb)。

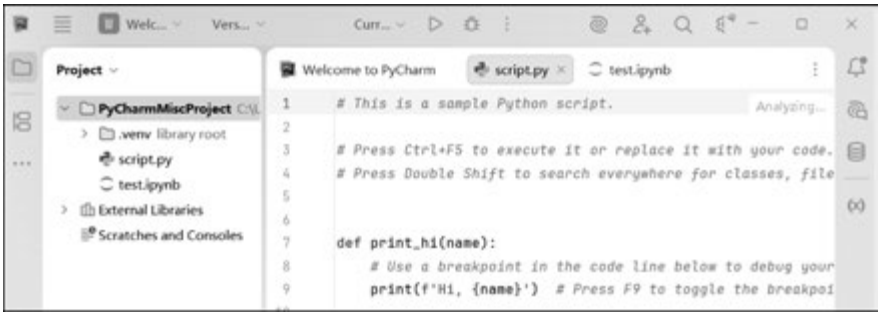


图 3-3 Pycharm 新建项目示例

Python 内置的调试器,允许设置断点、单步执行代码、查看变量值,逐行追踪程序的执行流程,这对于理解复杂逻辑和定位 bug(程序错误)至关重要。

▶ 3.2.2 Python 云端编程环境

云端环境无须复杂的安装配置过程,用户仅需通过浏览器即可访问配置完善的编程平台,这对于使用公用电脑或受限于系统权限的学习场景来说尤为实用。云端平台通常基于 Jupyter Notebook 技术构建,这种交互式的编程环境允许将代码、说明文字、公式和图表整合在同一个网页文档中,非常适合用于数据分析、科学计算以及编程教学。此外,现代云端平台往往深度集成了 AI 算力与智能助手,为学习者提供了从代码编写到调试的一站式体验。

1. **国内云端平台**

目前,国内科技公司与开源社区已推出了多款优秀的云端 AI 编程平台。这些平台普遍基于 Jupyter Notebook 或云端 IDE 技术构建,用户仅需通过浏览器即可获得配置完善的 Python 开发环境。这些平台不仅提供了基础的代码编写与运行功能,还集成了高性能的 GPU 算力、丰富的公开数据集以及预装的主流 AI 框架,极大地降低了编程与人工智能学习的门槛。目前常用的国内云端平台的功能与特点如表 3-2 所示。

表 3-2 国内云端平台对比

平台名称	主要功能	优缺点	适用用户
百度 AI Studio	Notebook 开发环境,深度学习框架,支持模型训练与部署	优点:社区教程丰富,免费算力较足 缺点:特定生态依赖	深度学习初学者、学生
智谱清言	智能对话与代码分析环境,强大的代码生成、解释与调试能力	优点:AI 能力强,交互佳,辅助智能高 缺点:训练功能较少	AI 辅助编程的初学者

续表

平台名称	主要功能	优缺点	适用用户
阿里云天池	Notebook开发环境，集成海量数据集与算法竞赛	优点：资源丰富，竞赛与业务场景多 缺点：偏向竞赛	数据科学爱好者、算法竞赛选手
腾讯云开发者平台	Cloud Studio云端IDE，支持多种编程语言与一键部署	优点：类似VS Code，协作强 缺点：AI算力较少	Web开发、团队协作
华为云ModelArts	全流程AI开发管理，包括数据处理、模型训练、部署等	优点：企业级功能强 缺点：界面相对复杂，上手较难	企业开发者、专业研发工程师
和鲸社区	数据科学协作平台，内置大量数据集与经典案例(K-Lab)	优点：案例丰富，适合项目实战 缺点：算力资源紧张	数据挖掘从业者、竞赛爱好者

2. **云平台使用示例**

下面以百度AI Studio为例，详细介绍云端AI编程环境的具体使用方法。

首先，在浏览器地址栏输入“aistudio.baidu.com”。进入官网后，单击右上角的“登录/注册”按钮，使用百度账号或手机号完成登录。登录成功后，单击左侧“开发广场”下的“项目大厅”按钮，进入后会显示所有项目。

将鼠标移至页面顶部的“创建项目”，在下拉菜单中单击“Notebook”，在弹出的新窗口中输入一个项目名称，如“myFirst”，单击“创建”按钮进入编辑界面。

或进入已有的项目。单击“新建项目”按钮后，需要填写项目名称，如“MyFirstPython”，并选择Python版本，通常推荐选择Python 3.7或更高版本。配置完成后，单击“创建”按钮，系统会自动跳转至Notebook编辑界面。

单击Notebook界面底部的代码“＋”按钮，新建一个代码单元格。在单元格中输入：

```
print('hello')
```

单击单元格左边的“▷”或按“Shift＋Enter”键，执行单元格中的代码，输出结果会显示在单元格下方，如图3-4所示。也可单击底部的“运行全部”按钮，运行该编辑界面文件中所有单元格中的代码。

图3-4　云端Python编程示例

▶ 3.2.3　AI辅助编程工具概览

1. **本地AI插件安装**

在本地编程环境中，AI辅助工具通常以插件的形式集成在主流代码编辑器中。这些插件利用大语言模型的能力，能够实时分析代码上下文，提供智能补全、代码生成、错误检查及自然语言解释等功能，降低了编程门槛。目前，国内已涌现多款优秀的国产AI编程插件，如表3-3所示，它们不仅对中文语义理解更加深入，而且更贴合国内开发者的使用习惯。

表 3-3　主流国产 AI 编程插件

插件名称	主要特点	适用场景
CodeGeeX(智谱)	支持多语言代码补全与生成,具备跨语言翻译功能,开源免费	个人日常开发、学习多种编程语言、代码迁移
通义灵码(阿里云)	基于通义大模型,支持代码生成,还能直接读取企业库知识进行精准编码	企业内部开发、阿里云生态用户
百度 Comate	基于文心大模型,代码生成速度快,企业级功能完善	企业级开发、快速构建项目、百度生态用户
腾讯云 AI 代码助手	集成腾讯混元大模型,支持技术问答和代码优化	云端开发、腾讯云用户

下面以 VS Code 为例,简要介绍安装 CodeGeeX 插件的主要步骤。

首先,在电脑上启动已配置 Python 扩展的 VS Code,单击左边的扩展图标,在搜索框中输入“CodeGeeX”,找到由智谱 AI 开发的插件,如图 3-5 所示。单击“安装”按钮。完成后选择登录,在弹出网页中登录 CodeGeeX 账号。

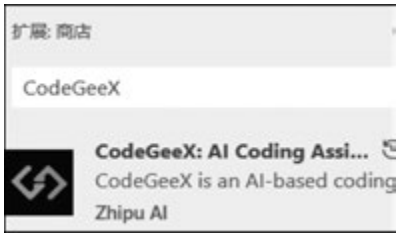


图 3-5　VS Code 中搜索 CodeGeex

下面举例说明 AI 辅助编程。新建一个扩展名为“.py”的文件,第一行输入“#计算列表平均值”作为注释,按 Enter 键换行后,会看到灰色提示代码“def average(lst)”,按下 Tab 键即可接受建议,继续按 Enter 键会补下一行,实现对列表 lst 求平均值并返回。接下来,再写出提示词“#测试函数”,继续逐行生成代码,如图 3-6 所示。

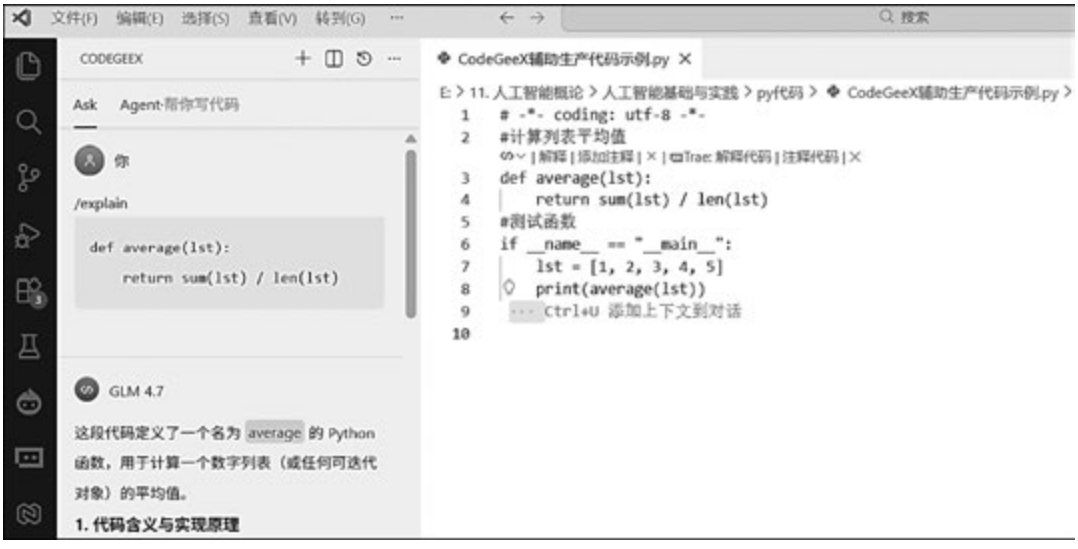


图 3-6　VS Code 中 AI 插件生成代码示例

也可打开 AI 插件的侧边栏对话框,输入“帮我写一个函数,计算数字列表的平均值”,AI 将生成完整的代码块,并解释代码含义。单击“插入”按钮即可。

2. AI 原生开发环境

AI 原生开发环境(AI-Native IDE)是指将人工智能技术从底层深度集成到开发工具中,使 AI 成为编写、调试和管理代码的核心助手,而不仅是一个外接插件。这种环境通常具备全项目上下文理解能力,开发者只需用自然语言提出要求,它就能直接生成功能模块、重构代码甚至创建一个完整应用,实现“所想即所得”的编程体验。目前,这类工具在国内外都很受关注,国外以 Cursor 最为知名,国内也涌现出如 Trae、CodeArts 等优秀产品。

