Python-Conda使用教程

相关文档：

Cheat sheet

Conda Documentation

https://docs.conda.io/projects/conda/en/stable/commands/index.html

前言

Python 的管理主要涉及版本管理、环境管理和包管理。这些管理任务对于保持开发环境的一致性、解决依赖性问题和避免版本冲突至关重要。以下是一些关键的工具和概念，帮助你有效管理 Python。

版本管理

由于不同的项目可能需要不同版本的 Python，版本管理允许你在同一台机器上安装和使用多个 Python 版本。

- **pyenv**：是一个流行的 Python 版本管理工具，允许你在同一台机器上安装和切换多个版本的 Python。它非常适合那些需要在不同项目之间切换使用不同 Python 版本的用户。

环境管理

环境管理涉及创建隔离的 Python 环境，每个环境都有自己的库版本集合。这对于处理不同项目的依赖性管理非常有用。

- **virtualenv**：是一个创建隔离的 Python 环境的工具。每个环境都可以有自己的依赖版本，而不会干扰其他环境。
- **Conda**：是一个跨平台的包和环境管理器，不仅可以用于 Python，还可以用于其他语言。Conda 允许你创建隔离的环境，以便不同的项目可以有自己的依赖版本。

包管理

包管理涉及安装、更新和管理 Python 库和依赖。

- **pip**：是 Python 的官方包管理工具，用于安装和管理来自 Python 包索引（PyPI）的包。
- **Conda**：除了环境管理外，Conda 也是一个强大的包管理工具，可以处理来自 Anaconda 仓库的包。Conda 特别适合于数据科学和机器学习项目，因为它可以轻松地安装在这些领域常用的大型库。

实践建议

- **依赖性文件**：对于大多数项目，使用 `requirements.txt`（pip）或 `environment.yml`（Conda）来管理项目依赖是一个好习惯。这样可以确保其他人（或你自己在不同环境中）可以轻松地重现环境。

- **虚拟环境**：始终在虚拟环境中工作，以避免依赖冲突和不一致性问题。这对于保持全局 Python 环境的干净和管理多个项目至关重要。
- **版本控制**：使用版本控制（如 Git）来管理你的代码和依赖文件，这样你可以跟踪依赖的变化并在需要时回滚到早期的状态。

Conda 的安装、更新和卸载

conda 分为 anaconda 和 miniconda，anaconda 是一个包含了许多常用库的集合版本，miniconda 是精简版本（只包含conda、pip、zlib、python 以及它们所需的包），剩余的通过 conda install command 命令自行安装即可；

1、区别：Conda、Miniconda 和 Anaconda

Conda、Miniconda 和 Anaconda 是常用于科学计算和数据科学的Python环境和包管理工具。它们之间有一些关键的区别：

1. **Conda**:

- Conda是一个开源的包管理系统和环境管理系统，用于安装、运行和升级包和环境。它是语言无关的，虽然最初是为Python项目设计的，但也可以用来打包和分发软件包的其他语言。
- Conda可以帮助用户在不同的项目之间切换环境，确保依赖关系不冲突。

2. **Miniconda**:
    - Miniconda是Conda的一个最小化安装版本。它包括Conda、Python以及少数必要的库，但不包括Anaconda发行版中预装的大量数据科学库和工具。
    - Miniconda是一个轻量级的替代品，适用于希望自定义安装必要包的用户。通过Miniconda，用户可以创建具有所需包的环境，而不需要下载Anaconda发行版中的所有预装包。

3. **Anaconda**:

   - Anaconda是一个开源的Python和R语言的分发版，专为科学计算（数据科学、机器学习应用、大数据处理等）而设计。
   - Anaconda包括Conda、Python以及一系列预安装的库和工具，如NumPy、Pandas、SciPy、Matplotlib等，以及用于数据科学和机器学习的IDE（如Jupyter Notebook）。
   - Anaconda是最适合希望直接使用大量数据科学库的用户，而无需单独安装每个库。

