**目录**

[【1 完整安装流程】 2](#_Toc28442)

[1.1 检查环境配置 2](#_Toc10715)

[1.2 确定自己需要安装的pytorch版本 2](#_Toc18097)

[1.3 安装cuda 3](#_Toc2130)

[1.4 安装cudann 5](#_Toc15537)

[1.5 环境变量的介绍 7](#_Toc11967)

[1.7 anaconda的安装（pycharm的安装） 11](#_Toc31303)

[1.8 anaconda的路径设置（下载包会比较快） 15](#_Toc1287)

[1.9 虚拟环境的创建 15](#_Toc25960)

[1.9 虚拟环境（补充） 17](#_Toc10323)

[1.10 虚拟环境的使用（jupyter） 17](#_Toc16795)

[1.11 虚拟环境的使用（pycharm） 22](#_Toc25887)

[1.12 安装pytorch框架 25](#_Toc11309)

[1.13 python自己包的配置命令 26](#_Toc800)

[1.14 pyG安装方式 26](#_Toc30496)

[1.15 R相关配置 26](#_Toc30984)

[1.16 项目路径设置 26](#_Toc16451)

[1.17 linux命令总结 27](#_Toc4011)

[1.18 Python→exe文件 27](#_Toc20055)

[1.19 VS连接远程服务器 27](#_Toc30489)

[1.20 服务器端配置rpy2模块 32](#_Toc19511)

[1.21 github项目包配置（setup.py方法） 34](#_Toc9629)

[1.22 虚拟环境调试 39](#_Toc31319)

[1.23 已经安装了某个包import 还是报错解决方案 40](#_Toc22482)

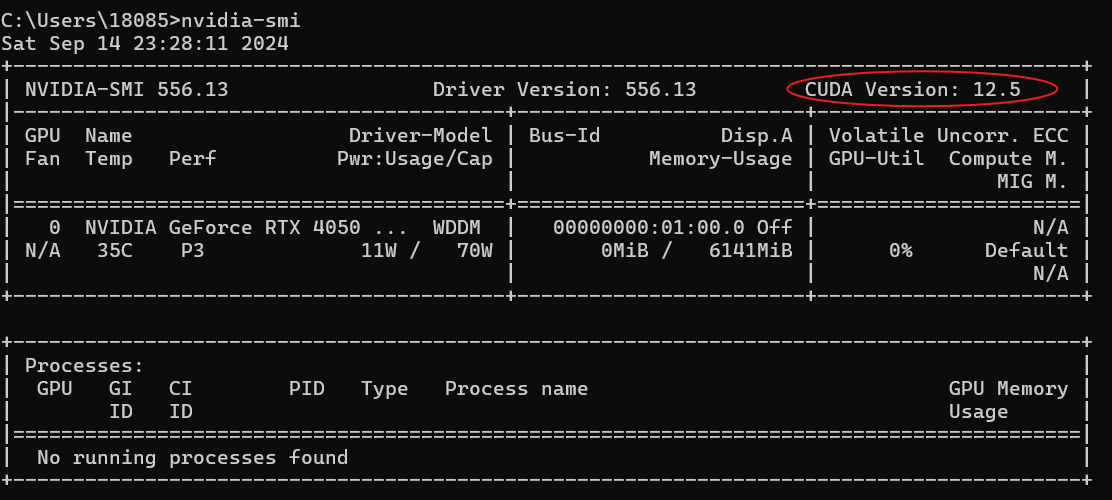
**【1 完整安装流程】**

Windows操作系统、NVIDIA显卡、c盘内存空闲>10G

**1.1 检查环境配置**

“Windows+r”，输入“cmd”进入控制台输入：

nvidia-smi



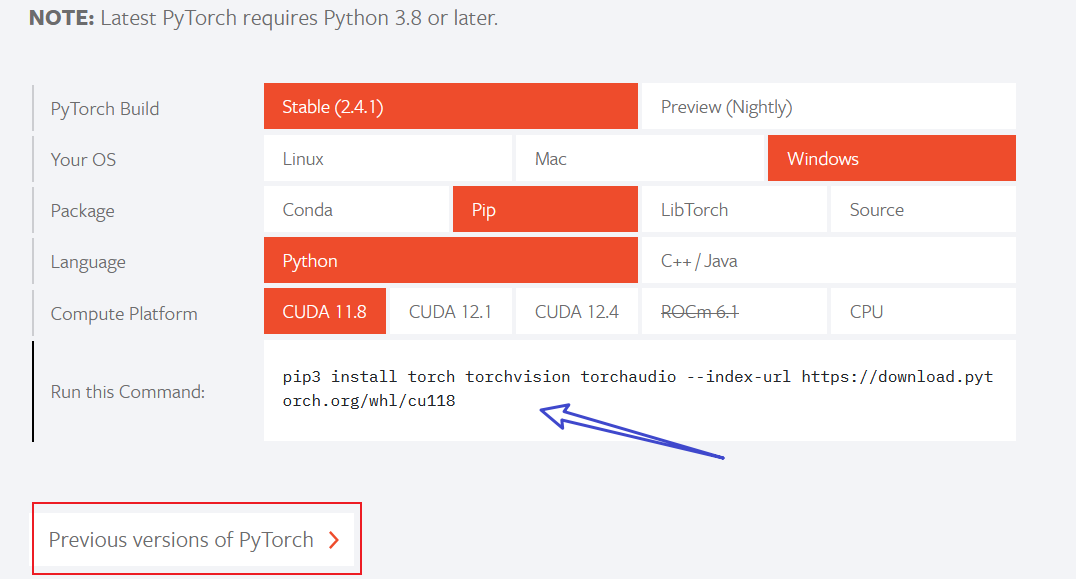
该电脑支持cuda的最高版本。

**1.2 确定自己需要安装的pytorch版本**

打开网址：

<https://pytorch.org/>

下滑找到你想要的那个版本，确定你可以安装的cuda版本



蓝色箭头就是你的安装指令，红色方框是更早的版本。

现在只是确定pytorch版本，请不要安装pytorch！！！

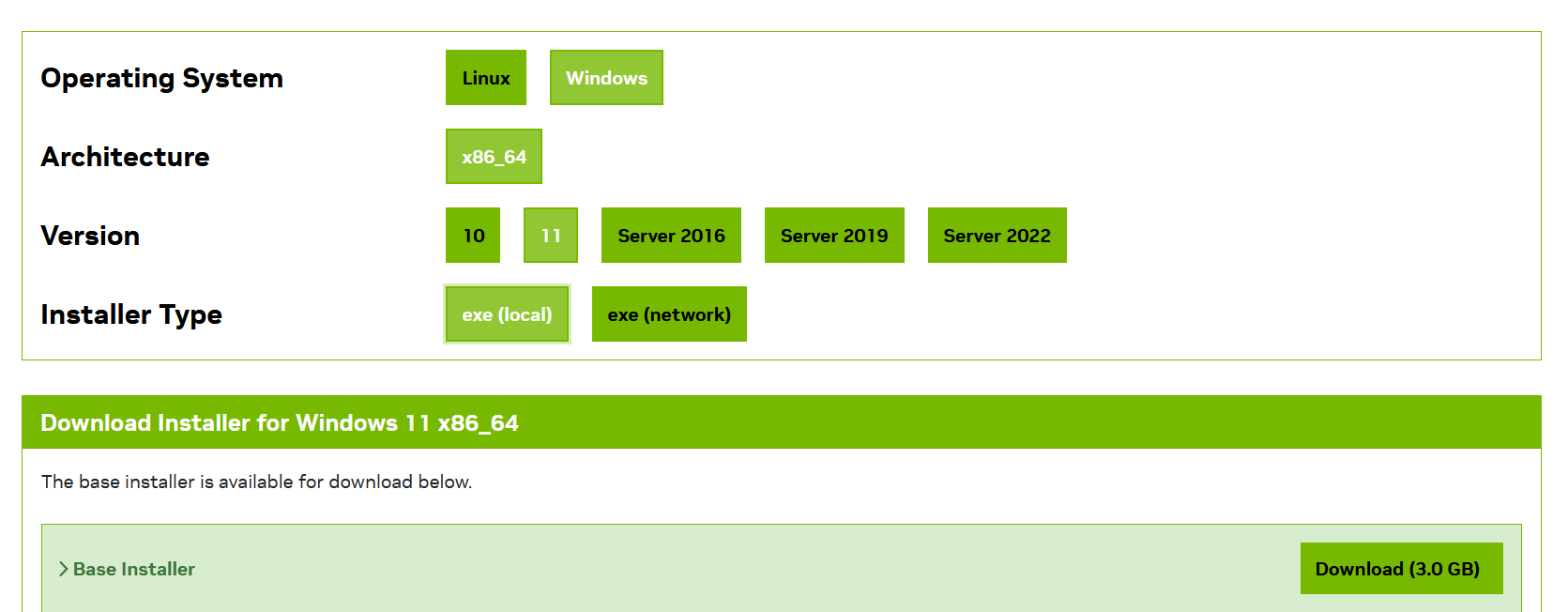
**1.3 安装cuda**

进入cuda官网：

<https://developer.nvidia.com/cuda-toolkit-archive>



选择自己想要的cuda版本，点击，选择自己的系统（推荐local版本），下载。



安装路径可以自行选择，安装选项“自定义”→**后续都勾选**。





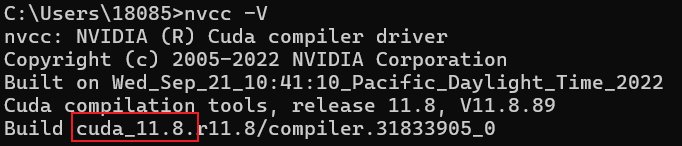
在安装CUDA工具包时，"Visual Studio Integration"选项是为了让CUDA与Visual Studio集成，这样可以在Visual Studio中更方便地创建、编译和调试CUDA项目。是否勾选取决于具体需求。如果不确定，那么默认勾选通常是一个安全的选择，尤其是对于初学者来说，这样可以简化开发流程。

