AI编程与智能体开发

林子雨　副教授　厦门大学
E-mail: ziyulin@xmu.edu.cn

2026年9月

主讲教师

- 2013年度厦门大学奖教金获得者
- 2017年度厦门大学奖教金获得者
- 2020年度厦门大学奖教金获得者
- 2023年度厦门大学奖教金获得者
- 2026年度厦门大学奖教金获得者
- 2018年国家精品在线开放课程
- 2020年国家级线上一流本科课程
- 2023年国家级线上一流本科课程
- 2020年福建省线上一流本科课程
- 2021年福建省线上线下混合一流本科课程
- 2018年福建省高等教育教学成果奖二等奖
- 2022年福建省高等教育教学成果奖特等奖
- 2024年福建省高等教育教学成果奖特等奖
- 2023年教育部国家智慧教育公共服务平台应用典型案例

配套教材

教材目录

第1章 大模型
第2章 AI编程概述
第3章 AI编程环境搭建
第4章 基于提示词的AI编程
第5章 基于规范的AI编程
第6章 流程驱动的AI编程
第7章 智能体概述
第8章 智能体开发框架LangChain
第9章 智能体开发框架LangGraph

教材特色

- 教材内容来自互联网大厂真实实践
- 以实践教学为依托
- 理论实践贯通
- 覆盖主流技术

《AI编程与智能体开发》

林子雨 编著

人民邮电出版社 预计2026年11月上市销售

教材官网提供详细信息和样书申请

官网： https://dblab.xmu.edu.cn/post/ai-programming-agent/

扫码访问教材官网
了解详情、获取资源、申请样书

第3章

AI编程环境搭建

目录

Part 01

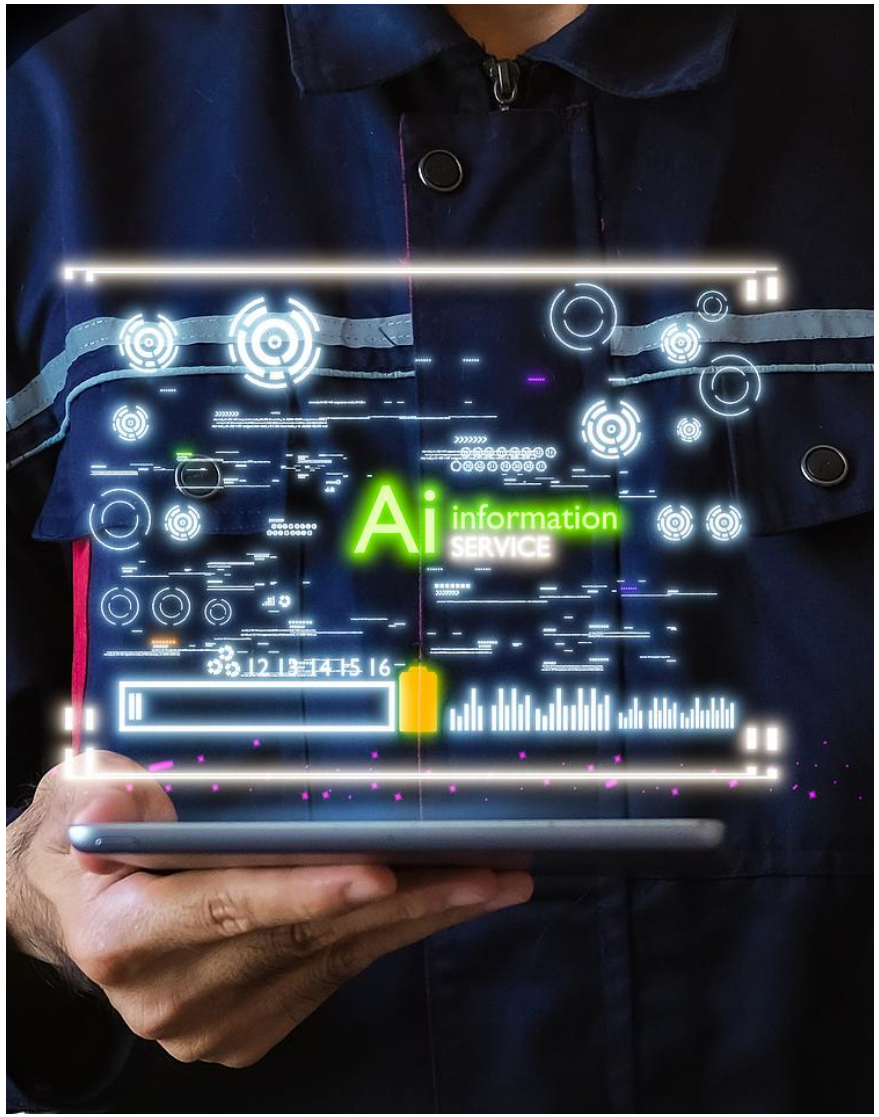
AI编程环境概述

3.1 AI编程环境概述

AI编程环境是指支持程序编写、运行、调试和依赖管理的一整套标准化工作条件，是连接开发者与AI大模型能力的核心桥梁，也是**开展AI编程开发、完成项目落地的基础前提**。

它并非单一的工具或软件，而是由多个功能模块相互配合、协同作用形成的完整体系，通常包括程序解释器（如Python解释器）、第三方包管理工具（如pip）、规范化的项目目录结构、代码编辑器或集成开发环境（IDE，如VS Code、Trae）、AI大模型以及必要的调试、测试工具等核心组成部分。

AI编程环境的核心作用，在于将AI大模型的生成、推理能力有效接入程序开发流程之中，使开发者能够通过编写代码的方式，精准控制AI请求的内容、参数设置、返回结果的处理逻辑以及各类异常情况的应对方式，从而**实现AI能力与具体项目需求的深度结合**。



3.1 AI编程环境概述

对于本书后续的实验操作和项目实践而言，搭建一套合格的AI编程环境，至少需要具备三个核心条件：

可正常运行的Python运行环境	这是多数AI编程项目的基础运行载体，确保代码能够被正确解析和执行；
便于编写、编辑和调试程序的集成开发环境	它能够提供代码高亮、自动补全、实时调试等功能，大幅提升开发效率，降低调试难度；
能够隔离项目依赖的软件管理方式	通过虚拟环境等工具，避免不同项目的依赖包版本冲突，保障项目运行的稳定性和可复现性。

Part 02

Python虚拟环境

3.2 Python虚拟环境

Python虚拟环境用于解决不同项目之间的依赖隔离问题。实际开发中，两个项目往往需要不同版本的第三方库。如果所有库都安装在系统全局环境中，就容易出现版本冲突，从而影响程序运行。虚拟环境的基本思想是：为每个项目建立相对独立的Python解释器和包安装目录，使项目之间互不干扰。

在本书的实验中，建议为每一个AI编程项目单独创建虚拟环境。这样做有两个好处：一是便于管理依赖关系，二是便于在项目迁移、实验复现和多人协作时保持一致的运行条件。

3.2 Python虚拟环境

在Windows系统的cmd命令行界面中，进入项目所在目录后，可以执行以下命令创建虚拟环境：

```
python -m venv ai-env
```

上述命令会在当前目录下生成一个名为 ai-env 的目录，其中包含独立的Python运行环境（名字为ai-env）。创建完成后，需要根据操作系统激活该环境。激活命令如下：

```
ai-env\Scripts\activate
```

激活成功后，终端提示符前通常会出现 (ai-env) 标记。此时，通过 pip install 安装的第三方库都会进入该虚拟环境，而不会影响系统中的其他Python项目。

在安装本章所需依赖之前，建议先升级 pip：

```
pip install --upgrade pip
```

当本次工作结束后，可执行以下命令退出虚拟环境：

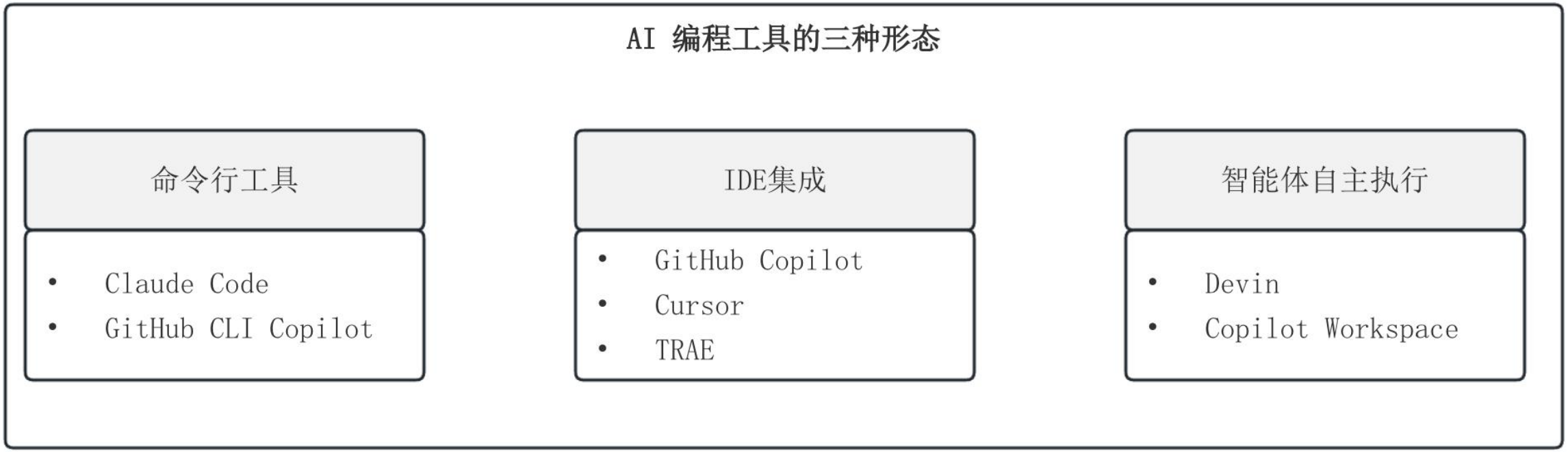
```
deactivate
```

Part 03

AI编程工具的三种形态

3.3 AI编程工具的三种形态

当前市场上的AI编程工具，形态各异，但本质上都是同一套逻辑——让人类开发者与大语言模型进行交互，辅助完成软件开发。根据交互方式的不同，可以将这些工具分为三种形态（如图所示）：命令行工具、与IDE集成、智能体自主执行。



3.3.1命令行工具

命令行工具是最基础的AI编程工具形态。顾名思义，这类工具运行在终端中，开发者通过文本命令与AI进行交互。

代表

Claude Code（Anthropic推出）、GitHub CLI Copilot扩展、OpenAI Codex CLI等。它们的共同特点是轻量、灵活、与现有开发工作流紧密结合。

优势

与开发者日常使用的终端环境无缝集成，无需切换到其他应用；占用系统资源少，对机器性能要求低；可以通过脚本自动化，适合集成到CI/CD流程中。

局限

缺乏图形界面，交互体验相对单一；对于不熟悉命令行的开发者存在上手门槛；不适合需要可视化展示效果的场景。

适合场景

服务器端开发、需要与Git等工具配合使用的工作流、追求极致效率的专业开发者。

3.3.2 与IDE集成

与IDE（集成开发环境）集成是目前最主流的AI编程工具形态。这类工具深度嵌入到开发者常用的代码编辑器中，提供代码补全、对话交互、代码审查等丰富功能。

代表

GitHub Copilot（集成于Visual Studio Code、JetBrains系列IDE）、Cursor（独立的AI优先IDE）、TRAE（字节跳动推出的AI IDE）、Amazon CodeWhisperer（支持多种IDE）等。

优势

与代码编辑环境深度融合，开发者无需切换上下文；提供丰富的图形交互（如下拉补全、对话框、代码高亮），使用体验友好；能够访问项目文件结构，提供更准确的上下文理解。

局限

对IDE本身存在依赖，不同IDE之间切换成本较高；功能受限于IDE的扩展机制；部分高级功能需要订阅付费版本。

适用场景

日常开发工作、界面开发、需要频繁查看和修改多个文件的复杂项目。

3.3.3智能体自主执行

智能体（Agent）形态代表了AI编程工具的前沿方向。这类工具不仅能够响应开发者的指令，还能够自主规划任务步骤、调用多种工具、执行复杂的多步骤操作。

代表

Devin（Cognition AI推出，被称为"第一个AI软件工程师"）、GitHub Copilot Workspace、Cline等。它们的共同特点是能够自主完成从需求理解到代码交付的完整流程。

