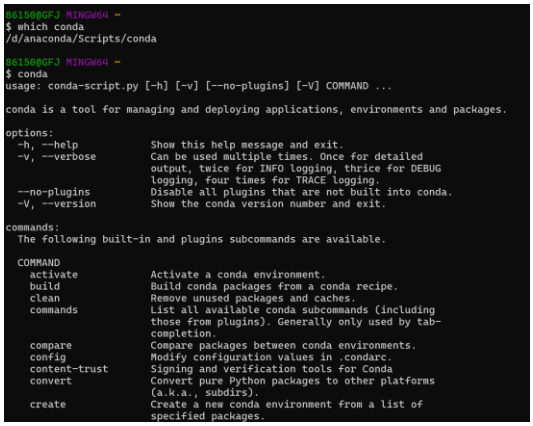
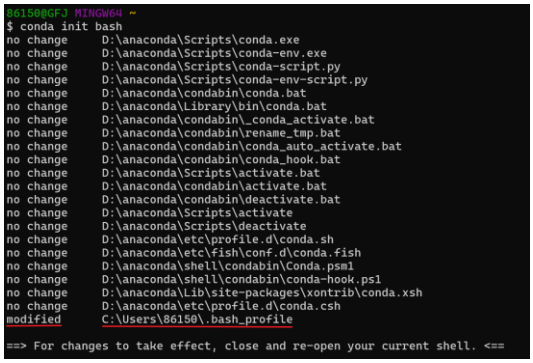
金融计算机第三周作业

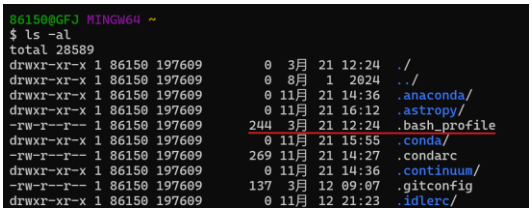
专题一 终端配置 Conda 初始化



1.配置 conda init

Conda init：作用是对 shell 环境进行初始化，这样在每次启动 shell 时，conda 就能自动被激活。





Bash. profile 文件就是 Bash 在启动时就会运行这个文件里的内容。

2.解决 base 换行问题

The current environment is shown on the line above the username in Git Bash on Windows10.

解决方案：

```
$ git-prompt.sh
if test -f /etc/profile.d/git-sdk.sh
then
    TITLEPREFIX=SDK-${MSYSTEM#MINGW}
else
    TITLEPREFIX=$MSYSTEM
fi

if test -f ~/.config/git/git-prompt.sh
then
    . ~/.config/git/git-prompt.sh
else
    PS1='\[\033]0;$TITLEPREFIX:$PWD\007\]' # set window title
    # PS1="$PS1"'\n'                 # new line
    PS1="$PS1"'\[\033[32m\]'       # change to green
    PS1="$PS1"'\u@\h '             # user@host<space>
    PS1="$PS1"'\[\033[35m\]'       # change to purple
    PS1="$PS1"'$MSYSTEM '          # show MSYSTEM
    PS1="$PS1"'\[\033[33m\]'       # change to brownish yellow
    PS1="$PS1"'\w'                 # current working directory
    if test -z "$WINELOADERNOEXEC"
    then
        GIT_EXEC_PATH="$(git --exec-path 2>/dev/null)"
        COMPLETION_PATH="${GIT_EXEC_PATH%/libexec/git-core}"
        COMPLETION_PATH="${COMPLETION_PATH%/lib/git-core}"
        COMPLETION_PATH="$COMPLETION_PATH/share/git/completion"
        if test -f "$COMPLETION_PATH/git-prompt.sh"
        then
            . "$COMPLETION_PATH/git-completion.bash"
            . "$COMPLETION_PATH/git-prompt.sh"
            PS1="$PS1"'\[\033[36m\]'  # change color to cyan
            PS1="$PS1"'`__git_ps1`'   # bash function
```

