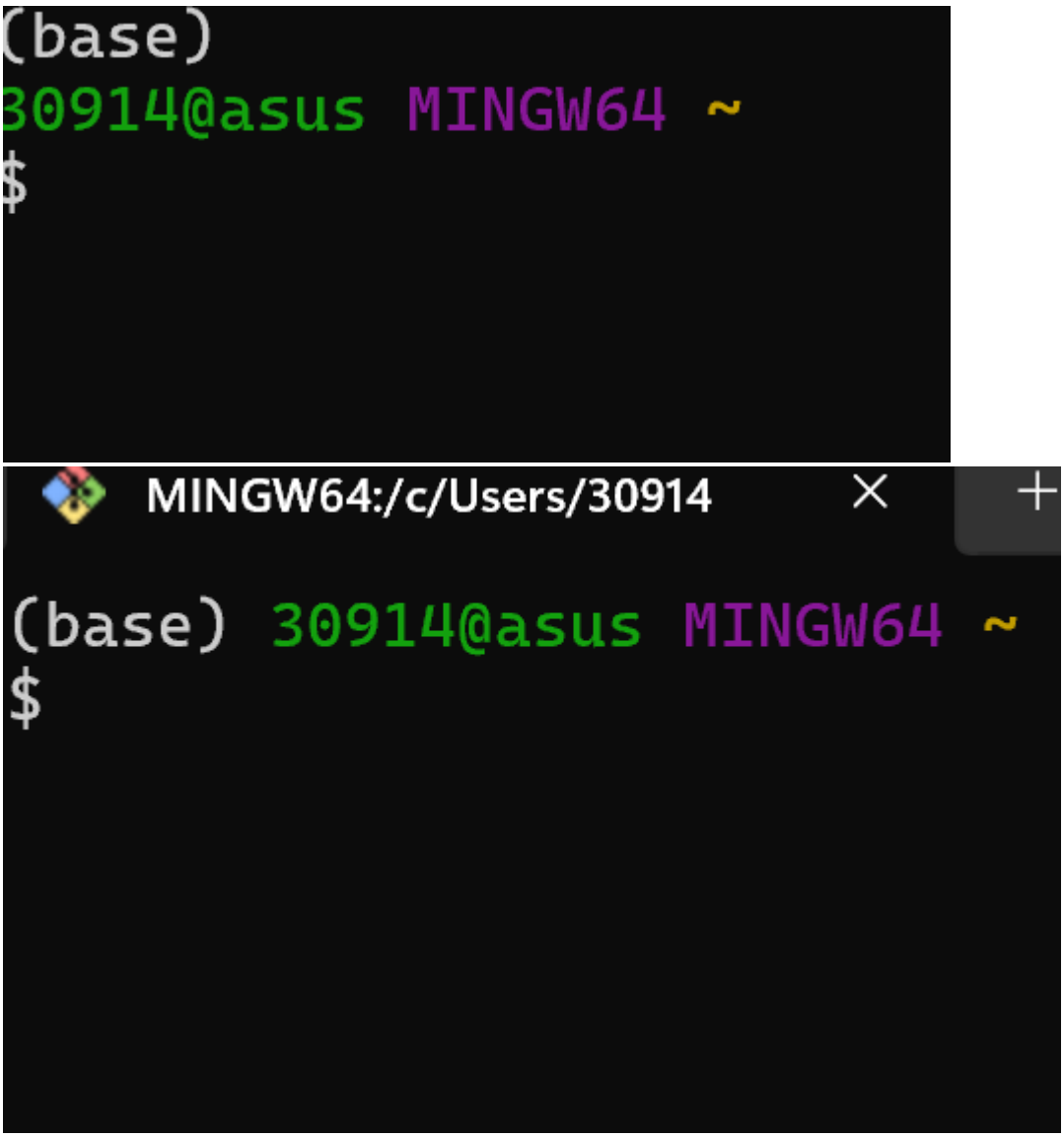
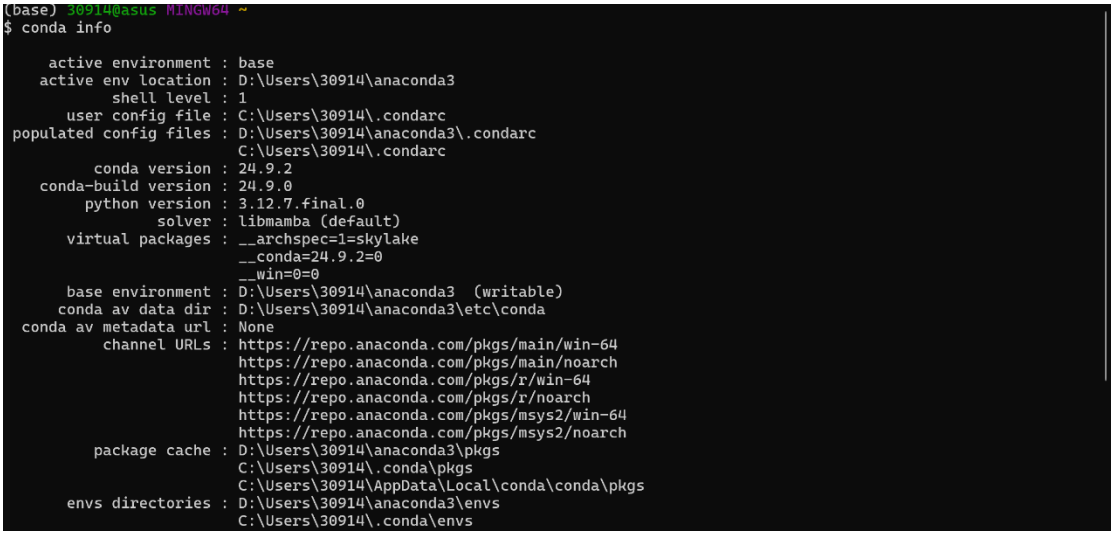
任务目标