2、安装conda

miniconda 官网： https://conda.io/miniconda.html

anaconda 官网： https://www.anaconda.com/download

点击安装即可，不需要另外安装 Python运行环境，安装过程中，出现 Advanced options选项，第一个选项是将Anaconda的路径加入环境变量，第二个是默认将conda安装的 Python 定为系统使用的默认版本：

Latest - Conda 23.11.0 Python 3.11.5 released December 20, 2023

3、添加镜像源

在 Python 的包管理工具中添加镜像源，可以加速包的下载速度，尤其是在某些地区访问默认源可能较慢或受限的情况下。下面将介绍如何在 pip 和 Conda 中添加镜像源。

对于 pip

1. **命令行指定镜像源**:
   在使用 pip 安装包时，可以使用 `-i` 选项临时指定镜像源。例如，使用清华大学的镜像源安装 flask：

```python
pip install flask -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2. **永久修改配置文件**:
   你可以修改 pip 的配置文件（在 Linux 或 macOS 上通常位于 `~/.pip/pip.conf`，在 Windows 上位于 `%USERPROFILE%\pip\pip.ini`），添加镜像源以便永久使用。
   例如，添加清华大学镜像源的配置如下：

   - 在 Linux 或 macOS 的 `~/.pip/pip.conf` 添加：

     PYTHON

     ```
     [global]
     index-url = https://pypi.tuna.tsinghua.edu.cn/simple
     ```

   - 在 Windows 的 `%USERPROFILE%\pip\pip.ini` 添加：

     PYTHON

     ```
     [global]
     index-url = https://pypi.tuna.tsing
     ```

```
hua.edu.cn/simple
```

对于 Conda：

Conda 允许通过 `.condarc` 配置文件或命令行来添加和使用镜像源。一些常用的 Conda 镜像源包括清华大学、中科大等。

1. **命令行添加镜像源**:
   使用以下命令添加镜像源（例如，添加清华大学的镜像源）：

```python
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --set show_channel_urls yes   #windows 用户无法直接创建 .condarc 文件，需要通过指令
```

这会将镜像源添加到你的 `.condarc` 文件中，并设置 Conda 以显示每次操作使用的镜像源 URL。

2. **编辑 `.condarc` 文件**:
   如果需要手动编辑或查看 `.condarc` 文件，可以在用户主目录下找到（如果文件不存在，可以手动创建一个）。添加镜像源，例如：

```yaml
channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
show_channel_urls: yes
```

这样配置后，pip 或 Conda 将默认使用指定的镜像源来下载和安装包，这通常可以显著提高下载速度和提升访问稳定性。

目前国内提供conda镜像的大学

清华大学: https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/

北京外国语大学: https://mirrors.bfsu.edu.cn/help/anaconda/

南京邮电大学: https://mirrors.njupt.edu.cn/

南京大学: http://mirrors.nju.edu.cn/

重庆邮电大学: http://mirror.cqupt.edu.cn/

上海交通大学: https://mirror.sjtu.edu.cn/

哈尔滨工业大学: http://mirrors.hit.edu.cn/#/home

要查看已添加的 Conda 通道（channels），你可以使用以下几种方法：

1. **使用命令行指令**：
   在 Anaconda Prompt 或其他支持 Conda 的命令行界面中，输入以下命令：

