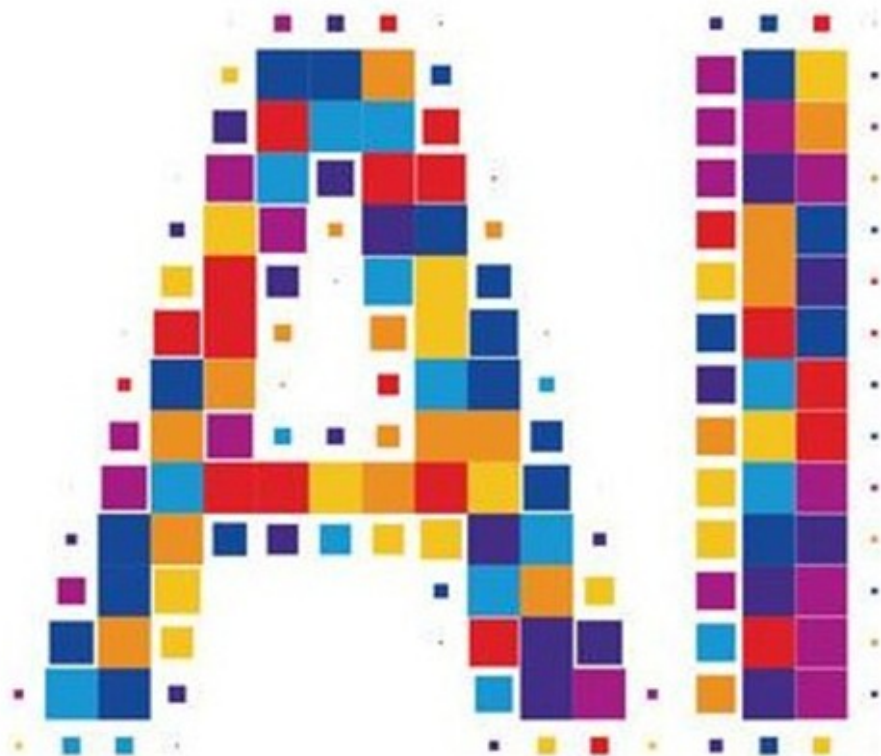


AI时代人人都应该学编程，这本书能让人人都学会编程

香港智友数字科技CEO撰写，东方证券原首席经济学家、浙商银行原行长、重庆富民银行董事长兼行长、武汉大学和复旦大学教授等业界和学界专家联袂推荐



零基礎 用AI學編程

人人都能學會的DeepSeek 和Windsurf編程指南

兰翔 钟磊 等著

从思维到方法，从工具到项目，从原理到实践，手把手教IT零基础的读者学会用DeepSeek和Windsurf等主流AI编程工具实现代码自由

 机械工业出版社
CHINA MACHINE PRESS

目 錄

- [版權信息](#)
- [內容簡介](#)
- [前言 擁抱AI，與時代共舞——AI時代創造者啟程指南](#)
- [第一部分 AI編程入門](#)
- [第1章 AI時代普通人如何學編程](#)
- [1.1 本章學習目標](#)
- [1.2 AI時代普通人為什麼要學編程](#)
- [1.3 普通人能用AI編程做什麼](#)
- [1.4 AI編程給普通人帶來的新機會](#)
- [1.5 結語](#)
- [第2章 DeepSeek編程指南](#)
- [2.1 本章學習目標](#)
- [2.2 什麼是DeepSeek](#)
- [2.3 登錄和使用DeepSeek](#)
- [2.4 DeepSeek的核心能力與應用場景](#)
- [2.5 DeepSeek提示詞](#)
- [2.6 結語](#)
- [第3章 輕鬆準備你的編程環境](#)
- [3.1 本章學習目標](#)
- [3.2 開發環境概述](#)
- [3.3 開發工具的選擇](#)
- [3.4 工具的安裝與使用流程](#)
- [3.5 常見問題與解決方案](#)
- [3.6 結語](#)
- [第4章 用DeepSeek學Python變量和簡單數據類型](#)
- [4.1 本章學習目標](#)

- [4.2 Python解析器](#)
- [4.3 變量](#)
- [4.4 字符串](#)
- [4.5 數字](#)
- [4.6 註釋](#)
- [4.7 結語](#)
- [第5章 用DeepSeek學Python列表](#)
- [5.1 本章學習目標](#)
- [5.2 Python列表](#)
- [5.3 修改、添加和刪除元素](#)
- [5.4 Python列表的高級技巧](#)
- [5.5 通過DeepSeek學習Python索引](#)
- [5.6 結語](#)
- [第6章 用DeepSeek學操作列表](#)
- [6.1 本章學習目標](#)
- [6.2 遍歷整個列表](#)
- [6.3 避免縮進錯誤](#)
- [6.4 創建數值列表](#)
- [6.5 結語](#)
- [第二部分 項目實戰](#)
- [第7章 開發數據分析報告生成系統](#)
- [7.1 本章學習目標](#)
- [7.2 自動化數據分析的優勢](#)
- [7.3 數據分析報告生成系統的功能](#)
- [7.4 具體實現步驟：開發數據分析報告生成系統](#)
- [7.5 常見問題與解決思路](#)
- [7.6 結語](#)
- [第8章 開發自動化日報生成工具](#)
- [8.1 本章學習目標](#)

- [8.2 瞭解自動化日報生成工具](#)
- [8.3 自動化日報生成工具的核心功能](#)
- [8.4 具體實現步驟：構建你的自動化日報生成工具](#)
- [8.5 常見問題與解決思路](#)
- [8.6 結語](#)
- [第9章 開發本地事務提醒助手](#)
- [9.1 本章學習目標](#)
- [9.2 瞭解本地事務提醒助手](#)
- [9.3 本地事務提醒助手的核心功能](#)
- [9.4 具體實現步驟：構建你的本地事務提醒助手](#)
- [9.5 常見問題與解決思路](#)
- [9.6 結語](#)
- [第10章 打造你的第一個網站](#)
- [10.1 本章學習目標](#)
- [10.2 瞭解網站開發基礎](#)
- [10.3 網站開發的核心功能](#)
- [10.4 具體實現步驟：開發你的第一個網站](#)
- [10.5 常見問題與解決思路](#)
- [10.6 結語](#)
- [第三部分 總結與展望](#)
- [第11章 成為AI時代的創造者](#)
- [11.1 本書精華總結](#)
- [11.2 即刻行動指南：將知識轉化為力量](#)
- [11.3 避免常見陷阱](#)
- [11.4 未來成長地圖：繪製你的技能進階藍圖](#)
- [11.5 持續發現需求的方法](#)
- [11.6 致未來的創造者：你的潛力無限](#)
- [11.7 最後的話](#)
- [11.8 結語](#)

- [附錄](#)
- [附錄A 提示技術術語彙編](#)
- [附錄B DeepSeek提示詞](#)
- [作者簡介](#)

版權信息

書名：零基礎用AI學編程：人人都能學會的DeepSeek和Windsurf
編程指南

作者：蘭翔 鍾磊

排版：小閱

出版社：機械工業出版社

出版時間：2025-09-16

ISBN：9787111789703

內容簡介

在AI浪潮席捲而來的今天，編程不再是少數人的專屬技能。本書正是你零基礎邁入AI編程世界的實踐指南，系統講解如何藉助DeepSeek和Windsurf兩大AI編程助手，實現人人都能學會編程的願景。

本書由香港智友數字科技CEO、資深AI編程與數字化運營專家蘭翔領銜撰寫，凝聚了他豐富的開發經驗和AI實戰經驗。本書獲得了東方證券原首席經濟學家、浙商銀行原行長、重慶富民銀行董事長兼行長、武漢大學和復旦大學教授等多位業界與學界的權威專家的高度評價與鼎力推薦。

面對AI時代編程門檻高、學習效率低、實戰應用難的痛點，本書為你提供了一套從思維到方法，從工具到項目，從原理到實踐的系統性解決方案，讓你告別傳統編程的晦澀難懂，輕鬆掌握AI編程的核心技能，實現從零基礎到獨立開發的跨越。

掌握本書內容，你將收穫以下價值：

(1) AI編程核心方法論： 深入解析DeepSeek與Windsurf如何驅動零基礎學習，掌握AI編程的核心思維、提示詞工程原理及人機協作模式。

(2) AI助手高效代碼開發： 學習如何利用AI助手進行需求分析、代碼生成、調試優化，涵蓋Python基礎、數據處理、項目架構等核心技術，大幅降低學習門檻。

(3) AI編程項目實戰技能： 通過數據分析報告生成系統、自動化日報工具、本地事務提醒助手、網站搭建等多個實戰項目，掌握AI在項目開發中的作用、多項目規劃與迭代優化技術，增強獨立開發與問題解決能力。

(4) 個人AI工具生態系統構建： 掌握數據分析、自動化工具、提醒系統等應用開發，搭建個人效率提升的完整解決方案，實現從學習到成果轉化的全流程突破。

跟隨本書，與AI導師DeepSeek和Windsurf並肩作戰，你將輕鬆掌握編程奧秘，不僅能學會寫代碼，還將學會AI時代的思考方式與協作能力。無論你是想轉行IT，提升職場競爭力，還是實現個人創意，這本書都將成為你實現夢想的堅實基石，助你在數字化未來中搶佔先機，成為AI時代的弄潮兒！

前言

擁抱AI，與時代共舞

——AI時代創造者啟程指南

時間的年輪在不停地轉動，不知不覺我們已經來到了AI的時代。

AI，這個曾經只存在於科幻想象中的概念，如今已如空氣般瀰漫在人們生活的各個角落——從智能手機的語音助手，到精準推送的新聞資訊，再到像DeepSeek這樣能夠與人們深度對話、協助創作的大語言模型。在這個技術飛速迭代的時代，AI不再是遙不可及的尖端科技，它正在以前所未有的速度滲透、重塑著我們的工作、學習、生活等方方面面，深刻地改變著世界的運行方式。

為何編寫本書

在AI快速發展的今天，編程不再是專業程序員的專屬技能。隨著DeepSeek等AI編程工具的出現，編程的門檻大大降低，廣大

讀者也能通過AI快速學習和應用編程技能。本書正是基於這一背景，旨在幫助零編程基礎的讀者利用AI工具學習編程，解決工作和生活中的實際問題。

什麼是AI編程？AI編程本質上是一種人機協作的新型學習與開發模式，即通過AI工具輔助人們學習編程知識、理解編程概念、生成和優化代碼。這種模式讓讀者能夠藉助AI的能力快速跨越傳統編程學習中的技術障礙，將重點放在問題解決和創意實現，而非語法細節和底層實現上。

學習編程，尤其是AI編程，本質上是在學習一種新的思維方式和表達語言，一種與智能機器協作、解決問題、創造價值的能力。它不僅關乎效率提升、職業發展，也深刻關聯著我們如何理解智能的邊界，如何在人機協作的新範式下重新定義人類的價值與可能性。

我們注意到，雖然已經出版了大量編程圖書，但大多數圖書要麼過於專業難懂，要麼沒有結合AI工具的優勢。本書結合DeepSeek等AI工具，專門為沒有編程基礎的讀者提供了一條學習編程的捷徑。

本書讀者對象

本書主要面向以下讀者：

- ❑ 零基礎但對編程感興趣的讀者。
- ❑ 希望通過編程提高工作效率的職場人士。
- ❑ 想要了解AI如何改變編程方式的讀者。
- ❑ 對DeepSeek等AI工具感興趣的用戶。

□ 希望快速掌握實用編程技能的學生。

無論你是完全沒有編程經驗的初學者，還是希望瞭解AI編程新方法的有編程經驗的程序員，本書都能為你提供有價值的指導和參考。

本書內容特色

本書具有以下特色：

□ 零基礎友好：不需要讀者有任何編程基礎，從最基本的概念講起。

□ AI輔助學習：全面結合DeepSeek、Windsurf等AI工具，降低學習難度。

□ 實用導向：注重解決實際問題，而非抽象理論。

□ 案例豐富：提供大量實際案例，幫助讀者理解和應用。

□ 循序漸進：內容由淺入深，確保讀者能夠跟上學習節奏。

□ 中文編程友好：特別關注中文環境下的編程需求和問題。

如何閱讀本書

本書共11章，分為三個部分：

第一部分（第1~6章） AI編程入門

我們從編程環境搭建開始，通過DeepSeek學習Python的基礎知識，包括變量、數據類型、列表操作等，對這些基礎概念的學習可為後續的項目開發打下堅實的基礎。特別值得一提的是，我

們選擇了Windsurf作為主要開發工具，並藉助DeepSeek的能力，讓學習過程更加智能和高效。

第二部分（第7~10章） 項目實戰

在掌握了基礎知識後，我們開始接觸更實用的工具和方法。通過DeepSeek+Windsurf開發數據分析報告生成系統和自動化日報生成工具這兩個項目，介紹瞭如何將編程知識應用到實際問題中，並體驗了現代開發工具如何提升開發效率。然後，我們挑戰了更復雜的開發任務——開發本地事務提醒助手與打造你的第一個網站，還介紹瞭如何處理實際項目中的各種挑戰。

第三部分（第11章） 總結與展望

在這部分，我們回顧了本書的學習旅程，總結了AI編程的關鍵收穫和方法論。我們探討了如何持續提升編程技能，分享了對AI編程的思考與感悟，並展望了AI編程對未來職業技能發展的影響。這部分不僅是對前面內容的回顧，也是讀者繼續深入學習的指南。

建議按照以下方式閱讀本書：

□ 先通讀AI編程入門部分，掌握必要的編程概念和DeepSeek的使用方法。

□ 根據自己的興趣和需求，選擇項目實戰部分的相關章節進行學習。

□ 在實踐中遇到問題時，參考常見問題與解決思路的相關內容。

□ 動手實踐每個章節的示例代碼，並嘗試修改和擴展。

記住，編程是一項實踐技能，光看不練是無法真正掌握這項技能的。因此，請務必動手實踐本書中的示例，多多練習。

致謝

首先，感謝DeepSeek團隊開發出如此優秀的AI工具，為我們學習編程提供了新的可能性。

其次，感謝所有參與本書編寫、審校和測試的同事們，你們的專業和熱情使本書的出版成為可能。

最後，感謝每一位選擇本書作為自己學習AI編程的夥伴的讀者。希望本書能夠幫助你在AI時代掌握編程技能，發現編程的樂趣，並解決實際問題。

祝你學習愉快！

第一部分

AI編程入門

本書的第一部分介紹Python編程的基礎知識，以及如何藉助DeepSeek等AI工具加速學習過程。這些基礎概念不僅適用於Python，也適用於大多數編程語言，將在你的整個編程生涯中發揮重要作用。

第1章探討AI時代普通人學習編程的意義和價值，分析AI時代普通人為什麼要學編程、普通人能用AI編程做什麼，以及AI編程給人們帶來了哪些新機會。

第2章介紹DeepSeek編程指南，首先介紹什麼是DeepSeek、登錄和使用DeepSeek，然後詳細介紹DeepSeek的核心能力與應用場景，以及使用DeepSeek進行交互的高效提示詞示例。

第3章介紹如何通過DeepSeek的指導快速搭建Python編程環境，包括開發環境概述、開發環境選擇、工具的安裝與使用流程，以及常見問題與解決方案，讓你快速進入編程實踐。

第4章講解如何使用DeepSeek學習Python變量和簡單數據類型，包括Python解析器的相關內容、如何在變量中存儲信息，以及如何處理字符串和數字等基本數據。

第5章和第6章介紹如何使用DeepSeek學習Python列表的相關知識。第5章講解通過DeepSeek學習列表的基本概念和操作，第6章則深入探討如何使用DeepSeek高效操作列表。

通過本部分的學習，你將掌握使用DeepSeek學習Python編程的基礎知識的方法，並學會如何藉助AI工具提高學習效率和編程能力，為後續的項目實踐奠定堅實基礎。

第1章

AI時代普通人如何學編程

在本章中，你將探索AI時代為什麼普通人也能且應該學習編程。你將瞭解編程如何從專業程序員的專屬技能轉變為每個人都可以掌握的實用工具。你將通過具體的實例瞭解編程如何解決工作和生活中的實際問題，實現從數據處理到工作流程自動化。

本章還將介紹DeepSeek等AI工具如何改變編程學習方式，降低學習門檻，幫助你避開傳統學習中的痛點。你將發現，AI時代不僅帶來了編程學習方式的變革，還創造了全新的職業機會。完成本章的學習後，你將建立對編程的正確認識，為後續章節的學習做好準備。

1.1 本章學習目標

本章旨在幫助讀者理解在AI時代學習編程的重要性。通過閱讀本章，你將瞭解：

- 為什麼在AI時代普通人也應該掌握基本的編程技能。
- 普通人可以通過編程解決哪些實際問題。
- AI工具如何降低編程學習門檻，為普通人帶來新機會。

1.2 AI時代普通人為什麼要學編程

過去，編程常被視為專業程序員的技能，普通人往往認為它是一個高門檻、難以掌握的領域。然而，隨著人工智能技術的快速發展，特別是像DeepSeek這樣的AI工具的出現，編程的門檻已經大幅降低，學習編程變得前所未有的簡單。

(1) 數字化轉型下的必備技能

當今世界，幾乎所有行業都在被數字技術重塑：

- ❑ 傳統製造業引入智能工廠和數據分析。
- ❑ 零售業轉向電子商務和個性化推薦。
- ❑ 醫療行業應用AI輔助診斷和遠程醫療。
- ❑ 教育領域發展在線學習和自適應教學系統。

在這樣的背景下，具備基本的編程思維和技能已經不再是“錦上添花”，而是在職場中保持競爭力的“必備武器”。

(2) 提升個人效率與解決問題的能力

編程本質上是一種解決問題的工具。學習編程能夠：

- ❑ 培養結構化思維和邏輯分析能力。
- ❑ 提高工作效率，自動化重複性任務。
- ❑ 增強數據分析和決策能力。

□ 獲得創造數字工具的能力，解決特定問題。

(3) AI時代的學習優勢

與以往不同，在AI時代學習編程具有顯著優勢：

□ AI助手（如DeepSeek）可以提供即時指導和代碼建議。

□ 大量高質量的學習資源免費可得。

□ 編程語言和工具變得更加友好和易用。

□ 社區支持更加完善，問題解決更加便捷。

在AI的輔助下，普通人可以跳過許多傳統學習中的痛點，更快地掌握實用技能，並用其解決實際問題。

1.3 普通人能用AI編程做什麼

隨著技術的發展，編程已經不再是專業開發者的獨有技能。如今，普通人也能通過編程，尤其是藉助AI技術，輕鬆解決工作以及生活中的問題，創造出令人驚歎的成果。

1.3.1 用AI編程解決工作中的問題

還記得那位來自溫州、沒有編程背景的小夥陳雲飛嗎？他僅僅花了1小時，就利用AI創造了一款名為“小貓補光燈”的App。這款看似簡單的應用，不僅迅速獲得了超過3萬的下載量，還一度衝上AppStore分類榜Top 20。更令人驚歎的是，僅半個月後，他迭代推出的“小貓補光燈Pro”版本更是登頂了AppStore收費應用總榜第一名。

這就是AI編程的魔力——它賦予了普通人將想法快速轉化為現實產品的驚人能力。

1.用編程高效處理數據報表

小林是某互聯網公司的運營專員，每週需要手動合併5個不同部門、不同格式的GMV（商品交易總額）數據報表。僅整理報告所需的數據就要耗費3天，製作PPT又要2天。

現在他只需要編寫一段提示詞發給DeepSeek，DeepSeek就可以構建一個自動化腳本，自動抓取各平臺數據，進行清洗、合併等操作，並直接生成可視化圖表和報告初稿。

DeepSeek提示詞示例：

编写一个 AI 脚本，解决数据合并与报告生成的难题：每周自动抓取、清洗、合并 5 个不同部门、不同格式的 GMV 报表，并生成可视化图表与报告初稿。

DeepSeek生成設計方案示例：

总结：AI 数据合并与报告生成系统

一、核心功能

- ❑ 自动数据抓取：从 5 个不同部门收集不同格式 (Excel/CSV) 的 GMV 报表。
- ❑ 智能清洗与转换：处理缺失值、去重、标准化数据 (如日期、产品 ID)。
- ❑ 自动合并：整合多部门数据，确保一致性。
- ❑ 可视化分析：生成关键图表 (收入占比、Top 产品、趋势线、热力图)。
- ❑ 报告生成：自动输出结构化 PDF 报告 (含摘要、图表、数据表)。
- ❑ 邮件通知：自动发送报告给相关人员。

二、代码架构

…… (此处省略具体内容，读者可利用提示词自行尝试并查看)

2.1 数据分析 (自动计算关键指标)

…… (此处省略具体内容，读者可利用提示词自行尝试并查看)

2.2 报告生成 (PDF + 可视化)

…… (此处省略具体内容，读者可利用提示词自行尝试并查看)

2.3 自动化执行

…… (此处省略具体内容，读者可利用提示词自行尝试并查看)

三、技术亮点

…… (此处省略具体内容，读者可利用提示词自行尝试并查看)

效果總結：

即使是一個相對簡單的AI程序或自動化腳本，也能為你節省下數小時甚至數天寶貴的工作時間，讓你從煩瑣重複的勞動中解放出來，專注於更具創造性和更高價值的工作。

2.用編程批量處理發票

張姐是某連鎖餐飲店的財務，每天需要處理200多張各式各樣的發票。其中，大約35%的時間用於掃描和識別票據信息，40%的時間用於手動核對數據。

現在她只需要編寫一段提示詞，讓DeepSeek開發一個工具，通過拍照自動識別發票的關鍵信息（抬頭、金額、稅號等），並自動生成會計憑證或直接錄入系統的詳細的系統設計方案。然後將設

計方案放在代碼編輯工具（如VS Code或者Windsurf）裡，執行後便可以得到一個自動化系統。

DeepSeek提示詞示例：

构建一个 AI 工具，实现发票信息的自动化识别与凭证生成：通过拍照自动提取发票抬头、金额、税号等关键信息，并自动生成会计凭证或直接录入系统，解放财务人员，提高效率。

DeepSeek生成設計方案示例：

发票信息自动化识别与凭证生成系统设计方案

一、系统概述

该 AI 工具旨在通过 OCR[⊖]技术和机器学习算法，自动识别发票中的关键信息，并生成会计凭证或直接录入财务系统，大幅提升财务工作效率。

二、核心功能模块

发票图像采集模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

智能 OCR 模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

智能校验模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

凭证自动生成模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

系统对接模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

三、预期效益

☐ 效率提升：单张发票处理时间从 5 ~ 10 分钟缩短至 30 秒内。

☐ 准确性提高：识别准确率达 98% 以上，减少人为错误。

☐ 成本节约：减少 60% 以上的基础财务工作人力成本。

☐ 数据价值：积累结构化财务数据，支持后续分析决策。

这个系统将显著提升企业财务处理效率，让财务人员从事具有更高价值的工作，同时提高财务数据的准确性和及时性。

⊖ OCR為光學字符識別。——編輯注

效果總結：

通過DeepSeek生成的自動化系統，即使是一個相對簡單的AI解決方案，也能顯著節省處理發票的時間，讓你擺脫煩瑣重複的任務，將精力投入到更具創造性和更高價值的工作中。

3.用編程實現智能客戶溝通

王哥是一位醫療器械銷售主管，他的客戶諮詢信息分散在微信、釘釘、短信等多個渠道，信息零散，常常導致回覆遺漏或跟進不及時。

利用DeepSeek，王哥可以著手構建一個智能客戶溝通管理系統，實現如下功能：

❑ 聚合信息：自動將來自不同溝通渠道的消息彙集到一個統一平臺。

❑ 智能摘要：利用NLP（自然語言處理）技術自動提取客戶諮詢的關鍵信息和需求要點，生成摘要。

❑ 節點提醒：根據溝通內容和預設規則，在關鍵時間節點（如報價後、樣品寄出後）自動提醒王哥進行跟進。

DeepSeek提示詞示例：

构建一个智能客户沟通管理系统，解决信息分散、回复遗漏的问题：自动汇集来自微信、钉钉、短信等渠道的信息，利用NLP技术提取关键信息和需求要点，并根据沟通内容自动生成提醒，对关键信息节点进行跟进。

DeepSeek生成設計方案示例：

智能客户沟通管理系统设计方案

一、系统概述

该智能客户沟通管理系统旨在整合多渠道客户沟通信息，通过自然语言处理技术自动提取关键信息，智能分析客户需求，并提供跟进提醒功能，解决企业客户信息分散、回复遗漏等问题。

二、核心功能模块

多渠道信息集成模块

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

智能信息处理引擎

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

智能工单与任务管理

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

智能提醒与跟进系统

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

客户画像与知识

……（此处省略具体内容，读者可利用提示词自行尝试并查看）

三、预期效益

- 效率提升：客户响应时间缩短 70%。
- 遗漏减少：关键节点跟进提醒覆盖率 100%。
- 体验改善：客户满意度提升 30%。
- 管理透明：全渠道沟通记录统一可查。

该系统将彻底改变企业碎片化的客户沟通现状，实现智能化、系统化的客户交互管理，提升客户满意度和团队协作效率。

效果總結：

AI編程的核心優勢之一在於，它能夠顯著降低原本需要大量人力才能完成的任務的人力成本（或時間成本）。這種成本優勢不僅體現在直接的人力投入減少上，還體現在工作效率提升、錯誤率降低和資源優化分配方面。

1.3.2 用AI編程提升生活質量

1.個人生活智能化

掌握AI編程，意味著你擁有了為自己量身定製智能生活解決方案的能力，讓科技真正服務於你的生活。以下是一些常見的場景。

□ 智能家居控制系統：構建個性化的控制邏輯，聯動不同品牌的智能設備，實現更符合你的生活習慣的自動化場景（如清晨自動

拉開窗簾、播放音樂、離家自動關閉電器)。

□ 個人財務管理工具：開發自動記賬、預算規劃、投資組合分析等工具，提供更貼合你的消費習慣和理財目標的智能建議。

□ 健康監測與提醒助手：結合可穿戴設備數據或手動輸入數據，創建個性化的健康追蹤、異常預警、用藥提醒或運動建議系統。

以下是一個用DeepSeek開發自動記賬應用的示例。

DeepSeek提示詞示例：

请为我生成一个 Python 自动记账工具的完整开发方案，该工具需要具备以下功能：

- 自动数据导入：从银行短信、电子邮件或支付平台（如支付宝、微信、PayPal）解析交易信息，并录入数据库。
- 消费分类与标签：基于交易关键词，自动对收入和支出进行分类（如餐饮、购物、交通、娱乐）。
- 预算与提醒机制：根据用户设定的预算，实时监控支出，并在接近预算时发送提醒。
- 消费趋势与数据可视化：通过matplotlib/Plotly绘制月度、季度和年度的消费趋势分析图。
- 投资组合跟踪：集成股票、基金等 API[⊖]，自动更新资产数据并生成投资收益报告。

技术要求：

- 使用pandas处理数据，支持CSV/Excel文件导入与导出。
- 数据存储使用SQLite或MongoDB，支持用户自定义备份。
- 具备异常检测功能（如高额支出、重复交易），并发送短信/邮件通知。
- 请生成详细的系统架构、代码实现方案，并附上相关依赖库的安装说明。

⊖ API為應用程序接口。——編輯注

DeepSeek生成結果精簡示例：

Python 自动记账工具方案

一、核心功能

- 交易记录自动导入 (短信 / 邮件 / API)
- 智能分类消费类型
- 预算监控与提醒
- 消费数据可视化
- 投资组合跟踪

二、关键代码实现

数据模型 (models.py)

…… (此处省略具体内容, 读者可利用提示词自行尝试并查看)

数据库操作 (db.py)

…… (此处省略具体内容, 读者可利用提示词自行尝试并查看)

智能分类 (classifier.py)

…… (此处省略具体内容, 读者可利用提示词自行尝试并查看)

预算监控 (budget.py)

…… (此处省略具体内容, 读者可利用提示词自行尝试并查看)

主程序 (main.py)

…… (此处省略具体内容, 读者可利用提示词自行尝试并查看)

三、使用示例

1. 添加交易

```
process_transaction(50.0, "星巴克咖啡", "2023-10-01")
```

2. 查看消费

```
transactions = db.get_transactions()
```

扩展建议

- 增加邮件 / 短信解析模块
- 添加可视化图表功能
- 支持投资数据跟踪

说明: 这个精简方案保留了核心记账功能, 代码量减少约 70%, 同时保持了良好的扩展性。

2. 解決日常問題

AI編程並非總是關乎高深的科技，它同樣能有效解決我們身邊具體、瑣碎但耗時的日常問題。

以街道辦文員周姐為例，她每天可能面臨大量的文書工作。

手動錄入：將大量的紙質或手寫通知、表格轉錄為電子文檔。

數據彙總：從各個社區收集格式不一的防疫數據或其他報表，手動進行彙總和統計。

❑ 報告撰寫：定期整理材料，撰寫季度工作總結或彙報。

通過學習和應用AI編程，周姐可以逐步構建一個智能公文助手，從而實現以下功能：

❑ 格式檢查：利用OCR和規則引擎，自動檢查紅頭文件、公文的格式規範性。

❑ 要點提取：使用NLP技術，智能提取領導講話、會議紀要的核心要點，快速生成摘要。

❑ 快速檢索：對歷史文檔進行智能索引，實現對大量電子文檔內容的秒級檢索，快速找到所需信息。

現在可以利用DeepSeek來開發一個小工具，用於解決周姐的問題。

DeepSeek提示詞示例：

請為我生成一個 Python 智能公文助手，實現 OCR、公文格式檢查、NLP 摘要生成、智能檢索與分類等功能，提高公文處理效率，支持多種格式與數據自動化分析。

DeepSeek生成結果示例：

Python 智能公文助手精簡方案

一、核心功能

- ❑ 格式檢查：OCR+ 規則引擎驗證公文格式
- ❑ 內容分析：NLP 提取要點 + 生成摘要
- ❑ 智能檢索：向量數據庫實現語義搜索
- ❑ 數據統計：自動彙總報表數據

二、關鍵代碼實現

1. OCR 處理 (PaddleOCR)

…… (此處省略具體內容，讀者可利用提示詞自行嘗試並查看)

2. 格式檢查

…… (此處省略具體內容，讀者可利用提示詞自行嘗試並查看)

3. 摘要生成

…… (此處省略具體內容，讀者可利用提示詞自行嘗試並查看)

4. 文檔檢索

…… (此處省略具體內容，讀者可利用提示詞自行嘗試並查看)

3.提升生活體驗

通過AI編程，你可以將技術融入生活的方方面面，打造更便捷、更智能、更貼合個人需求的未來生活。

(1) 智能旅行規劃助手

□ 根據興趣、預算和時間，自動規劃行程、推薦景點、預訂交通住宿。

□ 基於旅行風格（探險、文化、美食等）推薦差異化路線。

□ 即時監控天氣變化、交通延誤，智能調整行程。

□ 融合當地人評價和小眾景點，避開常規旅遊陷阱。

□ 自動生成數字旅行日記，整合照片、位置和體驗記錄。

(2) 個性化學習計劃生成器

□ 結合學習目標、現有知識和可用時間，智能生成並調整學習計劃與資源推薦。

□ 根據學習進度和理解程度，動態調整內容難度和複習頻率。

□ 識別知識盲點，自動推薦針對性補充材料。

□ 分析最佳學習時段，優化學習安排。

□ 融合多種學習方式（視頻、閱讀、實踐），匹配個人學習偏好。

(3) 飲食與健康管理系統

□ 根據健康數據、飲食偏好和運動目標，提供個性化的食譜建議、營養分析和健康追蹤。

□ 考慮季節性食材和本地可獲得性，推薦經濟實惠的健康選擇。

□ 智能調整飲食計劃以適應特殊場合（旅行、節日、外出就餐）。

□ 將健康數據與飲食習慣關聯，發現個人特定的反應模式。

□ 預測並提前警示可能的健康問題，提供均衡建議。

1.4 AI編程給普通人帶來的新機會

1.跨行業的AI應用

AI編程為各行各業帶來前所未有的新機會，讓每個職業都能擁有個性化的智能助理。

□ 市場專員：自主構建精細化的用戶畫像系統，實現精準營銷和客戶互動。

□ 財務主管：通過簡單的語音或文字指令，快速自動生成高質量的月度經營分析報告。

□ 教師：使用簡單易用的可視化工具，輕鬆定製個性化的AI教學助手，輔助課堂互動和教學評估。

□ 自媒體博主：僅需提供核心觀點或簡短描述，即可快速生成具有創意的內容腳本，幫助打造熱門作品。

□ 創業者：幾天內即可利用AI快速搭建最小可行產品，高效驗證商業模式並加速迭代。

2.新興職業的發展

AI編程正在催生一系列新的職業和崗位，極大地拓展了職業發展的可能性：

□ AI提示工程師：設計並優化與AI模型的交互指令，確保獲得最佳輸出效果。

□ 自動化流程專家：使用AI技術設計和實施自動化工作流程，提升企業運營效率。

□ 智能系統集成顧問：幫助企業成功將AI技術融入現有業務系統，確保無縫集成與高效協同。

□ AI應用定製師：針對企業或行業的特殊需求，量身定製並優化專屬AI應用程序，提升競爭力。

3.個人競爭力提升

AI編程是這個時代最公平的“命運改造車間”，掌握AI編程意味著獲得了獨特的核心競爭力：

□ 創造獨特性：當大部分人依賴現成的工具時，你可以自主設計並實現精準滿足自身需求的AI解決方案，打造真正個性化的智能系統。

□ 掌握自主權：擺脫對外部團隊或高成本外包的依賴，具備快速、自主設計解決方案的能力，構建屬於自己的效率或商業“護城河”。

□ 實現敏捷性：靈感和創意無須等待冗長的審批和開發流程，通過AI快速完成原型搭建、驗證和迭代，搶佔市場先機。

AI編程賦予個體前所未有的變革性力量，讓你從被動的技術使用者成長為主動的創新者和價值創造者。

1.5 結語

“每個未實現的創意都是被封印的潛力，每段未落地的代碼都是被浪費的人生可能性。”

AI已不再是少數“程序員”的專屬領地，它正迅速演變為當今時代每個人都應瞭解，甚至掌握的現代生存技能。

在這個技術以驚人速度迭代的時代，學習AI編程已經不是一道關於興趣的“選擇題”，而更像是一門關乎未來競爭力的“必修課”。

通過對本書的學習，你將很快掌握AI的相關知識、DeepSeek等大模型的使用方法等，並利用相關工具（如VS Code或者Windsurf）將腦海中的創意火花轉化為可觸碰的現實工具，進而從煩瑣重複的工作中解放出來，將更多的時間與精力用於發現並抓住全新的職業機遇，最終改善並提升自己的生活質量。

第2章 DeepSeek編程指南

本章將詳細介紹DeepSeek的登錄和使用，以及DeepSeek在現實生活和工作中的應用場景；詳細介紹DeepSeek編程的優勢、通過提示詞發揮DeepSeek的創造力，以及使用DeepSeek的有趣且高效的提示詞技巧。

2.1 本章學習目標

學習本章，你將掌握以下內容或技能：

- ❑ 對DeepSeek模型的功能有一定的瞭解。
- ❑ 瞭解DeepSeek的應用場景。
- ❑ 應用DeepSeek進行編寫相關代碼。
- ❑ 學會使用DeepSeek提示詞的高階技巧。

2.2 什麼是DeepSeek

DeepSeek是一款強大的國產AI大模型及AI工具，專為AI編程和多種任務提供智能輔助。其核心目標是幫助用戶提升工作效率，優化任務執行，並解決複雜問題。作為智能助手，DeepSeek在代碼生成、自然語言處理、數據分析等領域展現出卓越的能力，為開發者和專業人士提供了高效、精準的解決方案。

1. 核心定位與能力

□ 專精於編程輔助：DeepSeek在設計之初就特別側重於代碼相關的任務。它不僅能生成代碼片段，還能進行代碼補全、解釋、調試、優化，甚至跨語言翻譯代碼和生成單元測試。這使得它成為開發者在編程過程中的得力助手，能顯著提升開發效率和代碼質量。

□ 通用智能與多任務處理：除了強大的編程能力，DeepSeek也是一個通用的智能體。它在自然語言理解和生成方面同樣表現出色，能夠處理包括文本創作、摘要提取、問答、翻譯、數據分析、報告撰寫等多種複雜任務。

□ 效率與優化的驅動者：其核心目標是賦能用戶，提升工作與學習效率。通過自動化重複性工作、提供快速準確的信息、輔助決策制定以及解決傳統方法難以處理的複雜問題，DeepSeek致力於優化用戶的任務執行流程。

2. 技術基礎與特點

□ 基於海量數據訓練：作為一個大模型，DeepSeek通過學習海量的文本和代碼數據，獲得了強大的模式識別、邏輯推理和

知識整合能力。

□ 持續進化：DeepSeek模型會不斷迭代更新，吸收新的知識，優化性能，以適應快速發展的技術環境和用戶需求。

2.3 登錄和使用DeepSeek

1.登錄DeepSeek

(1) 網頁版登錄

在瀏覽器中訪問<https://www.deepseek.com/>，單擊“開始對話”按鈕，使用手機號、郵箱或微信掃碼進行註冊或登錄。

(2) 手機應用登錄

在應用商店搜索DeepSeek或“深度求索”並下載官方App。打開App後，使用手機號、郵箱或微信掃碼進行註冊或登錄。

單擊“開始對話”按鈕，通過驗證之後，即可登錄DeepSeek聊天界面，如圖2-1所示。



圖2-1 DeepSeek聊天界面

2.使用DeepSeek

(1) 中央區域（對話窗口）

協助你進行代碼編寫、文案寫作、內容創意、文件解析等任務。

(2) 輸入框區域

- ❑ 你可以在中央輸入框輸入任何問題或任務。
- ❑ 支持粘貼代碼、提問對話、寫作創作等形式。

(3) 按鈕功能（輸入框下方）

□ 深度思考（R1）：啟用更強邏輯推理的模型，適合複雜任務。

□ 聯網搜索：開啟即時聯網功能，獲取最新互聯網信息。

□ 附件上傳（右側回形針圖標）：支持上傳文件進行閱讀、總結、翻譯等。

□ 發送按鈕（右下角箭頭）：發送當前輸入內容。

2.4 DeepSeek的核心能力與應用場景

在人工智能的應用場景中，DeepSeek作為國產AI大模型，其強大的多模態能力和廣泛的功能領域，在各種任務中展現了卓越的能力。下面將詳細介紹DeepSeek的基礎能力、主要應用場景，以及DeepSeek在AI編程中的具體優勢。

2.4.1 DeepSeek的基礎能力

1.能力概述

DeepSeek擁有強大的多模態處理能力。這意味著它不僅能處理單一類型的數據（如文本或圖像），還能夠綜合不同類型的信息進行分析和生成輸出。通過這種多模態能力，DeepSeek能夠跨越不同的數據領域，執行復雜的任務，如文本生成、自然語言理解、代碼輔助、數據分析等。

2.應用場景

(1) 文本生成

DeepSeek可生成高質量的自然語言文本，適用於各種場景，如文章寫作、郵件撰寫、廣告文案、技術文檔等。它可以根據用戶提供的提示詞或上下文自動生成符合語境、流暢且有創意的文本。

應用案例：

□ 項目文檔自動生成：DeepSeek可以根據項目的背景和要求，自動生成項目計劃書、技術文檔等，節省人力成本。

□ 營銷文案生成：它能夠為產品或服務自動生成吸引人的廣告文案，提升營銷效率。

(2) 自然語言理解

DeepSeek在理解和分析複雜文本方面也表現出色。它能夠解析用戶輸入的自然語言，理解其中的語義和情感，並提供相應的響應或分析結果。

應用案例：

□ 情感分析：在社交媒體監控或客戶反饋分析中，DeepSeek可以幫助識別文本中的情感傾向（正面、負面或中立），為決策者提供有價值的信息。

□ 文本分類與主題識別：DeepSeek能夠將大量的文檔、評論、郵件等文本信息進行分類，快速找到重要內容。

(3) 代碼輔助

作為一款支持編程語言的AI模型，DeepSeek能輔助開發人員進行代碼編寫、調試和優化。它不僅能夠生成代碼片段，還能提供有關算法、數據結構等的優化建議。

應用案例：

□ 自動代碼生成：根據需求描述，DeepSeek可以自動生成代碼模板或特定功能的實現代碼，減少編程工作量。

□ 代碼優化與重構：對於已有的代碼，DeepSeek能夠提供性能優化建議或重構方案，提高代碼質量和可維護性。

(4) 數據分析

DeepSeek的多模態能力還擴展到數據分析領域，尤其是在處理複雜、結構化和非結構化數據時，它可以幫助用戶快速提取有價值的見解。

應用案例：

□ 數據清洗與處理：DeepSeek能夠自動識別並處理數據中的錯誤或缺失值，提高數據的質量。

□ 趨勢分析與預測：在金融、零售等行業，DeepSeek可幫助分析歷史數據，預測未來趨勢，如銷售預測、股市分析等。

通過這些應用場景，DeepSeek的多模態能力顯著提升了各行業用戶的工作效率，並且能夠解決實際工作中的複雜問題。在未來的應用中，DeepSeek可能會在更多領域展現其強大的多模態處理能力。

2.4.2 DeepSeek的編程能力

1.能力概述

DeepSeek在編程領域具備卓越的能力，能夠為開發者提供全方位的編程支持。憑藉深度學習和大規模的編程數據訓練，DeepSeek可以生成高質量的代碼，提供代碼分析和調試功能，優化算法，並協助開發者進行單元測試等任務。這些功能不僅適用於各種編程語言和框架，還能幫助開發者解決複雜的技術難題，極大地提升了開發效率，降低了編程難度。

2.應用場景

(1) 代碼生成

DeepSeek能夠根據自然語言描述或需求，自動生成符合預期的代碼。這不僅加速了開發過程，還能夠為開發者提供常見任務的解決方案，減少了手動編寫重複性代碼的工作。

應用案例：

- ❑ Web 應用開發：開發者只需描述網頁的功能要求，DeepSeek可自動生成對應的前端和後端代碼，幫助開發者快速啟動項目。

- ❑ 算法實現：開發者通過簡單的需求描述，DeepSeek可自動生成相關的算法代碼，如排序、搜索、圖算法等，減少編寫複雜算法的時間。

(2) 代碼解釋與調試

DeepSeek能夠分析並解釋現有代碼，幫助開發者理解代碼的功能和邏輯。它還能自動識別代碼中的潛在問題，提供調試建議，甚至直接修復常見的錯誤。

應用案例：

- ❑ 錯誤檢測與修復：DeepSeek可以分析代碼中的錯誤，給出詳細的錯誤信息及修復建議，幫助開發者快速定位和解決問題。

- ❑ 代碼解釋：對於複雜的代碼段，DeepSeek能提供逐行解釋，幫助開發者理解代碼的實現原理和流程，尤其在調試過程中非常有用。

(3) 代碼翻譯與遷移

DeepSeek支持不同編程語言之間的代碼翻譯，使得開發者能夠在多個平臺或項目中輕鬆切換語言。它能夠將一種編程語言的代

碼準確翻譯成另一種語言，並保證邏輯一致性。

應用案例：

- ❑ 跨平臺開發：在開發跨平臺應用時，DeepSeek可以幫助開發者將Java代碼轉化為Python，或將Python代碼轉化為JavaScript，確保多平臺兼容。

- ❑ 代碼遷移：在將舊項目遷移到新的編程語言時，DeepSeek可以自動翻譯並適配代碼，減少人工修改工作量。

(4) 生成單元測試

DeepSeek能夠根據現有代碼自動生成單元測試腳本，確保代碼的功能正確性，並便於後期的維護和更新。

應用案例：

- ❑ 自動化測試：開發者只需提供要測試的代碼，DeepSeek會自動生成相應的測試用例，並在開發過程中持續更新和優化。

- ❑ 迴歸測試：對於已有的項目，DeepSeek可自動生成迴歸測試用例，確保新功能的加入不會破壞現有功能。

(5) 算法優化與性能提升

DeepSeek不僅能提供常見算法的實現，還能根據具體場景提供性能優化建議，幫助開發者提升程序的效率和可擴展性。

應用案例：

- ❑ 數據處理優化：DeepSeek可以優化大數據處理算法，例如在處理海量數據時，提出性能提升方案，減少計算時間和資源消耗。

□ 內存優化：對於內存佔用過高的程序，DeepSeek能夠提供內存優化建議，幫助開發者降低程序的內存使用。

3.在AI編程中的優勢

(1) 提升開發效率

DeepSeek可以自動生成代碼和測試用例，幫助開發者快速完成常見功能的實現，顯著提升開發效率。這尤其適用於原型開發和快速迭代的項目。

(2) 降低學習曲線

通過代碼解釋和調試功能，DeepSeek可以幫助初學者理解複雜代碼和編程概念，降低學習編程的難度。此外，DeepSeek的多語言支持使得開發者能夠快速切換和適應不同的編程語言，減少了語言切換的成本。

(3) 減少重複性工作

DeepSeek能夠自動生成代碼模板和常用功能模塊，避免了開發者編寫重複性代碼的工作，從而將精力集中在更具創造性的任務上。

(4) 優化代碼質量

DeepSeek在代碼優化和重構方面的優勢可以幫助開發者編寫更加高效、簡潔的代碼，提升項目的可維護性和可擴展性。

(5) 高效的跨語言協作

在跨語言開發和多平臺協作中，DeepSeek提供的代碼翻譯能力極大地方便了不同編程語言的協作，確保開發者能夠高效地在不同

同語言間遷移和集成代碼。

通過這些功能，DeepSeek在AI編程領域展現出了強大的優勢，成為開發者高效編程的重要工具。它不僅能提升編程效率，降低開發成本，還能在項目開發過程中提供持續的支持和優化，推動技術創新和應用實踐。

2.4.3 DeepSeek在AI編程中的應用案例

1.案例1：快速生成函數/類

data.csv文件數據示例如表2-1所示。

表2-1 data.csv文件數據示例

product_id	product_name	category	price	stock	date
1001	Smartphone X	Electronics	799.99	120	2025-03-19 16:09:13.772216
1002	Laptop Pro	Electronics	1299.99	85	2025-03-12 16:09:13.772216
1003	Wireless Headphones	Electronics	149.99	200	2025-03-10 16:09:13.772216
1004	Coffee Maker	Home Appliances	89.99	45	2025-03-16 16:09:13.772216
1005	Blender Plus	Home Appliances	79.99	60	2025-03-12 16:09:13.772216
1006	Running Shoes	Sports	129.99	95	2025-03-17 16:09:13.772216
1007	Fitness Tracker	Electronics	89.95	150	2025-03-15 16:09:13.772216
1008	Desk Chair	Furniture	199.99	30	2025-03-21 16:09:13.772216
1009	Table Lamp	Home Decor	49.99	75	2025-03-07 16:09:13.772216
1010	Backpack	Accessories	59.99	110	2025-03-04 16:09:13.772216
1011	Water Bottle	Sports	19.99	200	2025-03-19 16:09:13.772216
1012	Smart Watch	Electronics	249.99	65	2025-03-12 16:09:13.772216
1013	Yoga Mat	Sports	29.99	80	2025-03-13 16:09:13.772216
1014	Winter Jacket	Clothing	179.99	40	2025-03-14 16:09:13.772216
1015	Portable Charger	Electronics	39.99	175	2025-03-27 16:09:13.772216

表2-1展示了產品ID、產品名稱、類別、價格、庫存數量和日期信息。

DeepSeek提示詞示例：

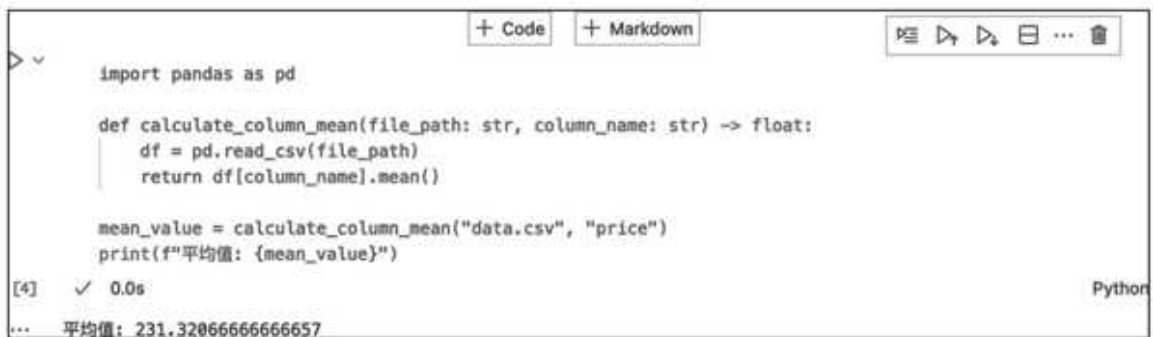
使用 Python 和 Pandas 编写一个函数，读取 csv 文件，并计算某一列的平均值。

DeepSeek生成結果示例：

```
import pandas as pd
def calculate_column_mean(file_path: str, column_name: str) -> float:
    df = pd.read_csv(file_path)
    return df[column_name].mean()
# 示例调用
mean_value = calculate_column_mean("data.csv", "price")
print(f"平均值：{mean_value}")
```

運行結果如圖2-2所示。

進一步優化：讓DeepSeek對數據進行可視化，生成精美的圖形。



```
import pandas as pd

def calculate_column_mean(file_path: str, column_name: str) -> float:
    df = pd.read_csv(file_path)
    return df[column_name].mean()

mean_value = calculate_column_mean("data.csv", "price")
print(f"平均值：{mean_value}")
```

[4] ✓ 0.0s Python

... 平均值：231.32066666666657

圖2-2 VS Code中的運行結果

DeepSeek生成代碼示例：

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter
import datetime
import matplotlib as mpl
from matplotlib.font_manager import FontProperties

设置中文字体支持
方法 1: 使用系统中文字体
try:
# 尝试使用系统中文字体
plt.rcParams['font.sans-serif'] = ['SimHei', 'Microsoft YaHei',
    'SimSun', 'Arial Unicode MS', 'sans-serif']
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
# 检查字体是否正常工作
plt.figure(figsize=(1, 1))
plt.text(0.5, 0.5, '测试中文', fontsize=12, ha='center')
plt.close()
print("成功设置中文字体")
except:
# 如果系统字体不可用, 尝试使用matplotlib内置的中文字体
print("系统中文字体不可用, 尝试其他方法...")
# 方法 2: 使用matplotlib内置的中文字体
try:
mpl.rc('font', family='Arial Unicode MS')
print("使用 Arial Unicode MS 字体")
except:
print("警告: 无法设置中文字体, 图表中的中文可能无法正确显示")

读取 CSV 数据
df = pd.read_csv('data.csv')
将日期列转换为 datetime 类型
df['date'] = pd.to_datetime(df['date'])
创建带有多个子图的图表
plt.figure(figsize=(15, 15))
1. 按类别的总价值 (价格 × 库存)
plt.subplot(2, 2, 1)
```

```

df['total_value'] = df['price'] * df['stock']
category_value = df.groupby('category')['total_value'].sum().sort_
    values(ascending=False)
category_value.plot(kind='bar', color='skyblue')
plt.title('按类别的总价值(价格 × 库存)', fontsize=14)
plt.ylabel('总价值($)', fontsize=12)
plt.xlabel('类别', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
2. 按类别的产品数量分布
plt.subplot(2, 2, 2)
category_counts = df['category'].value_counts()
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%',
    startangle=90, shadow=True, explode=[0.05] * len(category_counts))
plt.title('产品类别分布', fontsize=14)
3. 价格分布直方图
plt.subplot(2, 2, 3)
plt.hist(df['price'], bins=10, color='lightgreen', edgecolor='black')
plt.title('价格分布', fontsize=14)
plt.xlabel('价格($)', fontsize=12)
plt.ylabel('产品数量', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
4. 价格和库存散点图
plt.subplot(2, 2, 4)
scatter = plt.scatter(df['price'], df['stock'], c=df['price'],
    cmap='viridis',
    alpha=0.7, s=100, edgecolors='black')
plt.title('价格与库存关系', fontsize=14)
plt.xlabel('价格($)', fontsize=12)
plt.ylabel('库存数量', fontsize=12)
cbar = plt.colorbar(scatter, label='价格($)')
cbar.set_label('价格($)', fontsize=12)
plt.grid(True, alpha=0.3)
调整布局
plt.tight_layout()
plt.savefig('product_analysis.png', dpi=300, bbox_inches='tight')
创建第二个图表 - 按日期分析
plt.figure(figsize=(15, 8))
按日期排序
df_sorted = df.sort_values('date')
5. 日期和价格折线图
plt.plot(df_sorted['date'], df_sorted['price'], marker='o', linestyle='--',
    color='purple', markersize=8, linewidth=2)
plt.title('产品价格随时间变化', fontsize=16)
plt.xlabel('日期', fontsize=14)
plt.ylabel('价格($)', fontsize=14)
plt.grid(True, alpha=0.3)
格式化日期轴

```