优势

能够承担更复杂的任务，减少人类开发者需要关注的细节；自动化程度高，开发者可以将更多精力投入到需求定义和结果审核；理论上具备24小时不间断工作的能力。

局限

技术成熟度相对较低，可靠性有待验证；对于边界模糊或需要高度判断力的任务处理能力有限；自主执行可能带来安全风险（如未经授权修改代码）。

适用场景

需要多步骤协作的复杂项目、探索性开发、对自动化程度要求高的工作流。

Part 04

AI编程工具选型指南

3.4 AI编程工具选型指南

主流工具概述

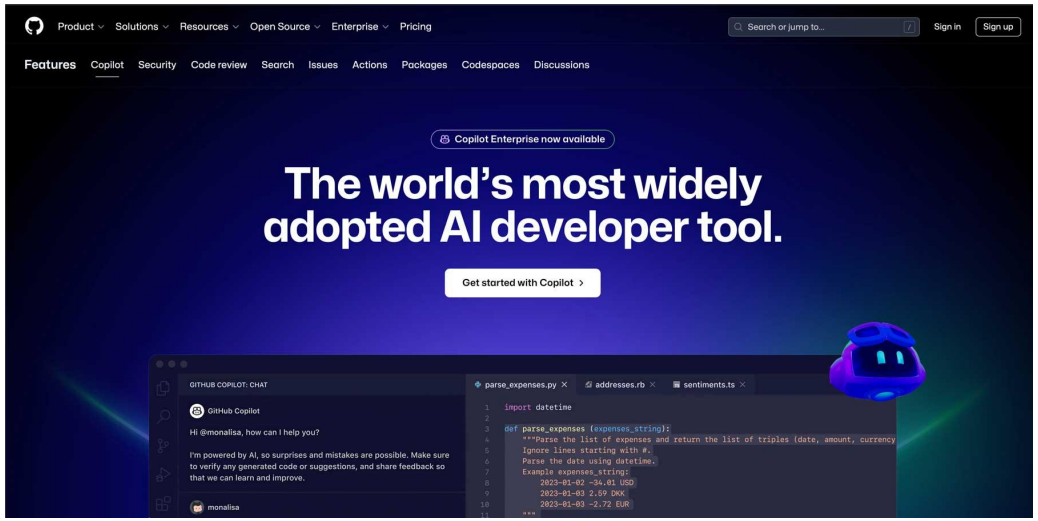
工具选择建议

3.4.1主流工具概述

当前市场上的AI编程工具呈现"同质化"特征，各家产品在功能层面差距不大。

GitHub Copilot凭借微软和GitHub的生态优势，拥有最广泛的用户基础；Cursor以创新性的产品设计吸引了大量专业开发者；Claude Code在长上下文处理和复杂推理方面表现突出；TRAE针对中文开发者优化，提供本地化体验。这些工具的核心能力——代码补全、对话生成、代码审查，几乎已经成为标配。

选择工具时，开发者无需过于纠结功能的细微差异。更值得关注的是，工具对特定编程语言的支持程度、与个人工作流的契合度、以及生态社区的活跃度。



3.4.2工具选择建议

综合考虑学习曲线、功能丰富度、社区生态等因素，这里我们选择TRAE作为AI编程工具。TRAE是一款面向中文开发者的AI优先IDE，由字节跳动开发，提供Chat模式、Agent模式和SOLO模式三种工作方式，能够满足从简单辅助到复杂项目开发的多样需求。

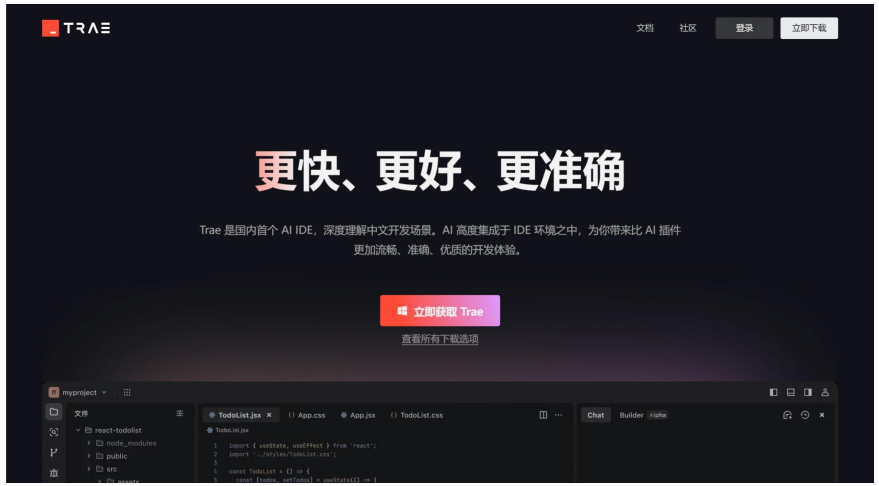
选择TRAE的理由主要包括以下几个方面：

TRAE针对中文开发者进行了深度优化。界面语言、文档资料、技术支持都以中文为主，降低了中国开发者的学习门槛。

Chat模式适合日常问答和代码理解；Agent模式适合需要多轮迭代的复杂任务，本书的编程实践将统一使用Agent模式；SOLO模式代表了AI更高自主性的探索方向，但目前仍在完善中。

TRAE的国内访问速度稳定，并且当前免费。

TRAE背后是字节跳动的技术积累。字节跳动在国内互联网企业中以技术能力著称，其AI产品在国内开发者中有较好的口碑。



Part 05

TRAE的安装与使用

3.5 TRAE的安装与使用

- TRAE概述
- TRAE的下载与安装
- TRAE的两种工作空间模式
- TRAE的三种智能体交互执行模式
- TRAE开发实例

3.5.1 Trae概述

TRAE是字节跳动推出的AI原生集成开发环境（IDE），基于VS Code内核构建，深度集成大语言模型能力，旨在为开发者提供从需求分析、项目搭建、编码调试到部署上线的全流程智能化支持。

TRAE的定位是“AI原生IDE”，这与传统IDE有着本质区别，具体如表所示。

维度	传统IDE	AI原生IDE
核心范式	人类编写，工具辅助	人机协作，AI主导
交互方式	键盘输入为主	自然语言对话为主
项目启动	手动搭建框架	Agent模式自动生成
代码编写	逐行手写	智能生成+人工审查
问题解决	人工搜索+调试	AI对话诊断修复

3.5.2 Trae的下载与安装

TRAE支持Windows、macOS和Linux三大操作系统。本书采用Windows系统，因此，这里以Windows系统为例，介绍下载和安装流程，具体如下：

第一步：访问官方网站

打开浏览器，访问TRAE官方网站：https://www.trae.cn/ 。点击右上角"下载"按钮，进入下载页面。

第二步：选择版本

选择TRAE IDE的Windows版本。注意，本书使用的是TRAE的IDE版本而不是SOLO版本。

第三步：运行安装程序

下载完成后，双击安装包文件，启动安装向导。按照屏幕提示完成安装，接受许可协议、选择安装位置、创建桌面快捷方式等步骤。安装过程通常需要3至5分钟。安装完成后，TRAE会自动启动。

第四步：登录账号

首次启动TRAE时，需要登录账号。如果已有TRAE账号，直接输入用户名和密码登录；如果没有账号，可以选择用邮箱注册新账号。登录成功后，TRAE会引导用户完成初始配置，包括选择主题颜色、配置默认编程语言等。

3.5.3 TRAE的两种工作空间模式

TRAE提供了两种工作空间模式：IDE模式和SOLO模式，在TRAE工作界面的左上角，有一个工作空间模式切换按钮（如图所示），点击该按钮就可以在两种模式之间来回切换。

3.5.3 TRAE的两种工作空间模式

两种工作空间模式的具体区别如下：

IDE模式

它是面向代码开发者的“专业编码工作台”，保留了传统代码编辑器的所有核心能力，比如语法高亮、断点调试、版本控制联动，同时，把AI能力嵌入到编码的每个环节——你写函数时它补全逻辑、报错时自动定位修复、甚至能按你的注释生成可运行的代码块。这种模式适合你全程主导、需要精细控制代码细节的开发场景，比如，初学者学习基础编程时，用这个模式可以让学习者聚焦代码逻辑本身，同时，借助AI降低语法试错成本。

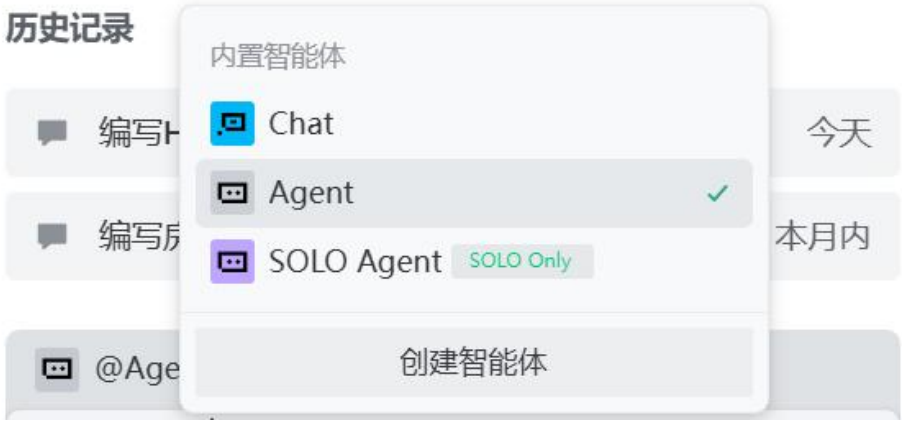
VS

SOLO模式

它是面向“任务目标”的智能体项目空间，你不用再逐行写代码，只要用自然语言描述需求，比如“写一个学生成绩统计的Python脚本，要求支持Excel导入和平均分排序”，SOLO模式下的智能体就会自动拆解任务、创建文件结构、完成编码、自测并输出可运行的项目包。这个模式适合学习者了解AI工程化思维时使用，可以让学习者理解“从需求到交付”的完整流程，而不是只停留在语法层面。

3.5.4 TRAE的三种智能体交互执行模式

TRAE提供了三种智能体交互执行模式，以适应不同的使用场景，包括Chat模式、Agent模式和SOLO模式。在TRAE界面右侧的“AI侧栏”中（如图左所示），点击顶部的“显示/隐藏”切换按钮，就可以显示或隐藏“AI侧栏”，点击顶部的“新建任务”按钮，就可以新建一个任务（或者称为对话）。在底部的智能体交互界面中，点击“@”或“@Agent”（图左中箭头所示位置）就可以弹出三种模式的选择界面（如图右所示）。

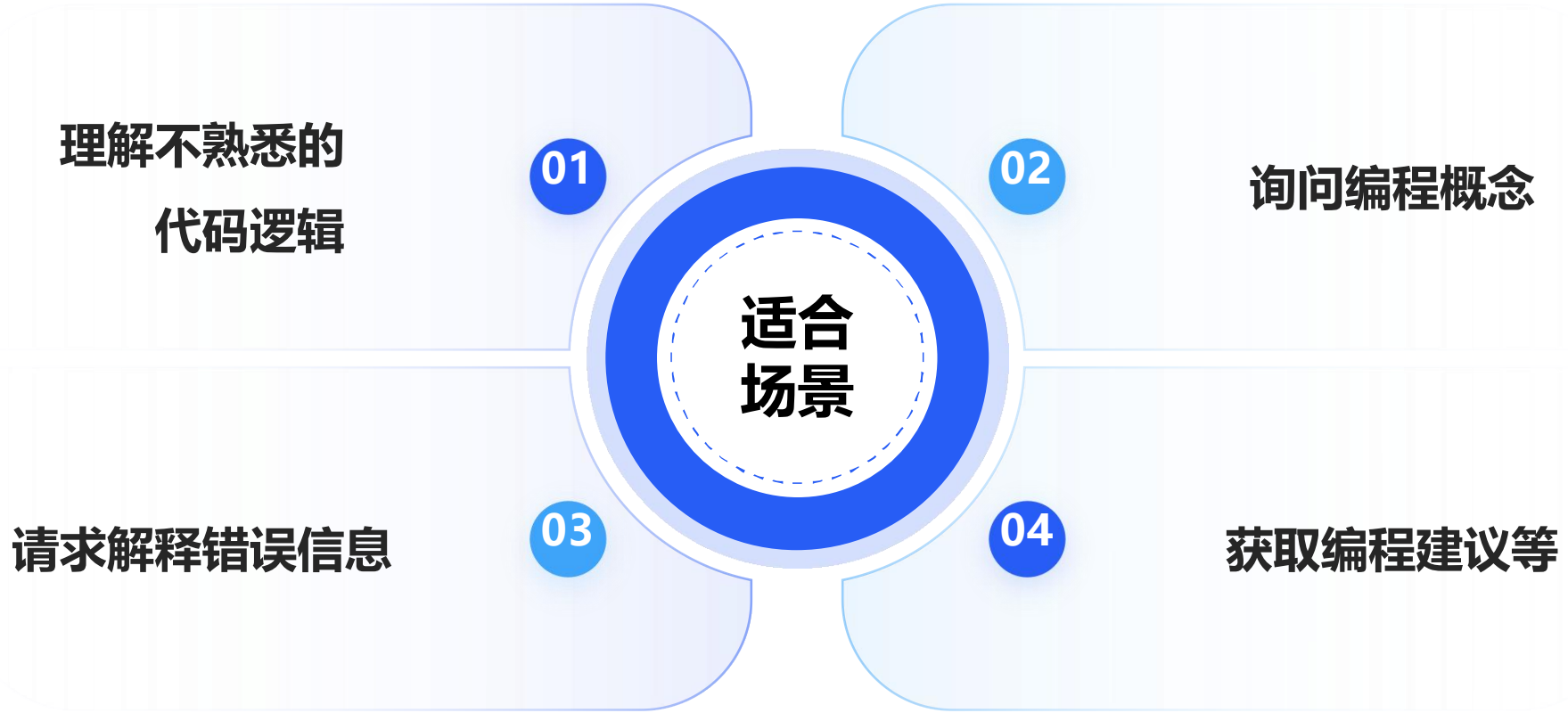


3.5.4 TRAE的三种智能体交互执行模式

■ Chat模式

Chat模式是TRAE最基础的工作方式，类似于与AI进行对话。用户可以在对话框中输入问题或需求，AI理解后给出回答或建议。Chat模式下TRAE并不会直接修改用户的工程目录，只会在聊天框中输出建议和代码示例，可以理解为是TRAE的只读模式。

