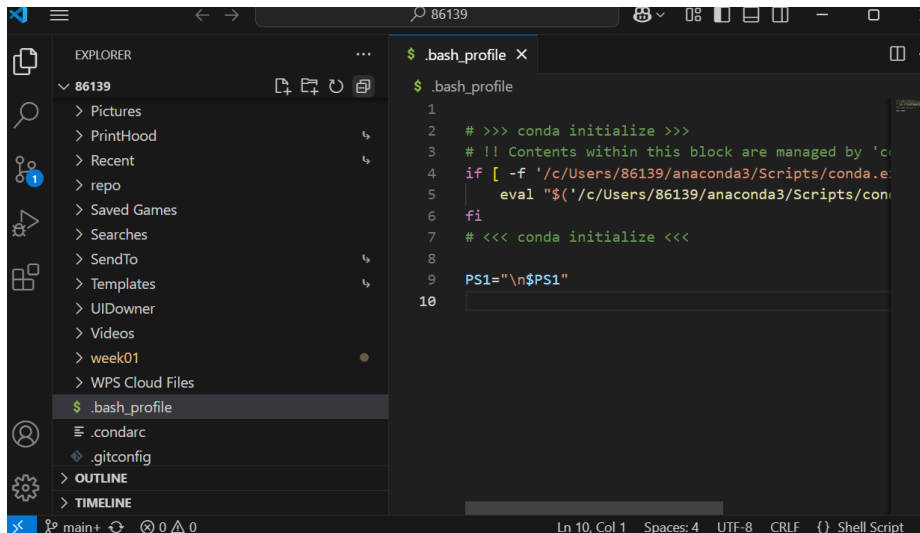


### 第3周 Python 项目 (初级)

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)

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~  
$
```



```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~  
$
```

2. 使用 conda info 命令查看本机 Conda 的配置信息

Ctrl L 清空

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda info  
  
active environment : base  
active env location : C:\Users\86139\anaconda3  
shell level : 1  
user config file : C:\Users\86139\.condarc  
populated config files : C:\Users\86139\anaconda3\.condarc  
C:\Users\86139\.condarc  
conda version : 24.9.2  
conda-build version : 24.9.0  
python version : 3.12.7.final.0  
solver : libmamba (default)  
virtual packages : __archspec=1=x86_64_v3  
__conda=24.9.2=0  
__win0=0  
base environment : C:\Users\86139\anaconda3 (writable)  
conda av data dir : C:\Users\86139\anaconda3\etc\conda  
conda av metadata url : None  
channel URLs : https://repo.anaconda.com/pkg/main/win-64  
https://repo.anaconda.com/pkg/main/noarch  
https://repo.anaconda.com/pkg/x/win-64  
https://repo.anaconda.com/pkg/x/noarch  
https://repo.anaconda.com/pkg/msys2/win-64  
https://repo.anaconda.com/pkg/msys2/noarch  
package cache : C:\Users\86139\anaconda3\pkgs  
C:\Users\86139\conda\pkgs  
C:\Users\86139\AppData\Local\conda\conda\pkgs  
envs directories : C:\Users\86139\anaconda3\envs  
C:\Users\86139\conda\envs  
C:\Users\86139\AppData\Local\conda\conda\envs  
platform : win-64  
user-agent : conda/24.9.2 requests/2.32.3 CPython/3.12.7 Windows/11 Windows/10.0.26100 solver/libmamba conda-libmamba-solver/24.9.0 libmambapy/1.5.8 aau/8.4.4 c/. s/. e/.  
netrc file : None  
offline mode : False
```

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

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda env list  
# conda environments:  
#  
base * C:\Users\86139\anaconda3
```

## 创建 Conda 环境的原因

- **版本隔离**：不同项目可能依赖于同一库的不同版本。例如，项目 A 需要使用 NumPy 1.18 版本，而项目 B 需要使用 NumPy 1.20 版本。通过创建不同的 Conda 环境，可以分别在不同环境中安装所需版本的 NumPy，避免版本冲突。
- **项目独立性**：每个项目都有自己独特的依赖项集合，使用 Conda 环境可以确保项目的依赖项不会与其他项目的依赖项相互干扰，保证项目的独立性和可移植性。

## Conda 环境的优势

- **易于管理**：Conda 提供了一系列简单易用的命令，方便用户创建、激活、停用、删除环境以及安装、更新和卸载软件包。
- **跨平台支持**：Conda 可以在 Windows、Linux 和 macOS 等多种操作系统上使用，保证了在不同平台上的 consistency。
- **依赖管理**：Conda 能够自动处理软件包之间的依赖关系，确保安装的软件包及其依赖项能够正常工作。

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

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~
$ conda create -n prj1 python=3.12 requests
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\86139\anaconda3\envs\prj1

added / updated specs:
- python=3.12
- requests
```

```
done
#
# To activate this environment, use
#
#     $ conda activate prj1
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~
$ conda create -n prj2 python=3.9 pandas statsmodels
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\86139\anaconda3\envs\prj2

added / updated specs:
- pandas
- python=3.9
- statsmodels
```

