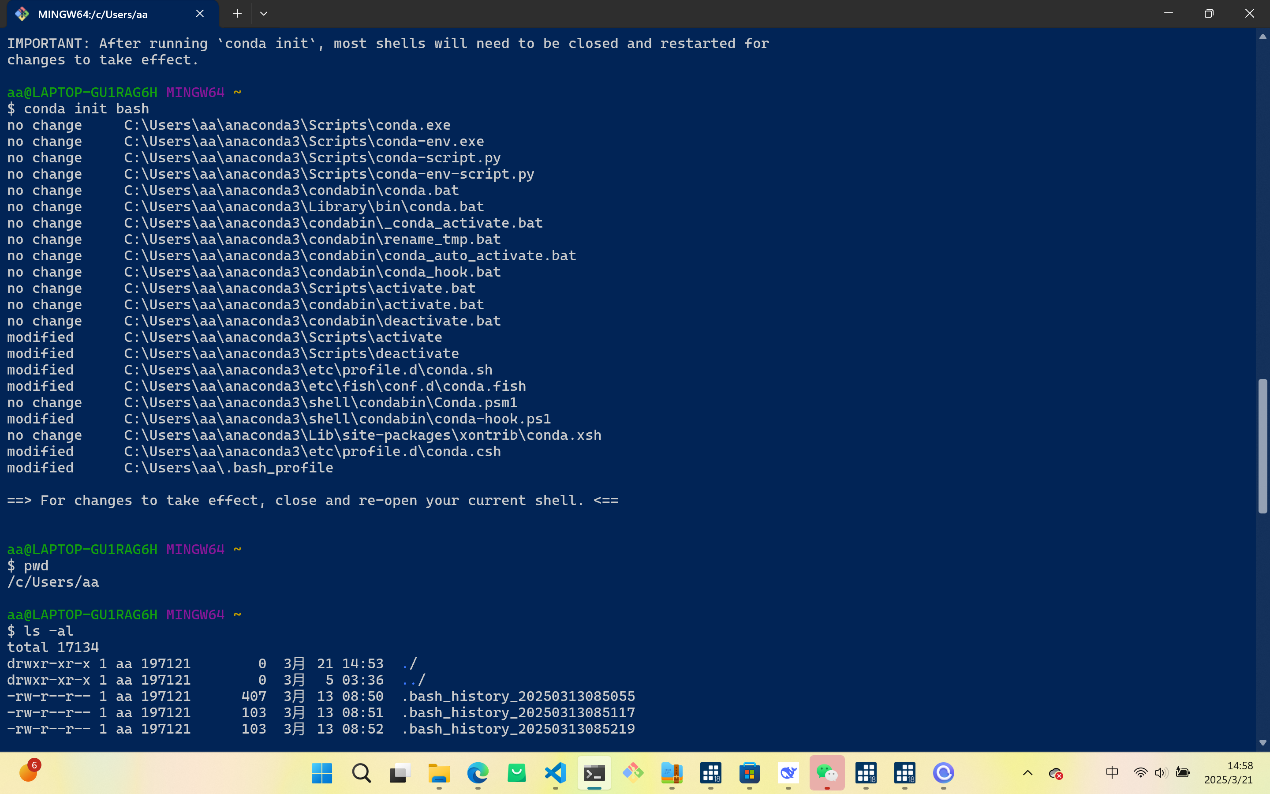
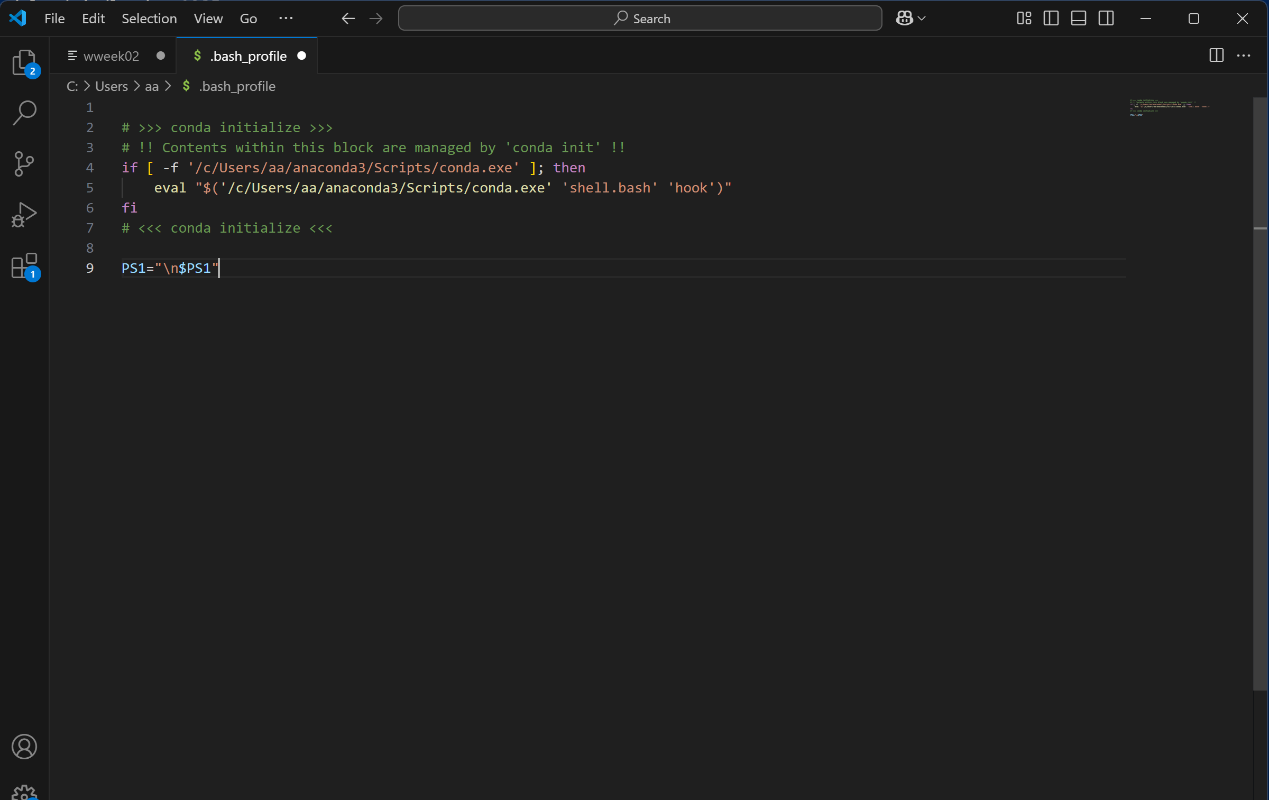
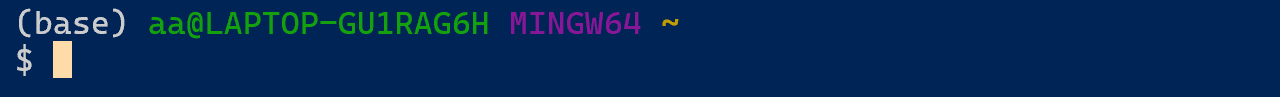
1.在终端配置好 Conda Init，启动终端后在提示符前能够看到 base