以 Trae 为例,它是由字节跳动推出的代表性国产 AI 原生开发环境。开发者仅需用中文描述需求,AI 便能自动拆解任务并生成包含前后端的项目代码。它支持跨文件理解、多模态交互(如根据截图生成代码)以及实时预览等功能。目前支持 Windows 和 macOS 系统,提供免费使用版本,具备 Builder(构建)和 Chat(对话)两种模式。同时,Trae 支持从 VS Code 或 Cursor 等编辑器一键导入插件和配置,极大地降低了开发者切换工具的学习成本。

Trae 的官网下载地址:https://trae.cn。

安装并启动 Trae 后,按照以下步骤开始创作。

(1) 启动 Solo 模式。选择 Builder(构建)模式,并单击“Solo”按钮,从空白页面开始,进入独立的开发环境。

(2) 在 Trae 的对话输入框中,用自然语言描述你的需求。例如:

创建一个简单的个人博客首页,包含导航栏、一张头像图片和三篇博客文章的摘要

(3) 创建项目。Trae 会提示新建项目,要求在本地电脑上选择或新建一个文件夹,作为项目的保存位置。输入项目名称,单击“确认”按钮。

(4) AI 生成与预览,创建项目后,Trae 开始“思考”,分析需求,并逐步分解任务。同时,窗口右边会自动生成相关代码,并在预览窗口中实时显示效果。稍等片刻后,会看到制作的博客网页,如图 3-7 所示。

图 3-7　Trae 中生成个人博客网页示例

3. 云端 AI 工具

这种方法下,用户无须安装任何软件,通过浏览器即可享受 AI 辅助编程的便捷。例如,打开文心一言、豆包、Kimi、DeepSeek 等大模型网页,直接通过自然语言提问,将 AI 生成的代码粘贴到本地或云端代码编辑器中。也可将编译中的错误信息以文本或图片方式粘贴到网页中提问,AI 会帮助纠错并解释原因。

3.3 AI 辅助下的 Python 入门

在人工智能技术的辅助下,我们可以将语法记忆等机械性工作交由 AI 处理,从而将精力集中于编程逻辑的理解与问题的解决,实现编程语言的高效入门。下面来学习简单的 Python 程序设计,并理解 AI 在辅助编程及调试中的作用。

▶ 3.3.1 第一个 Python 程序

1. 提出要求

如图 3-8(a)所示,向 AI 提出要求:

```
输出一句欢迎用语“Hello”
输出 10+20 的值
```

2. 观察代码

如图 3-8(b)所示,AI 会立即生成 Python 代码。Python 程序代码灵活且简洁,可以直接写语句,无须框架,每一行代码默认是一条独立的语句。

3. 理解代码

AI 通常还会提供代码解释,如图 3-8(c)所示,解释了 print 是输出函数,圆括号内是要输出的表达式,可以是一对双引号括起来的字符串,也可以是算术运算表达式。当执行 print 函数时,它会输出表达式的值。

(a) 提出要求　(b) 给出代码　(c) 给出代码解释

图 3-8　AI 给出的 Python 程序示例

4. 运行并观察结果

复制代码到 Python 开发环境中,运行并观察结果:

```
Hello
30
```

▶ 3.3.2 变量

如果把编写程序比作写作业,那么前面写的代码就像是固定的例题,而引入了变量和数据

类型，我们就可以根据不同的情况（数据）计算出不同的结果。

1. **变量的作用**

数学中经常用 x、y 等名字来代表一个未知的数。Python 中的变量概念与之类似，但更加灵活和强大。以一道数学题为例："已知一个正方形的边长是 5cm，计算它的面积"。你可能会写出下面的方程式：

```
a = 5
S = a × a
```

这里的 a 就是变量。而数字 5，就是存放在这个变量里的数据，也称为字面量。

在计算机内存中，我们需要开辟一块空间来存储数据，为了方便找到和使用这块空间，给它命名为变量。简单来说，变量就是存储数据的容器。

2. **变量的定义与赋值**

在 Python 中，创建一个变量（也称为"定义变量"）非常简单，不需要提前声明类型。只需要给变量起个名字，然后按照"变量＝数据"的格式书写。例如：

```
msg = "Hello, AI"
score = 95
pi = 3.14159
```

这里，msg、score、pi 都是变量名。请注意，"＝"并不是数学中的"等于"，而是"赋值"操作，对于这几个简单数据类型的变量，"＝"操作会将右边的数据值直接复制到左边变量的内存空间。

3. **变量名的命名规则**

变量名要遵守以下规则：

① 只能包含字母、数字和下画线（_）。例如，user_name 正确，而 user-name 错误。

② 不能以数字开头。例如，x2 正确，而 2x 错误。

③ 不能是 Python 的关键字。例如，if、while、for 是关键字，因此不能用作变量名。

在学习编程时，要养成良好的编程习惯。变量名取名时，尽量做到"见名知意"。例如：

```
age = 10                    # 一个年龄，用 age
ages = [10,12,15,12]        # 一组年龄，用 ages
```

Python 官方的 PEP8 编码规范明确指出，变量名和函数名推荐使用蛇形命名法，即全部小写并用下画线连接单词，如 student_name；而类名则推荐使用大驼峰命名法，即每个单词的首字母大写，如 MyClass。这种区分不仅让代码结构一目了然，也是编写优雅、规范 Python 程序的关键细节。

▶ 3.3.3　数据类型

为了区分不同的数据，Python 定义了不同的数据类型，它们决定了数据在计算机中如何存储以及能进行什么操作。数据分为数值数据和非数值数据，数值数据包括整数、浮点数、布尔值等，非数值数据包括字符、文字、图形等。每类数据有自己的存储、表示形式与编码规则，但是在计算机内部都是一组 0 和 1 的组合。掌握数据类型，是处理现实世界问题的第一关。

在后续的人工智能学习中,无论是处理图像的像素矩阵(数组),还是处理自然语言的文本信息(字符串与列表),都离不开这些基础知识的支撑。初学者要通过多次动手练习,尝试定义不同类型的变量,感受它们之间的区别与联系。

1. **简单数据类型**

我们将最基础的数据类型称为简单数据类型,基本数据类型变量名存储的是值,主要包括整数、浮点数、字符串、布尔值,如表 3-4 所示。

表 3-4 Python 简单数据类型对比

数据类型	说明	常见用途	代码示例
整数(int)	没有小数部分的数字,包括正数、负数、零	数量、计数、索引	age=20 temperature=-5
浮点数(float)	带有小数点或使用科学记数法表示的实数	精确计算、模拟物理量	pi=3.14159 f=1e-5
字符串(str)	字符序列,必须用一对引号('、"、''')括起来	文本信息	name = "张三" greeting = 'Hello'
布尔值(bool)	逻辑值,只有两个:True(真)、False(假)	条件判断、逻辑判断的结果	is_valid=True is_empty=False

1) 变量名的灵活指向机制

在程序运行时,解释器会根据赋值运算符右边的数据,创建相应的值,并让变量名指向这个值。然后,在程序中就可以通过变量名来读取该值,或者让它指向一个全新的值,新值的数据类型可以相同也可以不同。例如:

```
age = 20
age2 = age
age = "22 岁"
print(age)
print(age2)
```

首先,解释器创建整数 20,让 age 指向它。然后,解释器读取 age 的值(20),让 age2 指向新的整数 20。最后,解释器创建字符串"22 岁",让 age 指向这个新字符串。输出结果显示,age 值为"22 岁",age2 值为 20。

从这个例子可以看出,Python 的变量名就像一个灵活的标签,可以随时贴在不同类型的数据上。因此,我们不需要在创建变量时预先定义它的类型,Python 会在运行时根据赋值的内容自动确定其类型,这被称为动态类型。