删除第13行

```
(base) 86150@GFJ MINGW64 ~
$
```

但是此时 base 前无空行。

```
drwxr-xr-x 1 86150 197609
drwxr-xr-x 1 86150 197609
drwxr-xr-x 1 86150 197609
(base) 86150@GFJ MINGW64 ~
$
```

加空行：

```
$ .bash_profile
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
if [ -f '/d/anaconda/Scripts/conda.exe' ]; then
    eval "$('/d/anaconda/Scripts/conda.exe' 'shell.bash' 'hook')"
fi
# <<< conda initialize <<<

PS1="\n$PS1"
```

先换行

```
drwxr-xr-x 1 86150 197609        0
drwxr-xr-x 1 86150 197609        0
drwxr-xr-x 1 86150 197609        0

(base) 86150@GFJ MINGW64 ~
$
```

问题得到修复。

专题二 创建 Conda 环境

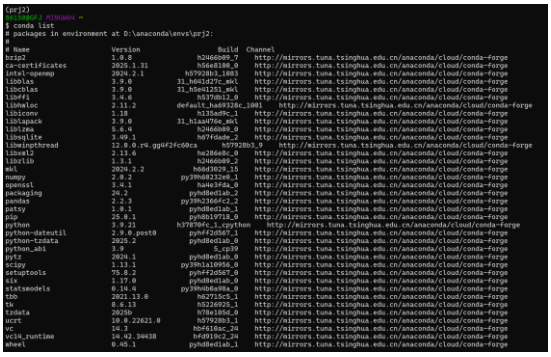
1. Conda info

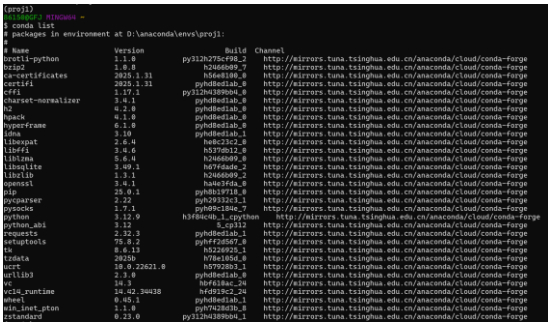
2. conda environment list

3. 用 conda create 命令创建 conda 环境

4. conda list

可以查看在 conda 环境下有哪些软件包





通过 conda list 命令，可以看出，在 conda 环境下不光有软件包，还有软件包的依赖项。

专题三 Conda-Forge 清华镜像

1.conda install

激活环境：

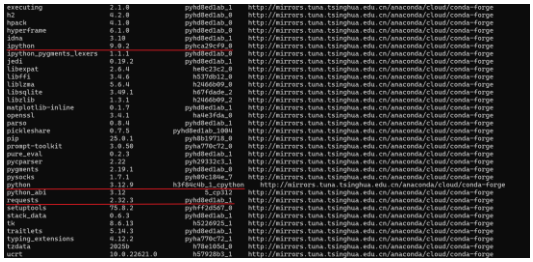
```
(proj1)
86150@GFJ MINGW64 ~
$ conda install ipython
Channels:
 - http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\anaconda\envs\proj1

  added / updated specs:
    - ipython
```