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~
$ conda env list
# conda environments:
#
base                  * C:\Users\86139\anaconda3
prj1                  C:\Users\86139\anaconda3\envs\prj1
prj2                  C:\Users\86139\anaconda3\envs\prj2
```

```
>>> quit()
(prj1)
86139@LAPTOP-J150R7EU MINGW64 ~
$ conda activate prj2
(prj2)
86139@LAPTOP-J150R7EU MINGW64 ~
$ which python
/c/Users/86139/anaconda3/envs/prj2/python
(prj2)
86139@LAPTOP-J150R7EU MINGW64 ~
$ python --version
Python 3.9.21
(prj2)
86139@LAPTOP-J150R7EU MINGW64 ~
$ python
Python 3.9.21 (main, Dec 11 2024, 16:35:24) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'requests'
>>> import pandas
>>> pandas.__file__
'C:\Users\86139\anaconda3\envs\prj2\lib\site-packages\pandas\__init__.py'
>>> pandas.__version__
'2.2.3'
>>> import statsmodels
>>> statsmodels.__version__
'0.14.4'
```

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

```
86139@LAPTOP-J150R7EU MINGW64 ~
$ conda list
# packages in environment at C:\Users\86139\anaconda3\envs\prj2:
python 3.9.21
pandas 2.2.3
statsmodels 0.14.4
ipython 8.30.0
py312haa95532_0

86139@LAPTOP-J150R7EU MINGW64 ~
$ conda activate prj1
(prj1)
86139@LAPTOP-J150R7EU MINGW64 ~
$ conda list
```

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

```
86139@LAPTOP-J150R7EU MINGW64 ~
$ conda install ipython

86139@LAPTOP-J150R7EU MINGW64 ~
$ conda list

ipython 8.30.0 py312haa95532_0
```

7. 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

```
86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda config --set show_channel_urls yes  
(prj1)
```

```
! .condarc  
1 channels:  
2   - conda-forge  
3   - defaults  
4 show_channel_urls: true  
5 default_channels:  
6   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main  
7   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r  
8   - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/msys2  
9 custom_channels:  
10  conda-forge: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud  
11  pytorch: https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud  
12 channel_priority: strict
```

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

```
(base) 86139@LAPTOP-J150R7EU MINGW64  
$ pip install tushare  
Collecting tushare
```

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

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~  
$ python -m pip install --upgrade pip  
Requirement already satisfied: pip in c:\users\86139\anaconda3\lib\site-packages  
(24.2)  
Collecting pip  
  Downloading pip-25.0.1-py3-none-any.whl.metadata (3.7 kB)  
  Downloading pip-25.0.1-py3-none-any.whl (1.8 MB)  
    1.8/1.8 MB 121.6 kB/s eta 0:00:00  
Installing collected packages: pip  
  Attempting uninstall: pip  
    Found existing installation: pip 24.2  
    Uninstalling pip-24.2:  
      Successfully uninstalled pip-24.2  
  Successfully installed pip-25.0.1
```

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

```
86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda env export -f environment.yml  
(prj1)  
86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda deactivate  
(base)  
86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda env list  
# conda environments:  
#  
base * C:\Users\86139\anaconda3  
prj1 C:\Users\86139\anaconda3\envs\prj1  
prj2 C:\Users\86139\anaconda3\envs\prj2  
  
(base)  
86139@LAPTOP-J150R7EU MINGW64 ~  
$ conda env remove -n prj1
```

```

! environment.yml
1  name: prj1
2  channels:
3    - defaults
4    - https://repo.anaconda.com/pkgs/main
5    - https://repo.anaconda.com/pkgs/r
6    - https://repo.anaconda.com/pkgs/msys2
7  dependencies:
8    - brotli-python=1.0.9=py312h5da7b33_9
9    - bzip2=1.0.8=h2bbff1b_6
10   - ca-certificates=2025.2.25=haa95532_0
11   - certifi=2025.1.31=py312haa95532_0
12   - charset-normalizer=3.3.2=pyhd3eb1b0_0
13   - expat=2.6.4=h8ddb27b_0

86139@LAPTOP-J15OR7EU MINGW64 ~
$ cd repo
(base)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ ls -l
total 0
drwxr-xr-x 1 86139 197609 0  3月 22 19:00 mywork/
(base)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ mkdir prj1
(base)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ cd prj1
(base)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/prj1
$ ls -l
total 0
(base)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/prj1
$ mv ~/environment.yml ./

```

### conda env create 重建

11. **Conda 与 Python 的关系:** conda 用于安装、管理 python 的第三方软件包，还能创建管理不同的 python 环境，每个环境可以有独立的 python 版本和软件包组合，有助于解决不同项目对 python 版本和软件包以来不同的问题。