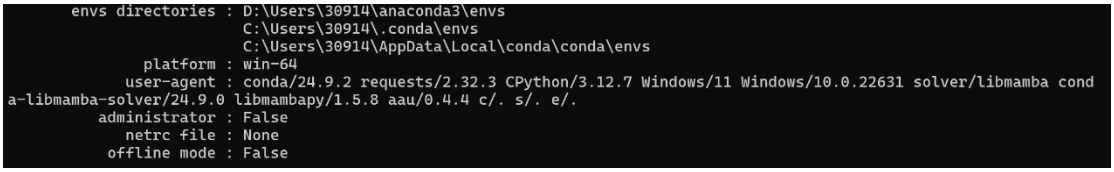
我们解决任何实际问题，基本上都是以 **项目** 为工作方式。所以我们首先应该了解如何 **创建** Python 项目，然后是如何 **使用** 她，最后是如何 *逐步地* **改进** 她。

1. 在自己的终端 (比如 Git Bash、Zsh 等) 配置好 Conda Init，使得启动终端后，在提示符 (比如 $、%) 前能够看到 (base)

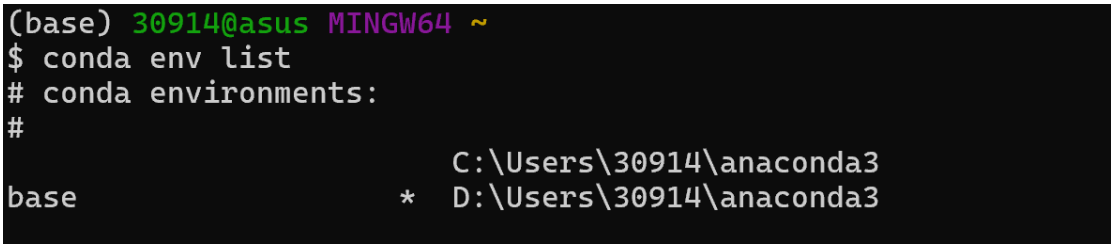


2. 使用 conda info 命令查看本机 Conda 的配置信息





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



Conda 是一个开源的包管理系统和环境管理系统，用于安装、运行和管理 Python 等编程语言的软件包及其依赖项。以下是对 Conda 环境的理解：Conda 环境是一个独立的软件环境，它可以包含特定版本的 Python 解释器、各种软件包以及它们的依赖项。每个 Conda 环境都是相互隔离的，这意味着在一个环境中安装和配置的软件包不会影响其他环境。