用 conda list 查看版本：



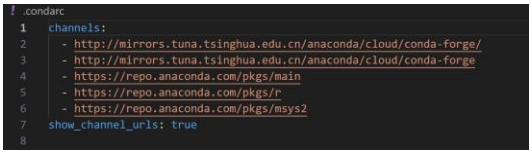
2. 配置 anaconda 清华镜像

Anaconda 是一个广泛用于数据科学和机器学习的开源 Python 发行版，而使用清华镜像可以显著提升在国内下载 Anaconda 包的速度。

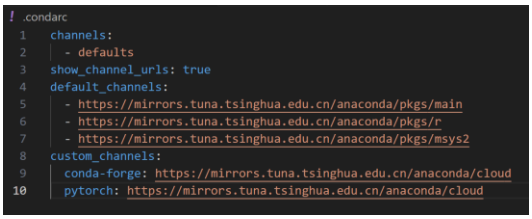
.condarc：conda 初始化文件夹

创建该文件夹

```
86150@GFJ MINGW64 ~
$ conda config --set show_channel_urls yes
```



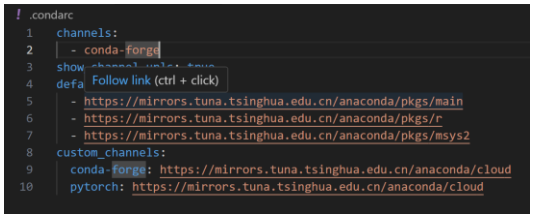
添加第三方源



Conda clean：是 Conda 包管理器中的一个实用命令，其主要功能是清理系统中不再需要的文件，以此释放磁盘空间并解决一些潜在的缓存问题。

-i：**清理索引缓存**。Conda 的索引缓存包含了软件包的元数据信息，用于加速包的搜索和安装过程。但这些缓存可能会过期或损坏，使用该命令清理索引缓存有助于避免因缓存问题导致的包搜索或安装异常。

修改 channel 为 conda-forge：



在 Conda 中，**channel** 指的是软件包的来源渠道，它是存储着各类软件包及其元数据的仓库。借助指定不同的 channel，你能够获取到各式各样的软件包，满足不同的使用需求。

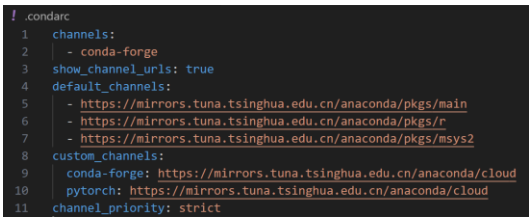
3. 设置默认 channel 为 conda forge

Conda-forge : Conda Forge 是一个由社区驱动的(开源的)、用于管理 Conda 软件包的渠道。它为用户提供了丰富的软件包资源，在数据科学、机器学习、科学计算等多个领域广泛应用。

- **特点：**①更新及时；②跨平台支持；③广泛的软件包选择。

优先使用 conda-forge 仓库：

```
(proj1)
86150@GFJ MINGW64 ~
$ conda config --set channel_priority strict
```



在原 anaconda 应用商店中，无法下载 polars 软件包。但在 conda forge 中可以找到该软件包并下载。

```
(base) 86150@GFJ MINGW64 ~
$ conda activate proj1
(proj1)
86150@GFJ MINGW64 ~
$ conda install polars
Channels:
 - conda-forge
 - https://repo.anaconda.com/pkgs/main
 - https://repo.anaconda.com/pkgs/r
 - https://repo.anaconda.com/pkgs/msys2
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\anaconda\envs\proj1

  added / updated specs:
    - polars
```

其下载路径为 pypi

Q：来源软件包出现问题时，不要直接删除（conda remove）该软件包，可能会造成更大的瘫痪。应该先取消激活 conda 环境，退回 base，运用 conda env remove -n [环境名]将整个环境移

除。

```
(base)
86150@GFJ MINGW64 ~
$ conda env remove -n proj1

Remove all packages in environment D:\anaconda\envs\proj1:


## Package Plan ##

  environment location: D:\anaconda\envs\proj1


The following packages will be REMOVED:
```

专题四 环境配置的导出与重建

1.pip install

pip install 是 Python 中用于安装软件包的核心命令，pip 是 Python 的包管理工具，借助 pip install 可以从 Python 包索引（PyPI）或者其他源安装各种 Python 软件包

- Tushare：是一个免费、开源的 Python 财经数据接口包，为金融分析和量化交易领域的开发者、研究者提供了便捷获取金融数据的途径。

```
86150@GFJ MINGW64 ~
$ conda install tushare
Channels:
 - conda-forge
 - https://repo.anaconda.com/pkgs/main
 - https://repo.anaconda.com/pkgs/r
 - https://repo.anaconda.com/pkgs/msys2
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: failed

PackagesNotFoundError: The following packages are not available from current channels:

  - tushare

Current channels:

  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r
  - https://repo.anaconda.com/pkgs/msys2

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org
```