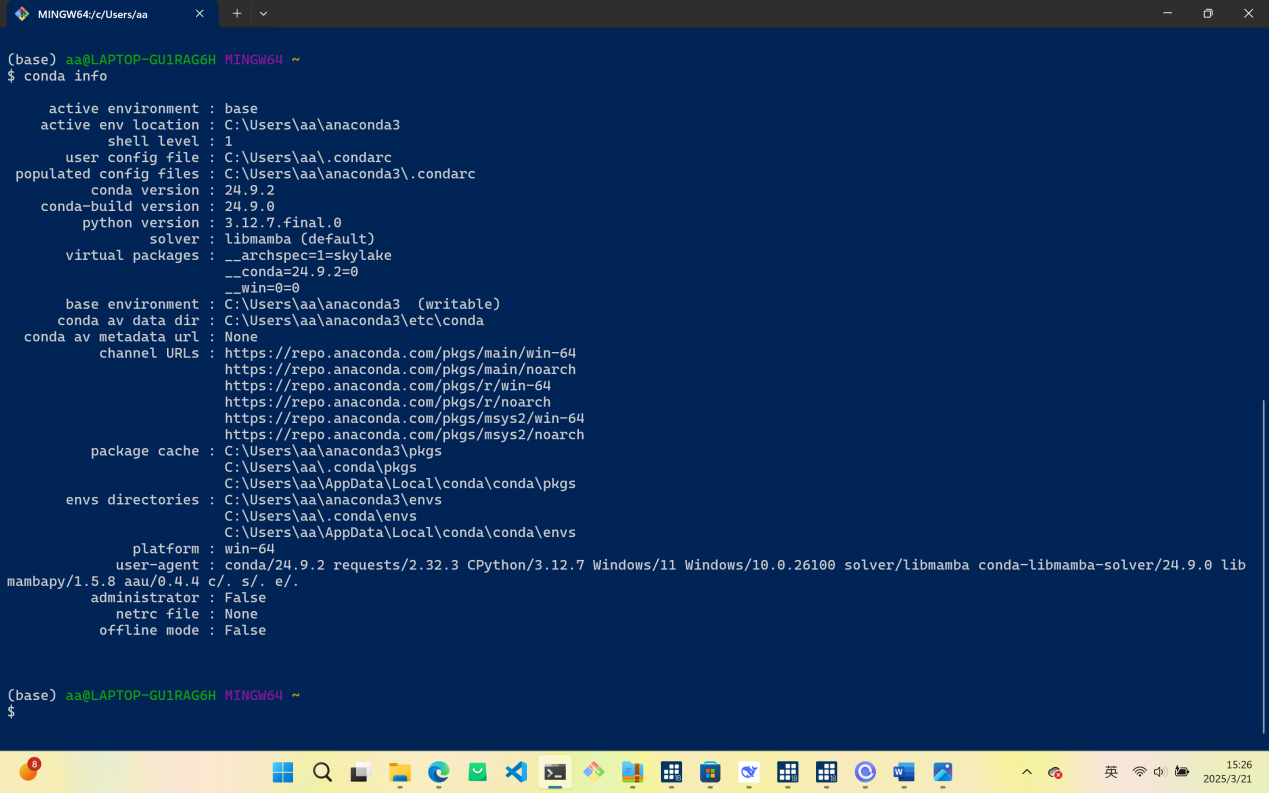




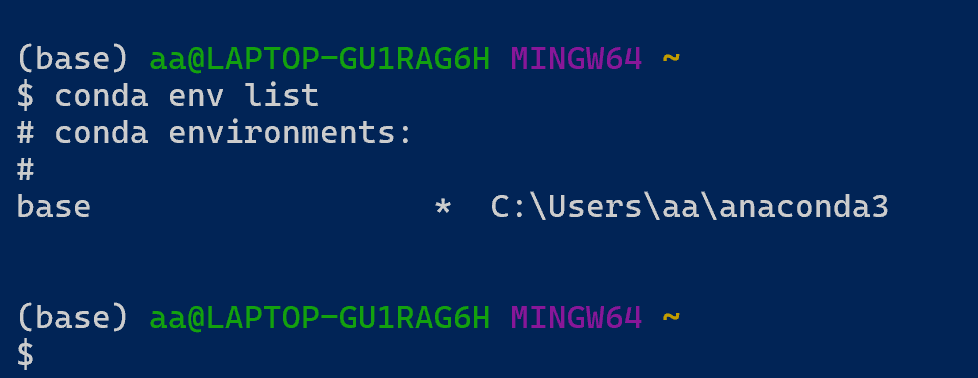




2.使用 conda info 命令查看本机 Conda 的配置信息

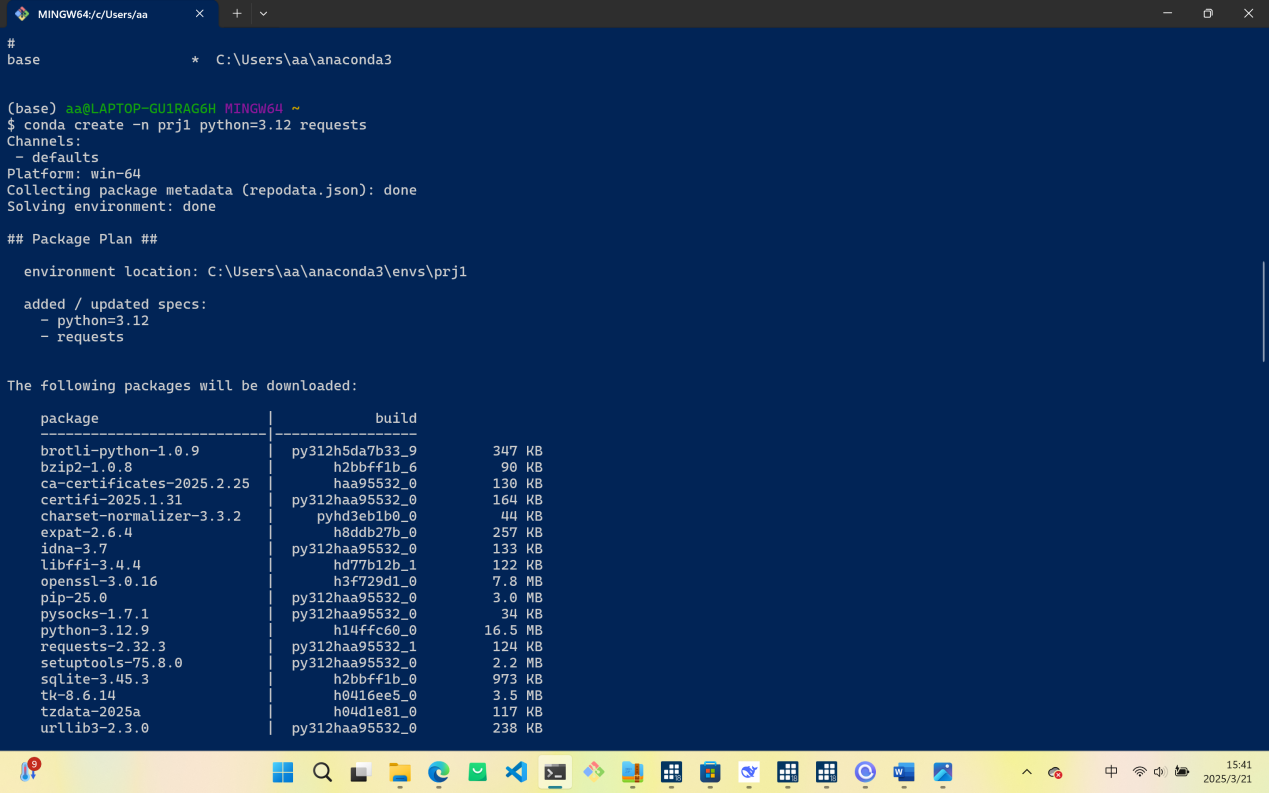


3.使用 conda env list 命令查看已有的 Conda 环境的名称和路径

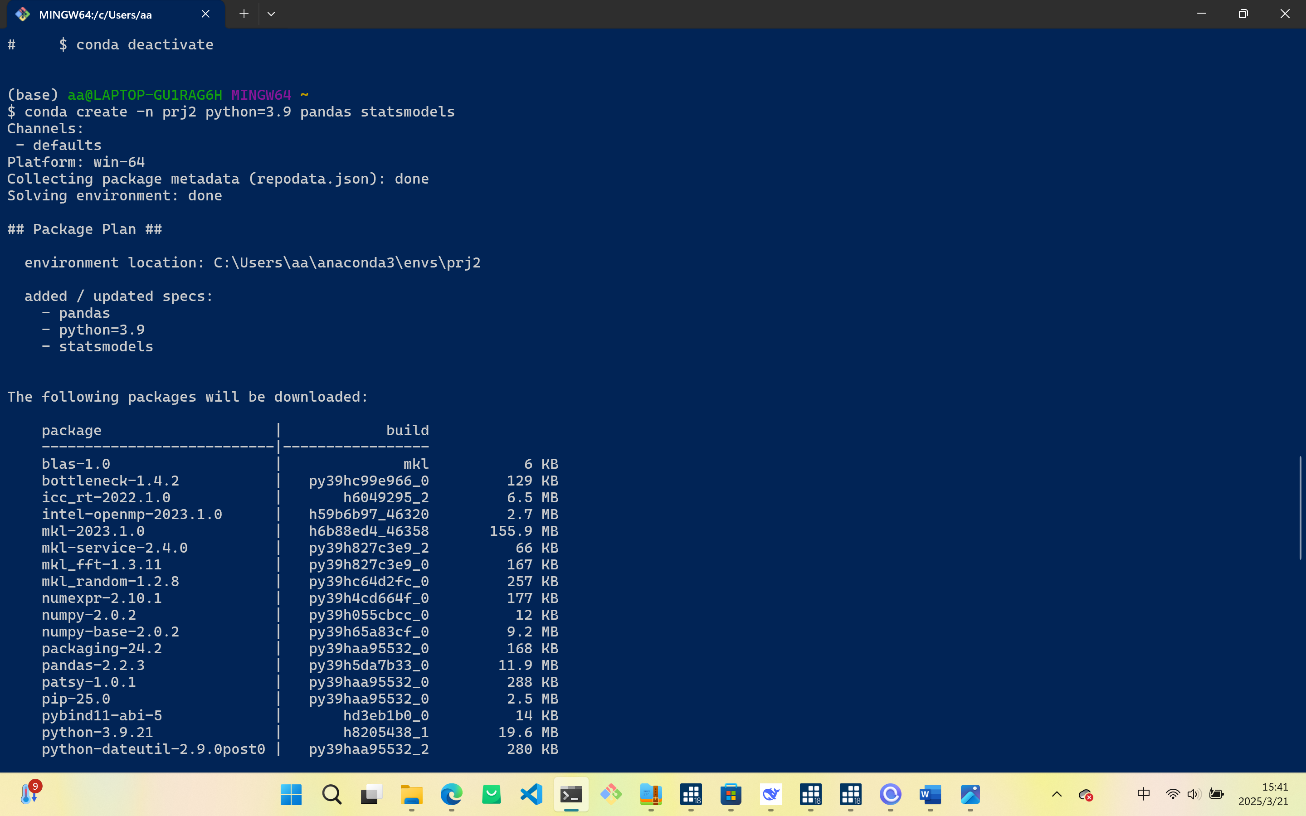


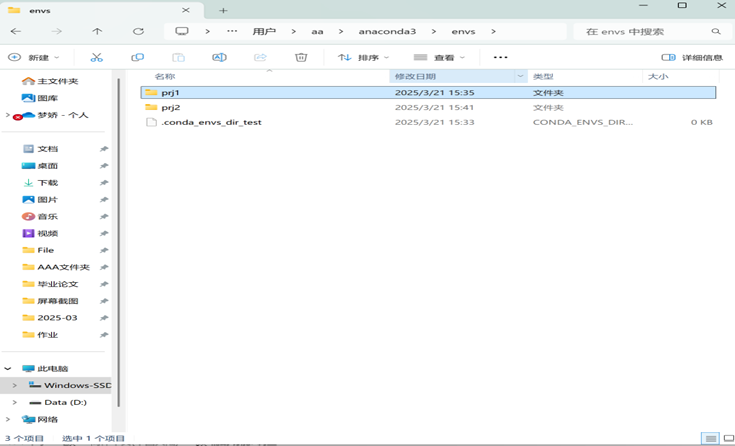
Conda 环境的概念：Conda 环境是一个独立的 Python 运行环境，它允许用户在同一台计算机上创建多个相互隔离的环境，每个环境可以有不同的 Python 版本、库及其依赖项。