一般第一次安装后会要你重启电脑。

打开控制台，输入：

nvcc -V

就可以看到安装的版本了。



这个时候就表明cuda安装成功了。

或者控制台输入：

echo %CUDA\_PATH%

会返回你的安装路径

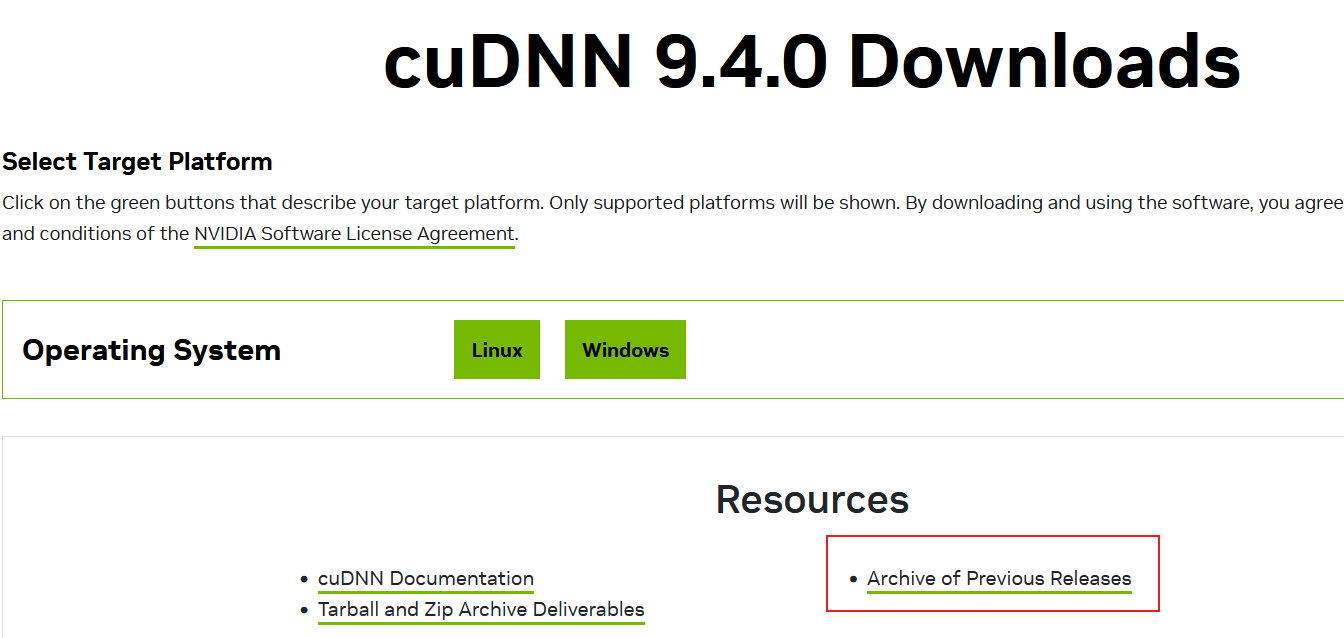


**1.4 安装cudann**

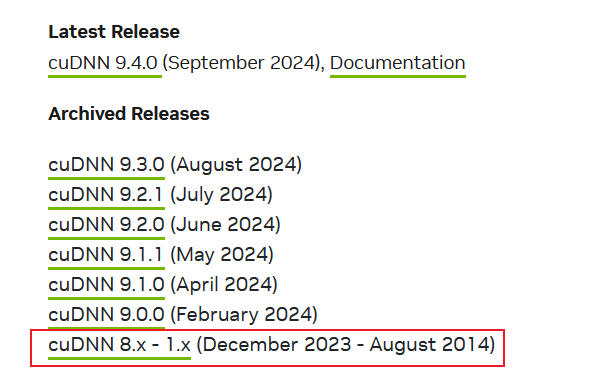
进入下载链接：

<https://developer.nvidia.com/cudnn-downloads>

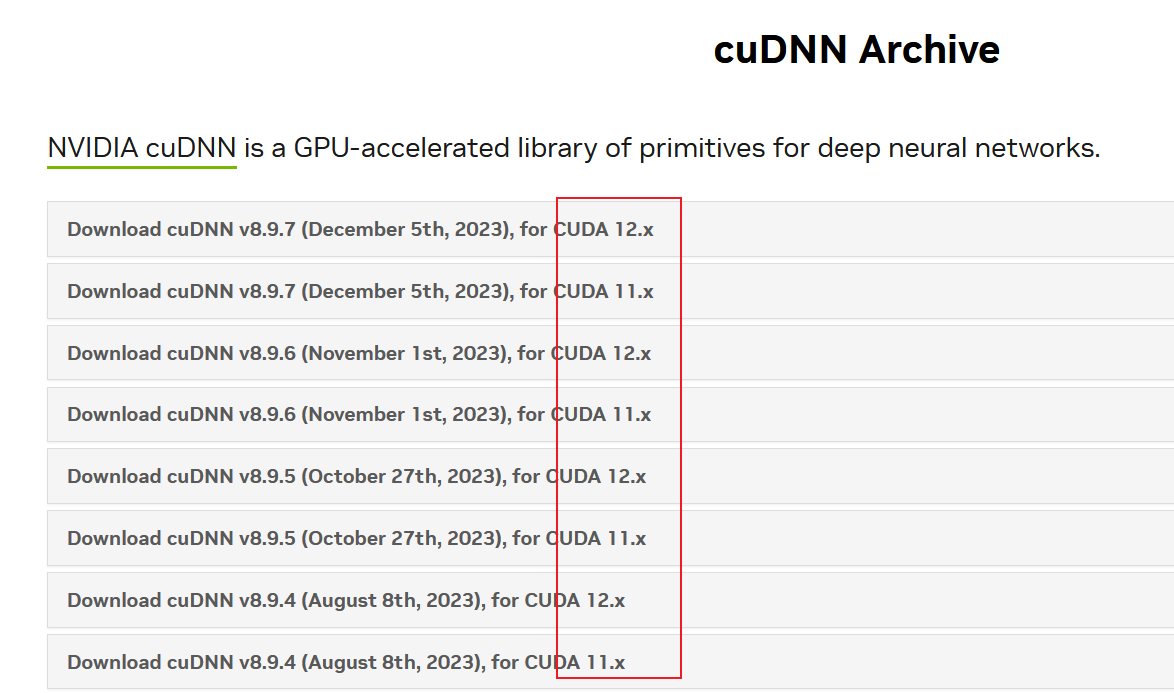
不一定需要下载最新的，点击即可进入老版本



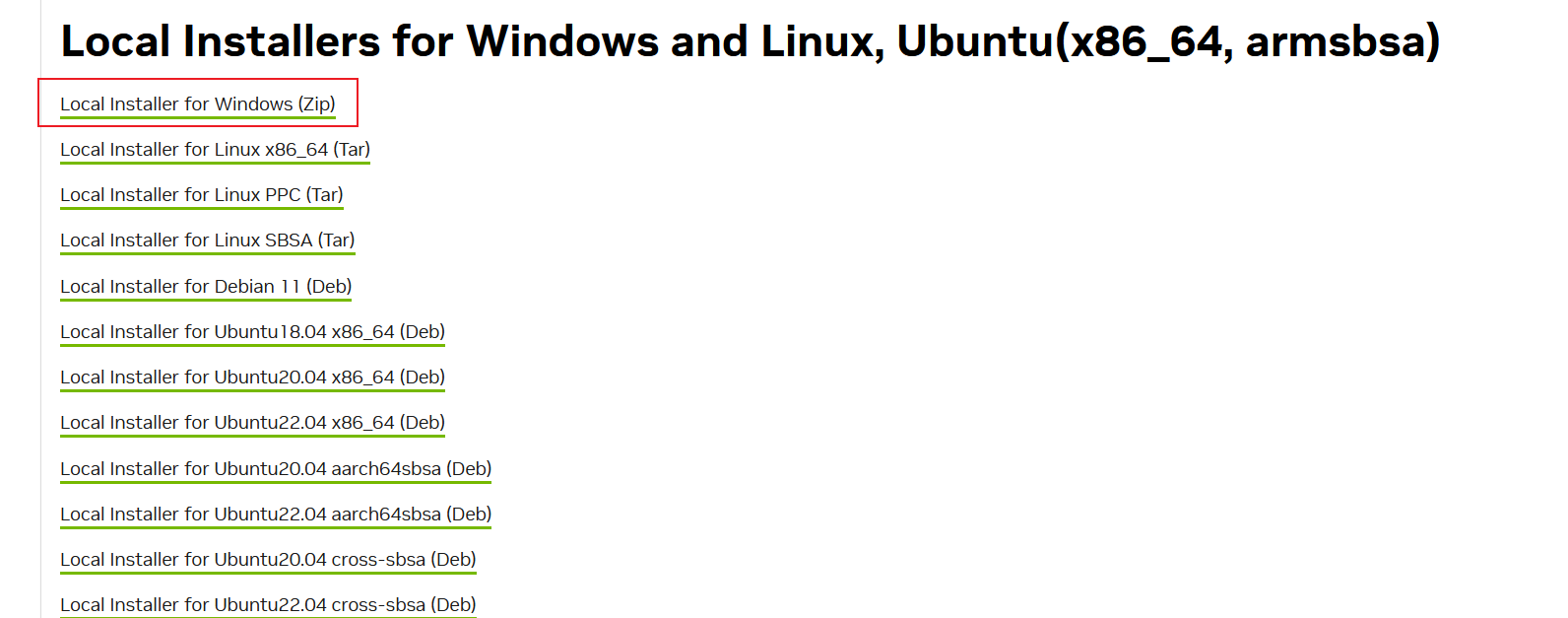
进入老版本：



查看其能够接受的cuda版本，后面的数字就表示cudann能够匹配的cuda版本：



选择所需要的版本，点击下载Windows版本：



注意，此时下载是需要注册账号的，按照他的步骤进行注册即可。

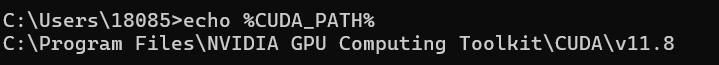
下载下来的是一个安装包，解压，里面的文件有：



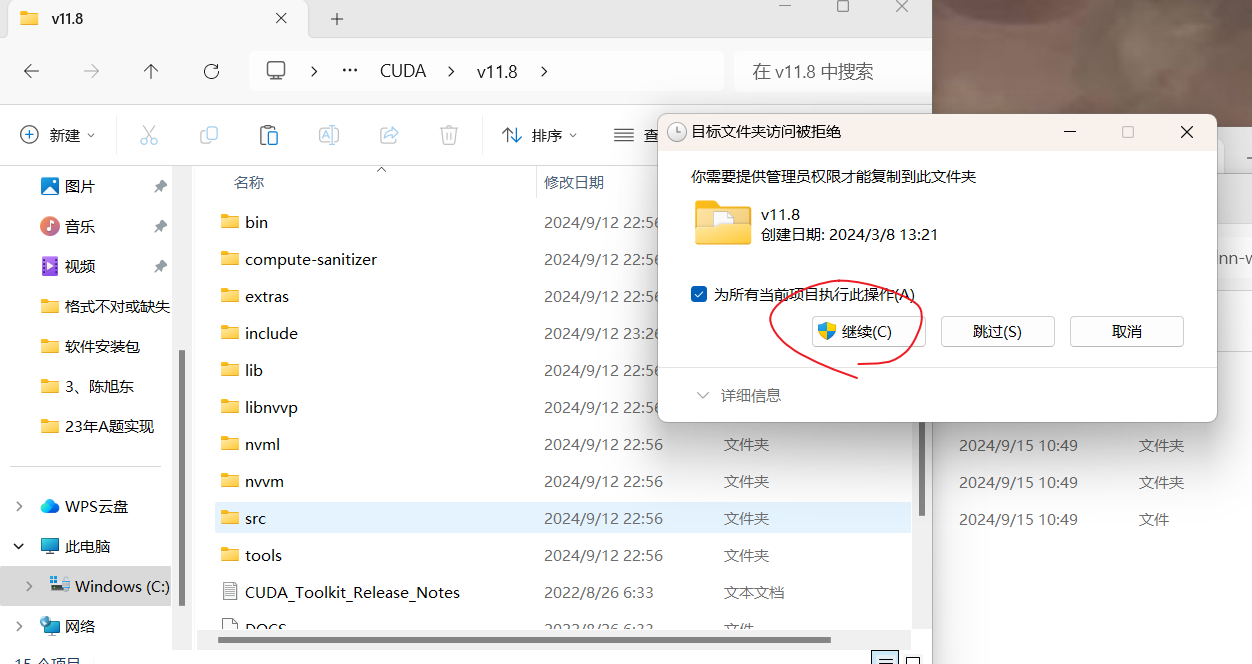
如果cuda装在默认路径，请将上述文件复制到路径C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v（版本号）即可。如果是其他路径，请在控制台输入：

echo %CUDA\_PATH%

会返回你的安装路径：



复制到该路径，并且覆盖原文件即可。



**1.5 环境变量的介绍**

Windows的环境变量是操作系统中的一个功能，它存储了用于操作系统和应用程序运行时的一系列变量值。环境变量可以用来定义系统范围内的设置，如程序路径、系统文件位置、控制台窗口属性等。初学者一般遇到问题就是缺少部分环境变量，系统不知道去哪找。