用 conda install 无法安装 tushare，得使用 pip install 安装

```
(prj1)
86150@GFJ MINGW64 ~
$ pip install tushare
Collecting tushare
  Downloading tushare-1.4.19-py3-none-any.whl.metadata (3.1 kB)
Collecting pandas (from tushare)
  Downloading pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: requests in d:\anaconda\envs\prj1\lib\site-packages (from tushare) (2.32.3)
Collecting lxml (from tushare)
  Downloading lxml-5.3.1-cp312-cp312-win_amd64.whl.metadata (3.8 kB)
Collecting simplejson (from tushare)
  Downloading simplejson-3.20.1-cp312-cp312-win_amd64.whl.metadata (3.4 kB)
Collecting bs4 (from tushare)
  Downloading bs4-0.0.2-py2.py3-none-any.whl.metadata (411 bytes)
Collecting websocket-client>=0.57.0 (from tushare)
  Downloading websocket_client-1.8.0-py3-none-any.whl.metadata (8.0 kB)
Collecting tqdm (from tushare)
  Downloading tqdm-4.67.1-py3-none-any.whl.metadata (57 kB)
Collecting beautifulsoup4 (from bs4->tushare)
  Downloading beautifulsoup4-4.13.3-py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: numpy>=1.26.0 in d:\anaconda\envs\prj1\lib\site-packages (from pandas->tushare) (2.2.4)
Collecting python-dateutil>=2.8.2 (from pandas->tushare)
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas->tushare)
  Downloading pytz-2025.1-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas->tushare)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
```

2.配置 PYPI 清华镜像

```
(prj1)
86150@GFJ MINGW64 ~
$ python -m pip install --upgrade pip                    检查升价pip到最新版本
Requirement already satisfied: pip in d:\anaconda\envs\prj1\lib\site-packages (25.0.1)
(prj1)
86150@GFJ MINGW64 ~                                      配置pypi清华镜像
$ pip config set global.index-url https://mirrors.tuna.tsinghua.edu.cn/pypi/web/simple
Writing to C:\Users\86150\AppData\Roaming\pip\pip.ini
```

```
polars                    1.26.0              pypi_0    pypi
polars-lts-cpu            1.26.0              pypi_0    pypi
pycparser                 2.22              pyh29332c3_1    conda-forge
pysocks                   1.7.1             pyh09c184e_7    conda-forge
python                    3.12.9          h3f84c4b_1_cpython    conda-forge
python-dateutil           2.9.0.post0         pypi_0    pypi
python_abi                3.12                5_cp312    conda-forge
pytz                      2025.1              pypi_0    pypi
requests                  2.32.3            pyhd8ed1ab_1    conda-forge
setuptools                75.8.2            pyhff2d567_0    conda-forge
simplejson                3.20.1              pypi_0    pypi
six                       1.17.0              pypi_0    pypi
soupsieve                 2.6                 pypi_0    pypi
tbb                       2021.13.0         h62715c5_1    conda-forge
tk                        8.6.13            h5226925_1    conda-forge
tqdm                      4.67.1              pypi_0    pypi
tushare                   1.4.19              pypi_0    pypi
typing-extensions         4.12.2              pypi_0    pypi
tzdata                    2025.2              pypi_0    pypi
ucrt                      10.0.22621.0      h57928b3_1    conda-forge
urllib3                   2.3.0             pyhd8ed1ab_0    conda-forge
vc                        14.3               hbf610ac_24    conda-forge
vc14_runtime              14.42.34438        hfd919c2_24    conda-forge
websocket-client          1.8.0               pypi_0    pypi
```