Chat模式适合的场景包括：



3.5.4 TRAE的三种智能体交互执行模式

■ Chat模式

Chat模式的操作流程如下：

3.5.4 TRAE的三种智能体交互执行模式

■ Agent模式

Agent模式是TRAE的进阶工作方式，适合需要AI协助完成具体编程任务时使用。在Agent模式下，AI不仅能回答问题，还能够直接操作项目文件、修改代码、执行命令。Agent模式适合的场景包括：

3.5.4 TRAE的三种智能体交互执行模式

■ Agent模式

Agent模式的操作流程如下（如图所示）：

01. 打开项目文件夹，在TRAE的智能体交互界面中选择Agent模式。

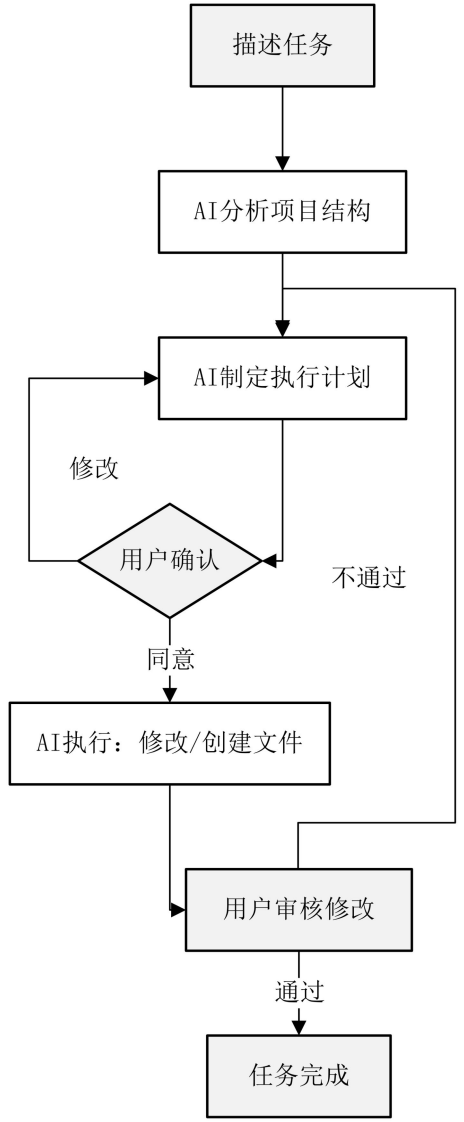
02. 在智能体交互界面的对话框中描述需要完成的任务。

03. AI分析项目结构，制定执行计划。

04. AI在用户确认后开始执行，可以是修改文件、创建文件、执行命令等。

05. 用户可以随时干预，修改AI的执行计划。

06. 任务完成后，用户审核AI的修改，确认无误则保存。



3.5.4 TRAE的三种智能体交互执行模式

■ Solo模式

TRAE还提供SOLO模式，这是一种更高自主性的工作模式。在SOLO模式下，AI可以独立完成从需求分析、技术方案设计、代码实现到测试验证的完整开发流程，用户只需提供需求描述，无需参与中间过程。SOLO模式代表了AI编程工具向更高自主性演进的方向，适合快速原型开发和简单功能实现。但由于该模式仍在探索阶段，其输出质量在复杂场景下尚不稳定，可能需要用户进行较多的后期调整。因此，本书的编程实践统一使用Agent模式，在保证教学效果的同时，让读者逐步建立对AI编程工作流的系统理解。

3.5.4 TRAE的三种智能体交互执行模式

■ **三种工作模式的对比**

下表给出了TRAE的三种工作模式的对比，对比的特性包括AI主动性、文件操作、用户参与度、适用任务、学习难度、输出可控性。

特性	Chat模式	Agent模式	SOLO模式
AI主动性	被动响应	中等主动	高度主动
文件操作	不可直接操作	可操作	完全自主操作
用户参与度	高	中等	低
适用任务	问答、解释、代码片段生成	具体功能实现、小型项目开发	快速原型、简单功能自动生成
学习难度	低	中	低（上手），高（掌握质量）
输出可控性	高	中	低

3.5.5 Trae开发实例

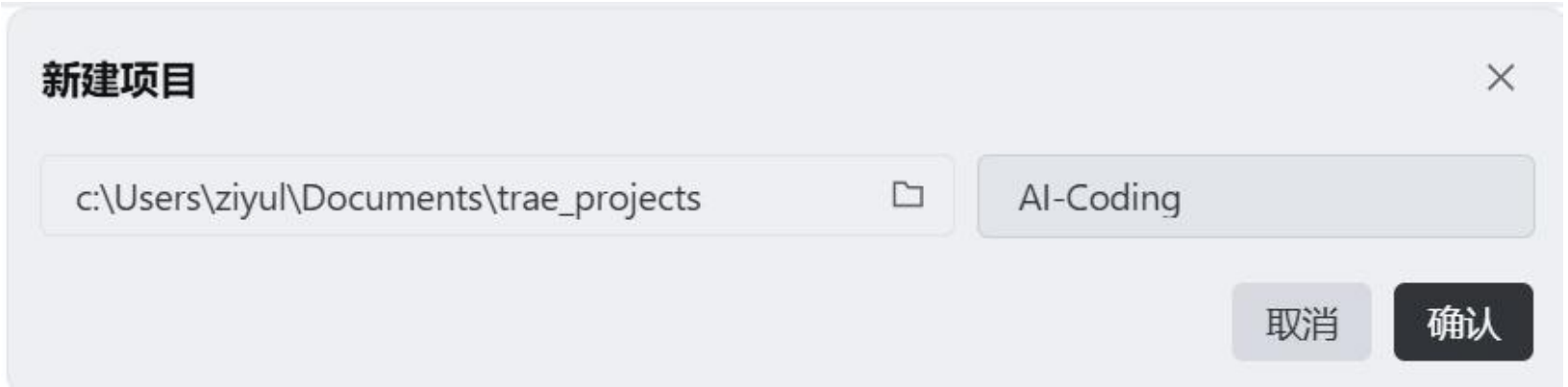
■ **步骤1：启动TRAE并切换至IDE模式**

启动本地已安装的TRAE IDE，完成账号登录，进入TRAE主界面。 TRAE默认提供IDE模式与SOLO模式，点击界面左上角的模式切换按钮，确认切换至“IDE模式”，该模式可保留传统开发逻辑，同时支持提示词生成代码，兼顾自主性与便捷性，适合入门阶段快速上手。

3.5.5 Trae开发实例

■ **步骤2：新建AI编程项目**

在TRAE主界面的顶部菜单栏中选择“文件”，在弹出的菜单中选择“新建窗口”，在出现的界面中（如图左所示）点击“新建项目”，在出现的界面中（如图右所示），输入项目名称为“AI-Coding”，点击“确认”，生成一个名为“AI-Coding”的新项目。



3.5.5 Trae开发实例

■ 步骤3：创建Python代码文件

右键点击左侧资源管理器中的“新建文件”图标（如图所示），在AI-Coding项目根目录下新建一个文件，命名为“hello_ai.py”；双击该文件，右侧编辑器区域自动打开该文件。

3.5.5 Trae开发实例

■ 步骤4：输入提示词，生成HelloWorld核心代码

本步骤核心的是通过提示词指令，让TRAE自动生成HelloWorld代码，无需手动编写，贴合AI编程的便捷性特点，提示词设计贴合入门场景。

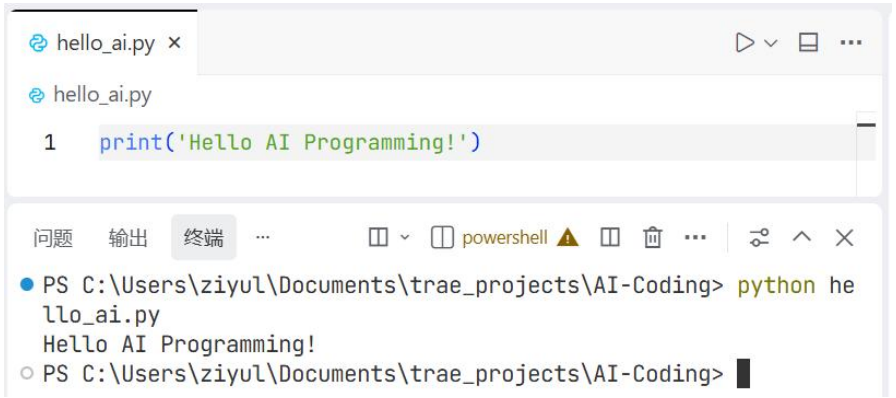
在界面右侧的“AI侧栏”的智能体交互界面的对话框框中（如图2-9所示），输入清晰、简洁的提示词（提示词需明确编程语言、功能需求，便于TRAE精准生成代码），这里推荐提示词为“使用Python语言，编写一个AI编程入门的HelloWorld程序，功能是输出‘Hello AI Programming!’”。点击“发送”按钮（智能体交互界面右下角的上箭头按钮），TRAE会自动在hello_ai.py中添加Python代码“print('Hello AI Programming!')”。点击hello_ai.py的代码区域，然后，按“Ctr+Enter”组合键就可以接受AI生成的代码。

3.5.5 Trae开发实例

■ 步骤5：运行程序并查看结果

生成代码hello_ai.py以后，可以执行如下步骤：

保存代码	使用快捷键Ctrl+S（Windows/Linux）或Cmd+S（macOS）保存生成的代码，避免代码丢失。
打开终端	点击TRAE顶部菜单栏的“查看”，选择“终端”，启动集成终端（无需额外打开系统终端）。
运行代码	在终端中（如图2-10所示），输入命令“python hello_ai.py”，按下回车键执行；或者也可以采用另一种方式运行代码，可以在顶部菜单栏选中「运行」，再选中「以非调试模式运行」，在出现的界面中选择「Python Debugger」，就可以运行代码。
查看结果	若终端输出「Hello AI Programming! 」，则说明TRAE IDE提示词功能、Python环境均正常，案例实操成功。