**路径设置（Path变量）**：PATH环境变量列出了操作系统在执行命令时搜索可执行文件的位置。当你输入一个命令时，操作系统会按照PATH变量中的目录顺序查找对应的程序。

%SystemRoot%：Windows操作系统的安装目录。

%ProgramFiles%：大多数程序默认安装的目录。

%APPDATA%：存储应用程序数据的目录。

%TEMP%或%TMP%：用于存储临时文件的目录。

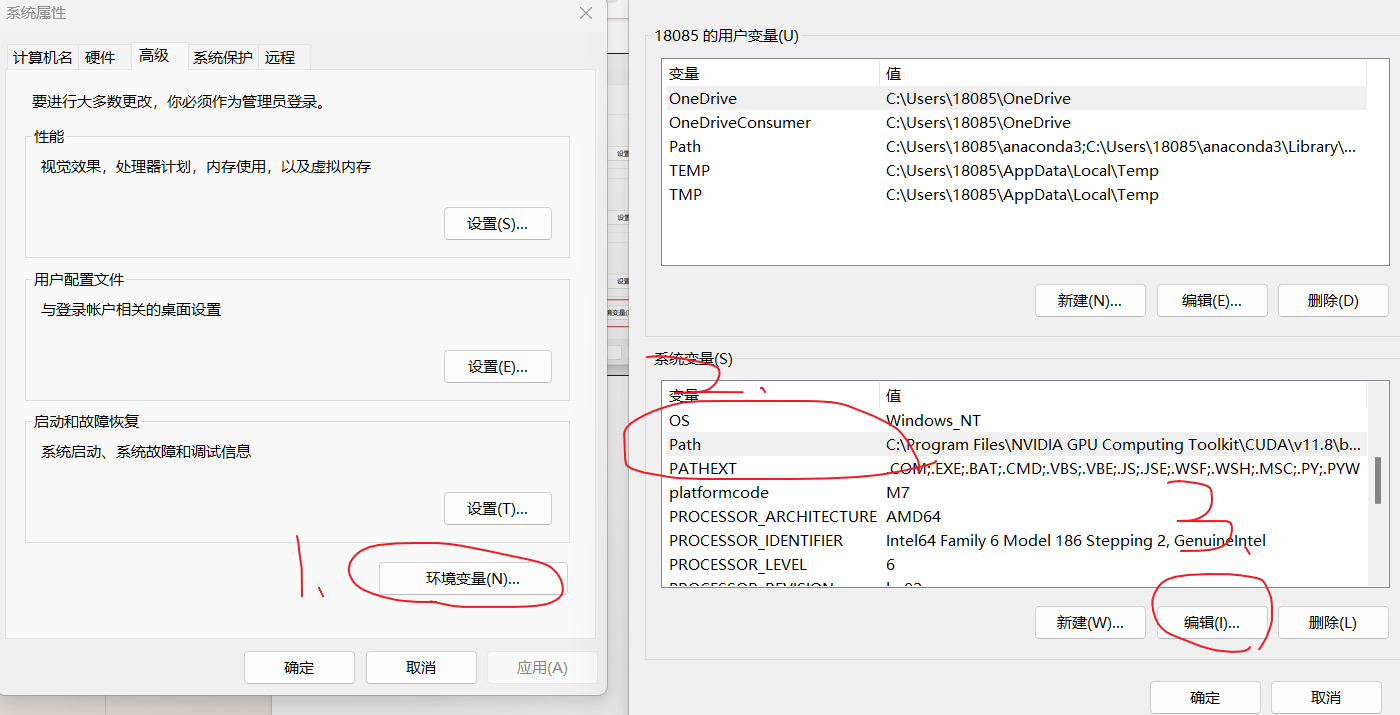
**%PATH%：执行命令时要搜索的目录列表**。

打开环境变量：

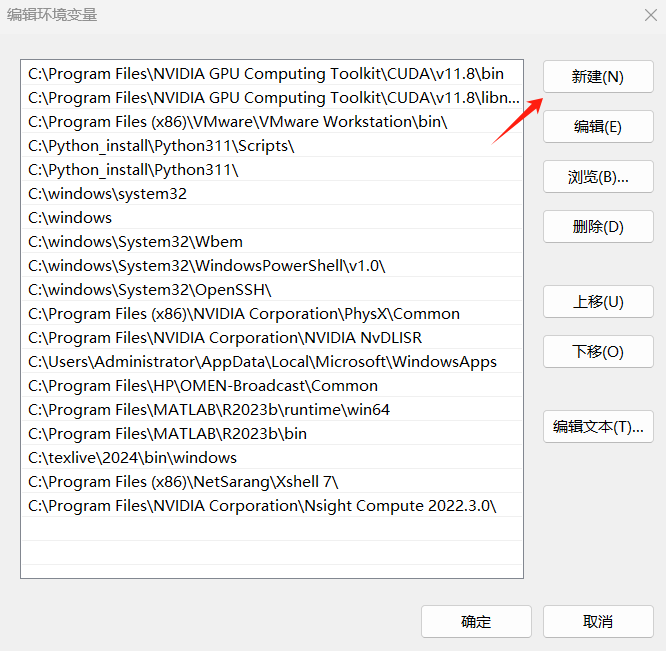
搜索“环境变量”，打开：



进入，点击红框：



点“新建”就可以添加环境变量了。



最后记得点击“确定”即添加成功。

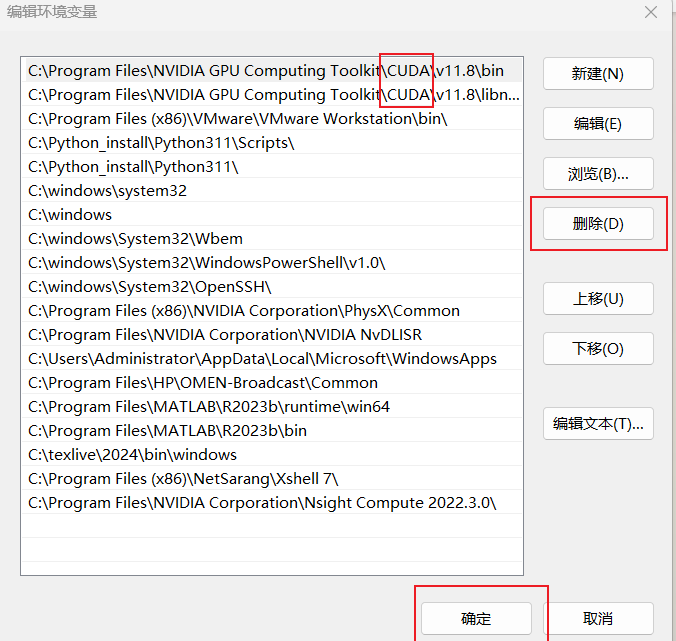
**1.6 cuda版本的替换**

如果你发现你需要一个其他的cuda版本，以下介绍cuda的卸载（不太推荐安装多个cuda，后面大项目一般都在服务器跑）。

1.在应用中搜索“cuda”，显示出来的全部卸载：

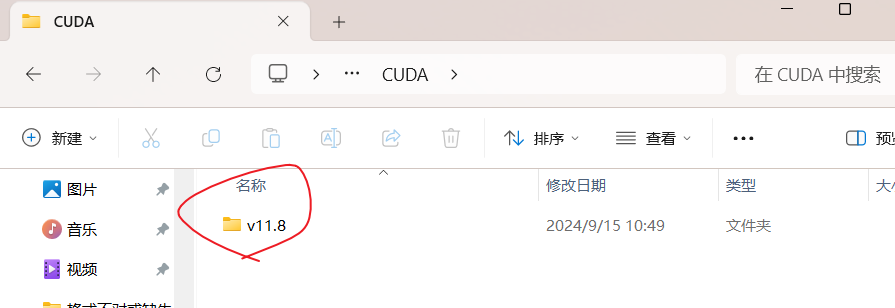


2.在环境变量中包含“cuda”的路径全部删除



3.在安装目录中删除原文件夹

找到你的安装目录，将这个文件夹删除。



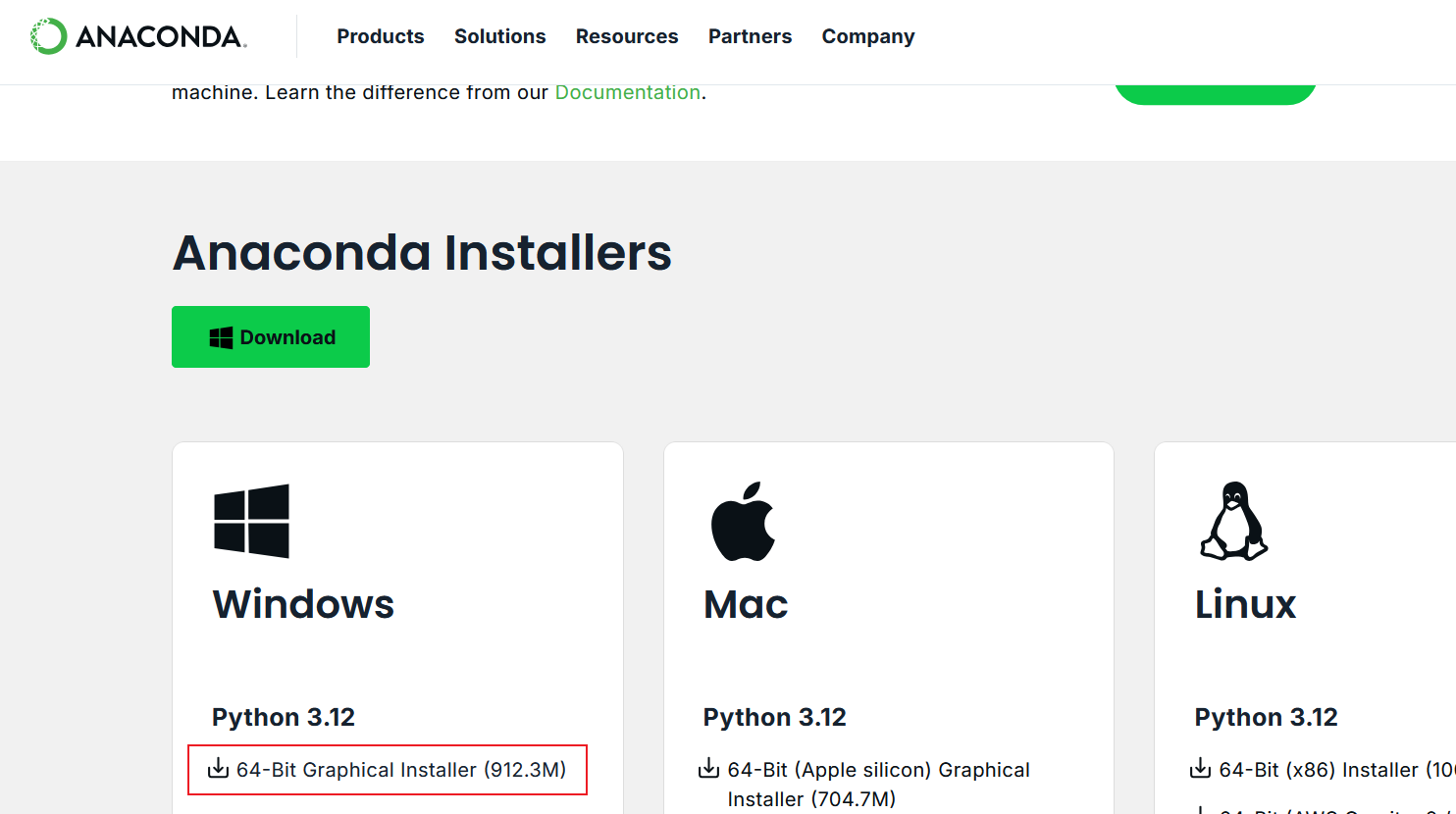
之后再重新安装就可以了。

**1.7 anaconda的安装（pycharm的安装）**

anaconda提供了我们所需要的各种东西，能够快捷、方便的管理各方面的东西，注意是不需要单独下载python的。

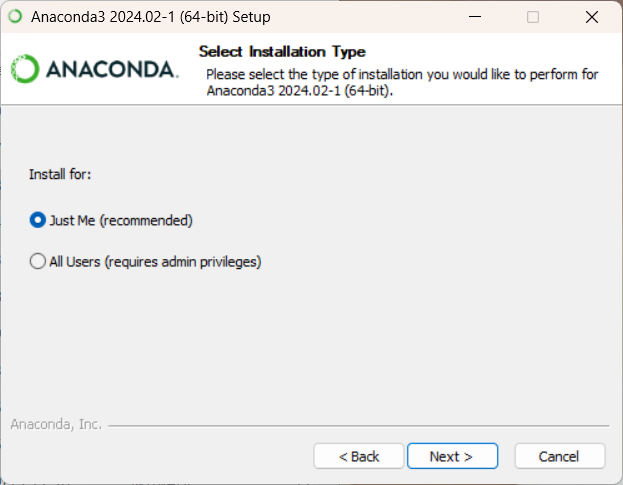
下载地址：

<https://www.anaconda.com/download/success>



安装过程：

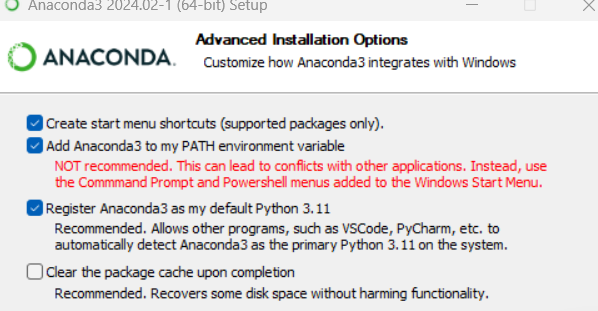
新手建议选择“just me”，选择另一个会比较麻烦



此时建议不要改路径（内存不够就去D盘）：



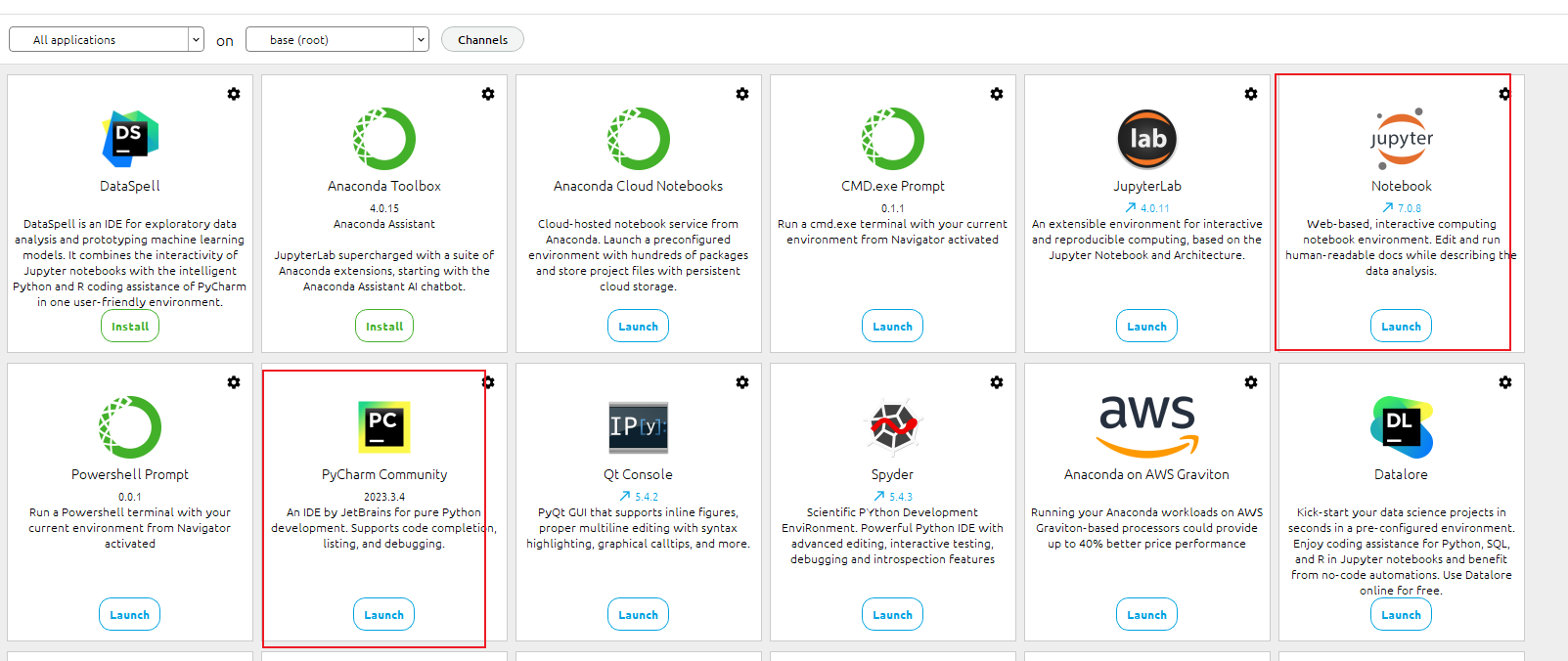
前三项打钩：



安装成功后，打开：



里面就有我们想要的各种东西，当然我们只需要jupyter和pycharm（社区版），点击“launch”启动，就能自动帮我们下载了。



现在我们来验证conda是否可用，在控制台输入：

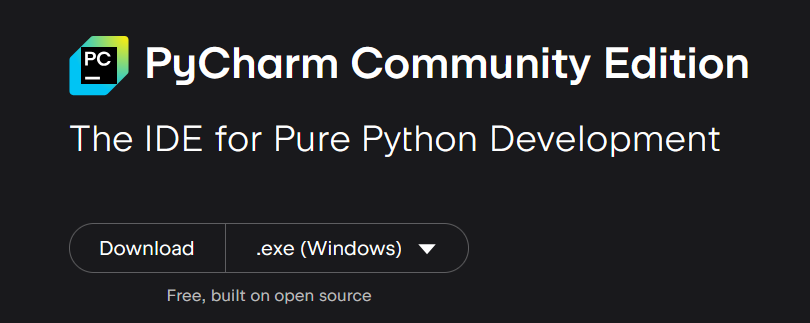
conda -V



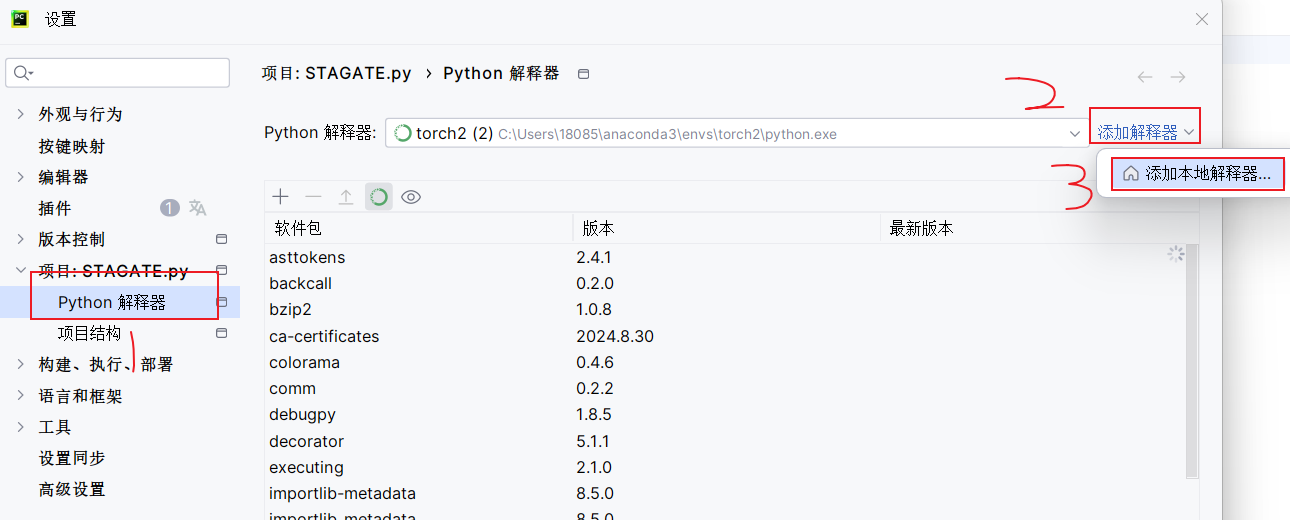
如果能正确显示就表明安装成功了。

如果anaconda没有显示pycharm community版本的，需要自行下载安装：

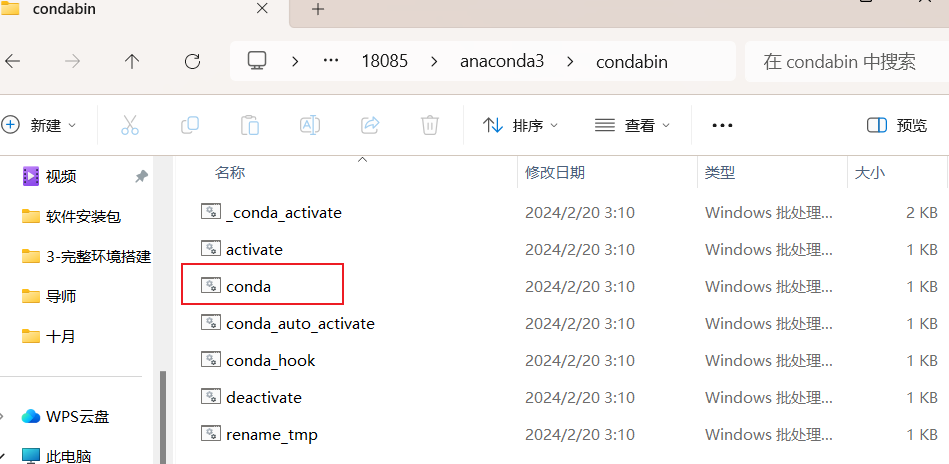
<https://www.jetbrains.com/pycharm/>

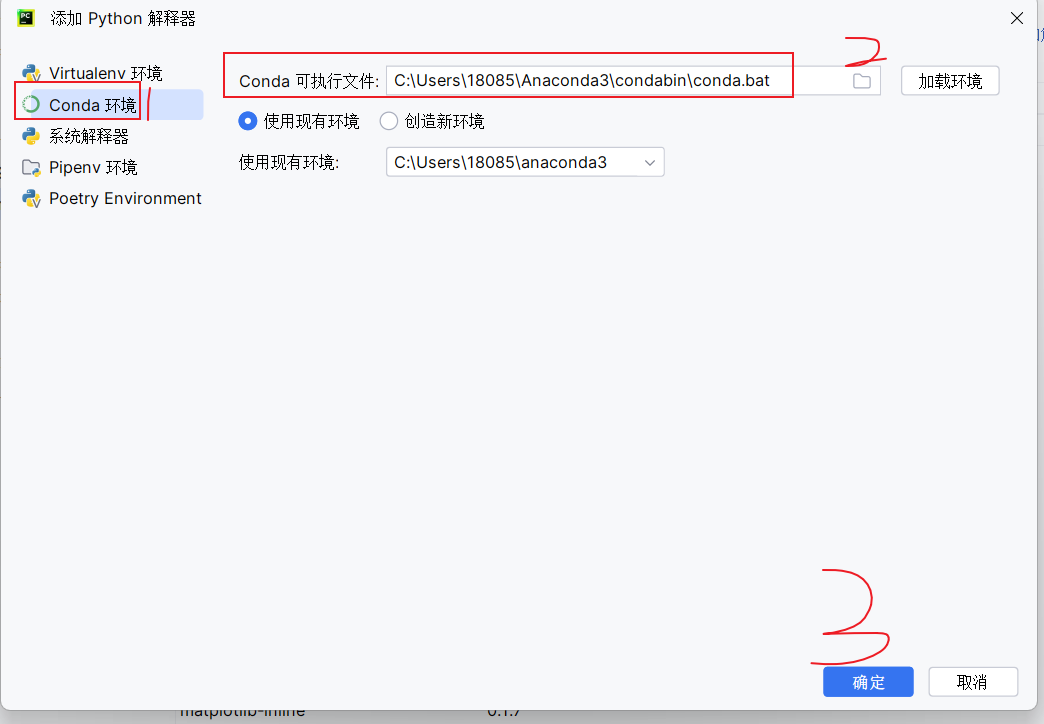


注意要选择社区版，然后安装完之后配置anaconda环境：



选择conda环境→找到你python安装的环境路径，找到类似下面的路径，点击确定即可。





**1.8 anaconda的路径设置（下载包会比较快）**

如果不设置，并且不搭梯子，那样每次都要设置镜像源，比较麻烦。

参考：

<https://blog.csdn.net/weixin_44914727/article/details/130513081>

[以下以pip安装为例，conda安装建议不要修改，一个就够了]