作用

- 隔离项目依赖：不同的项目可能依赖于不同版本的软件包，通过 Conda 环境，可以为每个项目创建独立的环境，避免因依赖冲突导致的问题。

- 方便项目迁移：将项目及其依赖环境打包，可以在不同的机器上快速搭建相同的运行环境，确保项目能够正常运行。

使用

- 创建环境：使用命令 conda create -n myenv python=3.8 可以创建一个名为 myenv 的环境，并指定 Python 版本为 3.8。

- 激活环境：在 Windows 系统中，使用 activate myenv 命令激活环境；在 Linux 和

macOS 系统中，使用 source activate myenv 命令。激活后，命令行提示符会显示当前环境的名称。

- 安装包：在激活的环境中，使用 conda install package_name 命令安装所需的软件包。

- 退出环境：使用 deactivate 命令退出当前环境。

管理

- 可以使用 conda env list 命令查看所有的 Conda 环境。

- 使用 conda remove -n myenv --all 命令删除指定的环境。

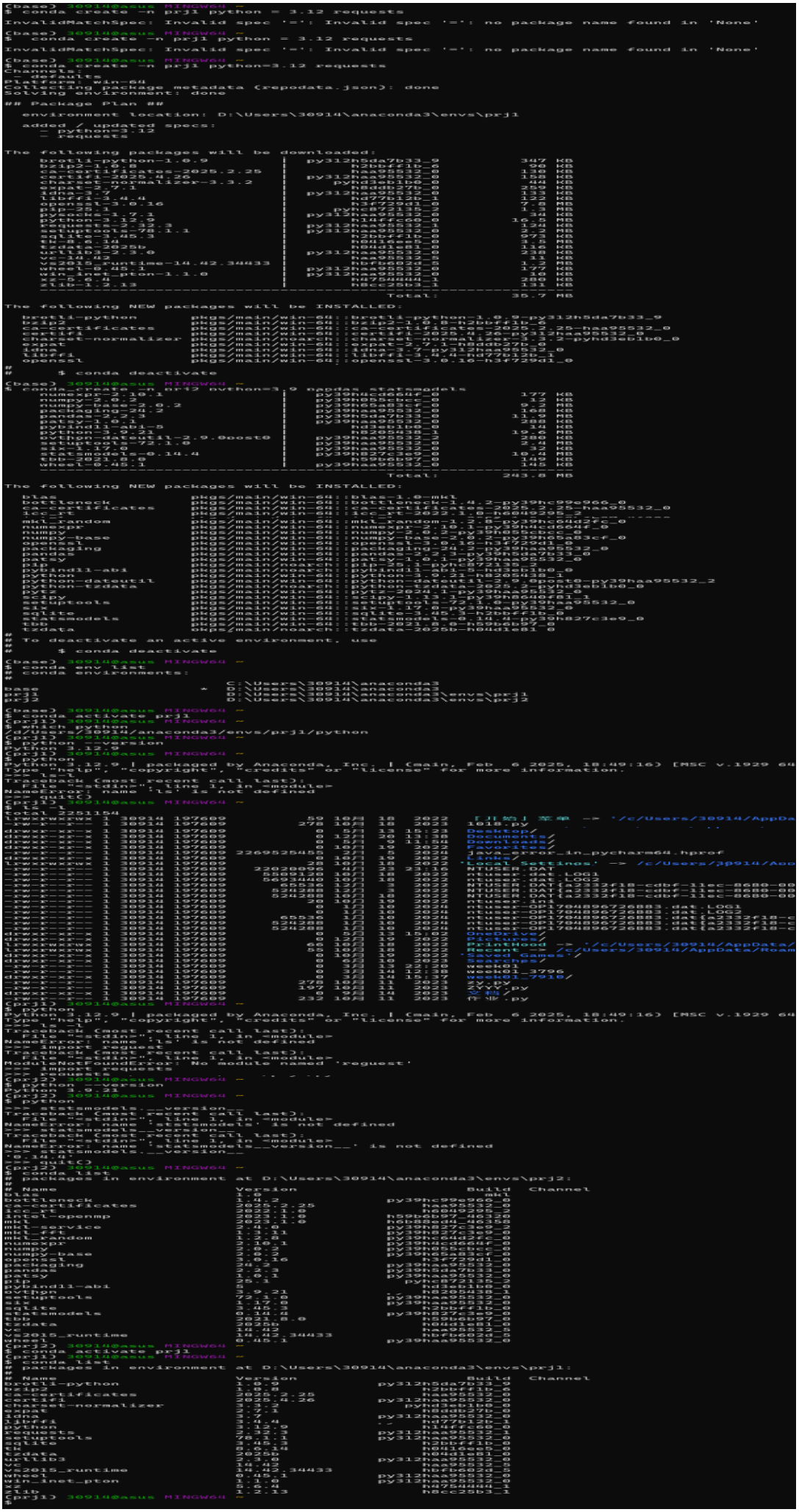
- 使用 conda update -n myenv package_name 命令更新指定环境中的软件包。