3.5.5 Trae开发实例

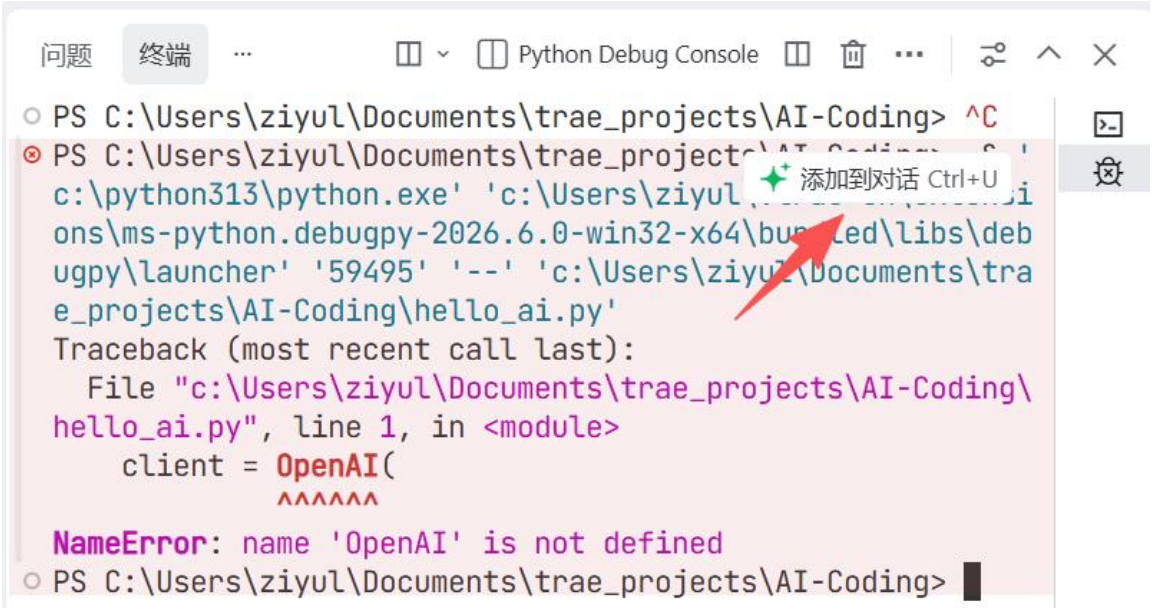
■ **步骤6：开发一个更加复杂的房贷计算器**

在智能体交互界面的对话框中，输入提示词“使用Python语言开发一个房贷计算器”，点击“提交”，TRAE会自动新建一个Python代码文件（比如mortgage_calculator.py），然后，调试该程序代码，就可以得到一个房贷计算器。

3.5.5 Trae开发实例

■ 步骤7：让AI自动解决程序错误

AI自动生成的代码，有时候可能存在错误，在TRAE中运行程序时会报错。当遇到错误时，可以让AI自动帮忙修正错误。如图所示，某个程序在调试运行时报错，这时，用鼠标单击代码错误信息，就会在界面上出现“添加到对话（Ctrl+U）”的按钮，点击这个按钮，就可以直接把这些错误信息添加到TRAE的智能体交互界面的对话框中，然后点击对话框右下角的“发送”按钮，提交这些对话内容，AI就会自动帮你修复错误。这种错误解决方法，在AI编程中非常典型，读者在AI编程中遇到任何错误，一定要首先想到用这种方式让AI帮你修复错误，而不是自己手动修正错误，这样可以大幅提高编程效率。



Part 06

为TRAE选择合适的编程大模型

3.6为TRAE选择合适的编程大模型

大模型选择的重要性

评估大模型编程能力的参考

TRAE的模型选择

3.6.1大模型选择的重要性

AI编程工具的能力上限，很大程度上取决于底层大模型的质量。选择合适的大模型，是提升AI编程效率的关键。

不同大模型在代码能力上存在差异。一些大模型在特定编程语言上表现突出，另一些大模型在复杂推理任务上更具优势；一些大模型响应速度快但能力有限，另一些大模型能力更强但可能需要更长处理时间。

此外，不同大模型的定价策略也不同。免费大模型和付费大模型在能力上通常存在差距，如何在预算和效果之间取得平衡，是每个AI编程工具用户都需要考虑的问题。

3.6.2评估大模型编程能力的参考

SWE-bench（Software Engineering Benchmark）是一个权威的大模型代码能力评估基准。它收集了来自真实开源项目的代码任务，要求大模型根据问题描述修复代码中的Bug。通过在SWE-bench上的表现，可以客观评估不同大模型解决真实编程问题的能力。SWE-bench的官网是

https://www.swebench.com/multilingual-leaderboard.html

根据最新的评估结果，Gemini 3 Flash、Claude 4.6 Opus、GLM-5等模型在代码任务上表现最为突出（如图所示）。当然，基准测试成绩仅供参考，实际使用体验可能因任务类型和个人偏好而异。

Model	% Resolved	Avg. $	Trajs	Org	Date	Agent
Gemini 3 Flash	72.70	$0.35			2026-02-13	2.0.0a0
Claude 4.6 Opus	72.00	$0.66			2026-02-13	2.0.0a0
Claude 4.5 Opus	70.70	$0.83			2026-02-13	2.0.0a0
GLM-5	69.70	$0.64			2026-02-13	2.0.0a0
Gemini 3 Pro	68.70	$1.02			2026-02-13	2.0.0a0
Minimax 2.5	68.30	$0.10			2026-02-16	2.0.0a0
Kimi K2.5	67.30	$0.69			2026-02-13	2.0.0a0
Claude 4.5 Sonnet	67.00	$0.67			2026-02-13	2.0.0a0
GPT-5.2 (high reasoning)	66.70	$0.54			2026-02-13	2.0.0a0
GPT-5-2 Codex	66.30	$0.66			2026-02-20	2.0.0
GPT 5.2 Codex	66.30	$0.66			2026-02-20	2.0.0
Claude 4.5 Haiku	64.70	$0.38			2026-02-13	2.0.0a0
DeepSeek V3.2	59.00	$0.38			2026-02-13	2.0.0a0

3.6.3 TRAE的模型选择

TRAE支持接入多个大模型，用户可以根据需要选择合适的模型。TRAE默认集成的大模型以国内大模型为主，包括：

Doubao系列大模型

字节跳动推出的Doubao系列大模型，在代码理解和生成方面表现优异。其中，Doubao-Seed-Code是经过代码专项优化的大模型，响应速度快，适合日常编程任务。

MiniMax系列大模型

MiniMax的M2系列大模型在多轮对话和复杂逻辑处理方面有出色表现。

GLM系列大模型

智谱AI的GLM系列大模型支持长上下文处理，适合代码审查和分析任务。

DeepSeek系列大模型

深度求索的DeepSeek-V4系列大模型在代码生成和数学推理方面能力突出。

Kimi系列大模型

Moonshot的Kimi大模型以超长上下文窗口著称，能够处理大规模代码库。

3.6.3 TRAE的模型选择

在TRAE中，大模型的选择分为两种模式：

Auto模式（推荐）

开启后TRAE会根据任务难度和后端资源空闲情况，自动选择最合适的大模型。Auto模式能够满足大多数场景的需求，无需手动干预，并且可以在后端繁忙的时候减少等待

手动模式

关闭Auto模式后，可以从大模型列表中手动选择特定大模型

3.6.3 TRAE的模型选择

关于大模型的选择，这里给出几点建议：

日常编程任务	**保持Auto模式开启，TRAE会自动选择最优模型**
实验复现	**如果想要与本书实验内容一致，可以手动选择Doubao-Seed-Code大模型。该大模型由字节跳动提供，资源充足，响应稳定，本书中所有示例均由该大模型生成**
结果不满意时	**如果对TRAE多次完成的任务都不满意，可以尝试手动切换其他大模型。不同大模型在不同任务上的表现存在差异，换用其他大模型可能会获得更好的结果**

3.6.3 TRAE的模型选择

在TRAE中切换模型的步骤如下：

（1）在智能体交互界面中（如图左所示），点击“Auto”下拉列表

（2）在弹出的界面中，关闭"Auto Mode"开关

（3）在弹出的界面中（如图右所示），从下拉列表中选择想要使用的模型（如Doubao-Seed-Code）

Part 07

大模型友好的技术栈

3.7大模型友好的技术栈

为什么技术栈选择很重要

选择主流技术栈的原则

技术栈选择清单

3.7.1为什么技术栈选择很重要

使用AI编程工具时，技术栈的选择对开发效率有显著影响。这与AI编程工具的底层原理密切相关。

大模型的代码能力来自训练数据。大模型见过的代码越多、理解得越深入，生成的结果就越准确、越符合人类期望。因此，大模型对某种编程语言、框架、工具的了解程度，直接决定了在该技术栈上使用AI编程工具的效果。

试想一个场景：开发者使用AI编程工具开发一个Web应用。如果选择“JavaScript+React”，从技术栈的主流程度而言，这种选择就意味着，模型见过大量React项目代码，理解组件化开发模式，知道常见的错误和最佳实践。

但是，如果选择某个自研的小众框架，模型可能从未见过这种技术栈，生成的代码质量就会难以保证。这就是技术栈选择对AI编程效果的影响机制。选对技术栈，AI能显著提升开发效率；选错技术栈，AI的输出质量将大打折扣。

3.7.2选择主流技术栈的原则

■ 原则一：选择主流的编程语言

编程语言的流行度决定了AI认知的"分水岭"。Java与Python凭借海量的开源代码样本，成为AI训练中最深厚的知识矿脉。选择这些主流语言，本质上是选择了与全球最成熟的大模型逻辑对齐。具体来说，推荐的编程语言包括：

Python

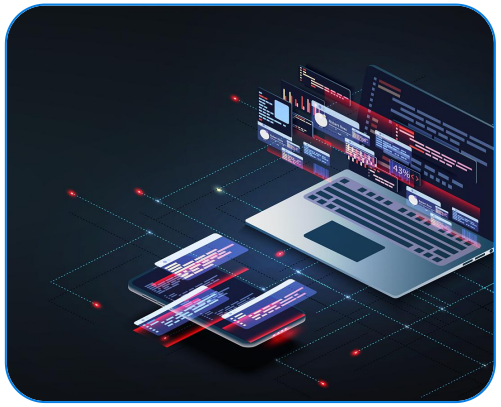
AI领域的"第一语言"，几乎所有主流大模型都对Python有出色的支持。Python的简洁语法、丰富生态使其成为AI编程的首选。