4.使用 conda create 命令创建两个Conda环境，一个里面装 Python3.12 和 requests 软件包

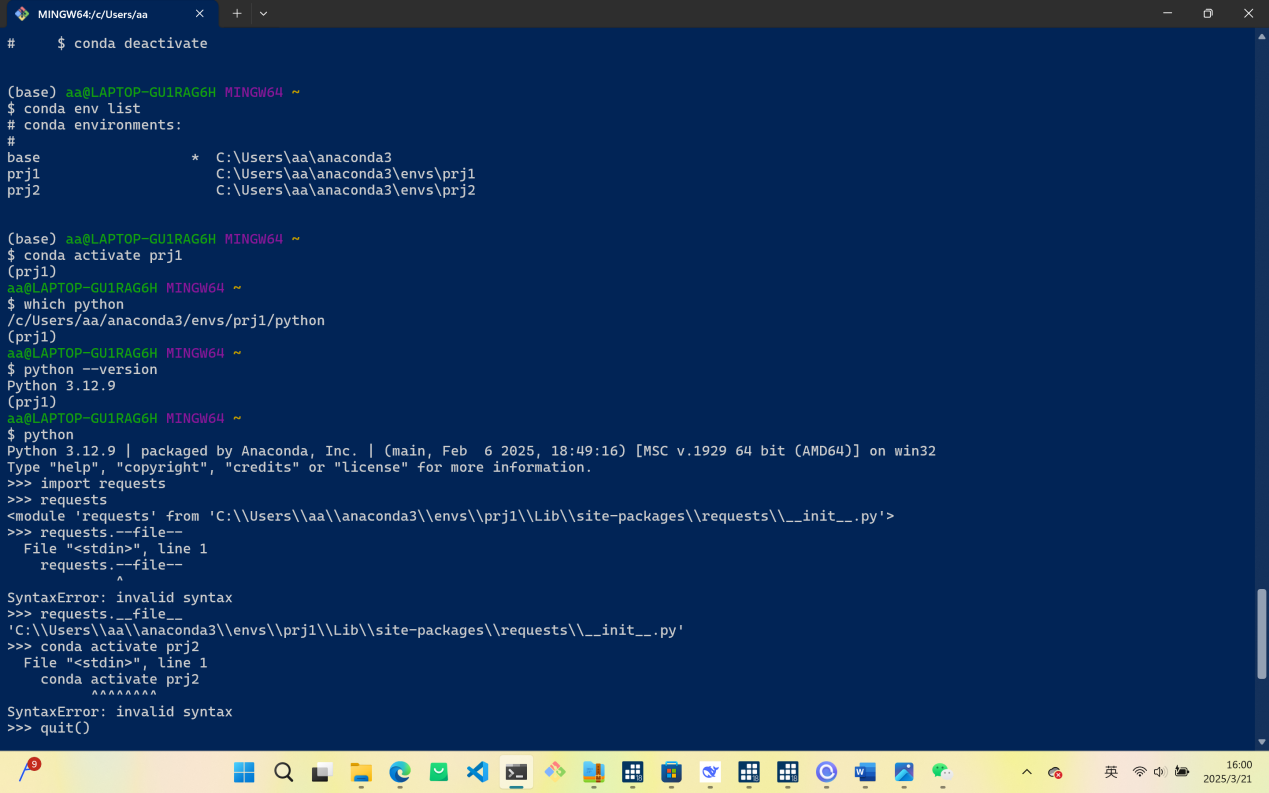


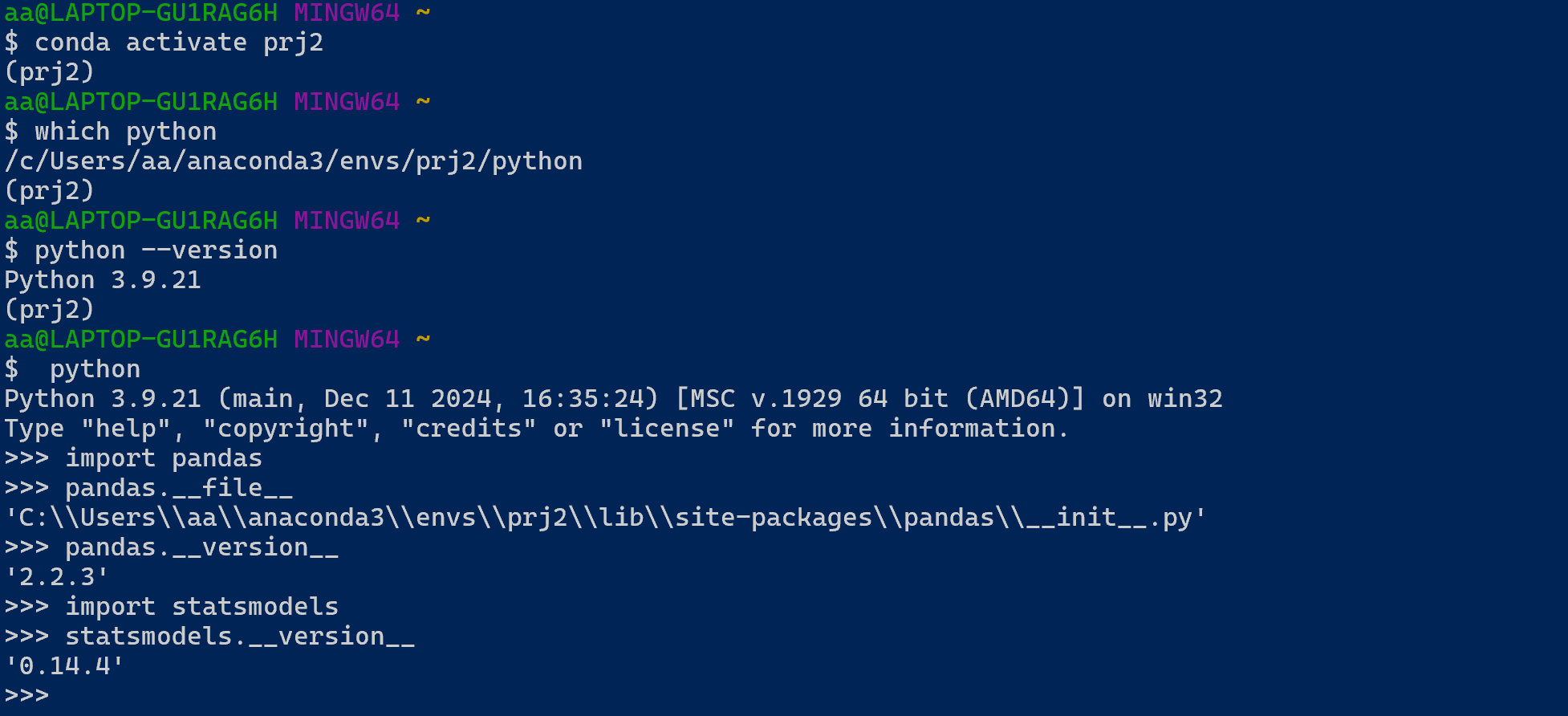
另一个里面安装 Python 3.9、pandas 和 statsmodels 软件包





在终端里切换 Conda 环境，验证 Python 和软件包的版本

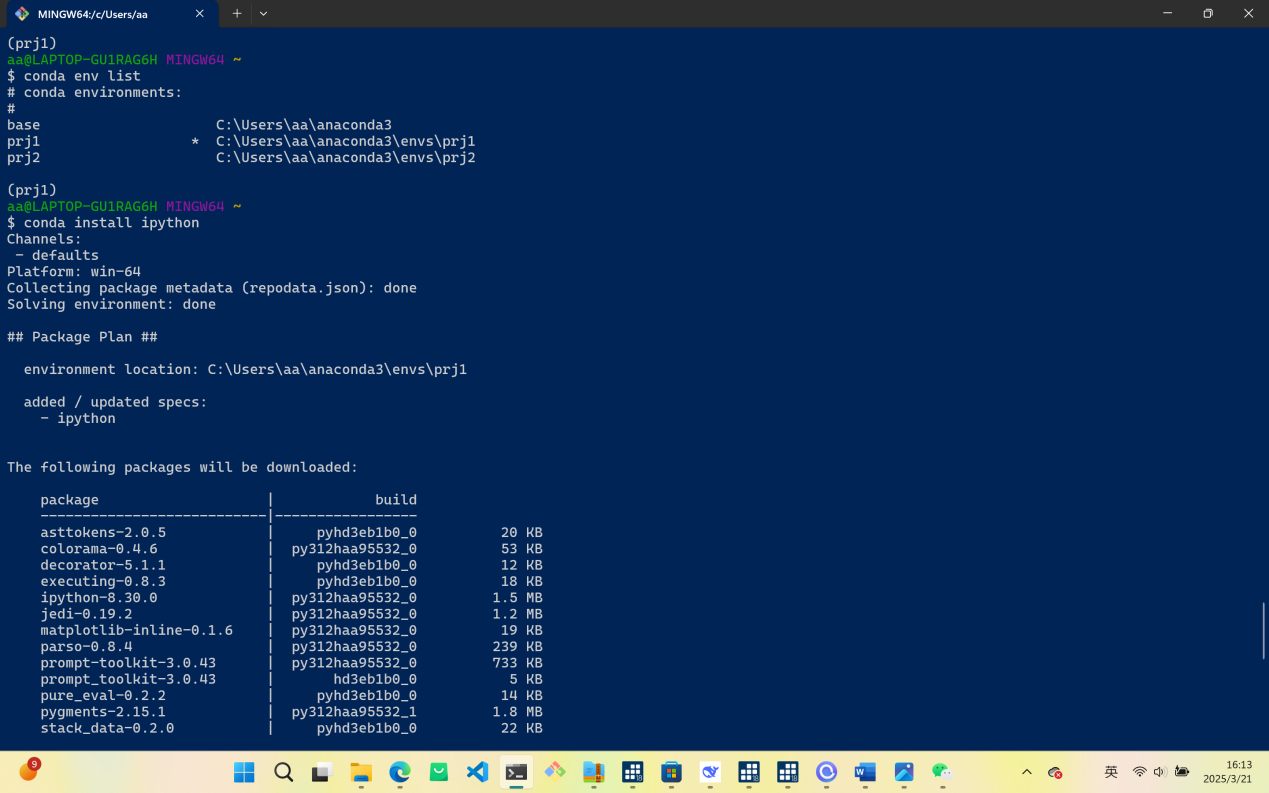




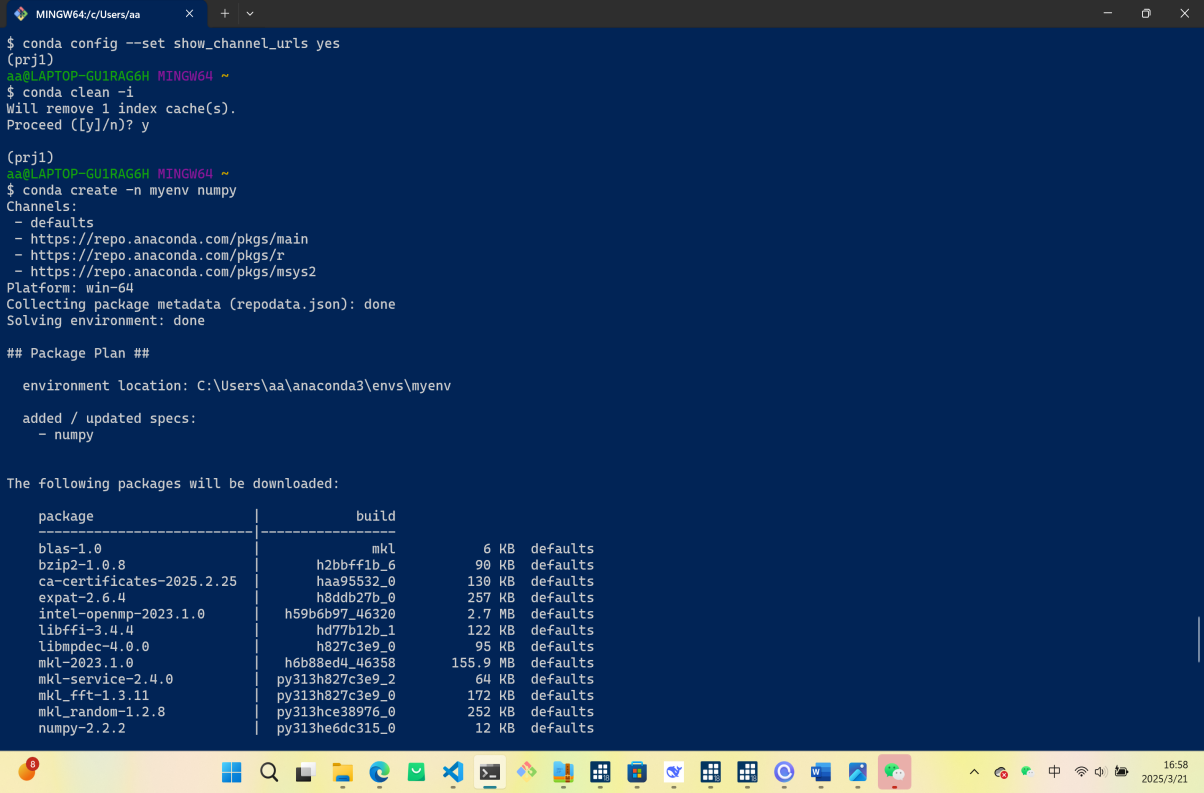
5．使用 conda list 命令显示 Conda 环境里的软件包列表及其版本信息