通过 Conda 环境，可以方便地管理不同项目的依赖关系，提高开发效率，减少因环境配置问题导致的错误。

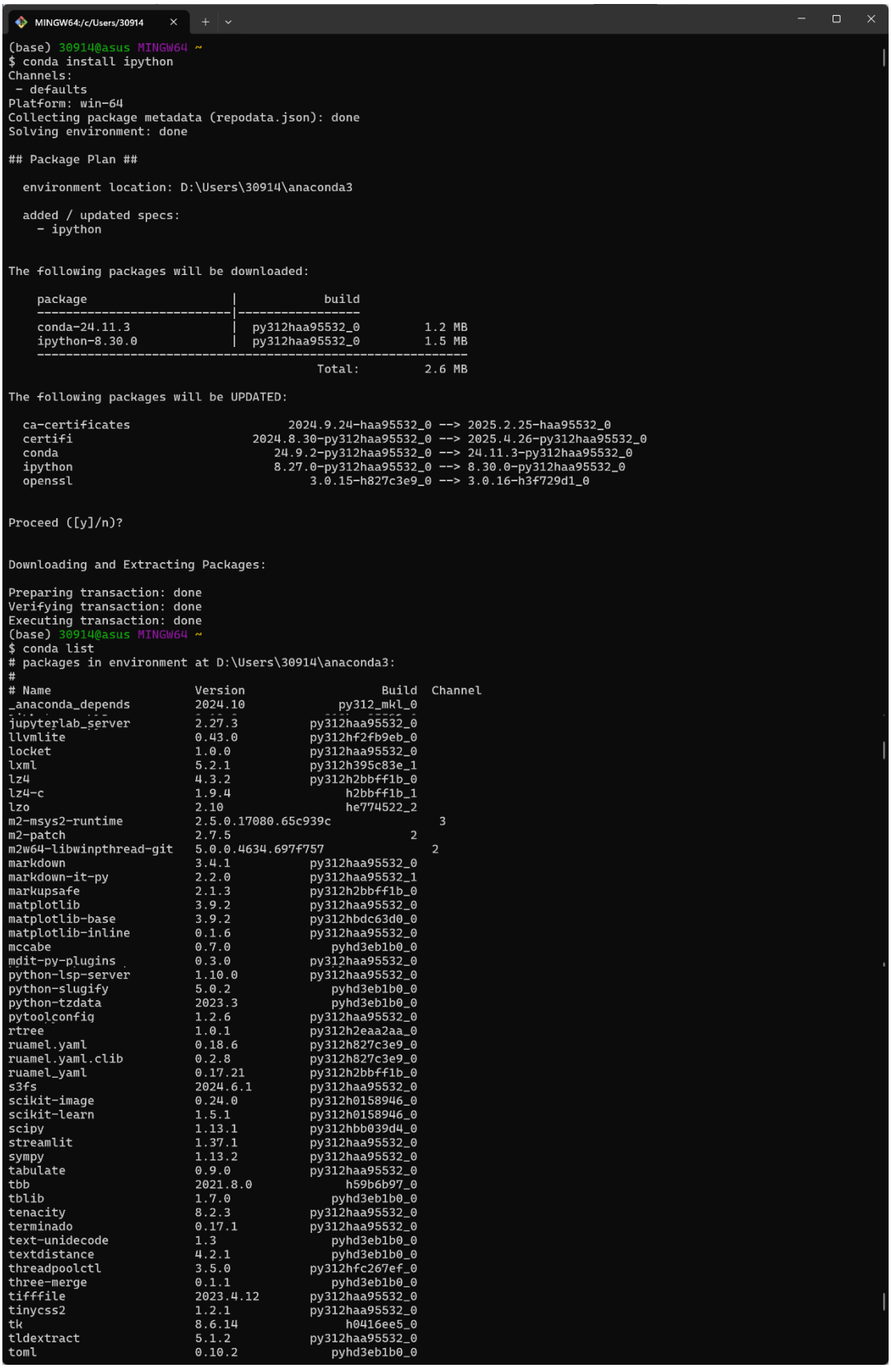
4. 使用 conda create 命令创建两个 Conda 环境，一个里面安装 Python 3.12 和 requests 软件包，另一个里面安装 Python 3.9、pandas 和 statsmodels 软件包，能够在终端里切换 Conda 环境，验证 Python 和软件包的版本