临时设置镜像源：

pip install some-package -i <https://pypi.tuna.tsinghua.edu.cn/simple>

some-package表示你要装的包

永久设置镜像源（base环境）：

pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple

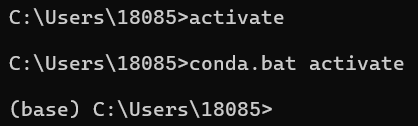
那样以后pip都是从清华镜像安装了。

**1.9 虚拟环境的创建**

推荐不同项目使用不同的虚拟环境，后面吃亏了就懂了。

在控制台输入：

activate

即可进入conda环境

如果进不去，请将：

C:\Users\18085\anaconda3

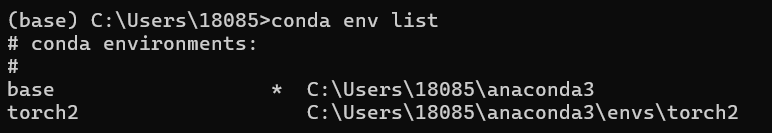
C:\Users\18085\anaconda3\Scripts

加入环境变量。

请注意base是最基础的conda环境，请不要在里面随意安装各种包。

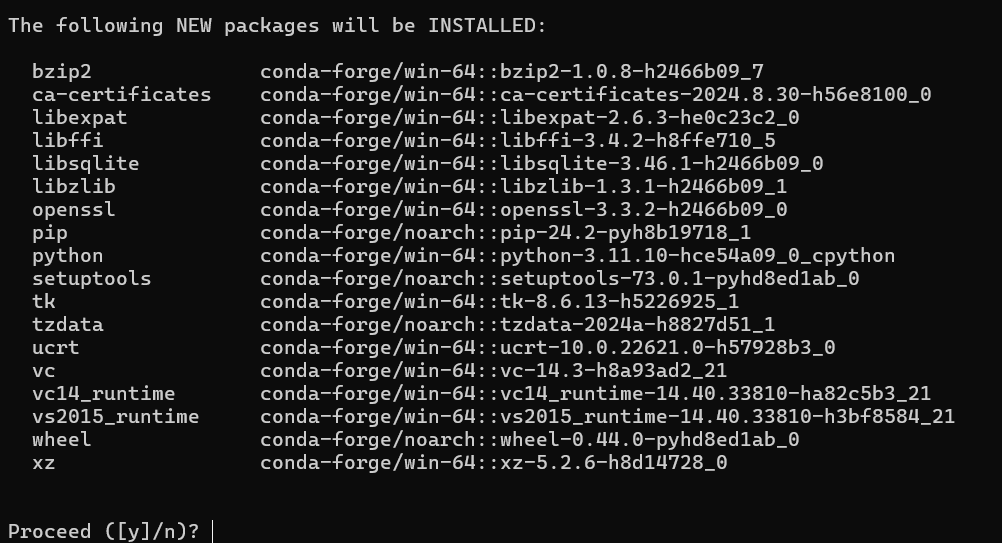
查看虚拟环境列表：

conda env list



创建自己的虚拟环境：

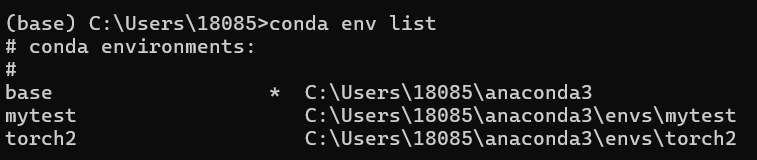
conda create -n mytest python=3.11



输入“y”按下回车即可。

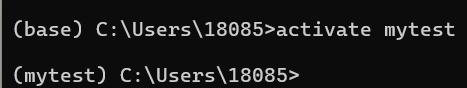
再次输入：

conda env list



可以看到我们的虚拟环境创建成功了。进入虚拟环境：

activate mytest



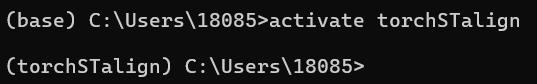
然后就可以在mytest这个虚拟环境进行相关操作了。

各个虚拟环境之间相互独立，这个虚拟环境安装的包不会在另一个虚拟环境中显示出来。更多虚拟环境的操作在这里就不赘述了，需要的时候自行查阅。

**1.9 虚拟环境（补充）**

克隆虚拟环境，相当于创建一个虚拟环境一模一样的副本，eg：将torch3里面所有的包克隆到新环境torchSTalign：

conda create --name torchSTalign --clone torch3

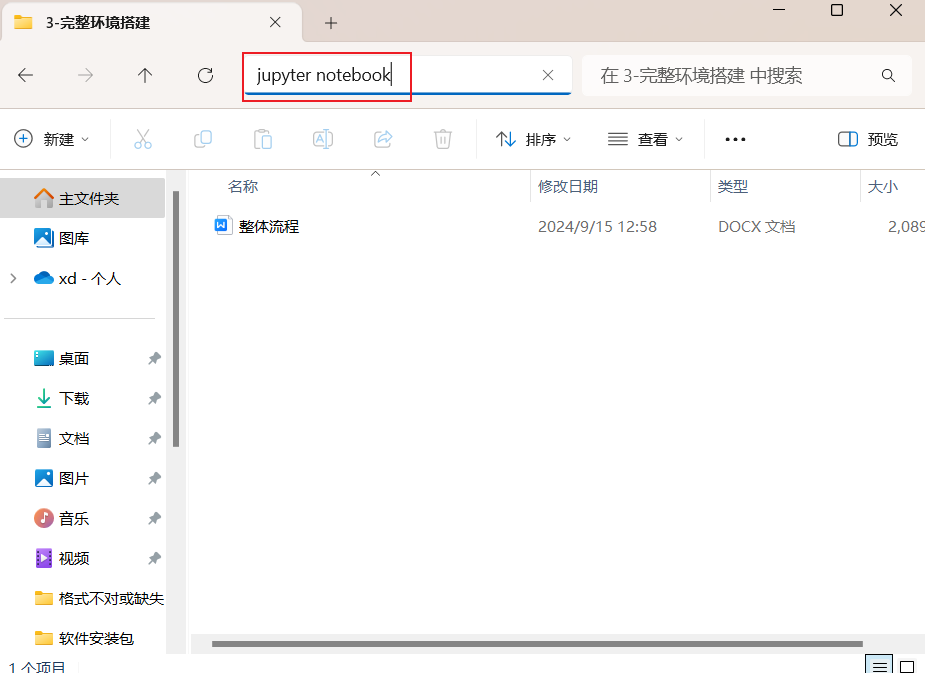


**删除虚拟环境：**

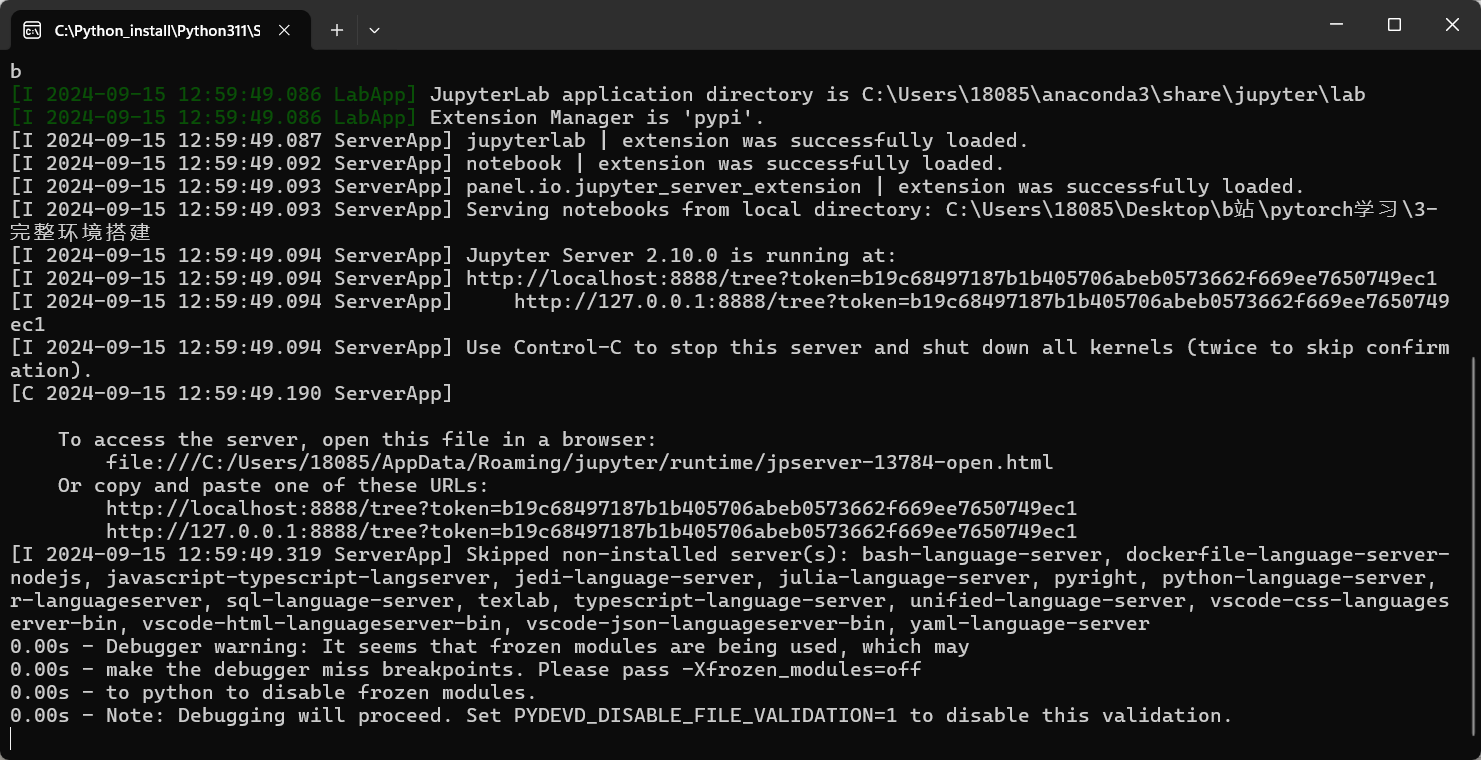
conda remove -n SDMBench --all

**1.10 虚拟环境的使用（jupyter）**

在jupyter notebook中配置虚拟环境，最简单的方式打开jupyter notebook，在任意路径输入jupyter notebook，然后按回车即可：



进入会使用你的默认浏览器进入，注意，打开的同时会打开与之对应的控制台，

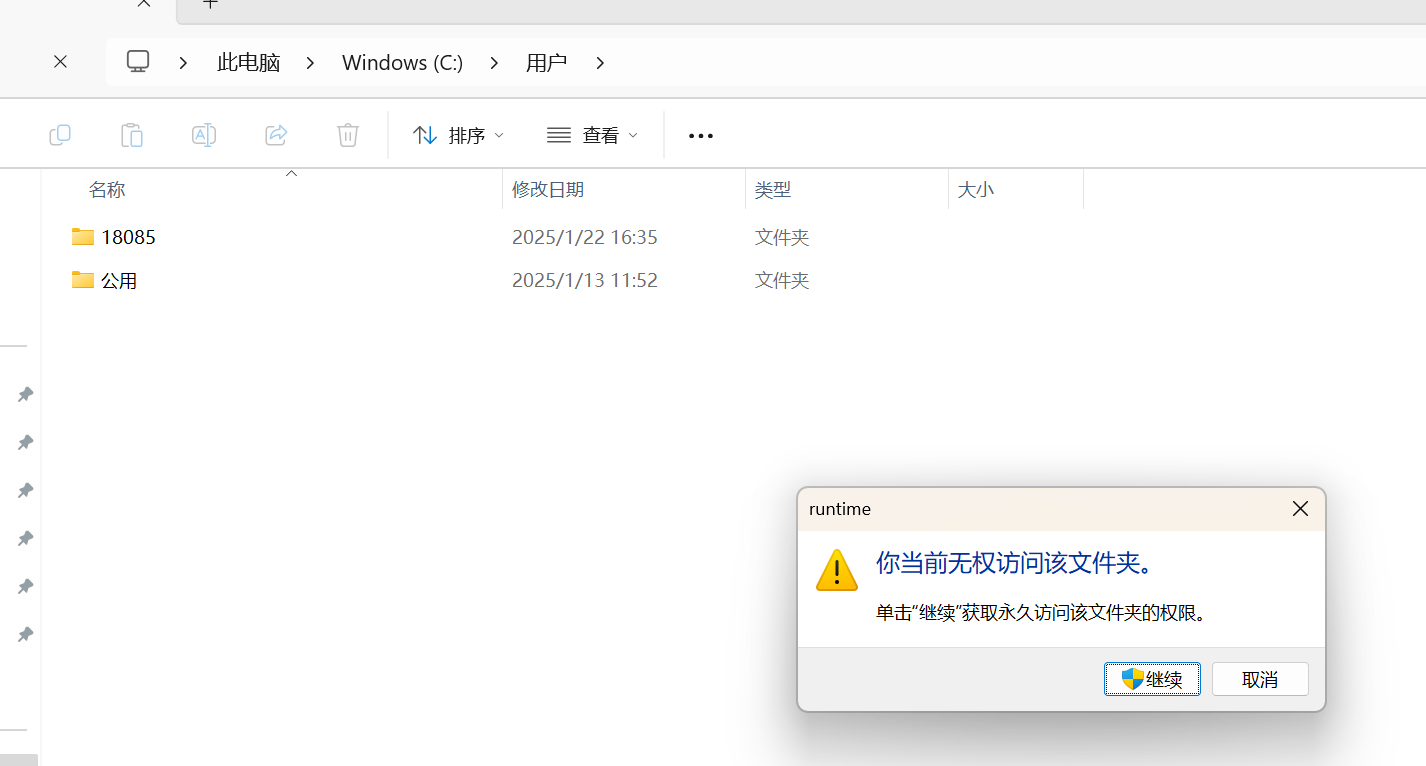


请不要关闭！！！！关闭了jupyter就无响应了。

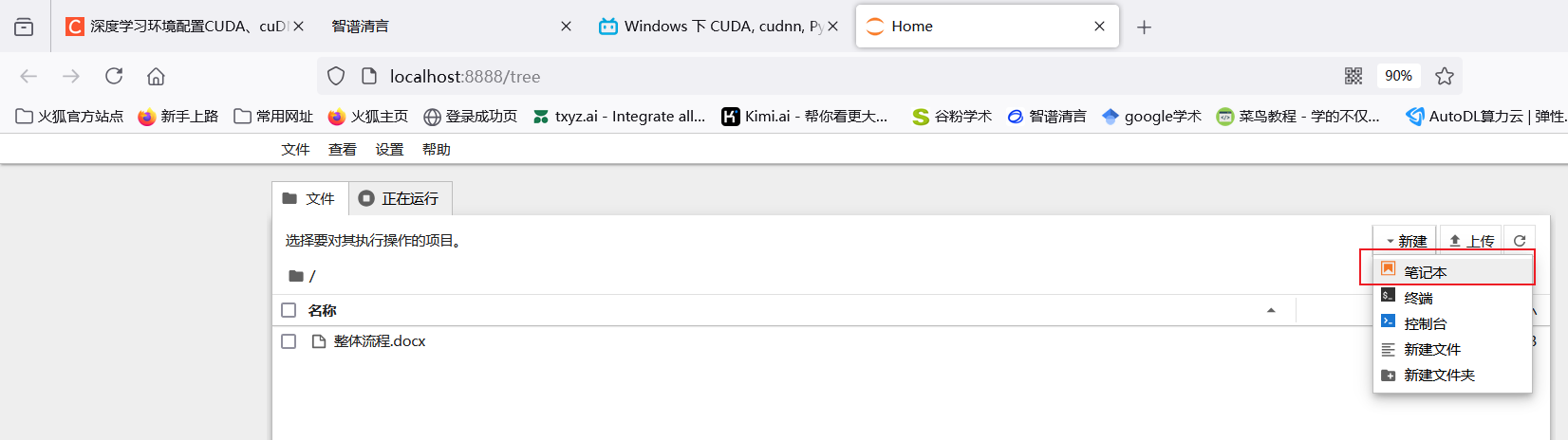
**（如果jupyter打不开，报错如下：）**



切换目录至：C:/Users/18085/AppData/Roaming/jupyter/runtime获得权限



新建选择：

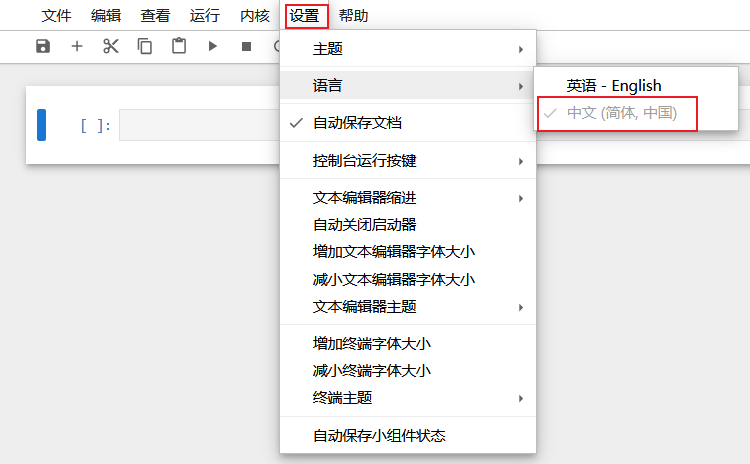


如果你想配置jupyter为中文，请在控制台输入：

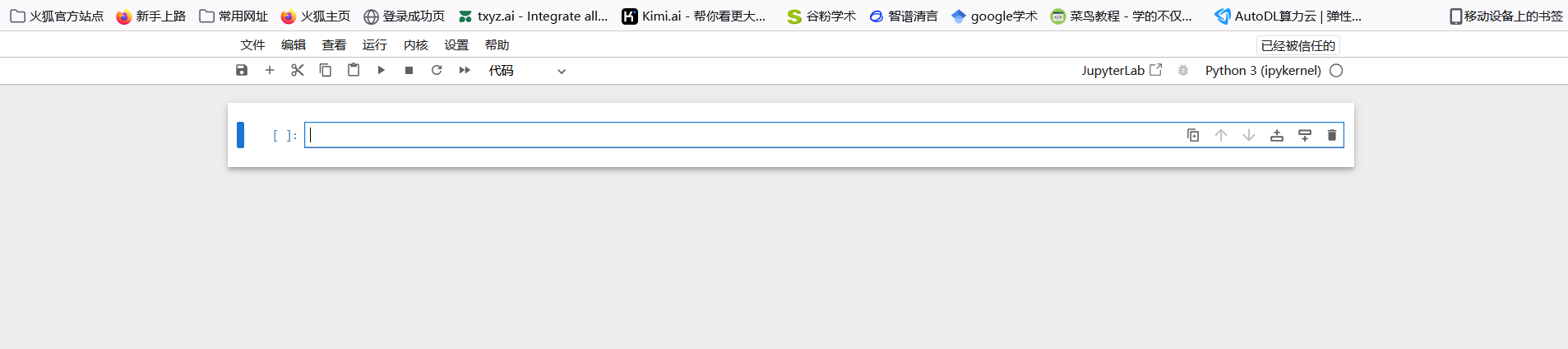
pip install jupyterlab-language-pack-zh-CN