```python
conda config --get channels
```

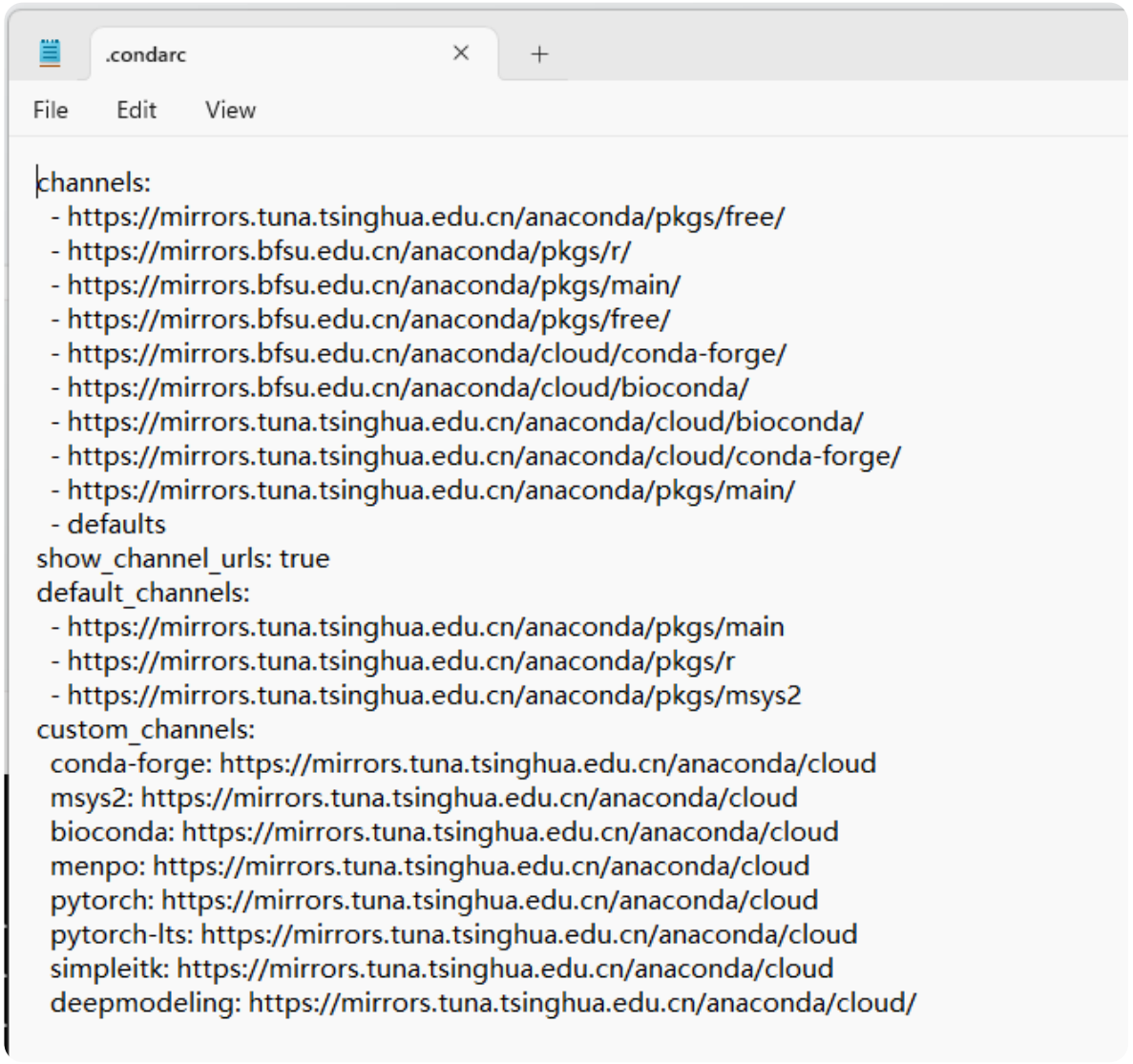
这个命令会列出所有已配置的通道。