**Conda 是基础软件包管理系统:** Conda 是一个开源的包管理系统和环境管理系统，用于安装、管理和更新软件包以及创建和管理隔离的软件环境。它可以在不同的操作系统上运行，支持多种编程语言，如 Python、R 等。Conda 允许用户轻松地安装和切换不同版本的软件包，解决软件包之间的依赖关系问题，使得开发和运行不同项目所需的特定环境配置变得更加容易。

**Conda-Forge 是软件包通道:** Conda-Forge 是为 Conda 提供软件包的一个通道 (channel)。Conda 可以从多个通道获取软件包，而 Conda-Forge 是其中一个非常重要且受欢迎的通道。它由社区维护和贡献，包含了大量的开源软件包，这些软件包经过了社区的测试和验证，以确保在不同平台上的兼容性和稳定性。与其他一些官方或商业通道相比，Conda-Forge 通常更注重开源和跨平台性，并且更新较为及时，能提供许多最新版本的软件包。

**Python 解释器:** Python 是一种解释型语言，Python 解释器的作用就是将 Python 代码翻译成计算机能够理解和执行的机器码。常见的 Python 解释器有 CPython: 这是 Python 官方实现的解释器，用 C 语言编写。它是使用最广泛的解释器，支持所有 Python 标准库和第三方库。当你从 Python 官方网站下载安装 Python 时，默认安装的就是 CPython。

**第三方软件包:** Python 社区非常活跃，开发者们创建了大量的第三方软件包来扩展 Python 的功能。这些软件包可以帮助你更高效地完成各种任务，例如：数据分析: pandas 提供了高效的数据结构和数据分析工具；numpy 用于处理多维数组和矩阵运算。机器学习: scikit - learn 包含了各种机器学习算法和工具；tensorflow 和 pytorch 是深度学习框架。Web 开发: Django 和 Flask 是流行的 Web 开发框架，可用于快速搭建 Web 应用。

**PyPI 软件仓库:** 是 Python 官方的软件包仓库，它是第三方 Python 软件包的集中存储地。开发者可以将自己开发的 Python 软件包上传到 PyPI，其他用户则可以使用包管理工具（如 pip）从 PyPI 下载和安装这些软件包。

**程序/软件包的路径问题:** 在 Python 中，程序在运行时需要知道在哪里找到所需的模块和软件包。Python 解释器会按照一定的顺序搜索路径，包括当前目录、Python 安装目录、site - packages 目录等。可以通过 sys.path 变量查看 Python 解释器搜索路径的列表。当导入模块时，解释器会在这些路径中查找对应的模块文件。如果软件包不在默认搜索路径中，需要将其所在路径添加到 sys.path 中，或者使用相对路径、绝对路径来正确导入模块。在使用 Conda 环境时，由于环境的独立性，每个环境都有自己的路径设置，确保了不同环境中的软件包和程序不会相互干扰。

12. **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**



### 【创建并激活一个环境】

conda env create

conda activate my-project

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~/repo/myproject
$ cat environment.yml
name: myproject
channels:
  - conda-forge
dependencies:
  - python=3.12
(base) 86139@LAPTOP-J150R7EU MINGW64 ~/repo/myproject
$ conda env create
```

```
(base) 86139@LAPTOP-J150R7EU MINGW64 ~/repo/myproject
$ conda env list
# conda environments:
#
base                  * C:\Users\86139\anaconda3
myproject             C:\Users\86139\anaconda3\envs\myproject
prj1                  C:\Users\86139\anaconda3\envs\prj1
prj2                  C:\Users\86139\anaconda3\envs\prj2

(base) 86139@LAPTOP-J150R7EU MINGW64 ~/repo/myproject
$ conda activate myproject
(myproject)
```

### 【创建一个简单的 Python 程序】

```
main.py > ...
1  def main():
2      print("Hello, conda!")
3
4
5  if __name__ == "__main__":
6      main()
```

```
86139@LAPTOP-J150R7EU MINGW64 ~/repo/myproject
$ python main.py
Hello, conda!
```

### 【Ruff 设定】

```
1  {
2      "workbench.startupEditor": "none",
3      "workbench.colorTheme": "Monokai",
4      "[python]": {
5          "editor.formatOnSave": true,
6          "editor.codeActionsOnSave": {
7              "source.fixAll": "explicit",
8              "source.organizeImports": "explicit"
9          },
10         "editor.defaultFormatter": "charliermarsh.ruff",
11     },
12     "notebook.formatOnSave.enabled": true,
13     "notebook.codeActionsOnSave": {
14         "notebook.source.fixAll": "explicit",
15         "notebook.source.organizeImports": "explicit"
16     },
17 }
```

### 【更新项目】 conda env update

```
! environment.yml
1  name: myproject
2  channels:
3  |   - conda-forge
4  dependencies:
5  |   - python=3.12
6  |   - pandas
```

```
main.py > ...
1  import pandas as pd
2
3
4  def main():
5      print("Hello, conda!")
6      print(pd.__version__)
7      print(pd.__file__)
8
9
10 if __name__ == "__main__":
11     main()
```