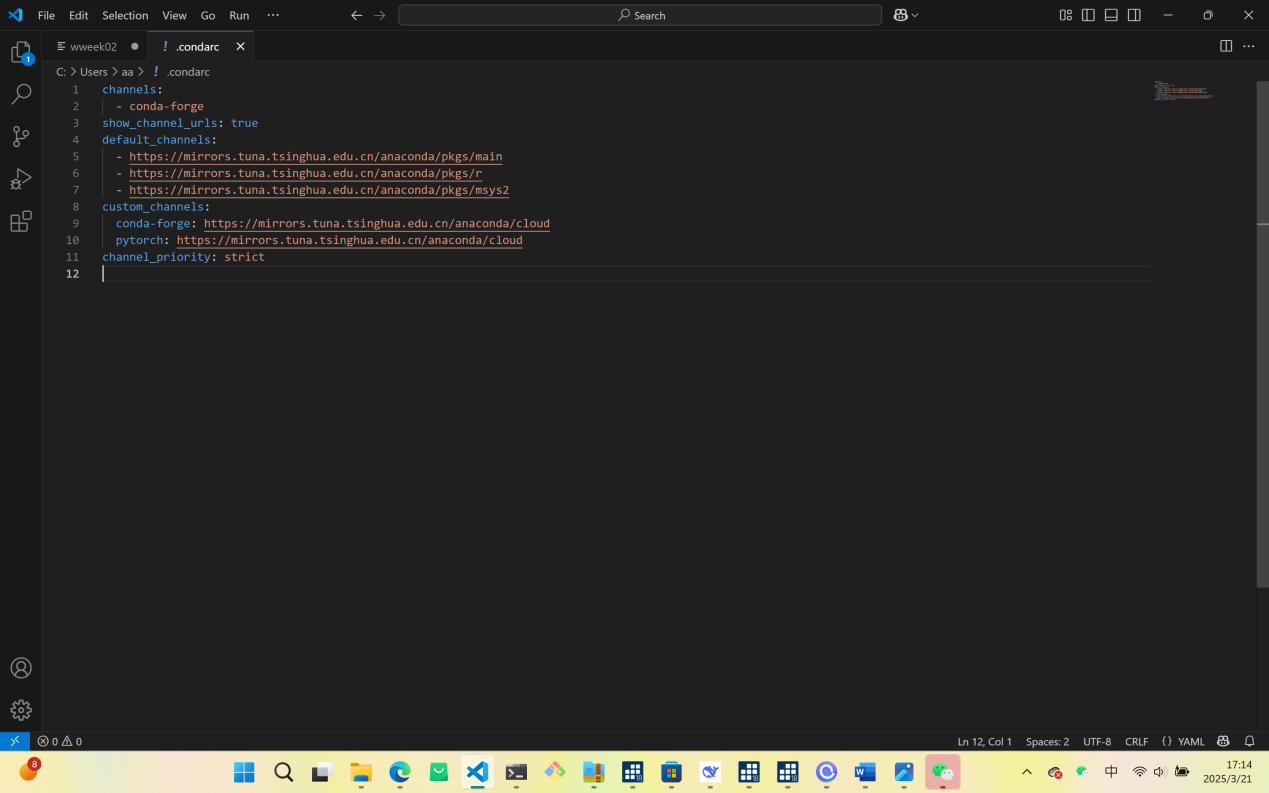
6.使用 conda install 命令往 Conda 环境里安装软件包，并验证版本