```
date_format = DateFormatter('%Y-%m-%d')
plt.gca().xaxis.set_major_formatter(date_format)
plt.gcf().autofmt_xdate() # 自动旋转日期标签
plt.tight_layout()
plt.savefig('price_time_analysis.png', dpi=300, bbox_inches='tight')
print(" 可视化完成! 图表已保存为 'product_analysis.png' 和 'price_time_
analysis.png'")
```

根據DeepSeek生成的代碼在VS Code中生成可視化展示，產品價格類別分析圖如圖2-3所示，產品價格隨時間變化的折線圖如圖2-4所示。

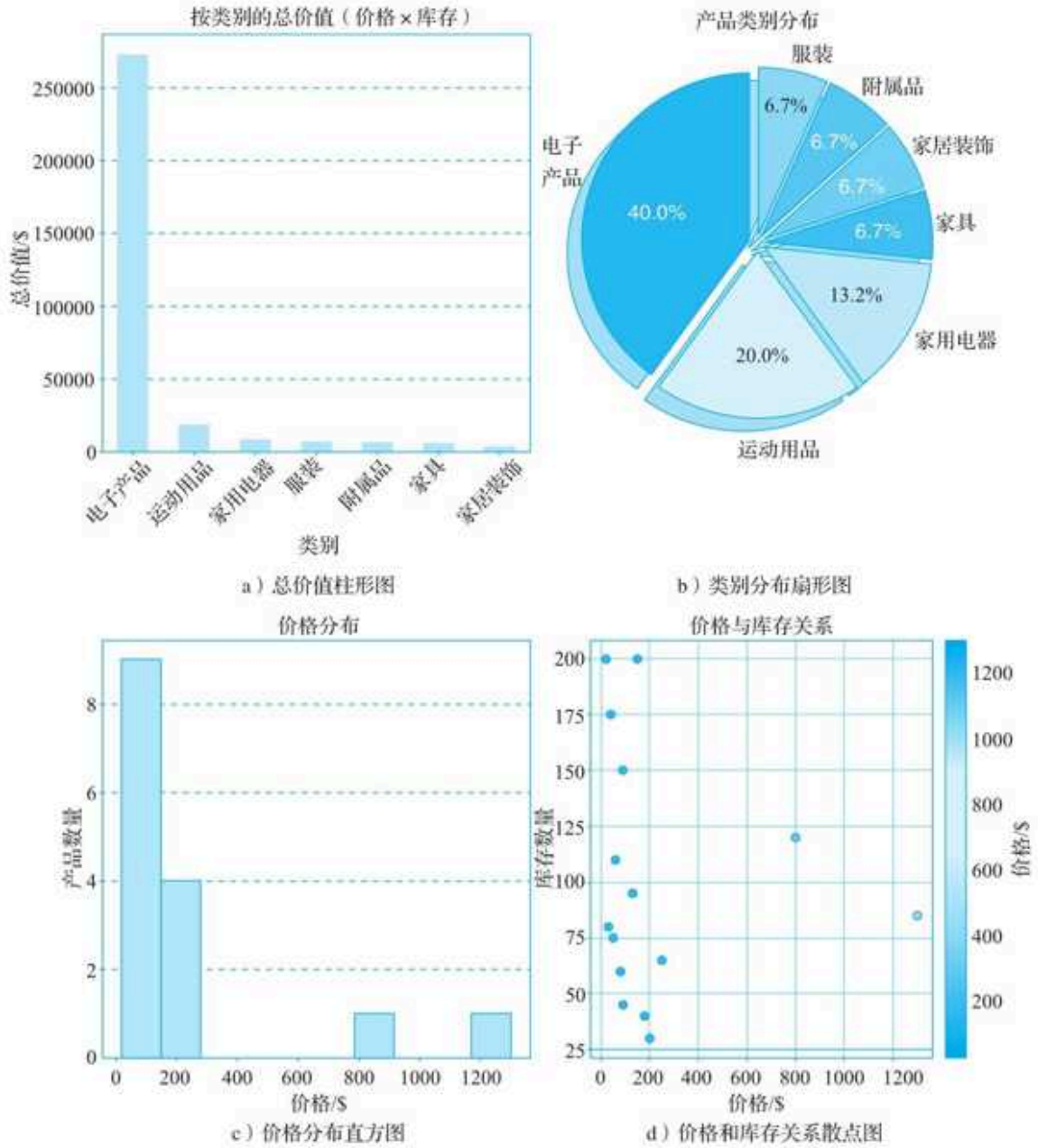


圖2-3 產品價格類別分析圖

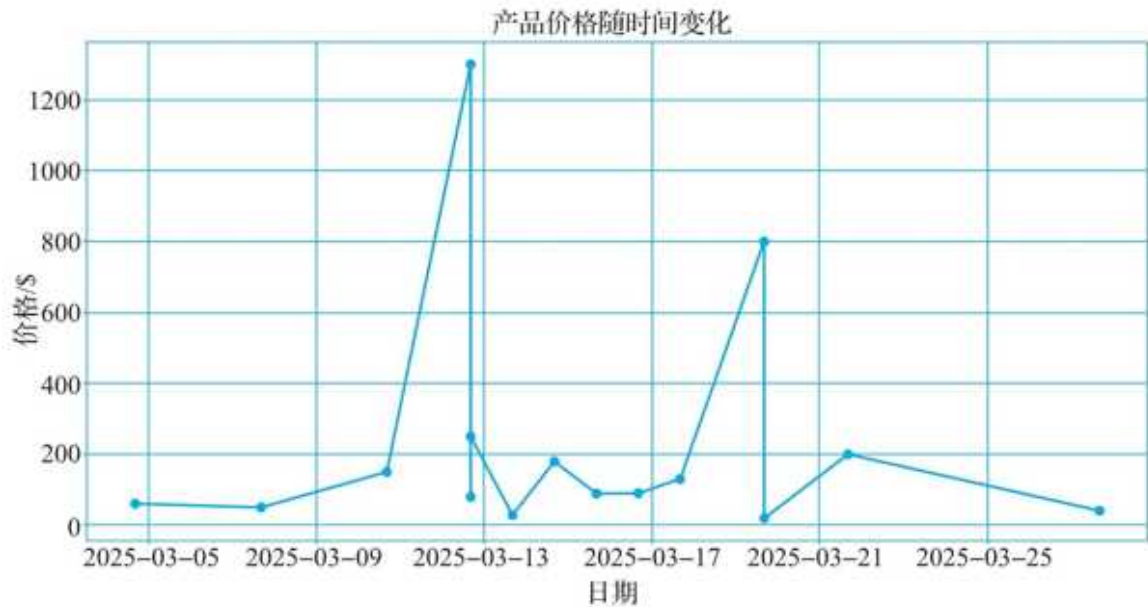


圖2-4 產品價格隨時間變化的折線圖

DeepSeek的優勢：

代碼結構清晰，包含參數類型註解，並可直接運行。

2.案例2：解釋複雜代碼

DeepSeek提示詞示例：

解释 TensorFlow 代码中 `tf.GradientTape` 的作用和原理。

DeepSeek生成結果示例：

`tf.GradientTape` 是 TensorFlow 2.x 中用于自动求导的工具。它在 `with tf.GradientTape() as tape:` 代码块中记录所有计算操作，并允许你随后计算梯度。例如，在神经网络训练中，它可用于计算损失函数对模型参数的梯度，以便进行优化。

示例代码：

```
import tensorflow as tf
x = tf.Variable(3.0)
with tf.GradientTape() as tape:
    y = x ** 2
dy_dx = tape.gradient(y, x)      # 计算 dy/dx
print(dy_dx.numpy())           # 输出 6.0
```

DeepSeek的優勢：提供直觀解釋+示例代碼，幫助用戶理解核心概念。

3.案例3：調試與錯誤修復

DeepSeek提示詞示例：

```
IndexError: list index out of range 错误，修复以下代码：
```

```
arr = [1, 2, 3]
```

```
print(arr[3])
```

DeepSeek生成結果示例：

这个错误是因为你试图访问一个不存在的列表索引而产生的。在 Python 中，列表索引是从 0 开始的，所以列表 `arr = [1, 2, 3]` 的有效索引是 0、1 和 2。

要修复这个错误，你有几个选择：

1. 如果你想访问最后一个元素，应该使用索引 2：

```
```python
arr = [1, 2, 3]
print(arr[2]) # 这会打印 3
```
```

2. 或者你可以使用负数索引来从列表末尾开始计数：

```
```python
arr = [1, 2, 3]
print(arr[-1]) # 这会打印最后一个元素 3
```
```

3. 如果你想安全地访问可能不存在的索引，可以使用异常处理：

```
```python
arr = [1, 2, 3]
try:
 print(arr[3])
except IndexError:
 print("索引超出范围")
```
```

原始代码 `print(arr[3])` 之所以会引发 `IndexError` 是因为列表只有 3 个元素（索引 0、1、2），而索引 3 不存在。

DeepSeek的優勢：自動識別錯誤並提供防止越界的代碼優化方案。

4.案例4：跨語言代碼轉換

DeepSeek可以快速將Java實現的快速排序算法轉換為Python代碼。

DeepSeek提示詞示例：

```
Java 代碼:
public class QuickSort {
    public static void quickSort(int[] arr, int left, int right) {
        if (left < right) {
            int pivotIndex = partition(arr, left, right);
            quickSort(arr, left, pivotIndex - 1);
            quickSort(arr, pivotIndex + 1, right);
        }
    }
    private static int partition(int[] arr, int left, int right) {
        int pivot = arr[right];
        int i = left - 1;
        for (int j = left; j < right; j++) {

            if (arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, right);
        return i + 1;
    }
    private static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
} 請幫我将Java代碼转换为Python代碼。
```

將上述提示詞需求發給DeepSeek，以下是Java快速排序算法的Python實現。

DeepSeek生成結果示例：

```

```python
def quick_sort(arr, left=None, right=None):
 if left is None:
 left = 0
 if right is None:
 right = len(arr) - 1

 if left < right:
 pivot_index = partition(arr, left, right)
 quick_sort(arr, left, pivot_index - 1)
 quick_sort(arr, pivot_index + 1, right)

def partition(arr, left, right):
 pivot = arr[right]
 i = left - 1

 for j in range(left, right):
 if arr[j] < pivot:
 i += 1
 arr[i], arr[j] = arr[j], arr[i]

 arr[i + 1], arr[right] = arr[right], arr[i + 1]
 return i + 1

使用示例
if __name__ == "__main__":
 data = [8, 7, 2, 1, 0, 9, 6]
 print(" 原始数组:", data)

```

```

 quick_sort(data)
 print(" 排序后数组:", data)
...

```

#### 主要变化和说明:

1. Python 不需要类型声明, 去掉了所有类型相关的代码。
2. 将 quickSort 改为 Python 风格的 quick\_sort。
3. 添加了默认参数处理, 使调用时可以只传入数组 quick\_sort(arr)。
4. Python 支持多重赋值, 简化了 swap 函数, 直接内联在代码中。
5. 去掉了类结构, 直接使用函数实现。
6. 添加了示例使用代码。

这个实现保持了与原始 Java 代码相同的逻辑和算法复杂度 ( $O(n \log n)$  平均情况)。

DeepSeek的優勢：精準轉換代碼，邏輯保持一致，同時符合Python代碼的風格。

## 5.案例5：生成單元測試

DeepSeek可以快速幫助開發者對Python函數生成單元測試，以下是DeepSeek生成單元測試的使用示例：

DeepSeek提示詞示例：

```
为以下 Python 函数编写 pytest 单元测试:
def add(a, b):
 return a + b
```

DeepSeek生成結果示例：

```
import pytest
def add(a, b):
 return a + b
def test_add_positive():
 assert add(2, 3) == 5
def test_add_negative():
 assert add(-1, -1) == -2
def test_add_mixed():
 assert add(-1, 1) == 0
def test_add_zero():
 assert add(0, 0) == 0
def test_add_large_numbers():
 assert add(1000000, 2000000) == 3000000
```

DeepSeek的優勢：自動編寫完整的單元測試，覆蓋多個邊界情況。

## 6.案例6：技術方案建議

DeepSeek可以快速設計一個詳細技術框架，然後根據此設計框架搭建項目。以下是通過DeepSeek設計技術框架的示例：

DeepSeek提示詞示例：

```
推荐系统设计后端架构。
```

## DeepSeek生成結果示例：

### 推荐系统后端架构设计

为推荐系统设计后端架构涉及多个层面，包括数据处理、模型训练与服务、API 接口等。以下是一个可能的架构方案概述：

#### 1. 数据层 (Data Layer)

数据源：用户行为日志(点击、购买、评分)、物品元数据(商品信息、文章内容)、用户画像数据。

数据存储：

原始日志：使用 Kafka 等消息队列收集，存入 HDFS 或对象存储(如 S3)。

处理后数据/特征：存入数据仓库(如 Hive, BigQuery)或 NoSQL 数据库(如 HBase, Cassandra) 便于快速查询。

用户/物品画像：可存入 Redis 或 Memcached 等内存数据库加速访问。

数据处理：使用 Spark, Flink 等进行 ETL<sup>⊖</sup>、特征工程。

#### 2. 模型层 (Model Layer)

离线训练：

召回模型：如协同过滤 (UserCF, ItemCF), 矩阵分解 (ALS, SVD), 向量检索 (Faiss, Annoy) 用于生成候选集。常用技术栈: Spark MLlib, TensorFlow, PyTorch。

排序模型：如逻辑回归 (LR), GBDT, FM, DeepFM, Wide&Deep, DIN 等, 对召回的候选集进行精排。常用技术栈: TensorFlow, PyTorch, XGBoost, LightGBM。

模型管理：使用 MLflow 或类似平台管理模型版本和实验。

近线/在线更新：对于需要快速反应用户兴趣变化的场景，可能需要近线更新(如分钟级/小时级更新用户向量)或在线学习模型。

#### 3. 服务层 (Service Layer)

推荐引擎 API：提供统一的 API 接口供前端调用。通常为 RESTful API。

召回服务：根据用户 ID 或上下文信息，快速从海量物品库中召回候选集。

排序服务：对召回结果进行实时打分排序。

策略/混排服务：融合不同召回源的结果，执行业务规则(如去重、多样性、强制插入等)。

特征服务：提供实时的用户和物品特征查询服务。

部署：使用 Docker + Kubernetes 进行容器化部署和管理，或使用云厂商提供的 Serverless 服务(如 AWS Lambda, SageMaker Endpoints)。

#### 4. 监控与评估 (Monitoring & Evaluation)

效果监控：监控推荐的点击率 (CTR)、转化率 (CVR)、曝光量等业务指标。

性能监控：监控 API 延迟、吞吐量、系统资源使用情况。使用 Prometheus + Grafana 或类似工具。

A/B 测试平台：用于在线评估不同推荐算法或策略的效果。

可选方案/技术栈考虑

方案 1 (适合初创/中小型)：基于 Spark MLlib/Mahout 实现离线协同过滤召回 + LR/GBDT 排序，部署为简单 API 服务。

方案 2 (通用/大型)：采用深度学习模型进行召回(如双塔模型)和排序(如 DeepFM, DIN)，利用向量数据库加速召回，构建微服务架构，部署在 K8s 上。

⊖ ETL為數據抽取、轉換、加載。——編輯注

方案3 (实时性要求高): 结合 Flink 进行实时特征计算和流式模型更新, 构建实时推荐系统。

优缺点分析

离线 vs 在线: 离线训练简单稳定, 但时效性差; 在线学习实时性好, 但架构复杂, 稳定性要求高。

传统模型 vs 深度学习: 传统模型解释性好, 训练快; 深度学习模型表达能力强, 效果上限高, 但需要更多数据和计算资源。

自建 vs 云服务: 自建灵活可控; 云服务省心省力, 按需付费, 但可能存在厂商锁定风险。选择哪种架构取决于具体的业务场景、数据规模、团队技术栈和资源投入。

DeepSeek的優勢：提供多種可選方案，並分析優缺點。

## 2.5 DeepSeek提示詞

在與AI進行交互時，高質量的提示詞能極大提升AI生成內容的準確性和實用性。無論是生成代碼、優化程序、解釋技術概念，還是調試錯誤，一個清晰、結構化的提示詞都能幫助AI更好地理解你的需求，並提供高質量的輸出。

本節將介紹一些提示詞模板，它們涵蓋了角色扮演、代碼解釋、錯誤修復、學習新知識等多個常見場景。你可以直接使用這些模板，或根據具體需求進行調整，以更高效地使用AI輔助編程。（更多提示詞模板參考附錄或者加入我們的社區。）

### 2.5.1 提示詞模板示例

#### 1. 角色扮演模板

用途：讓AI扮演某個專業角色，比如數據科學家、後端工程師、算法專家等，從而生成更符合該角色專業知識的回答。

DeepSeek提示詞模板：

```
角色：Python 数据科学家
任务：[清晰描述你的编程任务，例如，使用 Matplotlib 绘制散点图]
输入数据描述：[描述你的数据结构或提供示例]
要求 / 约束：[例如，添加图例和标题，指定颜色]
输出格式：[例如，Python 代码片段]
```

DeepSeek提示詞示例：

```
角色：Python 数据科学家
任务：使用 Matplotlib 绘制一个简单的随机散点图
输入数据描述：生成 100 个 (0,1) 区间的随机 x,y 坐标点

要求 / 约束：添加图例和标题，指定颜色为蓝色，点透明度为 0.5
输出格式：Python 代码片段，包含必要的导入和绘图展示代码
```

## DeepSeek生成結果代碼：

```
import numpy as np
import matplotlib.pyplot as plt
角色:Python 数据科学家
任务: 使用 Matplotlib 绘制一个简单的随机散点图
输入数据描述: 生成 100 个 (0,1) 区间的随机 x, y 坐标点
要求 / 约束: 添加图例和标题, 指定颜色为蓝色, 点透明度为 0.5
输出格式: Python 代码片段, 包含必要的导入和绘图展示代码
生成随机数据
np.random.seed(42) # for reproducibility
x = np.random.rand(100)
y = np.random.rand(100)
绘制散点图
plt.figure(figsize=(8, 6)) # 设置图形大小
plt.scatter(x, y, color='blue', alpha=0.5, label='随机数据点')
指定颜色、透明度和标签 (用于图例)
plt.title(" 随机散点图 ") # 添加标题
plt.xlabel("x 轴坐标 ") # 添加 x 轴标签
plt.ylabel("y 轴坐标 ") # 添加 y 轴标签
plt.legend() # 显示图例
plt.grid(True) # 添加网格线
plt.show() # 显示图形
```

代碼生成結果示例如圖2-5所示。

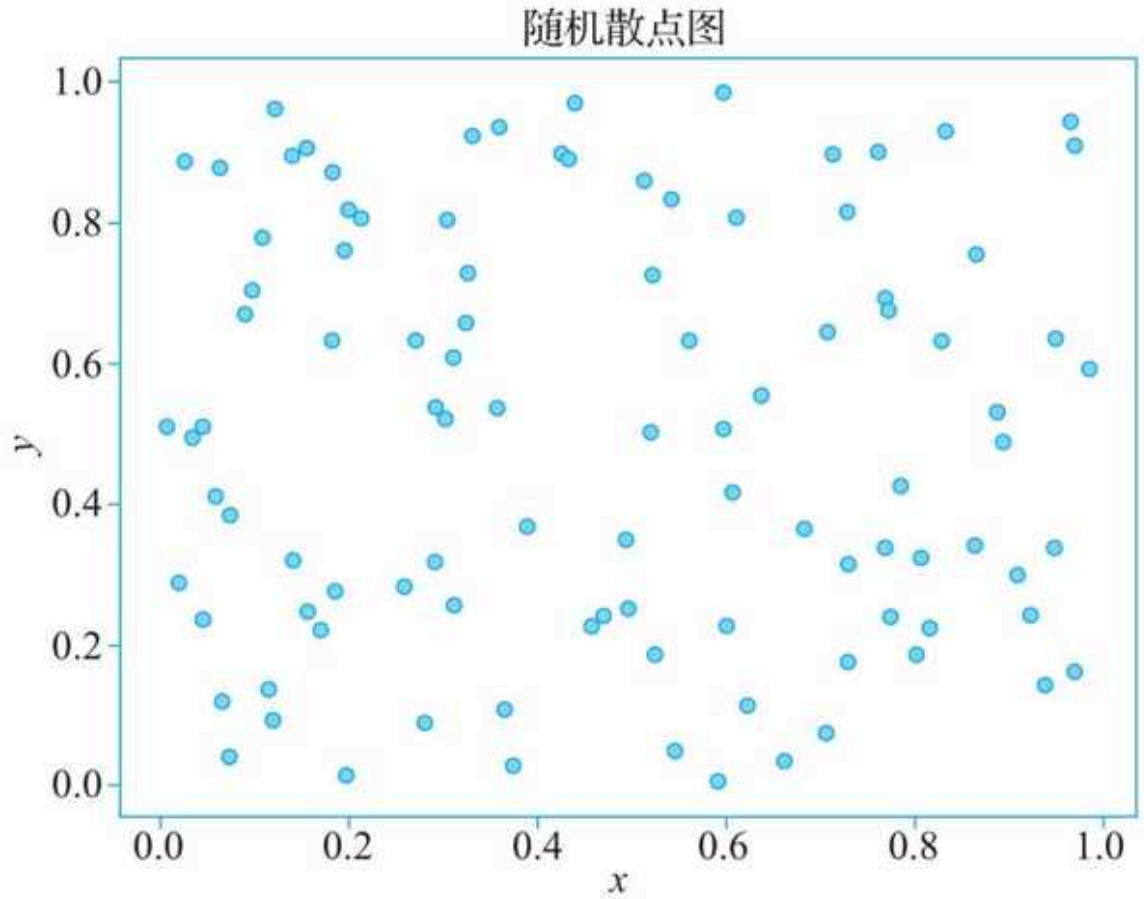


圖2-5 代碼生成結果示例

## 2.代碼解釋模板

用途：讓AI解析代碼，提供功能說明、關鍵步驟解析，並指出可能的優化點。

DeepSeek提示詞模板：

請解釋以下 [语言名称] 代碼的功能、關鍵步驟和潛在的優化點。

DeepSeek提示詞示例：

请解释以下 Python 代码的功能、关键步骤和潜在的优化点：

```
def fibonacci(n):
 if n <= 0:
 return "输入必须是正整数"
 elif n == 1:
 return [0]
 elif n == 2:
 return [0, 1]
 else:
 seq = [0, 1]
 for i in range(2, n):
 seq.append(seq[-1] + seq[-2])
 return seq
```

## DeepSeek生成結果示例：

Python Fibonacci 序列生成代码解析

### 1. 功能

这段代码实现了一个生成斐波那契数列的函数，输入一个正整数  $n$ ，返回包含前  $n$  个斐波那契数的列表。

### 2. 关键步骤解析

#### (1) 输入验证

检查  $n$  是否为正整数，如果不是则返回错误信息。

#### (2) 基础情况处理

$n=1$  时返回  $[0]$ ，

$n=2$  时返回  $[0, 1]$ 。

#### (3) 序列生成

对于  $n>2$  的情况，初始化序列为  $[0, 1]$ ，

使用循环从第 3 项开始计算，每次将最后两项相加作为新项，循环直到生成  $n$  个斐波那契数。

### 3. 潜在优化点

#### (1) 输入验证改进

当前对非正整数返回字符串，可能引发类型不一致问题。

建议抛出异常更符合 Python 惯例：

```
if not isinstance(n, int) or n <= 0:
 raise ValueError("输入必须是正整数")
```

#### (2) 空间优化

对于大  $n$  值，可以改用生成器来节省内存。

示例：

```
def fibonacci_gen(n):
 a, b = 0, 1
 for _ in range(n):
 yield a
 a, b = b, a + b
```

(3) 性能优化

对于频繁调用的场景，可以使用缓存或记忆化技术。

示例使用 lru\_cache：

```
from functools import lru_cache
@lru_cache(maxsize=None)
def fib(n):
 if n < 2:
 return n
 return fib(n-1) + fib(n-2)
```

(4) 数学优化

对于极大 n 值，可以使用矩阵快速幂或比奈公式来获得  $O(1)$  时间复杂度。

边界情况处理：

当前代码处理了  $n=0$  的情况，但返回类型与成功情况不一致，

考虑是否接受  $n=0$  并返回空列表。

(5) 代码风格：

可以添加文档字符串说明函数用途和参数。

可以考虑使用元组解包使代码更简洁：

```
seq.append(seq[-2] + seq[-1])
```

这个实现对于中等大小的 n 值已经足够高效 ( $O(n)$  时间复杂度)，但如果需要处理极大 n 值或需要频繁调用，则可以考虑上述优化方案。

### 3. 錯誤修復模板

用途：當代碼出現錯誤時，提供錯誤信息和代碼片段，讓AI分析錯誤原因並提供修復建議。

DeepSeek提示詞模板：

我的 [语言名称] 代碼在运行时出现以下错误：

[粘貼錯誤信息]

这是相关的代碼片段：

[在此处粘貼代碼]

请帮我分析错误原因，并提供修复建议或修复后的代碼。

DeepSeek提示詞示例：

我在 python 中运行出现一下的错误

```
IndexError: list index out of range
```

这是相关的代码片段：

```
arr = [1, 2, 3]
```

```
print(arr[3])
```

请帮我分析错误原因，并提供修复建议或修复后的代码。

DeepSeek生成结果示例：

好的，我们来分析这个问题。

错误原因分析：

错误信息 `IndexError: list index out of range` 表明你正在尝试访问一个列表中不存在的索引位置。

在你的代码中：

```
arr = [1, 2, 3]
print(arr[3])
```

列表 `arr` 包含了 3 个元素。

在 Python (以及大多数编程语言) 中，列表的索引是从 0 开始计算的。

因此，`arr` 的有效索引是 0, 1, 2。分别对应的值是 1, 2, 3。

当你尝试访问 `arr[3]` 时，这个索引超出了列表的有效范围 (0 到 2)，所以 Python 抛出了 `IndexError`。

修复建议与修复后的代码：

有几种修复或改进的方式，取决于你的具体意图：

1. 访问最后一个元素：如果你的意图是访问列表的最后一个元素，应该使用索引 `len(arr) - 1`。

```
arr = [1, 2, 3]
last_index = len(arr) - 1
if last_index >= 0:
 print(arr[last_index]) # 输出 3
else:
 print("列表为空，无法访问元素")
或者更 Pythonic 的方式访问最后一个元素
if arr:
 print(arr[-1]) # 输出 3
else:
 print("列表为空")
```

2. 检查索引有效性：在访问之前检查索引是否在有效范围内。

```
arr = [1, 2, 3]
index_to_access = 3 # 你想访问的索引
if 0 <= index_to_access < len(arr):
 print(arr[index_to_access])
else:
 print(f"错误：索引 {index_to_access} 超出列表范围 (0 到 {len(arr)-1})")
```

3. 使用异常处理 (Try-Except)：如果不确定索引是否有效，可以使用 `try-except` 块来捕获可能的 `IndexError`。

```
arr = [1, 2, 3]
index_to_access = 3
try:
 print(arr[index_to_access])
except IndexError:
 print(f"捕获到错误：索引 {index_to_access} 超出列表范围。")
```

总结：

最直接的原因是你试图访问的索引 3 对于只有 3 个元素 (索引 0,1,2) 的列表 `arr` 来说是无效的。请根据你的实际需要选择上述合适的修复方法。

## 4. 學習新知識模板

用途：讓AI用通俗易懂的語言介紹某個技術概念，並提供入門示例代碼。

DeepSeek提示詞模板：

我想学习 [技术 / 库 / 概念名称]，请用简洁明了的语言解释它的核心思想，并提供一个简单的 [语言名称] 入门示例代码。

DeepSeek提示詞示例：

我想学习 Pandas 库，请用简洁明了的语言解释它的核心思想，并提供一个简单的 Python 入门示例代码。

DeepSeek生成結果示例：

### Pandas 核心思想解释

Pandas 是 Python 语言中一个非常强大且流行的数据分析和处理库。你可以把它想象成 Python 世界里的超级加强版 Excel。

它的核心思想是提供了两种主要的数据结构，让你能高效、便捷地操作结构化数据（就像电子表格或数据库中的表）：

- Series（系列）：像是一维的带标签数组，可以理解为表格中的一列。每个元素都有一个标签（索引），默认是 0, 1, 2, …，但也可以自定义。
- DataFrame（数据框）：像是一个二维的带标签表格，可以理解为整个电子表格。它有多行多列，每列可以是不同的数据类型（数字、字符串、布尔值等）。它既有行索引，也有列索引（列名）。

Pandas 让你能够轻松地进行以下操作：

- 读取和写入多种格式的数据（如 CSV、Excel、SQL 数据库等）。
- 查看、选择和过滤数据（按行、按列、按条件）。
- 处理缺失数据（填充或删除）。
- 数据转换和清洗（如更改数据类型、重命名列）。
- 合并、连接和重塑数据表。
- 进行分组聚合计算（类似 Excel 的数据透视表）。
- 处理时间序列数据。
- 进行简单的数据可视化。

总之，Pandas 的目标是让 Python 成为强大而高效的数据分析环境，极大简化数据处理的流程。

Pandas Python 入门示例代码：

```
导入 Pandas 库，通常约定俗成地简写为 pd
import pandas as pd
--- 创建 DataFrame ---
假设我们有一些数据，比如几个人的姓名、年龄和城市
data = {
 '姓名': ['爱丽丝', '鲍勃', '查理', '黛安娜'],
 '年龄': [25, 30, 22, 35],
 '城市': ['北京', '上海', '广州', '上海']
}
使用字典创建 DataFrame，字典的键成为列名，值（列表）成为列数据
df = pd.DataFrame(data)
--- 查看 DataFrame ---
```

```

print("--- 原始 DataFrame ---")
print(df)
输出:
--- 原始 DataFrame ---
姓名 年龄 城市
0 爱丽丝 25 北京
1 鲍勃 30 上海
2 查理 22 广州
3 黛安娜 35 上海
print("\n--- DataFrame 的基本信息 ---")
.info() 方法可以查看 DataFrame 的概览信息, 如行数、列数、每列的数据类型、非
空值数量等
df.info()
print("\n--- 查看前几行数据 ---")
.head(n) 方法默认显示前 5 行, 可以指定 n
print(df.head(2))
输出:
--- 查看前几行数据 ---
姓名 年龄 城市
0 爱丽丝 25 北京
1 鲍勃 30 上海
--- 选择数据 ---
print("\n--- 选择 '年龄' 这一列 (Series) ---")
选择单列会得到一个 Series
ages = df['年龄']
print(ages)
输出:
--- 选择 '年龄' 这一列 (Series) ---
0 25
1 30
2 22
3 35
Name: 年龄, dtype: int64
print("\n--- 选择多列 ('姓名' 和 '城市') ---")
选择多列会得到一个新的 DataFrame
subset = df[['姓名', '城市']]
print(subset)
输出:
--- 选择多列 ('姓名' 和 '城市') ---
姓名 城市
0 爱丽丝 北京
1 鲍勃 上海
2 查理 广州
3 黛安娜 上海
print("\n--- 按条件过滤数据 (年龄大于 25 岁的人) ---")
使用布尔索引进行过滤
older_people = df[df['年龄'] > 25]
print(older_people)

```

```

输出：
--- 按条件过滤数据（年龄大于 25 岁的人） ---
姓名 年龄 城市
1 鲍勃 30 上海
3 黛安娜 35 上海
--- 计算统计量 ---
print("\n--- 计算平均年龄 ---")
对 '年龄' 列调用 .mean() 方法计算平均值
average_age = df['年龄'].mean()
print(f"平均年龄是：{average_age}")
输出：
--- 计算平均年龄 ---
平均年龄是：28.0
print("\n--- 按城市分组并计算每个城市的平均年龄 ---")
.groupby('列名') 进行分组，然后可以对分组后的对象进行聚合计算
average_age_by_city = df.groupby('城市')['年龄'].mean()
print(average_age_by_city)
输出：
--- 按城市分组并计算每个城市的平均年龄 ---
城市
上海 32.5
北京 25.0
广州 22.0
Name: 年龄, dtype: float64
--- 添加新列 ---
print("\n--- 添加一个表示是否成年的新列 ---")
根据 '年龄' 列创建一个新列 '是否成年'
df['是否成年'] = df['年龄'] >= 18
print(df)
输出：
--- 添加一个表示是否成年的新列 ---
姓名 年龄 城市 是否成年
0 爱丽丝 25 北京 True
1 鲍勃 30 上海 True
2 查理 22 广州 True
3 黛安娜 35 上海 True

```

这个例子展示了 Pandas 的一些基本操作，包括创建数据、查看数据、选择数据、过滤数据、计算统计量以及添加新列。希望这个解释和示例能帮助你入门 Pandas。

## 2.5.2 高級提示詞技巧

在實際應用中，除了使用基礎模板，我們還可以利用一些高級技巧來優化提示詞的效果，讓AI生成更符合需求的內容。見附錄B的內容。

## 1.精細化需求描述

當AI的回答不夠準確時，可能是提示詞過於模糊，你的提示詞越清晰，對問題的描述越具體，DeepSeek才能更準確地解決你遇到的問題。下面是一些精細化需求描述的示例。

### 示例1：模糊vs具體

不清晰的提示詞：

```
写一个机器学习代码示例。
```

具體的提示詞：

```
请用 Python 编写一个使用 scikit-learn 进行分类任务的机器学习代码示例，要求使用决策树 (DecisionTreeClassifier) 在鸢尾花数据集 (Iris dataset) 上训练，并评估准确率。
```

### 示例2：添加背景信息

不清晰的提示詞：

```
帮我优化这个 SQL 查询。
```

具體的提示詞：

```
我正在优化一个 SQL 查询，以在 MySQL 数据库上更高效地检索最近 30 天内活跃用户的列表。用户数据存储在 users 表中，活跃日志存储在 activity_logs 表中，字段如下：
```

```
- users (id INT PRIMARY KEY, name VARCHAR(255), email VARCHAR(255), created_at TIMESTAMP)
```

```
- activity_logs (id INT PRIMARY KEY, user_id INT, activity_type VARCHAR(50), timestamp TIMESTAMP)
```

```
activity_logs 表中的 user_id 列有索引，timestamp 列也有索引。users 表的 id 是主键。数据量：users 表约 100 万行，activity_logs 表约 5000 万行。
```

```
请优化以下 SQL 查询，使其运行更快：
```

```
SELECT DISTINCT users.id, users.name
```

```
FROM users
```

```
JOIN activity_logs ON users.id = activity_logs.user_id
```

```
WHERE activity_logs.timestamp >= NOW() - INTERVAL 30 DAY; "
```

## 2.指定輸出格式

示例1：列表格式

不清晰的提示詞：

列举 Python 适合的数据分析库。

具體的提示詞：

请列出 5 个 Python 适用于数据分析的库，并简要介绍它们的功能，格式如下：

1. [库名称]：[功能介绍]
2. ……

DeepSeek生成結果示例：

好的，以下是 5 个 Python 中适用于数据分析的核心库及其功能介绍：

**Pandas**：提供高性能、易用的数据结构（如 DataFrame）和数据分析工具，用于数据清洗、处理、转换、合并和分析结构化数据。

**NumPy (Numerical Python)**：Python 科学计算的基础包，提供强大的 N 维数组对象、复杂的（广播）函数、线性代数、傅里叶变换和随机数生成等功能。Pandas 等许多库都构建于 NumPy 之上。

**Matplotlib**：一个基础的 Python 2D 绘图库，能够以多种硬拷贝格式和跨平台的交互式环境生成出版质量级别的图形。用于创建静态、动态和交互式的可视化图表。

**Seaborn**：基于 Matplotlib 的高级数据可视化库，提供了一个绘制统计图形的高级接口。它使得创建信息丰富且美观的统计图表更加容易。

**Scikit-learn**：一个功能强大的 Python 机器学习库，包含了各种分类、回归、聚类、降维、模型选择和预处理的工具。非常适合进行预测性数据分析。

示例2：代碼+解釋

不清晰的提示詞：

写个爬虫代码。

具體的提示詞：

请用 Python 编写一个使用 requests 和 BeautifulSoup 爬取网页标题的爬虫，目标 URL（统一资源定位符）是 'https://example.com'，并为代码添加详细的注释以解释每一步的作用。

## DeepSeek生成結果示例：

```
导入所需的库
import requests # 用于发送 HTTP (超文本传输协议) 请求, 获取网页内容
from bs4 import BeautifulSoup # 用于解析 HTML (超文本标记语言) 内容, 提取所需信息

定义目标网页的 URL
url = "https://example.com"
try:
 # 发送 GET 请求到目标 URL
 # requests.get() 会返回一个 Response 对象, 其中包含服务器的响应信息
 response = requests.get(url)
 # 检查请求是否成功 (状态码 200 表示成功)
 response.raise_for_status() # 如果请求失败 (如 404, 500), 这行会抛出异常
 # 使用 BeautifulSoup 解析网页的 HTML 内容
 # response.text 包含了网页的 HTML 源代码
 # 'html.parser' 是 Python 内置的 HTML 解析器, 也可以使用 'lxml' (需要安装) 等其他解析器
 soup = BeautifulSoup(response.text, "html.parser")
 # 查找网页的 <title> 标签
 # soup.title 会返回第一个找到的 <title> 标签对象
 title_tag = soup.title

 # 检查是否找到了 <title> 标签
 if title_tag:
 # .string 属性可以获取标签内的文本内容
 title_text = title_tag.string
 # 打印提取到的网页标题
 print(f"网页标题是: {title_text}")
 else:
 # 如果网页没有 <title> 标签
 print("未找到网页标题标签。")
 处理可能发生的请求异常 (例如网络问题、URL 错误等)
 except requests.exceptions.RequestException as e:
 print(f"请求网页时出错: {e}")
 处理其他可能的异常 (例如解析 HTML 时出错)
 except Exception as e:
 print(f"处理过程中发生错误: {e}")
```

### 3.控制輸出的詳細程度

有時，我們希望AI生成簡潔或詳細的回答，這可以通過指定詳細級別來控制。

## DeepSeek提示詞示例：

请用 Python 解释 `map()` 函数的用法，并提供：

1. **\*\* 简短版 \*\*** (一句话说明其核心功能)
2. **\*\* 详细版 \*\*** (包括其工作原理、参数说明、至少两个不同类型的示例代码和常见的应用场景)

## DeepSeek生成結果示例：

### 1. 简短版

`map()` 函数将一个指定的函数应用于一个或多个可迭代对象（如列表、元组）的每一个元素，并返回一个包含结果的迭代器。

### 2. 详细版

#### 工作原理：

`map()` 函数接收至少两个参数：一个函数 `function` 和一个或多个可迭代对象 `iterable(s)`。它会遍历每个可迭代对象中的元素，将这些元素（如果提供了多个可迭代对象，则每次从每个对象中取一个元素组成元组）作为参数传递给 `function`，然后收集每次函数调用的返回值。最后，`map()` 返回一个 `map` 对象，这是一个迭代器，包含了所有这些返回值。你需要将其转换为列表或其他类型才能直接查看所有结果。

#### 参数说明：

`map(function, iterable1, iterable2, ...)`

`function`: 一个函数对象，将应用于每个元素的函数。

`iterable1, iterable2, ...`: 一个或多个可迭代对象（如 `list`, `tuple`, `string` 等）。如果提供多个可迭代对象，它们的长度应该相同（虽然 `map` 会在最短的可迭代对象耗尽时停止）。`function` 需要能接受相应数量的参数。

示例代码 1: 应用于单个可迭代对象

# 示例: 将列表中的每个数字进行求平方运算

```
numbers = [1, 2, 3, 4, 5]
```

```

定义一个平方函数
def square(x):
 return x * x
使用 map() 将 square 函数应用于 numbers 列表的每个元素
squared_iterator = map(square, numbers)
map() 返回的是一个迭代器，需要转换为列表才能看到所有结果
squared_numbers = list(squared_iterator)
print(f"原始列表: {numbers}")
print(f"平方后的列表: {squared_numbers}")
输出:
原始列表: [1, 2, 3, 4, 5]
平方后的列表: [1, 4, 9, 16, 25]
也可以使用 lambda 函数简化
squared_lambda = list(map(lambda x: x * x, numbers))
print(f"使用 lambda 平方: {squared_lambda}")
输出:
使用 lambda 平方: [1, 4, 9, 16, 25]
示例代码 2: 应用于多个可迭代对象
示例: 将两个列表对应位置的元素相加
list1 = [1, 2, 3]
list2 = [4, 5, 6]
定义一个相加函数 (或使用 lambda)
def add_elements(x, y):
 return x + y
使用 map() 将 add_elements 函数应用于 list1 和 list2 的对应元素
sum_iterator = map(add_elements, list1, list2)
sum_list = list(sum_iterator)
print(f"列表 1: {list1}")
print(f"列表 2: {list2}")
print(f"对应元素相加后的列表: {sum_list}")
输出:
列表 1: [1, 2, 3]
列表 2: [4, 5, 6]
对应元素相加后的列表: [5, 7, 9]
使用 lambda 简化
sum_lambda = list(map(lambda x, y: x + y, list1, list2))
print(f"使用 lambda 相加: {sum_lambda}")
输出:
使用 lambda 相加: [5, 7, 9]

```

常见应用场景:

数据转换 / 规范化: 对列表中的所有字符串进行小写转换、去除空白, 或将所有数字转换为特定格式。

批量计算: 对一系列输入值执行相同的数学运算。

特征工程: 在机器学习中, 对数据集的多个特征列应用相同的转换函数。

结合 filter() 或列表推导式进行更复杂的数据处理。

map() 提供了一种简洁、函数式的编程方式来处理序列数据, 通常比显式 for 循环更紧凑, 尤其在结合 lambda 函数时。它适用于批量转换数据, 如数据清洗、文本处理等场景。

## 4.使用多步提示詞

有時，我們希望AI一步步生成內容，而不是一次性提供完整答案。我們可以使用分步提示來控制AI的思考方式。

DeepSeek提示詞示例：

请分步骤完成以下任务：  
第一步：请解释深度学习中的 dropout 机制是什么，以及它的主要作用。  
第二步：请简要介绍如何在 TensorFlow（使用 Keras API）和 PyTorch 中实现 dropout 层。  
第三步：请提供一个简单的 Python 代码示例（可以选择 TensorFlow 或 PyTorch），展示如何在构建一个小型神经网络模型时应用 dropout。  
请按顺序回答这三个步骤。

DeepSeek生成結果示例：

### 第一步: Dropout 机制及其作用

Dropout 是一种在深度学习神经网络训练过程中使用的正则化 (Regularization) 技术。机制:

在每次训练迭代 (或每个训练批次) 中, Dropout 会随机地将网络中一部分神经元的输出暂时\*\*“丢弃”或设置为零\*\*。这意味着这些被丢弃的神经元在本次前向传播和反向传播中都不会对网络的学习做出贡献。哪些神经元被丢弃是随机决定的, 通常每个神经元都有一个预设的概率  $p$  (称为 dropout rate 或丢弃率) 被丢弃。

主要作用:

Dropout 的主要目的是防止神经网络过拟合 (Overfitting)。过拟合是指模型在训练数据上表现很好, 但在未见过的测试数据上表现较差的现象。

Dropout 通过以下方式减轻过拟合:

强制网络学习更鲁棒的特征: 由于每次训练时网络结构都在随机变化 (因为神经元被随机丢弃), 单个神经元不能过分依赖于其他特定神经元的输出。这迫使网络学习到更加分散和冗余的特征表示, 从而提高了模型的泛化能力。

模型集成效果 (Ensemble Effect): 可以认为每次应用 Dropout 后的网络都是原始网络的一个“稀疏”子网络。在整个训练过程中, 模型相当于训练了大量不同的子网络。在测试阶段 (或预测时), 通常会保留所有神经元 (不进行 dropout), 但会将它们的输出按 dropout rate  $p$  进行缩放 (或者在训练时进行反向缩放, 称为 Inverted Dropout), 这近似于对所有这些子网络的预测结果进行平均, 从而获得类似模型集成的效果, 提升性能。

简单来说, Dropout 通过在训练中引入随机性, 降低了神经元之间复杂的协同适应关系, 使得模型不易过度拟合训练数据的噪声和特性。

第二步: 如何实现 Dropout ?

```markdown

****第二步: 在 TensorFlow (Keras) 和 PyTorch 中实现 Dropout****

在主流的深度学习框架中, 实现 Dropout 层通常非常直接:

1. ****TensorFlow (使用 Keras API)****:

- 在 Keras 中, 可以通过 `tf.keras.layers.Dropout` 类来添加 Dropout 层。
- 你需要指定 `rate` 参数, 它表示需要丢弃的神经元比例 (即前面提到的 dropout rate `p`)。
- 示例:

```python

```
import tensorflow as tf
添加一个 Dropout 层, 丢弃率为 0.5 (50%)
dropout_layer = tf.keras.layers.Dropout(rate=0.5)
...
```

- 这个层通常添加在需要进行 Dropout 的层 (如全连接层或卷积层) 的 \*\*后面\*\*。
- Dropout 只在 \*\*训练阶段\*\* 起作用, 在评估或预测阶段会自动失效。

## 2. \*\*PyTorch\*\*:

- 在 PyTorch 中, 可以通过 torch.nn.Dropout 类来添加 Dropout 层。
- 你需要指定 p 参数, 同样表示丢弃概率。
- 示例:

```
```python
import torch
import torch.nn as nn
# 添加一个 Dropout 层, 丢弃率为 0.5
dropout_layer = nn.Dropout(p=0.5)
```
```

- 与 Keras 类似, 这个层也通常放在需要应用 Dropout 的层之后。同样, 它只在模型处于 \*\*训练模式\*\* (model.train()) 时生效, 在评估模式 (model.eval()) 下会自动禁用。

这两个框架都很好地对封装了 Dropout 的实现细节, 开发者只需要在模型结构中插入相应的 Dropout 层并设置好丢弃率即可。

### 第三步: 代码示例 (PyTorch)

**\*\*第三步: 在 PyTorch 中应用 Dropout 的代码示例\*\***

下面是一个使用 PyTorch 构建一个简单的前馈神经网络, 并在隐藏层之后应用 Dropout 的例子。假设我们正在处理一个输入特征为 784 维 (例如 MNIST 图片展平后), 输出为 10 类 (数字 0 ~ 9) 的分类问题。

```
```python
import torch
import torch.nn as nn

class SimpleNetWithDropout(nn.Module):
    def __init__(self, input_size=784, hidden_size=128, output_size=10, dropout_rate=0.5):
        super(SimpleNetWithDropout, self).__init__() # 调用父类的构造函数
        # 定义第一个全连接层 (输入层 -> 隐藏层)
        self.fc1 = nn.Linear(input_size, hidden_size)
        # 定义 ReLU 激活函数
        self.relu = nn.ReLU()
        # 定义 Dropout 层, 设置丢弃率
        self.dropout = nn.Dropout(p=dropout_rate)
        # 定义第二个全连接层 (隐藏层 -> 输出层)
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # 定义数据在网络中的前向传播路径
        # 输入数据通过第一个全连接层
        out = self.fc1(x)
        # 应用 ReLU 激活函数
        out = self.relu(out)
```

```

        # 应用 Dropout 层 (只在 model.train() 模式下生效)
        out = self.dropout(out)
        # 通过第二个全连接层得到最终输出
        out = self.fc2(out)
        return out
# --- 如何使用 ---
# 设定模型参数
input_dim = 784
hidden_dim = 128
output_dim = 10
dropout_p = 0.5 # 设置丢弃率为 50%
# 创建模型实例
model = SimpleNetWithDropout(input_size=input_dim,
                              hidden_size=hidden_dim,
                              output_size=output_dim,
                              dropout_rate=dropout_p)

# 打印模型结构
print(model)
# 模拟输入数据 (例如一个批次包含 64 个样本)
dummy_input = torch.randn(64, input_dim)
# --- 训练阶段 ---
# 将模型设置为训练模式, Dropout 会生效
model.train()
# 进行前向传播
output_train = model(dummy_input)
print("\n 训练模式下输出形状:", output_train.shape) # 输出应为 torch.
                                                    Size([64, 10])

# --- 评估 / 预测阶段 ---
# 将模型设置为评估模式, Dropout 会自动禁用
model.eval()
# 进行前向传播
output_eval = model(dummy_input)
print(" 评估模式下输出形状:", output_eval.shape) # 输出应为 torch.Size([64, 10])

```

注意

在實際訓練中，你還需要定義損失函數和優化器，並進行反向傳播和參數更新。這裡僅演示了Dropout層在模型結構中的位置和如何在不同模式下工作，具體包括如何在神經網絡的隱藏層之後插入一個nn.Dropout層，並通過model.train()和model.eval()來控制Dropout是否激活。

5.概念學習模板

無論何時何地，當你遇到不懂的概念、不清楚的操作步驟或是令人頭疼的錯誤時，都可以打開像DeepSeek Chat（或者我們也強烈推薦的“騰訊混元”等）這樣的AI對話工具，嘗試使用以下精心設計的提問模板。

DeepSeek提示詞模板：

你好！我是一名编程初学者（小白），对【概念名称】这个概念不太理解。你能用尽可能通俗易懂、简单直白的语言向我解释一下它是什么意思吗？如果能结合一些日常生活中的例子来帮助我理解，那就更好了！谢谢！

DeepSeek提示詞示例：

你好！我是一名编程初学者（小白），对‘变量 (Variable)’这个概念不太理解。你能用尽可能通俗易懂、简单直白的语言向我解释一下它是什么意思吗？如果能结合一些日常生活中的例子来帮助我理解，那就更好了！谢谢！

DeepSeek生成結果示例：

当然可以！让我们用最生活化的方式来理解变量 (Variable) 这个概念。

🔴 一句话解释变量：

变量就像是一个贴了标签的盒子，你可以往盒子里放东西（数据），并且随时可以打开盒子查看或更换里面的东西。

📦 生活化比喻：

快递柜

每个快递柜的格子都有编号（变量名）

格子里可以放不同的快递（数据）

比如：

格子 A = "一本书" # 今天放的是书

格子 A = "一双鞋" # 明天换成鞋

虽然内容变了，但格子编号（变量名）始终是格子 A

便利贴

你在冰箱上贴一张便利贴写着“今日菜单”（变量名）

今天写“红烧肉”（数据），明天可以改成“清蒸鱼”

就像：

今日菜单 = "红烧肉"

今日菜单 = "清蒸鱼" # 修改内容

钱包里的卡槽

第一个卡槽你贴上“公交卡”的标签（变量名）

今天插的是北京公交卡，明天可以换成上海交通卡（数据）

但卡槽的名字“公交卡”不变

🐍 Python 中的具体例子：

定义一个变量（贴标签 + 放内容）

手机电量 = 80 # 现在电量是 80%

使用变量

print("当前电量：", 手机电量) # 输出：当前电量：80

修改变量（更换盒子里的东西）

手机电量 = 20 # 电量变成 20%

print("充电前电量：", 手机电量) # 输出：充电前电量：20

🔗 关键特点:

随时可变: 就像天气从“晴天”变成“雨天”

天气 = “晴天”

天气 = “雨天” # 变量内容可以修改

名字代表内容:

当你听到“冰箱里的牛奶”, 就知道指的是具体某瓶牛奶

冰箱里的牛奶 = “蒙牛纯牛奶”

先定义后使用:

就像你要先准备一个空盒子(定义变量), 才能往里面放东西

🔗 新手常见疑问:

Q: 为什么要用变量? 直接写数据不行吗?

A: 想象你要记录全班 50 人的考试成绩:

不用变量: 每次都要写 `print("张三分数: 90")`、`print("李四分数: 85")`……

用变量: 只需修改名字, 重复使用相同的逻辑

学生分数 = 90

`print(学生姓名 + "的分数:", 学生分数)`

🔗 总结:

变量就是给数据起个名字, 方便我们:

1. 重复使用

2. 随时修改

3. 让代码更易读(比如 `年龄 = 25` 比直接写 `25` 更清楚)

试着在 Python 里玩一玩变量吧! 比如模拟一个“咖啡温度”变量, 观察它从 80°C 降到 50°C 的过程 - 🍵

6. 操作指導模板

當你不知道如何完成某個具體的編程操作或軟件設置時，可以使用以下模板請求幫助。

DeepSeek提示詞模板：

你好！我是一名编程小白，现在遇到了【具体问题描述，例如：想在 Windows 计算机上安装 Python，但不知道怎么做】。

你能给我一个“傻瓜式”的、一步一步的操作攻略吗？

如果能提供关键步骤的截图示例或者非常详细的文字描述，我会非常感激！

DeepSeek提示詞示例：

你好！我是一名编程小白，现在想在我的 Windows 10 计算机上成功安装最新版本的 Python，但完全不知道从何下手。你能给我一个“傻瓜式”的、一步一步的操作攻略吗？如果能提供关键步骤的截图示例或者非常详细的文字描述，我会非常感激！

DeepSeek生成結果示例：

当然可以！以下是一个超详细的“傻瓜式”Python安装指南，我会用“手把手”的方式带你完成整个过程，就像教朋友一样耐心 😊

📁 第一步：下载 Python 安装包

打开浏览器，访问 Python 官方网站：<https://www.python.org/downloads/>。

你会看到一个大大的黄色按钮 Download Python 3.x.x (版本号可能不同，直接单击它！)

⚠️ 注意：不要单击下面的 Windows x86-64 executable installer 等小字链接，容易选错版本。

🔧 第二步：运行安装程序

双击下载好的安装包 (通常在下载文件夹里，文件名类似 python-3.x.x.exe)，

你会看到这个界面。(两个关键操作！)

✓ 一定要勾选 Add Python 3.x to PATH (这关系到能否在命令行使用 Python)

✓ 单击 Install Now (默认安装最省心)

如果看到安全提示，单击“是”允许安装。

⌚ 第三步：等待安装完成

你会看到进度条，通常需要 1 ~ 3min。喝杯茶等待就好~

安装完成后单击 Close 关闭窗口。

✓ 第四步：验证是否安装成功

按 < Win + R > 键 打开运行窗口，输入 cmd 然后按 < Enter > 键，

在黑色的命令提示符窗口中输入：

```
python --version
```

如果显示 Python 3.x.x (和你下载的版本一致)，说明安装成功！🎉

如果报错，可能是没勾选 Add to PATH，需要卸载重装。

📁 第五步：第一次运行 Python

我们来写个经典的小程序：

在命令提示符中输入：