3.conda 环境导出

```
(prj1)
86150@GFJ MINGW64 ~
$ conda export -f environment.yml
```

environment.yml

```yaml
name: prj1
channels:
  - conda-forge
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r
  - https://repo.anaconda.com/pkgs/msys2
dependencies:
  - brotli-python=1.1.0=py312h275cf98_2
  - bzip2=1.0.8=h2466b09_7
  - ca-certificates=2025.1.31=h56e8100_0
  - certifi=2025.1.31=pyhd8ed1ab_0
  - cffi=1.17.1=py312h4389bb4_0
  - charset-normalizer=3.4.1=pyhd8ed1ab_0
  - h2=4.2.0=pyhd8ed1ab_0
  - hpack=4.1.0=pyhd8ed1ab_0
  - hyperframe=6.1.0=pyhd8ed1ab_0
  - idna=3.10=pyhd8ed1ab_1
  - intel-openmp=2024.2.1=h57928b3_1083
```

```
(prj1)
86150@GFJ MINGW64 ~
$ conda env export
name: prj1
channels:
  - conda-forge
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r
  - https://repo.anaconda.com/pkgs/msys2
dependencies:
  - brotli-python=1.1.0=py312h275cf98_2
  - bzip2=1.0.8=h2466b09_7
  - ca-certificates=2025.1.31=h56e8100_0
  - certifi=2025.1.31=pyhd8ed1ab_0
  - cffi=1.17.1=py312h4389bb4_0
  - charset-normalizer=3.4.1=pyhd8ed1ab_0
  - h2=4.2.0=pyhd8ed1ab_0
  - hpack=4.1.0=pyhd8ed1ab_0
  - hyperframe=6.1.0=pyhd8ed1ab_0
  - idna=3.10=pyhd8ed1ab_1
  - intel-openmp=2024.2.1=h57928b3_1083
  - libblas=3.9.0=31_h641d27c_mkl
  - libcblas=3.9.0=31_h5e41251_mkl
  - libexpat=2.6.4=he0c23c2_0
  - libffi=3.4.6=h537db12_0
  - libhwloc=2.11.2=default_ha69328c_1001
  - libiconv=1.18=h135ad9c_1
```

4. 移除 conda 环境

```
(base)
86150@GFJ MINGW64 ~
$ conda env remove -n prj1

Remove all packages in environment D:\anaconda\envs\prj1:


## Package Plan ##

  environment location: D:\anaconda\envs\prj1


The following packages will be REMOVED:
```

5. 利用导出环境，重建 conda 环境

```
(base)
86150@GFJ MINGW64 ~
$ cd repo
(base)
86150@GFJ MINGW64 ~/repo
$ ls -l
total 9
drwxr-xr-x 1 86150 197609   0  3月 18 15:51 mywork/
-rw-r--r-- 1 86150 197609 261  3月 11 23:42 script1.py
drwxr-xr-x 1 86150 197609   0  3月 12 00:27 week01/
drwxr-xr-x 1 86150 197609   0  3月 18 20:49 week02/
(base)
86150@GFJ MINGW64 ~/repo
$ mkdir prj1
(base)
86150@GFJ MINGW64 ~/repo
$ cd prj1
(base)
86150@GFJ MINGW64 ~/repo/prj1
$ mv ~/environment.yml ./
```

重建 conda 环境

```
(base)
86150@GFJ MINGW64 ~/repo/prj1
$ conda create
```

```
(base)
86150@GFJ MINGW64 ~/repo/prj1
$ conda env list
# conda environments:
#
base                  *  D:\anaconda
prj1                     D:\anaconda\envs\prj1
prj2                     D:\anaconda\envs\prj2
```

6. conda 与 python 的关系

- **核心概念差异**

Conda：它是一个跨平台的开源包管理系统和环境管理系统，能够在不同操作系统上使用。

Conda 并不局限于 Python 包管理，还可用于管理其他编程语言（如 R、Ruby、Java 等）的软件包，并且可以创建独立的运行

环境，确保各个项目的依赖不会相互干扰。

Python：这是一种高级、通用、解释型的编程语言，拥有简洁易读的语法和丰富的标准库与第三方库，在数据科学、机器学习、Web 开发、自动化脚本等众多领域广泛应用。