2) 字符串的灵活操作

字符串是人工智能数据处理中非常关键的类型。它是不可变序列,这意味着一旦创建就无法直接修改其中的某个字符,但支持索引和切片来灵活读取内容。字符串的索引(也称为下标)就是一个编号,书写时放在[]内,它可以是正向索引,即从 0 开始从左向右读取,也可以是逆序索引,即从-1 开始从右向左读取。例如,声明了一条语句:

```
str1 = 'abcd'
```

str1 字符串中各字符的下标如图 3-9 所示,str1[0]与 str1[-4]均表示字符串中的第一个字符'a'。

str1	a	b	c	d
正向索引	0	1	2	3
反向索引	-4	-3	-2	-1

图 3-9　字符串的下标

切片也是字符串等序列的一个灵活操作特性。通过切片访问一定范围内的字符串元素，并且产生一个新的字符串。切片的语法：

```
[起始下标:结束下标:步长]
```

切片得到的子序列不包括结束下标，如果步长为 1，可以省略冒号和步长；如果起始下标为 0 或结束下标为末尾，可以省略下标值但保留冒号。

下面以 2 个学号 student_a 与 student_b 为例，观察对学号字符串的索引、切片前 6 位，判断是否为同一专业，以及索引、成员运算等应用。

```
student_a = "1001152001"
student_b = "1001152101"
# --- 1. 切片提取信息 ---
major_a = student_a[0:6]                    # 提取前 6 位: 专业代码
major_b = student_b[:6]                     # 提取前 6 位,起始值默认为 0,直到下标 6(不包括 6)
# --- 2. 逻辑判断 ---
print(f"是否同一专业: {major_a == major_b}")   # 同一专业用 True 表示,不同专业用 False 表示
# --- 3. 更多字符串操作 ---
last_two_a = student_a[-1]                  # 反向索引,访问字符串最后 1 位
print(f"学生 A 的尾号:{last_two_a}")
has_five_a = '5' in student_a               # 成员运算符 in 查找字符串是否包含特定字符
print(f"学生 A 的学号里有数字'5'吗?{has_five_a}")
```

运行结果为：

```
是否同一专业: True
学生 A 的尾号:1
学生 A 的学号里有数字'5'吗?True
```

Python 中，使用 print 函数输出一行信息，print 的用法见 3.3.5 节。

2. **复杂数据类型**

处理实际问题时，数据往往是成组出现的。例如，一个班级有 30 名学生，一位学生的信息包含姓名、性别、班级等属性。如果为每位学生的每个属性都定义一个变量，显然难以控制。这时，我们需要复杂数据类型，也被称为容器类型（或组合数据），包括列表、字典、元组、集合等，它们提供了组织和管理数据的高级能力。

为了更好地理解复杂数据类型特点及应用，下面展示一些代码示例。

1）列表

列表是可变的有序序列，就像一个可以随意涂改的记事本。列表用方括号[]表示，里面的元素用逗号分隔。因为列表是有序的，所以每个元素有自己的位置，也就是索引。与字符串一样，列表也支持索引、切片等操作。

下面代码演示了列表的常见应用，包括增、删、改、查等操作。

```
students = ['小明','小刚','小军','小华']
print("原始名单:", students)
students.append('小丽')                    #使用 append()方法,在列表末尾添加学生'小丽'
students.remove('小明')                    #使用 remove()方法,在列表中删除学生'小明'
students[0] = '小红'                       #通过索引赋值,把第 1 个位置的元素改为 '小红'
print(f"第 2 位学生:{students[1]}")        #通过索引访问,查看第 2 个位置的元素
print(f"前 2 位学生:{students[0:2]}")      #通过切片,批量找到前 2 个位置的元素
print(f"最终名单:{students}")
```

运行结果为：

```
原始名单: ['小明', '小刚', '小军', '小华']
第 2 位学生:小军
前 2 位学生:['小红', '小军']
最终名单:['小红', '小军', '小华', '小丽']
```

2) 元组

元组与列表非常相似,也是一个有序序列,支持索引和切片等操作。区别是,元组是不可变的,一旦创建就无法修改、添加、删除元素,就像一本只能阅读的、不可涂改的书。元组用圆括号()表示,里面的元素用逗号分隔。例如：

```
scores = (85, 92, 78)
```

3) 字典

字典是 Python 中非常强大的数据结构,就像是一本通讯录。它没有顺序,不能通过索引来查找,而是通过一个唯一的“键”(key)来快速定位和存储对应的“值”(value)。字典用花括号{}表示,里面的内容为一组键值对(key:value)集合。其中,键必须是不可变且唯一的,就像通讯录里的名字不能重名,名字本身也不能变。键可以是字符串、数字或元组。值是键对应的具体数据,可以是任意类型,而且允许重复。

字典通过键来访问、修改或删除数据。下面以记录“小华”的个人信息为例,演示一个字典的创建、查询、修改和添加。

```
# --- 1. 字典的创建 ---
student_info = {'name': '小华', 'age': 12, 'grade': '六年级'}    #键值对用冒号分隔,每组之间用
                                                                 #逗号分隔
# --- 2. 查: 通过"键"快速访问 ---
print(f"学生姓名: {student_info['name']}")                       #输出: 小华
print(f"学生年龄: {student_info['age']}")                        #输出: 12
# --- 3. 改: 直接修改值 ---
student_info['age'] = 13                          #student_info 中键为'age'的值改为 13
print(f"修改后的年龄:{student_info['age']}")      #输出: 13
# --- 4. 增: 添加新的键值对 ---
student_info['hobby'] = '编程'                    #如果不存在'hobby'键,字典会添加该键值对
print("添加爱好后的信息:", student_info)
# --- 5. 删: 删除指定的键值对 ---
del student_info['grade']                         #使用 del 语句,通过键删除该键值对
print("删除年级后的信息:", student_info)
```

运行结果为：

```
学生姓名：小华
学生年龄：12
修改后的年龄:13
添加爱好后的信息：{'name': '小华', 'age': 13, 'grade': '六年级', 'hobby': '编程'}
删除年级后的信息：{'name': '小华', 'age': 13, 'hobby': '编程'}
```

通过这个例子可以看到，字典非常适合用来存储结构化的信息（如个人信息、配置参数等），这样我们可以通过有意义的名字来管理数据。

4）集合

集合是一个无序且不包含重复元素的容器，它相当于一个只有键没有值的字典，或者是一个没有编号的列表，没有顺序，不能通过索引访问元素。集合中的元素必须是不可变的，如数字、字符串等。在人工智能通识应用中，集合常用于数据清洗，例如自动去除重复的训练数据；或者进行高效的成员检测，例如快速判断某条数据是否存在。通过集合，我们可以轻松地完成数据的查找、新增、删除，以及不同数据集之间的数学运算，例如找交集。

下面以学生的爱好管理为例，演示如何用集合来管理学生的爱好，展示集合的自动去重和快速查找等特性。

```
# --- 1. 集合的创建与自动去重 ---
hobbies = {'编程', '阅读', '游泳', '编程'}    # 集合会自动保留一个'编程'
print(f"初始爱好：{hobbies}")                 # 因为无序，输出顺序可能不同
# --- 2. 增：添加新的兴趣 ---
hobbies.add('绘画')                           # add 方法添加元素
print(f"添加绘画后：{hobbies}")
# --- 3. 删：移除某个兴趣 ---
hobbies.remove('游泳')
print(f"移除游泳后：{hobbies}")
# --- 4. 查：快速判断成员 ---
has_reading = '阅读' in hobbies               # in 是成员运算符，判断元素是否存在
print(f"该学生喜欢阅读吗？{has_reading}")
# --- 5. 集合运算：比较两个学生的共同点 ---
student_b_hobbies = {'编程', '篮球', '音乐'} # 另一位学生的兴趣
common = hobbies & student_b_hobbies          # & 在集合运算中是交集，即两个集合中都有的元素
print("两人共同的爱好是：", common)
```

运行结果为：

```
初始爱好：{'编程', '阅读', '游泳'}
添加绘画后：{'编程', '阅读', '游泳', '绘画'}
移除游泳后：{'编程', '阅读', '绘画'}
该学生喜欢阅读吗？True
两人共同的爱好是：{'编程'}
```

5）复杂数据类型的内存管理与引用机制

为了提高内存效率，当我们将列表、字典等复杂数据类型的值赋给变量时，Python 并不复制整个数据，只是让变量指向该值在内存中的“地址”，称为“引用”。因此，多个变量名可以指向同一个值。例如：

```
scores = [85, 92, 78]
scores2 = scores
scores2.append(80)      #列表 scores2 增加一个值 80
print(scores)
print(scores2)
scores2 = [100, 90]     #对 scores2 重新赋值,使其指向一个新的列表地址,但此时 scores 不会变
print(scores)
print(scores2)
```

运行结果为：

```
[85, 92, 78, 80]
[85, 92, 78, 80]
[85, 92, 78, 80]
[100, 90]
```

首先,解释器创建列表[85,92,78],让 scores 指向它。然后解释器让 scores2 也指向 scores 指向的同一个列表。再通过 append 方法向 scores2 增加一个新的元素 80。输出结果显示,scores 与 scores2 的值都是[85,92,78,80],这说明 scores 与 scores2 指向的是同一个内存空间。

最后,scores2=[100, 90],是对 scores2 重新赋值的结果,让 scores2 指向列表[100, 90]。输出结果显示,scores2 变为新的值,而 scores 不变,两个变量分别指向了不同的数据对象。

▶ 3.3.4 运算符与表达式

运算符与表达式是实现数据计算与逻辑控制的核心工具。运算符用于执行算术计算、比较判断及逻辑组合,而表达式则是由运算符和操作数(如变量或常量)组合而成的能够产生值的代码片段。Python 中主要的运算符类型与表达式示例如表 3-5 所示。