```
(base) C:\Users\wengy>conda config --get channels
--add channels 'defaults'   # lowest priority
--add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/'
--add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge/'
--add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/bioconda/'
--add channels 'https://mirrors.bfsu.edu.cn/anaconda/cloud/bioconda/'
--add channels 'https://mirrors.bfsu.edu.cn/anaconda/cloud/conda-forge/'
--add channels 'https://mirrors.bfsu.edu.cn/anaconda/pkgs/free/'
--add channels 'https://mirrors.bfsu.edu.cn/anaconda/pkgs/main/'
--add channels 'https://mirrors.bfsu.edu.cn/anaconda/pkgs/r/'
--add channels 'https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/'   # highest priority
```

2. **查看配置文件**：

   Conda 的通道配置通常存储在用户主目录下的 `.condarc` 文件中。你可以使用文本编辑器打开该文件查看已添加的通道。在 Windows 上，文件路径通常是 `C:\Users\<你的用户名>\.condarc`；在 macOS 或 Linux 上，文件路径通常是 `~/.condarc`。

   打开文件后，你应该会看到类似下面的内容：



在这个 `channels` 列表下，你会看到所有已添加的通道。

3. **使用 `conda info` 命令**：
   虽然 `conda info` 主要用于显示 Conda 的详细信息，但它也会包括已配置的通道信息。执行 `conda info` 命令后，在输出信息中查找 "channels" 部分，那里会列出已配置的通道。

无论使用哪种方法，你都可以找到已添加的 Conda 通道列表。这些通道是 Conda 在搜索和安装包时所使用的源。如果你需要修改或删除通道，可以使用 `conda config --add channels` 来添加新通道，或使用 `conda config --remove channels` 来删除已添加的通道。

恢复默认镜像源：`conda config --remove-key channels`。

注释：

1. `conda info`：
    - 这个命令用于显示当前Conda安装的系统信息。它提供了一系列有用的信息，包括：
        - Conda版本：显示当前安装的Conda版本。
        - Python版本：显示Conda使用的Python版本。
        - 平台：显示操作系统类型（如Windows、Linux或macOS）。
        - 用户的Conda环境：列出系统中所有可用的Conda环境。
        - 激活的环境：显示当前激活的Conda环境。

        - 配置文件的位置：显示Conda配置文件（如`.condarc`）的路径。
        - 包缓存的位置：显示Conda包缓存的路径，Conda下载的包会被缓存以便将来使用。
    - 这个命令非常有用，特别是在诊断环境配置问题或者了解当前Conda设置的上下文中。
2. `conda config --get channels`：
    - Conda的`channels`是包的仓库，用户可以从中安装包。这个命令用于列出在Conda配置中设置的channel列表。
    - Channels在查找和安装包时按照优先级排序，这个命令还可以显示这些channels的优先级。

- 默认情况下，Conda使用`defaults` channel，但用户可以通过修改`.condarc`配置文件或使用`conda config`命令添加额外的channels，比如`conda-forge`。
- 了解哪些channels被配置和它们的优先级对于确保从预期源安装软件包很重要，特别是在多个channels提供同一软件包的版本时。

3. `conda config --set show_channel_urls yes`

   可以配置 Conda 以在安装包时显示每个包的来源 channel 的 URL。这是一个有用的设置，因为它可以帮助你了解安装的包来自哪个 channel，特别是在你配置了多个 channels 并希望跟踪包版本来源时。

此设置将被添加到你的 `.condarc` 配置文件中，这是 Conda 用于存储用户级别配置选项的文件。设置此选项为 `yes` 后，每当你使用 Conda 安装、更新或列出包时，Conda 将显示与每个包关联的 channel URL。

这个功能对于确保你的环境中的包来自于可信赖的 sources 非常有帮助，特别是在使用第三方 channels 如 `conda-forge` 时，你可能想要确认包的确切来源。此外，如果你在解决依赖问题或确保环境的一致性时遇到困难，了解包的来源 channel 也可能是一个有用的诊断工具。

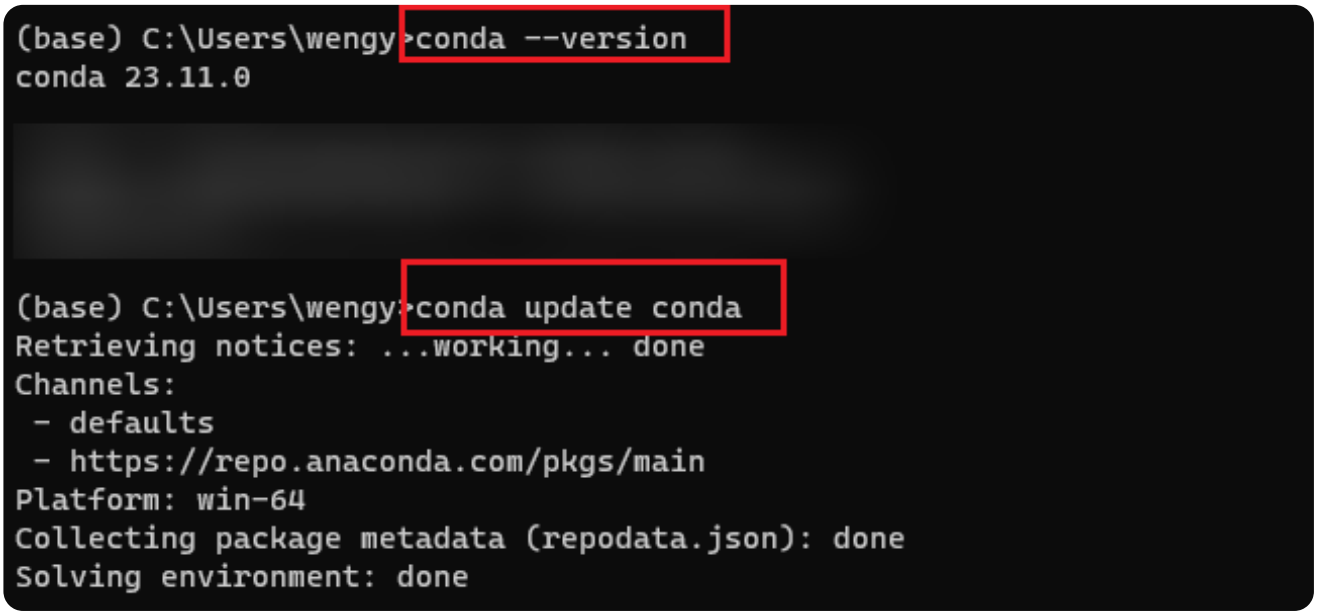
如果你之后想要关闭这个选项，可以使用相似的命令将 `yes` 改为 `no`：