- **关联表现**

Python 环境管理：Conda 可以创建和管理不同版本的 Python 环境。你能使用 Conda 轻松创建包含特定 Python 版本（如 Python 3.7、Python 3.9 等）的虚拟环境。

Python 包管理：Conda 可以作为 Python 包的管理工具。它能从不同的渠道（如 Anaconda 官方渠道、Conda Forge 等）下载和安装 Python 包，并且会自动处理包之间的依赖关系。

环境隔离：Conda 的环境隔离特性对于 Python 项目至关重要。不同的 Python 项目可能依赖于不同版本的 Python 和第三方库，使用 Conda 创建独立的环境可以避免这些依赖之间的冲突。

- 相互补充

功能互补：虽然 pip 是 Python 官方的包管理工具，但它主要专注于 Python 包的管理，而 Conda 不仅可以管理 Python 包，还能管理其他语言的包以及系统级依赖。此外，Conda 在处理包的依赖关系和环境管理方面更为强大，能够更好地解决复杂的依赖问题。

使用场景互补：在一些需要严格控制环境和依赖的场景下，如科学研究、企业级开发等，Conda 的优势更为明显；而在一些轻量

级的 Python 项目中，pip 可能就足以满足需求。同时，在某些情况下，也可以将二者结合使用，充分发挥它们各自的优势。

Conda 环境是基于 Python 等编程语言构建的虚拟运行环境，是对 Python 编程环境的一种有效管理和扩展，它为 Python 项目提供了更灵活、更高效、更可靠的运行环境，有助于提高开发效率、减少错误，并方便项目的管理和协作。

7. conda forge 和 conda 的关系

Conda Forge 是一个与 Conda 相关的重要社区驱动的软件包渠道，与 Conda 有着紧密的合作关系。

1.Conda 是包管理系统，Conda Forge 是软件包仓库。

2.Conda Forge 扩展了 Conda 的软件包资源。

3.Conda 通过配置使用 Conda Forge。

4.Conda Forge 与 Conda 社区协作紧密。

8. python 解释器

Python 解释器是 Python 编程中的核心组件，它负责将人类编写的 Python 代码转化为计算机能够理解和执行的机器语言。

工作原理：Python 是一种解释型语言，这意味着代码在运行时逐行被解释执行。Python 解释器读取 Python 代码文件或交互式命令，对代码进行词法分析、语法分析等处理，然后将其转换为字节码（Bytecode）。字节码是一种中间形式的代码，类似于汇编语言，它比源代码更接近机器语言，但仍然独立于具体

的硬件平台。接着，解释器会执行这些字节码，将其转化为机器能够执行的指令。

常用 python 解释器：CPython、PyPy、Jython、IronPython

9. 第三方软件包

第三方软件包是指由 Python 社区或其他开发者创建并维护，不属于 Python 标准库的软件包。

常用第三方软件包：Numpy、pandas、Matplotlib、TensorFlow、Django

10.Pypi 软件仓库

PyPI 是一个集中式的存储库，收集了大量的第三方 Python 软件包。这些软件包涵盖了各种领域和功能，包括科学计算、数据分析、Web 开发、机器学习、图形界面设计等。开发者可以将自己开发的软件包上传到 PyPI，供其他用户下载和使用；用户则可以通过 PyPI 方便地获取所需的软件包，无需逐个从各个项目的官方网站下载。

许多 Python 软件包可能依赖于其他软件包才能正常运行。PyPI 能够自动处理软件包之间的依赖关系。当用户通过 pip 等工具安装一个软件包时，如果该软件包有依赖项，pip 会自动从 PyPI 下载并安装这些依赖项，确保软件包能够正确运行。

11.conda forge 和 pypi 如何选择

Conda forge 不仅提供 python 软件包，还涵盖了其他编程语言。在安装某些程序软件等超出 python 以外，但需要 python 调用，则使用 conda install。Pypi 是 python 官方的软