表 3-5 Python 中主要的运算符类型与表达式示例

运算符类型	说　明	常见用途	表达式示例
算术运算符(+,-,*,/,**,//,%)	+-*/：分别对应加、减、乘、除运算 **：幂运算 //：整除(商的整数部分) %：模运算(两个数相除的余数)	基本的数学数值计算,如计算面积、矩阵运算、取模	area=3.14*r**2 avg=(a+b)/2 grp_id=num%3
比较运算符(==,!=,>,<,>=,<=)	比较两个值的大小或判断是否相等,结果返回布尔值(True 或 False)	判断是否满足条件,如判断成绩是否合格	if score>=60: is_pass=True
逻辑运算符(and,or,not)	and(与)：判断两个条件是否同时成立 or(或)：判断任意一个条件是否成立 not(非)：取反,True 变为 False,False 变为 True	组合多个条件,构建复杂的逻辑判断关系,用于流程控制中的条件	if x>0 and y<1: flag=not flag
赋值运算符(=,+=,-=)	保存计算结果 +=：加法赋值(a+=b 等同于 a=a+b)	计算结果存储到变量,如计数器更新、累加求和	w+=0.01*grad count=count+1

▶ 3.3.5　程序的输入与输出

在编程中，程序的输入与输出(I/O)是人与计算机进行交互的基础。程序需要通过输入来获取数据，通过输出来展示结果。本节简单介绍 Python 的输入与输出函数。读者在后续编写程序中，可以根据需要再深入掌握这些函数的更多灵活用法。

1. print 函数实现输出

Python 使用内置 print 函数实现在屏幕上显示信息，默认输出内容后自动换行。要输出的内容写到圆括号中，可以是数值、字符串等任何类型的值、变量，或者组合。print 函数非常灵活，可以通过不同的参数来控制输出的格式。表 3-6 展示了它的主要用法。

表 3-6　print 函数的主要用法及示例

用　法	说　明	示例代码	输出结果
输出字符串	输出一行文本内容	print("您好!")	您好!
输出变量	输出变量的值	name="小明" print(name)	小明
输出多个内容	用逗号分隔要输出的多个内容。注意：输出时，每个内容之间有一个空格	name="小明" age=18 print(name, "今年", age, "岁")	小明今年 18 岁
格式化输出(f-string)	用花括号括起要输出的变量名，可以更清晰地在输出字符串中嵌入变量	name="小明" age=18 print(f"{name}今年{age}岁")	小明今年 18 岁
控制分隔符(sep)	sep 参数指定多个内容之间的分隔符，默认是空格	print("A", "B", "C", sep="-")	A-B-C
控制结束符(end)	end 参数指定输出结束时的字符，默认是换行符\n	print("Hello", end="") print("World")	HelloWorld

2. input 函数实现输入

Python 使用内置 input 函数来获取用户的输入。当用户在键盘上输入一行文本，按下 Enter 键后，该函数返回该行字符串(不包括末尾的换行符)。即使用户输入的是数值，也会被当作字符串来处理。input 函数可以接受一个参数，为了提高程序的可读性，使用 input 函数时需要告诉用户应该输入什么内容。表 3-7 展示了它的主要用法。

表 3-7　input 函数的主要用法及示例

用　法	说　明	示例代码	输入示例	变量类型
基本输入	获取用户输入的字符串	name=input()	小明	name 是字符串类型
带提示信息	在获取输入前显示提示信息	age=input("请输入你的年龄：")	18	age 是字符串类型
输入后转换类型	用 int()或 float()等函数转换	num=int(input()) age=input() age=int(age)	5 20	num 是整型 age 是字符串类型 age 被转换为整型

▶ 3.3.6　Python 程序结构

1. 行与语句

Python 程序中，一个物理行默认表示一个逻辑行，也称为一条语句。

以字符 # 开始直到行尾的内容表示注释，程序读到这里并不执行，注释部分起到提高代码

可读性的作用。

如果一个表达式包含在圆括号()、方括号[]、花括号{}内,可以分成多行书写,并且每行末尾允许加注释以辅助说明。例如:

```
week = ['Sunday','Monday','Tuesday',                    #0,1,2
'Wednesday','Thursday','Friday','Saturday']             #3,4,5,6
print(week)
```

运行结果为:

```
['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
```

2. **代码缩进**

缩进是 Python 语法的重要组成部分,用于界定代码块的逻辑层次。同一层级的代码块必须保持相同的缩进量,通常建议使用 4 个空格作为一级缩进标准。若缩进不一致,最常见的是引发程序的 IndentationError 异常,出现错误提示:

```
IndentationError: unexpected indent
```

在条件与循环语句中,不正确的代码缩进会改变代码的逻辑执行路径,导致运行结果不符合预期。

3. **条件与循环**

程序是由一行行的语句构成的,就像一篇文章由一个个句子组成。但仅仅把句子罗列起来,并不能形成一篇有逻辑、有思想的文章。为了让程序能够根据情况作出判断,并能高效地处理重复任务,我们需要学习两种最基本的程序逻辑结构:条件结构和循环结构。

1) 条件结构

条件结构通常使用 if、elif、else 等关键字,让程序在某个路口具备了选择的能力。它就像十字路口的交通信号灯,根据条件指引程序选择对应的代码分支,从而实现智能决策。例如,根据今天的天气,决定出门做什么,Python 代码如下:

```
weather = "下雨"                          #用一个变量 weather 来存储今天的天气
#程序开始判断
if weather == "下雨":
    print("记得带伞!")                    #如果天气变量等于"下雨",就执行这行代码
else:
    print("天气不错,尽情享受阳光")        #否则就执行这行代码
```

2) 循环结构

循环结构则赋予了程序重复执行某项任务的能力。可以使用 for 循环遍历一个列表中的所有元素,也可以使用 while 循环在满足某个条件时持续工作。例如,老师点名,需要依次念出班级里每位同学的名字。Python 代码示例如下:

```
students = ["小明", "小红", "小刚"]        #一个列表,里面存放了所有同学的名字
for name in students:                      #程序开始循环
    print(name + "同学,到了吗?")           #对于列表里的每位同学的名字,都执行下面的代码
```

运行结果：

```
小明同学,到了吗?
小红同学,到了吗?
小刚同学,到了吗?
```

3.4　Python 数据处理库与可视化初探

在人工智能的基础架构中,数据处理与可视化占据着核心地位。数据是人工智能算法的基础对象,数据可视化是理解数据特征、挖掘潜在规律的关键手段。本节简要介绍 Python 语言在数据处理与可视化方面的基本应用。读者在本节可以学习辨析标准库与第三方库概念,了解核心数据科学库的入门操作,体验数据的读取、预处理及可视化流程,建立利用 Python 解决实际数据问题的初步认知。

▶ 3.4.1　标准库与第三方库的概念

Python 语言之所以功能强大,很大程度上来源于其有着丰富的代码库支持。了解这些库的分类,是掌握 Python 编程的第一步。

1. **标准库**

标准库是指随 Python 安装包一起发布的、内置的模块集合,无须额外安装即可直接使用。标准库提供了操作系统接口、文件管理、文本处理等基础功能,保障了 Python 程序运行的最低环境需求,是 Python 语言的“自带装备”。这些功能会封装成函数。函数是指一段封装好的、具有特定功能的代码块,程序可以通过调用它来重复执行这个功能。

对于一些极为基础和常用的功能,例如,求和函数 sum(),Python 解释器在启动时已将其加载到内存中,可以直接在代码中调用。例如,sum([1,2,3])返回 6。

然而,标准库中的功能并非全部默认加载到内存中,而是根据程序需求按需引入。对于一些功能特定、非核心的函数,如计算正弦值的 sin()函数,它们被归类在 math(数学)模块中,需要先通过 import math 语句将该模块引入当前的代码环境,随后才能通过 math.sin()的方式调用功能。这种按需加载的机制,既保证了编程的高效性,又避免了不必要的内存占用。表 3-8 展示了常用的内置函数及标准库函数的示例。

表 3-8　Python 常用标准库的函数示例

功能描述	函数示例	说　明	所属模块	函数应用示例
计算长度	len(obj)	返回对象(如列表、字符串)的长度	内置函数	length=len("Hello")
基础求和	sum(iter)	对序列 iter 的所有元素求和	内置函数	total=sum([1, 2, 3])
数值计算	sin(x)	计算 x 弧度的正弦值	math	import math result=math.sin(math.radians(90))
生成随机整数	randint(a, b)	生成 a 到 b 之间的随机整数	random	import random num=random.randint(1, 10)
获取系统时间	time()	返回当前时间戳	time	import time current=time.time()

续表

功能描述	函数示例	说　明	所属模块	函数应用示例
操作系统接口	os.getcwd()	获取当前工作目录的路径	os	import os path=os.getcwd()
文件管理	open	打开一个存在的文件,返回文件对象	内置函数	file=open("data.txt", "r")
文本处理	re.search()	在字符串中搜索匹配正则表达式的第一个位置	re	import re match=re.search(r"\d+", "abc123def")
帮助	help(函数名)	查看函数的帮助文档	内置函数	help(print)