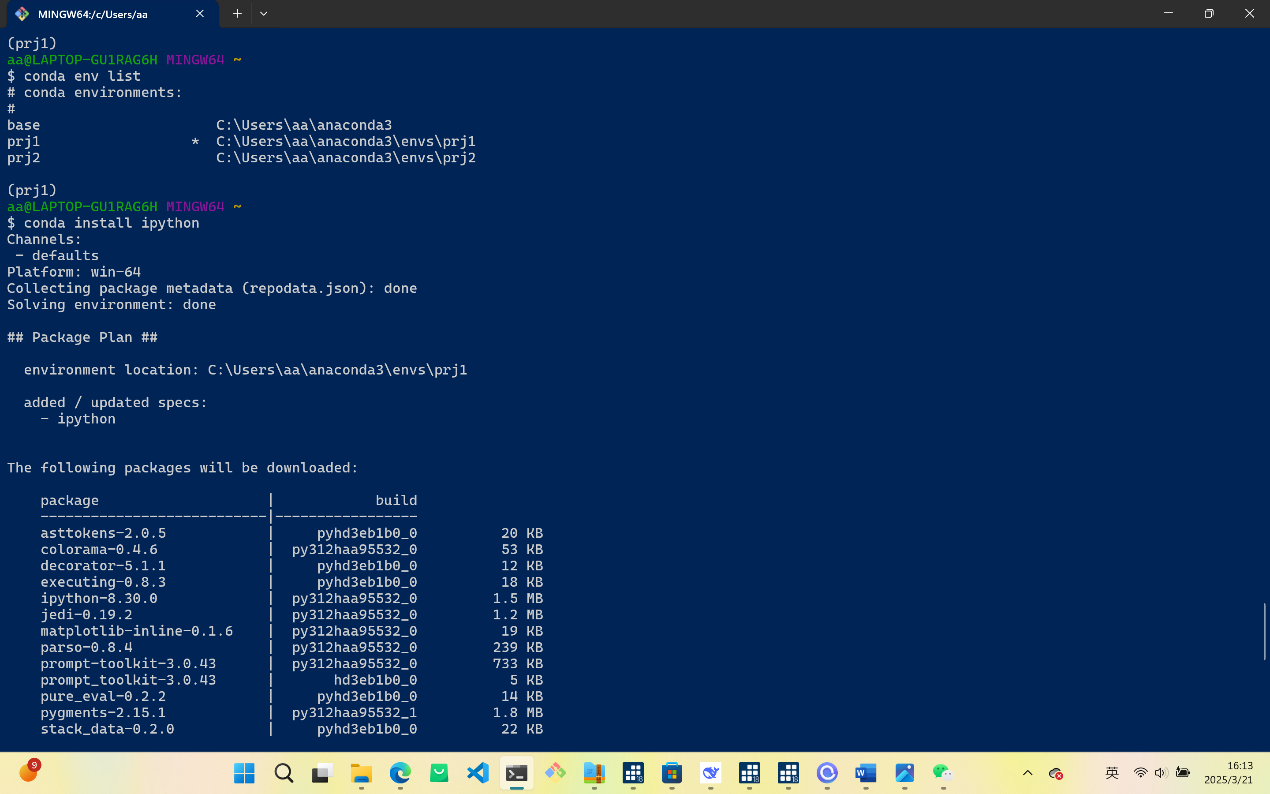
7. 配置 Anaconda 清华镜像



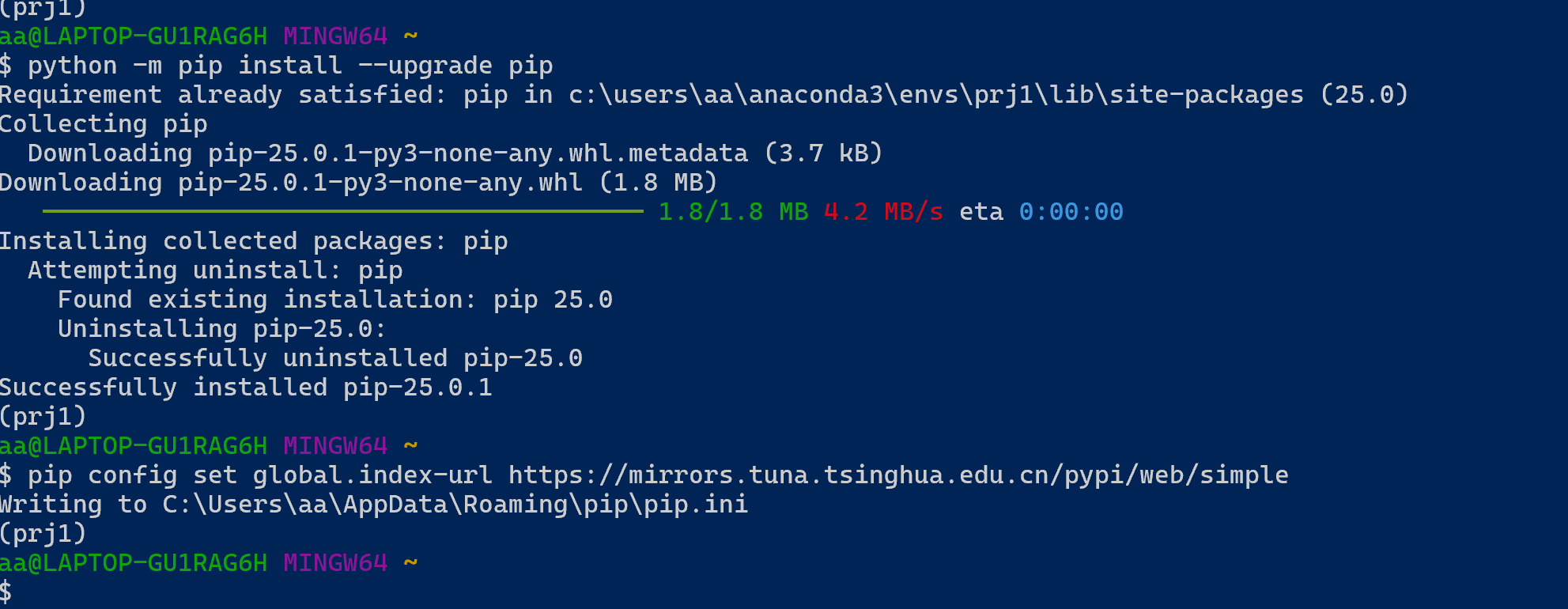
将 conda-forge 设置为默认 Channel

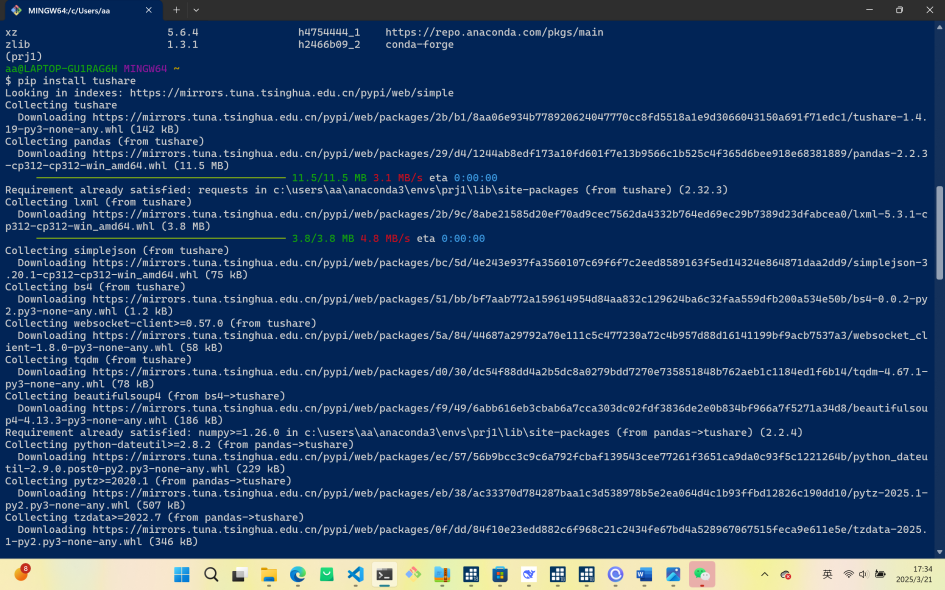


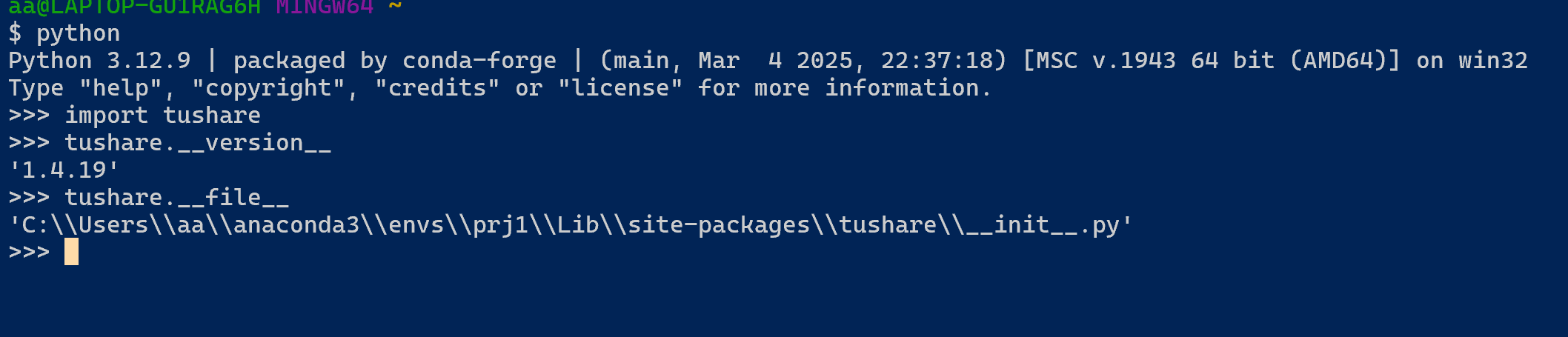
8. 使用 pip install 命令往 Conda 环境里安装 Python 软件包，并验证版本