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

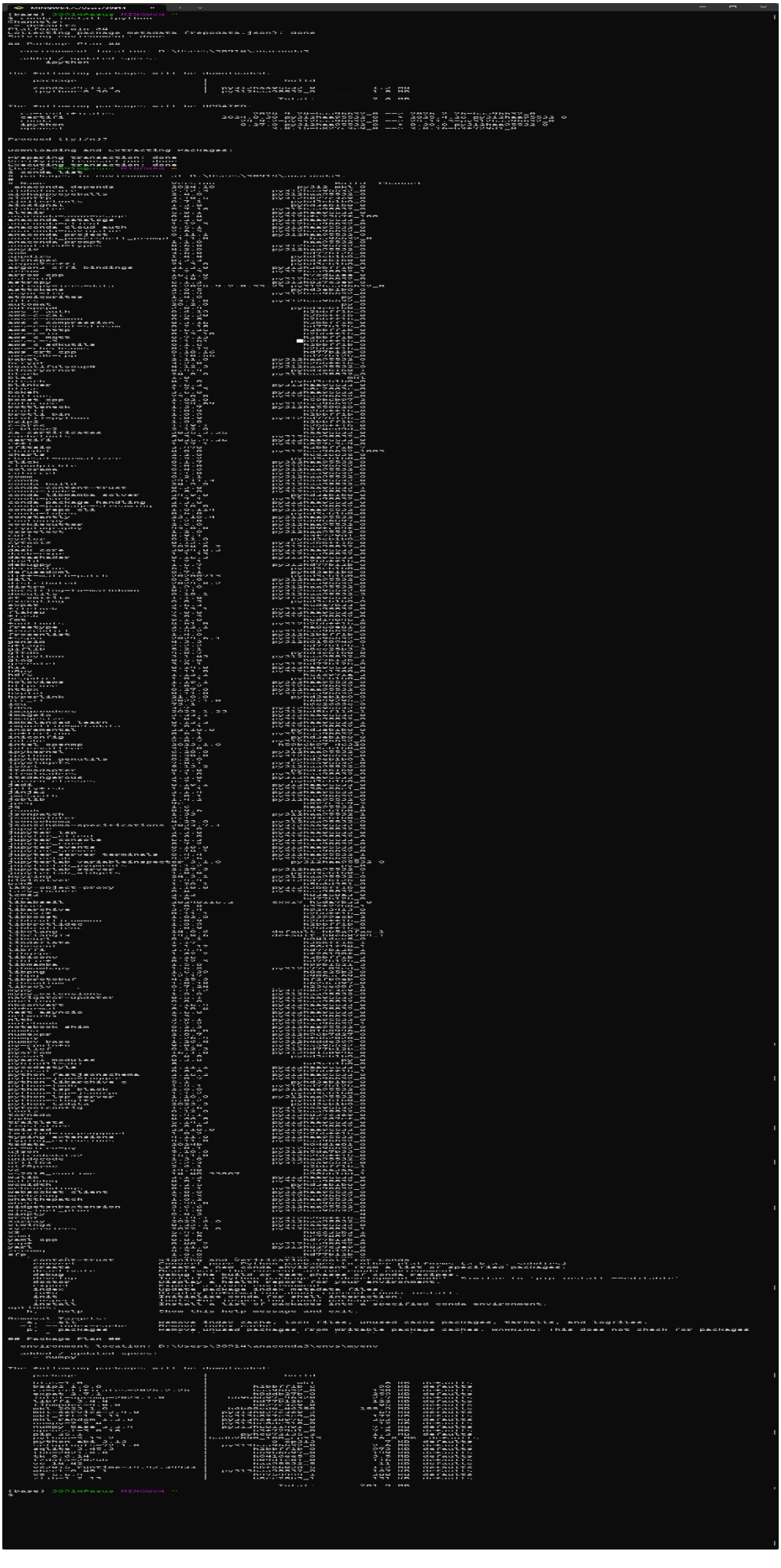
第四和第五一并



6. 使用 conda install 命令往 Conda 环境里安装更多的软件包，并验证版本



7. 根据 文档，配置 Anaconda 清华镜像，加快 conda install 安装软件包的速度，将 conda-forge 设置为默认 Channel，让 conda install 能够安装更多的软件包



```
MINGW64:/c/Users/30914
(base) 30914@asus MINGW64 ~
$ conda env list

# conda environments:
#
                         C:\Users\30914\anaconda3
base                   * D:\Users\30914\anaconda3
myenv                    D:\Users\30914\anaconda3\envs\myenv
prj1                     D:\Users\30914\anaconda3\envs\prj1
prj2                     D:\Users\30914\anaconda3\envs\prj2

(base) 30914@asus MINGW64 ~
$ conda deactivate
30914@asus MINGW64 ~
$ conda env remove -n
usage: conda-script.py env remove [-h] [-n ENVIRONMENT | -p PATH] [--solver {classic,libmamba}] [--json]
                                  [--console CONSOLE] [-v] [-q] [-d] [-y]
conda-script.py env remove: error: argument -n/--name: expected one argument
30914@asus MINGW64 ~
$ conda env remove -n prj1

Remove all packages in environment D:\Users\30914\anaconda3\envs\prj1:


## Package Plan ##

  environment location: D:\Users\30914\anaconda3\envs\prj1


The following packages will be REMOVED:

  brotli-python-1.0.9-py312h5da7b33_9
  bzip2-1.0.8-h2bbff1b_6
  ca-certificates-2025.2.25-haa95532_0
  certifi-2025.4.26-py312haa95532_0
  charset-normalizer-3.3.2-pyhd3eb1b0_0
  expat-2.7.1-h8ddb27b_0
  idna-3.7-py312haa95532_0
  libffi-3.4.4-hd77b12b_1
  openssl-3.0.16-h3f729d1_0
  pip-25.1-pyhc872135_2
  pysocks-1.7.1-py312haa95532_0
  python-3.12.9-h14ffc60_0
  requests-2.32.3-py312haa95532_1
  setuptools-78.1.1-py312haa95532_0
  sqlite-3.45.3-h2bbff1b_0
  tk-8.6.14-h0416ee5_0
  tzdata-2025b-h04d1e81_0
  urllib3-2.3.0-py312haa95532_0
  vc-14.42-haa95532_5