2. 第三方库

第三方库是由全球开发者社区编写并发布的、非 Python 官方内置的扩展库。这些库通常针对特定领域提供了高度专业化的工具,如数据分析、机器学习等。第三方库需要用户使用专门的安装工具(如 pip)来进行下载、安装和管理,它们极大地拓展了 Python 的应用边界。

1) 安装第三方库

pip 命令通常在终端(Terminal)或命令提示符(CMD)中运行。基本的安装语法是:

```
pip install <库名称>
```

执行后,pip 会自动下载并安装指定的库及其所有依赖项。

在不同的 Python 开发环境中,使用 pip 的方式可能不同。

① Jupyter Notebook。在代码单元格中,需要在 pip 前面加上一个感叹号"!"。例如:

```
!pip install pandas
```

② Anaconda。推荐使用其自带的包管理工具 conda 来安装库。例如:

```
conda install pandas
```

③ 集成开发环境。许多 IDE(如 PyCharm、VS Code)都提供了图形化的包管理界面。通常在设置/首选项中,找到 Project Interpreter(Python 解释器),单击+号搜索并安装所需的库。

2) 使用第三方库

安装完成后,使用第三方库与标准库类似。首先通过 import 语句导入第三方库,即可调用该库中的函数和对象。这种机制保证了 Python 核心环境的轻量化,又能根据项目需求灵活地扩展功能。表 3-9 展示了常用的第三方库及其示例。

表 3-9　Python 常用的第三方库及其示例

功能描述	安装库名	说　明	函数应用示例
数据分析	pandas	提供高性能、易用的数据结构和数据分析工具	import pandas as pd df=pd.read_csv("data.csv")
科学计算	numpy	提供强大的多维数组对象和数学函数库	import numpy as np arr=np.array([1, 2, 3])

续表

功能描述	安装库名	说　明	函数应用示例
数据可视化	matplotlib	提供创建静态、动态和交互式可视化的绘图库	import matplotlib.pyplot as plt plt.plot([1, 2, 3], [4, 5, 6])
机器学习	scikit-learn	提供简单高效的数据挖掘和数据分析工具	from sklearn.linear_model import LinearRegression model=LinearRegression()

3）查看第三方库

查看已安装的库，最常用的是在终端、代码、IDE设置中查看，如表3-10所示。

表3-10　查看当前环境的已安装库

方　法	适用场景	示例命令
终端或命令提示符	查看所有已安装的库	pip list
	查看特定库信息	pip show numpy
Jupyter Notebook	查看所有已安装的库	!pip list
	查看特定库信息	!pip show numpy
Python代码	快速检查单个库的版本	import numpy print(numpy.__version__)
IDE界面	图形化查看	设置→项目→Python解释器

4）更新第三方库

随着时间的推移，库的维护者会不断发布新版本。如果发现当前安装的库版本过旧（例如出现版本过低警告），或者想要最新的特性，可以使用--upgrade参数进行升级。语法为：

```
pip install --upgrade 包名
```

如果一个库不再使用，或者由于环境冲突需要将其移除，可以使用pip uninstall命令。语法为：

```
pip uninstall 包名
```

5）依赖关系与版本冲突问题

在升级或卸载第三方库时，可能会遇到依赖关系与版本冲突问题。

① 依赖关系。很多库并不是独立工作的，它们需要信赖其他库的功能，这种关系称为“依赖”。例如，Pandas库在进行某些计算时依赖NumPy库。因此，在升级Pandas时，安装工具可能会检测到现有NumPy版本太低，从而要求同时升级其依赖的库。

② 版本冲突。不同项目对库版本的要求可能存在不一致，产生冲突。例如，项目A需要NumPy 1.20，而项目B需要NumPy 2.0，如果只安装一个版本的NumPy，就无法同时满足两个项目。因此，建议在实际开发中，为每个项目创建独立的虚拟环境，相当于为每个项目分配一个独立的工作空间，它们各自拥有所需特定版本的库，互不干扰。

▶ 3.4.2　数据科学核心库简介

在Python的第三方库中，有三个库被公认为是数据科学领域的“三剑客”，它们是支撑人工智能应用构建的关键支柱，也是进行数据处理、分析与建模不可或缺的核心基础。

1. NumPy

NumPy 是 Python 科学计算的基础包,它提供了高效的多维数组对象以及用于处理这些数组的工具,能够以极高的速度完成大规模矩阵运算等复杂数学计算。

例如,要处理一组学生的成绩,利用 NumPy 对不及格的成绩(<60)进行提分(加 10 分),并筛选出成绩优秀(≥90)的学生信息。参考代码如下:

```
import numpy as np
#1. 生成数据
names = np.array(['小明', '小刚', '小红', '小华', '小军'])          #一组学生的姓名
scores = np.array([55, 90, 58, 88, 92])                          #对应学生的成绩
print(f"原始成绩: {scores}")
#2. NumPy 运算: 给所有不及格(<60)的同学加 10 分
scores[scores < 60] += 10
print(f"加分后成绩: {scores}")
#3. NumPy 筛选: 找出成绩大于或等于 90 分的同学索引
top_indices = np.where(scores >= 90)      #np.where 返回满足条件的元素下标
#根据索引找出名字和分数
print("优秀学生: ", names[top_indices],",分数: ", scores[top_indices])
```

运行结果:

```
原始成绩: [55 90 58 88 92]
加分后成绩: [65 90 68 88 92]
优秀学生: ['小刚' '小军'] ,分数: [90 92]
```

要读懂这段代码,需要注意 NumPy 的三个核心要点。

① 多维数组对象(ndarray)是 NumPy 中特有的数据类型,是一个专门用来存储同质数据的容器。所有数据在内存中都按顺序排列,可以通过索引(也称为下标)来访问数组中的值(也称为元素)。例如用 array 保存一组学生的姓名,另一个 array 保存对应学生的成绩,两者通过相同的下标来对应姓名与成绩。

② 整体运算高效。NumPy 的一个显著优势是不需要通过循环去逐个访问元素。整个数组可以直接做加、减、乘、除运算,它会自动将运算应用到每一个元素。这种批量处理方式称为向量化操作,其运算效率通常远高于普通的逐个处理方式。

③ []的含义。用一对[]将输出结果括起来,表示这是一个数组。[]里面可以有 0 到多个值,0 个值表示空数组。筛选出来的下标是一个数组,根据这些下标查找到的结果也是一个数组。

2. Pandas

Pandas 是基于 NumPy 构建的数据分析工具库,它提供了类似于 Excel 表格的核心数据结构,能够方便地对结构化数据进行读取、清洗、转换和合并,是进行数据预处理和探索性分析的必备工具。

例如,要处理一组学生的成绩,计算出平均年龄(保留 2 位小数),并筛选出成绩优秀(≥90)的学生姓名与成绩。参考代码如下:

```
import pandas as pd
import numpy as np
# --- 生成 NumPy 数据 ---
```

```
names = np.array(['小明', '小刚', '小红', '小华', '小军'])
scores = np.array([55, 90, 58, 88, 92])
# --- 1. 创建 DataFrame ---
df = pd.DataFrame({ '姓名': names, '成绩': scores})   #创建二维表,包含姓名与成绩列
print(" --- 学生信息 --- ")
print(df)
# --- 2. 计算平均成绩 ---
average_score = df['成绩'].mean()                     #Pandas 中的 mean()是求平均值函数
print(f"平均分: {average_score:.2f}")                 #.2f 表示输出值保留 2 位小数
# --- 3. 筛选出成绩大于或等于 90 分的同学 ---
top_students = df[df['成绩'] >= 90]
print(" --- 优秀学生 --- ")
print(top_students)
```

运行结果为：

```
 --- 学生信息 ---
   姓名  成绩
0  小明  55
1  小刚  90
2  小红  58
3  小华  88
4  小军  92
平均分: 76.60
 --- 优秀学生 ---
   姓名  成绩
1  小刚  90
4  小军  92
```

要读懂这段代码，需要掌握 Pandas 的以下三个核心要点。

① DataFrame 数据框。这是 Pandas 中最常用的数据结构，它就像是一个带标签的二维表格，类似于 Excel 表，定义时用字典来表示出每一列的列名及值，如{ '姓名': names, '成绩': scores}表示第 1 列是姓名、第 2 列是成绩。从输出结果观察 df，其左边一列是索引（自动生成的 0，1，2…），顶部一行是列名（如姓名、成绩）。NumPy 数组只有索引（下标）；相比之下，DataFrame 让每一列数据都有了具体的含义，这样数据更直观，可读性更强。

② 列操作与便捷统计。在 Pandas 中，可以通过列名直接获取某一列的数据。如 df['成绩'] 获取成绩列数据后，直接调用统计方法，如 mean()求平均值，不需要写循环去逐个累加后求平均，Pandas 会自动快速完成计算。

③ 条件筛选。Pandas 提供了非常人性化的筛选方式，使用时不需要关心数组的具体下标。可以直接写条件来过滤数据。例如 df[df['成绩'] >=90] 的含义是：在表格 df 中，找出所有“成绩”这一列大于或等于 90 的行。筛选结果依然是一个完整的 DataFrame，保留了原有的列名和索引。