```python
conda config --set show_channel_urls no
```

4、查看，更新 conda

查看版本号： `conda --version`

升级版本号： `cnoda update conda`



5、卸载 conda

对于 Windows 用户：

1. **通过控制面板卸载**:
    - 打开“控制面板” > “程序” > “程序和功能”，找到 Miniconda 并选择“卸载”。

2. **手动删除**:
    - 如果通过控制面板卸载后，可能还需要手动删除 Miniconda 目录（例如 `C:\Users\你的用户名\Miniconda3`）来确保所有文件都被移除。

- 删除任何相关的环境变量。右击“计算机” > “属性” > “高级系统设置” > “环境变量”，然后查找与 Miniconda 相关的任何条目并删除。

对于 macOS 和 Linux 用户：

1. **删除安装目录**:
   - 打开终端。
   - 使用 `rm -rf` 命令删除 Miniconda 的安装目录。默认情况下，Miniconda 可能安装在你的主目录下，如 `~/miniconda3` 或 `~/opt/miniconda3`。执行命令类似于 `rm -rf ~/miniconda3`。请确保你使用正确的路径，以避免误删除其他文件。
2. **编辑 `.bash_profile` 或 `.bashrc` 文件**:

- Miniconda 安装过程中会在你的 `.bash_profile`、`.bashrc` 或 `.zshrc` 文件中添加初始化脚本。你需要手动编辑这些文件并删除相关行。
- 使用文本编辑器打开对应的文件，如 `nano ~/.bashrc`，然后删除所有与 `conda` 初始化相关的行。

3. **更新或清理环境变量**:
   - 对于 bash 用户，执行 `source ~/.bashrc`（或重新启动终端）来应用更改。
   - 对于 zsh 用户，执行 `source ~/.zshrc`（或重新启动终端）来应用更改。

清理环境变量（适用于所有操作系统）：

确保从你的环境变量中删除任何与 Miniconda 相关的路径。这通常涉及编辑 `.bash_profile`、`.bashrc` 或系统的环境变量设置。

conda 管理环境

利用 Conda 可以创建单独的环境，其中包含不与其他环境交互的文件、包及其依赖项。当开始使用 conda 时，已经默认创建了一个 `base` 的环境，但是，如果不想将程序放入这个基础环境（`base`）中，可以选择创建单独的环境以使程序彼此隔离。

1、查看当前所有环境：

基本命令： `conda info --envs` **或** `conda info -e` **或** `conda env list`