JavaScript/TypeScript

Web开发的主流语言，Node.js生态中的大量开源项目为AI提供了丰富的训练语料，是前端和全栈开发的首选语言。

Java

企业级开发的主流语言，大型项目多、代码规范性好，AI对其理解程度高，是后端开发的首选语言。



Go

云原生时代崛起的新秀，语法简洁、性能优异，主流AI大模型对Go的支持也相当不错。

3.7.2选择主流技术栈的原则

■ 原则二：选择AI友好的框架和工具

框架层面，应该选择主流社区认可的框架。以下是各领域AI友好框架的推荐：

Web后端框架

Python的FastAPI/Django、Java的Spring Boot、JavaScript的Express/NestJS、Go的Gin/Echo

前端框架

React、Vue、Angular（按AI支持程度排序）

数据库

PostgreSQL、MySQL、Redis（都属于主流数据库，文档和社区比较丰富）

消息队列

RabbitMQ、Kafka（AI对这些工具有较好的理解）

3.7.2选择主流技术栈的原则

■ **原则三：避免使用过于小众的技术**

如果在技术选型时选择了过于小众的框架、库或自研DSL（Domain-Specific Language），AI编程工具的效果会大打折扣。具体表现包括：

具体表现

01 AI对技术不熟悉，生成的代码可能包含错误

02 遇到问题时，AI难以给出有效的解决方案

03 需要频繁查阅文档，AI的辅助价值降低

3.7.3技术栈选择清单

本书选择的后端技术栈如下：

编程语言

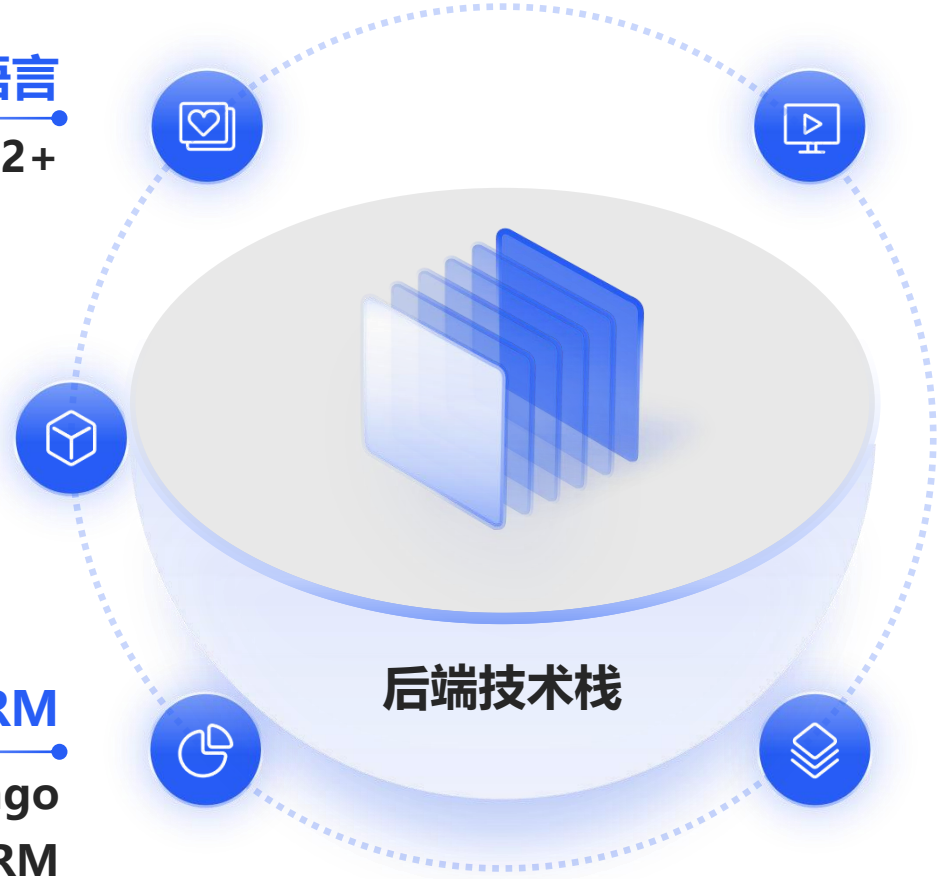
Python 3.12+

- **Web框架**

FastAPI（轻量）或
Django（全能）

- **认证**

JWT（JSON Web Token）



- **ORM**

SQLAlchemy或Django
ORM

- **数据库**

SQLite（学习用）或
PostgreSQL（生产用）

3.7.3技术栈选择清单

本书选择的前端技术栈如下：

框架

React（生态最丰富）

样式

Tailwind CSS

HTTP客户端

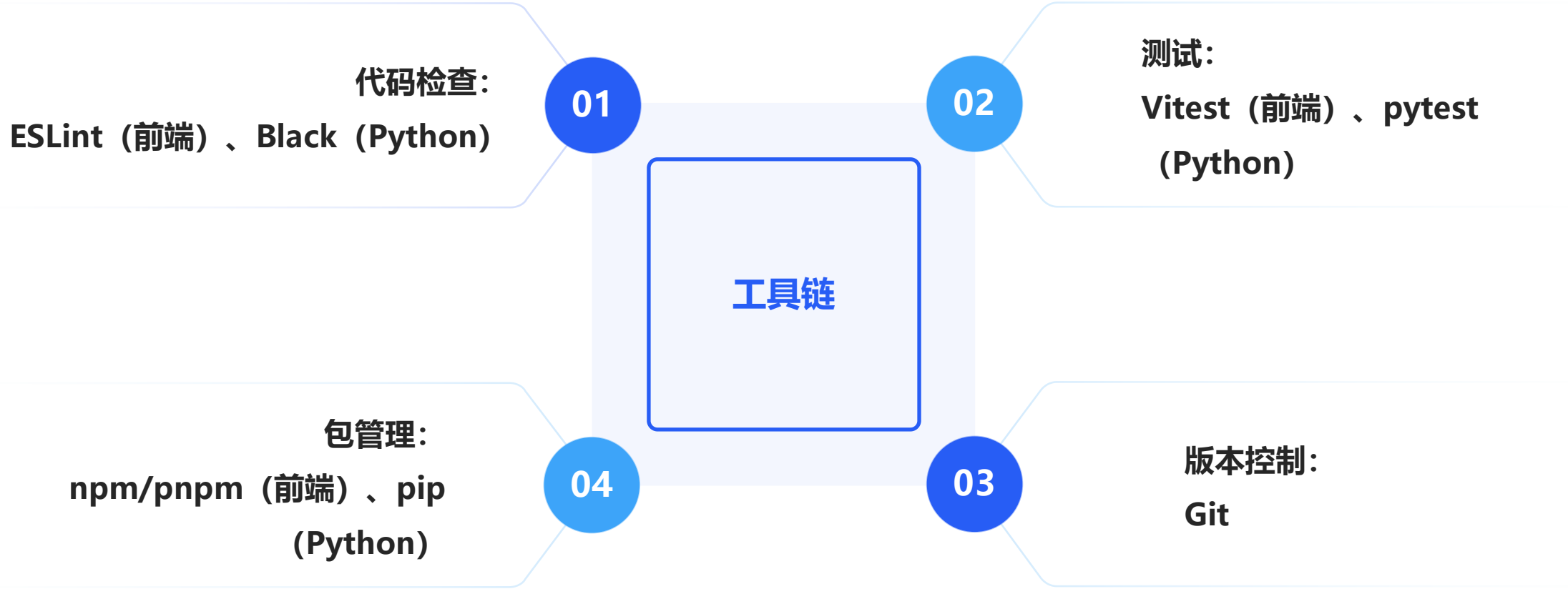
Axios

编程语言

TypeScript

3.7.3技术栈选择清单

本书选择的工具链如下：



3.7.3技术栈选择清单

下面是一个反面清单（AI编程效果差的技术）：

Part 08

用AI编程实现典型编程任务

3.8用AI编程实现典型编程任务

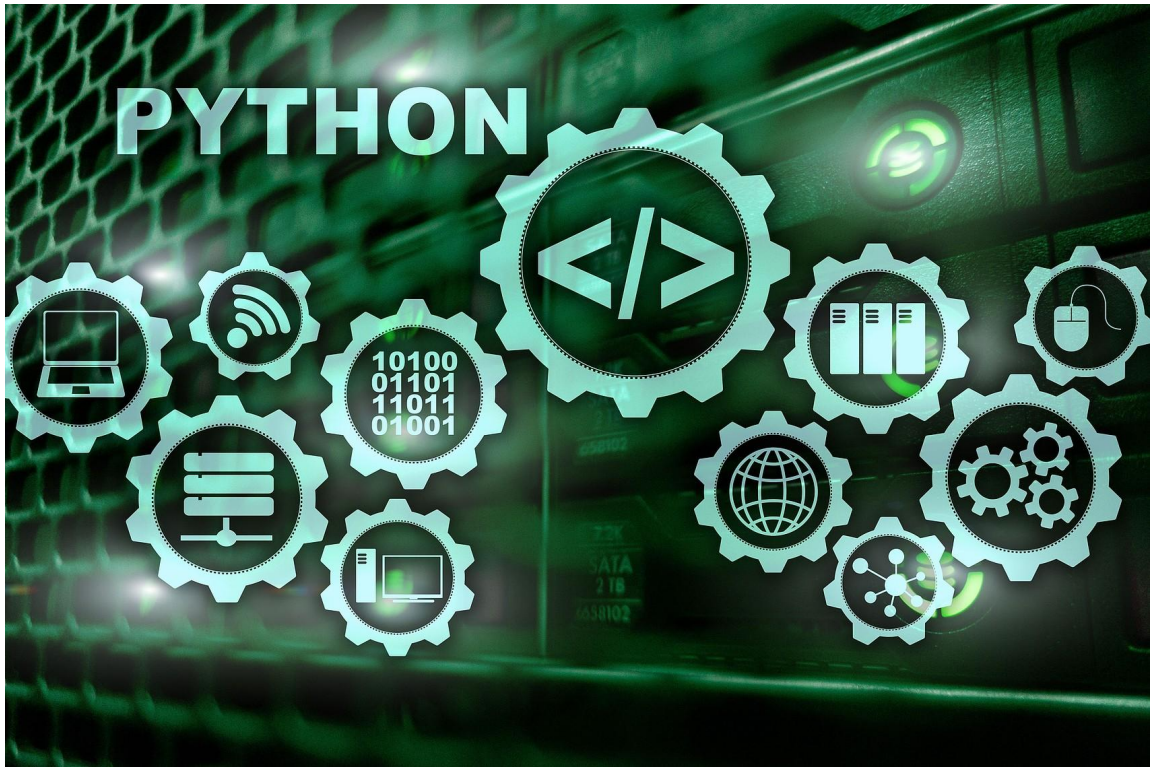
案例一：
Hello World

案例二：
二分查找

案例三：
快速排序

3.8.1案例一：Hello World

“Hello World”是编程入门的经典起点。本案例将通过创建一个最简单的Python程序，帮助读者熟悉AI编程工具的基本操作流程。本案例的任务是，创建一个Python程序，输出"Hello, World!"。虽然在2.3.5节已经介绍过“Hello World”的实例，但是，这里会按照“描述需求->AI生成代码->审查代码->运行并验证代码”的逻辑链条来重新介绍。



3.8.1案例一：Hello World

AI编程的流程具体如下：

（1）第一步：打开项目并描述需求。在TRAE中打开一个新目录，使用Chat模式向AI描述任务：

```
请用Python写一个程序，输出"Hello, World!"
```

（2）第二步：审核AI的代码产出。AI通常会生成如下代码：

```
print("Hello, World!")
```

对AI生成的代码进行人工审核，审核要点包括：语法是否正确、是否满足需求、是否符合Python代码规范。

3.8.1案例一：Hello World

（3）第三步：运行代码进行验证。可以使用如下命令运行代码：

```
python hello.py
```

预期输出结果是：Hello, World!

这是一个最简单的案例，用于让读者熟悉AI编程的基本流程：描述需求 → 审核代码 → 运行验证。即使是这样一个简单的任务，审核代码的习惯也应该从一开始就要建立起来。

3.8.2案例二：二分查找

二分查找是计算机科学中的经典算法，也是面试中的常考题目。相比“Hello World”，二分查找涉及更多的逻辑思考和边界条件处理，适合作为AI编程的中等难度案例。

本案例的任务是，实现一个二分查找函数，能够在有序数组中查找目标元素，返回其索引（若不存在返回-1）。

AI编程的流程具体如下：

AI编程的流程具体如下：

（1）第一步：描述需求。使用Agent模式，向AI详细描述任务：

```
请用Python实现一个二分查找函数，并增加相关测试。
```

（2）第二步：审核AI生成的代码。对AI生成的代码进行人工审核，比如，是否存在逻辑错误，功能是否满足需求等。

（3）第三步：运行测试验证。可以执行如下命令：

```
pytest test_binary_search.py -v
```

3.8.2案例二：二分查找

二分查找案例展示了AI编程的典型工作流程：

- 向AI描述详细的需求（包括类型注解、测试要求）
- 审核AI的方案设计和代码产出
- 运行测试验证功能正确性
- 发现问题及时反馈给AI进行修改

3.8.3案例三：快速排序

快速排序是面试中最常考的算法之一，实现涉及递归、分区和边界条件处理，对基础要求较高。对于基础扎实的读者来说可能轻车熟路，但对于基础薄弱的读者往往需要花费不少时间思考。然而，借助AI编程，只需一句话就能完成。

AI编程的流程具体如下：

（1）第一步：描述需求。使用Agent模式，向AI描述任务：

```
请用Python实现快速排序算法
```

（2）第二步：审核AI生成的代码。审核的要点包括：递归终止条件是否正确、分区逻辑是否正确处理重复元素、是否考虑了空数组和单元素数组的边界情况。

（3）第三步：运行测试验证。可以执行如下命令：

```
pytest test_quicksort.py -v
```

预期结果是所有测试都通过。

Part 09

Git版本控制系统

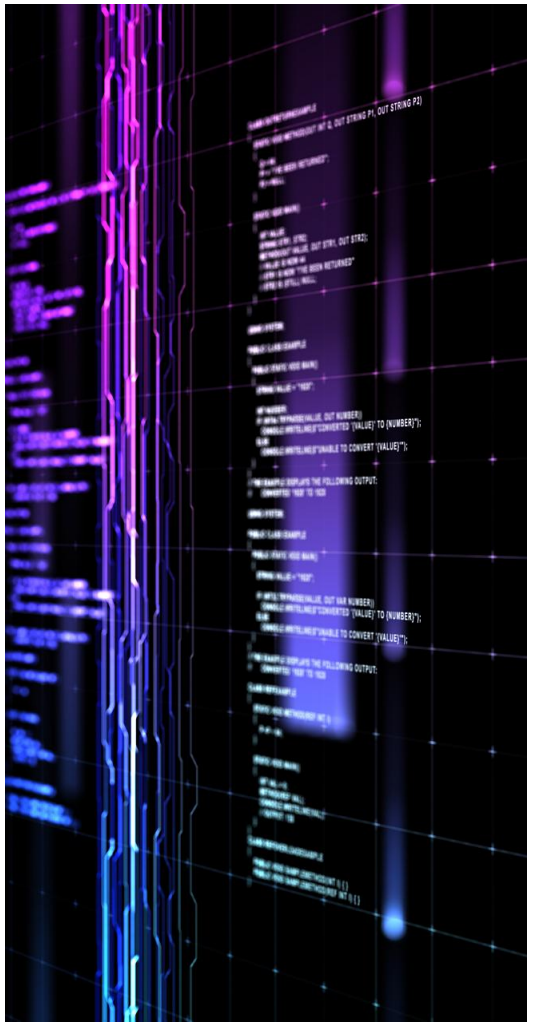
3.9 Git版本控制系统

- 版本控制
- Git是什么
- GitHub与Gitee
- 安装Git
- Git基本操作
- Git分支管理
- Git Worktree

3.9.1 版本控制

■ 什么是版本控制

在软件开发过程中，代码文件会随着功能迭代、Bug 修复和需求变更而不断修改。如果没有一个有效的管理机制，开发者很快就会面临这样的困境：刚刚还能正常运行的代码突然出错，却不知道哪里改动了；多人合作时，各自修改了同一个文件，合并时一片混乱；辛苦写了几天的代码想要退回到某个历史状态，却发现根本无从回溯。