件包仓库，聚焦于 python 的软件包，拥有海量 python 库。

如果进行多语言项目，需要同时管理不同语言的包和依赖，或者对依赖管理和跨平台兼容性有较高要求，则 conda forge 更合适。

如果只专注于 python 开发，只需要管理 python 包，并且对安装速度要求不是特别高，那么 Pypi 是很好的选择，它提供了丰富的 python 库资源。

专题五 编写和运行 Python 程序

1. 创建本地项目文件夹

```yaml
! environment.yml ×
! environment.yml
name: myproject
channels:
  - conda forge
dependencies:
  - python=3.12
```

```
(base) 86150@GFJ MINGW64 ~/repo/myproject
$ conda activate myproject
(myproject)
86150@GFJ MINGW64 ~/repo/myproject
$ conda list
# packages in environment at D:\anaconda\envs\myproject:
#
# Name                    Version                   Build  Channel
bzip2                     1.0.8                h2466b09_7    conda-forge
ca-certificates           2025.1.31            h56e8100_0    conda-forge
libexpat                  2.6.4                he0c23c2_0    conda-forge
libffi                    3.4.6                h537db12_0    conda-forge
liblzma                   5.6.4                h2466b09_0    conda-forge
libsqlite                 3.49.1               h67fdade_2    conda-forge
libzlib                   1.3.1                h2466b09_2    conda-forge
openssl                   3.4.1                ha4e3fda_0    conda-forge
pip                       25.0.1             pyh8b19718_0    conda-forge
python                    3.12.9          h3f84c4b_1_cpython    conda-forge
setuptools                75.8.2             pyhff2d567_0    conda-forge
tk                        8.6.13               h5226925_1    conda-forge
tzdata                    2025b                h78e105d_0    conda-forge
ucrt                      10.0.22621.0         h57928b3_1    conda-forge
vc                        14.3                hbf610ac_24    conda-forge
vc14_runtime              14.42.34438         hfd919c2_24    conda-forge
wheel                     0.45.1             pyhd8ed1ab_1    conda-forge
```

2. 创建 python 程序

```json
C: > Users > 86150 > AppData > Roaming > Code > User > {} settings.json > ...
{
    "workbench.startupEditor": "none",
    "workbench.colorTheme": "Visual Studio Dark",
    "[python]": {
        "editor.formatOnSave": true,
        "editor.codeActionsOnSave": {
        "source.fixAll": "explicit",
        "source.organizeImports": "explicit"
        },
        "editor.defaultFormatter": "charliermarsh.ruff"
    },

    "notebook.formatOnSave.enabled": true,
    "notebook.codeActionsOnSave": {
        "notebook.source.fixAll": "explicit",
        "notebook.source.organizeImports": "explicit"
    }
}
```

配置 ruff

```
main.py > ...
def main():
    print("Hello, conda!")


if __name__ == "__main__":
    main()

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS        Python Debug Console
ocuments\WindowsPowerShell\profile.ps1'
+    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\86150\repo\myproject>  & 'd:\anaconda\envs\myproject\python.exe' 'c:\Users\86150\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '58151' '--' 'C:\Users\86150\repo\myproject\main.py'
Hello, conda!
PS C:\Users\86150\repo\myproject>
```

输出“hello，conda”

```
(myproject)
86150@GFJ MINGW64 ~/repo/myproject
$ python main.py
Hello, conda!
(myproject)
```

```
(myproject)
86150@GFJ MINGW64 ~/repo/myproject
$ python main.py
Hello, conda!
2.2.3
D:\anaconda\envs\myproject\Lib\site-packages\pandas\__init__.py
```

3. 脚本运行：

下载网址对应文件（curl）

```
86150@GFJ MINGW64 ~/repo/myproject
$ curl -O https://edg.epa.gov/EPADataCommons/public/OA/EPA_SmartLocationDatabase_V3_Jan_2021_Final.csv
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  192M  100  192M    0     0   399k      0  0:08:16  0:08:16 --:--:--  899k
```