3. Matplotlib

Matplotlib 是绘图库，它能够生成高质量的图表，如折线图、散点图、直方图等，将抽象的数据以图形方式呈现，从而直观地观察数据分布和趋势。

例如，要将所有同学的成绩以折线图显示，并指定相关参数，包括画布尺寸、折线样式、数据点样式、字体、背景等。参考代码如下：

```
import pandas as pd
import matplotlib.pyplot as plt
#1. 创建 DataFrame 数据
df = pd.DataFrame({ '姓名': ['小明', '小刚', '小红', '小华', '小军'], '成绩': [55, 90, 58, 88,
92]})                                          #创建二维表,包含姓名与成绩列
#先设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']   #使用黑体
plt.rcParams['axes.unicode_minus'] = False     #正常显示负号(如果坐标轴有负数)
#2. 绘制成绩折线图
plt.figure(figsize=(10, 5))                    #设置画布大小
#横轴为姓名,纵轴为成绩,用圆点标记数据位置,折线为实线,颜色为蓝色
plt.plot(df['姓名'], df['成绩'], marker='o', linestyle='-', color='blue')
#3.设置标题和标签
plt.title('学生成绩图')                         #图的标题,黑体
plt.xlabel('姓名')                             #横轴标签为姓名,黑体
plt.ylabel('分数', fontproperties='KaiTi')     #纵轴标签为分数,楷体
#4. 在每个点上显示具体分数
for index, row in df.iterrows():               #循环,df 的每一行的索引 index 与行数据 row
    plt.text(row['姓名'], row['成绩'], f"{row['成绩']}")
#显示网格
plt.grid(True, linestyle='--', alpha=0.6)      #背景为网格线: 虚线、半透明
#显示图表
plt.show()
```

运行结果如图 3-10 所示。

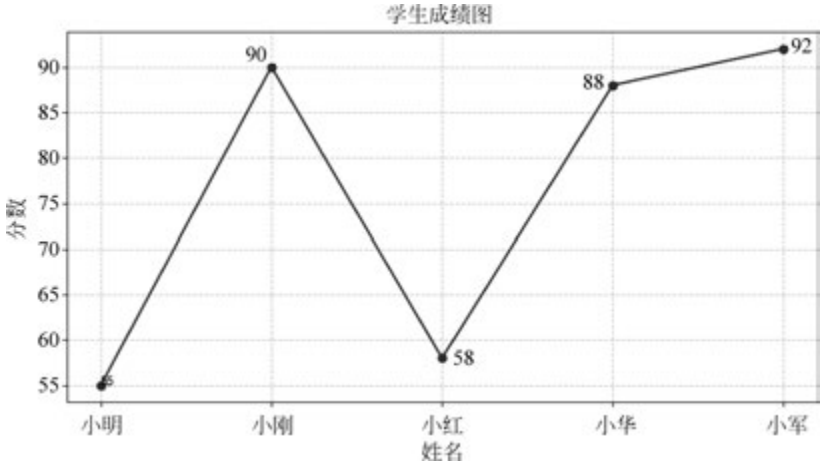


图 3-10　Matplotlib 显示学生成绩图

要读懂这段代码,需要掌握 Matplotlib 的以下五个核心要点。

① 图表画布与尺寸设置。使用 plt.figure(figsize=(宽, 高))来创建画布并设置大小。这里的单位是英寸(1 英寸≈2.54 厘米)。例如,figsize=(10, 5)表示画布宽度为 10 英寸,高度为 5 英寸,控制着图表在屏幕上显示或打印出来的物理尺寸比例。

② 字体与中文显示。Matplotlib 默认不支持中文显示,容易出现乱码。需要通过 plt.rcParams['font.sans-serif']配置全局中文字体,如'SimHei'表示黑体、'KaiTi'表示楷体。针对特定文本(如标题、标签),还可以通过 fontproperties 参数指定单一的特殊字体。同时,为了避免坐标轴负号显示异常,通常需要设置 plt.rcParams['axes.unicode_minus']=False。此外,除了字体样式,还可以设置字号等属性来优化显示效果。

③ 绘图核心参数。plt.plot()是绘制折线图的核心函数，拥有丰富的绘图方法与参数，可以精细控制线条样式。例如：

```
plt.plot(df['姓名'], df['成绩'], marker = 'o', linestyle = '-', color = 'blue')
```

其中，marker 定义数据点的形状，如'o'表示圆点，用来标记具体数据位置。linestyle 定义线条风格，如'-'表示实线。color 定义线条颜色，如'blue'表示蓝色。这些参数组合起来，决定了折线图如图 3-10 所示的外观。

④ 不透明度。用于控制颜色的深浅或透明程度，取值范围在 0(完全透明)到 1(完全不透明)之间。例如，alpha＝0.6 表示 60%的不透明度。常用于网格线或填充区域，使其看起来更柔和，不遮挡主要数据信息。

⑤ 文本标注与循环。为了让图表信息更丰富，通常需要在图中显示具体数值。例如：

```
plt.text(x,y,s)表示在指定坐标(x, y)处添加文本字符串 f"{row['成绩']}"。
```

for 是循环语句，利用 Pandas 的 df.iterrows()方法，可以逐行读取数据，将每个学生的姓名和成绩提取出来，动态地标注在对应的折线点上方。

▶ 3.4.3　数据读取与预处理

在现实世界的应用场景中，收集到的原始数据往往是不完美的，可能包含缺失值、异常值或存在格式不统一的问题。数据质量直接决定了人工智能模型的性能上限，因此在进行人工智能分析之前，必须对数据进行读取与预处理。这一过程涉及将外部存储的文件加载到程序中，更重要的是需要通过清洗和转换，将杂乱无章的原始信息转换为机器学习算法能够理解的高质量数据。对于初学者而言，掌握数据处理的逻辑流程比记忆具体的代码细节更为关键，因为代码实现可以借助 AI 工具辅助完成，但对数据问题的判断和处理思路必须由人来把控。

1. **读文件**

数据读取是数据分析的第一步，其目标是利用 Python 的 Pandas 库将存储在硬盘上的结构化数据文件(如 CSV 或 Excel 文件)加载到内存中并转换为 DataFrame 对象，以便进行后续的操作。在实际操作中，除了指定文件路径外，还需要关注文件的编码格式，这决定了中文字符能否被正确识别。为了让大家更好地理解使用 AI 辅助完成这一任务的方法，下面展示向 AI 工具提问以获取代码。

1) AI 提问示例

> 请帮我编写一段 Python 代码，使用 Pandas 库读取当前目录下名为'data.csv'的 CSV 文件，编码格式为 utf-8，并显示前 3 行数据。

2) AI 回答代码

```
import pandas as pd
# 读取 CSV 文件
df = pd.read_csv('data.csv', encoding = 'utf-8')
# 显示前 3 行数据
print(df.head(3))
```

3) 核心解释

pd.read_csv()是最常用的读取函数,它将文本格式的数据解析成表格。参数 encoding='utf-8' 非常重要,它能防止中文乱码。

df.head(3)则用于快速预览数据的前 3 行,不写数字表示默认预览数据的前 5 行。

4) 运行测试

打开 Excel 或 WPS 表格软件,输入数据,单击文件→另存为,将保存目录设置为与 Python 程序相同的目录,在保存类型中选择 CSV UTF-8(*.csv),文件名为 data.csv,单击"保存",如图 3-11(a)所示。

然后在 Jupyter Notebook 等工具中运行 AI 回答的代码,观察运行结果,如图 3-11(b)所示。

(a) 数据文件保存位置与类型　　(b) 读文件前3行的运行效果

图 3-11　读文件成功示例

注意数据文件与 Python 程序必须在相同目录,否则运行时会出现错误提示:

```
FileNotFoundError: [Errno 2] No such file or directory: 'data.csv'
```

2. 预处理

数据预处理是提升数据质量的关键环节。在实际的人工智能应用中,直接获取的原始数据往往是"脏"数据,存在缺失值、格式不一致或格式错误等问题。例如,数据集中可能存在某些人的年龄信息未填写、数据类型不匹配(如数字被保存为文本格式)等情况,这些问题会严重影响模型的学习效果,进而影响预测精度。

1) 预处理手段

因此,我们需要通过一系列预处理手段来优化数据。主要包括以下五方面。

① 数据清洗。这是最基础也是最重要的一步,目的是让数据可用。主要包括缺失值处理、异常值处理、重复值处理等。例如,针对缺失值,根据实际情况选择填充策略,如使用均值、中位数、众数等值来填充;删除多余的重复值或明显不合理的异常值。

② 数据筛选与切片。通过索引、切片或条件过滤,从海量数据中精准提取出需要的那些行(记录)与列(特征)。

③ 数据转换。将非数值型的标签(如男和女)转换为机器可读的数值编码(如数字 0 和 1),将连续变量(如具体年龄整数)离散化(如青少年、中年、老年)等,使算法能够处理这些特征值。