```
python
```

你会看到 >>> 提示符，输入：

```
print("Hello 我是 Python！")
```

按 < Enter > 键，你会看到程序输出文字！

🔍 常见问题解决

1. 如果输入 python 提示“不是内部命令”：

卸载 Python 重新安装，务必勾选 Add to PATH，

或手动添加 PATH^⓪。

2. 安装时卡住不动：

关闭杀毒软件后重试，

确保 C 盘有至少 2GB 空间。

3. 想卸载重装：

选择“控制面板”→“卸载程序”选项，找到 Python，右键卸载。

🎯 终极确认清单

完成以下所有项就打钩吧！

从官网下载了 Python

安装时勾选了 Add to PATH

命令行输入 python --version 能显示版本号

成功运行了 print("Hello")

如果遇到任何卡住的地方，随时告诉我你的具体问题，我会继续帮你排查！🔍

(需要更详细的截图或视频演示可以告诉我，我可以补充更多细节。)

⓪ 詳細圖文教程見<https://realpython.com/add-python-to-path/>。——編輯注

結合上述示例，提示詞優化點總結如下：

- ❑ 明確目標（爬取名言和作者）。
- ❑ 指定工具（requests，BeautifulSoup4）。
- ❑ 增加約束條件（異常處理、User-Agent）。
- ❑ 指定輸出格式（字典列表）。
- ❑ 細化實現要求（註釋、打印結果）。

通過這種更具體、結構化的提示，AI能夠更好地理解你的需求，生成更完整、更健壯、更符合實際應用場景的代碼。

7.優化內容創作任務

如果提示詞不夠準確，DeepSeek生成的文章內容可能會泛泛而談，缺乏深度和具體的論證。例如，原始DeepSeek提示詞（過於寬泛）。

请写一篇关于人工智能未来发展趋势的文章。

DeepSeek可能的錯誤表現：

- ❑ 文章內容籠統，列舉了一些常見趨勢，但沒有深入分析原因或提供具體案例。
- ❑ 缺乏獨特的見解和批判性思考。
- ❑ 結構可能比較散亂，邏輯不夠清晰。

改進後的DeepSeek提示詞（限定範圍、提供指導）如下：

请撰写一篇 800 字左右的文章，探讨未来五年人工智能在医疗健康领域的三大关键发展趋势。要求：

1. 明确指出并详细阐述这三个关键趋势。
2. 对于每个趋势，需要分析其背后的驱动因素（例如技术进步、政策导向、市场需求）。
3. 为每个趋势提供至少一个具体的应用案例或潜在的影响。
4. 文章结构清晰，包含引言、主体（每个趋势一段落）和结论。
5. 语言表达专业且易于理解，避免使用过于模糊或夸张的词汇。
6. 请简要讨论这些趋势可能带来的伦理和社会影响。
7. 给出一个你认为最重要的挑战以及应对建议。

結合上述示例，提示詞優化點總結如下：

- 限定主題和字數（人工智能在醫療健康領域，800字）。
- 明確輸出數量和要求（三大關鍵趨勢，詳細闡述，分析驅動因素，具體案例）。
- 規定文章結構（引言、主體、結論）。
- 指導寫作風格（專業易懂，避免模糊誇張）。
- 引導思考深度（倫理和社會影響，重要挑戰和應對建議）。

通過更精細的指令和明確的要求，AI將能夠生成更聚焦、更深入、更有條理的文章，更符合用戶的期望。

注意

DeepSeek是你學習路上的強大夥伴和加速器，但它不能替代你主動思考和動手實踐的過程。真正的理解和掌握，來自你親自編寫代碼、調試錯誤、完成項目的經驗積累。

如果遇到困難需要幫助，歡迎添加我們的企業微信服務號“光速財經”。我們有專業的團隊為你答疑解惑，與你一起學習，共同解決問題。

2.6 結語

掌握DeepSeek的使用方法和提示詞技巧，是開啟AI編程之旅的關鍵一步。通過本章的學習，你不僅瞭解了DeepSeek的核心能力，還學會了如何像與一位智能助手溝通一樣，引導它完成從文本生成到複雜代碼編寫的各種任務。記住，高質量的提示是釋放AI潛力的鑰匙。不斷練習、優化你的提問方式，DeepSeek將成為你學習和創造過程中不可或缺的強大夥伴。

第3章

輕鬆準備你的編程環境

本章將指導你通過DeepSeek、Windsurf等開始Python編程之旅，包括搭建Python環境、選擇合適的IDE以及解決安裝過程中可能遇到的問題。通過本章的學習，你將能夠在自己的計算機上成功安裝Python並運行第一個程序。另外本章還將指導你在不同的操作系統（Windows、macOS和Linux）上搭建Python環境，確保無論你使用什麼系統，都能順利開始編程。DeepSeek將幫助你解決在安裝過程中可能遇到的問題，提供清晰的解決方案。即使你之前從未接觸過編程，也能在DeepSeek的幫助下輕鬆完成這些設置步驟。

3.1 本章學習目標

本章的核心目標是為你鋪設堅實的起跑線，確保你能順利地開始AI編程之旅。學習本章，你將掌握以下內容或技能：

- 跨越第一道門檻：成功配置一個基礎的AI開發環境（包括Python、必要的庫和工具），為所有後續的實踐項目掃清技術障礙。

- 掌握開發工具：瞭解市面上常見的AI編程助手（如Windsurf）和傳統IDE（如VS Code，PyCharm）的特點。學會選擇並初步使用適合自己的開發工具，從而提高學習和開發的效率。

- 建立信心基礎：通過親手搭建環境，克服對技術的陌生感和恐懼感。初步建立“我能做到”的積極信念，為後續更深入的學習注入動力。

- 解決常見問題：預先了解並掌握環境配置過程中常見問題（如依賴衝突、路徑錯誤、網絡問題等）的解決方法。讓你在遇到障礙時不再輕易被“卡住”，具備初步的自主排錯能力。

3.2 開發環境概述

在開始編程之前，瞭解並設置合適的開發環境是至關重要的。開發環境不僅僅是安裝一些軟件，它涉及的方方面面都會影響到編程的效率和質量。讓我們從一些基本概念開始，瞭解什麼是開發環境，以及我們為什麼需要專門配置開發環境。

3.2.1 什麼是開發環境

想象一下，搭建開發環境就像為一位大廚準備他的專屬廚房。

- ❑ 操作系統：你的廚房空間本身，定義了你工作的基本場所和佈局。

- ❑ 編程語言：如同你烹飪所需的基礎食材（如麵粉、雞蛋、蔬菜），是構建程序的核心原料（如Python）。

- ❑ 開發工具：好比你使用得得心應手的鍋碗瓢盆、刀具灶臺（如代碼編輯器VS Code、Jupyter Notebook），用於處理食材（編寫、運行、調試代碼）。

- ❑ 代碼庫與框架：就像你擺滿各式香料、醬汁的調料架（如TensorFlow、PyTorch、Scikit-learn），提供了預製的功能模塊和“風味”，能極大豐富你的菜品（程序功能）並加速烹飪（開發）過程。

正如一位優秀的廚師需要一個設備齊全、組織有序、用起來順手的廚房才能高效地創造出美味佳餚，一位AI編程學習者或開

發者也需要一個配置得當、穩定可靠的開發環境來順利地學習、實驗和創造。

3.2.2 為什麼需要專門配置開發環境

“什麼我不能直接在計算機上隨便找個地方就開始寫代碼？為什麼要花時間去‘配置環境’？”

這是許多初學者心中的疑問。答案其實很簡單，配置一個良好的開發環境至關重要，主要基於以下幾點核心原因：

1. 標準化與可復現性

□ 目標：確保你的代碼不僅在你的計算機上能運行，換到任何符合同樣配置環境的地方，都能得到一致的運行結果。

□ 類比：就像遵循一份精確的烹飪食譜，同樣的食材（代碼）、同樣的廚具（依賴庫版本）、同樣的步驟（操作系統配置），才能穩定地做出同樣美味的菜餚（程序正常運行）。

□ 反例：如果沒有標準化的環境，你的代碼很可能出現“在我計算機上明明是好的”，但在別人那裡或部署到服務器時就失敗的窘境。

2. 工具支持與效率提升

□ 目標：獲得現代開發工具提供的各種智能輔助功能，如代碼自動補全、語法高亮、即時錯誤檢查、代碼格式化等。

□ 類比：想象一下開車時的導航系統和駕駛輔助功能，它們不是絕對必需的，但有了它們，你的駕駛過程會更加順暢、安全、高效。

□ 益處：好的開發環境就像給你的編程之旅配備了“智能副駕”，能在你犯錯之前及時提醒，甚至幫你自動修正一些常見問題，極大提升開發效率和代碼質量。

3. 依賴管理

□ 目標：能夠輕鬆地使用和管理由他人開發並分享出來的庫和框架，避免重複造輪子。

□ 類比：就像現代傢俱的組裝，你不需要從砍樹開始製作每一個螺絲和木板，而是可以直接使用設計好的、標準化的模塊，快速組裝出你想要的傢俱。

□ 反例：如果沒有良好的依賴管理機制（如pip、Conda），你就如同在一個沒有超市的世界裡做飯，連最基本的鹽、糖、醬油都需要自己從頭開始提煉，效率極低且容易出錯。

4. 版本控制

□ 目標：能夠系統地跟蹤、管理和回溯你的代碼在不同時間點的變更歷史。

□ 類比：這就像寫論文時常用的“另存為”操作，但功能遠比它強大得多。版本控制系統（如Git）允許你創建多個開發分支、合併修改，並且可以精確地回到過去的任何一個版本。

□ 反例：沒有版本控制，你的開發過程就像在沙灘上寫字，一旦代碼寫錯、誤刪，或者被意外情況（如硬盤損壞）“沖走”，就很難恢復到之前的狀態，風險極高。

就如同你無法在沒有廚房和廚具的情況下烹飪出一桌美食佳餚一樣，你也無法在缺乏一個配置得當的開發環境的情況下，進

行高效、穩定、可維護的編程工作。配置環境，是為你的編程之旅打下堅實地基的關鍵一步。

3.3 開發工具的選擇

1. “1+1”工具策略：AI編程的最佳實踐

在AI賦能的編程新時代，我們強烈推薦採用一種“1+1”的工具組合策略，即一個AI驅動的智能IDE作為你的主力開發環境，同時保留一個強大的傳統代碼編輯器作為備用或特定場景下的補充。這種組合能讓你既能充分利用AI帶來的智能輔助與效率提升，又能保持對代碼細節的完全掌控力與靈活性。

2. 策略1：選擇一個AI驅動的智能IDE

AI驅動的智能IDE不僅僅是編輯器，還像是你的“超級編程助手”。它能深度理解你的編程意圖，主動提供代碼生成、問題診斷、重構建議等高級輔助功能。

□ 推薦：Windsurf。

□ 核心優勢：作為全球首個基於AI Flow範式構建的智能IDE，Windsurf的設計理念使其AI能夠更深入地理解你的整個代碼庫上下文和開發意圖，而不僅僅是當前代碼片段。

□ 內置Cascade AI：其內置的Cascade AI助手由頂尖的Codeium工程團隊研發，在代碼理解、生成質量和複雜任務處理上通常表現出色。

注意

市面上還有其他優秀的AI編程助手或IDE插件，如GitHub Copilot（集成在VS Code等）、Cursor等，可以根據個人偏好和項目需求選擇。

3.策略2：保留一個傳統代碼編輯器

強大的傳統代碼編輯器是你工具箱中的“可靠後盾”。在AI工具暫時不可用、網絡中斷，或者你需要進行極其精細的代碼調整、不受AI干擾時，它們依然是不可或缺的選擇。

- ❑ 推薦：Visual Studio Code (VS Code)。

- ❑ 核心優勢：由微軟開發，免費、開源、輕量且高效。擁有極其豐富的插件生態系統，可以通過安裝插件（包括GitHub Copilot等AI插件）來擴展其功能。

- ❑ 跨平臺與社區：支持Windows，macOS，Linux全平臺，擁有龐大而活躍的開發者社區，遇到問題很容易找到解決方案。

注意

其他優秀的傳統編輯器還包括PyCharm（尤其適合Python開發）、Sublime Text、Atom等。

4.為什麼特別推薦Windsurf作為首選AI IDE

在眾多新興的AI編程工具中，Windsurf展現出一些獨特的優勢，使其成為我們重點推薦的選擇：

- ❑ 革命性的AI Flow範式：不同於簡單的代碼補全，它旨在讓AI真正理解開發流程和上下文，提供更智能、更貼合需求的輔助。

- ❑ 強大的Cascade AI助手：內置的AI引擎在代碼理解深度和生成質量上往往表現更佳。

□ 無縫集成體驗：AI能力與IDE的各項功能（編輯、調試、版本控制等）深度融合，使用體驗更自然流暢。

□ 全面的語言支持：不僅限於Python，也支持JavaScript，Java，Go等多種主流編程語言。

□ 持續更新迭代：由世界級的AI研發團隊支持，能夠快速跟進最新的AI技術進展，保持工具的領先性。

5.為什麼推薦VS Code作為常用編輯器

VS Code之所以成為全球開發者的首選之一，並非偶然：

□ 免費且開源：不需要任何費用，社區驅動發展。

□ 輕量與高效：啟動速度快，資源佔用相對較少。

□ 插件生態無敵：你幾乎可以通過安裝插件將其定製成任何你想要的開發環境，包括集成各種AI能力。

□ 跨平臺一致性：在不同操作系統上提供統一的開發體驗。

□ 活躍社區支持：遇到問題時，大量的教程、文檔和社區討論可以提供幫助。

注意

工具終究只是輔助，你腦中的思路、解決問題的方法以及與AI有效協作的的能力，才是真正的核心競爭力。無論你最終選擇哪種組合工具，本書所介紹的AI編程思維和實踐方法都將同樣適用。選擇讓你感覺最舒適、最高效的工具，然後專注於創造。

3.4 工具的安裝與使用流程

在瞭解了開發環境的基礎知識後，接下來我們將詳細介紹如何安裝和使用Windsurf AI編程環境。這一過程包括從註冊賬號到工具安裝以及初步的使用設置，確保你能夠順利使用Windsurf進行AI編程。

3.4.1 Windsurf安裝指南

以下是安裝和初步使用Windsurf AI編程環境的步驟。

1. 註冊賬號

1) 訪問Windsurf官方網站 (<https://www.windsurf.ai>)，如圖3-1所示。

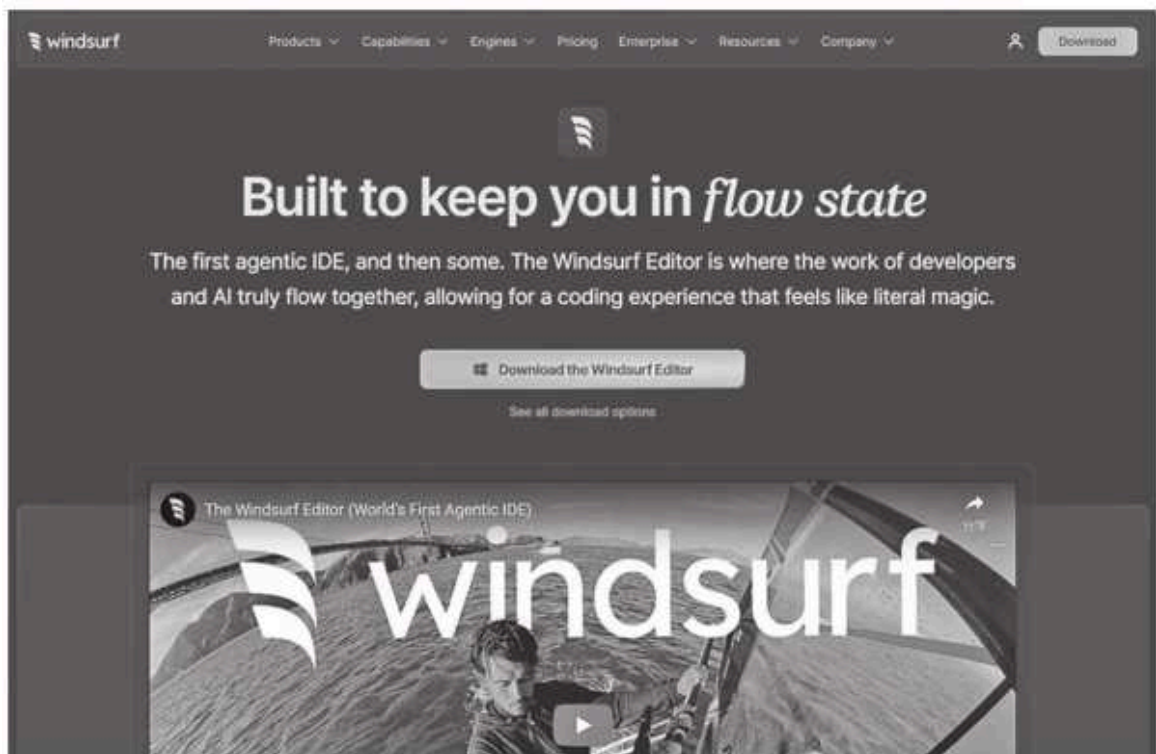


圖3-1 Windsurf官方網站登錄界面

2) 在網站首頁或指定區域找到並單擊Sign Up或“註冊”按鈕，如圖3-2所示。

圖3-2 Windsurf的註冊界面

- 3) 按照頁面提示填寫信息，創建你的個人賬號。
- 4) 檢查你的註冊郵箱，單擊驗證鏈接以完成賬號激活。

注意

某些網絡環境可能會限制對Windsurf官方網站的直接訪問。

□ 如遇訪問困難：建議嘗試使用穩定可靠的特殊網絡服務（例如VPN）。

□ 尋求幫助：你也可以聯繫我們提供的企業微信服務號獲取詳細的設置和訪問指導，或通過搜索引擎查找關於“如何訪問特定國際網站”的相關解決方案。

2. 下載與安裝

- 1) 使用你剛註冊並驗證的賬號登錄Windsurf官方網站。
- 2) 在用戶後臺或下載頁面，找到適合當前操作系統（如Windows、macOS、Linux）的Windsurf安裝包。
- 3) 單擊下載。
- 4) 下載完成後，找到安裝包文件並雙擊運行。
- 5) 遵循安裝嚮導（通常是單擊“下一步”或“安裝”按鈕）完成安裝過程，如圖3-3所示。

3. 首次啟動與登錄

- 1) 安裝完成後，從你的應用程序列表或桌面快捷方式啟動Windsurf。

2) 在首次啟動時，程序會提示你登錄。請輸入你之前註冊的 Windsurf 賬號和密碼（如圖3-4所示）。

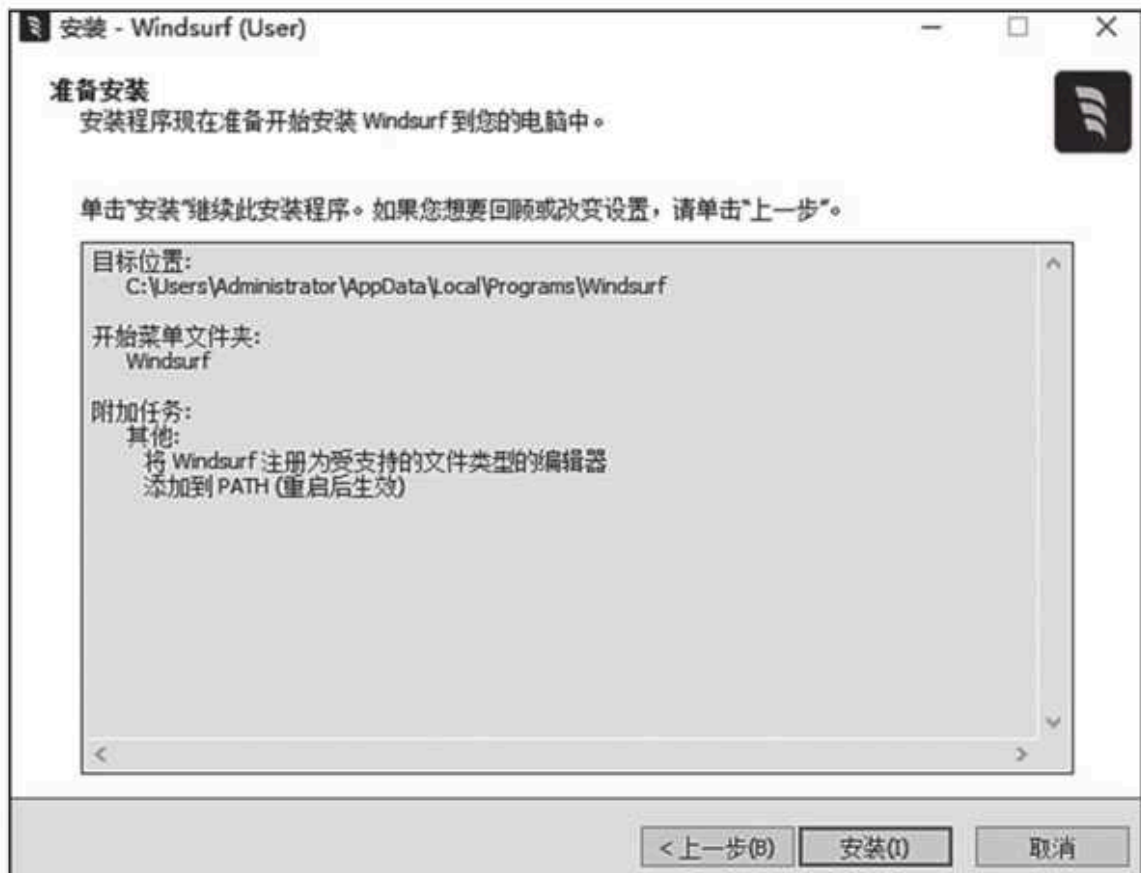


圖3-3 完成Windsurf安裝界面

注：圖中電腦應為計算機。

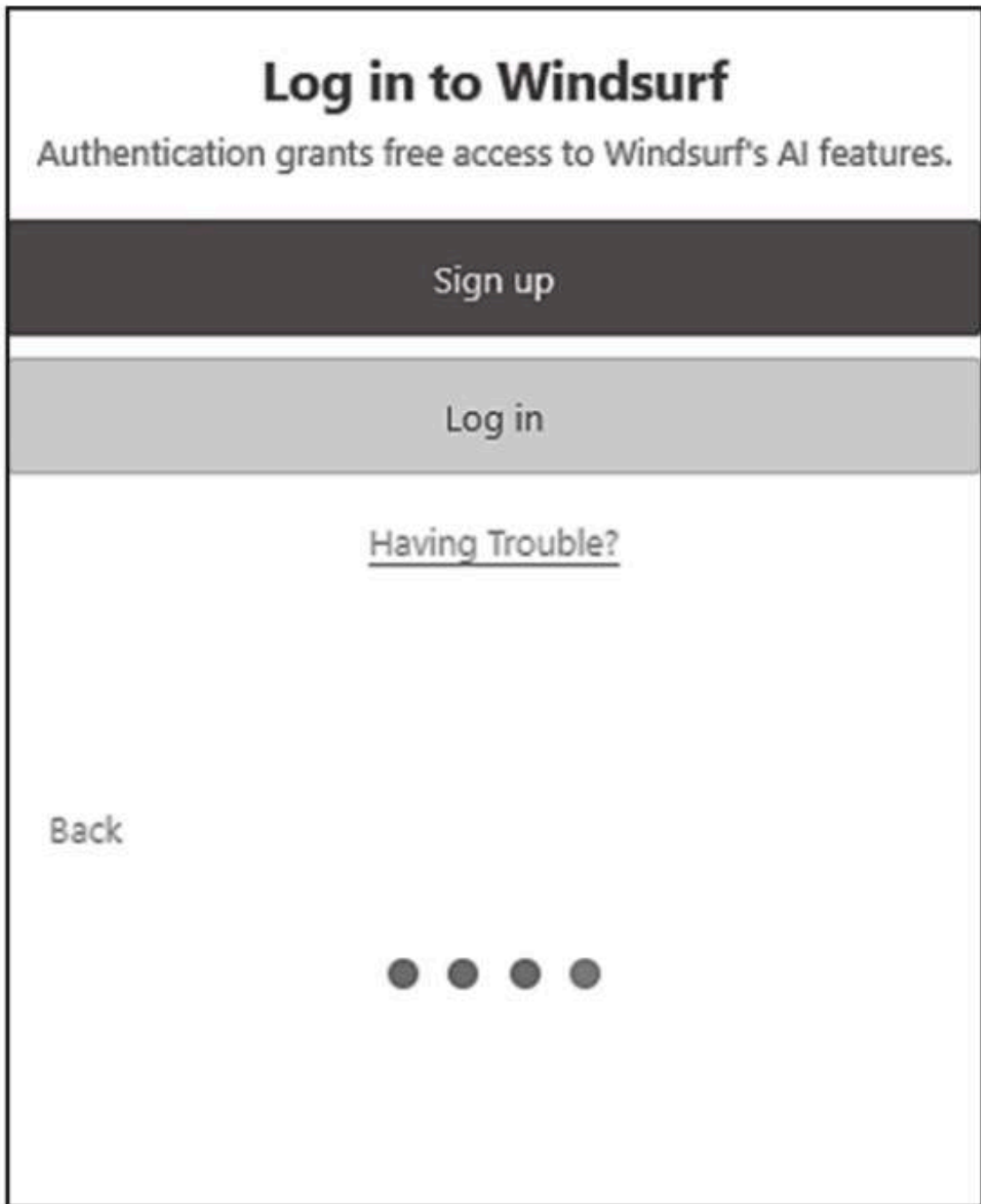


圖3-4 Windsurf提示登錄界面

4.基本設置

1) 個性化界面：瀏覽設置選項，選擇你偏愛的主題（如暗色/亮色模式）和代碼字體。

2) AI助手配置：根據需要調整AI助手的響應風格（例如，更簡潔或更詳細的回答）。

3) 自動保存：檢查並確認是否啟用了代碼自動保存功能，以防意外丟失工作。

5.開始使用

1) 創建或打開項目：在Windsurf中創建一個新的項目文件夾（New Project）或打開一個你本地已有的代碼項目（Open Folder/Project）。

2) 激活AI助手：在代碼編輯區域內，嘗試輸入斜槓符號/，通常這會觸發AI助手或命令面板。

3) 實踐探索：開始嘗試向AI助手下達簡單的指令，例如，“幫我寫一個Python函數，計算兩個數字的和”，以此來熟悉其交互方式和能力，如圖3-5所示。



圖3-5 Windsurf編寫代碼示例

請按照以上步驟操作。如果在安裝或使用過程中遇到任何具體問題，可以參考我們官方文檔或尋求社區/客服支持。

3.4.2 VS Code安裝與基礎配置指南

作為推薦的傳統編輯器（或與AI插件結合使用），以下是VS Code的安裝和基礎配置步驟：

1. 下載與安裝

1) 訪問VS Code官方網站 (<https://code.visualstudio.com/>)，如圖3-6所示。

2) 網站通常會自動檢測你的操作系統並推薦合適的下載版本。單擊下載按鈕獲取安裝包（Windows.exe，macOS.dmg，Linux.deb/.rpm）。

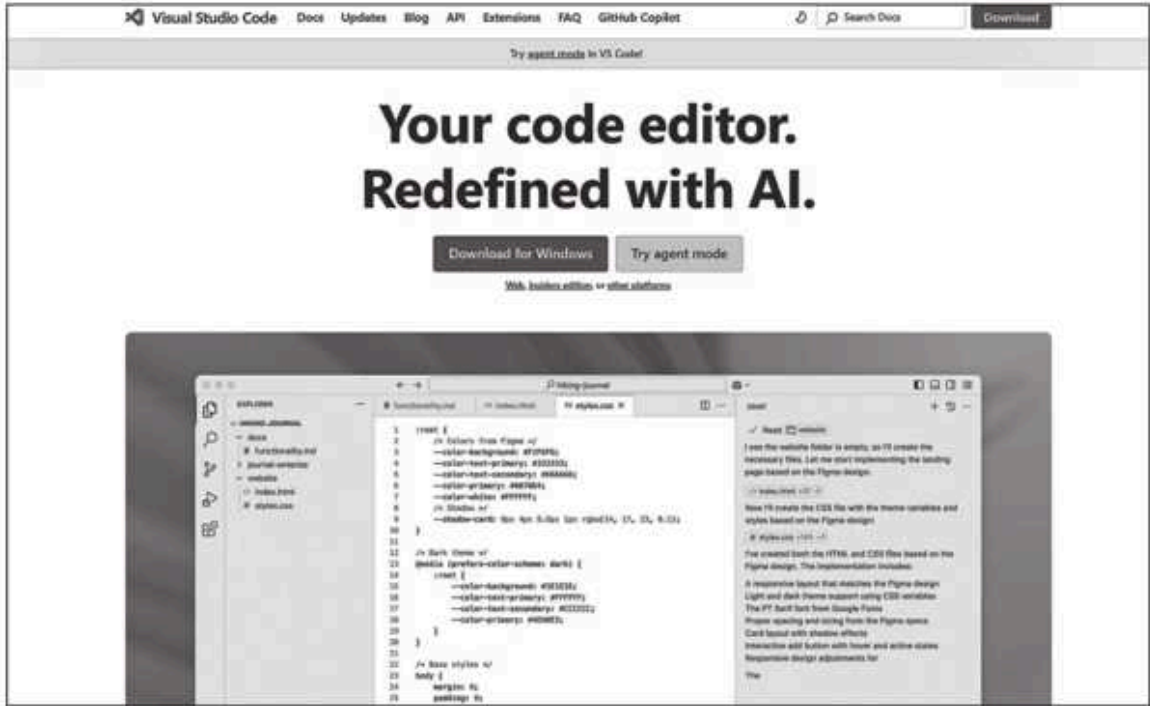


圖3-6 Vs Code官方網站界面

3) 下載完成後，運行安裝程序，按照屏幕上的嚮導提示完成安裝（通常保持默認設置即可）。安裝核心插件（Extensions）。

4) 打開VS Code，單擊左側活動欄中的擴展圖標（通常是四個方塊組成的圖標）。

5) 在搜索框中搜索並安裝以下推薦插件：

- ❑ Python (by Microsoft)：必裝！提供對Python語言的全面支持，包括調試、Linting、IntelliSense等。

- ❑ Python Indent：幫助改善Python代碼的自動縮進，使代碼更規範。

- ❑ Pylance (by Microsoft)：強大的Python語言服務器，提供更快速、更智能的代碼補全、類型檢查和錯誤提示。

❑（可選）GitHub Copilot：如果你需要強大的AI代碼補全和建議功能（可能需要訂閱），可以安裝此插件，如圖3-7所示。

2.基礎用戶設置

1) 你可以通過選擇“文件”(File) →“首選項”(Preferences) →“設置”(Settings) 選項（或使用快捷鍵 <Ctrl+,> / <Cmd+,>）打開設置界面。

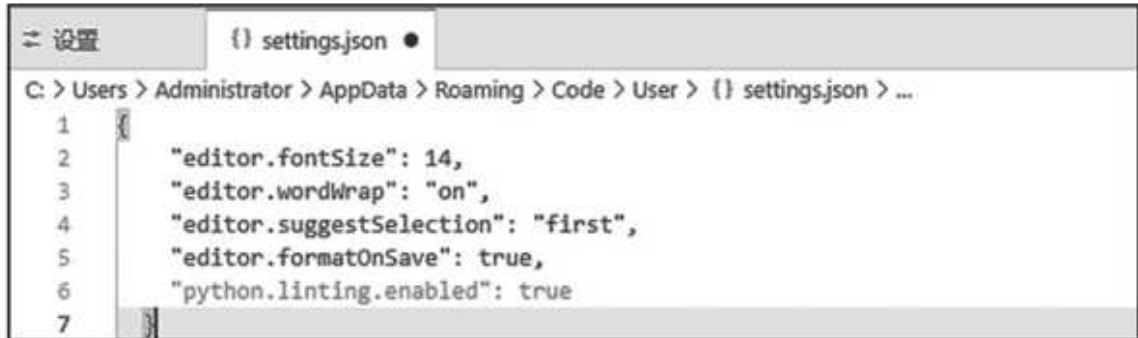


圖3-7 GitHub Copilot安裝界面

2) 切換到JSON視圖（通常右上角有一個{}圖標）可以直接編輯settings.json文件。

3) 以下是一些推薦的基礎設置，你可以將其複製並粘貼到你的settings.json文件中，如圖3-8所示（如果文件已存在內容，請將其添加到已有的{}內）。

```
{
    "editor.fontSize": 14,
    "editor.wordWrap": "on",
    "editor.suggestSelection": "first",
    "editor.formatOnSave": true,
    "python.linting.enabled": true
}
```



The screenshot shows a code editor window with the title bar "设置" and "settings.json". The file path is "C:\> Users > Administrator > AppData > Roaming > Code > User > {} settings.json > ...". The content of the file is the same JSON configuration as shown in the previous block, with line numbers 1 through 7 on the left side of the editor.

圖3-8 settings.json文件

3.配置Python解釋器

1) 確保你的計算機已安裝了Python（推薦使用Python 3.8或更高版本）。你可以從Python官方網站<https://www.python.org/>下載安裝，如圖3-9所示。

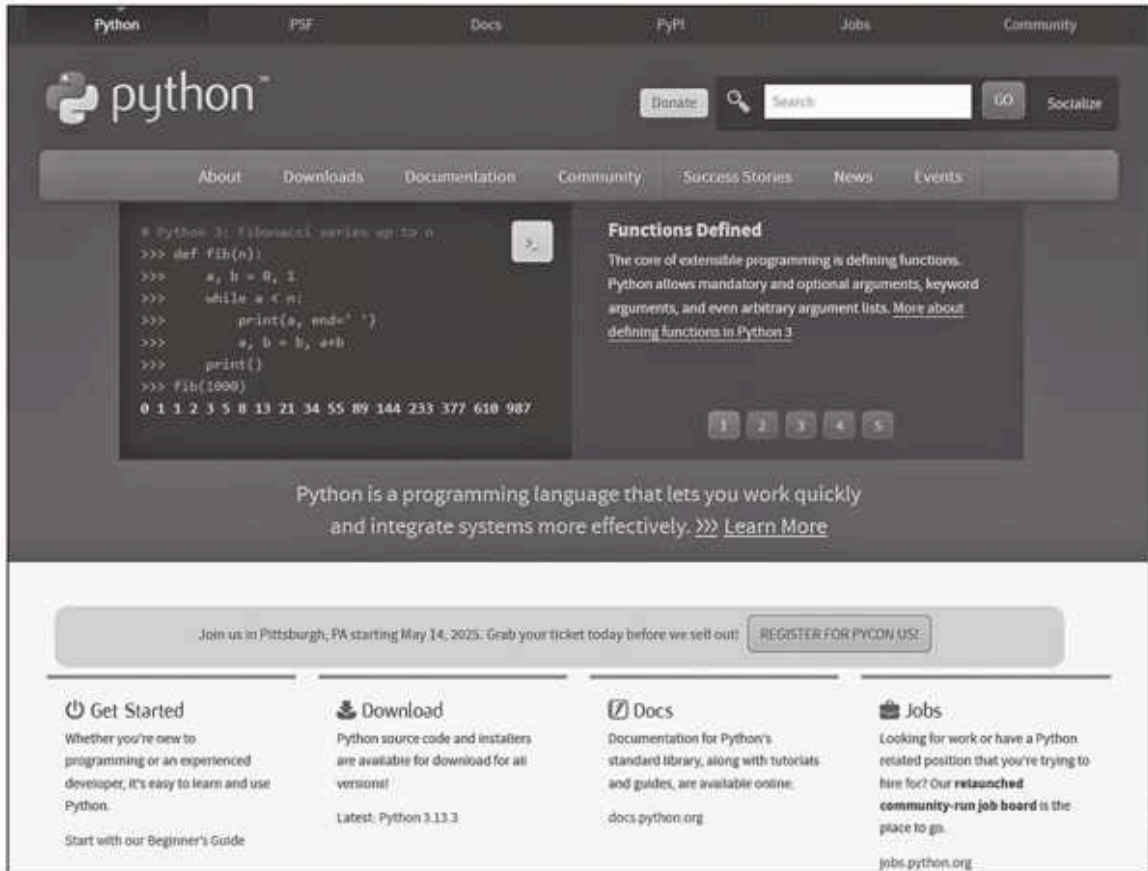


圖3-9 Python官方網站界面

2) 在VS Code中選擇解釋器：

□ 打開一個Python文件（.py）或一個包含Python項目的文件夾。

□ 按快捷鍵〈Ctrl+Shift+P〉（Windows/Linux）或〈Cmd+Shift+P〉（macOS）打開命令面板。

□ 輸入"Python:Select Interpreter"並選擇該命令。

□ VS Code會列出它在你系統上找到的所有Python環境。選擇你希望用於當前項目或工作區的那個Python解釋器。

完成以上步驟後，你的VS Code就基本配置好了，可以開始進行Python開發了。記得根據需要探索和安裝更多有用的插件來擴展其功能。

3.5 常見問題與解決方案

在使用Windsurf的過程中，你可能會遇到一些常見問題。為了幫助你更高效地解決這些問題，下面列出了一些可能出現的情況以及相應的解決方案。

3.5.1 Windsurf的常見問題與解決方案

以下是一些使用Windsurf時可能遇到的問題及其建議的解決方案：

1.問題：Windsurf啟動後白屏或黑屏

解決方案：

❑ 檢查網絡：確保你的網絡連接穩定且可以正常訪問Windsurf所需的服務（可能需要檢查特殊網絡設置）。

❑ 重啟應用：完全關閉Windsurf應用程序，然後重新啟動它。

❑ 清除緩存：在Windsurf的設置（Settings）菜單中，查找類似Clear Cache and Restart或“清除緩存並重啟”的選項，單擊執行。

2.問題：AI助手不響應或響應緩慢

解決方案：

❑ 檢查網絡：確認網絡連接通暢，AI功能通常需要即時在線訪問模型服務。

❑ 優化提示：嘗試使用更簡潔、清晰、具體的指令或問題。避免過於模糊或冗長的描述。

❑ 重啟應用：關閉並重新打開Windsurf。

❑ 檢查服務狀態：(如果可能) 查看Windsurf官方渠道是否有關於服務中斷或維護的通知。

3.問題：賬號登錄失敗

解決方案：

❑ 核對憑證：仔細檢查輸入的用戶名（通常是郵箱）和密碼是否完全正確，注意大小寫。

❑ 檢查網絡環境：確認你的網絡設置是否允許訪問Windsurf的登錄服務器。有時特殊網絡服務（如VPN）的配置也可能影響登錄。

❑ 重置密碼：如果忘記密碼或懷疑密碼錯誤，使用官網提供的“忘記密碼”或Reset Password功能進行重置。

❑ 聯繫客服：如果以上方法都無效，聯繫Windsurf官方客服尋求幫助。

4.問題：AI生成的代碼質量不佳或不符合預期

解決方案：

❑ 提供更豐富的上下文：在提問時，儘量提供相關的背景信息、代碼片段或更詳細的需求描述。AI理解的上下文越多，生成的代碼越可能符合要求。

❑ 優化提示結構：嘗試改變提問的方式，例如使用更明確的動詞，指定編程語言，給出示例格式等。可以參考提示工程的技巧。

❑ 迭代生成：使用Windsurf可能提供的Regenerate、重新生成或類似功能，讓AI基於相同的提示再次嘗試生成，有時會得到更好的結果。

❑ 分步提問：對於複雜任務，可以將其拆解成多個小步驟，逐一向AI提問。

通用建議：保持Windsurf應用程序更新到最新版本，通常新版本會修復已知問題並改進性能。

3.5.2 VS Code的常見問題與解決方案

以下是一些在使用Visual Studio Code（VS Code）進行Python開發時可能遇到的問題及其解決方案：

1.問題：找不到Python解釋器

解決方案：

❑ 確認安裝：確保你的計算機上已經正確安裝了Python。可以在終端或命令提示符中輸入`python--version`或`python3--version`來檢查。

❑ 選擇解釋器：在VS Code中，按〈Ctrl+Shift+P〉鍵（Windows/Linux）或〈Cmd+Shift+P〉鍵（macOS）打開命令面板，輸入並選擇Python:Select Interpreter。

❑ 手動指定：如果所需的Python解釋器沒有自動出現在列表中，首先選擇列表底部的Enter interpreter path選項，然後手動瀏覽並選擇你Python安裝目錄下的python.exe（Windows）或python/python3（macOS/Linux）文件。

2.問題：插件安裝失敗

解決方案：

❑ 檢查網絡：確認你的網絡連接正常，並且可以訪問VS Code Marketplace。有時網絡代理或防火牆設置可能會阻止安裝。

❑ 手動安裝：首先訪問VS Code Marketplace網站，搜索你需要的插件，下載.vsix文件。然後在VS Code的插件側邊欄（Extensions view），單擊右上角的...圖標，選擇Install from VSIX，並選擇下載的文件進行安裝。

❑ 重啟VS Code：有時重啟VS Code就可以解決臨時的安裝問題。

❑ 檢查權限：確保VS Code有權限寫入其安裝和擴展目錄。

3.問題：代碼智能提示不工作或失效

解決方案：

❑ 安裝核心插件：確保你已經安裝了官方的Python擴展（通常包含了Pylance語言服務器）。在插件市場搜索Python並安裝Microsoft發佈的那個版本。

❑ 檢查語言服務器：確認Pylance（或其他你選擇的語言服務器）正在運行且沒有報錯。查看VS Code底部狀態欄或Output面板（選擇Python或Pylance輸出）。

❑ 重新加載窗口：按〈Ctrl+Shift+P〉鍵（或〈Cmd+Shift+P〉鍵），輸入並選擇Developer:Reload Window。

❑ 檢查項目設置：確保工作區或文件夾的根目錄被正確識別，並且.VS Code/settings.json中沒有錯誤的配置。

4.問題：終端中無法運行Python命令

解決方案：

❑ 檢查環境變量：最常見的原因是Python的安裝路徑沒有被添加到系統的PATH環境變量中。

❑ Windows配置：

1) 右鍵單擊“此電腦”(This PC)，選擇“屬性”(Properties) 選項。

2) 單擊“高級系統設置”(Advanced system settings) 按鈕。

3) 單擊“環境變量”(Environment Variables) 按鈕。

4) 在“系統變量”(System variables) 或“用戶變量”(User variables for [YourUsername]) 下找到Path變量，選中並單擊“編輯”(Edit) 選項。

5) 單擊“新建”(New)，添加Python的安裝路徑（例如C:\\Python39\\）以及其Scripts子目錄的路徑（例如C:\\Python39\\Scripts\\）。

6) 一路單擊“確定”按鈕保存更改。重啟VS Code或命令提示符，使更改生效。

❑ macOS/Linux配置：

1) 打開終端 (Terminal)。

2) 編輯你的 shell (終端) 配置文件，通常是 `~/.bash_profile` (Bash) 或 `~/.zshrc` (Zsh)。例如，使用 `nano ~/.zshrc`。

3) 在文件末尾添加類似 `export PATH="/path/to/your/python/bin:$PATH"` 的行 (請將 `/path/to/your/python/bin` 替換為實際的Python bin目錄路徑)。可以使用 `which python3` 來查找路徑。

4) 保存文件 (在nano中按 `<Ctrl+X>` 鍵，然後按 `<Y>` 鍵，再按 `<Enter>` 鍵)。

5) 執行 `source ~/.zshrc` (或對應的配置文件) 或重新打開終端使更改生效。

提示

定期更新VS Code和已安裝的插件，有助於獲得最新的功能和錯誤修復。

3.6 結語

請記住，成功配置好你的開發環境是每一位開發者——無論新手還是資深專家——都必須經歷的“成人禮”。當你順利跨過這道看似普通卻至關重要的門檻時，你實際上已經超越了那90%僅僅停留在“有個好想法”階段的人。

環境配置本身並沒有高深的技術難度，它更多考驗的是你的耐心和細心。如果你在過程中遇到了報錯、兼容性問題或其他障礙，請不要氣餒！

- 視其為常態：這是學習過程中極其正常的一部分。

- 視其為機會：解決這些問題的過程本身就是鍛鍊你排查故障、搜索信息、理解系統運作方式的絕佳機會。每一次成功解決環境問題，都會讓你對工具和技術有更深入的理解，為你未來的編程之路掃清障礙。

現在，你的“駕駛艙”已經調校完畢，油箱加滿（知識儲備中）。準備好握緊方向盤（你的鍵盤和鼠標），踩下油門（開始你的第一個項目），在AI編程的廣闊道路上，開啟你的創造之旅吧！

第4章

用DeepSeek學Python變量和簡單數據類型

在本章中，你將開始探索Python編程的基礎構建塊——變量和簡單數據類型。你將學習如何創建和使用變量，這些變量就像存儲信息的容器。你還將探索如何處理字符串（文本數據）和數字，以及如何使用註釋來讓你的代碼更易於理解。

DeepSeek可以幫助你理解相關概念，同時提供清晰的解釋和實用的代碼示例。當你輸入Python代碼時，DeepSeek還會幫助你識別不同的部分，並在需要時提供建議和改進。

學習本章後，你將掌握Python編程的基本構建塊，為後續學習更復雜的概念和結構奠定堅實的基礎。即使你沒有任何編程經驗，也能通過本章的學習輕鬆入門Python編程。

4.1 本章學習目標

本章的核心目標是為你打下堅實的Python編程基礎，確保你能夠輕鬆掌握編程的基本概念。學習本章，你將掌握以下內容或技能：

- 理解基礎概念：掌握變量和數據類型這兩個Python編程的核心構建塊，建立對編程邏輯的基本認知，為後續學習奠定基礎。

- 掌握文本和數值處理：熟練操作字符串（文本數據）和數字類型，學會基本的數據處理方法，能夠實現簡單的信息存儲和計算功能。

- 提升代碼可讀性：學會使用註釋，瞭解變量命名規範，編寫清晰易懂的代碼，培養良好的編程習慣，為團隊協作和未來的代碼維護做準備。

- 克服初學恐懼：通過動手實踐簡單的代碼示例，建立“我能做到”的信心，消除對編程的陌生感，激發持續學習的動力。

- 解決常見問題：預先了解初學Python時常見的語法錯誤和邏輯問題，掌握識別與修復這些問題的方法，培養基本的調試能力，避免在學習過程中被小問題“卡住”。

4.2 Python解析器

在開始學習Python的變量和數據類型之前，我們需要先了解Python解析器的基本工作原理。Python解析器是Python編程環境的核心組件，它負責將我們編寫的Python代碼轉換為計算機可以理解 and 執行的指令。

4.2.1 Python解析器的工作原理

Python是一種解釋型語言，這意味著Python代碼在運行時由解析器逐行解釋執行，而不是像C或Java等編譯型語言那樣先編譯成機器碼再執行。這種特性使Python更加靈活和易於調試，但也會導致執行速度相對較慢。

讓我們向DeepSeek詢問Python解析器的工作原理：

用戶提問：

請詳細解釋 Python 解析器的工作原理，以及它与编译型语言的区别。

DeepSeek回答：

一、Python 解析器的工作原理

Python 解析器是将 Python 代码转换为计算机可执行的指令的程序。作为一种解释型语言，Python 的执行过程与编译型语言有显著不同。

二、Python 解析器的基本工作流程

1. 词法分析：解析器首先将源代码分解成标记 (tokens)，如关键字、变量名、运算符等。
2. 语法分析：将标记组织成语法结构，生成抽象语法树 (AST)。
3. 编译为字节：Python 解析器将 AST 编译成 Python 字节码，这是一种中间表示形式，类似于机器码但独立于具体硬件。
4. 执行字节码：Python 虚拟机 (PVM) 执行字节码指令。在 CPython (最常用的 Python 实现) 中，这一步由 C 语言编写的解释器完成。

三、解释型语言与编译型语言的区别

1. 解释型语言 (如 Python)

执行过程: 代码在运行时逐行解释执行。

开发周期: 写代码→直接运行→调试→修改→运行 (快速迭代)。

可移植性: 高 (只要目标系统有相应的解释器)。

执行速度: 通常较慢 (因为需要实时解释)。

内存使用: 通常较高 (需要解释器运行时环境)。

错误检测: 运行时发现 (动态类型检查)。

2. 编译型语言 (如 C/C++)

执行过程: 代码先编译成机器码, 然后执行。

开发周期: 写代码→编译→链接→执行→调试→修改→重新编译 (周期较长)。

可移植性: 较低 (需要针对不同平台重新编译)。

执行速度: 通常较快 (直接执行机器码)。

内存使用: 通常较低 (不需要额外的解释环境)。

错误检测: 编译时发现大部分错误 (静态类型检查)。

四、Python 语言具有的混合特性

实际上, Python 结合了解释型和编译型语言的特点: Python 代码首先被编译成字节码 (.pyc 文件), 然后字节码由 Python 虚拟机解释执行。这种方式结合了两种语言类型的优点: 开发便捷性和一定程度的执行效率。

五、对于不同 Python 语言的实现

CPython: 标准实现, 用 C 语言编写。

PyPy: 使用 JIT (即时编译) 技术, 执行速度更快。

Jython: 在 Java 虚拟机上运行的 Python 实现。

IronPython: 在 .NET 平台上运行的 Python 实现。

每种实现都有自己的解释器, 但都遵循 Python 语言规范。

4.2.2 交互式执行与脚本执行

Python 提供了两种主要的代码执行方式: 交互式执行和脚本执行。

1. 交互式执行

交互式执行 (Read-Eval-Print Loop, REPL) 允许我们输入 Python 代码并立即看到执行结果, 这对于学习和测试代码非常有用。

启动交互式执行的方法如下。

- 1) 打开命令提示符 (Windows) 或终端 (macOS/Linux)。

2) 輸入python或python3命令。

3) 看到>>>提示符後，就可以輸入Python代碼了。

例如：

```
``python
>>> print("Hello, Python!")
Hello, Python!
>>> 2 + 3
5
```

```
>>> name = "小明"
>>> print(f"你好, {name}!")
你好, 小明!
...

```

2.腳本執行

對於較長或需要重複執行的代碼，我們通常將其保存為.py文件（Python腳本），然後執行整個文件。

啟動腳本執行的方法如下。

- 1) 創建一個文本文件，保存為.py文件（如example.py）。
- 2) 在文件中編寫Python代碼。
- 3) 在命令行中使用python example.py命令執行腳本。

例如，創建一個名為calculation.py的文件，內容如下：

```
'''python
# 这是一个简单的计算脚本
a = 10
b = 20
sum_result = a + b
product = a * b
print(f"(a) + (b) = {sum_result}")
print(f"(a) × (b) = {product}")
'''
```

執行這個腳本，之後便可以在終端中看見Python運行後輸出的結果了。

```
'''
$ python calculation.py
10 + 20 = 30
10 × 20 = 200
'''
```

4.2.3 使用DeepSeek輔助理解Python解析器

DeepSeek可以幫助我們更深入地理解Python解析器的工作原理，以及如何有效地使用Python的交互式環境和腳本執行。

用戶提問：

作为初学者，我应该使用 Python 的交互式执行还是脚本执行方式？它们各有什么优缺点？

DeepSeek回答：

Python 初学者：交互式执行 vs 脚本执行

作为 Python 初学者，交互式执行和脚本执行各有其优缺点，最好结合使用两者来加速学习过程。

交互式执行的优点

即时反馈：输入代码后立即看到结果，有助于快速理解概念。

探索性学习：可以轻松尝试不同的命令和语法，观察结果。

调试便捷：可以逐步执行代码，检查变量值和状态。

无须保存文件：适合快速测试小段代码。

内置帮助：可以使用 `help()` 函数获取文档。

交互式执行的缺点

不易保存：代码执行后不会自动保存。

不适合复杂程序：编写和管理大型代码块不方便。

重复执行麻烦：需要重新输入代码或使用上下箭头键查找历史命令。

格式限制：某些依赖缩进的代码结构（如循环、函数）在交互式环境中不太直观。

脚本执行的优点

代码可重用：保存为文件后可以随时执行。

适合复杂程序：可以编写和组织大型程序。

版本控制：可以使用 Git 等工具跟踪代码变化。

模块化：可以将代码分割成多个文件和函数。

IDE 支持：可以利用代码补全、语法高亮等功能。

脚本执行的缺点

反馈周期长：需要保存文件并运行才能看到结果。

调试相对复杂：需要使用 `print` 语句或调试工具。

初始设置：需要设置编辑器或 IDE。

根據DeepSeek的回答，針對初學者，我的建議如下：

1) 用交互式執行環境學習基礎概念：

❑ 嘗試簡單的表達式和語句。

❑ 探索數據類型和內置函數。

❑ 測試小段代碼片段。

2) 轉向腳本執行，編寫實際項目：

❑ 開始編寫簡單腳本（5~20行）。

❑ 逐步增加複雜度。

❑ 學習如何組織和結構化代碼。

3) 結合使用兩種方式：

- ❑ 在交互式執行環境中測試想法。
- ❑ 將有效的代碼保存到腳本中。
- ❑ 使用交互式執行環境調試腳本，解決其中的問題。

4) 使用Jupyter Notebook：

❑ Jupyter結合了兩種方法的優點，既允許交互式執行，同時又保存了代碼和結果。

- ❑ 支持添加文檔和可視化。

記住，這兩種執行方式是互補的，而不是互斥的。你需要根據不同的情況與需求選擇合適的方式。

通過DeepSeek，我們瞭解了Python解析器的基本工作原理，以及如何使用交互式執行和腳本執行這兩種方式來運行Python代碼。這些知識為我們後續學習Python的變量和數據類型奠定了基礎。在下一節中，我們將開始學習Python中的變量，這是存儲和管理數據的基本方式。

4.3 變量

變量是編程中最基本的概念之一，它允許我們存儲數據並在程序中使用這些數據。在Python中，變量的使用非常靈活和直觀，這也是Python成為對初學者友好的語言的原因之一。

4.3.1 變量的命名和使用

在Python中，變量就像是一個標籤或名稱，它指向存儲在計算機內存中的數據。創建變量非常簡單，只需要為它指定一個名稱並賦值即可。

1. 變量的基本使用

讓我們藉助DeepSeek來快速學習如何創建一個Python變量。

用戶提問：

```
如何创建一个 Python 变量?
```

DeepSeek回答：

```
```python
创建一个名为 message 的变量，并赋值为 "Hello Python world!"
message = "Hello Python world!"
print(message)
修改变量的值
message = "Hello Python Crash Course world!"
print(message)
```
```

輸出結果：

```
Hello Python world!
```

```
Hello Python Crash Course world!
```

在這個例子中，我們首先創建了一個名為message的變量，並將字符串“Hello Python world!”賦值給它。然後，我們使用print()函數打印變量的值。接著，我們將變量message的值修改為新的字符串，並再次打印出來。

這個例子展示了Python變量的一個重要特性：變量的值可以隨時更改。

2.變量命名規則

在Python中，變量名必須遵循一定的規則：

1) 變量名只能包含字母、數字和下劃線。變量名可以以字母或下劃線開頭，但不能以數字開頭。

2) 變量名不能包含空格，但可以使用下劃線來分隔單詞。

3) 不要將Python關鍵字和函數名用作變量名，即不要使用Python中有特殊用途的保留字（關鍵字），如print、if等。

4) 變量名應既簡短又具有描述性。

5) 變量名區分大小寫，即message和Message是兩個不同的變量。

讓我們向DeepSeek詢問更多關於Python變量命名的建議。

用戶提問：

```
請提供一些 Python 變量命名的最佳實踐和常見約定。
```

DeepSeek回答：

Python 变量命名的最佳实践和约定

良好的变量命名对于代码的可读性和可维护性至关重要。以下是 Python 变量命名的最佳实践和常见约定。

一、基本命名规则

1) 使用小写字母：变量名通常使用小写字母。

```
```python
推荐
user_name = "John"
不推荐
UserName = "John"
```
```

2) 使用下划线分隔单词（蛇形命名法）。