```
(base) C:\Users\wengy>conda info --envs
# conda environments:
#
PY                       C:\Users\[redacted]\.conda\envs\PY
base                  *  D:\ProgramData\miniconda3


(base) C:\Users\wengy>conda info -e
# conda environments:
#
PY                       C:\Users\[redacted]\.conda\envs\PY
base                  *  D:\ProgramData\miniconda3
```

2、创建环境：

基本命令： `conda create --name env_name python='py_version'`

```
(base) C:\Users\wengy>conda create -n py python=3.10.6
Channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
 - defaults
 - https://mirrors.bfsu.edu.cn/anaconda/pkgs/free
 - https://mirrors.bfsu.edu.cn/anaconda/cloud/conda-forge
 - https://mirrors.bfsu.edu.cn/anaconda/cloud/bioconda
 - bioconda
 - conda-forge
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\wengy\.conda\envs\py

  added / updated specs:
    - python=3.10.6
```

指定创建环境目录，使用`--prefix`参数。

PYTHON

```
 conda create --prefix /path/to/env_directory python='py_version'
```

```
(base) C:\Users\wengy>conda create --prefix D:\ProgramData\mypyenv\py python=3.8
Channels:
 - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free
 - defaults
 - https://mirrors.bfsu.edu.cn/anaconda/pkgs/free
 - https://mirrors.bfsu.edu.cn/anaconda/cloud/conda-forge
 - https://mirrors.bfsu.edu.cn/anaconda/cloud/bioconda
 - bioconda
 - conda-forge
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: D:\ProgramData\mypyenv\py

  added / updated specs:
    - python=3.8
```

查看环境：

```
(base) C:\Users\wengy>conda env list
# conda environments:
#
py                       C:\Users\wengy\.conda\envs\py
base                  *  D:\ProgramData\miniconda3
                         D:\ProgramData\mypyenv\py
```

注意：

当你使用 `--prefix` 参数来创建 Conda 环境时，你实际上是直接指定了环境的安装路径，而不是使用 Conda 的默认环境目录并给环境命名。因此，使用 `--prefix` 创建环境时，Conda 不会让你通过一个额外的参数来指定环境名称，因为环境的“名称”在这种情况下实际上是由其安装路径定义的。

如果你想要在创建环境的同时指定一个环境名称，而不是使用 `--prefix` 来指定路径，你应该使用 `-n` 或 `--name` 选项来创建环境，并给它一个名称。

3、激活和退出环境：

PYTHON

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

使用 `--prefix` 选项指定环境的完整路径，激活这个环境时

PYTHON

```
# To activate this environment, use
#
#     $ conda activate D:\ProgramData\mypyenv\py
#
# To deactivate an active environment, use
```