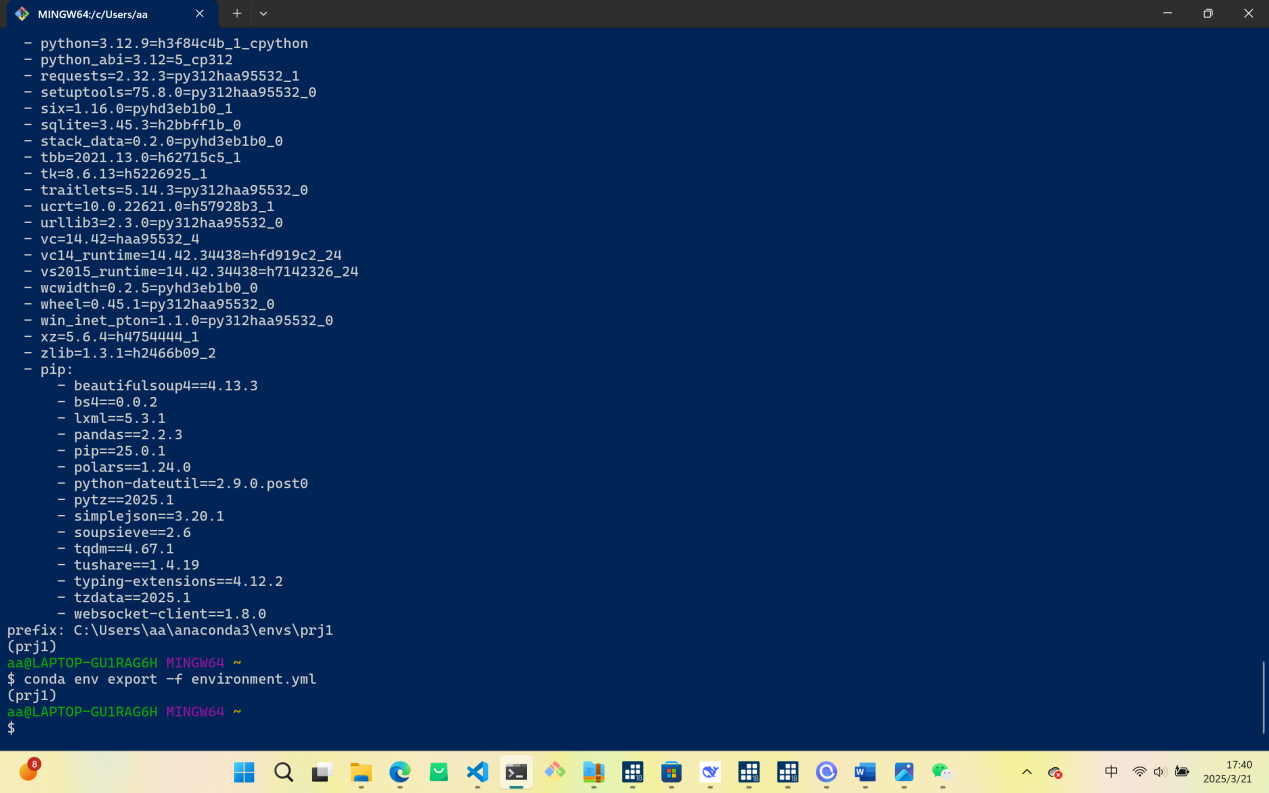
9.配置 PyPI 清华镜像，加快 pip install 安装软件包的速度





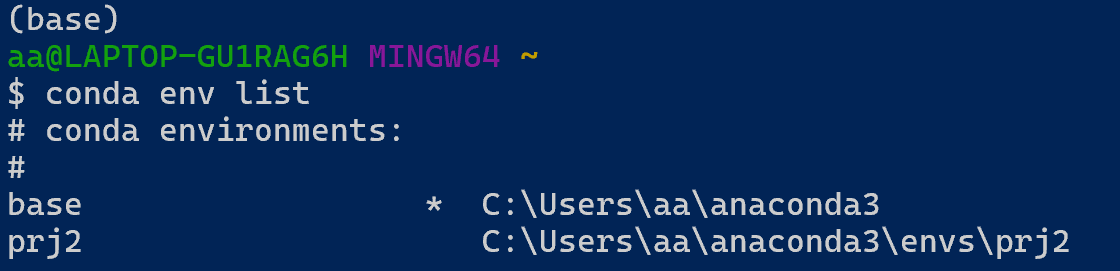


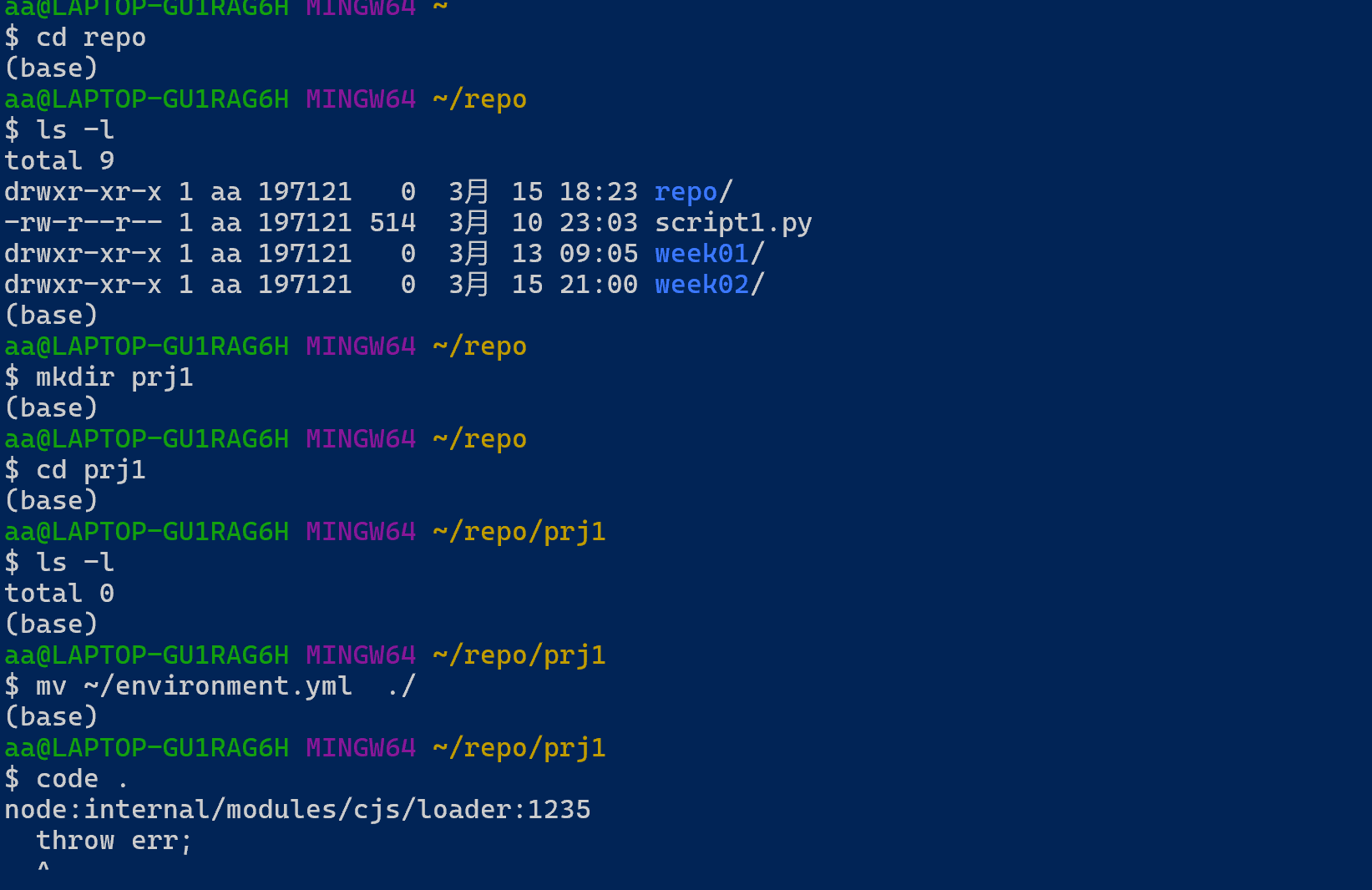
10. 导出 environment.yml Conda 环境配置文件



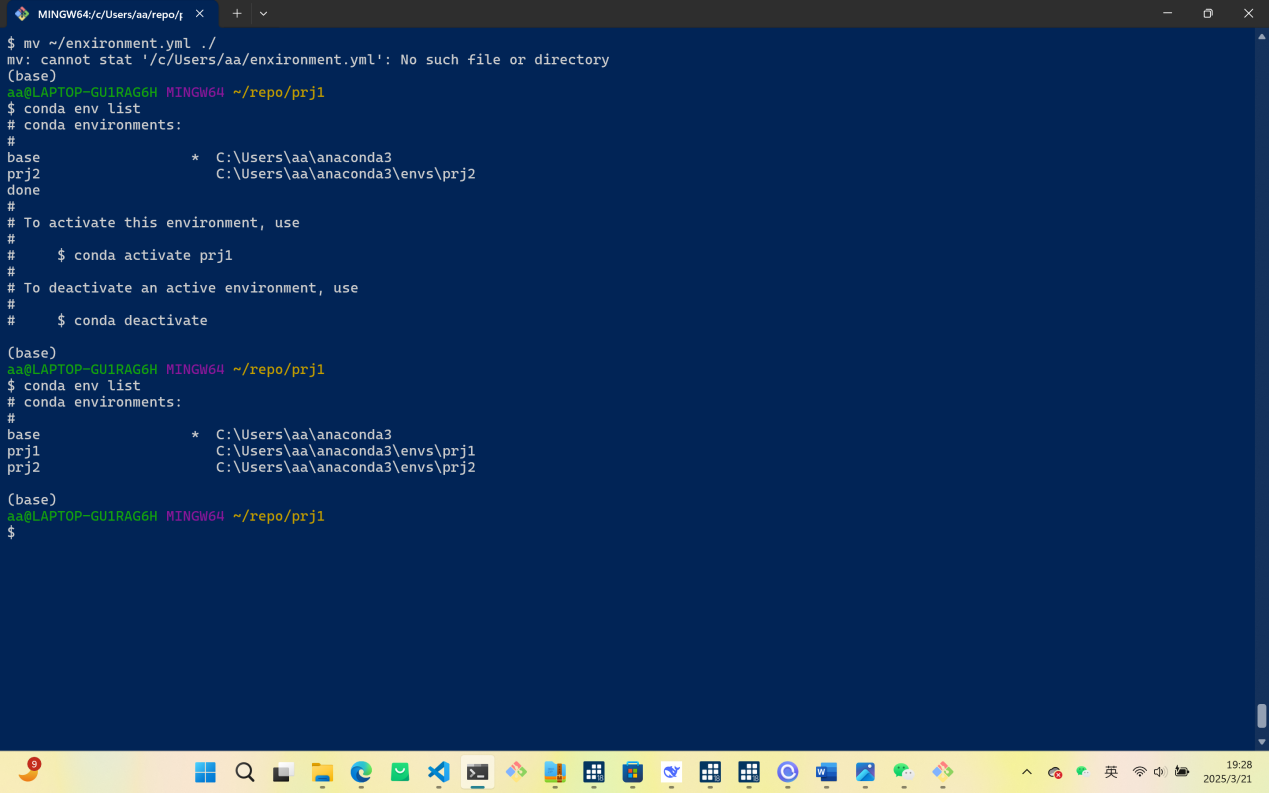
删除 Conda 环境







用 environment.yml 配置文件重建 Conda 环境



11. 理解 Conda 与 Python 的关系

**Python 环境管理**

Conda 最常用的功能之一就是管理 Python 环境。它可以创建多个独立的Python环境，每个环境都可以有不同的 Python 版本和软件包

**Python 包管理**

Conda 可以用于安装、更新和删除 Python 包。它拥有自己的包仓库，里面包含了大

量的 Python 包以及其他语言的软件包。

**Python 版本兼容性**  
由于 Conda 可以方便地创建不同 Python 版本的环境，因此在处理 Python 版本兼容性问题时非常有用。不同的 Python 版本可能对某些库或代码的支持有所不同，当你遇到这种情况时，就可以使用 Conda 快速切换到合适的 Python 版本环境中进行开发和测试。  
**跨平台支持**  
Conda 和 Python 都具有良好的跨平台性。Conda 可以在 Windows、Linux 和 macOS 等多种操作系统上使用，并且可以为不同操作系统的 Python 环境提供一致的管理方式。这使得开发者可以在不同的操作系统上轻松地部署和管理 Python 项目。

理解 Conda-Forge 与 Conda 的关系

Conda-Forge 和 Conda 存在着紧密的联系，前者依托于后者，同时又为后者提供了重要补充。Conda 是一个通用的包管理与环境管理系统，它可以在不同操作系统上管理软件包及其依赖，并且创建、切换和删除独立的环境。Conda-Forge 是基于 Conda 构建的一个社区驱动型包通道（channel）。这意味着 Conda-Forge 的正常运作依赖于 Conda 提供的基础功能，比如包的安装、更新和卸载等操作，都要借助 Conda 的命令来完成。

理解 Python 解释器

Python 解释器是一种能够执行 Python 代码的软件工具，它是将人类编写的 Python 代码转换为计算机能够理解和执行的机器语言的桥梁。Python 解释器按照顺序读取 Python 源代码中的字符，将其解析为一个个的语法单元，然后根据 Python 语言的语法规则和语义对这些单元进行解释和执行。在执行过程中，解释器会将代码中的变量、函数、类等元素加载到内存中，并按照代码的逻辑进行相应的操作，如计算表达式、调用函数、控制流程等。

第三方软件包：Python 标准库提供了许多基础功能，但在实际开发中，为了提高开发效率、实现特定功能，开发者往往会使用第三方软件包。这些软件包由不同的开发者或组织创建和维护，可满足各种不同的需求。

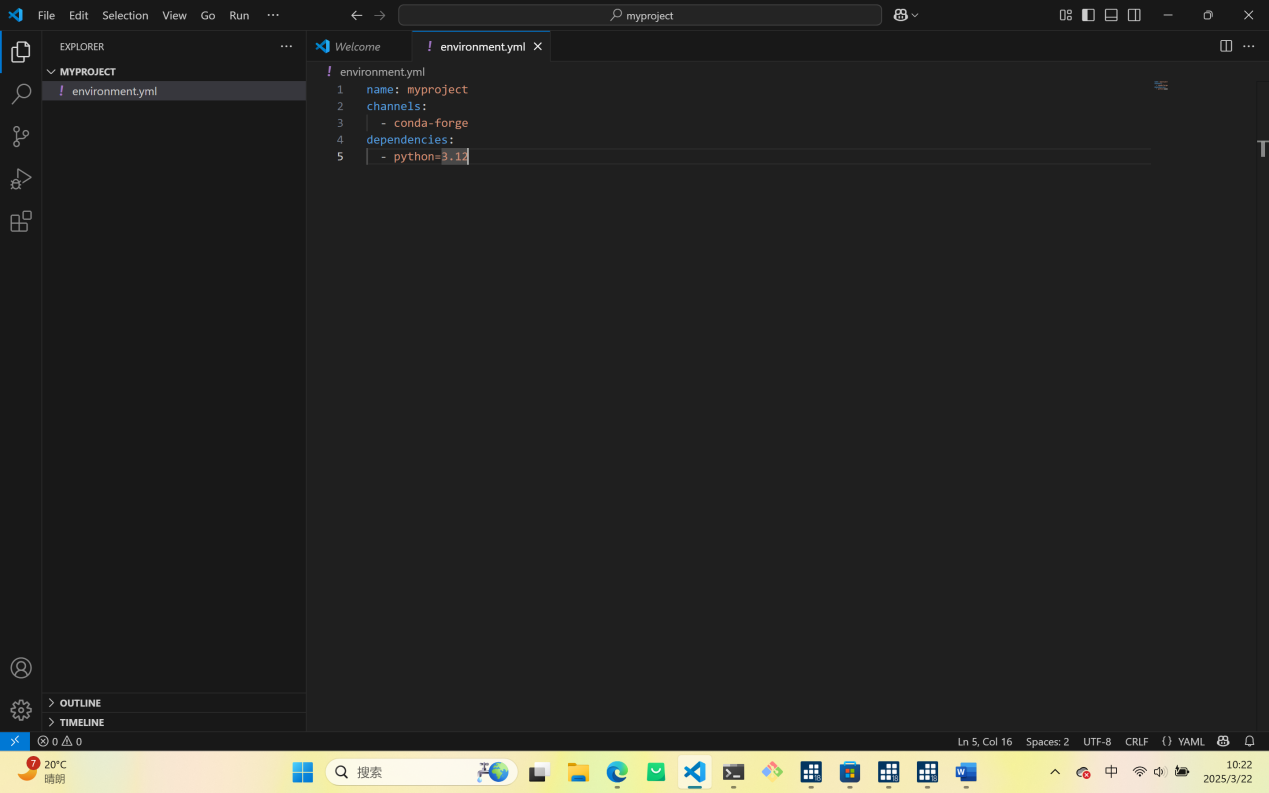
PyPI 软件仓库：Python Package Index（PyPI）是 Python 官方的软件包仓库，也被称为奶酪店（The Cheese Shop）。它是一个集中存储 Python 第三方软件包的地方，开发者可以将自己开发的软件包上传到 PyPI，其他开发者则可以从 PyPI 下载和使用这些软件包。

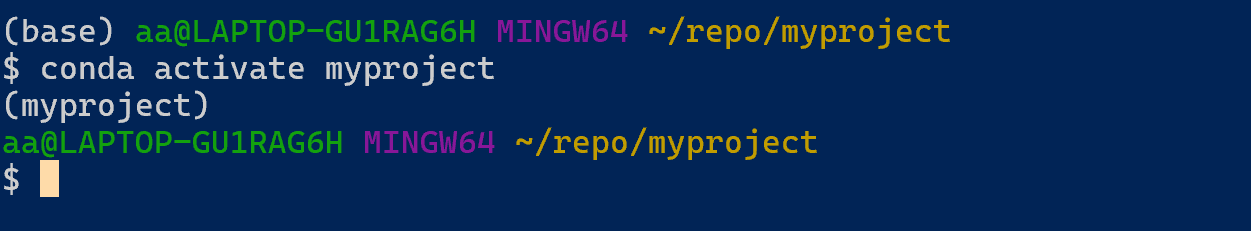
程序/软件包的路径问题：不同的 Python 环境和安装方式会导致软件包安装在不同的路径：