  vs2015_runtime-14.42.34433-hbfb602d_5
  wheel-0.45.1-py312haa95532_0
  win_inet_pton-1.1.0-py312haa95532_0
  xz-5.6.4-h4754444_1
  zlib-1.2.13-h8cc25b3_1


Proceed ([y]/n)? \
Invalid choice: \
Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Everything found within the environment (D:\Users\30914\anaconda3\envs\prj1), including any conda environment configurat
ions and any non-conda files, will be deleted. Do you wish to continue?
 (y/[n])?


CondaSystemExit: Exiting.

30914@asus MINGW64 ~
$ conda env list

# conda environments:
#
                         C:\Users\30914\anaconda3
base                     D:\Users\30914\anaconda3
myenv                    D:\Users\30914\anaconda3\envs\myenv
prj1                     D:\Users\30914\anaconda3\envs\prj1
prj2                     D:\Users\30914\anaconda3\envs\prj2

30914@asus MINGW64 ~
$ con env remove
bash: con: command not found
30914@asus MINGW64 ~
$ conda create -n prj1 python=3.12 requests polars-lts-cpu
WARNING: A conda environment already exists at 'D:\Users\30914\anaconda3\envs\prj1'

Remove existing environment?
This will remove ALL directories contained within this specified prefix directory, including any other conda environment
s.

 (y/[n])?


CondaSystemExit: Exiting.

30914@asus MINGW64 ~
$ conda env list

# conda environments:
#
                         C:\Users\30914\anaconda3
base                     D:\Users\30914\anaconda3
myenv                    D:\Users\30914\anaconda3\envs\myenv
prj1                     D:\Users\30914\anaconda3\envs\prj1
prj2                     D:\Users\30914\anaconda3\envs\prj2

30914@asus MINGW64 ~
$ conda activate prj1
(prj1) 30914@asus MINGW64 ~
$ python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import polars
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'polars'
>>>
```