```
#
#      $ conda deactivate
```

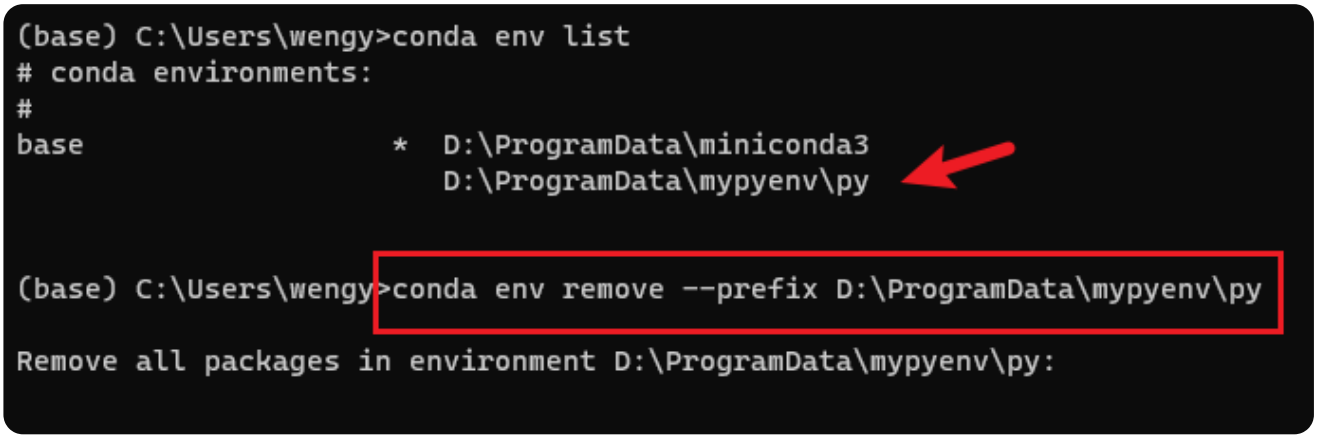
4、删除指定环境：

基本命令： `conda remove --name env_name --all`

```
(base) C:\Users\[redacted]>conda env list
# conda environments:
#
py                       C:\Users\wengy\.conda\envs\py
base                  *  D:\ProgramData\miniconda3


(base) C:\Users\[redacted]>conda remove --name py --all

Remove all packages in environment C:\Users\[redacted]\.conda\envs\py:


## Package Plan ##

  environment location: C:\Users\[redacted]\.conda\envs\py


The following packages will be REMOVED:

  bzip2-1.0.8-he774522_0
  ca-certificates-2023.12.12-haa95532_0
  libffi-3.4.4-hd77b12b_0
  openssl-1.1.1w-h2bbff1b_0
  pip-23.3.1-py310haa95532_0
  python-3.10.6-hbb2ffb3_1
  setuptools-68.2.2-py310haa95532_0
  sqlite-3.41.2-h2bbff1b_0
  tk-8.6.12-h2bbff1b_0
  tzdata-2023d-h04d1e81_0
  vc-14.2-h21ff451_1
  vs2015_runtime-14.27.29016-h5e58377_2
  wheel-0.41.2-py310haa95532_0
  xz-5.4.5-h8cc25b3_0
  zlib-1.2.13-h8cc25b3_0