④ 数据运算。基于初始数据,利用算术运算、统计运算、分组运算等方法,计算出更有价值的新特征(如利用成绩计算总绩点),进而挖掘数据深层的规律。

⑤ 标准化处理。由于不同数据的单位和范围往往存在显著差异,如果直接输入模型,数值大的特征会掩盖数值小的特征的重要性。例如,在分析学生对知识掌握的熟练程度时,"考

试分数”(0～100 分)和“做题用时”(30～90 分钟)相比,10 分的分数差距在数值上远大于做题 1 分钟的用时差距,这将导致模型误判,认为只有分数重要而忽略时间差异。标准化处理是通过将不同量纲的数据映射到统一的尺度范围内,例如:

$$x=\frac{\text{原始数据}-\text{平均值}}{\text{标准差}}$$

将考试得分 10 分与做题用时 1 分钟映射为 1.0 和 0.1,使它们都处于同一个尺度(如[－1,1]),从而确保各个特征在模型面前权重平等,提升算法的收敛速度和准确性。

2) Pandas 的预处理方法

例如,有 2 个 DataFrame 类型的对象,分别存储 2 个班级的学生信息,其中班级 1 学生的“年龄”列有缺失值,且无英语成绩。编写代码,用平均年龄填充缺失的年龄,无英语成绩用 0。并将“性别”列转换为数字,男为 0,女为 1。参考代码如下:

```
import pandas as pd
data1 = {                                    #班级 1 的数据(注意: 小红的年龄缺失)
    '姓名': ['小明', '小红'],
    '性别': ['男', '女'],
    '年龄': [18, None],                      #缺失值
    '计算机': [85, 90]
}
df_class1 = pd.DataFrame(data1)
data2 = {                                    #班级 2 的数据(注意: 多 1 列英语成绩)
    '姓名': ['小霞'],
    '性别': [ '女'],
    '年龄': [ 18],
    '计算机': [95],
    '英语': [99]                             #班级 1 没有英语列,拼接时会出现 NaN
}
df_class2 = pd.DataFrame(data2)
# ======== 数据预处理 ============================
df = pd.concat([df_class1, df_class2], ignore_index = True) #拼接 2 个表
print(f"拼接表:\n{df}")                      #输出拼接后的表,班级 1 的缺失值是 NaN
age_mean = round(df['年龄'].mean(),0)        #使用 mean()方法计算平均值,四舍五入保留 0 位小数
df.fillna({'年龄':age_mean,'英语':0},inplace = True)  #替换缺失值
df['年龄'] = df['年龄'].astype(int)                   #年龄列转换为整数类型
print(f"替换缺失值后: \n{df }")                       #输出替换后的拼接表
df['性别'] = df['性别'].map({'男': 0, '女': 1})       #使用 map 映射函数将文本映射为数值
df['总成绩'] = df['计算机'] + df ['英语']             #计算总成绩
print(f"清洗转换表:\n{df}")                           #输出转换后的拼接表
```

3) 核心解释

① Pandas 的 mean()方法计算年龄列的平均值,目的是用一个合理的数据替换空缺,使数据更完整。由于年龄是整数,所以使用 round 函数对平均值保留 0 位小数(浮点数),再用 astype(int)函数转换为整数类型。

② 填补缺失数据用的是 Pandas 中的 fillna()方法,它把刚才算出的平均年龄,一次性填充到所有缺失的年龄列中,方便后续的统计和分析。默认参数 inplace 值为 False,表示将修改结果生成新的副本,而 inplace＝True 表示直接在原始数据集 df 中修改。

③ 数据转换时,Pandas 中的 map()方法像是一个字典查找工具,它按照定义的规则将文本映射为数字。花括号{'男': 0, '女': 1}是一个字典,写着一组“key:value”映射。对于性别

列,遇到'男'就替换成 0,遇到'女'就替换成 1。

4) 测试运行

运行结果为:

```
拼接表:
   姓名  性别    年龄  计算机    英语
0  小明   男    18.0    85     NaN
1  小红   女    NaN     90     NaN
2  小霞   女    18.0    95     99.0
替换缺失值后:
   姓名  性别    年龄  计算机    英语
0  小明   男    18      85     0.0
1  小红   女    18      90     0.0
2  小霞   女    18      95     99.0
清洗转换表:
   姓名  性别    年龄  计算机    英语     总成绩
0  小明   0     18      85     0.0     85.0
1  小红   1     18      90     0.0     90.0
2  小霞   1     18      95     99.0    194.0
```

3. **异常或问题**

在处理数据时,除了常规的清洗,还必须警惕数据中的异常值或逻辑错误。异常值是指那些明显偏离正常范围的数据点,例如年龄字段出现了 200 岁,或者考试成绩出现了负数。这些异常值如果不被处理,则会严重干扰模型的训练过程,导致预测结果出现巨大偏差。此外,数据格式不统一(如日期格式混乱),也是常见的问题。

例如,筛选年龄大于 120 岁的异常行:

```
abnormal_data = df[df['年龄'] > 120]
#判断是否存在异常
if not abnormal_data.empty:
    print(f"发现异常数据: \n{abnormal_data}")
else:
    print("未发现明显的年龄异常值。")
```

布尔索引,即 df['年龄'] > 120,它会遍历每一行的年龄数据,将其与 120 比较,生成一个由 True 或 False 组成的布尔序列。然后,Pandas 据此筛选,只保留 True 对应的行。

这种检查机制能够帮助我们快速发现数据中的“地雷”,从而可以在分析前将其排除或修正,确保模型的可靠性。

▶ 3.4.4 数据可视化入门

在 3.4.2 节中,我们已经初步了解了 Matplotlib 库并绘制了基础的折线图。在本节中,我们将深入探讨数据可视化的核心价值、常见图形类型及其应用,并结合实际数据集进行实践。

1. **可视化的核心价值**

在科学计算中,我们经常要面对海量的数字表格。这些由行和列组成的原始数据往往难以理解。数据可视化的本质,就是通过图形化的手段,将这些抽象的数字转换为直观的视觉元素。其主要作用体现在以下两方面。

① 化繁为简,直观呈现逻辑关系。通过可视化,将原本隐藏在数字背后的信息,以直观的

图形显现出来。例如，折线图的起伏可以直观地展示时间序列的变化趋势，而散点图的分布则能揭示变量间的相关性。

② 辅助洞察与决策。构建人工智能模型时，数据可视化不仅是展示结果的工具，更是探索数据的利器。它能帮助我们快速捕捉数据中的关键特征，识别噪声与异常，从而为模型的选择、参数的调整提供直观的决策依据。

2. **常用图形类型及适用场景**

在 Matplotlib 等可视化库中，不同的图表类型适用于不同的分析需求。图 3-12 展示了四种最基础且常用的图形。

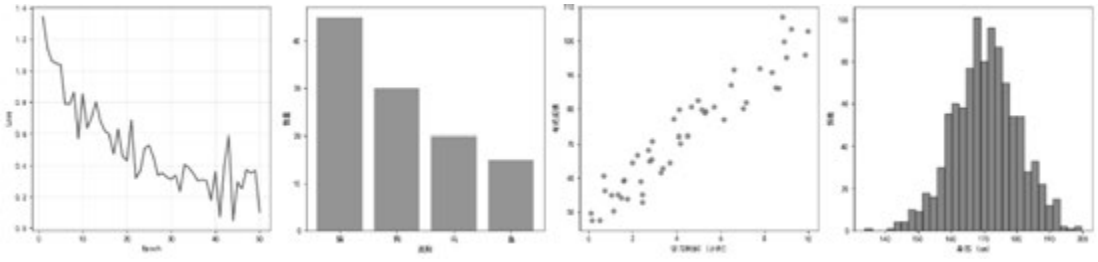


图 3-12　四种常见图形示例

折线图（Line Plot）主要用于展示数据随时间或有序类别变化的趋势。例如，展示某城市一年中每个月的平均气温变化，或某款产品在一天内不同小时的实时销量波动等。

柱状图（Bar Plot）主要用于比较不同类别之间的数值大小。例如，统计并对比不同类别的样本分布数量。

散点图（Scatter Plot）主要用于观察两个连续变量之间的关系。通过点的分布形态，可以初步判断变量之间是否存在线性或非线性关系，这对后续选择模型的类型非常有帮助。

直方图（Histogram）主要用于展示数据的分布情况，特别是频数分布。例如，查看数据是否符合正态分布，这对于数据预处理阶段的特征工程（如标准化、归一化）至关重要。

除上述基础图形外，数据可视化工具箱中尚有若干重要图形，可满足更精细化的分析需求。图 3-13 展示了饼图、箱线图、热力图及词云图形的示例。

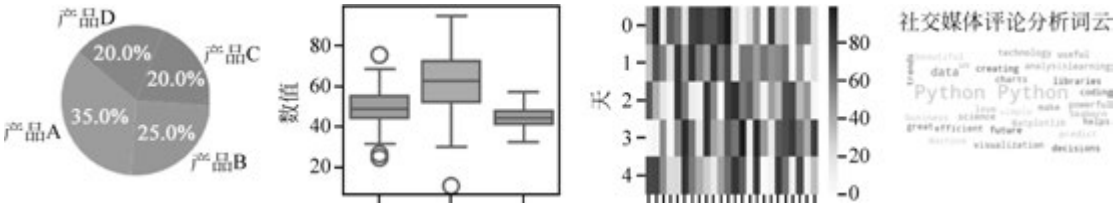


图 3-13　饼图、箱线图、热力图及词云图形示例

饼图（Pie Chart）主要用于呈现整体中各部分所占的比例关系。例如，不同产品线在总销售额中的份额，用户群体在不同年龄段的分布比例等。

箱线图（Box Plot）主要用于展示数据分布特征及识别异常值。它通过五个关键统计量来概括数据的集中趋势与离散程度，即最小值、第一四分位数（Q1）、中位数（Q2）、第三四分位数（Q3）和最大值。