版本控制（Version Control），是一种记录文件内容变化、以便将来查阅特定版本内容的系统。它不仅能够追踪每一次修改的内容、时间和作者，还能在需要时将文件恢复到任意历史状态。版本控制系统是现代软件开发的基础设施，任何稍具规模的项目都离不开它。

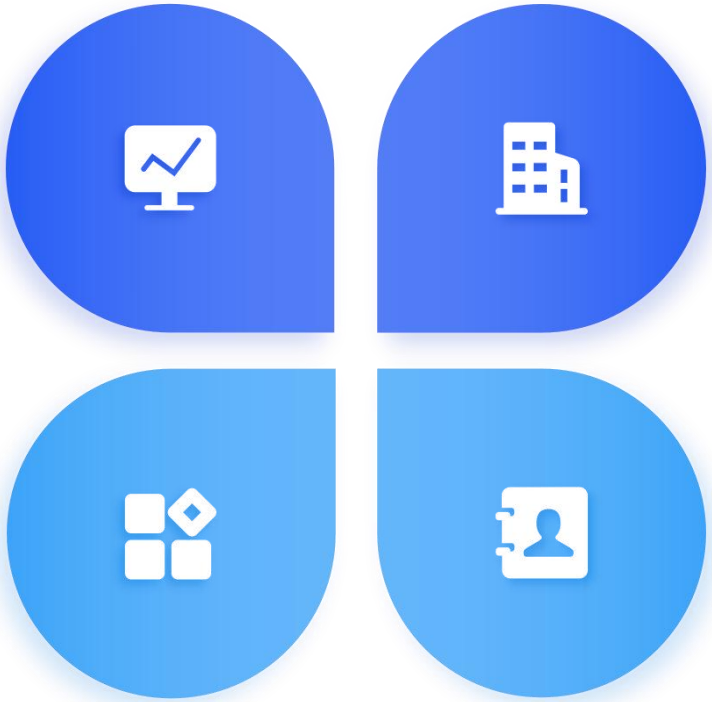
3.9.1 版本控制

■ 为什么需要版本控制

对于初学者来说，版本控制的必要性可能并不直观——在项目规模较小时，手动备份文件夹似乎也能应付。但随着项目复杂度增加，版本控制的价值就会逐渐凸显出来。版本控制能够解决以下几类核心问题：

可追溯性

每一次代码提交都会留下记录，包括修改了哪些文件、改动了哪些内容、由谁在什么时间提交，以及此次修改的目的说明。这使得代码的演进历史完全可追溯，任何问题都可以溯源。



并行协作

多人同时开发同一项目时，版本控制系统提供了分支机制，每个开发者可以在独立的分支上工作，互不干扰，最终再将各自的工作合并到主线。

可回退性

当新版本引入了严重 Bug，或者某个实验性功能的方向错误时，版本控制系统可以在几秒内将代码回退到任意历史版本，大大降低了试错成本。

备份与安全

将代码托管到远程版本控制服务器（如 GitHub），相当于在云端建立了代码备份，即便本地设备损坏，代码也不会丢失。

3.9.1 版本控制

■ 当前版本控制的主流选择

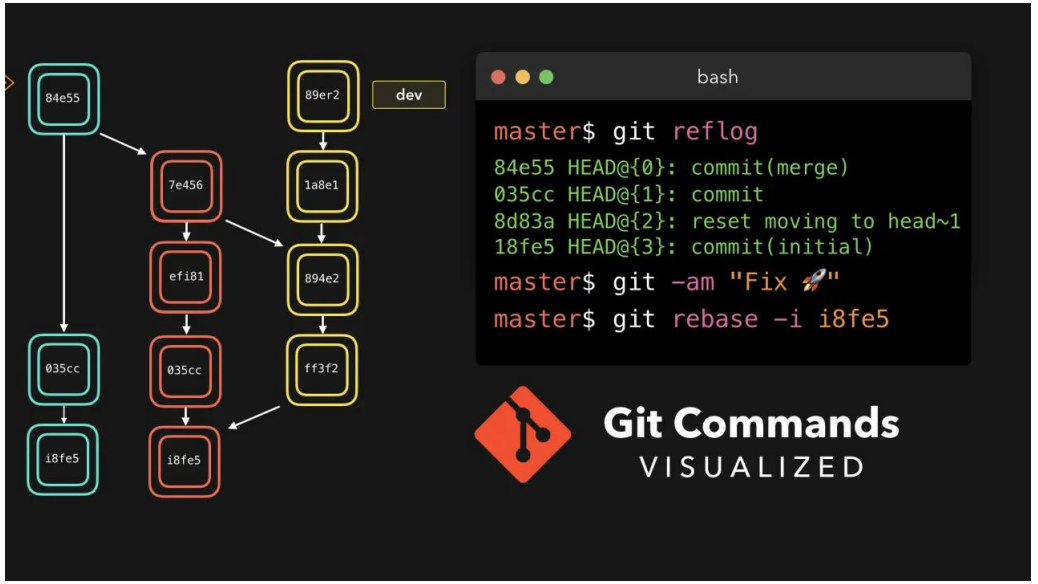
历史上存在多种版本控制系统，如 CVS、SVN（Subversion）、Mercurial 等，但是，在当今软件开发领域，Git 已经成为绝对主流的版本控制系统。无论是个人项目、企业内部系统，还是开源社区，Git 几乎是版本控制的唯一选择。

3.9.2 Git是什么

■ Git的诞生

Git 由 Linux 内核创始人 Linus Torvalds 于 2005 年开发。Git 的诞生背景颇为有趣，Linux 内核项目长期使用一款名为 BitKeeper 的商业版本控制系统，但在 2005 年，BitKeeper 因授权争议停止向 Linux 社区免费提供服务。Linus Torvalds 决定自己动手，在短短两周内开发出了 Git 的初始版本。

Git 在设计上借鉴了 BitKeeper 的经验，并针对大型分布式项目的需求做了大量优化。Git 采用分布式架构，每个开发者的本地仓库都是完整的版本库，不依赖中央服务器也能进行提交、查看历史、创建分支等操作。这种设计使 Git 天然具有高可用性和高性能。



3.9.2 Git是什么

■ Git的流行程度

自 2005 年诞生以来，Git 的普及速度超过了所有前辈版本控制系统。根据 Stack Overflow 开发者调查报告，自 2015 年起，Git 在版本控制系统中的使用率已超过 90%，且这一比例仍在持续上升。几乎所有主流开源项目都使用 Git 进行版本管理，并托管到 GitHub 平台上供全球开发者协作。

在企业环境中，无论是国内的阿里巴巴、腾讯、字节跳动，还是国际上的 Google、Microsoft、Meta，Git 都是标准的代码管理工具。学习 Git，是每一位软件开发者的必备技能。

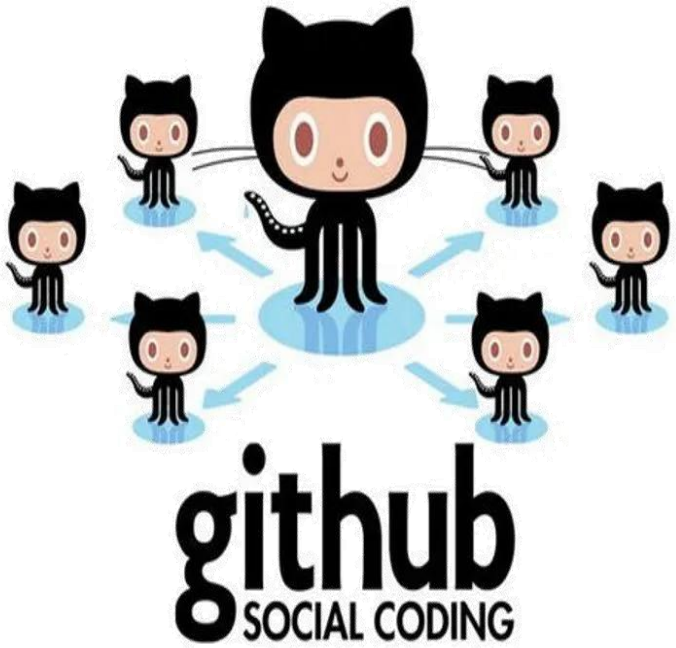
3.9.3 GitHub与Gitee

■ GitHub简介

01 GitHub 是全球最大的代码托管平台，由 Tom Preston-Werner、Chris Wanstrath 和 PJ Hyett 于 2008 年创立，2018 年被 Microsoft 以 75 亿美元收购。GitHub 基于 Git 版本控制系统构建，提供远程代码仓库托管服务，同时集成了 Issue 追踪、Pull Request 代码审查、Actions 持续集成等一系列协作开发功能。

02 截至 2024 年，GitHub 拥有超过 1 亿注册用户和超过 4.2 亿个代码仓库，是全球规模最大的开源社区。几乎所有知名开源项目，包括 Linux 内核、Python、TensorFlow、React 等，都将代码托管在 GitHub 上。对于开发者而言，GitHub 既是代码仓库，也是作品集和社交网络。

03 GitHub 的访问地址为 https://github.com。注册账号后，可以免费创建公开仓库（数量不限）和私有仓库。对于学生，GitHub 还提供了 GitHub Student Developer Pack，包含大量免费的开发工具和服务。



3.9.3 GitHub与Gitee

■ Gitee简介

Gitee（码云）是国内开源中国（OSChina）推出的代码托管平台，网址为 https://gitee.com。Gitee 同样基于 Git，提供与 GitHub 类似的代码托管和协作功能，且完全使用中文界面。

对于国内开发者，尤其是初学阶段，Gitee 有两个显著优势：一是访问速度稳定，不受网络波动影响；二是中文界面对于英文基础薄弱的同学更加友好。Gitee 上也有大量国内开发者的开源项目，涵盖 Java、Python、前端等各类技术栈。

建议初学者同时注册 GitHub 和 Gitee 账号。日常学习和个人项目可以优先选择 Gitee，如果 GitHub 因网络原因访问不稳定，Gitee 是一个可靠的替代方案。当项目需要与国际开源社区协作，或需要提升个人在全球开发者社区中的影响力时，应当优先选择 GitHub。

3.9.4 安装Git

■ Windows系统安装Git

在 Windows 系统上安装 Git 的步骤如下：

访问 Git 官方网站（https://git-scm.com/downloads）或 Git for Windows 项目页面（https://gitforwindows.org），下载最新版本的 Windows 安装包，文件名类似Git-2.x.x-64-bit.exe。

双击安装包，启动安装向导。安装过程中大多数选项保持默认即可。

安装完成后，在桌面或开始菜单中找到"Git Bash"图标，双击打开。

在 Git Bash 窗口中输入以下命令验证安装是否成功：

```
git --version
```

如果显示类似"git version 2.x.x.windows.x"的版本信息，则说明安装成功。

3.9.4 安装Git

■ Git Bash的使用

Git Bash 是 Git for Windows 自带的命令行工具，提供了类 Unix 的终端环境。对于 Windows 用户，推荐使用 Git Bash 来执行 Git 命令，而不是 Windows 的命令提示符（CMD）或 PowerShell，原因如下：

- Git Bash提供了标准的 Unix 命令（如 ls、pwd、cd、mkdir 等），与macOS和Linux的使用体验一致，便于跨平台学习。
- Git Bash直接集成了Git，无需额外配置路径变量。
- Git Bash支持颜色高亮显示Git状态信息，可读性更好。