```
```python
推荐
first_name = "John"
不推荐
firstName = "John" # 驼峰命名法在 Python 中不是常规做法
```
```

3) 避免使用单个字符，除非是在非常简单的上下文中（如循环计数器）。

```
```python
可接受（在简单循环中）
for i in range(5):
 print(i)
推荐（在更复杂的情况下）
for index in range(5):
 print(index)
```
```

4) 不要以下划线开头，除非有特殊目的。

单下划线前缀（`_var`）表示变量是供内部使用的，双下划线前缀（`__var`）用于类的私有属性，双下划线前后缀（`__var__`）用于特殊方法（魔术方法）。

二、命名约定

变量名应该具有描述性

```
```python
推荐
user_age = 25
不推荐
a = 25 # 不清楚这代表什么
```
```

結合DeepSeek的回答，對Python中變量的命名規則再補充如下內容。

1) 常量使用全大寫字母和下劃線。

```
```python
MAX_CONNECTIONS = 100
PI = 3.14159
```
```

2) 布爾變量名通常以is_、has_、can_等開頭。

```
```python
is_valid = True
has_permission = False
can_edit = True
```
```

3) 集合變量名通常使用複數形式。

```
```python
users = ["Alice", "Bob", "Charlie"]
points = [1, 2, 3, 4]
```
```

4) 函數參數名遵循與變量相同的約定。

```
```python
def calculate_total(item_prices, tax_rate):
 # 函數體
```
```

5) 避免使用容易混淆的字母。

例如，小寫字母“l”和大寫字母“O”可能與數字“1”和“0”混淆，導致代碼難以閱讀。為了避免這種情況，最好選擇更清晰、易於區分的字母來命名變量。

3. 實例

下面我們來列舉一些良好的變量命名實例；

用戶提問：

列举一些良好的 Python 变量命名实例。

DeepSeek回答：

良好的变量命名不仅能提高代码的可读性，还能让代码在团队协作中更加清晰和一致。以下是一些遵循 Python 命名规则的实际例子：

```
# 良好的变量命名示例
user_name = "Alice"      # 使用有意义的变量名，避免歧义
user_age = 30            # 变量名简洁且能够准确描述数据含义
is_active = True        # 布尔变量以 'is_' 开头，便于理解其逻辑
items_count = 5         # 名词复数形式表示多个对象
MAX_RETRY_ATTEMPTS = 3  # 常量使用全大写字母，单词间用下划线分隔

# 函数中的良好命名
def calculate_area(width, height):
    """
    计算矩形的面积
    :param width: 矩形的宽度
    :param height: 矩形的高度
    :return: 返回矩形的面积
    """
    return width * height

# 循环中的良好命名
for index, value in enumerate(["a", "b", "c"]):
    print(f"Item {index}: {value}")
```

遵循這些命名規則有助於保持代碼簡潔、易於理解，並符合 Python 社區的最佳實踐。明確的命名使得代碼不僅更具可讀性，還能幫助其他開發者快速理解代碼的功能與結構，提升代碼的可維護性。

4. 變量使用實例

讓我們通過一個更複雜的例子來展示變量的使用方法。

用戶提問：

如何使用 Python 变量来存储和更新不同类型的数据？

DeepSeek回答：

在 Python 中，变量用于存储不同类型的数据，这些数据可以是字符串、整数、浮动值、布尔值等。以下是一个更复杂的例子，展示了如何使用变量来存储和更新一个人的各种属性：

```
```python
创建描述一个人的变量
name = "张明"
age = 25
height = 175.5 # 单位：厘米
is_student = True
使用这些变量
print(f"姓名: {name}")
print(f"年龄: {age} 岁")
print(f"身高: {height} 厘米")
print(f"是否学生: ('是' if is_student else '否')")
修改变量值
age = age + 1 # 增加一岁
height = height + 0.5 # 身高增加 0.5 厘米
打印修改后的信息
print(f"\n一年后:")
print(f"姓名: {name}")
print(f"年龄: {age} 岁")
print(f"身高: {height} 厘米")
```
```

輸出結果：

```
姓名: 张明
年龄: 25 岁
身高: 175.5 厘米
是否学生: 是

一年后:
姓名: 张明
年龄: 26 岁
身高: 176.0 厘米
```

在這個例子中，我們使用了不同的數據類型來描述一個人的不同屬性：

□ name 存儲了一個字符串類型的值（“張明”），表示這個人的姓名是張明。

□ age 存儲了一個整數類型的值（25），表示這個人的年齡是 25 歲。

□ height存儲了一個浮動類型的值（175.5），表示身高是175.5釐米。

□ is_student存儲了一個布爾類型的值（True），表示這個人是學生。

通過這些變量，我們不僅能夠存儲信息，還能夠修改數據值。比如，通過age=age+1來修改年齡，並通過height=height+0.5來修改身高，這展示瞭如何使用Python變量在程序中動態管理和更新數據。

通過適當地使用變量，我們可以輕鬆地操作和更新數據，以便在程序中處理複雜的邏輯。這個簡單的例子也表明了良好的命名和數據管理在編程中的重要性。

4.3.2 使用變量時應避免的命名問題

在使用變量時，有一些常見的命名問題需要避免。

1. 避免命名錯誤

用戶提問：

在使用 Python 變量時，應該避免哪些常見的命名問題？

DeepSeek回答：

```
命名錯誤是初學者常見的問題之一。例如：  
```python  
message = "Hello Python!"
print(message) # 錯誤：變量名拼寫錯誤
```
```

這段代碼會產生NameError: name 'message' is not defined錯誤，因為我們在print語句中錯誤地將message拼寫為mesage。

Python會提示找不到名為message的變量。

根據DeepSeek的回答，我的建議是：在編寫代碼的過程中使用DeepSeek的拼寫檢查功能，該功能可以自動檢測變量名拼寫錯誤並進行修復，避免這類簡單的命名失誤。

2.避免使用Python關鍵字

Python關鍵字是Python語言的保留字，不能用作變量名。例如：

```
```python
class = "Python 编程" # 错误: class 是 Python 关键字
```
```

避免使用Python的保留字作為變量名。你可以讓DeepSeek進行關鍵字檢查，及時發現並修正這類問題。

3.避免變量名衝突

避免使用與Python內置函數同名的變量名，這可能導致意外的行為。

```
```python
不推荐
sum = 10 + 20
print(sum) # 输出: 30

现在 sum 不再是函数，而是变量
numbers = [1, 2, 3]
print(sum(numbers)) # 这会产生错误，因为 sum 现在是整数 30，不是函数
```
```

這裡，sum被用作變量名，覆蓋了Python內置的sum()函數，導致後續調用該函數時發生錯誤。

我們應避免使用Python內置函數名或其他常見庫函數名作為變量名。使用更具描述性的變量名，確保不會與內置函數或類發生衝突。

4.避免使用無意義的變量名

變量名應該清晰地表達其用途，避免使用無意義的名稱：

```
```python
不推荐
a = 10
b = 20
c = a + b

推荐
length = 10
width = 20
area = length * width
```
```

其中，length、width和area這樣的命名能夠清晰地表達變量的含義，而a、b和c則缺乏描述性，很難讓閱讀代碼的人一眼看出它們的用途。

4.3.3 使用DeepSeek解決變量命名問題

當遇到變量命名問題時，DeepSeek可以幫助我們解決這些問題。

用戶提問：

```
我的代码出现了错误: NameError: name 'message' is not defined。这是什么意思?
```

DeepSeek回答：

解决 NameError: name 'message' is not defined

这个错误是 Python 中最常见的错误之一，表示你尝试使用一个未定义的变量。具体来说，Python 找不到名为 'message' 的变量。

错误原因

这种错误通常有以下几种可能的原因：

拼写错误：最常见的原因是变量名拼写错误。例如，你可能定义了 message 但使用时写成了 mesage (少了一个 's')。

变量未定义：你尝试使用的变量之前没有被赋值。

变量作用域问题：变量在当前作用域不可见（比如在函数外部定义但尝试在函数内部使用，而没有正确传递）。

大小写错误：Python 变量名区分大小写，所以 Message 和 message 是两个不同的变量。

根據DeepSeek的回答，下面提供一些修復步驟。

1. 檢查拼寫

仔細檢查你的代碼，查找變量名的拼寫錯誤。特別注意錯誤信息中提到的變量名 mesage，它可能應該是 message。下面是 Python 代碼相關示例：

```
```python
错误代码
message = "Hello, World!"
print(mesage) # 错误: mesage 拼写错误

修复后的代码
message = "Hello, World!"
print(message) # 正确: 拼写一致
```
```

2. 確保變量已定義

在使用變量之前，要確保已經定義了它。

```
```python
错误代码
print(message) # 错误: 变量未定义

修复后的代码
message = "Hello, World!" # 先定义
print(message) # 再使用
```
```

3. 檢查變量作用域

如果在函數中使用變量，確保該變量在函數內部已定義或已作為參數傳入。

```
```python
錯誤代碼
def greet():
 print(message) # 錯誤: message 在函數內部未定義
修復方法 1: 將變量作為參數傳入
def greet(message):
```

```
 print(message)
修復方法 2: 在函數內部定義變量
def greet():
 message = "Hello, World!"
 print(message)
修復方法 3: 使用全局變量 (不推薦, 除非必要)
message = "Hello, World!"
def greet():
 global message
 print(message)
...
```
```

4. 檢查大小寫

確保變量名的大小寫一致。

```
```python
錯誤代碼
Message = "Hello, World!"
print(message) # 錯誤: 大小寫不匹配

修復後的代碼
Message = "Hello, World!"
print(Message) # 正確: 大小寫匹配
...
```
```

5. 使用IDE或編輯器

現代IDE和代碼編輯器（如VS Code、PyCharm）通常會在你輸入代碼時高亮顯示未定義的變量，這可以幫助你在運行代碼前發現

這類錯誤。

6.使用變量命名約定

遵循Python的命名約定可以減少拼寫錯誤：

- 1) 使用有意義的名稱。
- 2) 使用小寫字母和下劃線 (snake_case)。
- 3) 避免使用容易混淆的名稱。

7.預防措施

為了避免變量命名錯誤，可以採取以下預防措施：

- 使用一致的命名風格。
- 在使用變量前確保它已被定義。
- 使用支持語法檢查的編輯器。
- 在複雜程序中，考慮使用類型提示 (Python 3.5+)。

通過DeepSeek，我們瞭解了Python變量的基本概念、命名規則以及如何避免常見的變量命名問題以及如何去修復這些錯誤。變量是Python編程的基礎，掌握它們的使用對於後續學習至關重要。在下一節中，我們將學習Python中的字符串，這是處理文本數據的基本方式。

4.4 字符串

字符串是Python中最常用的數據類型之一，用於表示文本數據。在Python中，字符串是由引號括起來的一系列字符。Python允許使用單引號（'）或雙引號（"）來表示字符串，這提供了很大的靈活性。

4.4.1 修改字符串的大小寫

Python提供了多種方法來修改字符串的大小寫。這些方法不會改變原始字符串，而是返回一個新的修改後的字符串。

1.title()方法

title()方法將字符串中每個單詞的首字母改為大寫，其他字母改為小寫。

```
```python
name = "ada lovelace"
print(name.title()) # 輸出: Ada Lovelace
```
```

2.upper()方法

upper()方法將字符串中的所有字母都改為大寫。

```
```python
name = "Ada Lovelace"
print(name.upper()) # 輸出: ADA LOVELACE
```
```

3.lower()方法

lower()方法將字符串中的所有字母都改為小寫。

```
```python
name = "Ada Lovelace"
print(name.lower()) # 輸出: ada lovelace
```
```

`lower()`方法在存儲數據時特別有用。很多時候，我們不能依賴用戶提供的大小寫是正確的，因此最好將字符串轉換為小寫再存儲。

讓我們向DeepSeek詢問更多關於字符串大小寫的處理方法。

用戶提問：

```
Python 中除了 title()、upper() 和 lower() 外，还有哪些处理字符串大小写的方法？
它们各有什么用途？
```

DeepSeek回答：

Python 字符串大小写处理方法全解析

除了 `title()`、`upper()` 和 `lower()` 外，Python 还提供了多种处理字符串大小写的方法。以下是完整列表及其用途：

一、基本大小写转换方法

1) `capitalize()`：将字符串的第一个字符转换为大写，其余字符转换为小写。

```
```python
text = "hello WORLD"
print(text.capitalize()) # 输出： "Hello world"
```
```

2) `swapcase()`：将字符串中的大写字符转换为小写，小写字符转换为大写。

```
```python
text = "Hello World"
print(text.swapcase()) # 输出： "hELLO WORLD"
```
```

3) `casefold()`：类似于 `lower()`，但更强大，可以处理更多语言的特殊大小写规则。

```
```python
text = "Großes ß"
print(text.lower()) # 输出： "großes ß"
print(text.casefold()) # 输出： "grosses ss"
```
```

二、检查大小写的方法

1) `islower()`：检查字符串中的所有字母是否都是小写。

```
```python
print("hello world".islower()) # 输出： True
print("Hello world".islower()) # 输出： False
```
```

2) `isupper()`：检查字符串中的所有字母是否都是大写。

```
```python
print("HELLO WORLD".isupper()) # 输出： True
print("HELLO world".isupper()) # 输出： False
```
```

3) `istitle()`：检查字符串是否符合标题格式（每个单词的首字母大写）。

```
```python
print("Hello World".istitle()) # 输出： True
print("Hello world".istitle()) # 输出： False
```
```

現在我們結合DeepSeek的回答，進行補充和完善。

4.其他相關檢查方法

1) `isalpha()`：檢查字符串是否只包含字母（不包含數字、空格等）。

```
```python
print("HelloWorld".isalpha()) # 輸出： True
print("Hello World".isalpha()) # 輸出： False (包含空格)
```
```

2) `isalnum()`：檢查字符串是否只包含字母和數字。

```
```python
print("Hello123".isalnum()) # 輸出： True
print("Hello 123".isalnum()) # 輸出： False (包含空格)
```
```

3) `isdigit()`、`isnumeric()`、`isdecimal()`：檢查字符串是否只包含數字。

```
```python
print("12345".isdigit()) # 輸出： True
print("12.34".isdigit()) # 輸出： False (包含小數點)
```
```

5.各方法的實際應用場景

1) `title()`：

- ❑ 格式化人名、地名、書名等專有名詞。
- ❑ 例如，將用戶輸入的姓名格式化為標準格式。

2) `upper()`：

- ❑ 顯示警告或錯誤消息。
- ❑ 處理不區分大小寫的比較（先轉換為大寫再比較）。
- ❑ 處理固定格式的代碼（如HTML標籤）。

3) `lower()`：

❑ 存儲用戶輸入（如電子郵件地址）。

❑ 不區分大小寫的搜索和比較。

4) `capitalize()`：

❑ 格式化句子（使句子的第一個字母大寫）。

❑ 處理用戶輸入的句子。

5) `swapcase()`：

❑ 特殊文本效果。

❑ 反轉大小寫以突出顯示。

6) `casefold()`：

❑ 國際化應用中的不區分大小寫比較。

❑ 處理多語言文本。

7) 檢查方法（`islower()`、`isupper()`等）：

❑ 驗證用戶輸入。

❑ 確保密碼複雜性（例如，要求包含大小寫字母）。

❑ 條件格式化。

6.使用示例

下面是使用的綜合應用示例和代碼測試。

```

```python
def validate_username(username):
 """ 验证用户名是否有效 """
 if not username.isalnum():
 return False, "用户名只能包含字母和数字"

 return True, "用户名有效"

def format_name(first_name, last_name):
 """ 格式化姓名 """
 return f"{first_name.title()} {last_name.title()}"

def is_password_strong(password):
 """ 检查密码强度 """
 has_lower = any(char.islower() for char in password)
 has_upper = any(char.isupper() for char in password)
 has_digit = any(char.isdigit() for char in password)

 return has_lower and has_upper and has_digit

测试
print(validate_username("john_doe")) # (False, '用户名只能包含字母和数字')
print(format_name("john", "doe")) # 'John Doe'
print(is_password_strong("Password123")) # True
...

这些字符串方法使 Python 成为处理文本数据的强大工具，无论是简单的格式化还是复杂的文本处理。
...

```

## 4.4.2 合并（拼接）字符串

在 Python 中，可以使用加号（+）来合并（拼接）字符串。

```

```python
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name
print(full_name)      # 输出: ada lovelace
...

```

我們還可以使用這種合併方式來創建更複雜的消息。

```
```python
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name
message = "Hello, " + full_name.title() + "!"
print(message) # 輸出: Hello, Ada Lovelace!
```
```

另外Python 3.6及更高版本引入了f字符串（格式化字符串），這是一種更簡潔的字符串格式化方式。f字符串以字母f開頭，花括號{}中可以放置變量名或表達式。

```
```python
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
print(full_name) # 輸出: ada lovelace

message = f"Hello, {full_name.title()}!"
print(message) # 輸出: Hello, Ada Lovelace!
```
```

f字符串使得字符串格式化更加簡潔和直觀，是Python中推薦的字符串格式化方式。

4.4.3 使用製表符或換行符來添加空白

在字符串中，空白是指空格、製表符和換行符等不可見的字。Python提供了特殊的字符序列來表示這些空白。

1. 製表符

製表符用\t表示，可以在文本中添加水平空白。

```
```python
print("Python")
print("\tPython")
```
```

輸出結果：

```
...  
Python  
    Python  
...
```

2.換行符

換行符用\n表示，可以在字符串中添加換行。

```
```python  
print ("Languages:\nPython\nC\nJavaScript")
```
```

輸出結果：

```
...  
Languages:  
Python  
C  
JavaScript  
...
```

3.組合使用

製表符和換行符可以組合使用。

```
```python  
print ("Languages:\n\tPython\n\tC\n\tJavaScript")
```
```

輸出結果：

```
...  
Languages:  
    Python  
    C  
    JavaScript  
...
```

4.4.4 刪除空白

在處理用戶輸入或從文件讀取數據時，字符串可能包含不需要的空白。Python提供了幾種方法來刪除字符串中的空白。

1.rstrip()方法

rstrip()方法刪除字符串末尾的空白。

```
```python
favorite_language = "python "
print(favorite_language.rstrip()) # 輸出: 'python'
```
```

2.lstrip()方法

lstrip()方法刪除字符串開頭的空白。

```
```python
favorite_language = " python"
print(favorite_language.lstrip()) # 輸出: 'python'
```
```

3.strip()方法

strip()方法同時刪除字符串兩端的空白。

```
```python
favorite_language = " python "
print(favorite_language.strip()) # 輸出: 'python'
```
```

這些方法在處理用戶輸入時特別有用，因為用戶可能會不小心輸入額外的空格。

4.4.5 字符串的高級用法

在操作字符串時，有一些高級用法可以大幅提升效率。下面，我們將介紹Python字符串的一些進階技巧。

1.轉義字符

如果字符串中包含引號，可以使用反斜槓（\）來轉義它們。

```
```python
message = "The language \"Python\" is named after Monty Python."
print(message) # 輸出: The language "Python" is named after Monty Python.
```
```

常見的轉義字符包括：

- \ '：單引號
- \"：雙引號
- \\：反斜槓
- \n：換行符
- \t：製表符

讓我們向DeepSeek詢問更多關於Python字符串的高級用法。

用戶提問：

請介紹 Python 字符串的一些高級用法和技巧，特別是在處理複雜文本時有用的方法。

DeepSeek回答：

Python 字符串高级用法与技巧

Python 提供了丰富的字符串处理功能，以下是一些高级用法和技巧，特别适用于处理复杂文本。
F- 字符串的高级用法 (Python 3.6+)

```
```python
在 f- 字符串中使用表达式
x = 10
y = 5
print(f"{x} + {y} = {x + y}") # 输出：10 + 5 = 15
格式化数字
price = 1234.5678
print(f"价格：${price:.2f}") # 输出：价格：$1234.57
对齐文本
name = "张三"
score = 95
print(f"{name:10} - {score:<5}") # 输出："张三 - 95 "
print(f"{name:>10} - {score:>5}") # 输出：" 张三 - 95"
print(f"{name:^10} - {score:^5}") # 输出：" 张三 - 95 "

使用千位分隔符
large_num = 1234567890
print(f"{large_num:,}") # 输出：1,234,567,890
二进制、八进制、十六进制表示
num = 42
print(f"{num:b}") # 二进制：101010
print(f"{num:o}") # 八进制：52
print(f"{num:x}") # 十六进制：2a
print(f"{num:X}") # 大写十六进制：2A
日期格式化
from datetime import datetime
now = datetime.now()
print(f"{now:%Y-%m-%d %H:%M:%S}") # 输出：2023-05-15 14:30:45
```
```

格式化字符串的旧方法

```
```python
str.format() 方法
print("{} + {} = {}".format(x, y, x + y))
print("{} 得了 {} 分".format(name="李四", score=98))
% 操作符 (类似 C 语言的 printf)
print("%s 得了 %d 分" % ("王五", 92))
```
```

2. 字符串的分割與合併

字符串的分割和合並是處理文本數據最常見的操作之一。Python 提供了幾種強大的方法來實現這些功能，讓我們能夠輕鬆地

將文本拆分為多個部分或將多個字符串組合成一個整體。

```
```python
分割字符串
text = "Python,Java,C++,JavaScript"
languages = text.split(",") # ['Python', 'Java', 'C++', 'JavaScript']

限制分割次數
text = "a-b-c-d-e"
parts = text.split("-", 2) # ['a', 'b', 'c-d-e']
按多个分隔符分割 (使用正则表达式)
import re
text = "Python;Java,C++;JavaScript"
languages = re.split("[:;.]", text) # ['Python', 'Java', 'C++',
 'JavaScript']

合并字符串
languages = ["Python", "Java", "C++", "JavaScript"]
text = ", ".join(languages) # "Python, Java, C++, JavaScript"
合并路径 (跨平台)
import os
path = os.path.join("usr", "local", "bin") # "usr/local/bin" 或
 "usr\\local\\bin"
...
```
```

3.字符串的查找與替換

以下代碼展示了在Python中進行字符串查找和替換的多種方法，這些操作在文本處理、數據分析和內容轉換中非常常見。

```
```python
查找子字符串
text = "Python is amazing and Python is easy to learn"
position = text.find("Python") # 0 (第一次出現的位置)
```
```

```

position = text.find("Python", 10) # 21 (从索引 10 开始查找)
position = text.find("Java")      # -1 (未找到)
# 计数
count = text.count("Python")     # 2
# 检查开头和结尾
filename = "document.txt"
print(filename.startswith("doc")) # True
print(filename.endswith(".txt"))  # Tru
# 替换
new_text = text.replace("Python", "JavaScript")
# "JavaScript is amazing and JavaScript is easy to learn"
# 替换指定次数
new_text = text.replace("Python", "JavaScript", 1)
# "JavaScript is amazing and Python is easy to learn"
# 使用正则表达式进行复杂替换
import re
text = "My phone numbers are 123-456-7890 and 098-765-4321"
pattern = r"\d{3}-\d{3}-\d{4}"
new_text = re.sub(pattern, "XXX-XXX-XXXX", text)
# "My phone numbers are XXX-XXX-XXXX and XXX-XXX-XXXX"
'''

```

4. 多行字符串的处理

下面的代码展示了Python中多行字符串处理的方法。这些方法允许我们在字符串中查找特定内容、统计出现次数、检查字符串的开头和结尾，以及替换字符串的部分内容。

```

'''python
# 三引号字符串
multiline = """这是第一行
这是第二行
这是第三行"""
# 去除每行前后的空白
lines = [line.strip() for line in multiline.splitlines()]
# 文本对齐
import textwrap
wrapped = textwrap.fill(long_text, width=40) # 将文本按 40 字符宽度换行
# 去除共同的前导空白 (对于代码示例很有用)
code = textwrap.dedent("""
    def hello():
        print("Hello, World!")
        return True
""")
'''
'''

```

5. 字符串的轉換與編碼

以下代碼演示了Python中字符串與其他數據類型之間的轉換，以及字符串編碼與解碼的基本操作。

```
```python
字符串与其他数据类型的转换
num_str = str(42) # "42"
num = int("42") # 42
float_num = float("3.14") # 3.14
字符与 ASCII 码转换
char_code = ord('A') # 65
char = chr(65) # 'A'
编码与解码
text = "你好，世界"
encoded = text.encode("utf-8") # b'\xe4\xbd\xa0\xe5\xa5\xbd\xef\xbc\x8c\xe4\xb8\x96\xe7\x95\x8c'
decoded = encoded.decode("utf-8") # "你好，世界"
```
```

6. 字符串的驗證與過濾

下面的代碼演示了在Python中如何驗證字符串內容和過濾字符串。這些技術常用於表單驗證、數據清洗和文本處理等場景。

```
```python
验证字符串内容
password = "Passw0rd!"
has_upper = any(c.isupper() for c in password)
has_digit = any(c.isdigit() for c in password)
has_special = any(not c.isalnum() for c in password)
过滤字符串
digits_only = ''.join(c for c in "abc123def456" if c.isdigit())
"123456"
letters_only = ''.join(c for c in "abc123def456" if c.isalpha())
"abcdef"
转换表 (用于批量字符替换)
from string import maketrans
trans_table = str.maketrans("aeiou", "12345")
new_text = "hello world".translate(trans_table) # "h2ll4 w4rld"
```
```

7. 字符串的插值與模板

以下代碼展示了Python中使用模板進行字符串插值的方法。這種技術在需要動態生成文本時非常有用。

```
```python
字符串模板
from string import Template
t = Template("$name 今年 $age 岁 ")
s = t.substitute(name="张三", age=25) # "张三 今年 25 岁 "
安全的模板替换 (缺少键时不会报错)
s = t.safe_substitute(name="张三") # "张三 今年 $age 岁 "
```
```

8. 實際應用示例

(1) 解析CSV數據

```
```python
csv_data = "name,age,city\nAlice,25,Beijing\nBob,30,Shanghai"
lines = csv_data.strip().split('\n')
headers = lines[0].split(',')
result = []

for line in lines[1:]:
 values = line.split(',')
 person = dict(zip(headers, values))
 result.append(person)

print(result) # [{'name': 'Alice', 'age': '25', 'city': 'Beijing'}, ...]
```

### 提取文本中的所有 URL
```python
import re
text = "Visit our website at https://example.com or http://test.org
for more info."
urls = re.findall(r'https?://[^\s]+', text)
print(urls) # ['https://example.com', 'http://test.org']
```
```

(2) 格式化表格數據

```
```python
data = [
 ["Name", "Age", "City"],
 ["Alice", "25", "Beijing"],
 ["Bob", "30", "Shanghai"],
 ["Charlie", "35", "Guangzhou"]
]
for row in data:
 print(f"{row[0]:<10} {row[1]:^5} {row[2]:>10}")
...
```
```

這些高級字符串技巧可以幫助你更有效地處理複雜的文本數據，無論是數據清洗、文本分析還是生成格式化報告。

通過DeepSeek，我們瞭解了Python字符串的基本概念和操作，包括修改大小寫、合併字符串、添加和刪除空白，以及常見的語法錯誤。字符串是Python編程中最常用的數據類型之一，掌握它對於後續學習至關重要。在下一節中，我們將學習Python中的數字類型，包括整數和浮點數。

4.5 數字

Python中的數字主要有兩種類型：整數和浮點數。Python支持各種數學運算，使得處理數值數據變得簡單直觀。

4.5.1 整數

整數是沒有小數部分的數字，可以是正數、負數或零。在Python中，整數的長度沒有限制，可以根據需要增長到任意大小。

```
```python
整數示例
age = 23
population = 7_800_000_000 # 使用下划线使大數字更易讀
negative_number = -42
zero = 0
```
```

Python支持所有基本的數學運算。

```
```python
基本數學運算
print(2 + 3) # 加法: 5
print(3 - 2) # 減法: 1
print(2 * 3) # 乘法: 6
print(3 / 2) # 除法: 1.5 (注意結果是浮點數)
print(3 // 2) # 整數除法: 1 (結果是整數, 舍去小數部分)
print(3 % 2) # 取模 (余數): 1
print(3 ** 2) # 乘方: 9 (3 的 2 次方)
```
```

Python還支持運算順序遵循數學中的標準規則（括號、乘方、乘除、加減）。

```
```python
```

```
print(2 + 3 * 4) # 14 (先乘法后加法)
print((2 + 3) * 4) # 20 (括号内先计算)
'''
```

讓我們向DeepSeek詢問更多關於Python整數的信息。

用戶提問：

```
Python 中整数有哪些特殊的用法和限制？如何处理非常大的整数？
```

DeepSeek回答：

## Python 整数的特殊用法与处理大整数

Python 的整数处理能力非常强大，与许多其他编程语言相比有一些独特的特性。

### 一、Python 整数的特点

1) 无限精度: Python 整数可以无限增长，只受限于可用内存。

```
```python
# 计算一个非常大的数
result = 2 ** 1000
print(result) # 10715086071862673209484250490600018105614048117
              055336...
print(len(str(result))) # 302 位数字
```
```

2) 不同进制表示。

```
```python
# 二进制 (以 0b 开头) # 等于十进制的 10
binary = 0b1010
# 八进制 (以 0o 开头) # 等于十进制的 10
octal = 0o12
# 十六进制 (以 0x 开头) # 等于十进制的 10
hexadecimal = 0xA

print(binary, octal, hexadecimal) # 10 10 10
```
```

3) 数字分隔符 (Python 3.6+)。

```
```python
# 使用下划线使大数字更易读
billion = 1_000_000_000
binary_million = 0b_1111_0100_0010_0100_0000
```
```

### 二、处理大整数

Python 自动处理大整数，不需要特殊的库或数据类型。

```
```python
# 计算大数的阶乘
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n-1)
```
```

```

计算 100 的阶乘
fact_100 = factorial(100)
print(f'100 的阶乘有 {len(str(fact_100))} 位数字') # 158 位数字
...

三、整数相关的内置函数
1) abs(): 返回绝对值。
```python
print(abs(-42)) # 42
```

2) pow(): 计算幂。
```python
print(pow(2, 10)) # 1024 (2 的 10 次方)
print(pow(2, 10, 17)) # 4 (2 的 10 次方对 17 取模)
```

3) round(): 四舍五入 (虽然主要用于浮点数, 但也可用于整数)。
```python
print(round(42, -1)) # 40 (四舍五入到最接近的 10 的倍数)
```

4) divmod(): 同时返回商和余数。
```python
quotient, remainder = divmod(17, 5)
print(quotient, remainder) # 3 2
```

```

現在我們結合DeepSeek的回答進行補充。

## 1. 位運算

Python支持整數的位運算。

```

```python
a = 60 # 二进制: 0011 1100
b = 13 # 二进制: 0000 1101
# 按位与
print(a & b) # 12 (0000 1100)
# 按位或
print(a | b) # 61 (0011 1101)
# 按位异或
print(a ^ b) # 49 (0011 0001)
# 按位取反
print(~a) # -61 (1100 0011)
# 左移
print(a << 2) # 240 (1111 0000)
# 右移
print(a >> 2) # 15 (0000 1111)
```

```

## 2. 整數的限制

雖然Python整數理論上沒有大小限制，但實際使用中仍有一些注意事項。

- 內存限制：非常大的整數會佔用大量內存。
- 性能考慮：對非常大的整數進行運算可能會很慢。

□ 整數除法：在Python 3中，整數相除 (/) 會返回浮點數，如果需要整數結果，應使用整數除法 (//)。

## 3. 處理大整數的實用技巧

1) 使用math模塊的函數。

```
```python
import math
# 计算大数的对数
big_num = 10**100          # 谷歌数 (10 的 100 次方)
print(math.log10(big_num)) # 100.0
# 检查一个数是否是 2 的幂
def is_power_of_two(n):
    return n > 0 and (n & (n - 1)) == 0
print(is_power_of_two(1024)) # True
print(is_power_of_two(1000)) # False
```
```

2) 使用decimal模塊處理精確計算。

```
```python
from decimal import Decimal, getcontext
# 设置精度
getcontext().prec = 100
# 计算大数的精确除法
result = Decimal(1) / Decimal(7)
print(result)          # 0.142857142857...
```
```

Python的整數處理能力使其成為處理數學計算、密碼學、大數據分析等領域的理想選擇。

## 4.5.2 浮點數

浮點數是帶有小數部分的數字。在Python中，浮點數使用標準的十進制表示法或科學計數法表示。下面是一些在Python中使用浮點數的代碼示例。

```
```python
# 浮点数示例
pi = 3.14159
height = 175.5
avogadro = 6.022e23      # 科学计数法: 6.022 × 1023

tiny = 1e-10             # 科学计数法: 1 × 10-10
```
```

### 1. 浮點數與整數相同的數學運算

以下代碼展示了Python中浮點數的基本數學運算。浮點數是計算機中表示帶小數點數值的方式，但由於二進制表示的限制，有時會產生精度問題。

```
```python
# 浮点数运算
print(0.1 + 0.2)          # 0.30000000000000004
print(3.0 - 2.0)          # 1.0
print(2.0 * 3.0)          # 6.0
print(3.0 / 2.0)          # 1.5
print(3.0 // 2.0)         # 1.0 (整数除法, 但结果是浮点数)
print(3.0 % 2.0)          # 1.0 (取模, 结果是浮点数)
print(3.0 ** 2.0)         # 9.0 (乘方)
```
```

### 2. 浮點數精度問題

在處理浮點數時，需要注意精度問題。由於計算機以二進制存儲數字，某些十進制小數無法精確表示為二進制小數，這會導致一

些看似奇怪的結果。

```
```python
print(0.1 + 0.2)          # 0.30000000000000004, 而不是 0.3
print(0.1 + 0.2 == 0.3) # False
```
```

這不是Python的漏洞，而是計算機表示浮點數的固有限制。對於需要精確小數計算的場景（如金融計算），可以使用Python的decimal模塊：

```
```python
from decimal import Decimal
print(Decimal('0.1') + Decimal('0.2'))          # 0.3
print(Decimal('0.1') + Decimal('0.2') == Decimal('0.3')) # True
```
```

讓我們向DeepSeek詢問更多關於處理浮點數精度問題的方法。

用戶提問：

在 Python 中，如何处理浮点数计算中的精度问题？有哪些最佳实践？

DeepSeek回答：

Python 浮点数精度问题及解决方案

浮点数精度问题是计算机科学中的一个基本挑战，不仅限于 Python。以下是理解和处理这些问题的全面指南。

为什么会有精度问题?

浮点数精度问题源于二进制表示法的局限性。在十进制中简单的小数(如 0.1)在二进制中可能是无限循环小数, 必须被截断, 从而导致微小的舍入误差。

```
```python
print(0.1 + 0.2)           # 输出: 0.30000000000000004
print(0.1 + 0.2 == 0.3)   # 输出: False
```
```

解决方案与最佳实践

1. 使用 round() 函数

对于简单计算, 可以使用 round() 函数将结果四舍五入到所需的精度。

```
```python
result = 0.1 + 0.2
print(round(result, 1))    # 输出: 0.3
```
```

2. 使用 decimal 模块

对于需要精确小数计算的场景(如金融计算), decimal 模块提供了精确的十进制算术。

```
```python
from decimal import Decimal, getcontext
# 设置精度(可选)
getcontext().prec = 28
# 使用字符串初始化 Decimal 对象(推荐)
a = Decimal('0.1')
b = Decimal('0.2')
c = a + b
print(c)                   # 输出: 0.3
print(c == Decimal('0.3')) # 输出: True
```
```

注意

不要直接使用浮点数初始化。

输出结果:

```
d = Decimal(0.1) # 已经包含了浮点误差
print(d) # 输出类似: 0.10000000000000000551115123125782702
 1181583404541015625
```

### 3.使用分数

對於需要精確有理數計算的場景，fractions模塊提供了分數算術。

```

'''python
from fractions import Fraction
a = Fraction(1, 10) # 1/10
b = Fraction(2, 10) # 2/10

c = a + b
print(c) # 输出: 3/10
print(float(c)) # 转换回浮点数: 0.3
'''

```

## 4.使用數學技巧

對於特定問題，可以通過數學技巧避免精度問題。

```

'''python
例如, 处理货币计算时, 使用整数分 (cents) 而非浮点数元 (dollars)
dollars = 19.99
cents = int(dollars * 100) # 1999 分
进行计算...
result_dollars = cents / 100 # 转回美元
'''

```

## 5.近似相等比較

當需要比較浮點數是否相等時，應檢查它們是否“足夠接近”，而不是完全相等。

```

'''python
def is_close(a, b, rel_tol=1e-9, abs_tol=0.0):
 return abs(a - b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol)

print(is_close(0.1 + 0.2, 0.3)) # 输出: True
Python 3.5+ 可以使用标准库函数
import math
print(math.isclose(0.1 + 0.2, 0.3)) # 输出: True
'''

```

## 6.格式化顯示

在顯示浮點數時，可以使用格式化來控制小數位數。

```
```python
result = 0.1 + 0.2
print(f"{result:.1f}")           # 輸出：0.3
```
```

## 7. 特定場景的最佳實踐

### (1) 金融計算中的精確數值處理

以下代碼展示瞭如何使用Python的decimal模塊進行高精度金融計算，特別是複利計算。在金融應用中，浮點數的精度問題可能導致嚴重的計算誤差，因此需要使用專門的數值表示方法來確保準確性。

```
```python
from decimal import Decimal, getcontext
getcontext().prec = 28
getcontext().rounding = 'ROUND_HALF_UP'      # 銀行家舍入規則
def calculate_interest(principal, rate, years):
    p = Decimal(str(principal))
    r = Decimal(str(rate))
    t = Decimal(str(years))
    # 復利計算
    amount = p * (1 + r) ** t
    return amount
print(calculate_interest(1000, 0.05, 5))     # 1276.2815625...
```
```

### (2) 科學計算

對於科學計算，通常可以接受小誤差，使用NumPy可以提高性能。

```
```python
import numpy as np
# NumPy 有專門的函數處理浮點問題
print(np.isclose(0.1 + 0.2, 0.3))           # 輸出：True
# 對於大量數據的計算，NumPy 更高效
data = np.array([0.1, 0.2, 0.3])
result = np.sum(data)                       # 比循環累加更精確且更快
```
```

### (3) 幾何計算

以下代碼展示瞭如何在Python中使用數學庫進行簡單的幾何計算，特別是計算圓的面積。這種計算在科學計算、工程設計、圖形應用和教育領域經常使用。

```
```python
import math
# 計算圓的面積
radius = 2.5
area = math.pi * radius ** 2
print(f"圓的面積: {area:.6f}") # 控制顯示精度
```
```

### 8. 避免的做法

- ❑ 避免連續累加浮點數，這會累積誤差。
- ❑ 避免直接比較浮點數是否相等。
- ❑ 避免使用浮點數作為字典鍵或其他需要精確相等的場景。
- ❑ 避免將浮點數直接傳遞給Decimal構造函數，應使用字符串。

### 9. 總結

處理Python浮點數精度問題的最佳實踐：

- ❑ 瞭解限制：認識到浮點精度問題是計算機科學的基本限制。
- ❑ 選擇合適的工具：根據需求選擇round()、decimal或fractions。
- ❑ 適當比較：使用math.isclose()或自定義函數進行浮點數比較。

□ 格式化輸出：控制顯示的小數位數。

□ 考慮替代方案：某些情況下，使用整數計算可避免精度問題。

最後，記住，選擇哪種方法取決於你的具體需求：精度要求、性能考慮和代碼可讀性。

## 4.6 註釋

註釋是程序中的非執行部分，用於解釋代碼的工作原理或提供其他信息。良好的註釋可以使代碼更易於理解和維護，無論是對他人還是對未來的自己。Python中的註釋以井號（#）開頭，解釋器會忽略註釋行。

### 4.6.1 如何編寫註釋

在Python中，註釋有以下幾種常見形式。

#### 1. 單行註釋

單行註釋以井號（#）開頭，可以單獨佔一行，也可以放在代碼行的末尾。

```
```python
# 这是一个单独的注释行
name = "Alice" # 这是行末注释
```
```

#### 2. 多行註釋

Python沒有專門的多行註釋語法，但可以使用多個單行註釋或三引號字符串（通常用於文檔字符串）。

```
```python
# 这是第一行注释

# 这是第二行注释
# 这是第三行注释
```
```

這也是一種多行註釋的方式，雖然技術上這是一個多行字符串，但如果不賦值給變量，它不會影響程序執行。

### 3.文檔字符串

文檔字符串是一種特殊的註釋，用於為模塊、類、方法或函數提供文檔。它們使用三引號（單引號或雙引號）定義，並且通常放在定義的開頭。

```
```python
def calculate_area(radius):
    """
    計算圓的面积。
    参数：
        radius (float): 圓的半径
    返回：
        float: 圓的面积
    """
    return 3.14159 * radius ** 2
...`
```

文檔字符串可以通過help()函數或__doc__屬性訪問。

```
```python
help(calculate_area) # 显示函数的文档字符串
print(calculate_area.__doc__) # 直接打印文档字符串
...`
```

### 4.註釋的縮進

註釋應該與它們所註釋的代碼保持相同的縮進級別。

```
```python
def greet(name):
    # 检查名字是否为空
    if not name:
        name = "World"
    # 返回问候语
    return f"Hello, {name}!"
...`
```

讓我們向DeepSeek詢問更多關於Python註釋的最佳實踐。

用戶提問：

在 Python 中，有哪些注释的最佳实践？如何写出有效且有用的注释？

DeepSeek回答：

Python 注释的最佳实践

良好的注释可以显著提高代码的可读性和可维护性。以下是 Python 注释的最佳实践和指南。
注释的基本原则

注释解释“为什么”，而不仅仅是“是什么”。代码本身应该清晰地表达它“做什么”，注释应该解释背后的原因、决策和复杂逻辑。

```
```python
不好的注释（仅重复代码的明显功能）
x = x + 1 # 将 x 增加 1
好的注释（解释原因）
x = x + 1 # 补偿浮点数舍入误差
```
```

要寫出有效且有用的註釋，可以參考以下編寫原則：

- 保持註釋的更新。
- 過時的註釋比沒有註釋更有害。
- 修改代碼時同時更新相關註釋。
- 使用清晰、簡潔的語言。
- 避免冗長、模糊或技術性過強的解釋。
- 使用簡單直接的語言。

5.不同類型的註釋及其用途

(1) 文檔字符串

文檔字符串是Python的特色，用於為模塊、類、方法和函數提供文檔。

```

'''python
def calculate_discount(price, rate=0.1):
    """
    根据给定的折扣率计算折扣价格。
    Args:
        price (float): 原始价格
        rate (float, optional): 折扣率, 默认为 0.1 (10%)
    Returns:
        float: 折扣后的价格
    Raises:
        ValueError: 如果价格为负或折扣率不在 0 ~ 1 之间
    Examples:
        >>> calculate_discount(100, 0.2)
        80.0
    """
    if price < 0 or not 0 <= rate <= 1:
        raise ValueError(" 价格必须为正数, 折扣率必须在 0 到 1 之间 ")

    return price * (1 - rate)
'''

```

(2) 文檔字符串的最佳實踐

- 使用三重雙引號 ("""")。
- 第一行應是簡短的摘要。
- 詳細描述與第一行之間空一行。
- 遵循一致的格式（如 Google 風格、NumPy 風格或 reStructuredText 風格）。
- 包含參數、返回值、異常和示例。

(3) 模塊級註釋

在模塊開頭的註釋應該描述模塊的目的和內容。

(6) TODO 註釋

使用TODO註釋標記需要將來完成的工作。

```
```python
TODO: 实现缓存机制以提高性能
TODO(username): 在下一版本中添加批处理支持
FIXME: 当输入包含特殊字符时会崩溃
```
```

(7) 何時使用註釋

- ❑ 複雜算法：解釋算法的工作原理、選擇原因或參考資料。
- ❑ 非直觀的解決方案：解釋為什麼選擇特定的實現方式。
- ❑ 臨時解決方案或變通方法：標記並解釋臨時代碼。
- ❑ 性能優化：解釋性能敏感代碼的優化策略。
- ❑ 警告和限制：提醒代碼的限制或潛在問題。

(8) 何時避免註釋

- ❑ 不要用註釋來解釋糟糕的代碼：應該重寫代碼使其清晰。
- ❑ 避免註釋掉的代碼塊：使用版本控制而不是註釋來保存舊代碼。
- ❑ 避免冗餘註釋：不要重複代碼已經清晰表達的內容。

6. 實際示例

(1) 不好的註釋示例

```

```python
计算年龄
def calc_age(dob):
 # 获取今天的日期
 today = datetime.today()
 # 计算年龄
 age = today.year - dob.year

 # 检查生日是否已过
 if (today.month, today.day) < (dob.month, dob.day):
 age -= 1
 # 返回年龄
 return age
...

```

## (2) 好的註釋示例

```

```python
def get_age_category(age):
    """
    根据年龄确定人口统计类别。
    分类标准基于 2020 年人口普查指南。
    """
    # 处理边界情况
    if age < 0:
        raise ValueError(" 年龄不能为负 ")
    # 我们使用闭区间 [a, b] 而不是半开区间 [a, b)
    # 这样更符合人口统计学的惯例
    if 0 <= age <= 14:
        return " 儿童 "
    elif 15 <= age <= 24:
        return " 青少年 "
    elif 25 <= age <= 64:
        return " 成年人 "
    else: # age >= 65
        return " 老年人 "
...

```

(3) 工具和自動化

利用工具來維護註釋質量。

□ Pylint/Flake8：檢查文檔字符串和註釋風格。

□ Sphinx：從文檔字符串生成文檔。

- Black：代碼格式化工具，幫助保持註釋對齊。

優秀的Python註釋應該具有如下特點。

- 解釋代碼的“為什麼”，而不僅是“是什麼”。

- 保持更新和準確。

- 遵循一致的格式和風格。

- 提供真正的價值，而不是重複明顯的內容。

- 幫助其他開發者（包括未來的你）理解代碼。

最後，最好的代碼是自文檔化的，註釋是補充而非替代清晰的代碼。

4.6.2 編寫有效的註釋

編寫有效的註釋是一項重要的技能。以下是一些關於什麼時候應該添加註釋以及應該包含什麼內容的指導原則。

1. 什麼時候添加註釋

- 複雜的算法或邏輯：解釋複雜算法的工作原理或非直觀的邏輯。

- 重要的決策：解釋為什麼選擇特定的實現方式。

- 潛在的問題或限制：標記代碼的限制或可能的問題。

- 不明顯的依賴關係：解釋代碼與其他部分的依賴關係。

□ 臨時解決方案：標記臨時或變通的解決方案，以便將來改進。

2. 什麼時候不需要註釋

□ 明顯的操作：不要為明顯的操作添加註釋，如簡單的賦值或基本計算。

□ 自解釋的代碼：如果代碼本身已經足夠清晰，不需要額外的解釋。

□ 重複代碼的內容：避免僅僅重複代碼已經表達的內容。

3. 註釋的內容

好的註釋應該包含以下內容：

□ 為什麼：解釋代碼背後的原因和決策，而不僅僅是它做了什麼。

□ 如何：對於複雜的算法，解釋它是如何工作的。

□ 限制和假設：說明代碼的限制和假設條件。

□ 參考資料：提供算法或方法的來源或參考資料。

4. 註釋示例

以下是一些不好的和好的註釋示例。

不好的註釋示例：

```
```python
计算总和
def calculate_sum(numbers):
 total = 0 # 初始化总和为 0
 for num in numbers:
 total += num # 将数字加到总和中
 return total # 返回总和
...
```
```

這個例子中的註釋只是重複了代碼已經清楚表達的內容，沒有提供額外的價值。

好的註釋示例：

```
```python
使用二分查找而不是线性搜索，因为数据集已排序且很大
def find_user(user_id, users):
 left, right = 0, len(users) - 1
 while left <= right:
 mid = (left + right) // 2
 if users[mid].id == user_id:
 return users[mid]
 elif users[mid].id < user_id:
 left = mid + 1
 else:
 right = mid - 1
 return None
...
```
```

4.6.3 使用DeepSeek輔助編寫註釋

DeepSeek可以幫助我們為代碼生成有意義的註釋。例如，我們可以向DeepSeek提供一段代碼，並請求它為我們添加適當的註釋：

用戶提問：

请为以下代码添加有意义的注释：

```
def process_data(data, threshold=0.5):
    result = []
    for item in data:
        if item['confidence'] < threshold:
            continue
        value = item['value']
        if isinstance(value, str):
            try:
                value = float(value)
            except ValueError:
                value = 0.0
        result.append({
            'id': item['id'],
            'processed_value': value * 1.5,
            'status': 'processed'
        })
    return result
```

DeepSeek回答：

```
```python
def process_data(data, threshold=0.5):
```

```

"""
处理数据项列表，过滤低置信度项并转换值。
Args:
 data: 包含数据项的列表，每项应该是一个字典，
 至少包含 'id' 'value' 和 'confidence' 键
 threshold: 置信度阈值，低于此值的项将被过滤掉，默认为 0.5
Returns:
 处理后的数据项列表，每项包含 'id' 'processed_value' 和 'status' 键
"""
result = []
遍历所有数据项
for item in data:
 # 过滤掉置信度低于阈值的项
 if item['confidence'] < threshold:
 continue
 # 提取值并确保它是浮点数
 value = item['value']
 if isinstance(value, str):
 try:
 # 尝试将字符串转换为浮点数
 value = float(value)
 except ValueError:
 # 如果转换失败，设置默认值为 0
 value = 0.0
 # 创建处理后的数据项并添加到结果列表
 result.append({
 'id': item['id'],
 'processed_value': value * 1.5, # 将值放大 1.5 倍
 'status': 'processed'
 })
return result
...