EPA_SmartLocationDatabase_V3_Jan_2021_Final.csv

```
OBJECTID,GEOID10,GEOID20,STATEFP,COUNTYFP,TRACTCE,BLKGRPCE,CSA,CSA_Name,CBSA,CBSA_Nam
1,4.8113E+11,4.8113E+11,48,113,7825,4,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
2,4.8113E+11,4.8113E+11,48,113,7825,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
3,4.8113E+11,4.8113E+11,48,113,7825,3,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
4,4.8113E+11,4.8113E+11,48,113,7824,1,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
5,4.8113E+11,4.8113E+11,48,113,7824,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
6,4.8113E+11,4.8113E+11,48,113,7827,1,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
7,4.8113E+11,4.8113E+11,48,113,9301,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
8,4.8113E+11,4.8113E+11,48,113,1102,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-Fo
9,4.8113E+11,4.8113E+11,48,113,11401,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-F
10,4.8113E+11,4.8113E+11,48,113,11401,3,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
11,4.8113E+11,4.8113E+11,48,113,11500,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
12,4.8113E+11,4.8113E+11,48,113,11500,1,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
13,4.8113E+11,4.8113E+11,48,113,12301,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
14,4.8113E+11,4.8113E+11,48,113,12301,3,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
15,4.8329E+11,4.8329E+11,48,329,304,3,372,"Midland-Odessa, TX",33260,"Midland, TX",16
16,4.8329E+11,4.8329E+11,48,329,305,2,372,"Midland-Odessa, TX",33260,"Midland, TX",16
17,4.8113E+11,4.8113E+11,48,113,12302,1,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
18,4.8113E+11,4.8113E+11,48,113,12302,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
19,4.8113E+11,4.8113E+11,48,113,13105,3,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
20,4.8113E+11,4.8113E+11,48,113,13104,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
21,4.8113E+11,4.8113E+11,48,113,13400,1,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
22,4.8113E+11,4.8113E+11,48,113,13607,2,206,"Dallas-Fort Worth, TX-OK",19100,"Dallas-
23,4.8329E+11,4.8329E+11,48,329,305,1,372,"Midland-Odessa, TX",33260,"Midland, TX",16
24,4.8329E+11,4.8329E+11,48,329,305,5,372,"Midland-Odessa, TX",33260,"Midland, TX",16
```

运行

main.py > ...

```
import pandas as pd


def main():
    """
    Answers the question:                                   函数的文档

    What percentage of U.S. residents live highly walkable neighborhoods?

    "15.26" is the threshold on the index for a highly walkable area.
    """
    csv_file = "./EPA_SmartLocationDatabase_V3_Jan_2021_Final.csv"   指定路径
    highly_walkable = 15.26

    df = pd.read_csv(csv_file)     读取文件

    total_population = df["TotPop"].sum()       求和
    highly_walkable_pop = df[df["NatWalkInd"] >= highly_walkable]["TotPop"].sum()

    percentage = (highly_walkable_pop / total_population) * 100.0

    print(
        f"{percentage:.2f}% of U.S. residents live in highly" "walkable neighborhoods."
    )


if __name__ == "__main__":
    main()                                  程序入口
```

代码解释：4-10 行：

定义一个名为 main 的函数，函数的文档字符串对其功能进行了说明，即回答“有多少百分比饿美国居民住在高度适宜步行的社区？”，并且支出高度适宜步行区域的指数阈值为 15.26

csv_file 变量指定了要读取的 CSV 文件的路径

highly_walkable 变量储存了高度示意步行区域的指数与之

18 行：

df[“NatWalkInd”]>=highly_walkable 是一个布尔索引，用于筛选出 NatWalkInd 列中值大于或等于 highly_walkable 的行，这写行代表高度适宜步行的区域。

对筛选后的行进行选取 TotPop 列，并使用 sum 方法求和，得到居住在高度适宜步行区域的人口总数，结果存储于 highly_walkable_pop 变量中。

结果

```
(myproject)
86150@GFJ MINGW64 ~/repo/myproject
$ python main.py
10.69% of U.S. residents live in highlywalkable neighborhoods.
```