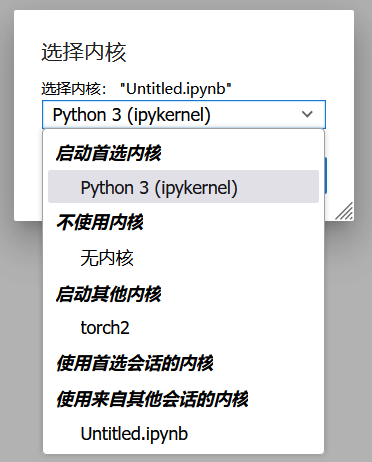
然后找到设置选择中文就好了：



这样打开的jupyter就是中文啦：



配置虚拟环境到jupyter 内核，此时我们可以发现，我们刚刚创建的虚拟环境用不了：

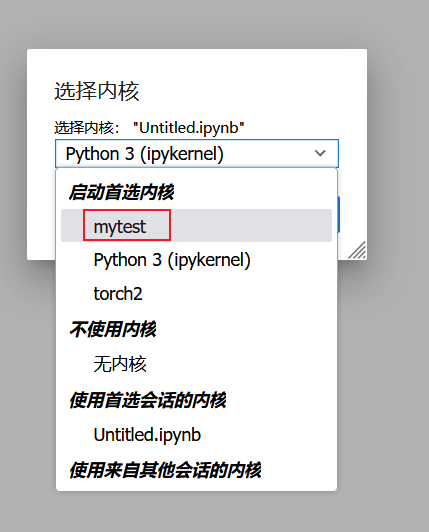


在控制台（切记是当前所选的虚拟环境）输入：

python -m ipykernel install --user --name=mytest



刷新jupyter就可以选择啦



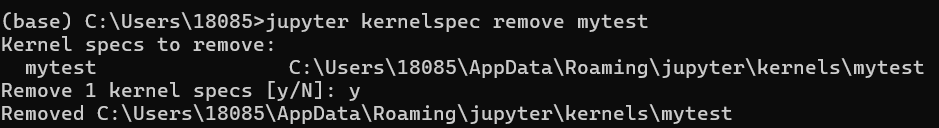
如果上述配置有报错显示没有ipykernel模块，请先**在那个虚拟环境中（不是base环境）**安装ipykernel：