3.9.4 安装Git

Git Bash的基本使用方法与标准的Bash终端相同。表2-2 列出了Git Bash中常用的基础命令。

命令	功能说明	示例
pwd	显示当前工作目录的完整路径	pwd
ls	列出当前目录的文件和子目录	ls / ls -la
cd	切换当前工作目录	cd /c/Users/student/Desktop
mkdir	创建新目录	mkdir my-project
touch	创建新的空文件	touch README.md
cat	查看文件内容	cat README.md
clear	清空终端屏幕	clear

注意：在 Git Bash 中，Windows 路径采用 Unix 风格表示，例如 C:\Users\student 应写成 /c/Users/student。这是 Git Bash 与 Windows CMD 在路径表示上的主要区别。

3.9.4 安装Git

■ Git的基本配置

安装 Git 后，在开始使用前需要进行基本配置。最重要的配置是设置用户名和电子邮件地址，这些信息会附加在每次提交记录中，用于标识提交者身份。

打开 Git Bash（Windows 系统）或终端（macOS/Linux 系统），输入以下命令进行配置：

```
# 设置全局用户名（把下面的linziyu替换为自己的姓名，建议使用姓名拼音）
git config --global user.name "linziyu"
# 设置全局邮箱地址（把下面的ziyulin@xmu.edu.cn替换为自己的邮箱）
git config --global user.email "ziyulin@xmu.edu.cn"
# 查看当前配置
git config --list
```

3.9.5 Git基本操作

Git 的工作区域分为三个层次：工作目录（Working Directory）、暂存区（Staging Area，又称索引）和本地仓库（Local Repository）。理解这三个区域的关系，是掌握 Git 操作的关键。工作目录是实际编辑文件的地方；暂存区是一个中间区域，用于整理即将提交的变更；本地仓库则是永久保存提交历史的`地方。如图所示，展示了 Git 三个工作区域及其关系。

3.9.5 Git基本操作

■ 初始化仓库：git init

git init 命令用于将当前目录初始化为一个 Git 仓库（每个新项目都要执行一次这个命令，为该项目初始化Git仓库）。执行后，Git 会在该目录下创建一个隐藏的 .git 子目录，里面存储了所有版本控制所需的元数据和对象数据库。 在Git Bash中执行如下命令初始化仓库：

```
# 在当前目录初始化Git仓库（建议采用这种方式）
git init
# 也可以指定目录名，Git会自动创建该目录并初始化（不建议采用这种方式）
git init my-project
```

执行 git init 后，终端会显示"Initialized empty Git repository in .../目录名/.git/"，表示仓库初始化成功。此时仓库为空，尚未有任何提交记录。

3.9.5 Git基本操作

■ 添加文件到暂存区：git add

git add 命令将工作目录中的文件变更添加到暂存区。只有加入暂存区的变更，才会被包含在下一次提交中。具体用法如下（在Git Bash中执行命令）：

```
# 添加单个文件到暂存区
git add filename.py
# 添加当前目录下所有变更到暂存区
git add .
# 添加特定类型的所有文件
git add *.py
```

建议养成有选择性地 git add 的习惯，而不是总是使用“ git add .”。明确指定要提交的文件，可以避免将不必要的临时文件或调试代码混入提交记录。

3.9.5 Git基本操作

■ 提交变更：git commit

git commit 命令将暂存区的内容永久保存到本地仓库，生成一个新的提交记录。每次提交都需要附带一条提交说明（commit message），描述此次修改的目的和内容。

```
# 提交并附带简短说明
git commit -m "Add login feature"
# 提交已追踪文件的所有变更（跳过 git add 步骤，但不包含新文件）
git commit -am "Fix calculation bug in utils.py"
```

3.9.5 Git基本操作

■ 查看状态：git status

git status 命令显示当前工作目录和暂存区的状态，包括哪些文件已修改但未暂存、哪些文件已暂存但未提交、哪些文件是新创建的（未被 Git 追踪）等信息。具体用法如下：

```
git status
```

■ 查看变更内容：git diff

git diff 命令显示工作目录中尚未暂存的变更内容，即文件与最后一次暂存（或提交）之间的差异，具体用法如下：

```
# 查看所有未暂存的变更
git diff
# 查看已暂存但未提交的变更
git diff --staged
# 查看两次提交之间的差异
git diff commit1 commit2
```

3.9.5 Git基本操作

■ 查看提交历史：git log

git log 命令显示当前分支的提交历史记录，包括每次提交的哈希值、作者、时间和提交说明。具体用法如下：

```
# 查看完整提交历史
git log
# 以简洁单行格式显示历史
git log --oneline
# 显示最近 5 次提交
git log --oneline -5
# 图形化显示分支合并历史
git log --oneline --graph --all
```

3.9.5 Git基本操作

■ 推送到远程仓库：git push

git push 命令将本地仓库的提交推送到远程仓库（如 GitHub 或 Gitee），使其他协作者或云端备份能够获取最新代码。在执行 git push 前，需要先将本地仓库与远程仓库关联。具体用法如下：

```
# 关联远程仓库（首次设置，origin 是远程仓库的别名）
git remote add origin https://github.com/username/repo-name.git
# 推送本地 main 分支到远程，-u 参数建立追踪关系
git push -u origin main
# 之后推送只需输入
git push
```

3.9.5 Git基本操作

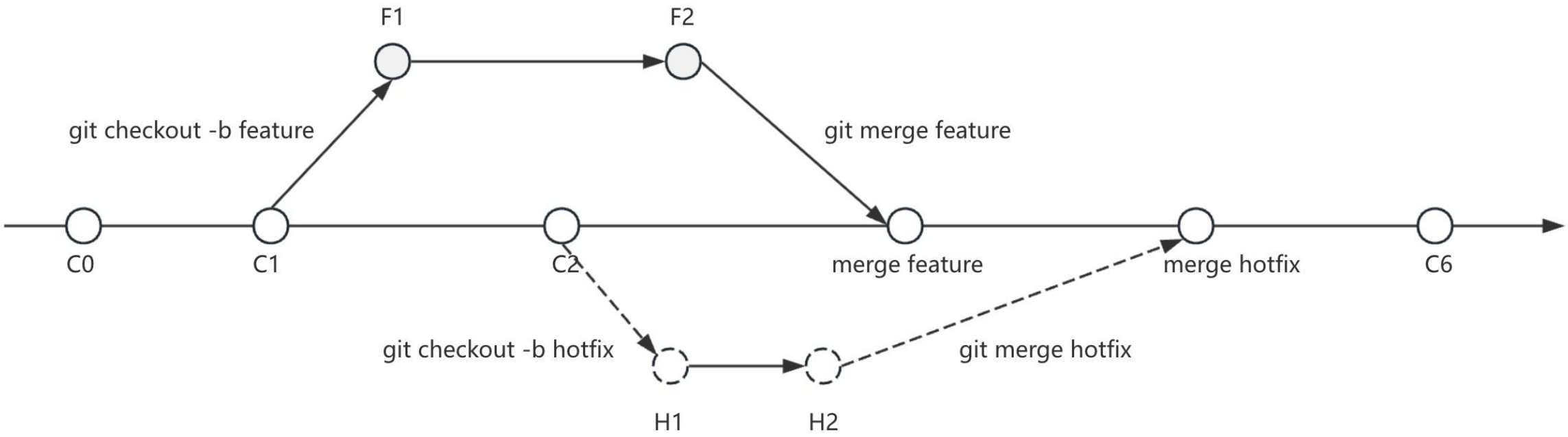
■ 命令汇总

下表汇总了 Git 基本操作命令，供读者参考。

命令	功能说明	常用参数
git init	初始化本地仓库	目录名（可选）
git add	添加文件到暂存区	. 表示所有变更
git commit	提交暂存区内容到仓库	-m "说明文字"
git status	查看工作目录和暂存区状态	无
git diff	查看文件变更内容	--staged 查看已暂存变更
git log	查看提交历史	--oneline 单行格式
git push	推送到远程仓库	-u origin main
git pull	从远程仓库拉取更新	无
git clone	克隆远程仓库到本地	仓库 URL

3.9.6 Git分支管理

分支（Branch）是 Git 最强大的特性之一。分支允许开发者从主线代码上创建一个独立的开发轨迹，在其中进行实验性开发、Bug 修复或新功能开发，不影响主线代码的稳定性。完成后再将分支合并回主线。下图展示了 Git 分支管理的典型工作流程。



3.9.6 Git分支管理

■ 分支操作基础命令

（1）git branch——分支管理

git branch 命令用于查看、创建或删除分支。具体用法如下：

```
# 查看本地所有分支（当前分支前有 * 标记）
git branch
# 创建新分支
git branch feature-login
# 删除分支（分支已合并后方可删除）
git branch -d feature-login
# 强制删除未合并的分支
git branch -D feature-login
```

3.9.6 Git分支管理

■ 分支操作基础命令

（2）git checkout——切换分支

git checkout 命令用于切换到指定分支，或恢复工作目录中的文件。具体用法如下：

```
# 切换到已有分支
git checkout feature-login
# 创建新分支并立即切换（常用的简洁写法）
git checkout -b feature-register
# 现代 Git（2.23+）推荐使用 git switch 替代 git checkout 的切换功能
git switch feature-login
git switch -c feature-register   # 创建并切换
```

3.9.6 Git分支管理

■ 分支操作基础命令

（3）git merge——合并分支

git merge 命令将指定分支的提交历史合并到当前分支。具体用法如下：

```
# 切换到主分支
git checkout main
# 将 feature-login 分支合并到当前（main）分支
git merge feature-login
```

如果两个分支修改了同一文件的同一位置，合并时会产生冲突（Conflict）。此时 Git 会在文件中标记冲突区域，开发者需要手动编辑文件解决冲突，然后再次执行 git add 和 git commit 完成合并。

3.9.6 Git分支管理

■ 典型分支工作流程

在实际开发中，一个规范的分支使用流程通常如下：

从 main 分支创建功能分支：git checkout -b feature-xxx;

在功能分支上进行开发，定期提交：git add . && git commit -m "...";

开发完成后，切换回 main 分支：git checkout main;

将功能分支合并到 main：git merge feature-xxx;

删除已合并的功能分支（可选）：git branch -d feature-xxx;

推送到远程仓库：git push。

这种工作方式被称为"功能分支工作流"（Feature Branch Workflow），是个人项目和小型团队最常用的Git工作流之一。

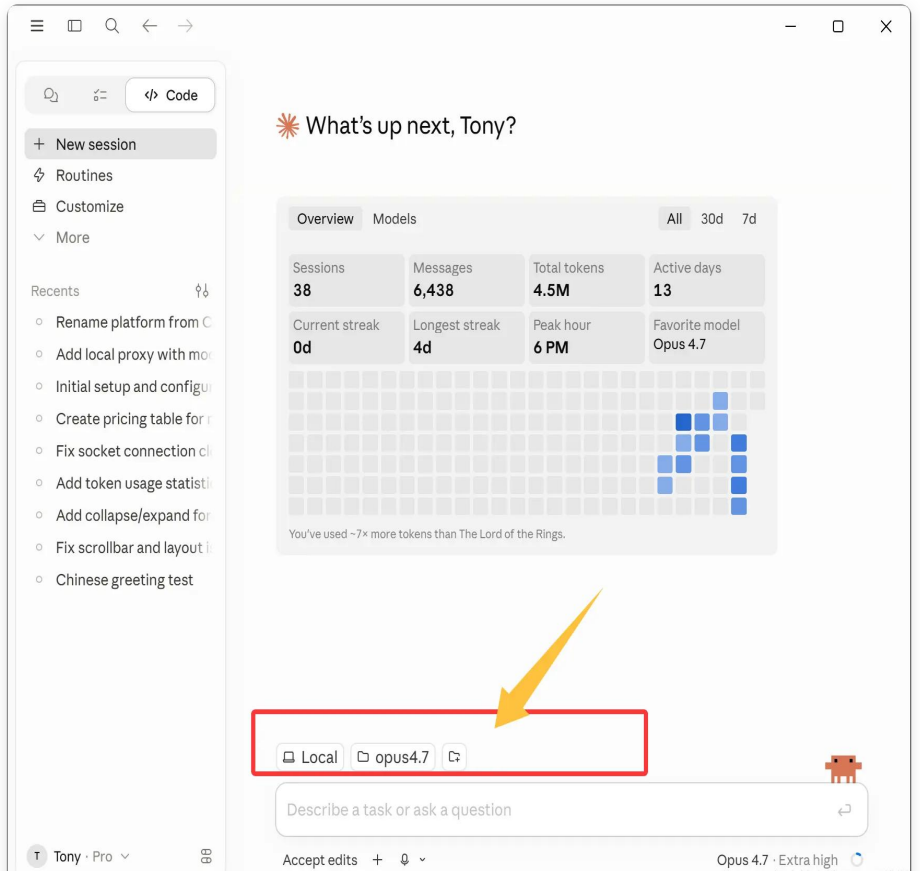
3.9.7 Git Worktree

■ Worktree的概念

在传统的 Git 使用模式中，一个仓库在同一时间只能检出一个分支到一个工作目录。如果需要切换到另一个分支工作，必须先将当前分支的修改暂存（git stash）或提交，再切换分支。这种方式在同时处理多个任务时显得繁琐。