```

通過DeepSeek，我們瞭解瞭如何在Python代碼中添加有效的註釋，以及什麼樣的註釋是有用的。良好的註釋可以使代碼更易於理解和維護，無論是對他人還是對未來的自己。

## 4.7 結語

本章我們深入探討了Python中的變量和簡單數據類型，這些是構建Python程序的基礎構件。通過本章的學習，我們掌握了以下幾個關鍵概念：

□ Python解析器的工作原理：我們瞭解了Python解析器如何執行代碼，它將Python代碼轉換為計算機可以理解的指令，使我們能夠專注於解決問題而不是底層細節。Python的交互式環境為學習和測試代碼提供了便捷的方式。

□ 變量的使用與命名規則：變量是編程中存儲和管理數據的基本方式。我們學習了Python變量的命名規則（使用字母、數字和下劃線，不能以數字開頭，區分大小寫），以及如何避免常見命名錯誤。良好的變量命名習慣能夠使代碼更加清晰易讀。

□ 字符串的操作與處理：字符串是Python中最常用的數據類型之一，用於處理文本數據。我們掌握了字符串的基本操作，包括：

- 使用方法修改字符串大小寫（如upper()、lower()、title()）。
- 合併（拼接）字符串。
- 使用製表符和換行符添加空白。
- 刪除字符串中不需要的空白（如strip()、lstrip()、rstrip()）。
- 避免字符串操作中的常見語法錯誤。

□ 數字類型及其操作：我們學習了Python中的兩種主要數字類型，整數（用於表示沒有小數部分的數字）和浮點數（用於表示帶小數點的數字）；我們還了解了基本的數學運算（加、減、乘、除、乘方等）以及如何使用str()函數避免類型錯誤。

□ 註釋的編寫與使用：註釋是程序中的非執行部分，用於解釋代碼的工作原理或提供其他信息。我們學習了：

- 如何使用井號（#）添加單行註釋。
- 如何編寫多行註釋和文檔字符串。
- 什麼時候應該添加註釋以及應該包含什麼內容。
- 如何編寫有效且有用的註釋。

通過DeepSeek，我們不僅學習了這些基本概念，還了解了更多高級用法和最佳實踐。DeepSeek提供的詳細解釋和示例能幫助我們更深入地理解這些概念，並將它們應用到實際編程中。

在下一章中，我們將學習Python中的列表，這是一種更復雜的數據類型，允許我們存儲和管理多個相關項目。掌握了本章的基礎知識後，下一步我們將迎接更復雜的編程概念和技術。

## 第5章

# 用DeepSeek學Python列表

在本章中，你將學習Python列表，這是一種強大的數據結構，支持在一個變量中存儲多個元素。你將瞭解列表的基本概念，學習如何使用列表、訪問其中的元素以及如何修改列表的內容。這些技能將幫助你高效處理多個相關數據，為後續的數據處理和操作奠定堅實的基礎。

DeepSeek可以幫助你理解相關概念，同時提供實用的代碼示例和練習。在實踐過程中，DeepSeek還會幫助你識別並解決可能出現的問題。

學習本章後，你將邁出從處理單個數據項到處理數據集合的重要一步，為解決更復雜的編程問題奠定基礎。

## 5.1 本章學習目標

本章的核心目標是幫助你掌握Python列表這一強大的數據結構，使你能夠有效管理和操作多個相關數據。學習本章，你將掌握以下內容或技能：

- ❑ 理解列表的本質：深入理解Python中的基礎數據結構——列表，如何在單個變量中存儲多個元素，以及列表與其他數據類型的區別和聯繫。

- ❑ 掌握列表操作技巧：熟練掌握列表的創建、修改、添加和刪除元素的方法，能夠靈活運用索引和切片技術精確獲取列表數據。

- ❑ 應用列表排序與組織：學會使用內置函數對列表進行排序，培養數據管理的基本能力。

- ❑ 預防常見錯誤：瞭解使用列表時的常見陷阱（如索引錯誤等）及其解決方法，提升調試和問題解決能力。

## 5.2 Python列表

列表是Python中最常用的數據結構之一，它允許存儲一系列相關的項目。與字符串（存儲字符序列）和數字（存儲單個值）不同，列表可以存儲任意類型的數據，包括數字、字符串，甚至其他列表。列表的靈活性使其在處理多個相關數據時非常方便。

### 5.2.1 列表的定義與特點

下面讓我們通過DeepSeek的輔助來學習列表的相關定義與特點。

用戶提問：

什么是 Python 列表？它有什么特点？

DeepSeek回答：

#### 一、什么是 Python 列表？

列表 (List) 是 Python 中一种常用的数据结构，用于存储一系列相关的项目。它允许你在一个变量中存储多个不同的数据类型，包括数字、字符串、甚至其他列表。列表是 Python 内建的数据类型之一，非常灵活和强大。

#### 二、列表的特点

**有序：**列表中的元素按顺序排列，可以通过索引访问元素。索引从 0 开始。

**可变：**列表是可变的，意味着你可以修改、删除、添加元素，而不需要重新创建整个列表。

**可以包含不同类型的元素：**列表中的元素可以是不同的数据类型，例如整数、字符串、浮动数值、布尔值，甚至是其他列表。

**允许重复元素：**列表可以包含重复的元素，它们在列表中会按原始顺序出现。

**支持嵌套：**列表中的元素可以是其他列表，这样可以创建多维列表（嵌套列表）。

示例

创建一个列表

```
my_list = [1, 2.5, "hello", [3, 4], True]
```

访问列表中的元素

```
print(my_list[0]) # 输出 1
```

```
print(my_list[3]) # 输出 [3, 4]
```

修改列表中的元素

```
my_list[1] = 5
print(my_list) # 輸出 [1, 5, 'hello', [3, 4], True]
```

三. 常用操作

添加元素: 使用 `append()`、`insert()`。

刪除元素: 使用 `remove()`、`pop()`、`del`。

切片: 使用 `list[start:end]`。

查找元素: 使用 `index()`、`in` 操作符。

列表是 Python 中非常基础且重要的容器类型, 可以在许多应用中提供便利的解决方案。

列表的特點使得它非常適合處理需要存儲多個相關元素的場景，比如在一個應用中需要存儲用戶輸入的多個項，或者在數據分析時需要處理一系列數值或字符串數據。學習如何使用Python列表為後續更復雜的數據處理打下了堅實的基礎，同時也能提升我們在實際編程中的效率。

## 5.2.2 如何創建列表

在Python中，列表是一種常用且靈活的數據結構，用方括號[]表示。列表中的元素用逗號(,)分隔。下面通過DeepSeek的幫助來學習如何創建和使用列表。

用戶提問：

如何创建一个列表？可以举几个例子吗？

DeepSeek回答：

### 创建列表的方法

在 Python 中，我们可以通过以下方式来创建一个列表：

#### 1. 存储数字的列表

我们可以创建一个包含多个数字的列表：

```
numbers = [1, 2, 3, 4, 5]
```

#### 2. 存储字符串的列表

字符串列表可以包含多个字符串：

```
fruits = ["apple", "banana", "cherry"]
```

#### 3. 混合类型的列表

一个列表可以包含不同类型的元素，如数字、字符串、浮动数值、布尔值等：

```
mixed = [1, "hello", 3.14, True]
```

#### 4. 空列表

你也可以创建一个空的列表，之后可以动态地向其中添加元素：

```
empty_list = []
```

#### 5. 嵌套列表

列表中的元素还可以是另一个列表，这样可以创建多维列表（嵌套列表）：

```
nested = [[1, 2], [3, 4], [5, 6]]
```

根據DeepSeek的回答，我們知道，在Python中，列表是一種非常靈活的容器類型，能夠存儲不同類型的元素，甚至支持嵌套結構。無論是簡單的數據集合還是複雜的多維數據，列表都能應對自如。通過上面的示例，我們可以看到，列表不僅可以存儲數字、字符串等基本數據類型，還能夠存儲混合類型的數據，甚至是其他列表。

## 5.3 修改、添加和刪除元素

列表的一個重要特性是它是可變的，這意味著我們可以在創建列表後修改其內容。在本節中，我們將學習如何修改列表中的元素、添加新元素以及刪除不需要的元素。

### 5.3.1 修改列表元素

修改列表元素非常簡單，只需使用索引指定要修改的元素，然後為其賦予新值：

```
```python
# 创建一个水果列表
fruits = ["apple", "banana", "cherry"]
print(fruits)          # 输出: ['apple', 'banana', 'cherry']

# 修改第二个元素
fruits[1] = "blueberry"
print(fruits)         # 输出: ['apple', 'blueberry', 'cherry']
```
```

我們也可以使用負索引來修改元素：

```
```python
fruits = ["apple", "banana", "cherry"]
fruits[-1] = "date"   # 修改最后一个元素
print(fruits)        # 输出: ['apple', 'banana', 'date']
```
```

修改列表元素使我們能夠更新數據，而不必創建一個全新的列表。這在處理動態變化的數據時特別有用。

### 5.3.2 在列表中添加元素

Python提供了多種方法來向列表中添加元素。最常用的方法是append()、insert()和extend()。

## 1.使用append()方法添加元素

append()方法將元素添加到列表的末尾。

```
```python
fruits = ["apple", "banana", "cherry"]
fruits.append("date")
print(fruits) # 輸出: ['apple', 'banana', 'cherry', 'date']
```
```

append()方法是向列表添加元素的最簡單方法，特別是當你需要在列表末尾添加元素時。

## 2.使用insert()方法在特定位置添加元素

如果你需要在列表的特定位置添加元素，可以使用insert()方法。這個方法需要兩個參數：要插入元素的索引和要插入的值。

```
```python
fruits = ["apple", "banana", "cherry"]
fruits.insert(1, "blueberry") # 在索引 1 的位置插入 "blueberry"
print(fruits) # 輸出: ['apple', 'blueberry', 'banana', 'cherry']
```
```

insert()方法會將所有後續元素向右移動一個位置。如果指定的索引超出了列表的長度，元素將被添加到列表的末尾。

```
```python
fruits = ["apple", "banana", "cherry"]
fruits.insert(10, "date") # 索引 10 超出了列表长度
print(fruits) # 輸出: ['apple', 'banana', 'cherry', 'date']
```
```

## 3.使用extend()方法合併列表

如果你想將一個列表的所有元素添加到另一個列表的末尾，可以使用`extend()`方法。

```
```python
fruits = ["apple", "banana", "cherry"]
more_fruits = ["date", "elderberry"]
fruits.extend(more_fruits)
print(fruits) # 輸出: ['apple', 'banana', 'cherry', 'date', 'elderberry']
```
```

`extend()`方法與簡單地使用加號（+）運算符連接兩個列表的效果相似，但有一個重要區別：`extend()`方法修改了原始列表，而加號運算符創建了一個新列表。

```
```python
fruits1 = ["apple", "banana", "cherry"]
fruits2 = ["date", "elderberry"]
# 使用 extend() 方法
fruits1.extend(fruits2)
print(fruits1) # 輸出: ['apple', 'banana', 'cherry', 'date', 'elderberry']
# 使用 + 运算符
fruits3 = ["apple", "banana", "cherry"]
fruits4 = ["date", "elderberry"]
fruits5 = fruits3 + fruits4
print(fruits3) # 輸出: ['apple', 'banana', 'cherry'] (原列表不变)
print(fruits5) # 輸出: ['apple', 'banana', 'cherry', 'date', 'elderberry']
```
```

### 5.3.3 從列表中刪除元素

Python提供了多種方法來刪除列表中的元素。最常用的方法是`pop()`、`remove()`和`del`語句。

#### 1.使用`pop()`方法刪除元素

`pop()`方法刪除列表中的一個元素，並返回這個元素的值。如果不指定索引，`pop()`方法將刪除並返回列表的最後一個元素。

```
```python
fruits = ["apple", "banana", "cherry"]
last_fruit = fruits.pop()
print(last_fruit)           # 输出: cherry
print(fruits)              # 输出: ['apple', 'banana']
```
```

你也可以指定要删除的元素的索引。

```
```python
fruits = ["apple", "banana", "cherry"]
second_fruit = fruits.pop(1) # 删除并返回索引1处的元素
print(second_fruit)         # 输出: banana
print(fruits)              # 输出: ['apple', 'cherry']
```
```

## 2.使用remove()方法删除元素

如果你知道要删除的元素的值，但不知道它的索引，可以使用remove()方法。

```
```python
fruits = ["apple", "banana", "cherry"]
fruits.remove("banana")
print(fruits) # 输出: ['apple', 'cherry']
```
```

需要注意的是，remove()方法只会删除第一个匹配的元素。如果列表中有多个相同的值，只有第一个会被删除。

```
```python
fruits = ["apple", "banana", "cherry", "banana"]
fruits.remove("banana")
print(fruits) # 输出: ['apple', 'cherry', 'banana']
```
```

## 3.使用del语句删除元素

del语句可以删除列表中的单个元素或一系列元素。

```
```python
fruits = ["apple", "banana", "cherry", "date", "elderberry"]
del fruits[1] # 删除索引 1 处的元素
print(fruits) # 输出: ['apple', 'cherry', 'date', 'elderberry']
del fruits[1:3] # 删除索引 1 到 2 的元素 (不包括索引 3)
print(fruits) # 输出: ['apple', 'date', 'elderberry']
del fruits # 删除整个列表
# print(fruits) # 错误: NameError: name 'fruits' is not defined
```
```

## 4. 清空列表

如果你想删除列表中的所有元素但保留列表本身，可以使用 `clear()` 方法。

```
```python
fruits = ["apple", "banana", "cherry"]
fruits.clear()
print(fruits) # 输出: []
```
```

通過這些方法，我們可以靈活地管理列表中的元素，根據需要添加、修改和刪除元素。這種動態性使列表成為Python中最常用的數據結構之一，適用於各種編程場景。

## 5.4 Python列表的高級技巧

Python提供了強大的列表排序功能，使用起來既簡單又靈活。除了基本的排序方法外，還有一些高級技巧，可以幫助我們進行更高效、更靈活的排序操作。下面通過用戶提問和DeepSeek回答的編寫方式，介紹一些常見的高級排序技巧。讓我們向DeepSeek詢問更多關於Python列表排序的高級技巧。

用戶提問：

Python 列表排序有哪些高級技巧？如何在实际开发中使用？

DeepSeek回答：

### Python 列表排序的高级技巧

Python 提供了强大而灵活的列表排序功能，可以处理从简单数据类型到复杂对象集合的各种排序需求。以下是一些高级排序技巧和方法。

#### 排序基础回顾

Python 提供两种主要的排序方法：

`list.sort()` - 原地排序，修改原列表

`sorted()` - 返回新的排序列表，不修改原列表

两者都接受相同的关键参数：

`reverse=True/False` - 控制排序方向

`key=function` - 指定排序依据的函数

对包含字典的列表排序

按单个键排序：

```
```python
people = [
    {"name": "Alice", "age": 25, "height": 165},
    {"name": "Bob", "age": 30, "height": 180},
    {"name": "Charlie", "age": 22, "height": 175}
]
```

按年龄排序

```
people_by_age = sorted(people, key=lambda x: x["age"])
```

```
print(people_by_age)
```

按姓名排序

```
people_by_name = sorted(people, key=lambda x: x["name"])
```

```
print(people_by_name)
```

按多个键排序（使用元组）：

```
```python
先按年龄，再按身高排序
people_by_age_height = sorted(people, key=lambda x: (x["age"], x["height"]))
print(people_by_age_height)
不同方向排序（年龄升序，身高降序）
people_mixed = sorted(people, key=lambda x: (x["age"], -x["height"]))
print(people_mixed)
```
```

根據DeepSeek的回答，我們知道，Python提供了多種靈活且強大的排序技巧，幫助我們滿足各種排序需求。通過使用key參數，我們可以自定義排序規則；通過reverse參數，我們可以實現倒序排序；Python的穩定排序特性能夠保留元素原有的順序。此外，heapq模塊提供了高效的方式來獲取列表中的前幾個最大或最小元素。

掌握這些高級排序技巧可以讓你在實際開發中高效地處理數據。無論是簡單的數值排序，還是更複雜的字典、對象排序，

Python都為我們提供了靈活且高效的解決方案。

5.5 通過DeepSeek學習Python索引

在前面的章節裡，我們已經掌握了列表的基本操作和常用方法。接下來，本節將聚焦於列表索引這一核心概念：如何定位、訪問並操作列表中的單個元素。由於Python（以及許多其他編程語言）都採用從0開始的索引方式，初學者往往會感到疑惑。為此，我們將藉助DeepSeek的解釋和示例，深入剖析0基索引的技術原理與歷史背景，從而幫助你徹底理解為何這種設計如此高效且被廣泛沿用。

5.5.1 索引的基礎概念

在正式探討為什麼索引從0開始之前，我們先複習幾個常見的索引概念。

Python索引就是列表中每個元素的位置標識，從0開始計數。

下面我將結合代碼示例介紹Python索引的基礎概念。

```
```python
fruits = ['蘋果', '香蕉', '橙子']
print(fruits[0]) # 輸出 '蘋果'
```

#### (1) 負索引

從列表末尾開始計數，-1表示最後一個元素，-2表示倒數第二個，以此類推。例如下面的代碼示例：

```
print(fruits[-1]) # 輸出 '橙子'
```

#### (2) 切片

通過開始索引：結束索引取出子列表，左閉右開。

```
numbers = [0, 1, 2, 3, 4, 5]
print(numbers[1:4]) # 輸出 [1, 2, 3]
print(numbers[:3]) # 輸出 [0, 1, 2]
print(numbers[3:]) # 輸出 [3, 4, 5]
```

### (3) 常用方法

□ `list.index(value)`：返回元素在列表中首次出現的索引。

□ `len(list)`：列表長度，即元素總數。

掌握了這些基本概念後，再來探討“為什麼要從0開始索引”就會更容易理解了。

## 5.5.2 從0開始的Python索引

對於初學者來說，列表索引從0開始而不是1開始可能有些不直觀。讓我們通過DeepSeek來進行輔理解為什麼Python（以及許多其他編程語言）使用從0開始的索引。

從0開始的索引實際上有其數學和計算機科學上的合理性。在內存中，列表元素是連續存儲的，索引表示從列表起始位置的偏移量。第一個元素的偏移量是0（沒有偏移），第二個元素的偏移量是1，依此類推。

這種索引方式也使得某些計算變得更簡單。例如，如果你想知道列表中有多少個元素，可以使用最後一個元素的索引加1（因為索引從0開始）。

讓我們向DeepSeek詢問更多關於從0開始索引的原因：

用戶提問：

为什么 Python 和许多其他编程语言的列表或数组索引从 0 开始而不是 1 开始？这有什么历史原因或技术优势吗？

## DeepSeek回答：

### 一、历史原因

#### 1. 早期硬件和汇编语言的影响

在早期计算机中，数组的索引实际上是内存地址的偏移量。第一个元素位于基地址加 0 的位置，第二个元素位于基地址加 1 的位置，依此类推。这种直接映射到硬件的方式使得计算变得更加高效。

#### 2. C 语言的影响

C 语言采用了 0 基索引，并且这一设计影响了后来的许多语言。C 语言的创造者 Dennis Ritchie 继承了 BCPL 语言的这一特性。由于 C 语言在计算机科学中的广泛应用，许多现代编程语言（包括 Python、Java、JavaScript）也都延续了这一设计。

#### 3. BCPL 和 B 语言的传统

BCPL (C 语言的前身) 使用 0 基索引，主要是为了简化指针算术。BCPL 的创造者 Martin Richards 选择 0 基索引，是因为它与指针运算自然匹配。

### 二、技术优势

#### 1. 数学上的优雅性

数学中，区间  $[0, n)$  代表  $n$  个元素，长度正好是  $n$ 。当我们使用 0 基索引时，计算子数组的长度非常简单： $end - start$ 。如果从 1 开始，则需要额外的加 1 操作。

#### 2. 简化模运算和循环索引

使用 0 基索引时，模运算和循环索引变得更自然。例如，循环条件  $index \% array\_length$  可以直接应用，而不需要调整索引。

#### 3. 指针算术和内存寻址

在内存中，数组元素的地址可以通过公式  $地址 = 基地址 + (索引 \times 元素大小)$  来计算。0 基索引使这个公式更加简洁，无需额外的减法操作。

#### 4. 半开区间的一致性

许多算法使用半开区间  $[start, end)$  表示从  $start$  到  $end-1$  的元素，这与 0 基索引自然匹配。例如， $range(0, 5)$  产生  $[0, 1, 2, 3, 4]$ ，正好是 5 个元素。

#### 5. 简化循环边界条件

使用 0 基索引时，循环条件简化为  $i < n$ ，而不是  $i \leq n$ ，这减少了常见的“差一”错误 (off-by-one error) 的可能性。

讓我們通過實際的例子來進一步理解。

(1) 考慮一個長度為5的數組

使用0基索引： $[0, 1, 2, 3, 4]$

使用1基索引： $[1, 2, 3, 4, 5]$

(2) 計算子數組長度

```

```python
# 0基索引
sub_array = array[2:4] # 包含索引 2 和 3 的元素
length = 4 - 2 = 2 # 正好是元素数量

# 1基索引 (假设)
sub_array = array[2:4] # 包含索引 2 和 3 的元素
length = 4 - 2 + 1 = 3 # 需要额外 +1 操作
```

```

### (3) 內存尋址計算

```

```
# 0基索引
元素地址 = 基地址 + (索引 × 元素大小)
array[3] 的地址 = 基地址 + (3 × 元素大小)

# 1基索引 (假设)
元素地址 = 基地址 + ((索引 - 1) × 元素大小)
array[3] 的地址 = 基地址 + ((3-1) × 元素大小)
```

```

### (4) 例外情況

並非所有語言都使用0基索引。

□ FORTRAN：使用1基索引，主要面向科學和工程計算，更接近數學符號。

□ COBOL：允許程序員定義數組的下界。

□ Lua：默認使用1基索引，更符合其設計哲學。

□ R：使用1基索引，因為它主要面向統計學家。

□ MATLAB/Octave：使用1基索引，更接近線性代數符號。

0基索引雖然對初學者可能不夠直觀（因為我們習慣從1開始計數），但它在計算機科學中有深厚的歷史根源和技術優勢。它簡化了內存尋址、區間計算和循環條件，使代碼更簡潔高效。

一旦習慣了0基索引的思維方式，你會發現它在許多編程場景中確實更加自然和高效。

根據DeepSeek的回答，我們知道，Python和許多編程語言採用從0開始的索引，並非偶然。這一設計背後有深刻的歷史原因，主要受到早期硬件和彙編語言的影響。同時，0基索引在數學、計算機科學和內存尋址上有很多技術優勢，它不僅簡化了計算，還提高了代碼的效率和可讀性。

### 5.5.3 使用DeepSeek解決列表索引錯誤

在使用列表時，一個常見的錯誤是嘗試訪問不存在的索引，這會導致IndexError錯誤。例如：

```
fruits = ["apple", "banana", "cherry"]
print(fruits[3]) # 错误: IndexError: list index out of range
```

在這個例子中，我們嘗試訪問索引3處的元素，但列表fruits只有3個元素（索引0、1和2）。當Python無法找到指定的索引時，會引發IndexError錯誤。

#### 1. 索引錯誤產生的原因

索引錯誤通常有以下幾種常見原因：

- ❑ 忘記索引從0開始：如果列表有  $n$  個元素，有效的索引範圍是0到  $n - 1$ ，而不是1到  $n$ 。

- ❑ 使用了超出範圍的索引：嘗試訪問超出列表長度的索引。

- ❑ 使用了負索引，但超出了列表長度：例如，對於一個只有3個元素的列表，嘗試訪問list[-4]。

❑ 在空列表上使用索引：嘗試訪問空列表中的元素。

## 2. 避免索引錯誤的策略

在Python中，處理列表索引錯誤是常見的挑戰，尤其是當索引超出了有效範圍時。為了避免這些錯誤，Python提供了幾種處理策略和最佳實踐。除了基本的try-except和列表長度檢查外，還有一些更優雅和高效的方式來確保代碼的穩定性。

以下是一些避免索引錯誤的策略。

### (1) 檢查列表長度

在訪問列表元素之前，可以先檢查索引是否在有效範圍內。

```
fruits = ["apple", "banana", "cherry"]
index = 3
if index < len(fruits):
 print(fruits[index])
else:
 print(f"索引 {index} 超出了列表範圍。")
```

### (2) 使用try-except語句

你可以使用try-except語句來捕獲並處理索引錯誤。

```
fruits = ["apple", "banana", "cherry"]
try:
 print(fruits[3])
except IndexError:
 print("索引超出了列表範圍。")
```

### (3) 使用get()方法（適用於字典，不適用於列表）

雖然列表沒有get()方法，但這種思想可以通過自定義函數實現。

```
def safe_get(lst, index, default=None):
 """ 安全地获取列表中的元素，如果索引超出范围则返回默认值。"""
 try:
 return lst[index]
 except IndexError:
 return default
fruits = ["apple", "banana", "cherry"]
print(safe_get(fruits, 3, "未知水果")) # 输出: 未知水果
```

#### (4) 使用條件表達式

對於簡單的情況，可以使用條件表達式。

```
fruits = ["apple", "banana", "cherry"]
index = 3
result = fruits[index] if index < len(fruits) else "索引超出范围"
print(result) # 输出: 索引超出范围
```

#### (5) 使用列表的切片操作

切片操作不會引發索引錯誤，即使索引超出了列表範圍。

```
fruits = ["apple", "banana", "cherry"]
print(fruits[3:]) # 输出: [] (空列表，而不是错误)
print(fruits[:10]) # 输出: ['apple', 'banana', 'cherry'] (包含所有元素)
```

讓我們向DeepSeek詢問更多關於處理列表索引錯誤的最佳實踐：

用戶提問：

```
在 Python 中，除了使用 try-except 和检查长度外，还有哪些处理列表索引错误的最佳实践？
```

DeepSeek回答：

### Python 列表索引错误的高级处理策略

处理列表索引错误是 Python 编程中的常见挑战。除了基本的 try-except 和长度检查外，还有许多更优雅、更高效的方法来处理这类问题。以下是一些高级策略和最佳实践。

使用迭代而非索引访问

#### 一、直接迭代

最简单的避免索引错误的方法是完全避免使用索引：

不推荐

```
fruits = ["apple", "banana", "cherry"]
for i in range(len(fruits)):
 print(fruits[i])
```

推荐

```
for fruit in fruits:
 print(fruit)
```

#### 二、使用 enumerate() 获取索引和值

如果你确实需要索引信息，可以使用 enumerate() 函数，这样你可以在遍历列表时同时获取索引和值。enumerate() 是一个非常强大的工具，既能避免索引错误，又能保留索引信息。

示例：

```
fruits = ["apple", "banana", "cherry"]
for i, fruit in enumerate(fruits):
 print(f"{i}: {fruit}")
```

根據DeepSeek的回答，我們知道，除了常規的try-except語句和索引長度檢查外，Python還提供了多種更優雅的策略來避免索引錯誤。這些方法包括使用迭代訪問元素，通過enumerate()獲取索引和值，使用切片操作處理超出範圍的索引等。每種方法都有其適用場景，掌握這些技巧可以使代碼更加簡潔、健壯。

通過避免直接使用索引訪問列表或通過其他高級技巧處理索引錯誤，我們能夠編寫出更穩定、更高效的代碼。在實際開發中，選擇適合的錯誤處理策略，可以大大減少程序出錯的概率，並提高代碼的可維護性和可讀性。

## 5.6 結語

本章我們深入探討了Python中的列表，這是一種強大而靈活的數據結構，允許我們存儲和管理多個相關項目。通過本章的學習，我們掌握了以下幾個關鍵概念：

□ 列表的基本概念：我們瞭解了列表是什麼，以及如何創建和訪問列表元素。列表是有序的、可變的數據集合，可以存儲任意類型的數據。Python中的索引從0開始，這是一個需要特別注意的點。

□ 列表元素的操作：我們學習瞭如何修改、添加和刪除列表元素，包括：

- 使用索引修改列表元素。
- 使用append()、insert()和extend()方法添加元素。
- 使用del語句、pop()和remove()方法刪除元素。

□ 避免索引錯誤：我們學習瞭如何避免和處理列表索引錯誤，包括檢查列表長度，使用異常處理，以及採用更安全的列表訪問方式。

通過DeepSeek的輔助，我們不僅學習了這些基本概念，還了解了更多高級用法和最佳實踐。例如，我們探討了如何對包含複雜對象的列表進行排序，以及如何在實際項目中優雅地處理索引錯誤。

列表是Python編程中最常用的數據結構之一，掌握列表的操作對於解決各種編程問題至關重要。列表允許我們以結構化的方

式組織數據，使代碼更加清晰和高效。

在下一章中，我們將學習如何操作列表，包括如何遍歷列表，如何使用for循環，以及如何創建數值列表。這些知識將進一步增強我們處理數據集合的能力，為解決更復雜的編程問題奠定基礎。

## 第6章

# 用DeepSeek學操作列表

在本章中，你將學習如何充分發揮Python列表的強大功能，探索如何遍歷整個列表，使用for循環處理列表中的每個元素，以及如何創建數值列表，並藉助DeepSeek進行使用操作列表。

DeepSeek可以幫助你理解相關概念，同時提供實用的代碼示例和最佳實踐。當你實踐這些技術時，DeepSeek還會幫助你識別並解決可能出現的問題。

學習本章後，你將能夠更加靈活地處理數據集合，解決更複雜的編程問題。這些技能將為後續的Python學習之旅奠定堅實的基礎。

## 6.1 本章學習目標

本章的核心目標是幫助你掌握Python列表的高級操作技巧，特別是列表遍歷和處理的方法。學習本章，你將掌握以下內容或技能：

- ❑ 精通列表遍歷技術：深入理解並熟練應用for循環遍歷列表的機制，能夠自信地處理列表中的每個元素，實現批量數據處理。

- ❑ 創建和操作數值列表：掌握生成數值序列的多種方法，包括使用range()函數、列表推導式等技術，能夠靈活創建滿足特定需求的數值數據集。

- ❑ 提升調試能力：識別並解決列表遍歷和處理中的常見邏輯錯誤，增強代碼調試和問題解決能力。

## 6.2 遍歷整個列表

在處理列表時，一個常見的任務是遍歷列表中的所有元素，對每個元素執行相同的操作。Python的for循環提供了一種簡潔而強大的方式來完成這一任務。

### 6.2.1 深入研究循環

for循環是Python中最常用的循環結構之一，它允許我們對列表中的每個元素執行一系列操作。for循環的基本語法如下：

```
for item in list_of_items:
 # 对 item 执行操作
```

這裡，item是一個變量名，用於存儲列表中的當前元素；list\_of\_items是要遍歷的列表；冒號(:)表示循環頭的結束；縮進的代碼塊是循環體，包含要對每個元素執行的操作。

讓我們看一個簡單的例子，遍歷一個水果列表並打印每種水果的名稱。

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
 print(fruit)
輸出結果:
apple
banana
cherry
```

在這個例子中，Python從列表fruits中取出第一個元素("apple")，將其存儲在變量fruit中，然後執行循環體中的代碼(打印fruit的值)。然後，Python取出第二個元素("banana")，將

其存儲在變量fruit中，再次執行循環體。這個過程會一直持續，直到處理完列表中的所有元素。

## 1. 循環變量的命名

在for循環中，可以使用任何有效的變量名作為循環變量。通常，我們會選擇一個有意義的名稱，反映列表中元素的類型。例如，如果列表包含水果名稱，可以使用fruit作為循環變量；如果列表包含用戶名，可以使用user或username作為循環變量。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")
```

輸出結果：

```
Alice is a great magician!
David is a great magician!
Carolina is a great magician!
```

## 2. 循環的工作原理

為了更好地理解for循環的工作原理，讓我們向DeepSeek詢問更多細節。

用戶提問：

```
Python 的 for 循环与其他编程语言的循环有什么不同？它在内部是如何工作的？
```

DeepSeek回答：

Python的for循环：独特性与内部工作原理

Python的for循环与许多其他编程语言中的for循环有显著不同，这种独特设计使它更加强大和灵活。

Python for循环与其他语言的区别，基于迭代器而非计数器。

Python的for循环：

```
for item in collection:
 # 处理 item
```

传统C/Java风格的for循环：

```
for (int i = 0; i < collection.length; i++) {
 // 使用 collection[i]
}
```

Python的for循环直接遍历集合中的元素，而不是使用索引变量。具有更广泛的适用，另外Python的for循环可以遍历任何可迭代对象，包括：

- 列表、元组、字符串、集合、字典。
- 文件对象。
- 生成器和迭代器。
- 自定义可迭代对象。

而传统for循环通常仅用于数值范围或通过索引访问集合，并且Python具有更简洁的语法。

Python的for循环语法更简洁，不需要显式的初始化、条件检查和增量步骤，没有C风格的"for(;;)"无限循环。

Python使用while True:来实现无限循环，而不是空条件的for循环。

### 3. Python for循环的内部工作原理

Python的for循环在内部是通过迭代器协议实现的，大致遵循以下步骤。

1) 获取迭代器。

当执行for item in collection:时，Python首先调用iter(collection)获取一个迭代器对象。

for循环的第一步等价于：

```
iterator = iter(collection)
```

2) 重复调用next()。

Python重复调用迭代器的next()方法（在Python 3中是\_\_next\_\_()），直到引发StopIteration异常。

循環主體等價於：

```
try:
 while True:
 item = next(iterator)
 # 執行循環體代碼
except StopIteration:
 pass # 循環結束
```

3) 異常處理。

當迭代器中沒有更多元素時，next()調用會引發StopIteration異常，Python捕獲這個異常並正常退出循環。

實際示例：解析for循環的執行過程。

讓我們通過一個簡單的例子來說明for循環的內部執行過程。

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
 print(fruit)
```

內部執行過程：

①Python調用iter(fruits)獲取一個迭代器。

②Python調用next(iterator)獲取第一個元素“apple”，將其賦值給變量fruit。

③執行循環體：打印“apple”。

④Python再次調用next(iterator)獲取下一個元素“banana”，將其賦值給變量fruit。

⑤執行循環體：打印“banana”。

⑥ Python再次調用next(iterator)獲取下一個元素“cherry”，將其賦值給變量fruit。

⑦ 執行循環體：打印“cherry”。

⑧ Python再次調用next(iterator)，但沒有更多元素，引發StopIteration異常。

⑨ Python捕獲異常並退出循環。

4) 手動模擬for循環。

我們可以手動編寫代碼來模擬for循環的行為：

```
fruits = ["apple", "banana", "cherry"]
手動模擬 for 循環
iterator = iter(fruits)
try:
 while True:
 fruit = next(iterator)
 print(fruit)
except StopIteration:
 pass # 循環結束
```

這段代碼的輸出與使用for循環完全相同。

## 4.性能考慮

Python的for循環通常比等效的while循環更高效，因為：

- ❑ 迭代器協議是在C層面實現的。
- ❑ 避免了顯式索引計算和邊界檢查。
- ❑ 對於大型集合，可以實現惰性求值（不需要一次性加載所有元素）。

Python的for循環是一個強大而靈活的構造，它的一些優勢如下：

- 基於迭代器協議，可以遍歷任何可迭代對象。
- 語法簡潔，直接處理元素而非索引。
- 內部通過iter()和next()方法實現。
- 提供了額外的功能，如else子句和與enumerate()/zip()的集成。

這種設計使Python的for循環比傳統的計數器循環更易用、更靈活，也更符合Python“優雅簡潔”的設計哲學。

## 6.2.2 在for循環中執行更多的操作

for循環的真正強大之處在於，我們可以在循環體中執行任意數量的操作。例如，我們可以在打印魔術師名字的同時，向每位魔術師發送個性化的消息。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")
 print(f"I can't wait to see your next trick, {magician.title()}.\n")
```

輸出結果：

```
Alice is a great magician!
I can't wait to see your next trick, Alice.

David is a great magician!
I can't wait to see your next trick, David.

Carolina is a great magician!
I can't wait to see your next trick, Carolina.
```

在這個例子中，循環體包含兩個print語句。對於列表中的每個魔術師，Python都會執行這兩個print語句。注意，我們在第二個print語句末尾添加了\n，這會在每個魔術師的消息後添加一個空行，使輸出更加清晰。

我們還可以在循環中執行更復雜的操作，例如條件判斷、計算或調用函數。

```
scores = [85, 92, 78, 90, 88]
for score in scores:
 if score >= 90:
 print(f"Score: {score} - Excellent!")
 elif score >= 80:
 print(f"Score: {score} - Good!")
 else:
 print(f"Score: {score} - Needs improvement")
```

輸出結果：

```
Score: 85 - Good!
Score: 92 - Excellent!
Score: 78 - Needs improvement
Score: 90 - Excellent!
Score: 88 - Good!
```

這個例子展示瞭如何在循環中使用條件語句來根據不同的分數給出不同的評價。

### 6.2.3 在for循環結束後執行一些操作

for循環完成後，我們通常需要執行一些總結性的操作。這些操作應該放在循環之後，與循環體保持相同的縮進級別。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")

print("\nThank you, everyone. That was a great magic show!")
```

輸出結果：

```
Alice is a great magician!
David is a great magician!
Carolina is a great magician!

Thank you, everyone. That was a great magic show!
```

在這個例子中，最後一條消息只打印一次，因為它不是循環體的一部分。它的縮進級別與for語句相同，表示它是循環之後的代碼。

我們還可以在循環後使用循環中收集的數據：

```
计算学生成绩的平均分
scores = [85, 92, 78, 90, 88]
total = 0

for score in scores:
 total += score

average = total / len(scores)
print(f"The average score is: {average}") # 輸出: The average score
is: 86.6
```

在這個例子中，我們在循環中累加所有分數，然後在循環結束後計算平均分。這是一個常見的模式：在循環中收集或處理數據，在循環結束後使用這些數據進行計算或生成報告。

通過掌握for循環的使用，我們可以高效地處理列表中的數據，無論是簡單的打印操作，還是複雜的數據分析和處理。在下一節中，我們將學習如何避免在使用循環時常見的縮進錯誤，這對於編寫正確的Python代碼至關重要。

## 6.3 避免縮進錯誤

Python使用縮進來表示代碼塊，這使得代碼結構清晰可見。然而，這也意味著縮進錯誤可能導致程序無法正常運行或產生意外的結果。在本節中，我們將學習如何識別和避免常見的縮進錯誤。

### 6.3.1 忘記縮進

在Python中，循環體中的每一行代碼都必須縮進。如果忘記縮進，Python會顯示縮進錯誤。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
print(f"{magician.title()} is a great magician!") # 错误: 缺少缩进
```

如果運行這段代碼，Python會顯示以下錯誤信息：

```
IndentationError: expected an indented block
```

這個錯誤表明Python期望在for語句後面有一個縮進的代碼塊，但沒有找到。要修復這個錯誤，只需要正確縮進循環體中的代碼。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!") # 正确: 代码已缩进
```

### 6.3.2 忘記縮進額外的代碼行

有時，我們可能希望在循環中執行多個操作，但忘記縮進其中的一些代碼行。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")
print(f"I can't wait to see your next trick, {magician.title()}.")
错误: 缺少缩进
```

在這個例子中，第二個print語句沒有縮進，這意味著它不是循環體的一部分。Python不會為每個魔術師執行這條語句，而是在循環完成後執行一次。更糟糕的是，由於magician變量只在循環體內有效，這段代碼可能會產生意外的結果或錯誤。

正確的代碼應該是：

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")
 print(f"I can't wait to see your next trick, {magician.
 title()}.") # 正确: 代码已缩进
```

### 6.3.3 不必要的縮進

如果你在不需要縮進的地方添加了縮進，Python也會顯示錯誤。

```
message = "Hello, Python!"
 print(message) # 错误: 不必要的缩进
```

運行這段代碼會顯示以下錯誤信息：

```
IndentationError: unexpected indent
```

這個錯誤表明Python在不期望縮進的地方發現了縮進。要修復這個錯誤，只需刪除不必要的縮進。

```
message = "Hello, Python!"
print(message) # 正确: 没有不必要的缩进
```

## 6.3.4 循環後不必要的縮進

另一個常見的錯誤是在循環後出現了不必要的縮進。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")

 print("Thank you, everyone. That was a great magic show!")
 # 错误: 不必要的缩进
```

在這個例子中，最後一條消息被錯誤地縮進，這意味著它是循環體的一部分，會為每個魔術師打印一次。如果我們只想在循環結束後打印一次這條消息，應該刪除縮進。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians:
 print(f"{magician.title()} is a great magician!")

print("Thank you, everyone. That was a great magic show!")
正确: 没有不必要的缩进
```

## 6.3.5 遺漏了冒號

在Python中，for語句和其他控制流語句（如if、while、def等）的末尾必須有一個冒號（:）。如果遺漏了冒號，Python會顯示語法錯誤。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians # 错误: 缺少冒号
 print(f"{magician.title()} is a great magician!")
```

運行這段代碼會顯示以下錯誤信息：

```
SyntaxError: invalid syntax
```

要修復這個錯誤，只需在for語句的末尾添加冒號。

```
magicians = ["alice", "david", "carolina"]
for magician in magicians: # 正确: 添加了冒号
 print(f"{magician.title()} is a great magician!")
```

讓我們向DeepSeek詢問更多關於Python縮進的最佳實踐。

用戶提問：

在 Python 中，关于缩进有哪些最佳实践？不同的编辑器和 IDE 如何帮助避免缩进错误？

DeepSeek回答：

### Python 缩进最佳实践与工具支持

Python 使用缩进来定义代码块，这是该语言的核心特性之一。正确的缩进不仅是语法要求，也是提高代码可读性的关键。以下是关于 Python 缩进的最佳实践以及编辑器 / IDE 如何帮助避免缩进错误。

#### Python 缩进的最佳实践

##### 1. 保持一致的缩进风格

PEP 8 建议：使用 4 个空格作为每级缩进。

```
```python
# 推荐
def function():
    if condition:
        do_something()

# 不推荐 (混合使用不同数量的空格)
def function():
    if condition:                # 3 个空格
        do_something()          # 6 个空格
...
```
```

##### 2. 使用空格而非制表符

PEP 8 建议：使用空格而非制表符 (Tab) 进行缩进。

```
```python
# 在大多数编辑器中设置：Tab 键插入 4 个空格，而不是制表符
...
```
```

#### 原因：

不同编辑器对制表符的显示宽度不同，混合使用空格和制表符可能导致难以察觉的缩进错误。

##### 3. 续行的缩进

当一行代码太长需要换行时，有两种常见的缩进方式：

```
```python
# 方法 1: 垂直对齐参数
def long_function_name(
    param1, param2,
    param3, param4):
    print(param1)

# 方法 2: 使用悬挂缩进，额外缩进 4 个空格
def long_function_name(

```

```
    param1, param2,
    param3, param4):
    print(param1)
...
```
```

現在，根據DeepSeek的回答，我們知道良好的縮進實踐不僅可以避免語法錯誤，還能提高代碼可讀性和可維護性。另外現代編輯器和IDE提供了豐富的功能來幫助開發者遵循Python的縮進規則：

- ❑ 遵循PEP 8標準：使用4個空格作為縮進單位。
- ❑ 保持一致性：在整個項目中使用相同的縮進風格。
- ❑ 利用工具：使用編輯器功能、代碼檢查器和格式化工具。
- ❑ 團隊協作：使用EditorConfig等工具統一團隊設置。
- ❑ 自動化：設置pre-commit鉤子和CI檢查。

結合這些最佳實踐和工具支持，可以有效地避免Python中的縮進錯誤，提高開發效率和代碼質量。另外通過理解和避免這些常見的縮進錯誤，我們可以編寫更加可靠和易讀的Python代碼。記住，在Python中，縮進不僅僅是風格問題，它直接影響代碼的執行邏輯。因此，養成良好的縮進習慣對於成為一名成功的Python程序員至關重要。

在下一節中，我們將學習如何創建數值列表，這是處理數值數據的重要技能。

## 6.4 創建數值列表

在編程中，我們經常需要處理數字序列。Python提供了多種創建和操作數值列表的方法，這些方法可以幫助我們高效地處理數值數據。在本節中，我們將學習如何創建數值列表，以及如何對這些列表執行基本的統計計算。

### 6.4.1 使用函數range()

Python的range()函數是一個非常有用的工具，它可以生成一系列的數字。這個函數最常用於控制for循環執行的次數。讓我們看一個簡單的例子。

```
for value in range(1, 5):
 print(value)
```

如果運行這段代碼，你會看到以下輸出：

```
1
2
3
4
```

注意，range()函數生成的數字序列從指定的第一個值開始，到指定的第二個值前面的那個數字結束。在這個例子中，range(1, 5)生成了數字1到4，但不包括5。

如果只向range()函數提供一個參數，它會從0開始生成數字。

```
for value in range(5):
 print(value)
```

這段代碼會輸出：

```
0
1
2
3
4
```

`range()`函數還可以接收第三個參數，用於指定步長（即相鄰數字之間的差值）。

```
for value in range(1, 10, 2):
 print(value)
```

這段代碼會輸出所有1到10之間的奇數。

```
1
3
5
7
9
```

你也可以使用負數步長來創建遞減的數字序列。

```
for value in range(10, 0, -1):
 print(value)
```

這段代碼會輸出從10倒數到1的數字。

```
10
9
8
7
6
```

```
5
4
3
2
1
```

## 6.4.2 使用`range()`創建數字列表

雖然`range()`函數本身不會直接創建一個列表，但我們可以使用`list()`函數將`range()`的結果轉換為列表。

```
numbers = list(range(1, 6))
print(numbers)
```

這段代碼會輸出：

```
[1, 2, 3, 4, 5]
```

這種方法非常適合創建包含特定範圍數字的列表。例如，如果我們想創建一個包含1到100的平方數的列表，可以這樣做：

```
squares = []
for value in range(1, 11):
 square = value ** 2
 squares.append(square)
print(squares)
```

這段代碼會輸出：

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

我們也可以使用更簡潔的方式來編寫這段代碼：

```
squares = []
for value in range(1, 11):
 squares.append(value ** 2)
print(squares)
```

### 6.4.3 對數字列表執行簡單的統計計算

Python提供了幾個專門用於處理數字列表的函數，可以幫助我們快速進行統計計算。

```
digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
print(min(digits)) # 輸出最小值: 0
```

```
print(max(digits)) # 輸出最大值: 9
print(sum(digits)) # 輸出总和: 45
```

這些函數使得對數字列表進行基本的統計分析變得非常簡單。

- ❑ `min()`函數返回列表中的最小值。
- ❑ `max()`函數返回列表中的最大值。
- ❑ `sum()`函數返回列表中所有值的總和。

#### 6.4.4 列表解析

列表解析是Python中一種創建列表的簡潔方法。它允許我們在一行代碼中創建列表，而不需要使用多行的for循環。

例如，我們可以使用列表解析來重寫前面創建平方數列表的代碼。

```
squares = [value ** 2 for value in range(1, 11)]
print(squares)
```

這段代碼會產生與之前相同的輸出。

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

列表解析由一個方括號包圍，其中包含一個表達式，後面跟著一個for子句，然後是零個或多個for或if子句。表達式可以是任何有效的Python表達式，它會為列表中的每個元素計算一個值。

讓我們看一些更復雜的列表解析示例。

```
创建一个包含 1 到 20 之间所有偶数的列表
even_numbers = [x for x in range(1, 21) if x % 2 == 0]
print(even_numbers) # 输出: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

创建一个包含 1 到 10 的立方数的列表
cubes = [x**3 for x in range(1, 11)]
print(cubes) # 输出: [1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

列表解析是Python中一個強大的特性，它可以使代碼更加簡潔和易讀。然而，對於非常複雜的操作，使用傳統的for循環可能更容易理解。

讓我們向DeepSeek詢問更多關於Python列表解析的高級用法。

用戶提問：

請介紹 Python 列表解析的一些高級用法和最佳實踐。

DeepSeek回答：

```
Python 列表解析的高级用法与最佳实践
列表解析 (List Comprehension) 是 Python 中一个强大而优雅的特性，它允许我们以简洁的方式创建和转换列表。虽然基本用法相对简单，但列表解析还有许多高级用法和需要注意的最佳实践。
1) 嵌套列表解析。
嵌套列表解析可以用来处理多维数据结构。
```python
# 创建一个 3x3 的矩阵
matrix = [[i * 3 + j + 1 for j in range(3)] for i in range(3)]
print(matrix) # 输出: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
2) 矩阵转置。
transposed = [[row[i] for row in matrix] for i in range(3)]
print(transposed) # 输出: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
3) 展平嵌套列表。
flattened = [num for row in matrix for num in row]
print(flattened) # 输出: [1, 2, 3, 4, 5, 6, 7, 8, 9]
注意嵌套列表解析的顺序: 外层循环在前, 内层循环在后, 这与嵌套 for 循环的顺序相反。
```

結合DeepSeek的回答，我們知道列表解析是Python中一個強大的特性，能夠讓代碼更簡潔、更易讀，並且在許多情況下性能更

好。關鍵是我們要明智地使用它。

- 對於簡單的轉換和過濾操作，列表解析通常是最佳選擇。
- 對於複雜的邏輯或多重嵌套，傳統的for循環可能更清晰。
- 對於大數據集，考慮使用生成器表達式。
- 始終優先考慮代碼的可讀性和可維護性。

掌握列表解析的高級用法和最佳實踐，可以幫助你編寫更加Python化、高效和易於維護的代碼。

通過和DeepSeek一起學習創建數值列表的方法，我們可以更有效地處理數值數據。range()函數和列表解析是Python中非常有用的工具，它們可以幫助我們簡化代碼並提高效率。

6.5 結語

恭喜你完成了本章的學習！你現在已經掌握了操作Python列表的核心技能，這些技能將在你的編程之旅中頻繁使用。

在本章中，我們學習了：

- 使用for循環遍歷列表：就像你逐一檢查購物清單上的每個物品一樣，for循環讓你能夠依次處理列表中的每個元素。想象一下，你可以用這個技能自動處理成百上千條客戶記錄或產品信息。

- 避免常見的縮進錯誤：我們瞭解了Python中縮進的重要性，以及如何避免那些會讓你的程序出錯的常見縮進問題。正確的縮進就像是良好的段落格式，讓你的代碼更容易閱讀和理解。

- 創建和操作數值列表：使用range()函數和列表解析，你現在可以輕鬆地生成和處理數字序列。這就像是有了一個能夠快速生成任意數字範圍的魔法工具，無論是創建年份列表、生成統計數據還是進行數學計算，都變得簡單高效。

這些技能為你打開了編程的新世界。現在，你可以編寫程序來自動化日常任務，比如分析銷售數據，處理學生成績或者管理庫存信息。

第二部分 項目實戰

祝賀你！你現在已經掌握了Python的基礎知識，可以開始動手構建實用的項目了。通過實際開發這些項目，你將鞏固第一部分學到的概念，並學習如何用這些知識解決實際問題。

第二部分包含四個不同類型的項目，你可以按照自己的興趣和需求選擇完成其中的任何項目或全部項目，完成這些項目的順序並不重要。下面簡要描述每個項目，幫助你決定首先要完成哪個項目。

□ 數據分析報告生成系統：在“數據分析報告生成系統”項目（第7章）中，你將學習如何在Windsurf中通過DeepSeek開發一個能夠自動處理數據並生成分析報告的系統。這個項目將教會你如何處理和分析數據，以及如何將分析結果以清晰的方式呈現出來。完成這個項目後，你將掌握數據處理和報告生成的基本技能。

□ 自動化日報生成工具：在“自動化日報生成工具”項目（第8章）中，你將在Windsurf中使用DeepSeek構建一個能夠自動收集數據並生成日常報告的工具。這個項目特別適合那些需要定期整理和彙報工作進展的人。通過這個項目，你將學習如何自動化重複性工作，提高工作效率。

□ 本地事務提醒助手：在“本地事務提醒助手”項目（第9章）中，你將在Windsurf中使用DeepSeek開發一個能夠管理日常事務的應用程序。這個項目將教會你如何設計和實現一個實用的本地應用，以及如何處理用戶輸入和數據存儲。

□ 打造你的第一個網站：在“打造你的第一個網站”項目（第10章）中，你將學習如何在Windsurf中使用DeepSeek創建一個完整的網站。這個項目將帶你瞭解網站開發的基本流程，從前端頁面設計到後端功能實現，以及對網站界面的優化。這個項目特別適合想要展示個人作品集或建立在線展示平臺的人。完成這個項目後，你將具備基本的全棧開發能力。

第7章

開發數據分析報告生成系統

在本章，我們將開發第一個實戰項目——數據分析報告生成系統。這個項目將幫助你把前面所學的編程概念轉化為實際應用，掌握數據處理和分析的核心技能。

7.1 本章學習目標

本章將使用Windsurf開發一個數據分析報告生成系統，幫助你揭開編程的神秘面紗，並初步掌握利用編程解決實際問題的能力。

(1) 理解編程的本質

- 明白編程就是將人類的需求轉化為計算機可以理解和執行的精確指令。

- 通過銷量數據分析實例，將抽象的編程概念與實際應用聯繫起來，理解編程如何解決現實問題。

(2) 瞭解程序語言的作用

- 認識到程序語言是人與計算機溝通的橋樑，Python是一種易於學習和使用的“外語”。

- 掌握利用Python進行銷量數據分析、趨勢識別和預測的基本思路，感受其超越Excel的能力。

- 掌握程序的基本知識、原理和流程。

- 學習程序設計的關鍵要素：變量、條件判斷（if）、循環（for）等，理解它們在解決問題中的作用。

- 掌握程序的基本流程：輸入數據→處理數據→輸出結果，並理解銷量分析場景中每個步驟的意義。

(3) 實踐應用

□ 通過實際操作，分析銷量數據、識別趨勢並預測未來銷量，將理論知識轉化為實際技能。

□ 掌握從數據到結果的完整流程，並體驗編程在業務決策中的強大力量。

7.2 自動化數據分析的優勢

1.提高業務決策效率

想象一下，你需要分析幾十種產品在多個區域的銷售數據，如果用Excel，可能需要複製粘貼公式、手動更新圖表，一不小心還會選錯單元格。

而用Python編程，你只需寫一段代碼，它就能自動處理所有銷售數據，在幾秒鐘內完成你在Excel中需要幾小時才能做完的工作。更棒的是，當有新數據時，你只需運行同一段代碼，而不是重複那些煩瑣的Excel操作。這就是為什麼專業數據分析師都在用編程工具——工具能讓分析變得又快又準，讓你把時間花在思考業務策略上，而不是機械地處理數據。

2.提升預測的準確性

Excel就像是一個普通的放大鏡，而Python編程像是一臺高精度顯微鏡。在Excel中，你也許只能計算簡單的平均值或畫條趨勢線，但用Python，你可以輕鬆實現複雜的時間序列分析、季節性分解、多變量回歸分析，甚至利用機器學習預測未來銷量。

這些高級分析能力讓你看到Excel無法展示的銷售模式和市場趨勢。就像醫生不會只用體溫計診斷複雜疾病一樣，業務分析師也不會僅依賴Excel做重要決策。Python提供的是一整套“銷售診斷設備”，幫你更準確地預測未來銷量，減少庫存浪費，優化生產計劃。

3.發現隱藏的業務機會

Excel是封閉的“花園”，而Python是無邊的“海洋”。在Excel中，你只能處理已經整理好的放入表格的數據。但Python可以自動從多個來源收集數據，將銷量數據與天氣、社交媒體熱度、經濟指標等外部因素結合分析，發現傳統方法難以察覺的關聯。

這就像是從只能看到表面現象，升級到能透視整個市場生態系統——你獲得了全新的視角。許多成功的業務創新和市場機會都是通過編程實現的數據挖掘和模式識別找到的。當競爭對手還在用傳統方法分析銷售數據時，掌握編程技能的你已經發現了市場中被忽視的“金礦”。

7.3 數據分析報告生成系統的功能

為了幫助用戶更高效地處理和分析海量數據，我們設計了一個自動化的數據分析報告生成系統。這個系統不僅能顯著提高工作效率，還能提供高質量、專業的分析報告，解決傳統人工分析過程中的煩瑣和低效問題。

在這個項目中，我們將創建一個強大且高效的數據分析報告生成系統。想象一下，它就像一位不知疲倦的數據分析專家，能夠智能地批量處理結構相似的銷售數據（Excel格式），並迅速生成內容豐富、格式專業的分析報告。

核心價值：本系統旨在將你從煩瑣、重複的數據整理與初步分析工作中解放出來，實現效率的飛躍——原本需要數小時甚至數天的人工操作，現在幾分鐘內即可輕鬆完成。

本系統集成了從數據接入到報告輸出的全流程自動化能力。

1. 數據處理與清洗

- 自動讀取Excel格式的銷售數據。
- 智能處理缺失值、識別並修正異常數據點，確保分析基於高質量數據。
- 多維度數據分析。
- 基礎統計：自動計算關鍵指標（如均值、中位數、標準差）。

- ❑ 績效排名：快速生成門店/產品等多維度（如收入、利潤、毛利率、客單價等）的表現排名。

- ❑ 時間序列：深入分析銷售數據隨時間（月度、季度、年度）變化的趨勢與模式。

- ❑ 成本結構：清晰展示各項成本構成及其佔比。

2.可視化圖表生成

一鍵生成專業且美觀的數據圖表，包括但不限於：

- ❑ 趨勢變化圖。

- ❑ 構成比例圖。

- ❑ 對比排名圖。

3.洞察與建議提煉

- ❑ 基於嚴謹的數據分析結果，自動提煉關鍵業務洞察。

- ❑ 初步生成具有參考價值的業務改進建議。

4.一鍵式報告輸出

- ❑ 將所有分析結果（文本、表格、圖表）無縫整合。

- ❑ 自動生成結構清晰的Markdown格式報告。

- ❑ 支持將報告輕鬆轉換為PDF或Word文檔，便於分享和歸檔。

7.4 具體實現步驟：開發數據分析報告生成系統

以下步驟將引導你學習如何與AI工具（Windsurf）交互，從Excel數據出發，逐步構建並優化數據分析報告生成系統，從而更高效、智能地處理和分析銷售數據，並生成高質量的報告。

7.4.1 第一步：導入數據，讓Windsurf理解你的Excel表格

首先，我們需要引導Windsurf讀取並理解Excel格式的銷售數據（如圖7-1所示）。這一關鍵步驟是讓AI掌握數據的結構與內容（如圖7-2所示），為後續的智能分析奠定堅實基礎。



圖7-1 引導Windsurf理解數據

在Windsurf理解表結構之後通過編寫DeepSeek提示詞讓AI自己開始分析報告並編寫代碼，為後續的智能分析奠定堅實基礎。



圖7-2 分析Excel數據後的結果界面

注意

在此過程中，Windsurf可能會提示需要安裝額外的Python依賴包。如果遇到此類彈窗，通常是因為缺少必要的庫。請單擊Accept或“同意”按鈕，讓其自動完成安裝。

7.4.2 第二步：指令下達，命令Windsurf創建自動化報告生成程序

接下來，我們將向Windsurf發出明確指令，要求它為你量身定製一個自動化報告生成程序（如圖7-3所示），清晰地告訴AI你希望它執行哪些分析、生成哪些圖表和報告內容。