conda install ipykernel

然后再添加内核就好啦。

移除内核：

jupyter kernelspec remove mytest

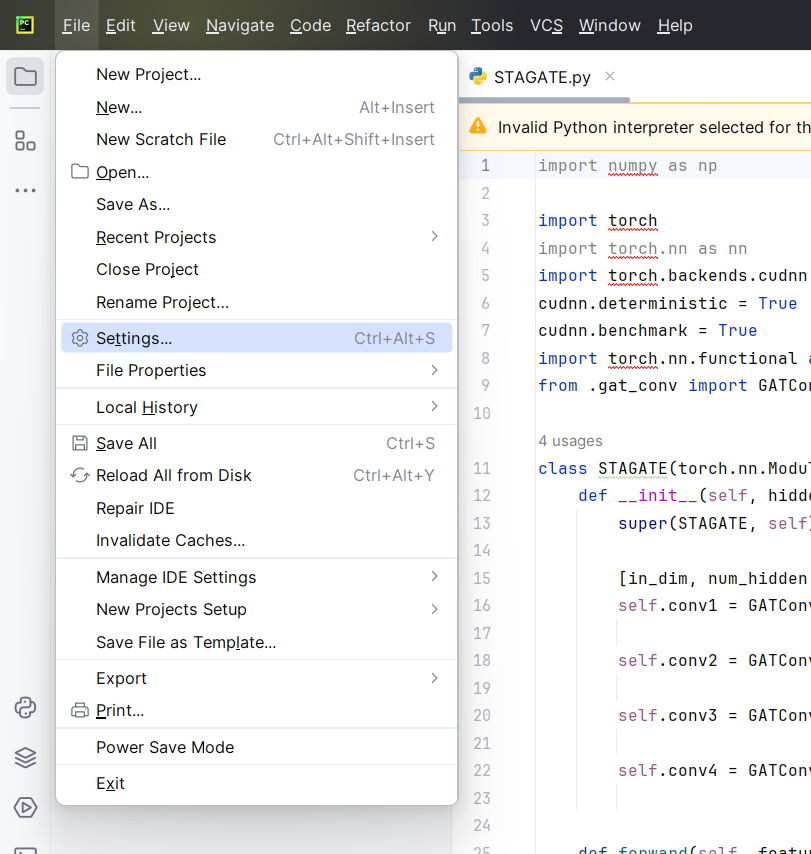


这样jupyter里面就没有这个内核了。

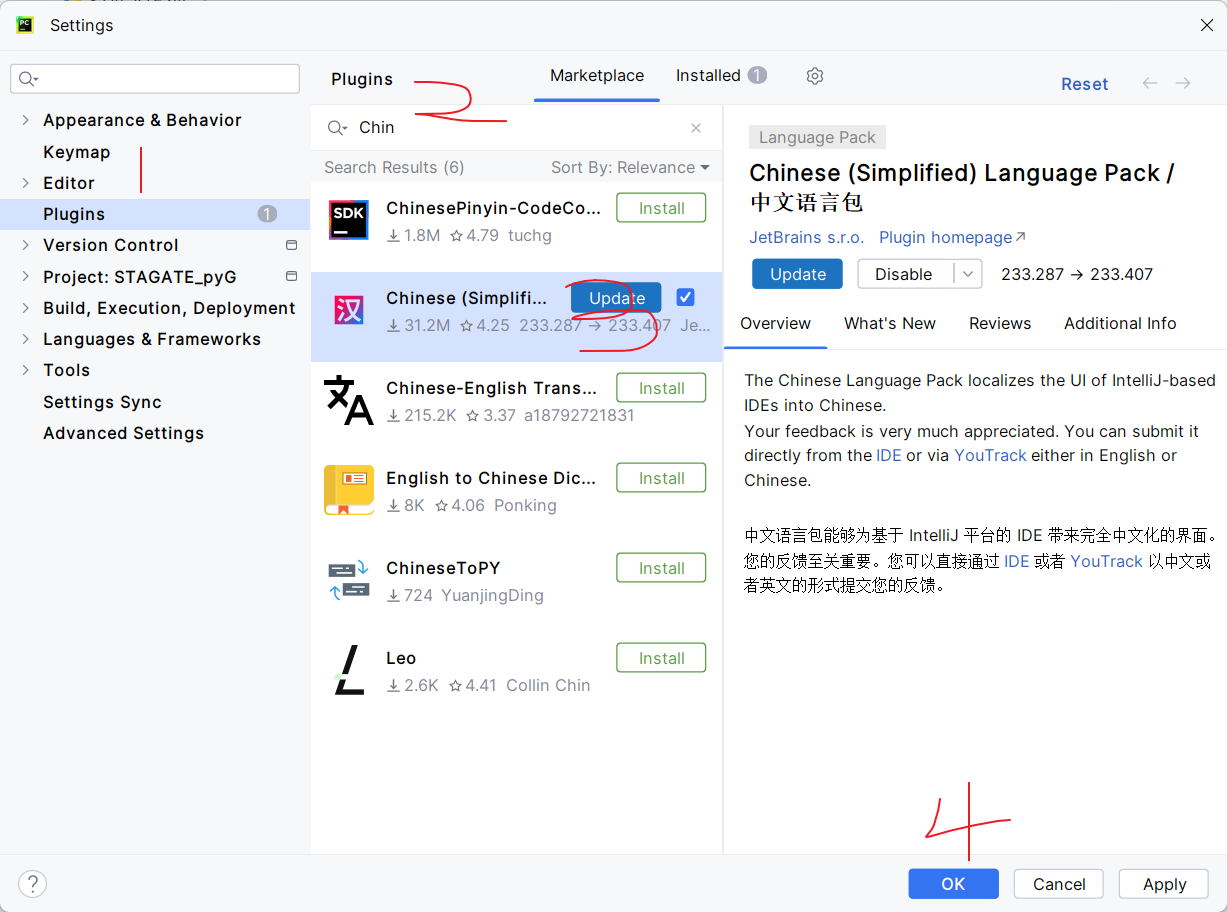


**1.11 虚拟环境的使用（pycharm）**

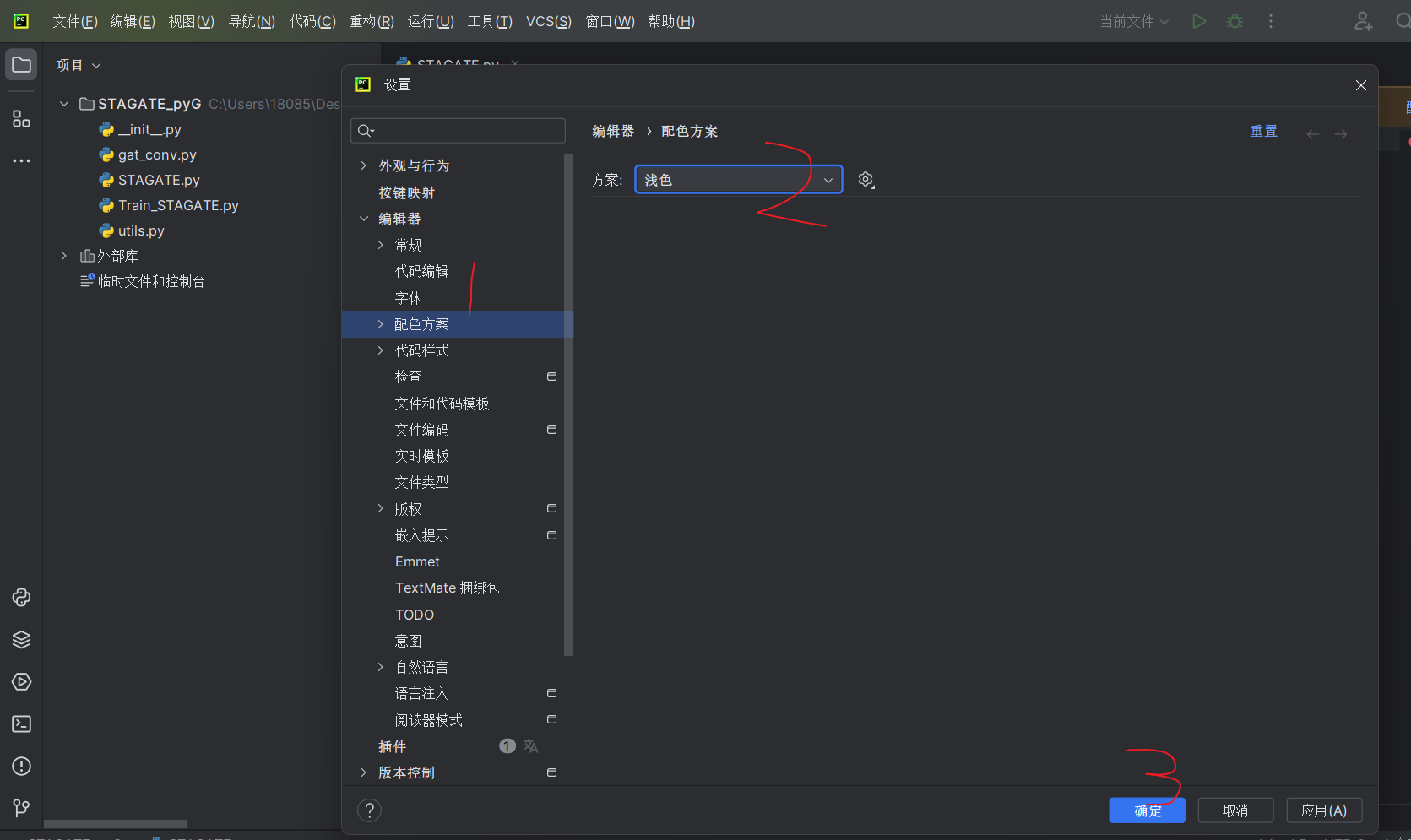
pycharm汉化，（如果有需要的话）



随后按照步骤：



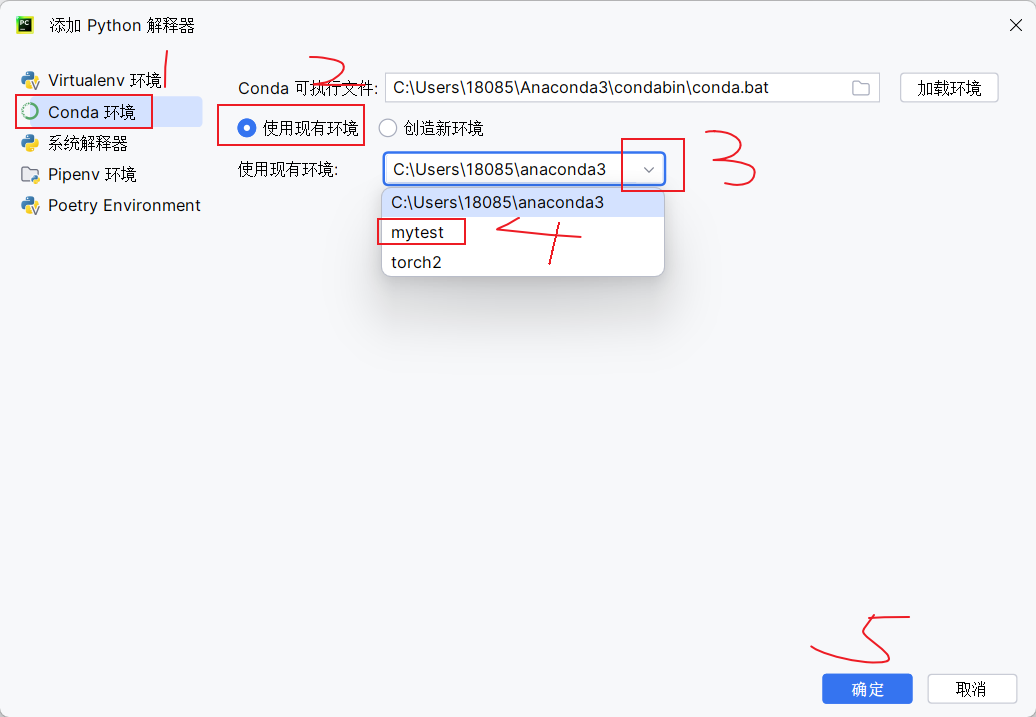
pycharm变成白底配色：



接下来配置虚拟环境，在设置中找到python解释器：



按照以下步骤选择就好啦，那么这个项目所用的解释器就是虚拟环境mytest中的python了。

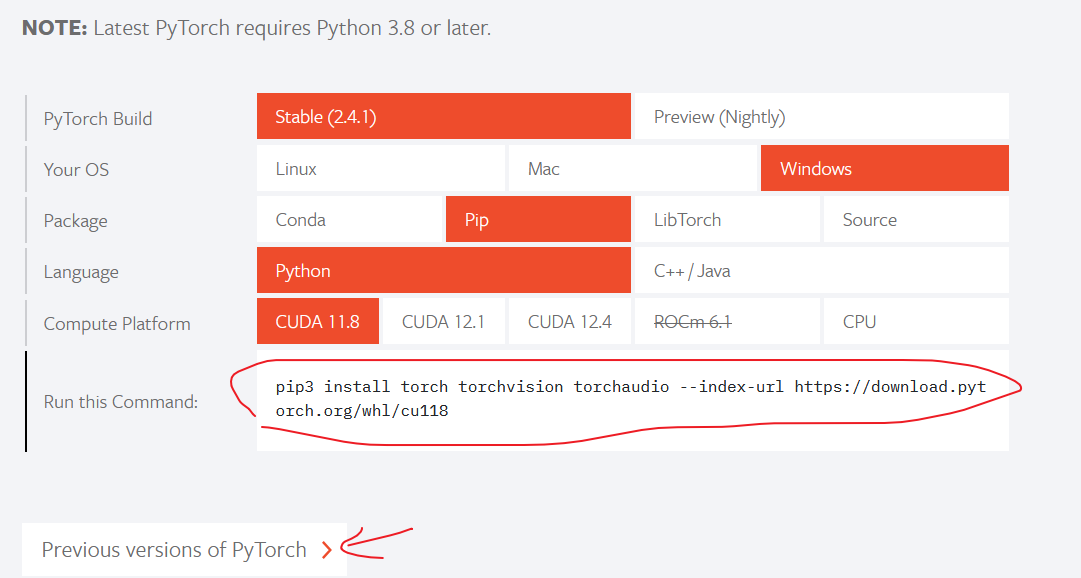


**1.12 安装pytorch框架**

进入pytorch官网获取安装指令：

<https://pytorch.org/>

下滑找到：



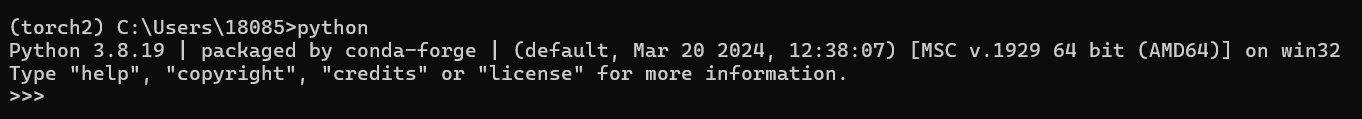
圆圈处就是你的安装命令，箭头表示更早的pytorch版本。复制到控制台，进入你想要安装的虚拟环境，安装即可：



验证是否安装成功：

首先输入

python

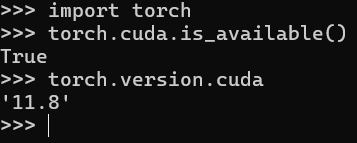


再输入下述指令就好了：

import torch

torch.cuda.is\_available()

torch.version.cuda



能正常显示就表示没问题了。

当然将上述指令也可以复制到jupyter和pycharm中，如果返回一样的结果，就说明虚拟环境已经配置pytorch成功。

**1.13 python自己包的配置命令**

cd STAGATE\_pyG-main

python setup.py build

python setup.py install

注意加上：zip\_safe = False,

**1.14 pyG安装方式**

|  |
| --- |
| pip install --no-index torch-scatter -f https://pytorch-geometric.com/whl/torch-1.9.0+cu111.html  pip install --no-index torch-sparse -f https://pytorch-geometric.com/whl/torch-1.9.0+cu111.html  pip install --no-index torch-cluster -f https://pytorch-geometric.com/whl/torch-1.9.0+cu111.html  pip install --no-index torch-spline-conv -f https://pytorch-geometric.com/whl/torch-1.9.0+cu111.html  pip install torch-geometric |

**1.15 R相关配置**

|  |
| --- |
| linux安装R命令：  conda create -n r\_env -c conda-forge r=4.2.0  Windows配置环境：  os.environ['R\_USER'] = r'C:\Users\18085\anaconda3\envs\torch3\Lib\site-packages\rpy2'  os.environ['R\_HOME'] = 'C:\Program Files\R\R-4.3.3' |

**1.16 项目路径设置**

|  |
| --- |
| # 查看当前工作目录 import os  print(os.getcwd())  # 添加  import sys  sys.path.append(r'xxxx')  # 查看所有目录路径  import sys  for i in range(0,len(sys.path)):  print(sys.path[i])  # 删除  sys.path.remove(r’xxx’)  **# 查看python解释器的命令**  import sys  print(sys.executable)  which python  **# 查看jupyter 内核**  !jupyter kernelspec list |

**1.17 linux命令总结**

|  |
| --- |
| nvidia-smi  ls -la 【显示隐藏文件】  cat setup.py 【查看文件】  cd -【返回上一层路径并返回】  cd ..  cd / cd~ 回到开始路径  【快速进入相关路径】  find . -name setup.py  cd ./some\_directory  切换快捷命令：cd "$(dirname "$(find . -name setup.py | head -n 1)")"  du -sh \*【查看用户内存使用情况】  df -h【查看硬盘内存情况】  ctrl + u 【清空命令行】  rm -rf /home/user/test 【删除文件夹】 |

|  |
| --- |
| 修改配置文件 |
| vim ~/.bashrc  输入命令后 保存并退出编辑器（在 vim 中按 Esc，然后输入 :wq 回车）  source ~/.bashrc配置文件让文件生效 |

**1.18 Python→requirements文件**

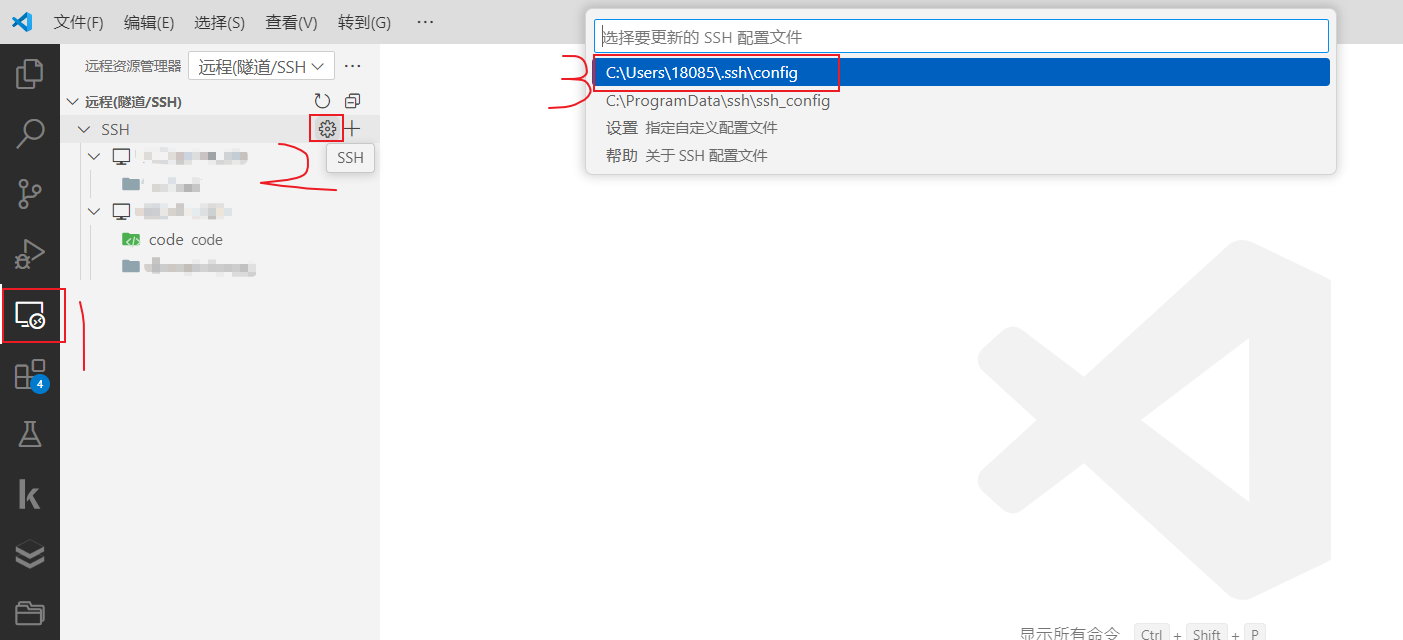
|  |
| --- |
| pip freeze > C:\Users\18085\Desktop\requirements.txt  pip install -r requirements.txt |

**1.19 VS连接远程服务器**

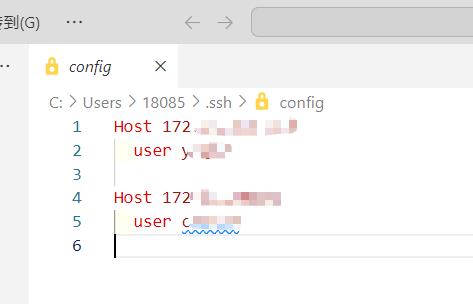
首先假设你已经下好VS code了，安装以下拓展（可以把和Remote的都装了哈哈哈，开始我虚拟环境一直过不来，原来是要装第二个拓展。）：



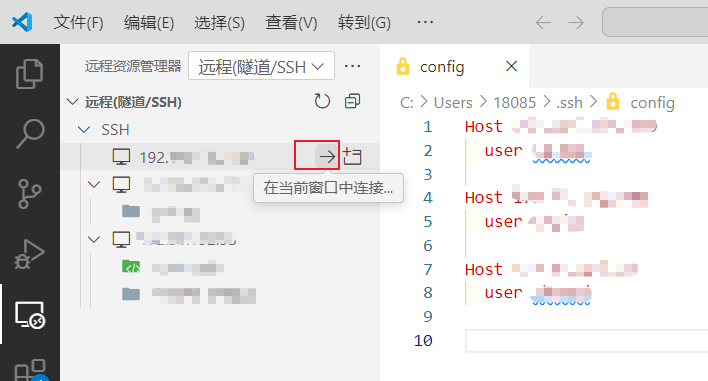
然后依次点击



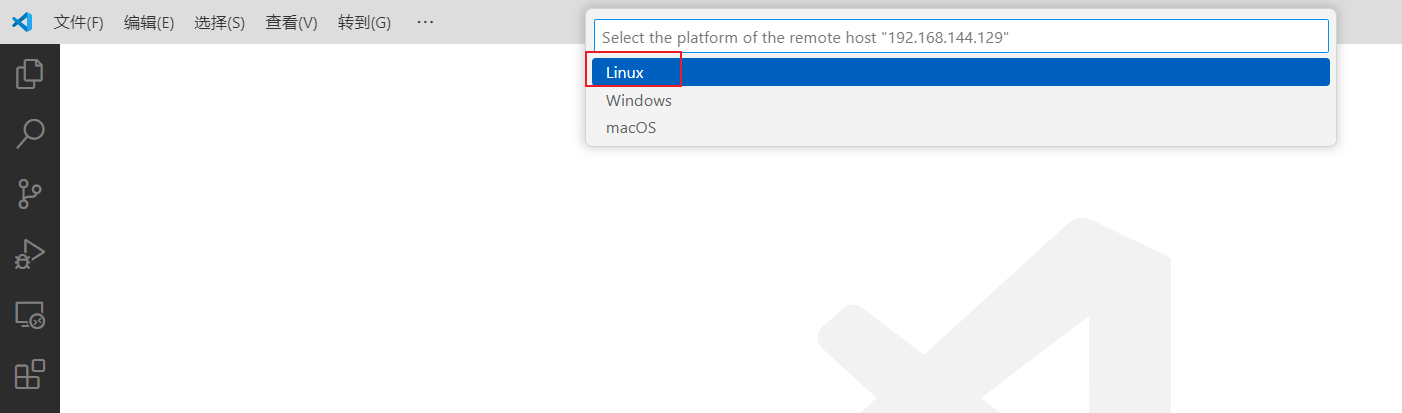
就可以打开配置文件，然后写上IP地址和用户名，其他的可以删掉。你有多个服务器就可以往下写就行：