Git Worktree（工作树）功能允许将同一个仓库的不同分支同时检出到多个独立的目录中，使开发者能够在不切换目录的情况下，同时在多个分支上工作。其核心思想是：一个目录绑定到仓库的一个分支，多个目录可以同时绑定到同一仓库的不同分支。

例如，在修复生产环境 bug 的同时，可以在另一个目录中继续开发新功能，两者互不干扰，且共享同一个 .git 对象数据库，节省磁盘空间。



3.9.7 Git Worktree

■ Worktree基本操作

（1）添加新的工作树

添加新的工作树的基本语法如下：

```
git worktree add <路径> <分支名>
```

具体用法如下：

```
# 在 ../project-hotfix 目录检出 hotfix 分支
git worktree add ../project-hotfix hotfix
# 创建新分支并检出到指定目录
git worktree add -b feature-new ../project-feature
```

3.9.7 Git Worktree

■ Worktree基本操作

（2）查看所有工作树

查看所有工作树的具体命令如下：

```
git worktree list
```

执行后会列出所有工作树的路径、当前提交哈希和分支名称。

（3）移除工作树

移除指定工作树的具体用法如下：

```
# 移除指定工作树（注意：这不会删除该分支，只是取消目录与分支的绑定）
git worktree remove ../project-hotfix
```

3.9.7 Git Worktree

■ 3.Worktree的典型使用场景

Git Worktree 在以下几种场景中特别有价值：

紧急 Bug 修复

当前正在 feature 分支开发新功能，生产环境突然发现严重 Bug。无需中断当前工作，直接创建一个指向 main 分支的 Worktree，在其中修复 Bug 并发布，然后继续原有的功能开发。

代码对比参考

需要参照旧版本的实现方式编写新代码时，可以将旧版本分支检出到另一个目录，在两个编辑器窗口中对比查看。

并行测试

同时运行不同分支的代码进行对比测试，无需频繁切换。

AI 辅助编程场景

在使用 AI 工具进行大规模代码修改时，可以在一个 Worktree 中让 AI 工作，同时在主 worktree 中保留稳定的代码副本，随时可以参考或回滚。

Part 10

SQLite数据库的安装和使用

3.10 SQLite数据库的安装和使用

SQLite简介

下载SQLite

安装SQLite

SQLite的
基本操作

常用SQLite命令
和技巧

3.10.1 SQLite简介

SQLite是一款开源的嵌入式关系型数据库管理系统（RDBMS）。与MySQL、PostgreSQL等传统数据库不同，SQLite以库的形式嵌入到应用程序中，直接读写本地磁盘上的单个数据库文件。

SQLite的核心特点包括：零配置、单文件存储、跨平台、轻量级、事务支持、标准SQL语法。适用场景包括学习和教学、嵌入式设备和移动应用、桌面应用程序以及快速原型开发。

SQLite的主要下载文件包括：sqlite-tools（包含sqlite3.exe命令行工具、sqldiff.exe数据库差异比较工具、sqlite3_analyzer.exe分析工具、sqlite-sync.exe同步工具）、sqlite-dll（包含动态链接库文件）。

3.10.2 下载SQLite

在Windows系统下手工安装SQLite，需要从SQLite官方网站下载所需的工具包。

打开浏览器，访问SQLite官方网站（https://www.sqlite.org/），点击顶部导航栏的"Download"链接进入下载页面。在下载页面的"Precompiled Binaries for Windows"部分，下载以下两个文件：

- sqlite-tools-win-x64-*.zip：包含命令行工具的压缩包
- sqlite-dll-win-x64-*.zip：包含SQLite动态链接库的压缩包

其中，"*"代表版本号，建议下载最新的稳定版本。将两个压缩包保存到本地磁盘。

3.10.3 安装SQLite

SQLite的安装过程通过手动解压文件和配置环境变量的方式完成，不提供图形化的安装向导。

在系统盘（通常为C盘）的根目录下创建“C:\sqlite”目录，使用Windows自带的解压功能或第三方压缩工具，将下载的两个压缩包解压到“C:\sqlite”目录中。解压完成后，“C:\sqlite”目录下应该至少包含以下关键文件：

- sqlite3.exe：SQLite命令行工具，用于交互式操作数据库
- sqldiff.exe：数据库差异比较工具，用于比较两个SQLite数据库的差异
- sqlite3_analyzer.exe：数据库分析工具，分析数据库文件的结构和空间使用情况
- sqlite3-rsync.exe：数据库同步工具

3.10.3 安装SQLite

为了能够在Git Bash中使用“sqlite3”命令，需要将SQLite的安装目录添加到环境变量中。在Git Bash终端中执行以下命令编辑“.bashrc”文件：

```
vim ~/.bashrc
```

在打开的文件末尾添加以下内容：

```
export PATH="/c/sqlite:$PATH"
```

保存文件后关闭编辑器，在Git Bash中执行如下命令让配置生效：

```
source ~/.bashrc
```

重启Git Bash，在新的Git Bash窗口中输入以下命令验证SQLite是否安装成功：

```
sqlite3 --version
```

如果安装成功，命令行窗口将输出SQLite的版本信息。

3.10.4 SQLite的基本操作

安装完成后，我们可以通过“sqlite3”命令行工具进行数据库的基本操作。以下通过一个完整的示例，演示从打开数据库到执行基本数据操作的流程。

在Git Bash的命令行界面中执行以下命令，打开名为“example.db”的数据库并进入交互式命令行界面：

```
sqlite3 example.db
```

SQLite会自动处理数据库文件，如果当前目录下不存在“example.db”文件，SQLite会自动创建；如果文件已存在，则直接打开。进入成功后，命令行提示符将变为`sqlite>`，表示已进入SQLite的交互式界面。

3.10.4 SQLite的基本操作

在SQLite交互式界面中，使用CREATE TABLE语句创建表。以下示例创建一个用于存储学生信息的`students`表：

```
CREATE TABLE students (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    age INTEGER,
    major TEXT,
    enrollment_date TEXT
);
```

3.10.4 SQLite的基本操作

表创建完成后，可以使用“.tables”命令查看当前数据库中的所有表，使用“.schema students”命令查看表的结构。除了点命令之外，SQLite还提供了PRAGMA语句用于查询数据库的元信息，以下命令可以获取students表的字段结构：

```
PRAGMA table_info(students);
```

还可以使用“PRAGMA table_list”查询当前数据库中的所有表。使用INSERT INTO语句向表中插入数据：

```
INSERT INTO students (id, name, age, major, enrollment_date) VALUES (1, '张三', 20, '计算机科学与技术', '2024-09-01');
INSERT INTO students (id, name, age, major, enrollment_date) VALUES (2, '李四', 21, '软件工程', '2023-09-01');
INSERT INTO students (id, name, age, major, enrollment_date) VALUES (3, '王五', 19, '人工智能', '2024-09-01');
```

3.10.4 SQLite的基本操作

在SQLite中，单条SQL语句以分号结尾。如果在交互式界面中输入的命令没有以分号结尾，SQLite会认为命令尚未结束，提示符会变为`...>`。

使用SELECT语句查询表中的数据。最简单的查询方式是查询表中的所有记录：

```
SELECT * FROM students;
```

如果需要以更友好的格式显示查询结果，可以设置输出模式和显示表头：

```
.mode column
.headers on
SELECT * FROM students;
```

3.10.4 SQLite的基本操作

设置列模式和表头显示后，查询结果将以对齐的列格式显示，并且第一行显示字段名称。也可以根据条件筛选数据，例如查询年龄大于等于20岁的学生：

```
SELECT name, age FROM students WHERE age >= 20;
```

使用UPDATE语句更新表中的数据。以下示例将编号为1的学生的专业修改为"数据科学"：

```
UPDATE students SET major = '数据科学' WHERE id = 1;
```

更新操作中WHERE子句的作用非常重要，如果省略WHERE子句，将更新表中所有记录的对应字段。使用DELETE语句删除表中的数据，以下示例删除编号为3的学生记录：

```
DELETE FROM students WHERE id = 3;
```

3.10.5 常用SQLite命令和技巧

在日常使用SQLite的过程中，可以使用点命令显著提高操作效率。SQLite交互式界面中的点命令是SQLite特有的元命令，以点号开头，用于控制输出格式、查看数据库信息等。常用的点命令如表所示。

命令	说明
`.help`	显示所有可用的点命令及其说明
`.tables`	列出当前数据库中所有表的名称
`.schema [表名]`	显示表的创建语句，不指定表名则显示所有表
`.databases`	列出当前连接的所有数据库文件及其路径
`PRAGMA table_info(表名)`	查询指定表的字段结构（列编号、列名、类型、非空约束、默认值、主键）
`PRAGMA table_list`	查询当前数据库中所有表的名称和属性
`.mode column`	设置输出为列对齐模式，便于阅读查询结果
`.headers on`	显示查询结果的字段名称（表头）
`.read 文件名`	从指定文件中读取并执行SQL语句
`.exit` 或 `.quit`	退出SQLite交互式界面

3.11本章小结

本章系统介绍了AI编程工具的选型和使用方法，帮助读者建立实操能力。

AI编程工具的三种形态

命令行工具轻量灵活、IDE集成功能丰富、智能体形态代表前沿方向。不同形态适合不同场景，开发者应根据需要选择。

工具选型

本书选择TRAE作为主要教学工具，选择依据包括中文优化、多种工作模式、国内访问稳定等。

大模型选择

不同大模型在代码能力上存在差异，选择合适的大模型可以显著提升编程效率。SWE-bench是评估模型代码能力的权威基准。

技术栈选择

主流技术栈拥有丰富的训练数据，AI对其理解程度更高；小众技术可能超出AI的能力范围。选择AI友好的技术栈是有效使用AI编程工具的前提。

谢谢！

林子雨　副教授　厦门大学

附录A：主讲教师林子雨简介

主讲教师：林子雨

单位：厦门大学计算机科学与技术系

E-mail: ziyulin@xmu.edu.cn

个人网页：http://dblab.xmu.edu.cn/post/linziyu

数据库实验室网站：http://dblab.xmu.edu.cn

- 林子雨，男，1978年出生，博士（毕业于北京大学），全国高校知名大数据与人工智能教师，入选“2021年高校计算机专业优秀教师奖励计划”。现为厦门大学计算机科学与技术系副教授，厦门大学信息学院实验教学中心主任，曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会执行委员，中国计算机学会信息系统专业委员会执行委员。
- 国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度、2017年度、2020年度、2023年度和2026年度厦门大学教学类奖教金获得者，荣获2024年福建省高等教育教学成果奖特等奖（个人排名第七）、2022年福建省高等教育教学成果奖特等奖（个人排名第一）、2018年福建省高等教育教学成果奖二等奖（个人排名第一）、2018年国家精品在线开放课程、2021年国家级线上一流本科课程、2020年国家级线上一流本科课程。编著出版了20余本大数据与人工智能系列教材，被国内1000多所高校采用，3本教材入选教育部“十四五”普通高等教育本科国家级规划教材；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过400万次，累计访问量超过3200万次。大数据系列MOOC课程入选“2023年教育部国家智慧教育公共服务平台应用典型案例”。2025年发布4个大模型科普报告，全网累计浏览量超过1000万，并应邀为高校、政府和企业做170余场大模型讲座。
- 主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题、1项2016年教育部产学协作育人项目、1项2024年教育部产学协作育人项目。

附录B：大数据学习路线图

大数据学习路线图访问地址：http://dblab.xmu.edu.cn/post/10164/

附录C：林子雨大数据系列教材

了解全部教材信息：http://dblab.xmu.edu.cn/post/bigdatabook/

附录D：林子雨人工智能通识系列教材

厦门大学林子雨人工智能通识系列教材

被众多高校采用，满足不同高校差异化教学需求

了解全部教材信息：http://dblab.xmu.edu.cn/post/ai-book/

附录E：林子雨Python系列教材

了解全部教材信息：https://dblab.xmu.edu.cn/post/python-books/

谢谢！

林子雨　副教授

厦门大学