热力图（Heatmap）通过颜色的深浅或渐变来表示数值的大小或密度，适用于揭示矩阵形式数据中的模式与规律。例如，展示用户在网页上的单击行为、不同时间段的活跃度分布，变量之间的相关性矩阵等，其直观的色彩编码使得复杂数据集中的高值区域与低值区域一目了然。

词云(Word Cloud)是一种专门用于文本数据可视化的图形,通过调整词语的字体大小来反映其在文本中出现的频率或重要性。词云常用于社交媒体评论分析、客户反馈挖掘以及新闻热点追踪,能够快速呈现文本主题与关键词。

3. **可视化的建议与注意事项**

在进行数据可视化时,为了确保图表的有效性与专业性,需要注意以下几点。

① 明确目的。在绘图前应明确图表是为了展示趋势、比较差异还是探索关系,根据不同的目的来选择最合适的图表类型。

② 保持真实与客观。可视化必须基于真实数据,避免通过扭曲坐标轴比例、截断纵坐标等手段误导读者对数据形态的判断。诚实是科学研究的基本准则。

③ 注重图表的可读性。确保图表拥有明确的标题、坐标轴标签及图例,让读者能够独立理解图表内容。

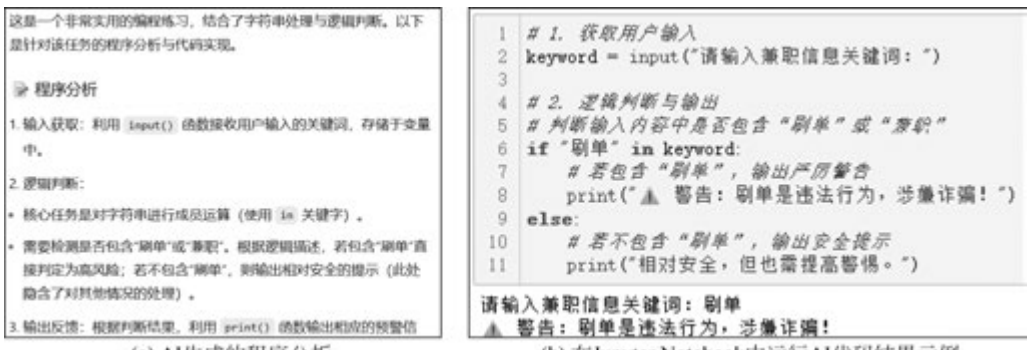
④ 关注数据异常。在绘图过程中,要留意那些异常的分布,这往往是数据质量问题或特殊规律的重要线索。

3.5 实训——程序设计基础

▶ 3.5.1 任务 1:反诈防骗“第一道防线”

为避免陷入“刷单”诈骗事件,编写一个简单的预警程序。要求输入包含“兼职”或“刷单”的内容,程序判断并输出风险提示。如果包含“刷单”,输出“⚠警告:刷单是违法行为,涉嫌诈骗!”,否则输出“相对安全,但也需提高警惕。”

说明:将此提示词交给任意 AI 工具,观察程序分析思路,以及示例代码。运行代码,观察结果,如图 3-14 所示。



(a) AI生成的程序分析　　(b) 在Jupyter Notebook中运行AI代码结果示例

图 3-14　AI 生成程序分析与代码运行示例——刷单诈骗预警程序

▶ 3.5.2 任务 2:校园单车“文明骑行”提醒

为了规范校园单车停放,指定了停放区域为 allowed_zones = ["A", "B", "C"]。要求编写一个程序,输入停放区域编号(zone),如果 zone 在列表里,输出“停放合规,谢谢文明出行!”,否则输出“请将单车停放到指定白线区域内,方便他人。”

类似地,利用 AI 生成程序分析,观察程序的分析思路和示例代码。运行代码,观察结果,如图 3-15 所示。


程序分析

1. 数据定义：建立一个名为 allowed_zones 的列表，作为"白名单"，存储所有合规的停放区域编号（"A"、"B"、"C"）。

2. 输入获取：程序运行时，提示用户输入停放的区域编号，并存入变量 zone。

3. 逻辑判断：核心步骤是使用 in 运算符，程序会检查输入的 zone 是否存在于 allowed_zones 这个名单中。

- 存在：合规。
- 不存在：违规。

4. 输出结果：根据判断结果，打印出对应的提示信息。

(a) AI生成的程序分析

```
# 1. 定义合法停放区域列表
allowed_zones = ["A", "B", "C"]

# 2. 获取用户输入的停放区域
zone = input("请输入您当前的停放区域：")

# 3. 判断并输出结果
# 使用 in 运算符判断 zone 是否在 allowed_zones 列表中
if zone in allowed_zones:
    print("停放合规，谢谢文明出行！")
else:
    print("请将单车停放到指定白线区域内，方便他人。")
```

```
请输入您当前的停放区域：D
请将单车停放到指定白线区域内，方便他人。
```

(b) 在Jupyter Notebook中运行AI代码结果示例

图 3-15　生成程序分析与代码运行示例——校园单车停放区域提醒程序

3.6　习题与思考

一、选择题

1. 关于 Python 语言的特点，下列说法不正确的是（　　）。

A. Python 是解释型语言，代码不需要编译即可运行

B. Python 语法简洁，强制使用缩进来体现代码层次

C. Python 是弱类型语言，变量定义后类型不可改变

D. Python 拥有丰富的第三方库，被称为"胶水语言"

2. 下列变量名中，符合 Python 命名规则且符合 PEP8 编码规范的是（　　）。

A. 2_name　　B. my-score　　C. MyClass　　D. student_name

3. 执行语句 x=5；y=2；print(x//y)后，输出的结果是（　　）。

A. 2.5　　B. 2　　C. 3　　D. 2.0

4. 若要实现从键盘获取用户输入并将其存储在变量 s 中，应使用的函数是（　　）。

A. print(s)　　B. input(s)　　C. s=input()　　D. s=print()

5. 关于 Python 中的注释，下列描述正确的是（　　）。

A. 注释用于解释代码逻辑，程序运行时会执行注释内容

B. 单行注释以//开头，多行注释可以使用/ ** /

C. 单行注释以＃开头，多行注释可以使用三引号'''或"""

D. 注释内容不能包含中文

6. 在数据科学核心库中，用于处理多维数组和矩阵运算的库是（　　）。

A. Pandas　　B. Matplotlib　　C. NumPy　　D. Sklearn

7. 在数据处理项目中，通常用来展示数据趋势和分布图的库是（　　）。

A. numpy　　B. pandas　　C. matplotlib　　D. math

8. 若要从 Pandas 库中读取 CSV 格式的数据文件，应使用的函数是（　　）。

A. pandas.read_excel()　　B. pandas.read_csv()

C. pandas.to_csv()　　D. pandas.open()

9. 以下代码的输出结果是（　　）。

```
a = [1, 2, 3]
b = a
b.append(4)
print(a)
```

A. [1, 2, 3]　　B. [1, 2, 3, 4]　　C. [4]　　D. 程序报错

10. 在 Python 程序结构中,用于根据条件判断选择不同执行路径的语句是(　　)。

A. for 语句　　B. while 语句　　C. def 语句　　D. if...else 语句

11. 关于 AI 辅助编程工具,下列说法正确的是(　　)。

A. AI 生成的代码可以直接使用,无须人工检查

B. AI 工具只能生成 Python 代码,不能解释代码含义

C. AI 工具可以帮助初学者理解报错信息,提供修改建议

D. 使用 AI 工具会阻碍编程能力的提升,应完全避免使用

二、讨论题

1. 在 Python 中,列表和元组都是序列类型,请对比两种类型的特点与区别。讨论在处理校园单车停放区域数据时,为什么选择列表 allowed_zones=["A", "B", "C"]而不是元组?

2. 在编写程序时,可以采取哪些简单的方法来提示用户规范输入,有哪些输出方法能够提高程序的友好度?

3. 请结合日常生活或学习案例(如期末成绩分析),讨论数据可视化相较于单纯的数字表格,有哪些独特的优势?

4. 本章介绍了 AI 辅助编程工具。请谈谈如何正确利用 AI 工具来提升编程学习效率,如何分辨 AI 生成的代码是否存在逻辑漏洞。

5. 利用本章学习的 AI 辅助编程工具,输入以下代码片段,要求 AI 逐行解释这段代码的含义,并根据 AI 的解释,用自己的话总结该段代码的功能:

```
scores = [85, 92, 78, 90]
total = 0
for score in scores:
    if score >= 90:
        total += 1
print(f"优秀人数: {total}")
```

6. 编写程序。制作一个期末成绩“奖学金”预判器,用户输入期末绩点(GPA),判断能否拿奖学金。如果 GPA>=3.9,输出“学霸! 冲击国奖!”;如果 GPA>=3.5,输出“有望获得校级奖学金”;否则输出“继续加油,来年再战!”。可以利用 AI 工具生成提示词并验证。