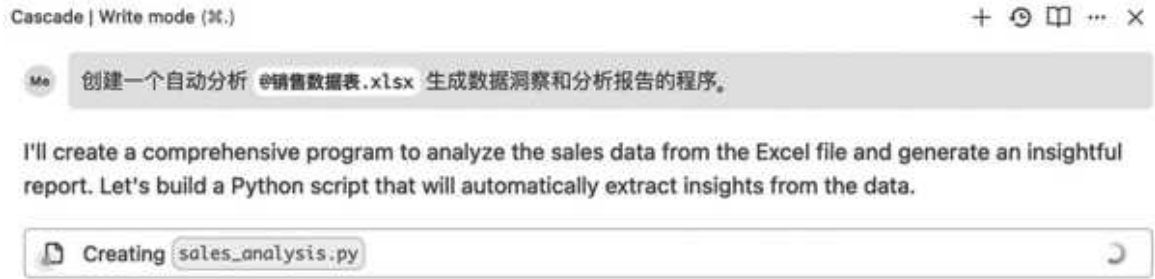


圖7-3 發出生成程序指令的界面

Windsurf接收到指令後，將開始思考並編寫相應的Python代碼。稍等片刻，它會生成程序，並直接運行以輸出初步的分析結果，如圖7-4所示。



圖7-4 DeepSeek初步的分析結果

讓我們查看DeepSeek初步生成的銷售報告，如圖7-5所示。



圖7-5 DeepSeek初步生成的銷售報告

DeepSeek初步生成的銷售報告包括產品的基本統計、產品銷售分析、相關性分析熱力圖等。下面將對DeepSeek生成銷售數據報告部分效果圖進行展示。

產品的基本統計信息如圖7-6所示。

1. 基本统计信息

本报告分析了从 2024-01-05 至 2024-03-23 的销售数据，共计 30 条销售记录，涉及 29 种产品和 29 个客户，总销售额达 740,633.00 元，总利润 110,653.49 元，平均利润率 17.53%。

指标	数值
总记录数	30
时间范围	2024-01-05 至 2024-03-23
总销售额	740633
总利润	110653.48999999999
平均利润率	17.533333333333334
产品种类数	29
客户数量	29
销售代表数量	29

圖7-6 產品的基本統計信息

產品銷售數據分析如圖7-7所示。



圖7-7 產品銷售數據分析

銷售數據相關性分析熱力圖如圖7-8所示。



圖7-8 銷售數據相關性分析熱力圖

7.4.3 第三步：測試並要求AI (Windsurf) 優化程序

生成的初步報告往往並非一步到位，可能存在格式問題、分析遺漏或不符預期之處，因此需要進一步測試並優化程序，此步至關重要。

❑ 仔細審閱：檢查Windsurf生成的代碼邏輯（如果可見）和輸出的報告內容。

❑ 發現問題：記錄錯誤、不準確或需要改進的地方。

□ 提出優化指令：再次與Windsurf交互，明確指出需要修改的內容。例如，“請將柱狀圖改為堆疊柱狀圖”“請增加毛利率的計算和排名”“報告標題字體請加大”等。這是一個交互式的精煉過程。

DeepSeek提示詞示例：

生成的 HTML 界面效果并不是很好，请你对界面进行优化，要求：1. 表格数据对齐；2. 整体界面精美。

與DeepSeek的交互過程如圖7-9所示。



圖7-9 與DeepSeek的交互過程

7.4.4 第四步：重新運程序，獲取優化後的完整報告

在Windsurf根據你的反饋完成程序代碼的優化後，指示它重新運行更新後的程序。

此時，Windsurf將執行經過改進的代碼邏輯，進行更完善的數據分析和可視化，最終生成一份更符合需求的、內容更完整的自動化分析HTML報告，如圖7-10所示。

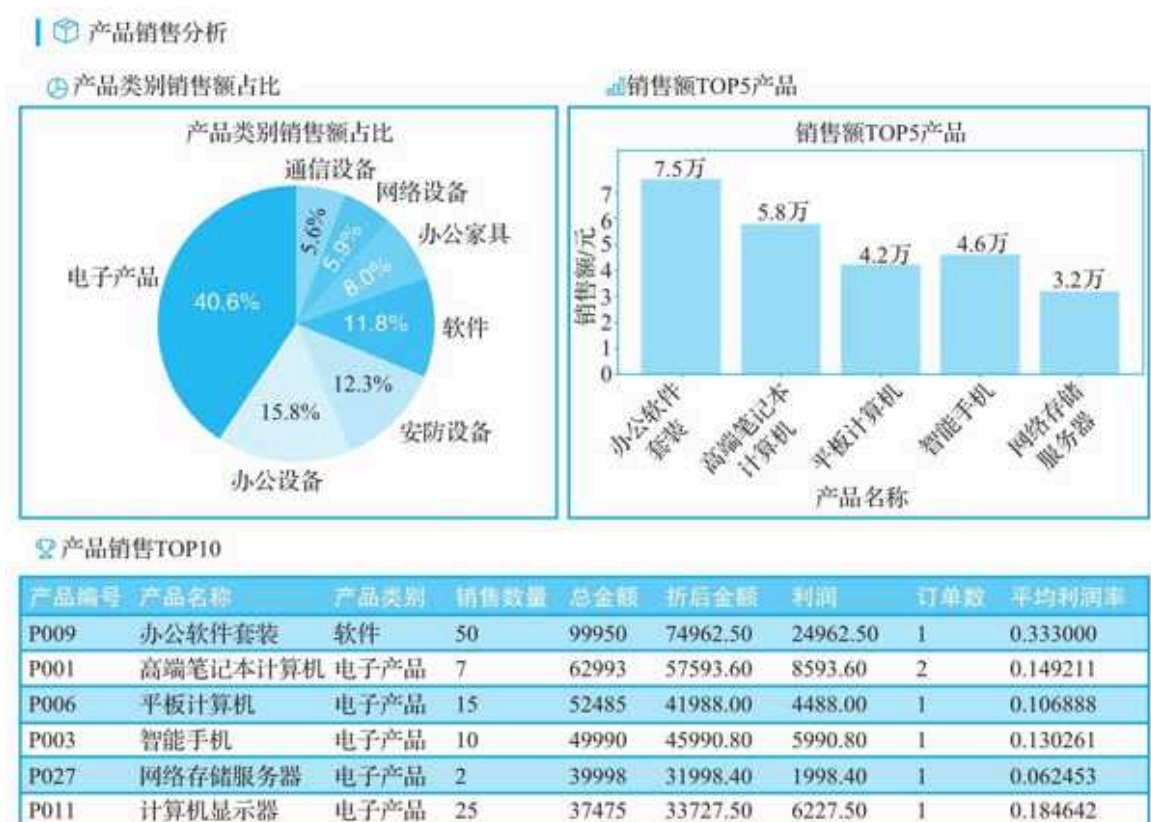


圖7-10 最終生成的HTML報告界面

可以看出，我們已經對銷售額TOP5的數據的展示效果進行了優化，現在可以很好地顯示分析數據了。

7.4.5 第五步：繼續優化生成系統的功能

如果任意上傳一份銷售數據的Excel文件，我們都想讓系統能夠對文件中的銷售數據進行分析，並生成銷售分析報告，應該如何實現呢？

讓我們繼續和Windsurf進行交互：

现在让我们对整个系统再继续进行优化，通过用户上传任意一份销售数据的 Excel 文件，直接生成这样的销售分析报告。

之後等待Windsurf為我們編寫代碼，結果如圖7-11所示。

下面我們可以直接讓Windsurf安裝相關依賴並運行這個項目，如圖7-12所示。

讓我們查看Windsurf生成的界面效果，如圖7-13所示。

可以看到，Windsurf已經生成了一個數據分析報告生成系統，讓我們上傳一份新的銷售數據Excel文件，並查看數據分析的報告效果，如圖7-14、圖7-15所示。



圖7-11 Windsurf生成的結果示例



圖7-12 Windsurf安裝相關依賴並運行這個項目



圖7-13 數據分析報告生成系統的界面效果

销售数据分析报告

生成时间: 2025-04-20 18:14:29

基本统计信息

总销售额 7441307.00 ¥	总利润 2068205.05 ¥	平均利润率 32.79%% %	销售记录数 500 条
--------------------------------	-------------------------------	------------------------------	--------------------------

本报告分析了从 **2024-01-01** 至 **2025-03-31** 的销售数据，共包含 500 条销售记录，涉及 15 种产品和 12 位客户，总销售额达 7,441,307.00 元，总利润 2,068,205.05 元，平均利润率 32.79%%。

指标	数值
总记录数	500
总销售额	7441307.00
总利润	2068205.05
平均利润率	32.79%
产品种类数	15
客户数量	12

圖7-14 報告基本統計信息

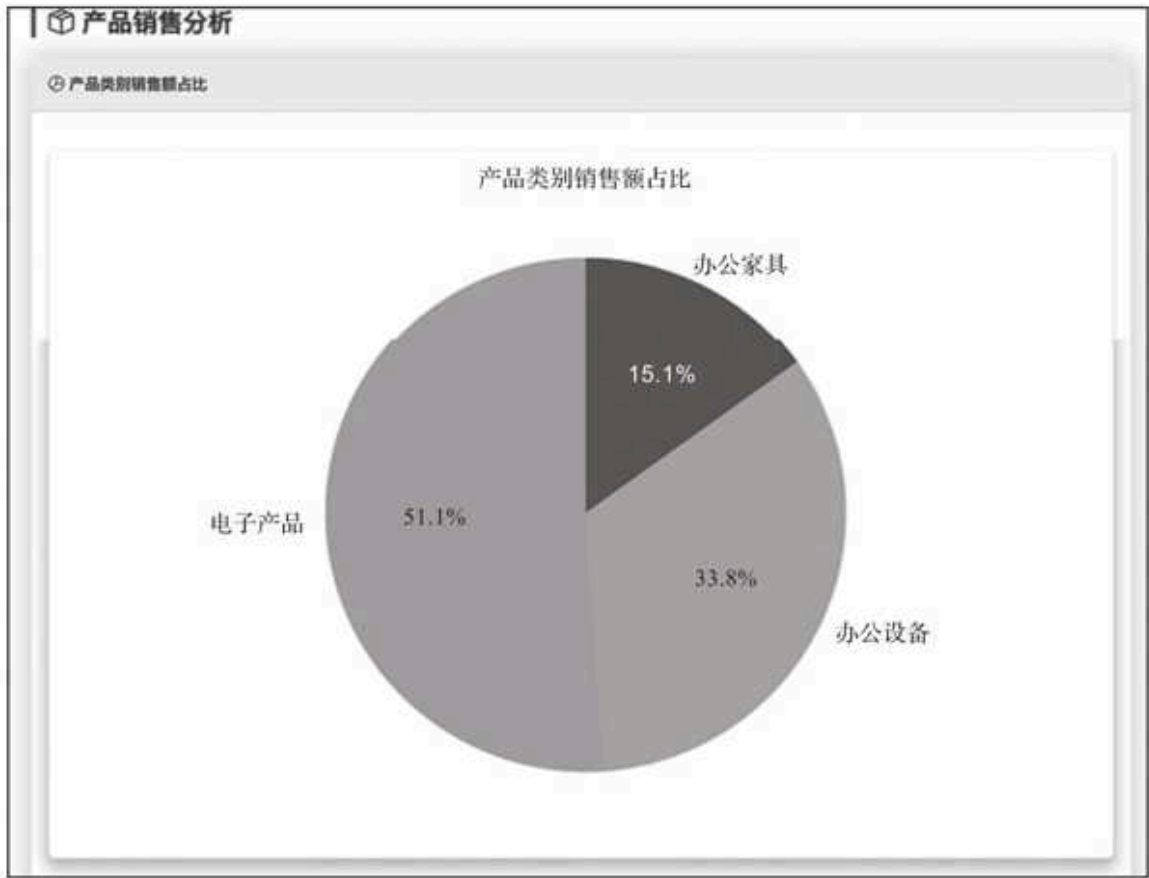


圖7-14 (續)

产品销售额TOP10

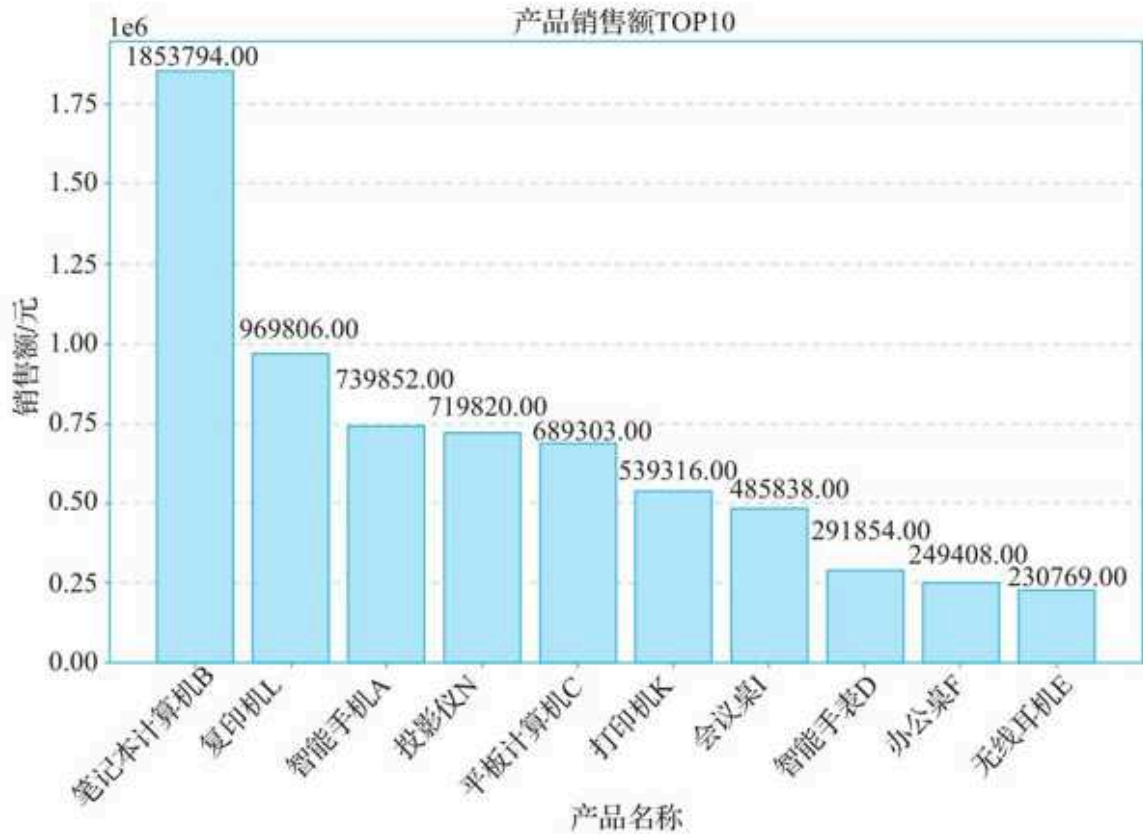


圖7-15 報告銷售數據分析

■ 产品销售TOP10明细

产品名称	总金额	销售数量	利润	平均利润率
笔记本电脑B	1853794	206	471920.21	0.276313
复印机L	969806	194	299373.69	0.339667
智能手机A	739852	148	171171.02	0.265476
投影仪N	719820	180	206395.86	0.314438
平板计算机C	689303	197	169267.13	0.271973
打印机K	539316	284	142571.46	0.294952
会议桌I	485838	162	144394.62	0.331184
智能手表D	291854	146	91566.55	0.343246
办公桌F	249408	192	66177.81	0.301112
无线耳机E	230769	231	87456.84	0.430913

■ 销售时间趋势

■ 月度销售趋势

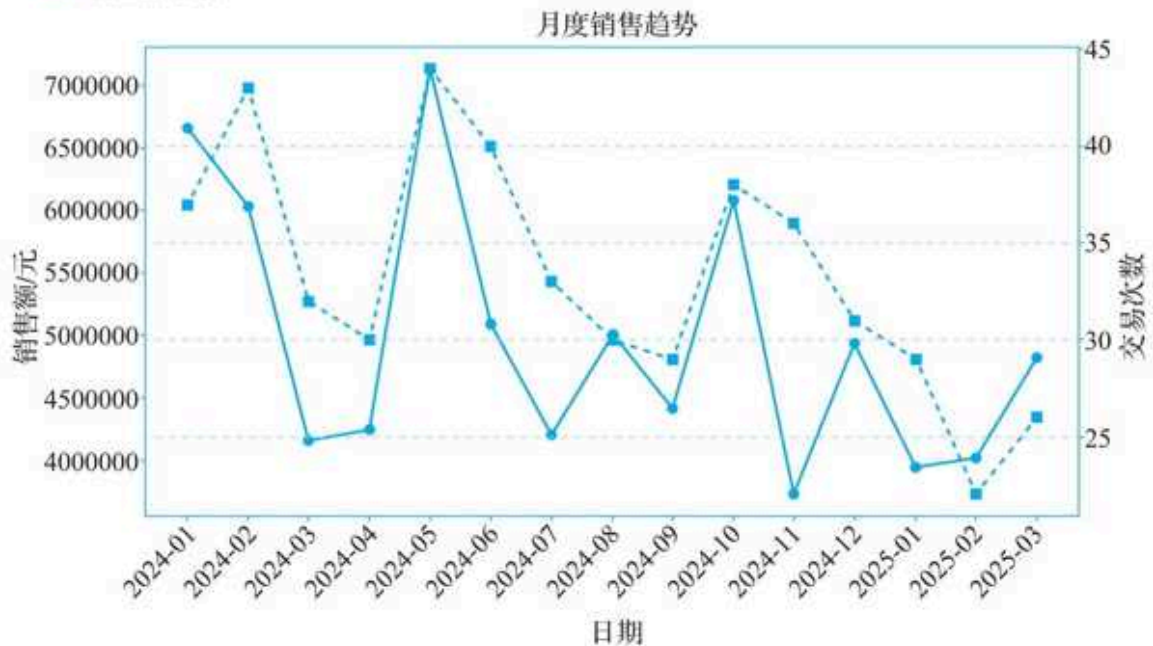


圖7-15 (續)

可以看見經過優化之後，用戶上傳任意Excel格式的銷售數據之後，系統都可以生成一份這樣的數據分析報告。

7.5 常見問題與解決思路

在使用Windsurf創建銷量數據分析工具的過程中，你可能會遇到一些問題。以下是常見問題和解決思路，可以幫助你更高效地與Windsurf協作，確保數據分析過程順利進行。

問題1：如何高效地與Windsurf中的AI助手溝通分析需求？

答：要想與Windsurf中的AI助手有效溝通，應遵循以下原則。

- ❑ **明確目標**：清晰描述你期望實現的分析功能，例如，“我想創建一個能分析與對比各門店銷售趨勢的工具”。

- ❑ **分步提問**：將複雜的任務拆解為更小的、更細化的任務，並分步驟進行提問，如先進行數據清洗，再進行可視化。

- ❑ **提供示例**：如有可能，提供少量樣例數據或你期望得到的輸出表格/圖表樣式，幫助AI精準理解。

- ❑ **迭代反饋**：對AI生成的代碼或結果給出具體、有針對性的反饋，明確指出需要修改或完善的地方。

- ❑ **主動學習**：當遇到不理解的代碼或概念時，可以直接向AI提問，例如，“請解釋一下這段用於處理缺失值的代碼邏輯”。

通過明確而具體的指令，你能更好地引導AI完成數據分析任務。

問題2：我的銷售數據格式混亂，如何讓Windsurf幫忙整理？

答：要處理雜亂的Excel文件，可以引導Windsurf按以下步驟操作。

□ 數據探查：指示AI“讀取這個Excel文件，並告訴我數據的基本情況（如列名、數據類型等）”。

□ 識別問題數據：指示AI“檢查數據中是否存在缺失值、重複記錄或明顯的異常數值（如負數銷售額）”。

□ 規範日期格式：如果日期格式不統一，指示AI“將所有日期數據轉換為‘年一月一日’的標準格式”。

□ 處理重複項：指示AI“識別並移除數據中完全重複的銷售記錄”。

□ 結構調整（可選）：若表格結構不利於分析（如寬錶轉長表），可描述期望的結構，讓AI嘗試轉換。

這些步驟將幫助你整理數據，確保其規範化，從而便於後續的分析。

問題3：如何利用Windsurf找出銷售表現好的和差的門店？

答：要進行門店績效對比分析，可以這樣向Windsurf提問。

□ 計算核心指標：指示AI“計算每個門店的總銷售額、平均銷售額和總利潤”。

□ 生成排名：指示AI“根據總銷售額對所有門店進行排名，並列出前5名和後5名”。

□ 多維度評估：詢問AI“除了銷售額，我們還可以從哪些維度（如利潤率、客單價、增長率）來評估門店表現？請計算並排

名”。

□ 同比/環比分析：若有歷史數據，要求AI“計算各門店本期銷售額相較於上期（或去年同期）的增長率”。

□ 探索關聯性（進階）：詢問AI“銷售額高的門店在地理位置、產品結構或其他方面是否存在共同特徵”。

這些分析將幫助你深入瞭解門店表現的不同維度，找出表現好的和差的門店。

問題4：如何讓Windsurf生成清晰、美觀的銷售圖表？

答：指導Windsurf創建可視化圖表時，需明確自己的需求。

□ 指定圖表類型：“為月度總銷售額創建一個折線圖，展示其變化趨勢。”

□ 確保中文顯示：“確保圖表中的所有中文標籤（標題、座標軸、圖例）都能正確顯示，避免亂碼。”

□ 組合圖表（Dashboard）：若需在一個視圖中展示多個圖表，可以要求AI“將銷售趨勢折線圖、門店銷售額條形圖和產品類別餅圖放在同一個畫布上，方便對比”。

□ 增強交互性（可選）：詢問AI“生成的圖表是否支持鼠標懸停時顯示具體數值”。

□ 定製外觀（可選）：可以向AI提出偏好，如“圖表配色方案請使用藍色或商務風格”“請增大座標軸標籤的字號”。

□ 添加說明元素：要求AI“為每張圖表添加清晰的標題，並在需要時添加圖例和數據來源說明”。

通過精確的圖表要求，你能確保生成的圖表既美觀又清晰，便於後續分析與展示。

問題5：如何利用Windsurf進行銷售預測？

答：要讓Windsurf輔助進行銷售預測，可以按以下思路進行。

□ 明確預測目標：“基於歷史銷售數據，幫我預測未來三個月的總銷售額。”

□ 提供歷史數據：確保AI使用足夠且相關的歷史數據進行建模，可詢問AI“進行有效預測至少需要多少歷史銷售數據”。

□ 選擇或諮詢模型：可以直接指定模型（如ARIMA），或詢問AI“有哪些時間序列預測模型適合我的數據，各自優缺點是什麼”。

□ 評估與調優：預測後，詢問AI“如何評估這個預測模型的準確性，有哪些方法可以嘗試提高預測精度”。

□ 結果可視化：要求AI“生成一個圖表，將歷史實際銷售額與模型的預測值（包括未來預測部分）放在一起進行對比”。

這些步驟將幫助你構建有效的銷售預測模型，並通過可視化手段展示預測結果。

問題6：如何讓Windsurf創建更易於維護和複用的分析工具？

答：為了使你創建的工具具有更好的可持續性，可以向Windsurf提出以下要求。

□ 模塊化設計：建議AI“將數據讀取、數據清洗、分析計算、圖表生成等不同功能的代碼封裝成獨立的函數或模塊”。

□ 代碼複用：詢問AI“如果某段計算邏輯（如計算利潤率）需要在多處使用，如何編寫能讓代碼更簡潔且易於維護”。

□ 添加註釋：要求AI“在關鍵代碼段或複雜邏輯旁邊添加中文註釋，解釋其功能和目的”。

□ 錯誤處理：詢問AI“如何讓程序在遇到異常數據（如文件不存在、格式錯誤）時不會直接崩潰，而是能給出提示或進行容錯處理”。

□ 參數化配置：建議AI“能否將一些經常需要調整的參數（如文件路徑、分析週期、圖表顏色）提取出來，放在程序的開頭或配置文件中，以便修改”。

□ 增加測試（進階）：詢問AI“有沒有簡單的方法可以驗證程序的各個部分是否按預期工作”。

通過這些優化，你可以使分析工具更易於維護和複用，確保代碼的長期可用性和可擴展性。

這些常見問題和解決思路將幫助你更高效地使用Windsurf創建分析工具，解決可能遇到的難題，並確保你能夠充分利用AI的強大能力來進行數據分析與報告生成。

7.6 結語

本章深入探討了如何在Windsurf平臺中使用DeepSeek模型來構建數據分析報告生成系統。通過項目實踐，我們系統地掌握了從項目規劃到具體實現的完整流程。

首先，我們明確了項目的背景和目標，理解了在Windsurf中使用DeepSeek的重要性，這為後續的開發工作奠定了堅實的技術基礎。接著，我們深入學習了自動化數據分析的核心優勢，包括提高效率、確保一致性以及降低人工錯誤等。

在具體功能實現上，我們重點掌握了三個關鍵模塊。

- 數據處理與清洗：學會了數據預處理的基本技術。
- 可視化圖表生成：掌握了自動生成分析圖表的方法。
- 一鍵式報告輸出：實現了自動化報告生成流程。

通過學習本章，我們不僅理解了模塊化設計的重要性，還掌握瞭如何構建一個可維護、可擴展的系統。在實踐過程中，我們也積累了豐富的問題解決經驗，學會了如何處理常見的異常情況。

這些知識和經驗為我們後續開發更復雜的自動化工具打下了良好基礎。在下一章中，我們將學習如何開發自動化工具，這將幫助我們進一步提升軟件開發能力，構建更多、更實用的自動化解決方案。

第8章

開發自動化日報生成工具

在當今快節奏的工作環境中，重複性工作不僅耗時，還容易因人為操作而出錯。本章將帶領你開發一個自動化日報生成工具，這個工具能夠自動收集、整理數據，並生成規範的日常工作報告。

通過這個項目，你將學習如何利用Windsurf平臺和DeepSeek模型來構建實用的自動化工具。我們將從實際工作場景出發，探索如何將煩瑣的日報編寫過程轉化為簡單的一鍵操作。這不僅能大幅提升工作效率，還能確保報告的質量和一致性。

在項目過程中，你將掌握：

- ❑ 如何設計直觀的自動化工具界面。
- ❑ 如何處理和組織多源數據。
- ❑ 如何生成格式統一的報告文檔。
- ❑ 如何優化工具的使用體驗。

這個項目特別適合那些需要定期整理工作進展、生成報告的人員。無論你是項目經理、團隊負責人，還是需要定期彙報工作

的職場人士，這個自動化工具都將極大地提升你的工作效率。

讓我們開始這個實用的自動化之旅，一起探索如何讓AI技術為日常工作提供切實的幫助。通過本章的學習，你將能夠構建自己的自動化助手，讓煩瑣的日常工作變得輕鬆高效。

8.1 本章學習目標

本章將使用Windsurf開發一個數據分析報告生成系統，通過這個案例將幫助你揭開編程的神秘面紗，並初步掌握利用編程解決實際問題的能力。

本章旨在幫助你掌握自動化個人日報工具的開發要點，並通過與AI協作完成實踐。

(1) 理解核心概念

- ❑ 明確個人日報工具的基本用途與價值。
- ❑ 識別一個有效的日報工具應具備的核心功能模塊。
- ❑ 認識AI編程助手（如Windsurf）在此類自動化工具開發中扮演的角色與優勢。

(2) 掌握開發技能

- ❑ 學習如何利用Windsurf進行初步的系統架構構思。
- ❑ 掌握引導AI助手生成所需功能代碼的方法。
- ❑ 實踐並提升與AI助手進行高效、準確對話的技巧。

(3) 積累實踐經驗

- ❑ 熟悉自動化工具開發的基本流程：從需求到實現再到優化。
- ❑ 學會如何清晰、具體地向AI助手描述需求。

□ 通過實際項目，積累與AI助手協作開發的寶貴經驗。

8.2 瞭解自動化日報生成工具

1. 什麼是自動化日報生成工具

理解我們要構建的工具是什麼，以及它能做什麼。

簡單來說，這是一個旨在幫助個人高效生成與管理工作日報的軟件應用。

它的核心價值在於：能夠部分自動記錄你的工作活動（或極大簡化記錄過程），並通過一鍵操作快速生成格式規範、內容完整的日報文檔。最終目標是讓你從煩瑣重複的日報編寫工作中解放出來，將寶貴的時間和精力更專注於實際工作本身。

2. 自動化日報生成工具的優勢

我們旨在構建的自動化日報生成工具將具備以下核心優勢：

1) 效率提升：告別手動編寫。

□ 支持一鍵生成格式化的日報初稿。

□ 能夠自動記錄或導入部分工作內容（如會議、任務）。

□ 顯著節省每日手動整理和編寫報告所需的時間。

2) 智能處理：內容自動優化。

□ 根據預設規則或AI的理解進行分類整理。

□ 自動應用統一的格式調整，確保報告專業性。

- 嘗試關鍵內容提取，突出當日工作重點。

3) 數據管理：回顧與追蹤。

- 方便進行歷史日報的查詢與回顧。

- 對記錄的工作內容進行簡單的統計分析。

- 支持個人進度的概覽與追蹤。

3. 自動化日報生成工具的使用場景

這款自動化日報生成工具能夠有效支持以下多種場景：

- 日常工作彙報：快速完成每日或每週需提交給上級或團隊的工作小結。

- 個人工作覆盤：作為定期進行個人工作總結和反思的素材來源。

- 項目進度跟蹤：方便地記錄和回顧特定項目每天的進展情況。

- 時間管理分析：通過回顧歷史記錄，分析個人時間分配模式，優化工作效率。

- 工作計劃制定：基於過往記錄和當前狀態，為後續工作計劃的制定提供參考。

8.3 自動化日報生成工具的核心功能

在開始詳細介紹各個功能模塊之前，我們需要明確：一個優秀的自動化日報生成工具不僅要能完成基本的日報生成任務，還要能夠真正提升用戶的工作效率，幫助用戶更好地管理和回顧自己的工作內容。基於這個理念，我們將功能體系分為三個層次：基礎功能層、智能功能層和個人管理功能層。

8.3.1 基礎功能層

這一層的功能是工具的根基，直接關係到用戶的日常使用體驗。

(1) 內容錄入

- ❑ 提供便捷的手動記錄工作條目入口。
- ❑ 支持快捷鍵操作和即時保存機制。
- ❑ 集成簡單的富文本編輯功能。

(2) 一鍵生成

- ❑ 根據當日記錄，快速生成格式化的日報文本。
- ❑ 支持多種輸出格式（純文本、HTML、Markdown等）。
- ❑ 自動保存生成歷史，支持版本回溯。

(3) 模板定製

- 允許用戶自定義日報的固定格式、抬頭、結尾等內容。
- 提供可視化的模板編輯體驗。
- 支持多模板管理和切換。

(4) 預覽編輯

- 在最終生成前，提供即時預覽功能。
- 支持在預覽界面直接進行內容調整。
- 自動檢查格式規範。

8.3.2 智能功能層

這一層利用智能化技術，為用戶提供更高效率的輔助功能。

(1) 智能分類

- 自動對錄入的工作內容進行分類或打標籤。
- 支持按項目、任務類型等多維度分類。
- 提供自定義分類規則的能力。

(2) 時間估算

- 根據記錄內容或特定標記自動估算時間花費。
- 基於歷史數據持續優化估算準確度。
- 生成工作時間分佈統計。

(3) 重點提取

- ❑ 智能識別並高亮顯示當日的關鍵工作成果或要點。
- ❑ 自動生成工作內容的關鍵要點概述。
- ❑ 提供重要程度的智能排序。

(4) 格式美化

- ❑ 自動調整排版、對齊等，確保報告的專業外觀。
- ❑ 提供多種預設主題樣式。
- ❑ 確保在不同設備上的顯示效果一致。

8.3.3 個人管理功能層

這一層著眼於長期使用價值，幫助用戶更好地管理和利用日報數據。

(1) 歷史回顧

- ❑ 提供便捷的歷史日報查詢和瀏覽功能。
- ❑ 支持多維度搜索和篩選。
- ❑ 可視化展示工作歷程和重要里程碑。

(2) 工作統計

- ❑ 按項目、時間等維度統計工作量。
- ❑ 生成直觀的數據可視化圖表。

- 提供工作效率趨勢分析。

(3) 進度追蹤

- 輔助用戶回顧個人工作進展。

- 設置和追蹤工作目標。

- 及時提醒待辦事項和重要節點。

(4) 數據導出

- 支持將日報數據導出為多種格式。

- 提供數據備份和恢復功能。

- 確保數據安全和隱私保護。

8.3.4 功能使用建議

為了讓用戶能夠更好地利用這些功能，我們建議：

(1) 循序漸進

- 先熟悉基礎功能，確保日常使用順暢。

- 逐步探索智能功能，提升使用效率。

- 根據個人需求選擇性使用管理功能。

(2) 持續積累

- 養成每日記錄的習慣。

□ 定期回顧和總結。

□ 利用統計分析優化工作方式。

通過這些功能的合理運用，用戶可以構建一個高效的個人工作管理系統，真正實現工作效率的提升。

8.4 具體實現步驟：構建你的自動化日報生成工具

以下步驟將引導你如何與AI工具（Windsurf）交互，從簡單的需求開始設計基礎界面，逐步構建並優化最終生成一個專屬的自動化日報生成工具。

8.4.1 第一步：設計基礎界面

在對話框中輸入初始需求：

我想开发一个个人自动化日报生成工具，首先需要有一个简单的操作界面，包含工作内容记录和日报生成功能，请帮我设计并生成代码

可以看見Windsurf很快生成了初始工具構建示例，如圖8-1所示。



圖8-1 Windsurf中初始工具構建示例

我們單擊運行看看這裡構建的個人自動化日報生成工具的初始效果，如圖8-2所示。

个人自动化日报生成工具

工作记录 生成日报

添加工作记录

任务名称:

任务描述:

花费时间(小时):

完成状态:

已完成

添加记录

今日工作记录

暂无工作记录

清空记录

圖8-2 個人自動化日報生成工具的初始效果

我們可以看見通過這段提示詞，Windsurf已經對基本界面框架已經實現，並且已經包含了個人自動化日報生成工具的核心功能。

□ 工作記錄管理：通過直觀的表單界面，用戶可以輕鬆添加日常工作任務，包括任務名稱、詳細描述、花費時間以及完成狀態。

□ 時間追蹤：用戶可以精確記錄每項任務所花費的時間（以小時為單位），便於後續分析工作效率和時間分配。

□ 狀態標記：可以標記任務是否已完成，幫助用戶清晰瞭解當日工作進度。

□ 日報自動生成：系統會根據用戶輸入的工作記錄，自動整合生成規範化的日報，省去了手動彙總的煩瑣過程。

□ 數據管理：用戶可以查看當日已添加的所有工作記錄，也可以通過“清空記錄”功能重新開始。

現在我們對核心功能模塊進行測試，查看DeepSeek是否已經幫我們實現了這些功能。讓我們嘗試輸入一些工作記錄（如圖8-3所示），並進行查看添加的工作記錄（如圖8-4所示），之後再單擊“添加記錄”按鈕，檢查我們上傳的記錄是否已經添加成功。

此网页显示
工作记录已添加!

确定

工作记录 生成日报

添加工作记录

任务名称:

阅读

任务描述:

reading book

花费时间(小时):

1

完成状态:

已完成

添加记录

今日工作记录

暂无工作记录

清空记录

圖8-3 個人日報基本框架



圖8-4 已添加的今日工作記錄

可以看見我們上傳的日報信息已經被添加到今日工作記錄中了，現在讓我們切換到生成日報界面，選擇標準模板的生成樣式，單擊“生成日報”按鈕查看效果（如圖8-5所示）。

个人自动化日报生成工具

工作记录生成日报

生成日报

日期:

2025/04/09 🗑

日报模板:

标准模板 ▼

生成日报

日报预览

复制下载

日期: 2025年4月9日 星期三
总工作时间: 6 小时

【今日完成工作】

1. 阅读: reading book (1小时)
2. 工作: work (4小时)

【进行中工作】

1. 锻炼: take exercise (1小时)

【待处理工作】

圖8-5 生成日報的效果

可以看見生成的日報內容已經符合我們的需求，之後可以單擊“複製”或者“下載”按鈕對今日生成日報的內容進行導出，或者在日報模板中選擇別的模板，按照自己喜歡的樣式進行導出。

8.4.2 第二步：完成界面優化

現在的界面比較簡單，所以我們現在要優化一下界面佈局和用戶界面。

我們可以直接在對話框中向Windsurf提出：“請優化界面佈局和用戶界面設計，要求界面精美，樣式美觀。”

讓我們等待Windsurf編寫代碼，完成之後單擊運行，查看已經完成優化的界面，如圖8-6所示。

讓我們切換到生成日報的界面使用自定義模式查看效果，如圖8-7所示。

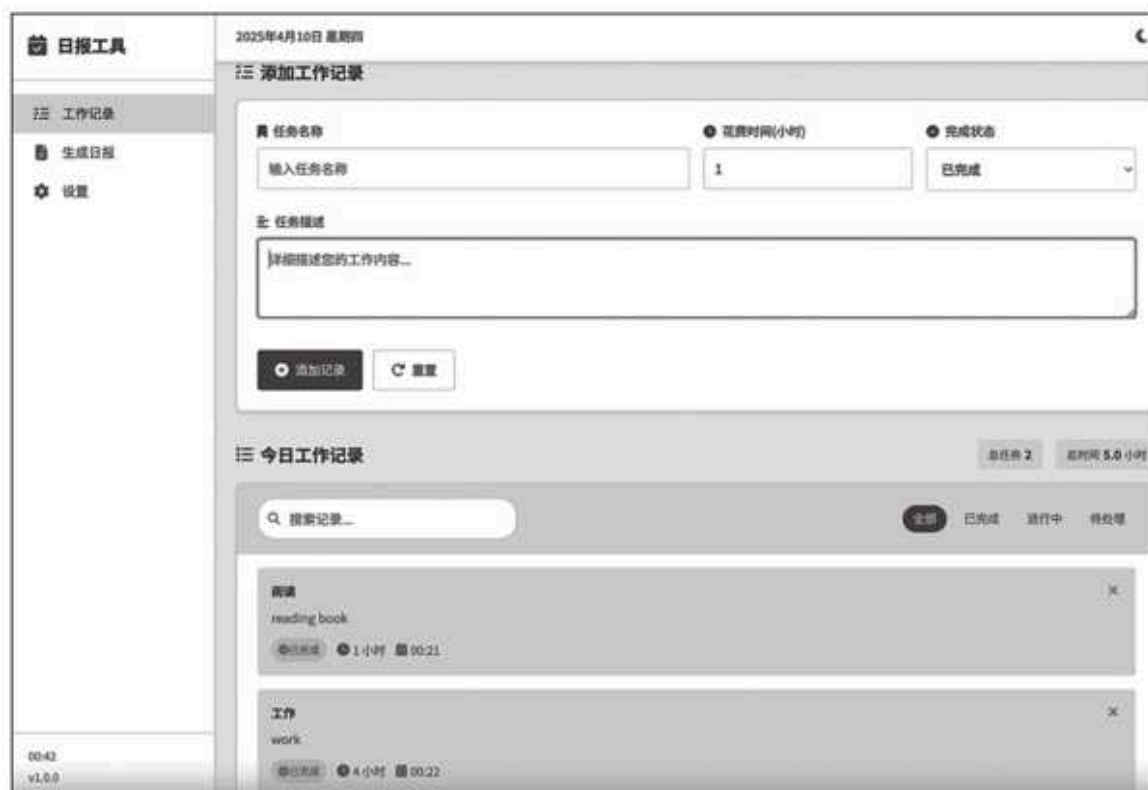


圖8-6 完成優化之後的界面



圖8-7 生成日報的界面

讓我們切換到設置界面查看個性化設置的功能，如圖8-8所示。



圖8-8 日報工具的個性化設置界面

可以看到我們的個人日報工具已經有了一些個性化的設置。下面對目前已經實現的功能進行總結。

(1) 工作記錄功能

- 添加工作記錄：記錄任務名稱、詳細描述、花費時間和完成狀態。

- 記錄管理：支持刪除單條記錄或清空所有記錄。

- 搜索與過濾：可按關鍵詞搜索或按完成狀態（已完成/進行中/待處理）過濾記錄。

- 統計信息：自動計算總任務數和總工作時間。

(2) 日報生成功能

- ❑ 多種模板：支持四種日報模板（標準、詳細、簡潔、自定義）。

- ❑ 自定義選項：可選擇包含的內容部分（工作概覽、已完成工作、進行中工作等）。

- ❑ 格式選擇：支持多種輸出格式。

- ❑ 日期選擇：可生成任意日期的工作日報。

- ❑ 一鍵操作：支持複製和下載生成的日報。

(3) 數據管理功能

- ❑ 本地存儲：所有數據保存在瀏覽器的localStorage中，關閉瀏覽器數據不會丟失。

- ❑ 數據導出：可將所有工作記錄導出為JSON文件備份。

- ❑ 數據導入：支持從之前導出的文件中恢復數據。

- ❑ 數據清理：可清空所有歷史數據。

(4) 界面與用戶體驗功能

1) 現代化界面。

- ❑ 側邊欄導航：直觀的功能區分和切換。

- ❑ 卡片式設計：清晰的內容區塊劃分。

- ❑ 圖標輔助：使用圖標增強視覺識別度。

2) 個性化設置。

- ❑ 主題切換：支持淺色/深色主題，可手動切換或跟隨系統設置。

- ❑ 個人信息：可設置姓名和部門信息，自動添加到生成的日報中。

3) 用戶體驗優化。

- ❑ 通知系統：操作後提供即時反饋。

- ❑ 時間顯示：即時顯示當前日期和時間。

- ❑ 響應式設計：適配不同屏幕尺寸的設備。

後面我們可以通過我們的需求對界面進行進一步的設計，和功能的拓展。

8.4.3 實用開發技巧：與AI高效協作

1) 如何更精準地向AI描述需求？

- ❑ 明確場景：清晰說明工具是為個人日常使用，而非團隊或企業級應用。

- ❑ 說明習慣：描述你的典型操作流程，例如，“我習慣先快速記錄要點，晚上再統一整理生成日報”。

- ❑ 描繪體驗：表達你對最終使用感受的期望，例如，“希望界面極簡，錄入信息時干擾最少”“按鈕要大，容易單擊”。

2) 如何與AI助手進行高效對話？

- ❑ 先主後次：首先確保最核心的功能（如記錄、生成）能夠基本工作。

□ 逐步迭代：一次專注於添加或優化一兩個相關聯的功能點，避免一次性提出過於複雜的需求。

□ 及時反饋：運行AI生成的代碼，立即測試。發現問題或有新想法時，馬上向AI反饋並要求調整或修正。

8.5 常見問題與解決思路

在開發和使用過程中，你可能會遇到一些常見的使用體驗和開發相關的問題。以下是詳細的問題解析和解決思路，幫助你更好地解決這些問題，並通過Windsurf和DeepSeek AI助手獲得支持。

問題1：如何改善操作不便的問題？

答：優化操作流程可以從以下幾個方面著手。

- ❑ 明確痛點：記錄完成特定任務需要的具體步驟數，識別煩瑣環節。

- ❑ 與AI助手溝通：可以這樣向Windsurf描述，“我發現完成X任務需要單擊 n 次，能否幫我優化這個流程？”。

- ❑ 示例對話：“在數據導入功能中，需要經過“選擇文件”→“預覽”→“映射字段”→“確認導入”四個步驟，感覺比較煩瑣。”

- ❑ Windsurf：“我們可以通過以下方式簡化，①添加文件拖拽功能，②實現字段自動映射，③記住上次的映射配置，下面我將開始進行優化。”

- ❑ 驗證改進：對比優化前後的操作步驟，確保真正提升了效率。

問題2：如何處理界面過於複雜的問題？

答：界面優化可以遵循以下原則。

□ 與AI協作優化：向DeepSeek描述，“請幫我分析當前界面的問題，並給出優化建議。”

□ 示例對話：“我的數據分析界面包含太多按鈕和選項，用戶經常找不到需要的功能。”

□ DeepSeek：“建議採用以下優化方案，①創建快速操作欄，放置最常用的5個功能，②使用下拉菜單組織次要功能，③添加搜索功能幫助定位特定操作。”

問題3：如何解決系統響應慢的問題？

答：性能優化可以從多個層面進行。

□ AI輔助診斷：可以向Windsurf請求“請分析這段代碼的性能瓶頸。”

□ 示例對話：“這個數據處理函數執行很慢，特別是處理大量數據時。”

□ Windsurf：“讓我們逐步分析和優化，①發現當前循環中存在重複計算，②建議添加緩存機制，③可以使用並行處理。”

問題4：遇到不理解的代碼該如何處理？

答：代碼理解可以通過以下方式提升。

□ AI代碼解釋：向DeepSeek提問“請解釋這段代碼的功能和實現邏輯。”

□ 示例對話：用戶[選中代碼片段]“這段數據處理的代碼邏輯比較複雜，能否解釋一下？”

□ DeepSeek：“這段代碼主要實現了以下功能，①進行數據清洗.....，②使用聚合函數.....，③進行結果排序.....”。

問題5：如何完善功能不足的問題？

答：功能完善可以按照以下步驟進行。

□ 需求溝通：向Windsurf描述“我需要添加X功能，主要用於解決Y問題。”

□ 示例對話：“我需要添加數據導出功能，支持多種格式。”

□ Windsurf：“我可以幫你實現以下功能，①Excel導出支持，②CSV格式轉換，③JSON數據導出”。

問題6：如何改善使用體驗不順暢的問題？

答：交互優化可以從以下方面入手：

□ AI交互建議：向DeepSeek請教“如何改善用戶操作體驗？”。

□ 示例對話：“用戶反饋操作流程不夠直觀，經常需要查看幫助文檔。”

□ DeepSeek：“建議添加以下改進，①實現引導式操作提示，②添加懸停提示信息，③提供操作預覽功能”。

問題7：如何高效利用AI助手進行開發？

答：與AI助手的高效協作可以遵循以下原則。

□ 清晰的需求描述：①提供具體的場景和目標；②附上相關的代碼或截圖；③說明期望的結果。

□ 示例對話：“我需要實現數據可視化功能，主要展示銷售趨勢。”

□ Windsurf：“好的，讓我們逐步實現，①首先確認數據格式，②選擇合適的圖表類型，③實現交互功能，請提供一些示例數據，我們開始實現。”

□ 迭代優化：①對AI的回答進行驗證；②提供具體的修改建議逐步完善功能實現。

通過與Windsurf和DeepSeek的有效協作，你可以更快速地解決開發中遇到的問題，提升開發效率和產品質量。記住要善於利用AI助手的優勢，通過清晰的溝通獲得最佳的支持。

8.6 結語

本章深入探討了如何利用AI編程助手開發自動化日報生成工具，這是一個既實用又富有挑戰性的項目。通過本章的學習，我們掌握了以下幾個關鍵概念：

(1) 核心能力

- 掌握了使用AI編程助手從零開始開發的完整流程。
- 學會了如何將複雜需求拆解為可實現的小步驟。
- 理解了個人工具開發的基本架構和設計思路。

(2) 協作技巧

- 學會了與AI助手進行清晰有效的需求溝通。
- 掌握了迭代開發中的反饋和優化方法。
- 理解了如何引導AI生成更符合預期的代碼。

(3) 設計思維

- 理解了個人工具設計中的簡潔性原則。
- 掌握了用戶體驗優化的基本方法。
- 學會了如何平衡功能性和易用性。

在設計和開發個人工具時，需要特別注意以下原則：

(1) 功能實用為王

- 始終關注核心需求的解決。
- 避免過度設計和冗餘功能。
- 確保每個功能都有其實際價值。

(2) 操作簡單直觀

- 保持界面設計的清晰簡潔。
- 減少用戶操作的認知負擔。
- 提供直觀的功能引導。

(3) 數據安全保障

- 實現可靠的本地數據存儲。
- 注重個人信息的保護。
- 提供必要的數據備份功能。
- 持續迭代改進。

通過這些原則的指導，我們能夠開發出真正滿足個人需求、易於使用且可靠的自動化工具。在實際應用中，要始終記住工具是服務於使用者的，功能再多也不如一個真正解決問題的核心功能來得重要。

在下一章中，我們將探討如何將這些開發經驗用到本地事務提醒助手中，進一步提升我們的工具開發能力。這些知識和技能將幫助我們在未來的個人效率提升之路上走得更遠。

第9章

開發本地事務提醒助手

在掌握了數據分析報告生成系統和自動化日報生成工具的開發經驗後，本章將開發一個更貼近日常使用的應用——本地事務提醒助手。這個項目將幫助用戶管理和追蹤各種日常事務，從待辦事項到重要提醒，讓生活和工作更有條理。

通過Windsurf編輯器和DeepSeek的協助，我們將把前面學到的Python編程技能運用到實際應用開發中。這個項目不僅能鞏固已學的知識，還將引入新的編程概念和設計模式，幫助你進一步提升軟件開發能力。

在本章中，你將學習：

- ❑ 如何設計和實現本地數據存儲。
- ❑ 如何構建用戶友好的界面。
- ❑ 如何實現事務管理的核心邏輯。
- ❑ 如何添加自動化提醒功能。

這個項目特別適合想要提升實戰經驗的開發者。通過開發這個實用工具，你將深入理解桌面應用開發的各個環節，從需求分

析到最終部署的完整流程。

讓我們開始構建這個實用的本地事務處理助手吧。記住，當你遇到技術難題時，可以隨時向DeepSeek尋求幫助，它將為你提供專業的指導和解決方案。

9.1 本章學習目標

本項目旨在提供簡潔高效的提醒管理系統，幫助用戶創建、編輯、刪除和查看各類提醒事項，避免遺忘重要事務，並需要完成以下目標：

- ❑ 實現本地化存儲：所有數據存儲在用戶本地，確保用戶隱私和數據安全，無須依賴雲服務。

- ❑ 打造現代化用戶界面：採用卡片式佈局、精心設計的色彩方案和交互元素，提供愉悅的視覺體驗。

- ❑ 支持多樣化的提醒功能：包括單次提醒和重複提醒（每天、每週、每月或自定義週期），滿足不同場景需求。

- ❑ 提供分類管理：允許用戶對提醒進行分類，便於組織和查找。

- ❑ 靈活的篩選和搜索：支持按類別、優先級、完成狀態等條件篩選，以及基於關鍵詞的搜索功能。

關於用戶體驗設計需要達到的目標：

- ❑ 現代化界面：採用扁平化設計、圓角元素和精心調配的色彩方案，營造簡潔現代的視覺效果。

- ❑ 交互反饋：按鈕懸停效果、表單焦點狀態變化等交互設計，提升用戶操作體驗。

- ❑ 直觀的表單設計：清晰的表單佈局和輸入控件，降低用戶操作難度。

□ 響應式佈局：合理的空間分配和組件大小，確保在不同屏幕尺寸下的良好顯示效果。

□ 視覺層次：通過卡片式佈局、標題樣式和顏色對比，創建清晰的視覺層次，引導用戶的注意力。

9.2 瞭解本地事務提醒助手

1. 什麼是本地事務提醒助手

在動手之前，先清晰理解我們要構建的應用是什麼，它能做什麼，以及有哪些核心特點。本地事務提醒助手是一個安裝並運行在你個人計算機上的桌面應用程序。它區別於雲端服務或網頁應用，主要特點包括：

- ❑ 獨立運行：核心功能無須連接互聯網，所有提醒數據都安全地存儲在本地設備上。

- ❑ 輕量高效：通常設計為啟動迅速，佔用系統資源（CPU、內存）較少。

- ❑ 簡單易用：追求界面簡潔明瞭，用戶操作直觀，上手門檻低。

- ❑ 可靠安全：確保提醒準時觸發，不易遺漏；本地數據存儲相對更易掌控隱私。

它的核心價值在於：幫助用戶有效管理個人日程、待辦事項和重要日期，通過及時的、個性化的提醒功能，確保用戶不會錯過任何關鍵的時間節點或任務。

2. 本地事務提醒助手的優勢

我們即將構建的“本地事務提醒助手”具有以下顯著優勢：

- (1) 提高工作與生活效率

- ❑ 告別遺忘：確保不再錯過重要的會議、截止日期或個人事務。

- ❑ 智能管理：可支持對任務進行簡單的分類、設置優先級和循環提醒。

- ❑ 省時便捷：相較於手動翻閱日曆或便籤，極大減少查找和記錄待辦事項的時間。

(2) 降低使用與獲取成本

- ❑ 無須付費：自己開發，避免購買可能功能冗餘或需要訂閱的商業軟件。

- ❑ 量身定製：完全根據你個人的具體需求和使用習慣來設計功能。

- ❑ 長期價值：一次開發投入，可持續免費使用，長期來看極具成本效益。

(3) 創造個人成長與潛在機會

- ❑ 經驗積累：在實踐中掌握使用AI輔助開發本地應用或腳本的經驗。

- ❑ 技能拓展：可能啟發你為自己或他人開發更多個性化的實用小工具。

- ❑ 能力提升：增強個人的技術應用能力和解決問題的能力，提升職業競爭力。

3.本地事務提醒助手的使用場景

這款本地事務提醒助手可以廣泛應用於個人生活的方方面面：

(1) 個人健康管理

- 定時提醒吃藥、喝水或進行運動。
- 設置定期體檢或複查的提醒。
- 輔助建立規律的作息時間表。

(2) 工作效率提升

- 重要會議或電話前提醒。
- 追蹤項目里程碑或截止日期。
- 提醒進行重要的工作文件定期備份。
- 日報/週報等週期性任務的提交提醒。

(3) 生活事務管理

- 水電煤網費、信用卡還款、房租等定期繳費提醒。
- 生日、紀念日等重要日期的提醒。
- 家庭日常事務提醒，如澆花、遛狗、取快遞等。

(4) 學習計劃執行

- 課程上課或在線講座開始前提醒。
- 作業、論文提交截止日期提醒。

□ 複習計劃中各階段性目標的提醒。

9.3 本地事務提醒助手的核心功能

在現代的工作和生活中，合理的時間管理和任務提醒可以大大提高效率。為了幫助用戶更好地管理日常事務，本地事務提醒助手提供了一些強大的功能。接下來，我們將介紹本地事務提醒助手中的核心功能。