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

使用 `--prefix` 参数创建的 Conda 环境，进行删除

基本命令： `conda env remove --prefix /path/to/env_directory`



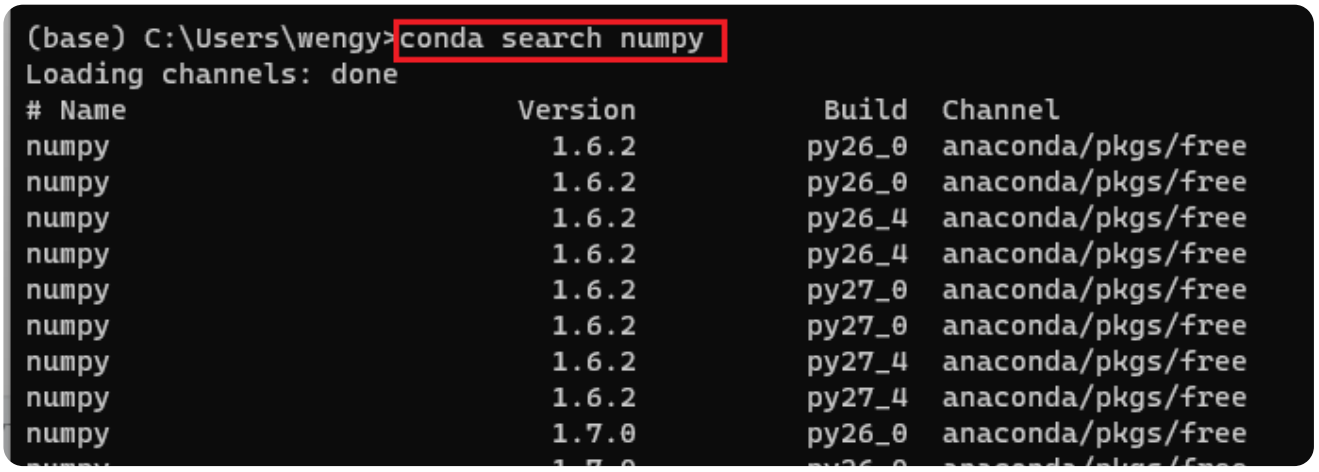
5、Conda 包管理

检索 Package ..

`conda search` **命令用于在 Conda 仓库中搜索可用的包。你可以使用它来查找特定包的可用版本，或者检查某个包是否存在于 Conda 的公共仓库中。这个命令非常有用，尤其是在你需要确定哪个版本的包与你的环境兼容时。**

基本用法 ::

基本命令： `conda search package_name`



高级搜索 ::

- **搜索特定版本的包：**如果你想要搜索特定版本的包，可以使用 `=` 符号。例如，搜索 `numpy` 版本 1.18.1 的命令如下：

```python
conda search 'numpy=1.18.1'
```

- **限制搜索到特定平台**：你也可以限制搜索结果只显示特定平台（如 linux-64 或 win-32）的包。这通常在你为跨平台环境准备包时非常有用。
- **使用正则表达式**：`conda search` 支持使用正则表达式来进行模糊搜索。这在你不确定包的完整名称时非常有用。

搜索特定通道中的包

如果你想要在特定的通道中搜索包，可以使用 `--channel` 或 `-c` 选项指定通道。例如，搜索 `conda-forge` 通道中的 `numpy` 包：

PYTHON

```
conda search -c conda-forge numpy
```

查看包信息

`conda search` 还可以用来显示包的详细信息，包括它的依赖关系。这可以通过增加 `--info` 选项来完成。

PYTHON

```
conda search --info numpy
```

注意

`conda search` 命令显示的信息可能会根据你的 Conda 配置和你添加的通道而有所不同。确保你的 `.condarc` 文件或环境设置正确，以便找到你需要的包版本。

安装 Package

`conda install` **命令用于在 Conda 环境中安装包。这个命令非常灵活，允许你指定安装包的版本、来源通道（channel），以及具体的环境。**

基本用法

基本命令： `conda install package_name`

指定版本

如果你需要安装特定版本的包，可以这样指定：

PYTHON

```
conda install package_name=version_number
```

指定环境

如果你想要在特定的环境中安装包，而不是当前激活的环境，可以使用 `-n` 或 `--name` 选项指定环境名称，或使用 `--prefix` 选项指定环境的路径。例如，安装包到名为 `myenv` 的环境：

PYTHON

```
conda install -n myenv package_name
```

从特定通道安装

有时你可能需要从特定的通道安装包，特别是当这个包在默认通道中不可用或你需要的是通道中的特定版本时。使用 `-c` 或 `--channel` 选项可以指定通道：

PYTHON

```
conda install -c channel_name package_name
```

例如，从 `conda-forge` 安装包：

PYTHON

```
conda install -c conda-forge package_name
```

安装多个包

你可以一次性安装多个包，只需要在命令中列出所有需要安装的包名：

PYTHON

```
conda install package_name1 package_name2
```

更新包

虽然 `conda update package_name` 是更新包的标准方式，你也可以通过重新运行 `conda install package_name` 来更新到最新版本的包。