**全局环境**：在全局 Python 环境中使用 pip 安装的软件包通常会被安装到 Python 解释器的 site-packages 目录下。

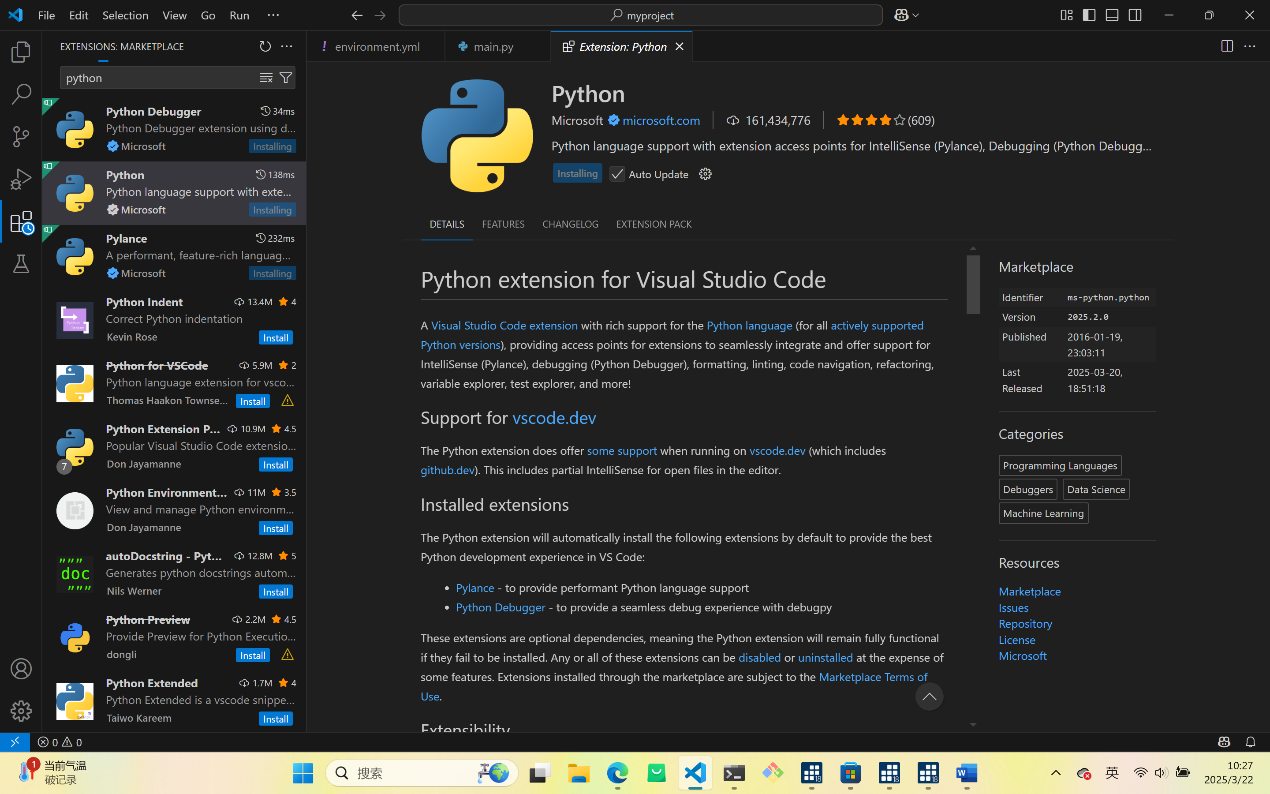
**虚拟环境**：当使用虚拟环境（如 venv 或 conda 环境）时，软件包会被安装到虚拟环境的 site-packages 目录下。