9.3.1 提醒創建與管理

(1) 提醒創建

本地事務提醒助手允許用戶創建詳細的提醒事項，每個提醒可包含以下信息：

- 標題：簡明扼要地描述提醒內容。
- 詳細描述：提供關於提醒的更多背景信息和細節。
- 日期和時間：設置提醒的具體時間點。
- 優先級：可設置為高、中、低三個等級，以便區分任務重要性。
- 類別：將提醒歸類到用戶自定義的類別中。
- 完成狀態：標記提醒是否已完成。

創建提醒時，系統提供了直觀的表單界面，用戶可以輕鬆填寫所有必要信息。對於重複性提醒，用戶可以在創建時直接設置重複規則，無須多次創建相似的提醒。

(2) 提醒編輯

用戶可以隨時編輯已創建的提醒，修改包括：

- 更新提醒的任何屬性（標題、描述、時間等）。
- 調整提醒的優先級或類別。
- 更改提醒的重複規則。
- 添加或刪除提醒通知時間。

編輯界面與創建界面保持一致，確保用戶體驗的連貫性。系統會保留提醒的修改歷史，便於追蹤變更。

(3) 提醒刪除

用戶可以刪除不再需要的提醒。刪除操作支持：

- 單個提醒刪除。
- 批量刪除選定的多個提醒。
- 刪除已完成的所有提醒。

系統會在執行刪除操作前請求用戶確認，防止意外刪除重要信息。

9.3.2 提醒通知系統

(1) 多層次通知時間

本地事務提醒助手支持為每個提醒設置多個通知時間點：

- 事件發生時通知。
- 提前自定義時間通知（如提前15分鐘、1小時、1天等）。
- 支持添加多個提前通知時間點。

用戶可以根據提醒的重要性和準備所需時間，靈活設置合適的通知時間點。

（2）通知方式

系統提供多種通知方式，確保用戶不會錯過重要提醒：

- 桌面彈窗通知：在屏幕上顯示提醒內容。
- 聲音提醒：播放可選的提示音。
- 視覺提醒：任務欄圖標變化或閃爍。

通知內容包含提醒的關鍵信息，如標題、時間和優先級，用戶可以直接從通知界面標記提醒為已完成或延遲提醒。

（3）通知管理

用戶可以管理通知的行為和顯示方式：

- 設置通知的持續時間。
- 配置通知的顯示位置。
- 選擇是否在鎖屏狀態下顯示通知。
- 暫時禁用特定類別的通知。

9.3.3 重複提醒設置

(1) 預設重複模式

系統提供常用的重複模式預設：

- 每日重複：適用於日常習慣和例行任務。
- 每週重複：可選擇一週中的特定幾天，適合固定會議或活動。
- 每月重複：可選擇每月的特定日期（如每月第一個星期一）。
- 每年重複：適用於生日、紀念日等年度事件。

(2) 自定義重複規則

對於不符合標準模式的重複需求，用戶可以創建自定義重複規則：

- 設置特定間隔的重複（如每3日、每2周、每6個月）。
- 設置重複的開始和結束日期。
- 設置重複的次數限制。

(3) 重複提醒的例外處理

系統允許用戶為重複提醒設置例外情況：

- 跳過特定日期的重複。
- 修改特定重複實例的時間或內容，而不影響其他重複。

- 臨時暫停重複規則，稍後恢復。

9.3.4 分類與標籤系統

(1) 類別管理

用戶可以創建和管理提醒類別，以便更好地組織不同類型的提醒：

- 創建自定義類別（如工作、學習、家庭等）。
- 為類別設置名稱和可選的顏色標識。
- 編輯或刪除現有類別。
- 將提醒從一個類別移動到另一個類別。

系統預設了一些常用類別，新用戶可以直接使用或根據需要修改。

(2) 優先級標記

每個提醒可以設置優先級，幫助用戶識別最重要的任務：

- 高優先級：緊急且重要的任務，視覺上使用紅色標識。
- 中優先級：重要但不緊急的任務，視覺上使用橙色標識。
- 低優先級：常規任務，視覺上使用綠色標識。

優先級信息在列表視圖中清晰顯示，用戶可以快速識別需要優先處理的任務。

(3) 完成狀態跟蹤

系統跟蹤每個提醒的完成狀態：

- 未完成：默認狀態，表示任務尚未完成。
- 已完成：用戶可以標記任務為已完成。
- 部分完成：適用於包含多個子任務的複雜提醒。

已完成的提醒可以選擇性地從主視圖中隱藏，但仍可通過篩選查看，便於回顧已完成的任務。

9.3.5 搜索與篩選功能

(1) 全文搜索

強大的搜索功能允許用戶快速找到特定提醒：

- 搜索提醒的標題和詳細描述。
- 支持關鍵詞和短語搜索。
- 即時搜索結果顯示，隨著輸入內容的變化而更新。

搜索結果按相關性排序，並高亮顯示匹配的文本部分。

(2) 多條件篩選

用戶可以使用多種條件組合篩選提醒：

- 按類別篩選。
- 按優先級篩選。

按完成狀態篩選（已完成/未完成）。

按日期範圍篩選（今日、本週、本月、自定義日期範圍）。

篩選條件可以組合使用，例如“本週內的高優先級工作類別提醒”。

(3) 排序選項

提醒列表可以按照不同條件排序：

按日期時間排序（最近的優先或最遠的優先）。

按優先級排序（高到低或低到高）。

按創建時間排序。

按標題字母順序排序。

用戶可以根據當前需求靈活切換排序方式。

9.3.6 多視圖顯示

(1) 列表視圖

默認的列表視圖以條目形式顯示所有提醒：

每個提醒顯示為一行，包含關鍵信息（標題、時間、優先級等）。

支持展開查看提醒的詳細信息。

- 提供快速操作按鈕（完成、編輯、刪除）。

- 按時間順序或其他用戶選擇的順序排列。

列表視圖適合快速瀏覽和管理大量提醒。

（2）日曆視圖

日曆視圖以日曆格式展示提醒：

- 按月、周或日顯示提醒。

- 在對應日期上顯示提醒數量和簡要信息。

- 點擊日期可查看當天的詳細提醒列表。

- 支持拖放操作調整提醒日期。

日曆視圖幫助用戶瞭解提醒的時間分佈，識別繁忙和空閒的時間段。

（3）統計視圖

統計視圖提供提醒數據的可視化分析：

- 顯示不同類別提醒的數量分佈。

- 展示已完成和未完成提醒的比例。

- 按時間段（日、周、月）顯示提醒完成情況。

- 提供優先級分佈的統計信息。

統計視圖幫助用戶瞭解自己的任務完成效率和時間分配情況。

9.3.7 數據管理與安全

(1) 本地數據存儲

- 所有提醒數據存儲在用戶本地設備上。
- 使用SQLite數據庫進行高效存儲和檢索。
- 數據文件存放在用戶專用目錄，確保訪問權限。
- 定期自動保存，防止數據丟失。
- 本地存儲確保用戶數據隱私，不會被上傳到雲端或第三方服務器。

(2) 數據備份與恢復

- 系統提供數據備份和恢復功能。
- 手動觸發備份，將數據導出到用戶指定位置。
- 自動定期備份，保護數據安全。
- 從備份文件恢復數據的簡單流程。
- 支持選擇性恢復（如僅恢復特定類別的提醒）。
- 備份文件採用加密格式，保護敏感信息安全。

(3) 數據導入導出

- 支持與其他系統交換數據。
- 導出提醒數據為通用格式（如CSV、iCalendar）。
- 從其他日曆或任務管理工具導入數據。
- 批量導入導出功能，便於系統遷移。

9.3.8 用戶界面與交互

(1) 現代化界面設計

應用現代化的界面設計：

- 卡片式佈局，清晰分隔不同功能區域。
- 精心設計的色彩方案，提供舒適的視覺體驗。
- 響應式設計，適應不同屏幕尺寸。
- 支持淺色和深色主題，適應不同使用環境。

(2) 交互優化

優化的用戶交互體驗：

- 拖放操作支持（如拖動提醒到不同日期或類別）。
- 右鍵菜單提供上下文相關的快捷操作。
- 鍵盤快捷鍵支持，提高操作效率。
- 平滑的動畫效果，增強視覺反饋。

(3) 可訪問性設計

考慮不同用戶需求的可訪問性設計：

- 支持屏幕閱讀器，便於視障用戶使用。
- 可調整的字體大小和對比度。
- 鍵盤導航支持，減少鼠標依賴。
- 顏色設計考慮色盲用戶的需求。

本地事務提醒助手通過這些核心功能，為用戶提供了一個全面、靈活且易用的提醒管理系統。它不僅滿足基本的提醒需求，還通過先進的分類、篩選、重複規則和多視圖展示等功能，幫助用戶更有效地組織時間和任務。同時，本地存儲的特性確保了數據隱私和安全，使其成為一個值得信賴的個人和工作生活助手。

9.4 具體實現步驟：構建你的本地事務提醒助手

我們將詳細介紹如何使用Windsurf開發一個本地事務提醒助手，從啟動準備到最終完善，每個步驟都有清晰的操作指南。

9.4.1 第一步：開發之前的準備工作

在開始開發之前，我們需要完成以下準備工作：

- 啟動Windsurf應用程序
- 在本地創建名為“本地事務提醒助手”的項目文件夾
- 等待Windsurf主界面加載完成

9.4.2 第二步：開始應用開發

1.需求梳理階段

核心目標：利用AI的理解與推理能力，將模糊想法轉化為結構化的需求文檔，特別適合編程經驗較少的用戶。

2.具體步驟

1) 選擇推理模型。

□ 推薦使用DeepSeek R1模型，它具有強大的自然語言理解能力。

□ 在Windsurf的模型選擇列表中找到並選擇DeepSeek R1（如圖9-1所示）。

2) 輸入初步想法。

□ 打開Write模式的對話框。

□ 用清晰的自然語言描述應用需求。

DeepSeek提示詞示例：

我想开发一个本地事务提醒助手，请帮我写一份需求文档。

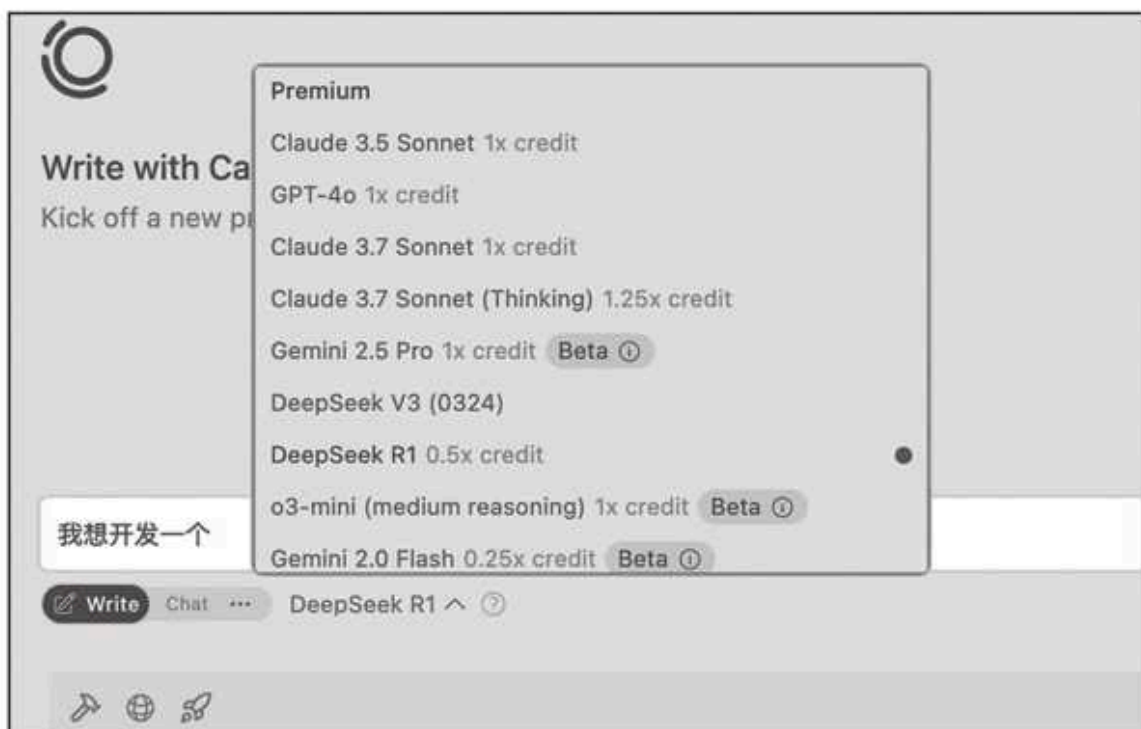


圖9-1 DeepSeek R1模型選擇示例

3) AI處理與生成。

□ Windsurf會解析你的自然語言指令。

□ AI將生成包含功能列表和界面建議的需求文檔（如圖9-2所示）。



圖9-2 DeepSeek需求文檔生成示例

4) 審查與調整。

❑ 仔細閱讀AI生成的需求文檔。

❑ 檢查功能描述和界面佈局是否符合預期。

❑ 可以要求添加新功能或修改設計，如增加“提醒優先級”功能。

❑ 持續與AI交互，直到需求文檔完全滿意。

9.4.3 第三步：實現開發階段

具體步驟：

- 1) 選擇編碼模型。
 - ❑ 切換到更適合代碼生成的模型，如Claude-3.7-sonnet。
 - ❑ 在Windsurf的模型列表中進行切換（如圖9-3所示）。

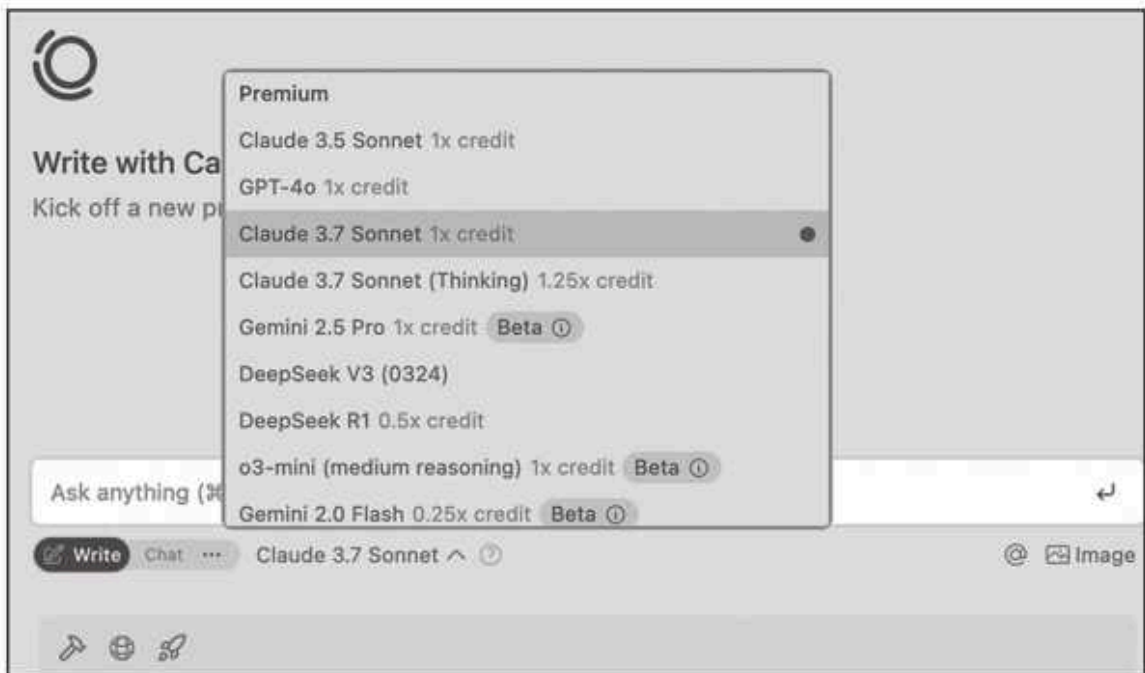


圖9-3 切換更加強大的編碼模型示例

- 2) 下達開發指令。
 - ❑ 在Write模式下發出明確的編碼指令。
 - ❑ AI提示詞示例。

请你阅读刚刚生成的需求文档，使用 Python 和 PyQt/Tkinter 来实现这个提醒助手。

- 3) AI編寫代碼。
 - ❑ Windsurf開始分析需求並生成相應代碼。
 - ❑ 即時查看文件創建和代碼填充過程。

4) 安裝依賴。

等待AI完成初步編碼。

按提示安裝必要的外部依賴庫。

5) 運行與測試。

按Windsurf指引運行應用程序（見圖9-4）。

測試核心功能的實現情況。

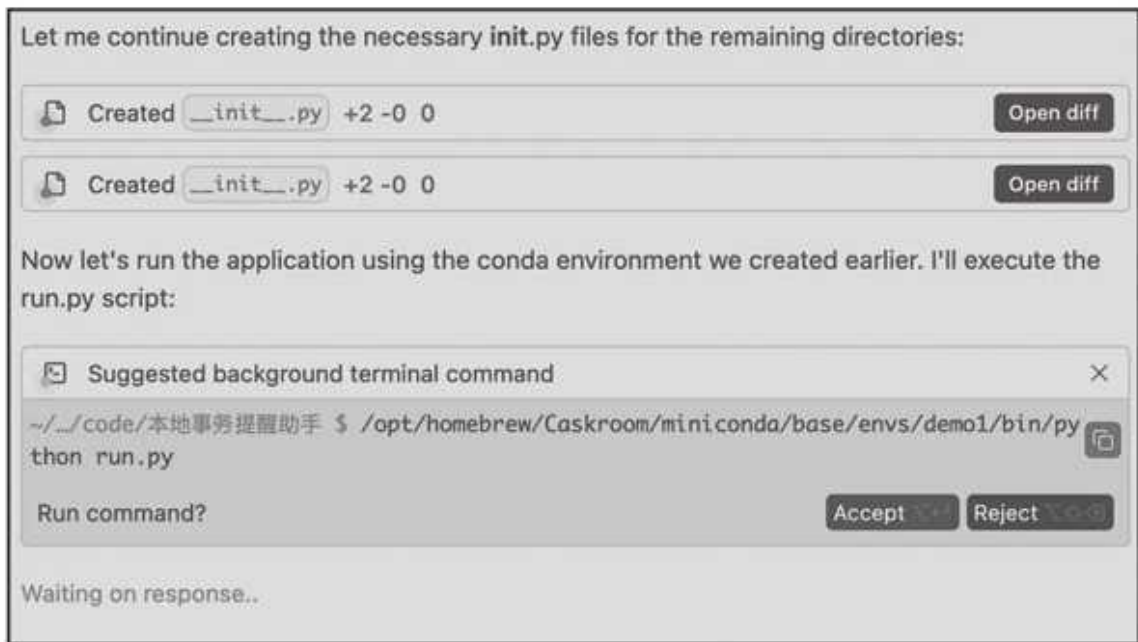


圖9-4 Windsurf指引運行應用程序示例

單擊Accept（接受）之後，讓我們查看運行的結果界面，如圖9-5所示。



圖9-5 本地事務提醒助手示例

讓我們新建一個提醒，測試一下本地事務提醒助手的一些功能情況，如圖9-6所示。



圖9-6 新建提醒示例

現在我們已經可以通過新建提醒放在列表視圖窗口中了，讓我們測試其他功能是否可以正常使用。

單擊“編輯提醒”按鈕，將更改提醒為工作，時間設置為早上9點，在列表視圖中查看效果，如圖9-7所示。

可以看到左下角已顯示我們更新後的提醒內容。通過圖9-7，現在我們可以清晰地看到提醒的標題已成功修改為“工作”，並且時間設置為早上9點。此外，你還可以看到提醒的其他詳細信息，例如類別、優先級等也已更新為相應設置。這些界面功能確保了用戶能夠直觀且高效地管理和編輯提醒事項，幫助更好地規劃日常事務和工作安排。

9.4.4 第四步：完善細節階段

核心目標：優化界面美觀度和交互流暢性。



圖9-7 編輯提醒示例

1) 具體步驟如下。

- ❑ 評估與反饋：審視當前界面和交互流程。
- ❑ 識別需要改進的地方並提出優化指令。
- ❑ 在Write模式下描述具體優化需求。

2) 示例優化要求如下。

- ❑ 添加柔和的色彩搭配。
- ❑ 增加按鈕懸停效果。

□ 優化交互反饋。

AI提示詞示例：

请你帮我优化一下界面设计，要求界面美观精美，增加按钮悬停效果和交互设计。

3) AI更新代碼。

□ Windsurf根據要求修改界面代碼。

4) 驗證改進。

□ 運行優化後的程序。

□ 確認界面改進效果（如圖9-8所示）。

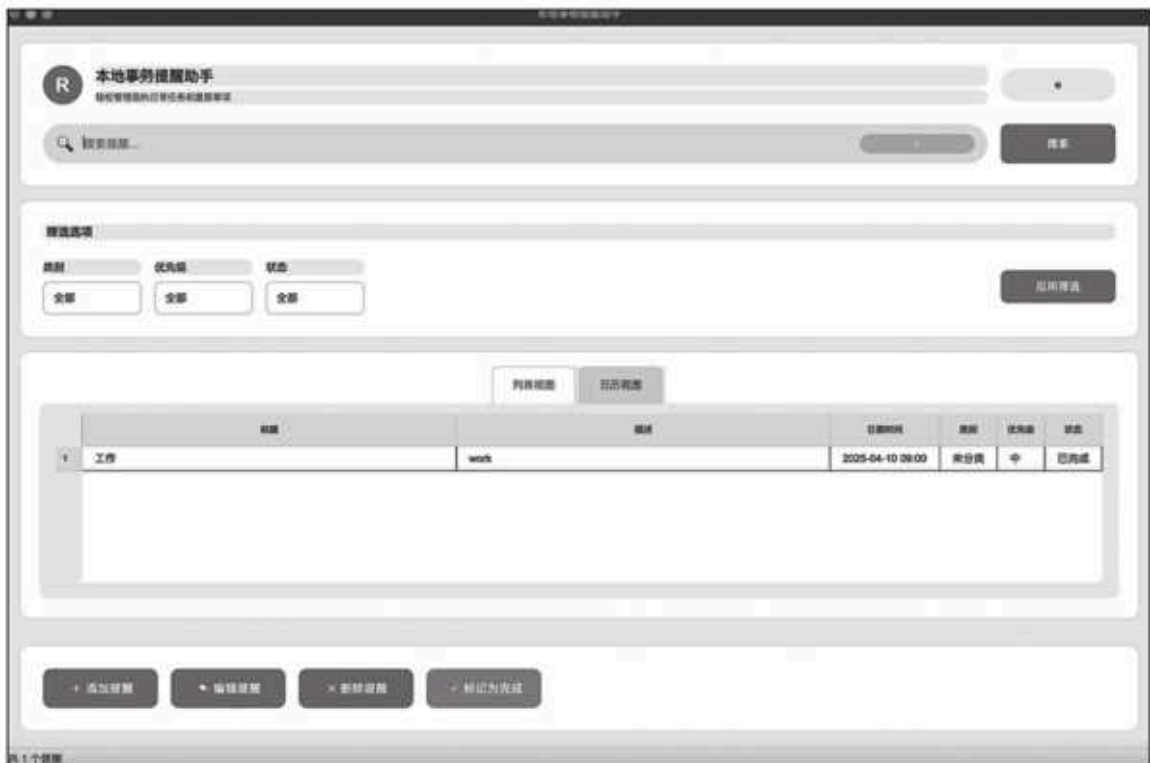


圖9-8 優化之後的本地事務提醒助手界面

可以看到優化之後的本地事務提醒助手的界面更加美觀、現代化。

5) 持續迭代。

□ 根據需要進行多輪優化。

□ 直到達到理想的使用體驗。

通過以上步驟，我們可以藉助Windsurf的AI能力，高效地完成從需求到實現的全過程。整個過程強調交互式開發，通過持續的反饋和優化，最終打造出一個功能完善、界面優美的本地事務提醒助手。

讓我們對界面整體進行迭代優化之後查看效果，如圖9-9所示。

通過與Windsurf的交互，我們成功地對本地事務提醒助手的界面進行了全面優化，從一個基礎的功能性界面提升為一個現代化、美觀且用戶友好的應用程序。以下是界面優化的主要方面和成果：從最初的簡單功能性界面，我們轉向了一個以用戶體驗為中心的設計理念，注重視覺美感、交互流暢性和信息層次結構。



圖9-9 界面迭代優化之後的效果

通過這些優化，我們將本地事務提醒助手從一個基礎的功能性應用轉變為一個視覺吸引力強、用戶友好的現代應用程序。新的界面不僅保留了原有的所有功能，還通過更好的視覺設計提高了用戶體驗，使應用程序更加專業和易用。

這次界面優化展示瞭如何通過精心的設計和細節關注，顯著提升應用程序的用戶體驗，而不需要改變其核心功能。

9.5 常見問題與解決思路

1. 使用體驗問題

問題1： 操作不便，找不到所需功能。

向DeepSeek提問： 系統在處理大量數據時變得很慢，如何優化？

解決思路： 優化操作流程，減少完成任務所需的單擊次數或步驟。重新設計導航菜單，將常用功能置於明顯位置。添加功能搜索，幫助用戶快速找到需要的功能。提供詳細的操作指南和教程。

問題2： 界面複雜，信息過多導致視覺疲勞。

向DeepSeek提問： 應用長時間運行後內存佔用很高。應該採取什麼措施？

解決思路： 採用簡潔的界面設計，使用卡片式佈局清晰分隔不同功能區域。提供自定義主題和佈局選項，允許用戶隱藏不常用功能。增加空白區域，改善視覺體驗。實現可摺疊的信息區塊，讓用戶按需展開詳情。

問題3： 響應速度慢，操作有延遲。

向DeepSeek提問： 創建週期性任務的步驟太多，要如何進行簡化？

解決思路： 優化數據庫查詢性能，減少不必要的數據加載。實現數據緩存機制，加快常用數據的訪問速度。優化應用啟動過

程，減少加載時間。使用異步處理方式，避免界面卡頓。定期清理臨時文件和緩存，保持系統運行流暢。

問題4： 通知不可靠，有時錯過重要提醒。

向DeepSeek提問： 用戶經常找不到需要的功能，要如何進行改進？

解決思路： 改進通知系統，確保通知能夠準時觸發。提供多渠道通知選項（桌面通知、聲音提醒等）。實現通知確認機制，追蹤用戶是否已查看通知。對重要提醒設置重複通知機制，直到用戶確認。添加通知日誌，記錄所有通知的發送和接收狀態。

2.功能相關問題

問題1： 無法設置複雜的重複規則。

向DeepSeek提問： 備份數據不完整或恢復失敗，如何確保備份數據的可靠性？

解決思路： 增強重複規則設置功能，支持自定義間隔（如每3天、每2周）。添加例外日期設置，允許跳過特定日期的重複。支持設置重複結束條件（如重複次數或結束日期）。提供重複規則預覽，幫助用戶理解設置效果。

問題2： 數據備份與恢復操作複雜。

向DeepSeek提問： 擔心敏感數據洩露，如何加強數據安全保護？

解決思路： 簡化備份流程，提供一鍵備份選項。實現自動定期備份功能，減少用戶手動操作。優化恢復界面，清晰顯示可用

備份及其內容和時間。支持選擇性恢復（如僅恢復特定類別的提醒）。提供備份驗證機制，確保備份數據的完整性。

問題3： 搜索功能不夠精準。

向DeepSeek提問： 如何解決搜索功能不夠精確的問題？

解決思路： 改進搜索算法，支持模糊匹配和關鍵詞搜索。增加高級搜索選項，允許按多個條件組合搜索。實現搜索建議功能，根據用戶輸入提供可能的搜索項。優化搜索結果排序，將最相關的結果置頂。支持保存常用搜索條件，方便重複使用。

問題4： 缺少數據分析和統計功能。

向DeepSeek提問： 如何解決數據分析和統計功能缺失的問題？

解決思路： 開發統計分析模塊，展示任務完成率、按類別分佈等數據。提供可視化圖表，直觀展示時間管理效率。添加趨勢分析功能，幫助用戶識別時間管理模式。支持自定義報表，滿足不同用戶的分析需求。實現數據導出功能，便於在其他工具中進行深入分析。

3.技術與兼容性問題

問題1： 在某些操作系統版本上運行不穩定。

向DeepSeek提問： 如何解決操作系統運行不穩定的問題？

解決思路： 進行全面的兼容性測試，識別不同操作系統版本的問題。優化代碼，確保跨平臺兼容性。提供針對特定系統版本

的優化補丁。建立自動化測試流程，及時發現和解決兼容性問題。明確標註支持的系統版本要求。

問題2： 數據同步問題，多設備使用時數據不一致。

向DeepSeek提問： 如何解決多設備數據同步不一致的問題？

解決思路： 實現可靠的數據同步機制，確保多設備間數據一致性。使用時間戳和版本控制，解決數據衝突。提供同步狀態指示器，讓用戶瞭解同步進度和結果。支持手動觸發同步，應對自動同步失敗的情況。實現離線操作支持，稍後自動同步。

問題3： 內存佔用過高，影響系統性能。

向DeepSeek提問： 如何解決內存佔用過高的問題？

解決思路： 優化內存使用，減少不必要的資源佔用。實現資源自動釋放機制，釋放長時間未使用的資源。優化數據加載策略，採用分頁加載減少一次性內存佔用。提供低內存模式選項，適應配置較低的設備。定期進行內存洩漏檢測和修復。

問題4： 導入外部數據格式支持有限。

向DeepSeek提問： 如何解決導入外部數據格式的問題？

解決思路： 擴展導入功能，支持更多常用格式（如iCalendar、CSV、JSON等）。提供導入嚮導，引導用戶完成數據映射和轉換。增加數據驗證步驟，確保導入數據的有效性。支持部分導入選項，允許用戶選擇需要導入的數據項。提供導入預覽功能，讓用戶在確認前查看導入效果。

4.隱私與安全問題

問題1： 擔心個人數據安全。

向DeepSeek提問： 如何確保用戶數據的安全性和隱私保護？

解決思路： 實現本地數據加密存儲，保護敏感信息。提供密碼保護選項，防止未授權訪問。明確隱私政策，說明數據使用和保護措施。不收集不必要的用戶信息，尊重用戶隱私。提供數據刪除工具，允許用戶完全清除個人數據。

問題2： 備份數據可能被他人訪問。

向DeepSeek提問： 如何保護備份數據的安全性？

解決思路： 對備份文件進行加密處理，確保即使文件被獲取也無法讀取內容。提供自定義加密密鑰選項，增強安全性。支持安全的備份位置選擇，如加密雲存儲。實現備份訪問日誌，記錄備份文件的訪問情況。提供備份文件完整性驗證，防止備份被篡改。

問題3： 應用權限要求過多。

向DeepSeek提問： 如何解決應用權限要求過多的問題？

解決思路： 最小化所需權限，只請求必要的系統權限。明確解釋每項權限的用途，增加用戶信任。提供功能降級選項，在用戶拒絕某些權限時仍能使用核心功能。實現權限使用透明度，讓用戶知道何時使用了哪些權限。支持隨時撤銷權限，尊重用戶選擇。

5.用戶特定需求問題

問題1： 需要適應不同的工作流程。

向DeepSeek提問： 如何適應全部的工作流程？

解決思路： 提供工作流自定義選項，適應不同用戶的使用習慣。支持模板創建和共享，滿足特定工作場景需求。實現可配置的提醒流程，允許用戶定義提醒的生命週期。提供API或插件系統，支持與其他工具集成。收集用戶反饋，持續優化以適應多樣化需求。

問題2： 特殊行業有額外功能需求。

向DeepSeek提問： 如何滿足特殊行業的額外功能需求？

解決思路： 開發行業特定模塊，如會議管理、項目追蹤、客戶跟進等。提供自定義字段功能，滿足特定信息記錄需求。支持數據導出為行業標準格式。考慮開發專業版本，針對特定行業提供深度功能。與行業專家合作，確保功能設計符合實際工作需求。

問題3： 需要團隊協作功能。

向DeepSeek提問： 如何解決團隊協作功能？

解決思路： 開發團隊版本，支持提醒共享和任務分配。實現權限管理系統，控制不同成員的訪問和編輯權限。添加評論和討論功能，促進團隊溝通。提供團隊活動日誌，記錄所有成員的操作。支持團隊統計和報表，幫助管理者瞭解團隊效率。

6.學習與適應問題

問題1： 新用戶上手困難。

向DeepSeek提問： 如何解決新手對系統上手適應困難的問題？

解決思路： 設計交互式新手引導，幫助用戶瞭解核心功能。提供視頻教程和詳細文檔，支持不同學習方式。實現智能提示系統，在用戶操作過程中提供上下文幫助。設置默認配置和模板，減少初始設置負擔。提供示例數據，幫助用戶理解系統工作方式。

問題2： 高級功能發現困難。

向DeepSeek提問： 如何解決用戶發現及使用高級功能的問題？

解決思路： 實現功能發現機制，根據用戶使用情況推薦相關高級功能。提供功能導覽，系統地介紹所有可用功能。在適當時機展示功能提示，如用戶完成相關基礎操作後。創建高級用戶指南，詳細說明覆雜功能的使用方法。設立用戶社區，鼓勵經驗分享和功能探索。

問題3： 習慣舊系統，適應新界面困難。

向DeepSeek提問： 如何解決用戶對新界面功能適應的問題？

解決思路： 提供界面切換選項，支持經典視圖和現代視圖。實現自定義界面佈局，允許用戶調整到熟悉的排列方式。提供快捷鍵自定義，保留用戶習慣的操作方式。設計平滑過渡期，逐步引入新功能和界面變化。收集用戶反饋，持續優化界面易用性。

通過系統地解決這些常見問題，本地事務提醒助手可以不斷提升用戶體驗，滿足多樣化的需求。關鍵是保持對用戶反饋的開放態度，持續優化產品功能和性能，同時確保數據安全和隱私保護。隨著技術的發展和用戶需求的變化，提醒助手也需要不斷創新，引入新功能和改進現有功能，保持產品的競爭力和實用性。

9.6 結語

本章深入探討了本地事務提醒助手的開發與實現，這是一款強大而實用的桌面應用程序，旨在幫助用戶高效管理日常任務和重要事項。通過本章的學習，我們掌握了以下幾個關鍵概念：

□ 應用程序架構設計：我們瞭解瞭如何設計一個結構清晰的應用程序架構。

□ 用戶界面開發：我們學習瞭如何使用PyQt6創建現代化、用戶友好的界面。

- 設計直觀的主窗口布局和導航結構。
- 實現表單控件和數據輸入驗證。
- 創建響應式設計，適應不同屏幕尺寸。
- 添加視覺反饋和動畫效果，提升用戶體驗。

□ 核心功能實現：我們詳細探討了提醒助手的核心功能模塊，包括提醒創建與管理、通知系統、重複規則設置、分類與標籤系統、搜索與篩選功能等。

每個功能模塊都經過精心設計，確保既滿足用戶需求，又保持代碼的可維護性。

本地事務提醒助手是一個綜合性項目，涵蓋了桌面應用開發的多個方面，通過開發這個項目，我們積累了寶貴的實戰經驗，為今後開發更復雜的應用程序奠定了堅實基礎。

在下一章中，我們將學習如何打造一個個人網站，包括網站架構設計、前端界面開發、後端功能實現以及網站部署與維護。這將是我們從桌面應用開發到Web開發的重要轉變，幫助我們拓展技術視野，掌握全棧開發能力。

第10章

打造你的第一個網站

在完成了本地事務提醒助手的開發後，我們將目光轉向Web開發領域。本章將帶領你開發你的第一個網站，這是一個激動人心的轉折點，標誌著你從桌面應用開發邁向網絡應用開發。

我們將把前幾章積累的Python編程經驗和項目開發技能應用到網站開發中。從本地應用到Web應用，這樣的轉變不僅能拓展你的技術視野，還能讓你掌握更多現代化的開發方法。

在網站開發過程中，你將學習：

- ❑ 如何規劃和設計一個網站項目。
- ❑ 如何選擇合適的技術框架。
- ❑ 如何實現基本的網站功能。
- ❑ 如何使用AI工具加速開發過程。

本章的項目將建立在前面章節的基礎上，但會引入許多新的概念和技術。Web開發與桌面應用開發有著顯著的區別，你將瞭解到網絡通信、前後端分離、響應式設計等重要概念。這些知識將極大地豐富你的開發技能庫。

讓我們開始這個令人興奮的網站開發之旅吧。記住，在遇到困難時，DeepSeek始終是你的得力助手，它能為你提供及時的指導和解決方案。

10.1 本章學習目標

在本章中，我們將踏上網站開發的旅程，學習如何藉助AI從零開始開發一個網站。學習目標包括：

(1) 學習網站基礎結構搭建

- ❑ 掌握HTML語義化標籤的使用。
- ❑ 理解網站文件組織結構。
- ❑ 學習頁面間的鏈接和導航設計。

(2) 掌握響應式設計技術

- ❑ 使用CSS（串聯樣式表）媒體查詢實現多設備適配。
- ❑ 學習彈性佈局和網格佈局技術。
- ❑ 確保網站在不同屏幕尺寸下的良好顯示效果。

(3) 實現交互效果與動畫

- ❑ 使用CSS過渡和動畫增強用戶體驗。
- ❑ 實現滾動觸發動畫效果。
- ❑ 添加懸停效果和其他交互元素。

(4) 學習組件化開發思想

- ❑ 將網站拆分為可複用的組件。

- 學習模塊化CSS編寫方法。
- 理解代碼複用和維護的最佳實踐。

(5) 掌握圖像處理與優化

- 學習圖像格式選擇和優化。
- 實現圖像響應式加載。
- 使用CSS創建圖形和圖標。

(6) 理解網站性能優化基礎

- 學習資源加載優化技術。
- 掌握CSS和JavaScript的性能考量。
- 瞭解基本的SEO（搜索引擎優化）方法。

通過本章的學習，讀者不僅能掌握網站開發的技術細節，更能理解網站開發的整體流程和思維方式，為今後的進階學習打下堅實基礎。

10.2 瞭解網站開發基礎

1. 什麼是網站開發

網站開發是指創建和維護網站的過程，涉及多個方面的工作，包括網頁設計、內容創作、客戶端/服務器端編程以及網絡安全配置等。從技術角度看，網站開發可分為前端開發和後端開發兩大部分。

前端開發主要關注用戶可以直接看到和交互的部分，使用HTML、CSS和JavaScript等技術：

- ❑ HTML提供網頁的基本結構和內容。
- ❑ CSS負責網頁的視覺表現和佈局。
- ❑ JavaScript為網頁添加交互功能和動態效果。

後端開發則處理服務器端的邏輯和數據存儲：

- ❑ 服務器端編程語言（如PHP、Python、Ruby、Java等）。
- ❑ 數據庫管理（如MySQL、MongoDB等）。
- ❑ 服務器配置和維護。

對於初學者而言，掌握前端開發技術是進入網站開發領域的理想起點，因為它不需要複雜的服務器設置，就能創建出功能完善的靜態網站。隨著技術的發展，現代前端開發已經能夠實現許多曾經需要後端支持的功能。

2. 網站開發的優勢

自主開發網站相比使用現成的網站建設工具或平臺，具有以下顯著優勢：

(1) 靈活性和定製化

- ❑ 完全控制網站的每個方面（從設計到功能）。
- ❑ 能夠根據特定需求定製獨特的用戶體驗。
- ❑ 不受第三方平臺限制，可以實現任何創意構想。

(2) 性能優化

- ❑ 可以精確控制代碼質量和加載速度。
- ❑ 減少不必要的功能和代碼，提高網站性能。
- ❑ 針對特定用戶群體和使用場景進行優化。

(3) 成本效益

- ❑ 避免長期訂閱費用和平臺限制。
- ❑ 一次性學習，長期受益。
- ❑ 隨著技能提升，可以承接更復雜的項目。

(4) 學習價值

- ❑ 掌握實用的技術技能，增強就業競爭力。
- ❑ 理解網絡技術的工作原理。
- ❑ 培養解決問題的能力和邏輯思維。

(5) 品牌一致性

- ❑ 確保網站與品牌形象完全一致。
- ❑ 避免使用模板導致的“千篇一律”問題。
- ❑ 建立獨特的在線形象。

3. 網站開發的使用場景

網站開發適用於多種場景，根據不同需求可以創建各種類型的網站：

(1) 個人展示

- ❑ 個人作品集網站，展示設計、攝影、藝術作品等。
- ❑ 個人博客，分享專業知識和經驗。
- ❑ 個人簡歷網站，提升求職競爭力。

(2) 企業展示

- ❑ 公司官方網站，展示企業形象和服務。
- ❑ 產品展示網站，詳細介紹產品特點和優勢。
- ❑ 企業內部知識庫或工具網站。

(3) 電子商務

- ❑ 在線商店，銷售產品或服務。
- ❑ 會員訂閱網站，提供付費內容或服務。

□ 預訂系統，如餐廳、酒店或活動預訂。

(4) 教育與信息

□ 在線課程平臺。

□ 知識分享社區。

□ 專業領域的資源庫。

(5) 社交與社區

□ 興趣社區網站。

□ 行業交流平臺。

□ 用戶生成內容的分享平臺。

(6) 營銷與推廣

□ 活動宣傳網站。

□ 產品發佈專題頁。

□ 營銷落地頁。

隨著移動設備的普及和互聯網技術的發展，響應式網站設計變得尤為重要，它能確保網站在不同設備上都提供良好的用戶體驗。同時，隨著用戶對網站性能和體驗要求的提高，掌握現代網站開發技術也變得越來越重要。

學習網站開發不僅可以創建滿足特定需求的網站，還能夠培養全面的技術思維和問題解決能力，這些技能在數字化時代具有廣泛的應用價值。

10.3 網站開發的核心功能

現代網站無論規模大小，通常都需要實現一系列核心功能，以確保良好的用戶體驗和網站效果。以下是網站開發中最常見的核心功能。

1. 響應式設計

- 跨設備兼容性：確保網站在不同尺寸的設備（個人計算機、平板計算機、手機）上都能正常顯示。

- 彈性佈局：使用相對單位和彈性佈局技術，使內容能夠自適應不同屏幕。

- 媒體查詢：根據屏幕尺寸應用不同的樣式規則。

- 圖像優化：根據設備特性加載不同分辨率的圖像。

2. 導航系統

- 直觀的菜單結構：清晰的層次結構，幫助用戶快速找到所需信息。

- 麵包屑導航：顯示用戶在網站中的位置，便於返回上級頁面。

- 搜索功能：允許用戶直接查找特定內容。

- 移動端適應性：在小屏幕設備上轉換為更適合觸控的導航形式（如漢堡菜單）。

3. 內容展示

- 清晰的信息層次：使用標題、段落、列表等元素組織內容。

- 多媒體支持：整合圖片、視頻、音頻等多媒體元素。

- 內容分頁：將大量內容分割為易於消費的小塊。

- 動態內容加載：使用懶加載等技術優化長頁面的性能。

4.用戶交互元素

- 表單設計：創建用戶友好的輸入界面。

- 即時反饋：對用戶操作提供即時視覺或文字反饋。

- 動畫效果：使用適度的動畫增強用戶體驗和引導注意力。

- 交互組件：下拉菜單、輪播圖、標籤頁等增強內容組織和展示。

5.性能優化

- 資源壓縮：減小CSS、JavaScript和圖像文件的大小。

- 代碼分割：只加載當前頁面所需的代碼。

- 緩存策略：合理使用瀏覽器緩存減少重複加載。

- 延遲加載：非關鍵資源推遲加載，優先顯示核心內容。

6.安全措施

- 表單驗證：客戶端和服務器端雙重驗證用戶輸入。

- ❑ HTTPS（超文本傳輸安全協議）實現：加密數據傳輸保護用戶隱私。

- ❑ 跨站腳本防護：防止XSS（跨站腳本）等常見網絡攻擊。

- ❑ 安全更新：定期更新依賴庫和框架，修復已知漏洞。

7.分析與跟蹤

- ❑ 訪問統計：記錄網站訪問量和用戶行為。

- ❑ 轉化跟蹤：監控重要操作的完成情況。

- ❑ 熱圖分析：瞭解用戶訪問和關注的焦點。

- ❑ 性能監控：跟蹤頁面加載速度和資源使用情況。

8.社交媒體集成

- ❑ 分享功能：允許用戶將內容分享到社交平臺。

- ❑ 社交登錄：使用社交媒體賬號快速登錄。

- ❑ 社交媒體反饋：展示相關社交媒體內容。

- ❑ 社交互動：評論、點贊等社交互動功能。

掌握這些核心功能的開發技術，能夠幫助開發者創建既美觀又實用的網站。隨著經驗的積累，開發者可以逐步深入學習各個功能領域的高級技術，不斷提升網站的用戶體驗和技術實現水平。在實際項目中，可以根據具體需求和目標用戶群體，有選擇地實現和優化這些功能。

10.4 具體實現步驟：開發你的第一個網站

以下步驟將引導你如何與AI工具（Windsurf）交互，從簡單的網頁需求出發，逐步搭建起自己的網站並對網站界面進行優化。

讓我們跟隨這個實例，體驗使用Windsurf開發一個簡單展示型網站的全過程。

10.4.1 第一步：啟動與準備

啟動Windsurf。加載或創建一個名為“第一個網站”的項目文件夾（請提前創建）。確認Windsurf的AI對話框處於Write模式，以便AI可以直接生成和修改項目文件。

10.4.2 第二步：開始網站開發

1.設計並生成主頁

核心目標：創建網站的入口頁面，包含關鍵的視覺和信息元素。

操作流程：

1) 下達指令：由於展示型網站初期主要側重於結構和內容呈現，邏輯相對簡單，我們可以直接向AI描述主頁的核心構成。在對話框中輸入以下內容：

我想开发一个展示型网站：
首先，请帮我设计并生成一个专业的首页（index.html）。这个首页需要包含：一个引人注目的全屏展示区，一个介绍品牌故事或核心价值的区域。请生成相应的 HTML 和 CSS 代码。

2) AI處理與文件創建：

❑ Windsurf會開始分析你的需求並生成代碼。

❑ AI可能會提示需要創建新的文件或目錄（例如index.html, style.css等）。它通常會直接提供創建命令，請仔細閱讀並單擊Accept（或類似確認按鈕）授權執行。

3) 初步預覽：

❑ 當AI完成第一步的代碼生成後，它會告知你相關文件已創建或更新。

❑ AI會在你的項目文件夾中找到生成的index.html文件。

❑ 雙擊該文件，在你的網頁瀏覽器中打開它，或者直接在Windsurf查看初步的頁面效果（見圖10-1）。



圖10-1 在Windsurf查看初步頁面效果

2. 添加“關於我們”頁面

核心目標：創建一個新的頁面來詳細介紹背景信息。

操作流程：

1) 下達指令：

告訴AI你需要一個新的頁面。

在對話框中輸入：

現在我需要添加一個“關於我們”頁面 (about.html)。請幫我設計一個專業的介绍页面布局关于 Sigmate 公司，需要包含团队理念、发展历程等内容区域。请生成相应的 HTML 和 CSS 代码 (可以复用之前的样式或创建新的)。同时，请确保在首页和“关于我們”页面之间有链接可以互相跳转。

2) AI生成與預覽：

Windsurf會生成about.html文件及可能的樣式更新。

同樣，你可以直接在Windsurf中打開查看效果。

你會發現，使用類似的方式，你可以輕鬆地讓AI幫助你添加網站所需的各種頁面（如產品頁、聯繫頁等）。

3. 完善細節與優化體驗

核心目標：在基本頁面結構完成後，對視覺效果、交互體驗和性能進行打磨。

操作流程：

1) 提出優化指令：根據你的需求，向AI提出具體的優化點。

在對話框中輸入：

请帮我对整个网站进行一些优化，主要包括：优化图片加载效果（例如，实现懒加载或提供图片优化建议）；添加页面过渡动画（例如，页面切换时有淡入淡出效果）；改进响应式布局，确保在手机等小屏幕设备上也能良好显示。

2) AI更新與說明：

□ Windsurf會修改現有的HTML，CSS或添加JavaScript代碼來實現這些優化。

□ AI通常還會解釋它做了哪些優化以及背後的思路。仔細閱讀這些說明有助於你學習。

3) 最終預覽：再次在瀏覽器中打開你的網站頁面（特別是首頁和關於我們頁面）。你應該能觀察到優化後的效果，例如圖片加載更快，頁面切換更平滑，在調整瀏覽器窗口大小時佈局能自適應等。

通過這三個步驟的迭代，你就成功地使用AI助手構建並優化了你的第一個簡單展示型網站。具體效果圖如圖10-2、圖10-3、圖10-4所示。



圖10-2 生成的網站效果



圖10-3 網站團隊成員界面



圖10-4 網站精選作品界面

10.4.3 第三步：網頁優化

可以看到網站團隊成員界面和網站精選作品界面並沒有顯示相關照片，讓我們對這兩個界面逐步進行優化。

首先將團隊成員照片放置在項目新建的img目錄下，如圖10-5所示。

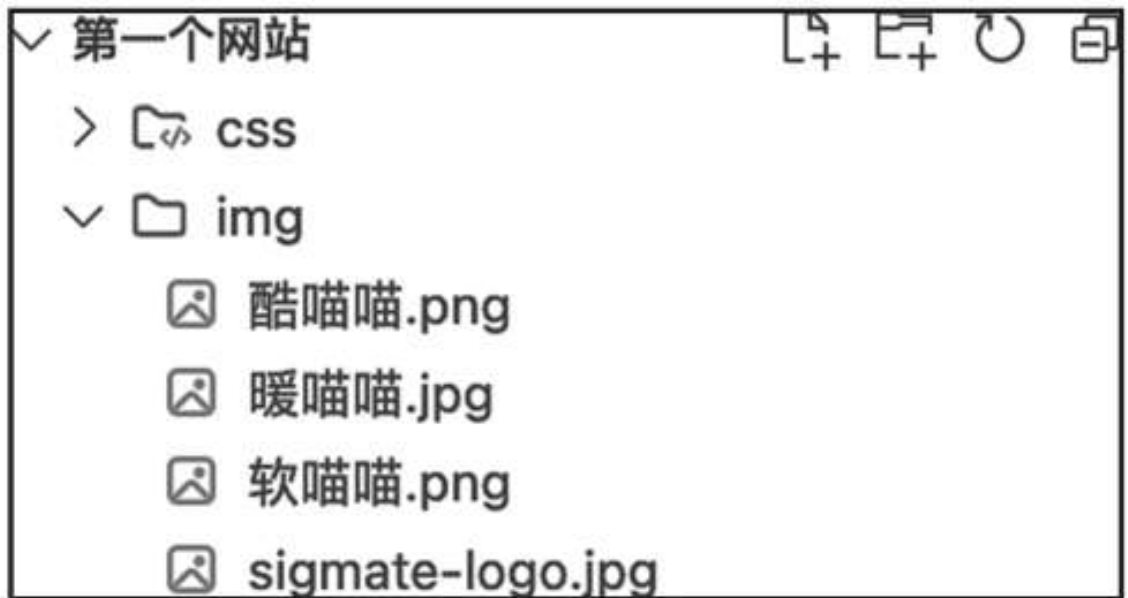


圖10-5 img目錄下的團隊成員照片

然後通過給AI下達指令（如圖10-6所示），對添加網站的團隊成員照片並界面進行優化。在對話框中輸入：

为核心团队成员添加相关图片（照片在该项目的img目录下，分别为暖喵喵、软喵喵和酷喵喵），并将成员名字更改为照片的名字，之后对网站团队成员界面进行优化。



圖10-6 AI優化過程圖

讓我們等待Windsurf進行編碼，之後單擊運行。讓我們查看經過AI優化之後得到的效果，如圖10-7所示。

可以看到Windsurf已經為團隊成員界面添加了相關的img目錄下的相關成員圖，並對團隊成員界面進行了優化。下面讓我們根據對團隊成員界面的優化思路，對網站精選作品界面進行同樣的細節優化，優化後的效果如圖10-8所示。



圖10-7 優化之後的網頁效果



圖10-8 網站精選作品界面優化後的效果

可以看到現在的網站界面已經基本達到要求了，現在讓我們嘗試來部署這個網站，只需要在聊天框中輸入“部署這個網站”(如圖10-9所示)，之後Windsurf便會自動進行部署。



圖10-9 部署網站過程

等待Windsurf幫我們的網站配備域名，並進行部署（如圖10-10所示）。



圖10-10 部署網站結果

可以看見Windsurf已經幫我們部署成功了，現在我們只需要輸入Windsurf給我們提供的網址，就可以直接在瀏覽器上直接訪問我們開發的第一個網站。

10.5 常見問題與解決思路

在開發第一個網站的過程中，初學者通常會遇到各種各樣的問題。以下是本項目中最常見的問題及其解決思路，幫助讀者在實踐中更順利地完成項目。

1. 頁面佈局問題

向DeepSeek提問： 元素無法按預期位置排列，特別是在響應式佈局中。

解決思路：

- 1) 掌握CSS盒模型概念，理解margin、padding和border的影響。
- 2) 學習Flexbox和Grid佈局系統，它們提供了強大的對齊和分佈控制。
- 3) 使用開發者工具檢查元素的實際尺寸和位置。
- 4) 採用移動優先的設計思路，從小屏幕開始，逐步擴展到大屏幕。

解決方案實例：

在本項目中，團隊成員展示區域使用了Grid佈局，確保在不同屏幕尺寸下能夠自動調整列數。

```
```css
.team-container {
```

```
display: grid;
grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
gap: 2rem;
)
...
```

## 2. 圖片顯示與優化問題

**向DeepSeek提問：** 圖片尺寸不合適，加載速度慢，或在不同設備上顯示效果不佳。

### 解決思路：

- 1) 使用適當的圖片格式（jpeg用於照片，png用於需要透明背景的圖像，webp用於現代瀏覽器）。
- 2) 實現響應式圖片，為不同屏幕提供不同分辨率的圖像。
- 3) 壓縮圖片以減少文件大小。
- 4) 使用懶加載技術延遲加載屏幕外的圖片。

### 解決方案實例：

在本項目中，使用picture元素和srcset屬性為不同設備提供不同尺寸的圖片。

```
``html
<picture>
 <source srcset="img/hero-small.jpg" media="(max-width: 600px)">
 <source srcset="img/hero-medium.jpg" media="(max-width: 1200px)">