8. 使用 pip install 命令往 Conda 环境里安装 Python 软件包，并验证版本
9. 根据 文档 配置 PyPI 清华镜像，加快 pip install 安装软件包的速度
10. 能够导出 environment.yml Conda 环境配置文件，能够删除 Conda 环境，能够用 environment.yml 配置文件重建 Conda 环境
11. 理解 Conda 与 Python 的关系，理解 Conda-Forge 与 Conda 的关系，理解 Python 解释器、第三方软件包、PyPI 软件仓库、以及程序/软件包的路径问题
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

    第 8 至 12 如下

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

首先，激活你的 Conda 环境（假设环境名为 myenv）：

```
bash
 conda activate myenv
```

然后使用 pip install 安装一个软件包，比如 numpy：

```
bash
 pip install numpy
```

安装完成后，打开 Python 解释器，验证 numpy 的版本：

```
python
 import numpy
print(numpy.__version__)
```

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

打开命令行，编辑 pip 的配置文件（如果没有则创建）：

```
bash
 mkdir -p ~/.pip
nano ~/.pip/pip.conf
```

在打开的文件中添加以下内容：

plaintext

[global]

index-url = https://pypi.tuna.tsinghua.edu.cn/support-packages

[install]

trusted-host = pypi.tuna.tsinghua.edu.cn

保存并退出 nano 编辑器。

10. 能够导出 environment.yml Conda 环境配置文件，能够删除 Conda 环境，能够用 environment.yml 配置文件重建 Conda 环境

导出当前 Conda 环境的配置文件：

bash

conda env export > environment.yml

删除一个 Conda 环境（假设环境名为 myenv）：

bash

conda env remove -n myenv

使用 environment.yml 文件创建新的 Conda 环境：

bash

conda env create -f environment.yml

11. 理解 Conda 与 Python 的关系，理解 Conda-Forge 与 Conda 的关系，理解 Python 解释器、第三方软件包、PyPI 软件仓库、以及程序/软件包的路径问题

- Conda 与 Python 的关系：Conda 是一个包管理系统和环境管理系统，可以用于安装、卸载和管理 Python 软件包，也可以创建、管理和删除 Python 环境。Conda 可以安装不同版本的 Python，并且可以在不同的环境中使用不同版本的 Python。

- Conda-Forge 与 Conda 的关系：Conda-Forge 是一个社区驱动的 Conda 软件包集合，提供了大量的开源软件包。Conda 可以从 Conda-Forge 渠道安装软件包，丰富了 Conda 的软件包资源。

- Python 解释器：是运行 Python 代码的程序，负责解析和执行 Python 代码。

- 第三方软件包：是由其他人开发的 Python 代码库，可以在项目中引入并使用，如 numpy 、 pandas 等。

- PyPI 软件仓库：Python Package Index，是 Python 官方的软件包仓库，包含了大量的 Python 软件包，可以使用 pip 从 PyPI 安装软件包。

- 程序/软件包的路径问题：Python 在运行时会根据一定的规则查找软件包和程序，如 sys.path 中包含的路径，了解这些路径有助于解决软件包导入失败等问题。

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

1. 创建项目目录：

bash

mkdir my_project

cd my_project

1. 打开 VS Code 并打开项目目录：

bash

code .

1. 在 VS Code 中安装 Python 扩展（在扩展市场搜索 Python 并安装）。

2. 编写 main.py 脚本，例如：

python

print("Hello, World!")

创建项目专用的 Conda 环境（假设环境名为 my_project_env ）：

bash

conda create -n my_project_env python=3.8

. 激活 Conda 环境：

bash

conda activate my_project_env

在激活环境下运行 main.py 脚本：

bash

python main.py