填写完左边就会出来连接的箭头了：

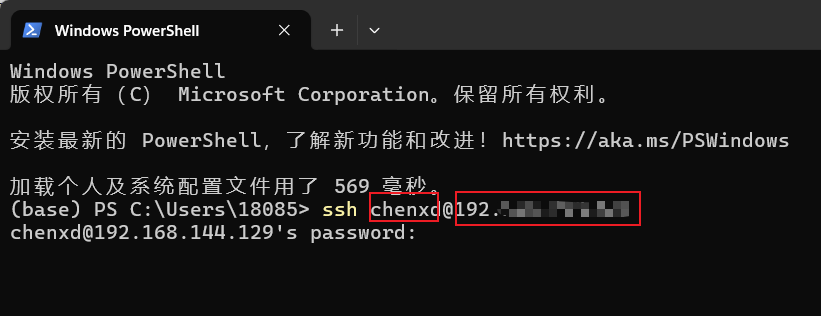


然后选择你的服务器类型，一般都是linux哈：



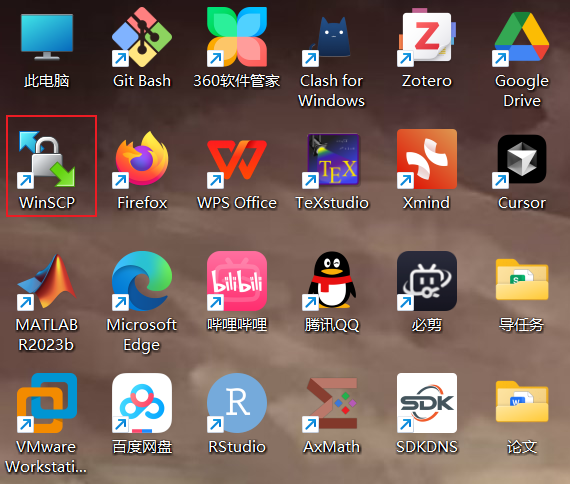
然后输入密码就好啦！

本地电脑终端连接服务器方法，ssh 用户名@IP地址，然后输入密码（不会显示哈）就好了：

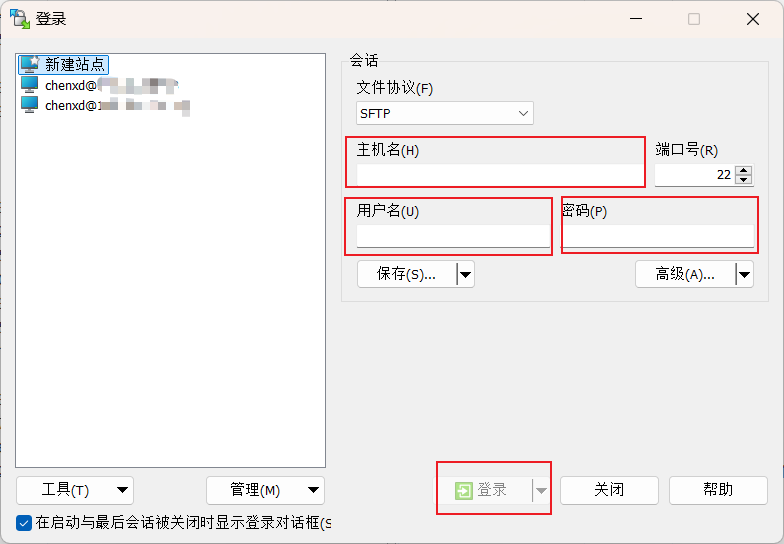




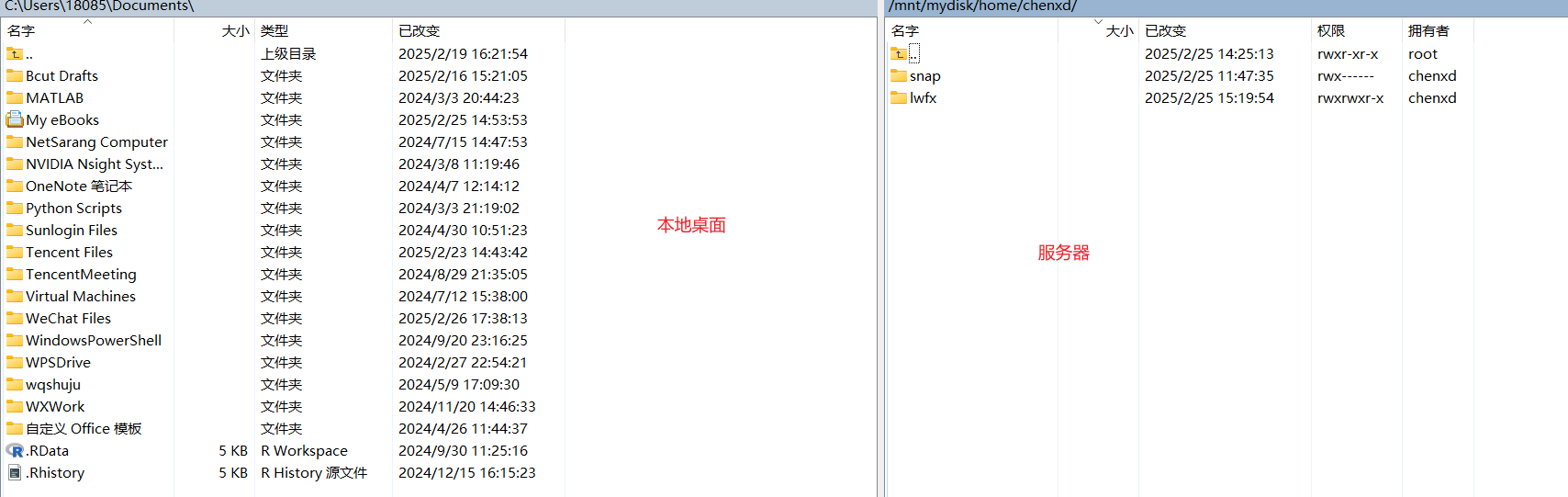
如果你用不惯终端，那么下面这个软件可以帮你（网页直接下载即可）



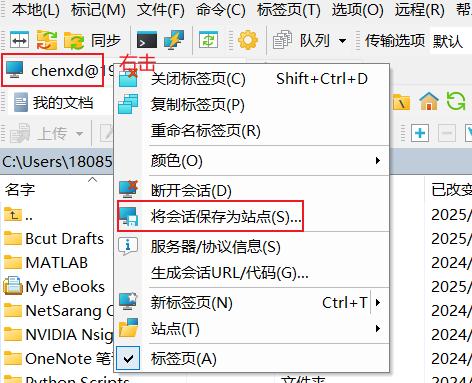
填上相关信息就好啦：



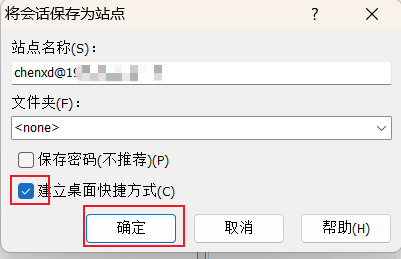
你就可以直接通过“拖拉拽”将本地和服务器进行文件互通啦：



当然我们也可以保存为桌面快捷方式：



点击

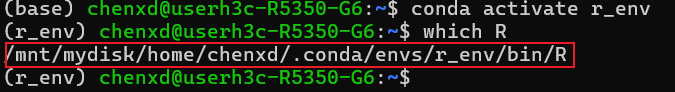


以后直接点这个就可以啦！

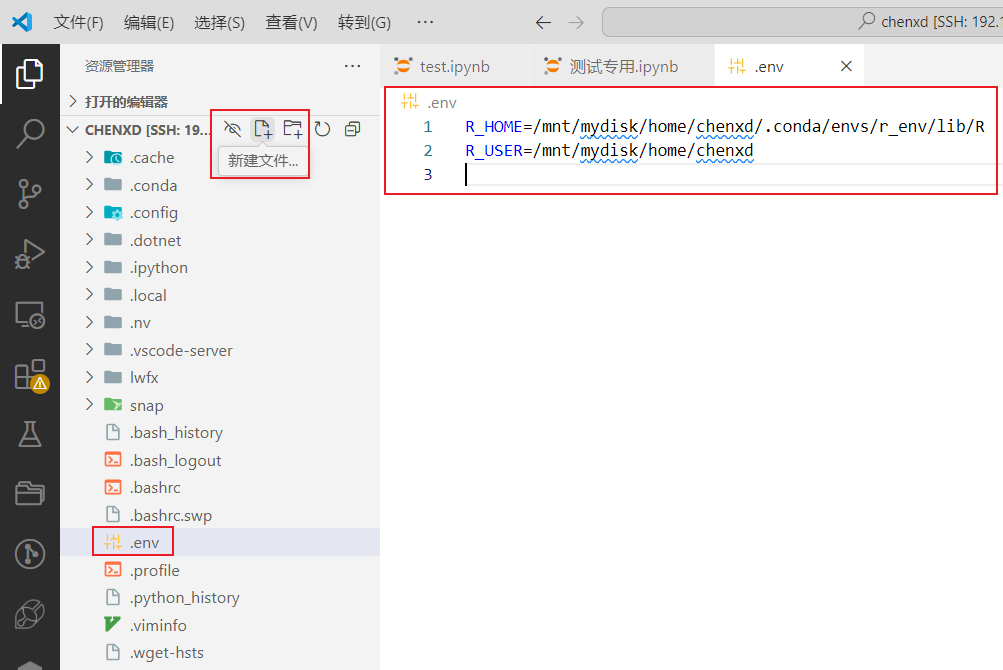
**1.20 服务器端配置rpy2模块**

有时候我们需要在Python的虚拟环境中调用R的模块。不会还有人和我一样去真的单独装R吧哈哈哈，其实利用conda创建一个虚拟环境就好啦！连接好服务器后，在终端输入（r\_env是虚拟环境的名字，可以改）：

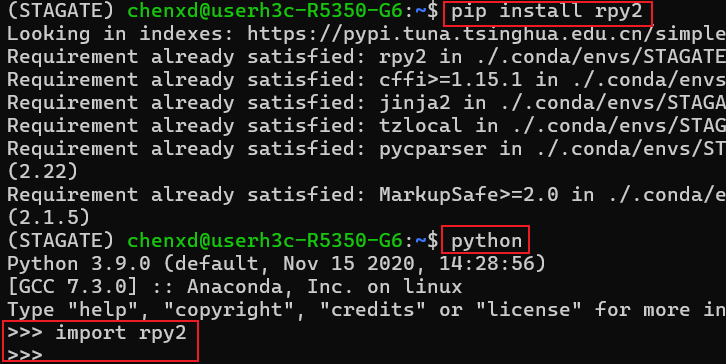
conda create -n r\_env r=4.3

然后我们进入r的虚拟环境，查看r的位置：  


接下来配置r语言环境变量，新建一个.env的文件，配置如下命令：



好啦，我们在Python环境中测试一下是否可以安装并导入rpy2：



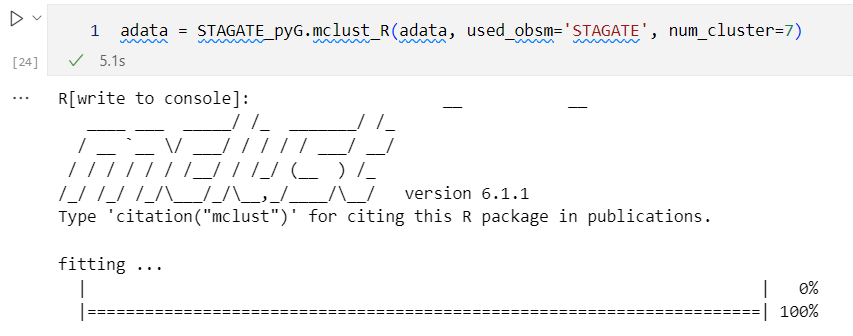
如果不报错就说明没问题了。下一步，我们需要在R语言环境中去安装我们要用到的包，这里以mclust为例（下面步骤我就不截图了）：

step1：切换到r的虚拟环境：conda activate r\_env；

step2：输入‘R’进入r编译器；

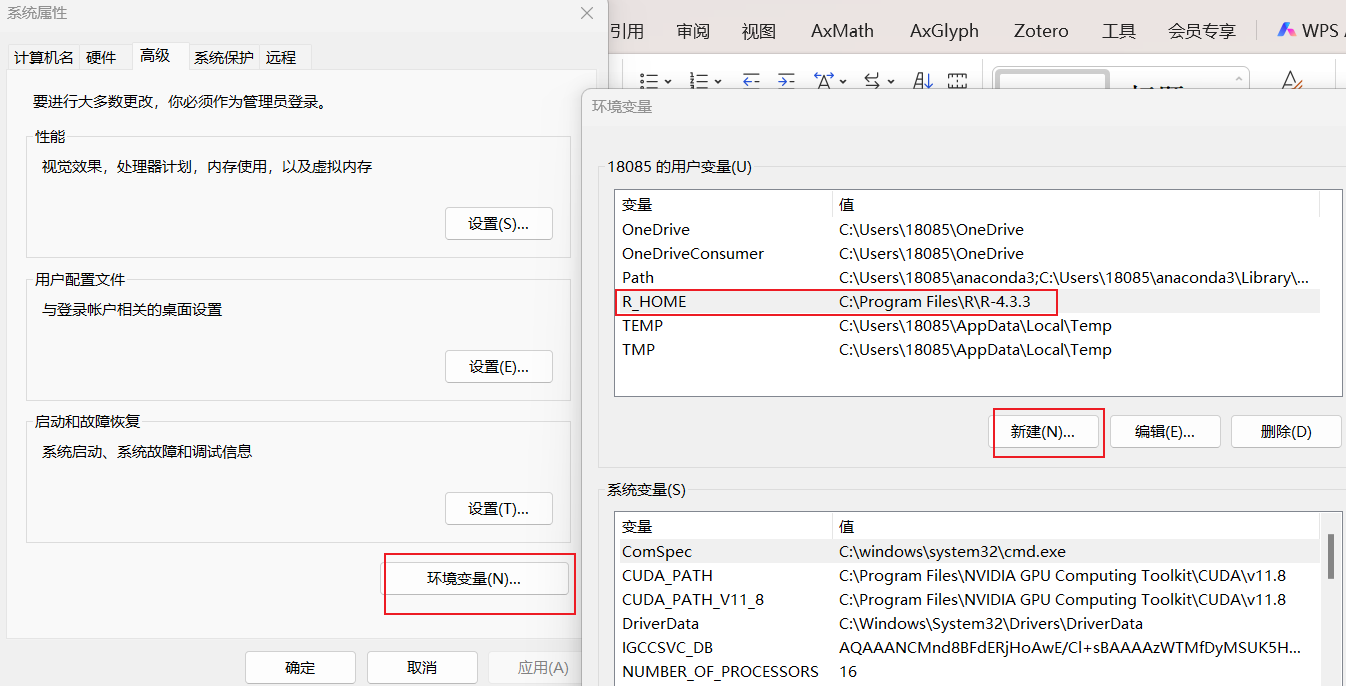
step3：安装需要的包：install.packages("mclust")，然后选择来源就好了（比如中国深圳就是23）；

最后来验证一下mclust是否可用：



可以发现已经没问题了！

Windows端一定要新建一个R\_HOME对应到你的R哈：



然后再代码运行前设置这两个参数就好啦！（找到自己电脑对应的位置）

os.environ['R\_USER']=

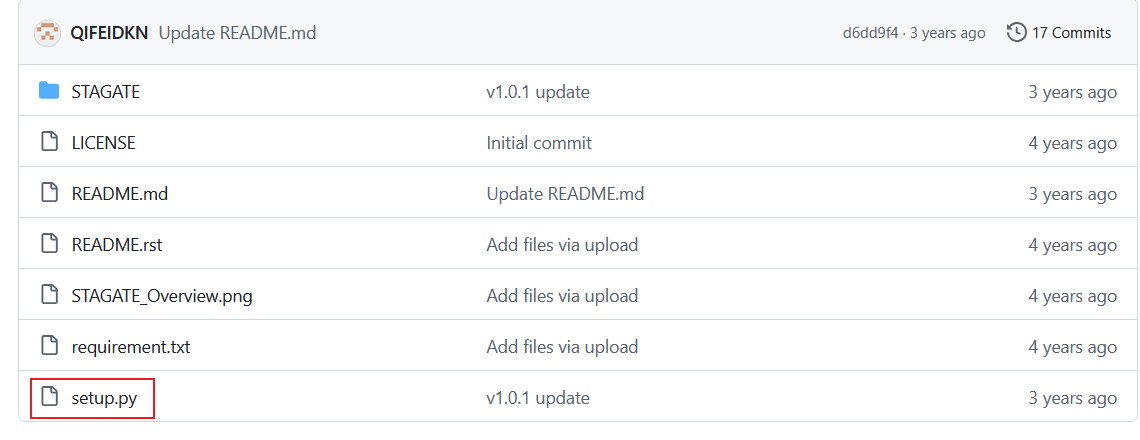
r'C:\Users\18085\anaconda3\envs\torch3\Lib\site-packages\rpy2'

os.environ['R\_HOME'] = 'C:\Program Files\R\R-4.3.3'

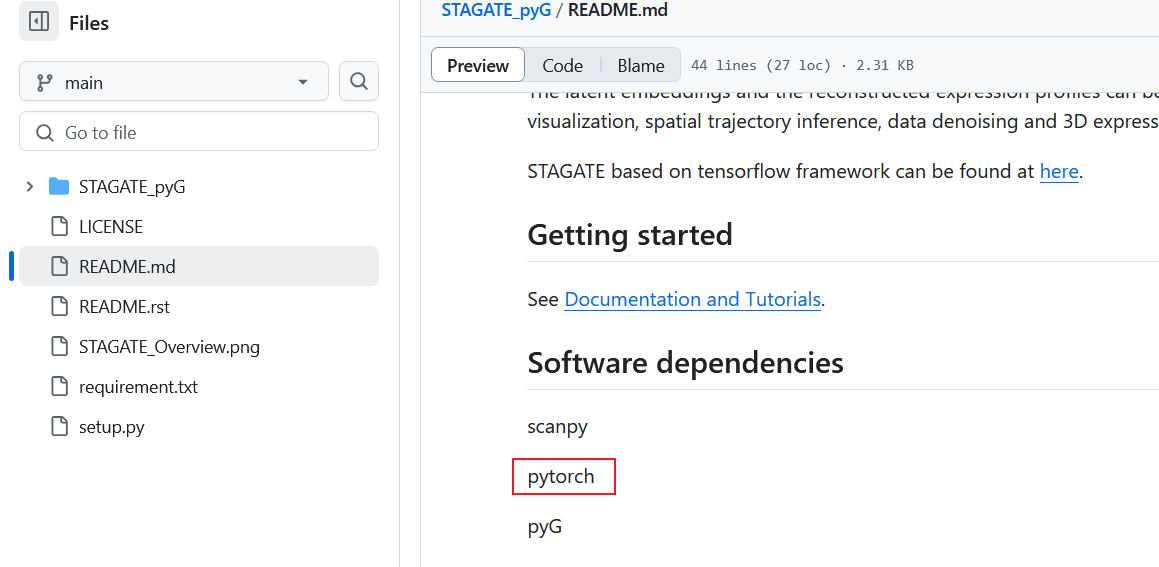
**1.21 github项目包配置（setup.py方法）**

github项目通常有好几种方式配置，我用过的有直接pip安装（这种比较少）、通过yml或者yaml或者requiremen.txt进行安装，或者放置在同一项目直接import，或者使用Docker容器，最好用的还是今天分享的setup.py方法。