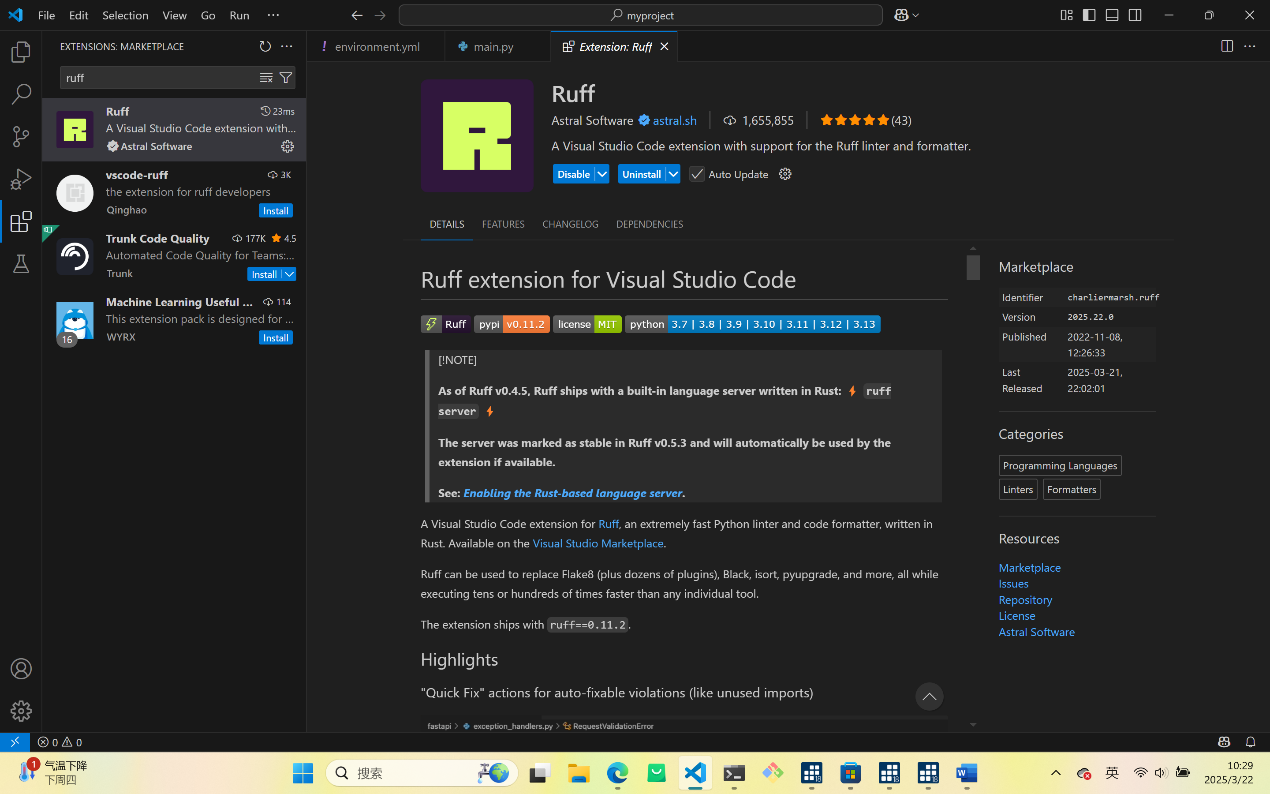
12. 按照教程创建项目目录，

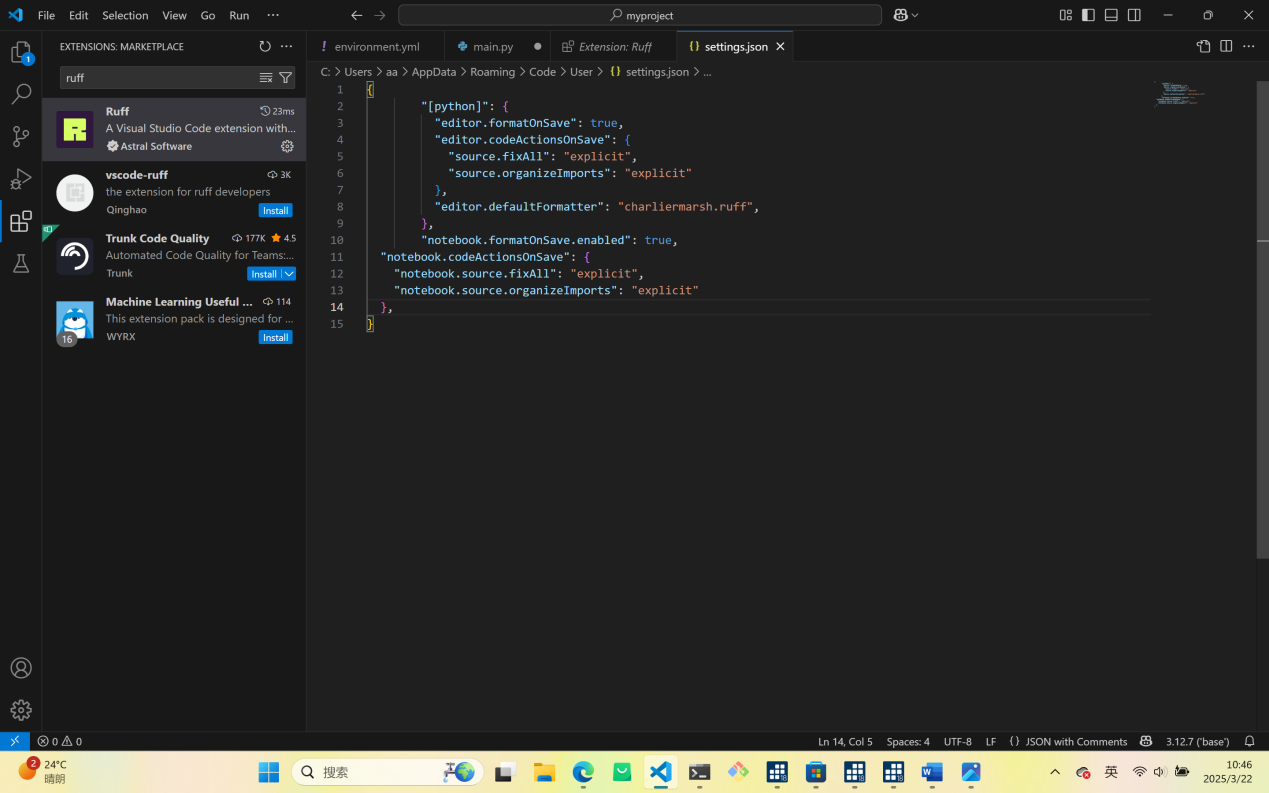


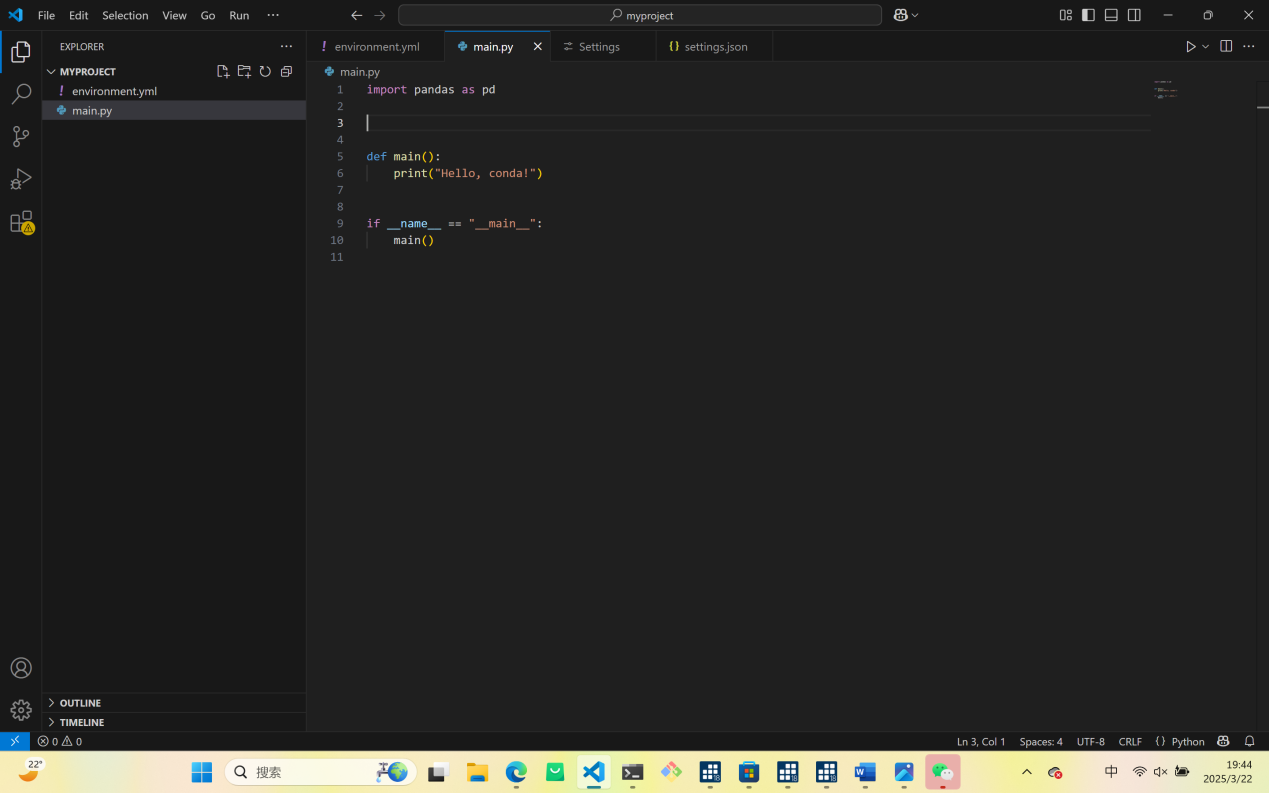


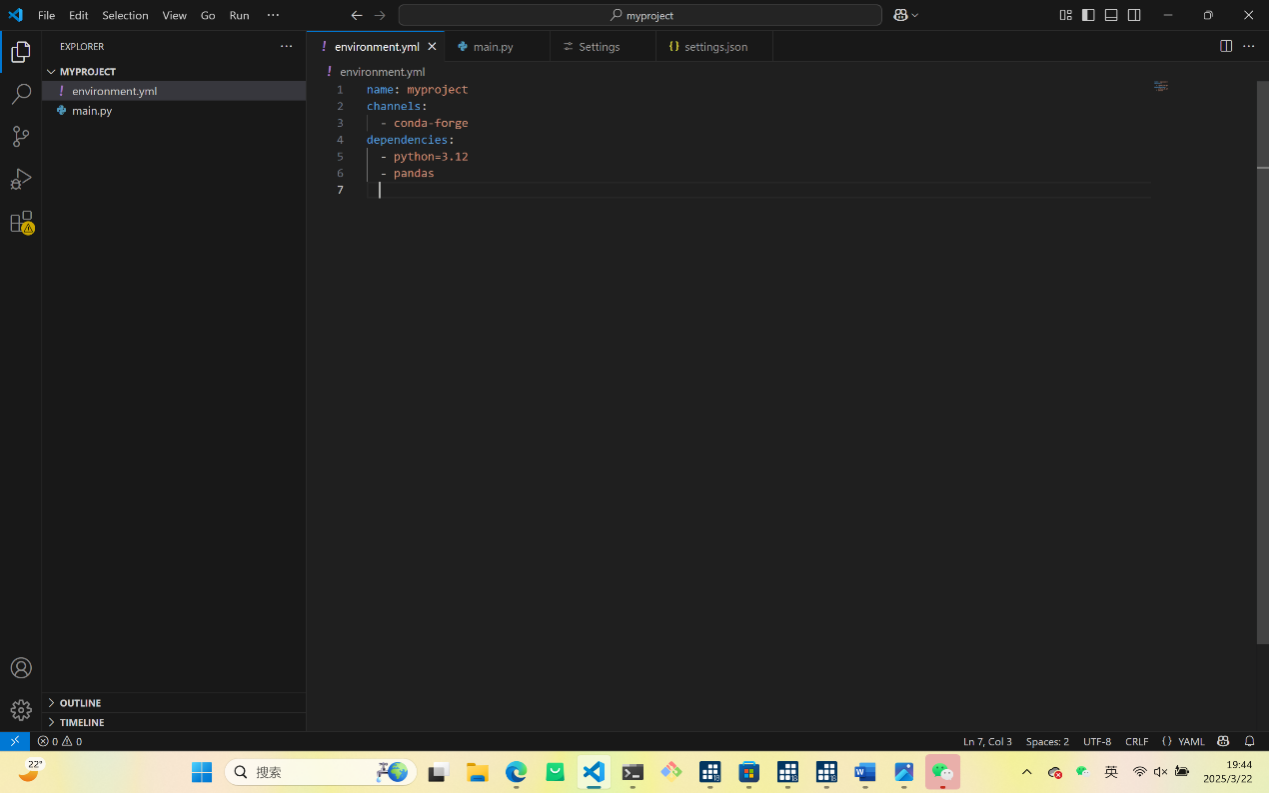
在VS Code文本编辑器里安装一些支持 Python 开发的常用扩展

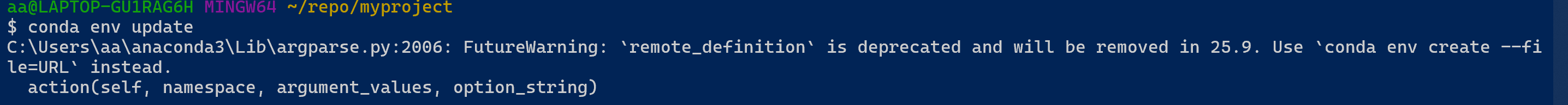


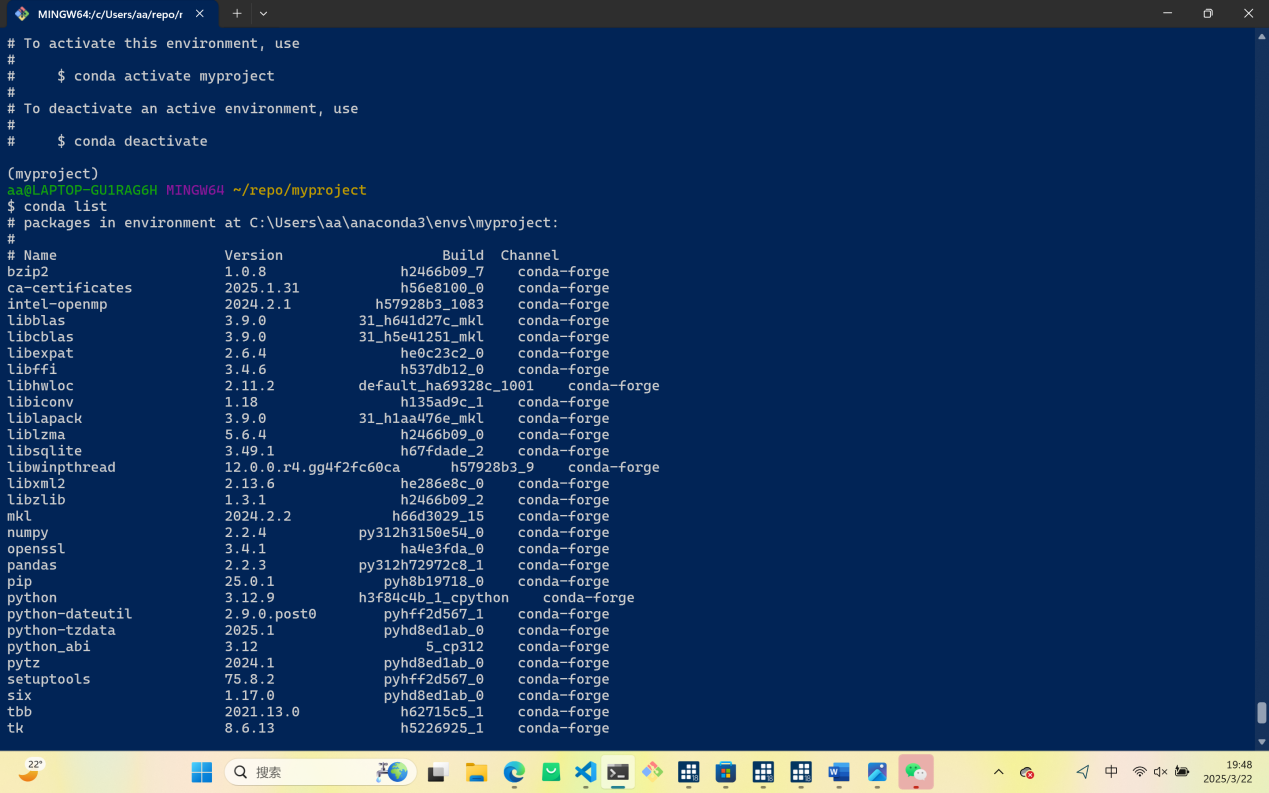


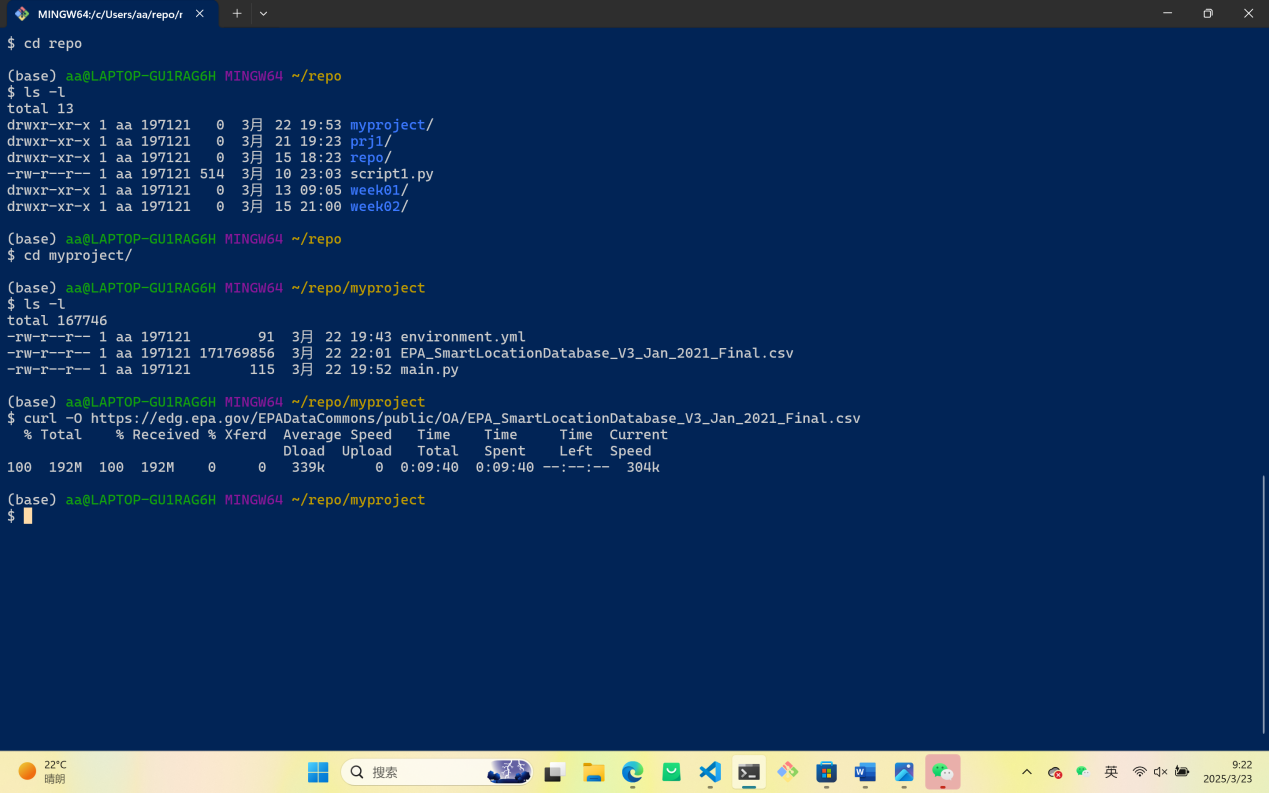
编写 main.py 脚本，创建该项目专用的 Conda 环境，在终端里激活该环境并成功运行该脚本











安装Rainbow CSV扩展