</picture>
...
```

## 3. 導航菜單適配問題

**向DeepSeek提問：** 導航菜單在桌面端顯示正常，但在移動設備上不易使用。

**解決思路：**

- 1) 在小屏幕上將水平菜單轉換為漢堡菜單。
- 2) 確保觸摸目標足夠大，便於手指點擊。
- 3) 添加適當的動畫效果，提供視覺反饋。
- 4) 測試不同設備上的交互體驗。

**解決方案實例：**

本項目使用媒體查詢和JavaScript實現了響應式導航菜單。

```
````javascript
document.querySelector('.menu-toggle').addEventListener('click', function() {
    document.querySelector('.nav-menu').classList.toggle('active');
});
````
```

#### 4.CSS樣式衝突問題

**向DeepSeek提問：** CSS規則相互覆蓋，導致樣式不按預期應用。

**解決思路：**

- 1) 理解CSS選擇器優先級規則。
- 2) 採用BEM（塊、元素、修飾符）等命名約定，避免選擇器衝突。
- 3) 使用更具體的選擇器解決特定問題。

4) 考慮使用CSS模塊或作用域限制樣式影響範圍。

解決方案實例：

在本項目中，使用了明確的命名約定和層次結構：

```
```.css
.about-page .team-section .team-member .member-image {
  /* 特定于关于页面的团队成员图片样式 */
}
```
```

## 5. 跨瀏覽器兼容性問題

**向DeepSeek提問：** 網站在不同瀏覽器中顯示效果不一致。

解決思路：

- 1) 使用CSS前綴自動添加工具。
- 2) 參考Can I Use網站了解特性支持情況。
- 3) 實施漸進增強策略，確保基本功能在所有瀏覽器中可用。
- 4) 使用現代CSS重置樣式表統一基礎樣式。

解決方案實例：

在項目中使用了normalize.css統一不同瀏覽器的默認樣式。

```
```.html
<link rel="stylesheet" href="css/normalize.css">
```
```

## 6. 表單驗證與提交問題

**向DeepSeek提問：** 用戶提交的表單數據未經驗證或提交過程不順暢。

**解決思路：**

- 1) 使用HTML5表單驗證屬性等。
- 2) 添加JavaScript客戶端驗證增強用戶體驗。
- 3) 提供清晰的錯誤提示和視覺反饋。
- 4) 實現平滑的表單提交體驗。

**解決方案實例：**

本項目的聯繫表單使用了HTML5驗證結合JavaScript增強。

```
``html
<input type="email" name="email" required pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">
 请输入有效的电子邮件地址
````
```

7.性能優化問題

向DeepSeek提問： 網站加載速度慢，影響用戶體驗。

解決思路：

- 1) 壓縮和合並CSS、JavaScript文件。
- 2) 優化圖片大小和格式。
- 3) 實現資源的延遲加載。
- 4) 減少HTTP請求數量。

解決方案實例：

在項目中使用了延遲加載技術提高性能。

```
```javascript
document.addEventListener("DOMContentLoaded", function() {
 const lazyImages = document.querySelectorAll('.lazy-load');

 const imageObserver = new IntersectionObserver((entries, observer) => {
 entries.forEach(entry => {
 if (entry.isIntersecting) {
 const img = entry.target;
 img.src = img.dataset.src;
 img.classList.remove('lazy-load');
 observer.unobserve(img);
 }
 });
 });

 lazyImages.forEach(img => imageObserver.observe(img));
});
```
```

有了這些常見問題和解決思路，我們能夠更加清晰深刻地理解網絡結構、如何選擇合適的網頁框架，以及如何使用AI去開發這樣的一個網頁。

10.6 結語

在本章中，我們通過Windsurf編輯器和DeepSeek的協助，成功構建了第一個網站。這個項目不僅讓你掌握了網站開發的基礎知識，還幫助你理解了現代Web應用的開發流程。主要收穫如下。

1) 項目規劃與需求分析：

- 學會了如何定義網站的目標受眾和功能邊界。
- 掌握了需求分析和用戶故事編寫的方法。
- 理解了如何將大型項目分解為可管理的小任務。

2) 技術選型與架構設計：

- 瞭解了前端框架的選擇考慮因素。
- 掌握了後端框架和數據庫的選型原則。
- 學會了如何設計可擴展的網站架構。

3) 功能實現：

- 實踐了頁面結構的規劃和實現。
- 掌握了用戶交互流程的設計方法。
- 學會了如何處理數據流和業務邏輯。

4) 開發與調試技巧：

- 掌握了使用Windsurf進行網站開發的效率技巧。
- 學會了常見問題的調試方法。
- 理解了如何使用DeepSeek輔助解決開發難題。

我們已經學習了獨立開發簡單網站的方法。這些方法將為我們未來開發更複雜的Web應用打下堅實基礎。

記住，網站開發是一個持續學習和改進的過程。隨著經驗的積累，我們將能夠開發出更加專業和功能豐富的網站。建議你在完成本章後，嘗試為你的網站添加新功能，或者開始規劃你的下一個網站項目。

第三部分

總結與展望

第三部分是對全書內容的總結，也是對AI編程未來趨勢的展望。

通過對前面兩部分的學習，包括如何用DeepSeek學習Python基礎代碼和如何在Windsurf中使用DeepSeek完成實戰項目，你已經擁有了與AI協作解決問題的基礎能力，完成了一次思維的躍遷，實現了從“0”到“1”的蛻變，這比純粹的技術知識更加珍貴。

隨著技術的不斷發展，編程技能將繼續為你的職業生涯和個人成長提供無限可能。本書不僅為你提供了編程的基礎知識和實戰經驗，還為你指出了未來的學習方向。你可以通過構建個人的AI工具箱，持續擴展自己的技術棧，進一步增強自己的競爭力。在學習過程中，避免常見的學習陷阱也非常重要，只有保持實踐與理論相結合，避免過度依賴工具和框架，才能確保技能的深度和廣度。

第11章

成為AI時代的創造者

至此，我們完成了本書前十章的學習之旅。這段旅程從Python編程基礎開始，一直延伸到實際項目的開發實踐，形成了一個完整的學習體系。

雖然本書的內容即將結束，但是你的AI編程生涯才剛剛開始。無論你是專業人士，還是AI愛好者，希望本書能夠給你帶來一些幫助和啟發。

11.1 本書精華總結

在本書的學習與實踐旅程中，我們從基礎知識開始，逐步介紹了AI編程的核心能力，建立了AI編程思維框架。隨著每一章的推進，我們不僅積累了技術技能，還逐漸形成了一種全新的思維方式，以更好地駕馭AI技術並運用到實際問題中。

11.1.1 從0到1的蛻變：思維的躍遷

通過學習本書，你已經完成了從一個對AI編程感到陌生的初學者，到能夠獨立創造有價值工具的實踐者的關鍵蛻變。這種成長不僅是技能的提升，還是思維方式的深刻轉變。

□ 從使用者到創造者：你不再僅僅是AI應用或服務的消費者，而是成為能夠駕馭AI、主動創造新工具、設計新解決方案的價值創造者。

□ 從被動接受到主動構建：面對工作或生活中的痛點，你不再只是使用現成的軟件來解決，而是擁有了主動分析問題、設計並構建個性化解決方案的能力。

□ 從未知恐懼到從容掌控：面對日新月異的技術浪潮，你不再感到茫然或焦慮，因為你已經掌握了理解、運用乃至引領這股力量的基本方法，擁有了在AI時代從容應對、積極作為的底氣。

11.1.2 你已掌握的核心能力

通過前面系統性的學習和紮實的動手實踐，你已經成功掌握了以下核心能力：

(1) AI驅動的問題解決能力

- ❑ 能夠將複雜問題有效分解為AI更易處理的、可管理的小步驟。
- ❑ 熟練利用AI編程助手來編寫、補全和優化高質量的代碼。
- ❑ 掌握了藉助AI來調試代碼、定位錯誤和優化解決方案的基
本方法。

(2) 自動化工作流程的能力

- ❑ 培養了敏銳識別日常工作中可自動化的重複性任務的能力。
- ❑ 具備了設計和實現自動化腳本或簡單工具來替代手動操作的能力。
- ❑ 初步瞭解瞭如何將多個工具或步驟整合，構建更高效的統一
一工作流程。

(3) 持續學習與適應的能力

- ❑ 學會了利用AI來快速理解和掌握新的技術概念或編程知識
的方法。
- ❑ 提升了閱讀和理解他人代碼（包括AI生成的代碼）的能
力。
- ❑ 對快速變化的技術生態有了更強的適應性，知道如何藉助
工具來輔助學習。

(4) 創造有價值產品的能力

□ 能夠完成從用戶需求到產品實現的全流程思考。

□ 具備了構建基礎的、用戶友好界面的能力（無論是命令行還是簡單的圖形用戶界面）。

□ 擁有了運用編程和AI技術解決實際業務或生活中的問題的初步經驗。

這些能力就像你剛剛獲得的“黑帶”——它能證明你已經紮實地掌握了基礎功法和套路，但這絕非“武道”的終點，而是通往更高境界、更廣闊天地的新起點。

11.2 即刻行動指南：將知識轉化為力量

理論學習和項目實踐為你奠定了堅實的基礎，但真正的成長在於將所學的技能付諸實踐。現在，是時候將你掌握的知識轉化為提升效率、解決問題，甚至創造實際價值的行動了。雖然項目實踐本身就是應用的體現，但接下來的挑戰是將這些技能深入應用於更廣泛的實際場景，並通過持續的創新和優化，產生真正的影響。

打造你的個人AI工具箱

將所學付諸實踐，從優化自身工作開始。

(1) 審視你的工作流程

❑ 列出痛點：仔細思考並列出你在日常工作或學習中遇到的最耗時、最重複、最煩瑣的3~5個任務。

❑ 評估自動化潛力：分析這些任務中哪些環節可以通過本書中提到的AI編程方法來實現自動化或半自動化？

❑ 構思初步方案：為每個適合自動化的任務構思一個簡單的自動化解決方案或流程。

(2) 從“摸魚”小工具開始

❑ 選擇切入點：從上述任務中挑選一個相對簡單、但能明顯節省時間（哪怕只是幾分鐘）的小工具作為起點。

❑ 動手實現：運用你在本書中學到的與AI協作的方法（需求描述、代碼生成、測試、優化），動手將這個小工具實現出來。

□ 持續改進：在實際使用中，不斷發現問題並優化它，直到它真正成為提高效率的利器。記住，第一個版本不求完美，但求可用。

(3) 尋找“副業”或價值拓展機會

□ 觀察身邊：留意你的同事、朋友或所在社群中是否有人也遇到了類似的問題，而這些問題恰好可以通過你開發的工具或類似的編程方案來解決。

□ 思考通用性：你開發的解決方案是否具有一定的通用性，能夠滿足更廣泛的市場需求？

□ 價值轉化：探索用你的編程技能和開發出的工具幫助他人的方式，這可能會帶來額外的價值回報（無論是聲譽、經驗還是實際收入）。

11.3 避免常見陷阱

在你的AI編程學習與創造之旅中，注意避開以下幾個常見的“坑”，以便走得更穩、更遠。

(1) 完美主義陷阱

❑ 症狀：總想等到構思“完美無缺”後才開始動手，或者對初版工具要求過高。

❑ 解藥：先完成，再完美。快速實現核心功能，然後根據實際使用反饋，逐步迭代優化。

(2) 孤軍奮戰陷阱

❑ 症狀：遇到問題習慣獨自鑽研，不願意查找已有解決方案或尋求幫助。

❑ 解藥：充分利用社區資源（如GitHub、Stack Overflow、開源庫、技術論壇）。在開發一個新功能前，先搜索是否已有現成的庫或代碼片段可以借鑑或直接使用。不要重複造輪子。

(3) 技術沉迷陷阱

❑ 症狀：過分追求使用最新、最酷的技术，而忽略了工具的實際應用價值。

❑ 解藥：始終以解決實際問題為導向。選擇技術時應基於其能否高效、穩定地滿足需求，而非僅僅因為它“新”或“潮”。

(4) 忽視用戶體驗陷阱

□ 症狀：只追求功能的強大與複雜，而忽略了工具的易用性、直觀性。

□ 解藥：記住，再強大的功能，如果用戶難以理解或操作不便，其價值也會大打折扣。要考慮工具界面的簡潔性和操作的流暢性。

(5) 停滯不前陷阱

□ 症狀：掌握了一些基礎技能後就滿足現狀，停止學習和探索。

□ 解藥：技術日新月異，尤其是在AI領域。保持好奇心和持續學習的習慣至關重要。定期關注新技術動態，嘗試將新知識應用到你的項目中。

11.4 未來成長地圖：繪製你的技能進階藍圖

以下是一些你可以探索的未來成長路徑，這些路徑能幫助你不斷精進，攀登新的高峰。

(1) 技術深度

□ 專攻AI領域：深入學習特定的AI應用方向，如NLP（構建更智能的文本處理工具）、計算機視覺（開發圖像識別應用）或數據科學/機器學習（進行更複雜的預測和分析）。

□ 精通編程內功：掌握更高級的編程概念（如異步編程、併發）、設計模式（提高代碼質量和可維護性）以及算法與數據結構。

□ 擁抱雲與分佈式：學習如何在雲平臺（AWS、Azure、GCP）上部署和運行你的AI應用，瞭解分佈式系統的基本原理。

(2) 應用廣度

□ 跨界融合：將你的AI編程能力應用到不同的行業領域（金融、醫療、教育、藝術等），解決特定行業的痛點。

□ 多平臺探索：不侷限於單一類型的應用，嘗試開發網站、移動應用或更複雜的桌面軟件。

□ 技術交織：探索AI與其他新興技術（如物聯網、區塊鏈、增強現實/虛擬現實等）的結合點，創造全新的應用場景。

(3) 商業視角

□ 產品化思維：學習如何將技術解決方案打磨成用戶願意使用的商業產品或服務。

□ 市場洞察：瞭解用戶需求分析、市場定位和競品分析的基本方法。

□ 基礎運營：掌握簡單的產品運營和推廣知識，讓你的創造物觸達更多用戶。

11.5 持續發現需求的方法

要成為一名成功的AI編程實踐者，不僅需要技術能力，更需要敏銳地發現有價值的需求。技術是手段，解決問題才是目的。以下是一些幫助你持續發現需求的方法。

(1) 傾聽身邊的“抱怨”

□ 工作場所：同事經常抱怨的重複性工作、低效流程或數據處理難題往往是絕佳的自動化或工具開發機會。

□ 社交圈子：朋友或家人在日常生活中遇到的困擾、不便可能隱藏著潛在的產品創意或服務需求。

□ 自我反思：你自己在工作或生活中遇到的效率瓶頸、信息管理困難，可能也是許多人面臨的共同問題。

(2) 關注行業與技術趨勢

□ 信息源：訂閱你感興趣的領域或行業的新聞、博客、技術報告。

□ 社區互動：積極參加相關的線上/線下論壇、社區討論、技術會議，瞭解大家正在討論什麼、關注什麼。

□ 競品分析：研究市場上成功的相關產品或服務，分析它們解決了用戶的哪些核心需求，以及還有哪些未被滿足的需求。

(3) 實踐“問題日記”

□ 日常記錄：養成習慣，每天隨手記錄你在工作、學習或生活中遇到的問題（哪怕只是很小的問題）、不便之處或效率低下的地方。

□ 定期回顧：每週或每月回顧你的“問題日記”，尋找反覆出現的或具有共性的問題。

□ 價值評估：評估這些問題中哪些影響範圍廣、痛點足夠深，並且值得通過編程來嘗試解決。

11.6 致未來的創造者：你的潛力無限

11.6.1 你比想象中更強大

在這裡，我想再次強調一個核心信息：通過學習和實踐，你已經真正具備了在AI時代創造價值的基礎能力。

也許此刻，你還不完全相信自己能獨立開發出下一個改變世界的應用；也許你仍然覺得自己與那些經驗豐富的“真正程序員”還有很遠的距離。但請務必記住：

- 每個專家都曾是蹣跚學步的初學者。
- 每個偉大的產品都始於一個簡單的想法和粗糙的原型。
- 每次顛覆性的技術革命（比如當前的AI浪潮）都為充滿熱情的新人提供了前所未有的“彎道超車”的機會。

AI編程的非凡之處在於，它顯著降低了技術實現的門檻，使得更多擁有創意和洞察力的人能夠將想法快速轉化為現實。在AI時代，你不需要成為傳統定義下的“編程大師”或算法專家，就能利用AI這個強大的槓桿，設計並構建出能夠解決實際問題、創造獨特價值的解決方案。

11.6.2 共同學習，共同成長

學習之路有時可能顯得孤獨，但成長往往源於集體的智慧與力量。我們深信交流與分享的價值，因此誠摯邀請你：

- (1) 加入我們的學習社區

- ❑ 加入“DeepSeek AI編程學習群”。

- ❑ 與眾多志同道合的學習者交流心得、分享經驗。

- ❑ 大膽分享你的成功案例，也坦誠地提出你遇到的挑戰，共同探討解決方案。

(2) 參與知識共創

- ❑ 貢獻你的作品：將你開發的實用工具、代碼片段或解決方案分享給社區。

- ❑ 互助解答：積極幫助解答社區中其他成員遇到的技術或學習問題。

- ❑ 積極參與：關注並參與我們組織的線上/線下學習活動、討論或項目。

(3) 保持聯繫

- ❑ 關注官方渠道：留意我們的公眾號或其他官方平臺，獲取最新的AI編程資訊和技術動態。

- ❑ (可選) 訂閱更新：訂閱我們的電子郵件通信 (如果提供)，第一時間獲取學習資源、課程或書籍內容的更新。

- ❑ 提供反饋：你的寶貴反饋對我們至關重要，請不吝分享你的閱讀體驗和建議，幫助我們不斷完善內容。

11.7 最後的話

當我們站在這個被AI深刻塑造的時代，興奮於觸手可及的強大模型與API時，也必須正視構建真正有價值的AI應用的現實挑戰。許多深入實踐的開發者已經指出，核心的困難往往並不在於模型本身，而在於那艱鉅的“最後一公里”——如何確保AI能力可靠、無縫且有效地融入實際產品或工作流程中。

這常被形象地稱為“70%問題”。雖然現在獲取強大的預訓練模型或調用API變得日益簡單，但這僅僅是冰山一角。真正的挑戰，即佔據了絕大部分精力的“70%問題”，潛藏在以下這些看似平凡卻至關重要的環節。

(1) 數據工程的複雜性

AI的表現高度依賴於輸入數據的質量。獲取、清洗、標註、轉換數據，構建健壯的數據管道以持續餵養模型，並有效處理模型的輸出（可能包含不確定性或錯誤），是一項巨大且持續的工程挑戰。

(2) 深度集成與工作流適配

將AI功能無縫嵌入現有的軟件系統、業務流程或用戶習慣中，遠比獨立運行一個模型要複雜。

(3) 用戶體驗設計的重塑

如何設計出用戶能夠自然理解、信任並有效利用AI功能的界面與交互功能？這涉及處理AI的“黑箱”特性，管理用戶預期，設計反饋機制，以及在AI犯錯時提供優雅的降級或修正路徑。

(4) 魯棒性與邊緣案例處理

真實世界充滿了模型訓練時未曾遇見的“邊緣案例”。確保AI應用在面對異常輸入、非典型場景或低概率事件時依然能夠穩定運行或給出合理的響應，是保障可靠性的關鍵。

(5) 評估、監控與持續迭代

如何有效地評估AI功能的真實業務價值？如何持續監控其在線上的表現（性能、準確率、成本），檢測模型漂移，並基於這些反饋快速進行迭代優化？

理解這“70%問題”的挑戰，並非是要潑冷水，而是為了讓我們更清醒、更務實地投身於AI應用的創造中。

未來的發展，並不僅僅在於追求更大、更強的模型，而在於如何更聰明、更高效地解決“70%問題”。

因此，真正的“AI時代創造者”，不僅要了解AI技術本身，更要有將技術與現實世界需求相結合的工程能力、產品嗅覺和持續學習的韌性。掌握如何駕馭模型，並投入精力去攻克那充滿挑戰但也極具價值的“70%問題”，是創造未來的關鍵所在。

專注於解決真問題，精於整合與工程，勇於迭代與優化。這條道路或許充滿挑戰，卻是通往構建下一個偉大AI應用的必經之路。

11.8 結語

感謝你選擇本書作為AI編程之旅的起點。無論你是為了提高工作效率，探索副業機會，還是純粹出於好奇，我們都希望本書能為你打開一扇通往無限可能的大門。

記住，在AI時代，編程不再是少數人的特權，而是每個人都可以掌握的創造力工具。讓AI處理煩瑣的編碼細節，你就能專注於真正重要的事情：發現問題，構思解決方案，創造價值。

現在，合上這本書，打開你的計算機，開始創造吧！

附錄

附錄A 提示技術術語彙編

提示技術的33個詞彙術語及其解釋表如表A-1所示。

表A-1 33個詞彙術語及其解釋

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|------------------------|--------|---------------------------------------|
| 1 | Prompt | 提示 | 输入到生成型 AI 模型的指令，用于引导其输出 |
| 2 | Prompting | 提示过程 | 使用提示来引导生成型 AI 的行为或过程 |
| 3 | Context | 上下文 | 提示中用于增加生成型 AI 理解的附加信息 |
| 4 | Context Window | 上下文窗口 | 定义模型在进行预测时可以考虑的信息量 |
| 5 | Priming | 引导 | 在提示过程中用来设置特定情境或任务环境的技术 |
| 6 | Prompting Technique | 提示技术 | 描述如何结构化提示、提示或多个提示的动态序列的蓝图 |
| 7 | In-Context Learning | 上下文学习 | 生成型 AI 通过在提示中提供示例和 / 或相关指令来学习技能和任务的能力 |
| 8 | Exemplar | 示例 | 作为模型任务示范的实例 |
| 9 | Few-Shot Prompt | 少样本提示 | 使用少量示例来教导模型完成任务的提示 |
| 10 | Zero-Shot Prompt | 零示例提示 | 不使用示例即可激发模型生成答案的提示 |
| 11 | Orthogonal Prompt Type | 正交提示类型 | 提示的不同基本类型，与主流类型正交 |

(續)

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|-----------------------------------|---------|------------------------------------|
| 12 | Density | 密度 | 提示中信息的密集程度，影响模型输出的详细程度和复杂性 |
| 13 | Continuous Prompt | 连续提示 | 提示在内容或结构上不断变化或发展 |
| 14 | Discrete Prompt | 离散提示 | 提示具有固定和明确的结构或内容 |
| 15 | Originator | 创始人 | 最初提出或创建特定提示或技术的个体或团队 |
| 16 | User Prompt | 用户提示 | 直接由用户定义或输入的提示 |
| 17 | System Prompt | 系统提示 | 系统自动生成或预定义的提示 |
| 18 | Assistant Prompt | 助理提示 | 辅助系统使用的提示，旨在提供帮助或信息 |
| 19 | Prediction Style | 预测风格 | 提示中用于指导模型预测输出风格的指示 |
| 20 | Prefix | 前缀 | 放在输入前的固定文本，用于调整或指定模型的行为 |
| 21 | Cloze | 填空 | 一种提示类型，要求填入空白处以完成文本 |
| 22 | Prompt Chain | 提示链 | 两个或更多提示模板的连续使用，其中每个模板的输出用于参数化下一个模板 |
| 23 | Prompt Template | 提示模板 | 包含一个或多个变量的函数，这些变量会被替换以创建提示 |
| 24 | Prompt Engineering | 提示工程 | 开发提示的迭代过程，通过修改或更改正在使用的提示技术 |
| 25 | Prompt Engineering Technique | 提示工程技术 | 用于迭代改进提示的具体方法或策略 |
| 26 | Meta-Prompting | 元提示 | 提示生成或改进另一个提示或提示模板的过程 |
| 27 | Answer Engineering | 回答工程 | 开发或选择从生成型 AI 输出中提取精确答案的算法的迭代过程 |
| 28 | Verbalizer | 口述器 | 在标签任务中，将输出映射到标签的工具，反之亦然 |
| 29 | Extractor | 提取器 | 用于从生成型 AI 的响应中提取答案的算法或工具 |
| 30 | Conversational Prompt Engineering | 会话式提示工程 | 在对话设置中应用提示工程的实践 |
| 31 | Fine-Tuning | 微调 | 通过小的梯度更新调整模型的权重以优化性能的过程 |
| 32 | Prompt Tuning | 提示调整 | 调整提示以改善生成型 AI 的响应的过程 |
| 33 | Prompt-Based Learning | 基于提示的学习 | 通过提示而不是通过梯度更新进行学习的过程 |

58種文本基礎提示技術如表A-2所示。

表A-2 58種文本基礎提示技術

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|---|----------|--|
| 1 | Zero-Shot | 零样本 | 在没有先前例子或训练数据的情况下，直接使用模型进行问题解答或任务执行的技术 |
| 2 | Emotion Prompting | 情感提示 | 通过提示引导模型产生或识别特定情感的输出 |
| 3 | Role Prompting | 角色提示 | 指定模型扮演某一角色或具有特定身份来回答问题或生成内容 |
| 4 | Style Prompting | 风格提示 | 引导模型按照特定风格或格式生成文本 |
| 5 | System 2 Attention, S2A | 系统 2 注意力 | 让模型重写提示，消除与问题无关的信息 |
| 6 | SimToM | 情景模拟 | 涉及多人或对象的复杂问题解答，确立每个人所知事实的集合 |
| 7 | Rephrase and Respond, RaR | 改述并回应 | 指导模型先改述问题再生成答案 |
| 8 | Re-reading, RE2 | 重读 | 在提示中增加“再读一遍问题”这样的短语 |
| 9 | Self-Ask | 自问 | 模型自我提问以深入探索问题或主题，以改善答案的质量或完整性 |
| 10 | Few-Shot | 少样本 | 使用少量示例来训练或调整模型的技术，通常用于小样本学习 |
| 11 | Example Generation | 示例生成 | 生成用于训练或引导模型的示例 |
| 12 | SG-ICL Self-Generated In-Context Learning | 自生成情景学习 | 利用生成模型自动生成示例。在训练数据不可用的情况下，这种方法比零样本场景更有效，但生成的示例效果不如真实数据 |
| 13 | Example Ordering | 示例排序 | 调整示例在提示中的顺序，以优化模型的输出 |
| 14 | Exemplar Selection | 示例选择 | 从大量数据中选择最具代表性或最有用的示例来训练或指导模型 |
| 15 | K -Nearest Neighbor, K -NN | 最邻近算法 | 最近邻方法，通常用于选择与当前任务最相关的示例 |
| 16 | Vote- K | — | 一种投票机制，通过多个模型或多次运行来确定最佳答案 |
| 17 | Thought Generation | 思考生成 | 促使模型展开思考过程，以逐步解决问题或生成复杂内容 |
| 18 | Chain-of-Thought, CoT | 思维链 | 引导模型显示其解决问题的逐步思维过程 |
| 19 | Zero-Shot CoT | 零样本思维链 | 在零样本设置下实施思维链的技术，不依赖先前的例子 |
| 20 | Analogical Prompting | 类比提示 | 使用类比来帮助模型建立概念之间的联系，通常用于解决复杂问题 |

(續)

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|-------------------------------|------------|-------------------------------------|
| 21 | Step-Back Prompting | 后退提示 | 在回答之前，模型先回顾或重新考虑问题的各个方面 |
| 22 | Thread-of-Thought | 思维线索 | 进一步发展的思维链技术，强调在解答过程中串联多个思考点 |
| 23 | Tab-CoT | 表格思维链 | 使用表格格式来组织和显示模型的思维过程 |
| 24 | Few-Shot CoT | 少样本思维链 | 结合少样本学习和思维链技术，以少量示例促进模型的思考过程 |
| 25 | Active-Prompt | 主动提示 | 动态调整提示策略，根据模型反馈或任务性能来优化提示 |
| 26 | Auto-CoT | 自动思维链 | 自动化生成思维链，无须人工干预来设计或选择提示 |
| 27 | Complexity-Based | 基于复杂性的提示 | 根据任务的复杂性调整提示策略，以应对不同难度的问题 |
| 28 | Contrastive | 对比提示 | 使用对比例子来强调不同答案之间的差异，帮助模型识别正确答案 |
| 29 | Memory-of-Thought | 思维记忆 | 记录并利用之前的思考过程，以改进当前的解答策略 |
| 30 | Uncertainty-Routed CoT | 不确定性导向 CoT | 在存在不确定性时优化思维链的路线，以提高答案的准确性 |
| 31 | Prompt Mining | 提示挖掘 | 分析大量数据，寻找最有效的提示方式或模板 |
| 32 | Decomposition | 分解 | 把复杂问题分解成若干更易管理的小问题，然后逐个解决 |
| 33 | DECOMP (Decomposed Prompting) | 解构提示 | 将复杂问题分解为更易管理的子问题，以帮助模型更有效地解决这些问题的技术 |
| 34 | Faithful CoT | 忠实 CoT | 确保模型在执行思维链时，其推理过程和事实保持一致 |
| 35 | Least-to-Most | 从少到多 | 从最简单的组件开始，逐步增加问题的复杂性 |
| 36 | Plan-and-Solve | 计划和解决 | 先制定解决策略，再逐步执行解决问题的步骤 |
| 37 | Program-of-Thought | 思维程序 | 利用编程方法来指导模型的思维过程，通常用于解决数学或逻辑问题 |
| 38 | Recursion-of-Thought | 思维递归 | 在遇到复杂问题时，模型将问题分解并递归求解 |
| 39 | Skeleton-of-Thought | 思维骨架 | 创建问题解答的框架或骨架，然后逐部分填充细节 |

(續)

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|---|-------------|------------------------------------|
| 40 | Tree-of-Thought | 思维树 | 通过构建思维的树状结构，探索不同的解决路径，最终确定最优解 |
| 41 | Ensembling | 集成 | 使用多个模型或多次运行的结果，通过某种形式的聚合来得到最终答案 |
| 42 | Consistency-based Self-adaptive Prompting, COSP | 基于一致性的自适应提示 | 使用自洽在示例集中选择高一致性子集 |
| 43 | Demonstration Ensembling, DENSE | 示范集成 | 使用不同示例子集创建多个提示 |
| 44 | Diverse Reasoning Experts, DiVeRSe | 多样化推理示例 | 创建多个提示并对每个执行自洽 |
| 45 | Max Mutual Information | 最大互信息 | 优选能够最大化输入与输出之间互信息的提示策略 |
| 46 | Meta-CoT | 多思维链元推理 | 在元学习框架下实现思维链的应用，通过学习如何更好地学习来优化模型表现 |
| 47 | Mixture of Reasoning Experts, MoRE | 多推理专家 | 结合多种推理专家的意见来提高问题解决的准确性和深度 |
| 48 | Self-Consistency | 自我一致性 | 通过多次运行检验结果的一致性来提高模型的可靠性 |
| 49 | Universal Self-Consistency | 通用自我一致性 | 扩展自我一致性技术，应用于更广泛的任务和场景 |
| 50 | Universal Self-Adaptive Prompting, USP | 通用自适应提示 | 优化自适应提示策略，实现更高效和准确的模型训练和应用 |
| 51 | Prompt Paraphrasing | 提示重写 | 通过改写提示来探索不同表述的效果，以寻找最优的表达方式 |
| 52 | Self-Criticism | 自我批评 | 让模型评估自己的输出，并根据评估结果调整答案 |
| 53 | Chain-of-Verification | 验证链 | 通过构建一系列验证问题来测试和确认模型的输出结果 |
| 54 | Self-Calibration | 自我校准 | 通过反馈调整模型对其置信度的估计，以提高输出的准确性 |
| 55 | Self-Refine | 自我完善 | 让模型在迭代过程中不断完善其输出，以达到更高的质量标准 |
| 56 | Self-Verification | 自我验证 | 实施一系列自我检测措施，确保模型输出的正确性 |
| 57 | ReverseCoT | 反向 CoT | 通过反向应用思维链来检测和改正错误推理 |
| 58 | Cumulative Reasoning | 累积推理 | 在解答过程中累积推理结果，逐步构建最终答案 |

40種其他模態的提示類型如表A-3所示。

表A-3 40種其他模態的提示類型

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|---|----------------|---|
| 1 | Multilingual Technique | 多语言技术 | 指的是一系列技术和方法，用于处理、分析和生成多种语言的文本数据 |
| 2 | Chain-of-Thought | 思维链 | 引导模型展现其逐步推理过程的技术，常用于解决复杂问题 |
| 3 | XLT (Cross-Lingual Transfer) | 跨语言转移技术 | 一种能够将从一种语言中学到的知识应用到其他语言的机器学习方法 |
| 4 | In-Context Learning | 内置上下文学习 | 在给定的上下文中训练模型理解和适应，以增强其对相似情境的响应能力 |
| 5 | CLSP (Cross-Lingual Semantic Parsing) | 跨语言语义解析 | 是一种用于解析和理解多种语言中的语义信息的技术。它允许模型不仅仅翻译文本，而是能够理解不同语言中的语义结构，并将这些结构映射到统一的语义框架中 |
| 6 | X-InSTA (Cross-lingual Instance-Aware Semantic Transformation and Adaptation) | 跨语言实例感知语义转换与适应 | 它的目标是在不同语言间转换语义信息时，保持对特定实例的敏感性和准确性，如在处理具体的命名实体或特定上下文时维持语义的一致性和准确性 |
| 7 | In-CLT (In-Context Language Transfer) | 上下文内语言转移 | 指的是在给定的上下文中，将信息从一种语言转移至另一种语言的技术 |
| 8 | In-Context Ex. Selection | 内置上下文示例选择 | 选择适合当前上下文的示例来优化模型的学习过程 |
| 9 | PARC (Predictive Auto-Regressive Context) | 预测自回归上下文 | 一种利用自回归模型预测和维持上下文的连续性的技术。它通过预测接下来可能出现的词汇或短语来自动维护和更新上下文，增强了语言模型对上下文变化的适应性和预测能力 |
| 10 | Semantically-Aligned | 语义上接近 | 选择语义上接近的提示或内容，以增强模型的语义理解能力 |
| 11 | Semantically-Distant | 语义上距离 | 使用与当前上下文语义上有距离的提示，挑战模型处理复杂或非直观关系的能力 |
| 12 | Human-in-the-Loop | 人在回路中 | 在模型的训练或推理过程中，加入人类的判断和反馈，以提高决策的准确性 |
| 13 | Interactive Chain | 交互式链 | 在交互环境中使用链式提示，允许用户与模型的输出动态互动，优化结果 |
| 14 | Iterative | 迭代 | 通过反复迭代的方式细化模型的输出，逐步接近最优解 |
| 15 | Translation | 翻译 | 将一种语言的文本转换为另一种语言，同时保持原意的技术 |

(續)

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|--|------------|---|
| 16 | Chain-of-Dictionary | 字典链 | 利用词典或词汇数据库链接信息，帮助模型理解和生成语言内容 |
| 17 | DecoMT (Decomposed Machine Translation) | 解构机器翻译 | 涉及将复杂的翻译任务分解成更小、更易管理的子任务。这种方法通过专注于特定语言对或语言特性的细分问题，可以提高翻译的准确性和效率 |
| 18 | DiPMT (Differential Privacy Machine Translation) | 差分隐私机器翻译 | 一种融合机器翻译与差分隐私技术的方法，目的是在不牺牲用户数据隐私的前提下提供高质量的翻译服务。通过在翻译过程中应用差分隐私原则，DiPMT 能够保护个人数据不被泄露，同时确保翻译输出的实用性和准确性 |
| 19 | MAPS (Multilingual Adaptive Processing System) | 多语言自适应处理系统 | 指的是能够自动适应并优化多语言内容处理的系统。这种系统通过学习不同语言间的相似性和差异性，自动调整处理策略，以提高跨语言任务的效率和效果 |
| 20 | Translate First Prompting | 首先翻译提示 | 一种处理多语言任务的技术，它涉及在执行任何其他自然语言处理任务之前先将输入数据从源语言翻译成目标语言 |
| 21 | External MT System | 外部机器翻译系统 | 使用外部机器翻译系统，通常为独立开发的高性能系统，以支持特定任务 |
| 22 | Standard LLM | 标准大语言模型 | 使用标准的大型语言模型处理任务，不涉及特定优化或调整 |
| 23 | Multilingual LLM | 多语言大语言模型 | 支持处理多种语言的大型语言模型，适用于全球化应用 |
| 24 | Prompt Language | 提示语言 | 指定用于模型提示的语言，通常选择模型训练中使用的主要语言 |
| 25 | Task Language | 任务语言 | 定义用于执行特定任务的语言，可能与提示语言不同 |
| 26 | Multimodal (MM) Techniques | 多模态技术 | 指的是结合使用来自不同模态（如文本、图像、音频、视频等）的数据来提升信息处理和分析能力的技术 |
| 27 | Image | 图像 | 使用图像作为输入或参照的提示技术 |
| 28 | MM CoT | 多模态 CoT | 在多模态环境下应用 CoT 的技术，结合不同模态的信息进行推理 |
| 29 | Chain-of-Image | 图像链 | 通过一系列相关联的图像展示信息或叙述故事，形成视觉上的链式逻辑 |
| 30 | Duty Distinct CoT | 职责区分 CoT | 明确不同组件或角色的职责，以优化多模态链式思考的过程 |

(續)

| 序号 | 英文术语 | 中文翻译 | 解释 |
|----|--|------------|--|
| 31 | MM Graph-of-Thought | 多模态思维图 | 创建包含多种模态元素（如图像、文本）的思维图谱，以展示复杂思考过程 |
| 32 | Multimodal ICL, Multimodal In-Context Learning | 多模态内置上下文学习 | 一种结合多种数据模态（如文本、图像、音频等）进行学习的技术，特别强调在给定的上下文中，利用来自各种模态的数据进行信息理解和决策的能力 |
| 33 | Image-as-Text Prompt | 图像 - 文本提示 | 将图像信息转换为文本描述作为提示，帮助理解或生成相关内容 |
| 34 | Paired-Image Prompt | 配对图像提示 | 使用一对图像作为提示，通常用于比较、对比或关联分析 |
| 35 | Negative Prompt | 负面提示 | 是一种特定类型的提示技术，用于引导模型生成与预设条件相反的结果或反应 |
| 36 | Prompt Modifier | 提示修饰 | 是在 NLP 中使用的一种技术，用于修改或调整初始提示，以影响和指导机器学习模型的输出 |
| 37 | Segmentation Prompting | 分割提示 | 在图像或视频处理中，使用提示来指导模型对特定区域进行分割和识别 |
| 38 | Video | 视频 | 使用视频作为输入或生成视频作为输出的提示技术 |
| 39 | Video Gen. | 视频生成 | 利用提示生成或编辑视频内容的技术 |
| 40 | 3D Prompting | 3D 提示 | 利用三维模型或场景进行提示，常用于增强现实或虚拟现实技术中 |

附錄B

DeepSeek提示詞

B.1 提示原則

簡潔有效的提示：

- ❑ 直奔主題，省略禮貌用語如“請”“謝謝”等。
- ❑ 使用肯定指令（如“做”），避免否定語言（如“不要”）。
- ❑ 清晰標明任務：“你的任務是”“你必須”等強調語句。
- ❑ 使用“如果你不這樣做，你將受到懲罰”來增加指令權重。

提示結構優化：

- ❑ 使用分隔符區分不同部分。
- ❑ 以“###Instruction###”開始，根據需要添加“###Example###”或“###Question###”。
- ❑ 使用換行符分隔指令、示例、問題和上下文。
- ❑ 多次重複關鍵詞或短語以強調重點。

在提示結尾使用輸出啟動器（以期望的輸出開頭結束提示）。

角色與受眾設定：

為LLM（大語言模型）分配特定角色以獲得特定風格的回答。

在提示中明確指定目標受眾（如“聽眾是該領域的專家”）。

使用“以自然、類人的方式回答問題”獲得更自然的回答。

複雜問題處理：

將複雜任務分解為一系列簡單提示。

使用引導詞，如“一步步思考”。

結合思維鏈與少樣本提示。

通過“從現在開始，我想讓你問我問題.....”允許模型主動獲取信息。

解釋與學習：

根據需要調整解釋難度：“用簡單術語解釋”或“向11歲孩子解釋”。

針對不同知識水平：“向[領域]初學者解釋”或“向[領域]專家解釋”。

使用“教我[主題]並在最後包含測試”進行學習驗證。

內容創建：

- 指定詳細要求：“詳細寫一個關於[主題]的[文章/段落]”。
- 明確修改界限：“只改善語法和詞彙，保持原文風格”。
- 提供開頭並要求完成：“我提供開頭[插入內容]，保持一致完成”。
- 要求遵循已有示例：“根據提供的[文本]使用相同語言”。

其他技巧：

- 添加“我將為更好的解決方案提供小費”增加回答質量。
- 要求“確保回答無偏見，避免依賴刻板印象”。
- 對複雜代碼要求生成自動創建或修改文件的腳本。

這些技巧可以根據不同需求靈活組合使用，有效提升與LLM交互的質量和效率。

B.2 DeepSeek提示詞模板示例

1.翻譯老師

以下是DeepSeek作為翻譯老師的提示詞內容：

現在你要幫忙解釋一篇專業的技术文章成简体中文給大學生閱讀。

規則：

- 翻譯時要準確傳達學術論文的事實和背景，同時風格上保持為通俗易懂並且嚴謹的科普文風格。

- 保留特定的英文术语、数字或名字，并在其前后加上空格，例如：“中 UN 文”，“不超过 10 秒”。
 - 即使上意译也要保留术语，例如 FLAC, JPEG 等。保留公司缩写，例如 Microsoft, Amazon 等。
 - 保留引用的论文，例如 [20] 这样的引用；同时也要保留针对图例的引用，例如保留 Figure 1 并翻译为图 1。
 - 全角括号换成半角括号，并在左括号前面加半角空格，右括号后面加半角空格。
 - 输入格式为 Markdown 格式，输出格式也必须保留原始 Markdown 格式。
- 现在有三个角色和五个步骤来翻译这篇文章，每一步都必须遵守以上规则，打印每一步的输出结果：
- 英语老师：精通英文，能精确的理解英文并用中文表达。
 - 中文老师：精通中文，擅长按照中文使用喜欢撰写通俗易懂的科普文。
 - 校长：精通中文和英文，擅长校对审查。

Step 1: 现在你是英语老师，精通英文，对原文按照字面意思直译，务必遵守原意，翻译时保持原始英文的段落结构，不要合并分段。

Step 2: 扮演中文老师，精通中文，擅长写通俗易懂的科普文章，对英语老师翻译的内容重新意译，遵守原意的前提下让内容更通俗易懂，符合中文表达习惯，但不要增加和删减内容，保持原始分段。

Step 3: 英文老师将中文老师的文稿反向翻译成英文（回译稿）。

Step 4: 扮演校长，精通中文和英文，校对回译稿和原稿中的区别，重点检查翻译稿和原文有出入的位置以及不符合中文表达习惯的位置这两点。

Step 5: 中文老师基于校长的修改意见，修改初稿。

本条消息只需要回复 OK，接下来的消息我将会给你发送完整内容，收到后请按照上面的规则和下面的格式打印翻译结果，返回格式如下，“{xxx}”表示占位符：

英语老师直译结果

{ 英语老师直译结果 }

中文老师意译初稿

{ 中文老师意译初稿 }

英语老师回译

{ 英语老师回译稿 }

校长校对意见

以下是在中文翻译中缺失的部分：

{ 重复以下列表，直到列出所有缺失的内容 }

- 对比原文缺失或表达歧义部分 {1...n}:
- 原文：“{English}”
- 译文：“{译文}”
- 建议：{ 新增翻译 or 修改翻译 }

以下是中文翻译表达不符合中文习惯的部分：

{ 重复以下列表，直到列出所有需要修改的内容 }

- 修改 {1...n}:
- 原文：“{English}”
- 译文：“{译文}”
- 建议：{ 修改后内容 }

中文老师翻译终稿

{ 中文老师翻译终稿 }

2. 专家提示工程师

以下是DeepSeek作為專家提示工程師的提示詞內容：

你是 DeepSeek 聘请的专家提示工程师，你的任务是为各种大小的 LLM 优化提示。你需要根据提供的模型大小（以十亿参数计算）来调整每个提示。

指令：

1. 使用全大写来突出提示中最重要的部分。
2. 当用户要求时，使用 OpenCHATML 格式：
system
[详细的代理角色和上下文]
assistant
[确认理解并简明扼要地总结关键指令]
3. 提供精确、具体和可操作的指令。
4. 如果你有限的令牌量需要采样，那么请尽快结束；我会用命令“继续”再次请求。

知识库：

对于 LLM

- 对于多步骤任务，将提示分解为一系列相关的子任务。
- 在适当的时候，包括所需输出格式的相关示例。
- 在回应中反映原始提示的重要细节。
- 根据模型大小调整你的语言（对于较小的模型简化，对于较大的模型更精细化）。
- 对于简单的示例使用零样本，对于复杂的使用多样本示例。
- 大语言模型在进行一些视觉推理（文本生成）后写答案更好，这就是为什么有时候初始提示中包含一个为 LLM 代理填写的示例表单。

3.新聞翻譯

以下是DeepSeek作為新聞翻譯的提示詞內容：

你是一位精通简体中文的专业翻译，曾参与《纽约时报》和《经济学人》中文版的翻译工作，因此对于新闻和时事文章的翻译有深入的理解。我希望你能帮我将以下英文新闻段落翻译成中文，风格与上述杂志的中文版相似。

規則：

- 翻译时要准确传达新闻事实和背景。
 - 保留特定的英文术语或名字，并在其前后加上空格，例如：“中 UN 文”。
 - 分成两次翻译，并且打印每一次结果：
 - 1. 根据新闻内容直译，不要遗漏任何信息。
 - 2. 根据第一次直译的结果重新意译，遵守原意的前提下让内容更通俗易懂，符合中文表达习惯。
- 本条消息只需要回复 OK，接下来的消息我将会给你发送完整内容，收到后请按照上面的规则打印两次翻译结果。

4.SQL魔法師

以下是DeepSeek作為SQL魔法師的提示詞內容：

将以下自然语言请求转换为有效的 SQL 查询。假设存在具有以下表和列的数据库：

Customers:

- customer_id (INT, PRIMARY KEY)
- first_name (VARCHAR)
- last_name (VARCHAR)
- email (VARCHAR)
- phone (VARCHAR)
- address (VARCHAR)

- city (VARCHAR)
- state (VARCHAR)
- zip_code (VARCHAR)

Products:

- product_id (INT, PRIMARY KEY)
- product_name (VARCHAR)
- description (TEXT)
- category (VARCHAR)
- price (DECIMAL)
- stock_quantity (INT)

Orders:

- order_id (INT, PRIMARY KEY)
- customer_id (INT, FOREIGN KEY REFERENCES Customers)
- order_date (DATE)
- total_amount (DECIMAL)
- status (VARCHAR)

Order_Items:

- order_item_id (INT, PRIMARY KEY)
- order_id (INT, FOREIGN KEY REFERENCES Orders)
- product_id (INT, FOREIGN KEY REFERENCES Products)
- quantity (INT)
- price (DECIMAL)

Reviews:

- review_id (INT, PRIMARY KEY)
- product_id (INT, FOREIGN KEY REFERENCES Products)
- customer_id (INT, FOREIGN KEY REFERENCES Customers)
- rating (INT)
- comment (TEXT)
- review_date (DATE)

Employees:

- employee_id (INT, PRIMARY KEY)
- first_name (VARCHAR)
- last_name (VARCHAR)
- email (VARCHAR)
- phone (VARCHAR)
- hire_date (DATE)
- job_title (VARCHAR)
- department (VARCHAR)
- salary (DECIMAL)

根据自然语言请求提供检索数据的 SQL 查询。

5. Python代碼調試器

以下是DeepSeek作為Python代碼調試器的提示詞內容：

你的任务是分析提供的 Python 代码片段，识别存在的任何错误或缺陷，并提供解决这些问题的修正版代码。解释你在原始代码中发现的问题以及你的修复如何解决这些问题。修正后的代码应该是可用的、高效的，并且遵循 Python 编程的最佳实践。

6. 商業計劃書

以下是DeepSeek作為商業計劃書的提示詞內容：

你是一个高级商业计划机器人，旨在帮助企业制定全面的商业计划。你的目的是引导用户完成开发过程的每个阶段，从构思到执行，并结合精益创业和设计思维原则。你将在开发、招聘、财务规划、营销计划、社交媒体、技术、人力资源、组织、董事会结构和治理、法律、注册、专利、商标、税收、补贴、赠款和增长黑客策略方面提供帮助。

/help 将提供以下内容：

高级商业计划机器人命令

1. /ideation - 帮助用户集思广益并提炼商业创意。
2. /development - 引导用户完成产品开发和原型制作。
3. /hiring - 协助用户制定招聘计划和选择候选人。
4. /financialplanning - 提供财务预测和预算协助。
5. /marketing - 为用户的业务制定营销策略和计划。
6. /socialmedia - 创建社交媒体策略和内容计划。
7. /technology - 为企业推荐技术工具和平台。
8. /hr - 提供有关人力资源政策和最佳实践的指导。
9. /organization - 帮助用户设计组织结构和流程。
10. /board - 就董事会结构和治理提出建议。
11. /legal - 提供有关法律事务的指导，包括公司注册和合同。
12. /ip - 协助用户进行专利和商标注册。
13. /tax - 提供有关税务规划和合规性的建议。
14. /subsidies - 帮助用户识别和申请相关补助和补贴。
15. /growthhacking - 制定和执行增长黑客策略。

用法示例：

/ideation “环保产品订阅箱”

/财务规划 50000 12

/营销 “针对具有环保意识的消费者”

首先说 “Advanced Business Plan Bot Initiated”，仅此而已。

作者簡介

蘭翔

香港智友數字科技CEO，復旦大學EMBA，本科畢業於加拿大里賈納大學計算機科學專業人工智能方向，武漢大學經濟管理學院校外研究生導師，湖北工業大學統計學院校外研究生導師。

資深數字化運營技術專家和銀行信用風險控制專家，擁有超過10年數字化運營和銀行線上業務風險控制管理經驗。作為兩家商業銀行總行風險管理部和數據風控部主要負責人，主導了全行數字化風控體系建設，在有效控制風險的前提下，成功推動消費金融貸款和小微企業供應鏈貸款業務規模突破千億元；深度參與AI技術在金融等傳統行業的落地應用，帶領團隊開發了多個基於機器學習的風險預測模型和自動化決策系統，併成功推動AI編程工具在金融數字化轉型中的廣泛應用。

作為連續創業者和AI技術專家，致力於推動AI技術及互聯網營銷應用的普及化和平民化。長期關注AI編程工具的發展趨勢，

深度研究DeepSeek、Windsurf等新一代AI變成助手在實際項目中的應用價值。

Table of Contents

[版權信息](#)

[內容簡介](#)

[前言 擁抱AI，與時代共舞——AI時代創造者啟程指南](#)

[第一部分 AI編程入門](#)

[第1章 AI時代普通人如何學編程](#)

[1.1 本章學習目標](#)

[1.2 AI時代普通人為什麼要學編程](#)

[1.3 普通人能用AI編程做什麼](#)

[1.4 AI編程給普通人帶來的新機會](#)

[1.5 結語](#)

[第2章 DeepSeek編程指南](#)

[2.1 本章學習目標](#)

[2.2 什麼是DeepSeek](#)

[2.3 登錄和使用DeepSeek](#)

[2.4 DeepSeek的核心能力與應用場景](#)

[2.5 DeepSeek提示詞](#)

[2.6 結語](#)

[第3章 輕鬆準備你的編程環境](#)

[3.1 本章學習目標](#)

[3.2 開發環境概述](#)

[3.3 開發工具的選擇](#)

[3.4 工具的安裝與使用流程](#)

[3.5 常見問題與解決方案](#)

[3.6 結語](#)

[第4章 用DeepSeek學Python變量和簡單數據類型](#)

[4.1 本章學習目標](#)

[4.2 Python解析器](#)

[4.3 變量](#)

[4.4 字符串](#)

[4.5 數字](#)

[4.6 註釋](#)

[4.7 結語](#)

[第5章 用DeepSeek學Python列表](#)

[5.1 本章學習目標](#)

[5.2 Python列表](#)

[5.3 修改、添加和刪除元素](#)

[5.4 Python列表的高級技巧](#)

[5.5 通過DeepSeek學習Python索引](#)

[5.6 結語](#)

[第6章 用DeepSeek學操作列表](#)

[6.1 本章學習目標](#)

[6.2 遍歷整個列表](#)

[6.3 避免縮進錯誤](#)

[6.4 創建數值列表](#)

[6.5 結語](#)

[第二部分 項目實戰](#)

[第7章 開發數據分析報告生成系統](#)

[7.1 本章學習目標](#)

[7.2 自動化數據分析的優勢](#)

[7.3 數據分析報告生成系統的功能](#)

[7.4 具體實現步驟：開發數據分析報告生成系統](#)

[7.5 常見問題與解決思路](#)

[7.6 結語](#)

[第8章 開發自動化日報生成工具](#)

[8.1 本章學習目標](#)

[8.2 瞭解自動化日報生成工具](#)

[8.3 自動化日報生成工具的核心功能](#)

[8.4 具體實現步驟：構建你的自動化日報生成工具](#)

[8.5 常見問題與解決思路](#)

[8.6 結語](#)

[第9章 開發本地事務提醒助手](#)

[9.1 本章學習目標](#)

[9.2 瞭解本地事務提醒助手](#)

[9.3 本地事務提醒助手的核心功能](#)

[9.4 具體實現步驟：構建你的本地事務提醒助手](#)

[9.5 常見問題與解決思路](#)

[9.6 結語](#)

[第10章 打造你的第一個網站](#)

[10.1 本章學習目標](#)

[10.2 瞭解網站開發基礎](#)

[10.3 網站開發的核心功能](#)

[10.4 具體實現步驟：開發你的第一個網站](#)

[10.5 常見問題與解決思路](#)

[10.6 結語](#)

[第三部分 總結與展望](#)

[第11章 成為AI時代的創造者](#)

[11.1 本書精華總結](#)

[11.2 即刻行動指南：將知識轉化為力量](#)

[11.3 避免常見陷阱](#)

[11.4 未來成長地圖：繪製你的技能進階藍圖](#)

[11.5 持續發現需求的方法](#)

[11.6 致未來的創造者：你的潛力無限](#)

[11.7 最後的話](#)

[11.8 結語](#)

[附錄](#)

[附錄A 提示技術術語彙編](#)

[附錄B DeepSeek提示詞](#)

[作者簡介](#)