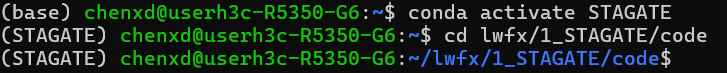
首先打开github项目，查看其是否有这个文件，如果有就可以用这种方式进行配置：



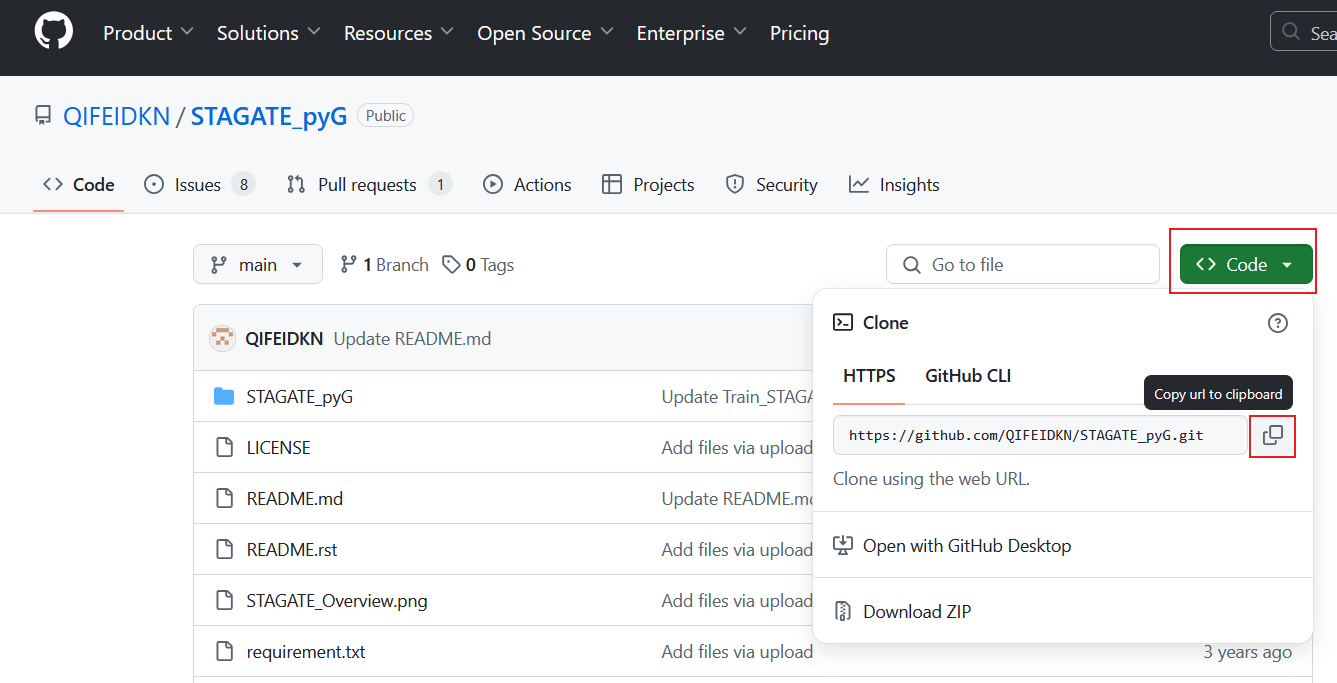
然后我们打开readme文件，看这个项目使用的框架，可以看到这个项目是pytorch框架：

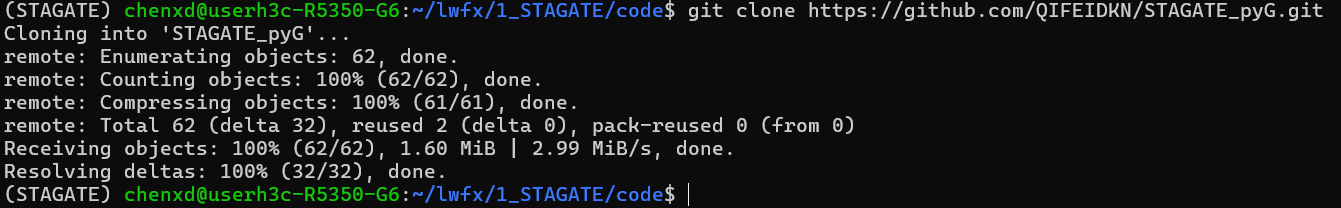


**step1：**现在进入你想要使用的虚拟环境和路径：

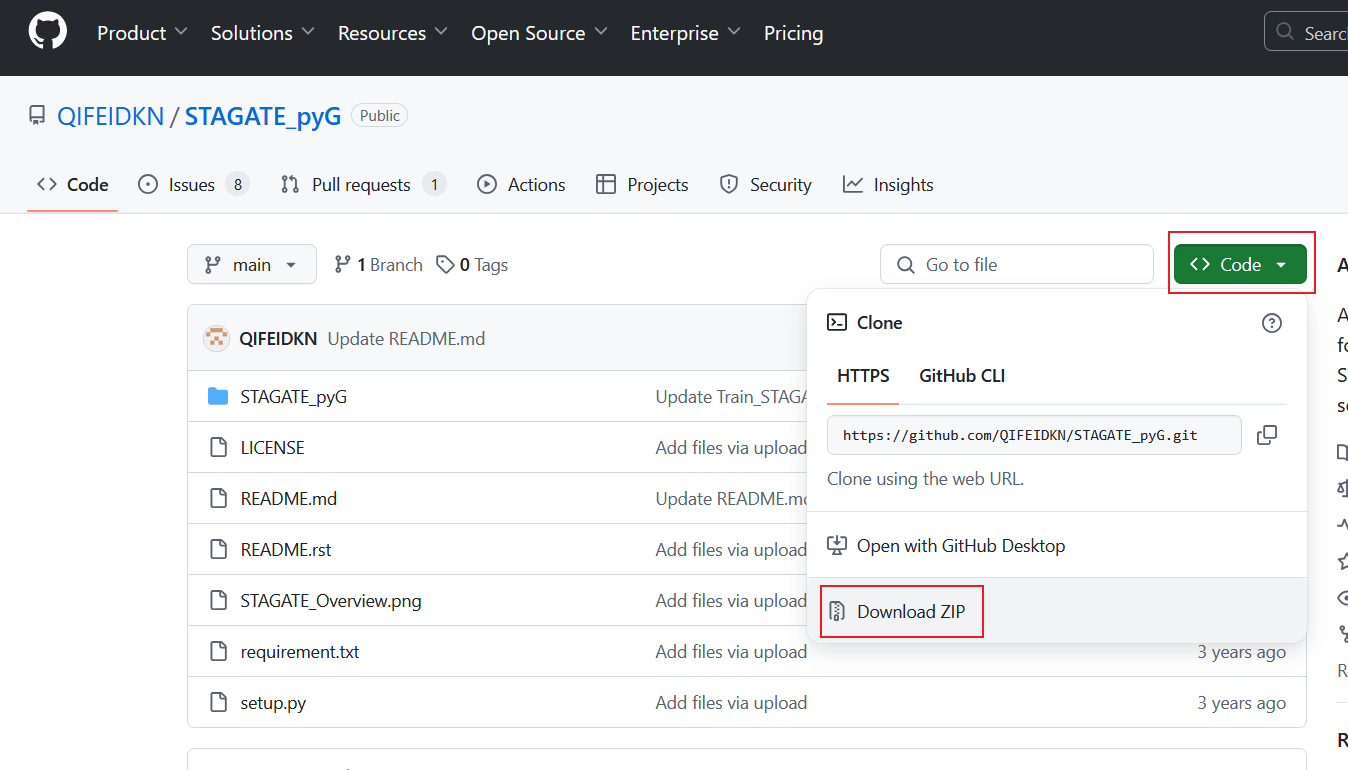


**step2：**然后我们就要获取github项目包，我们可以直接使用git clone命令：

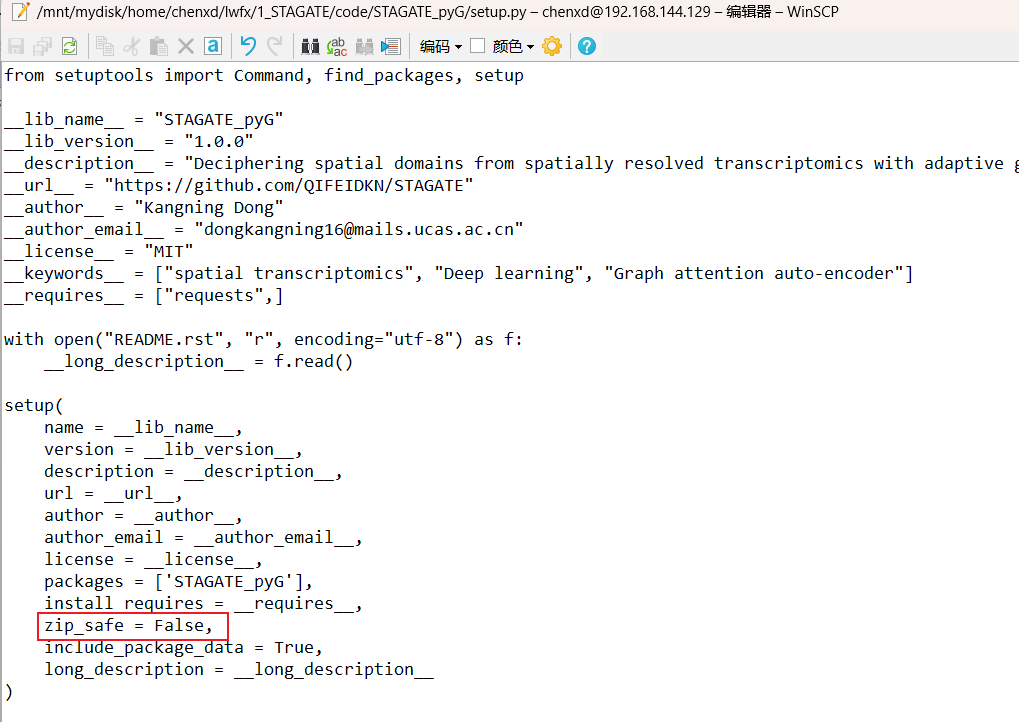




**step2（备选）：**如果上面不成功，我们也可以直接下载压缩包后解压，然后在通过vscode或者WinSCP转移到服务器中：



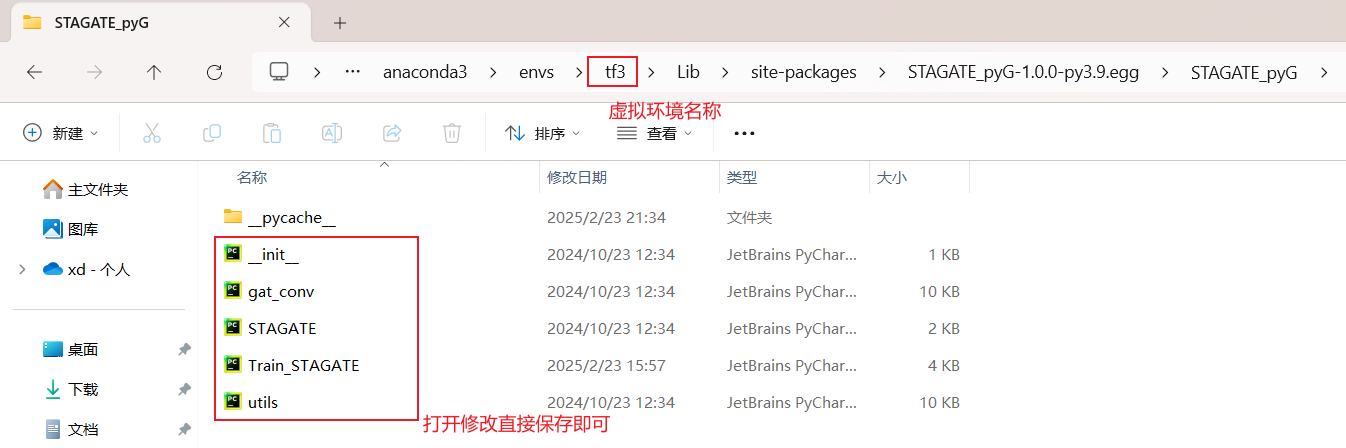
**step3：**打开setup.py，一定要确定有zip\_safe = False，我们难免有修改别人代码的情况，这个指令能让我们轻松修改源码。



**step3（续）：如何找到配置的软件包并修改**

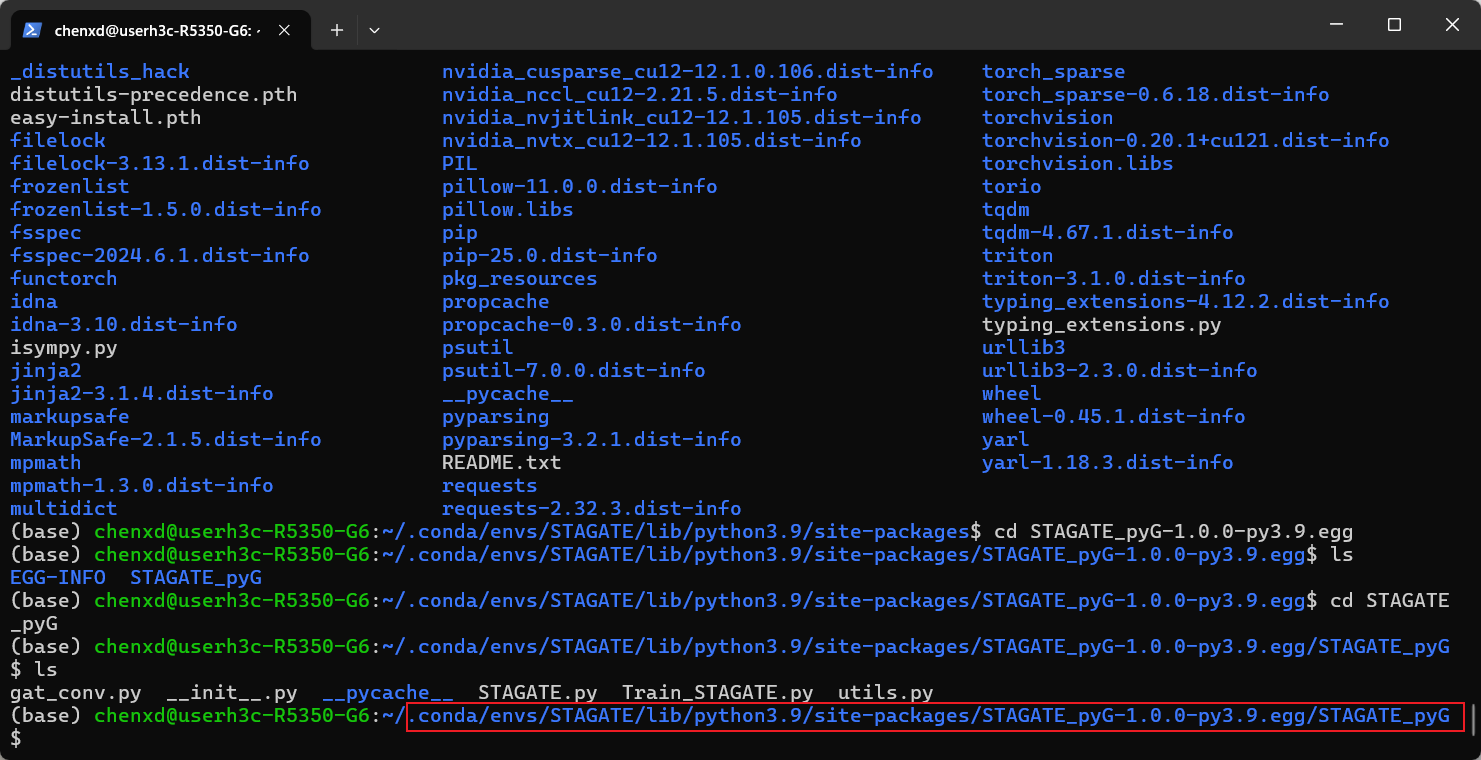
Windows一般源码路径：

C:\Users\18085\anaconda3\envs\tf3\Lib\site-packages\STAGATE\_pyG-1.0.0-py3.9.egg\STAGATE\_pyG

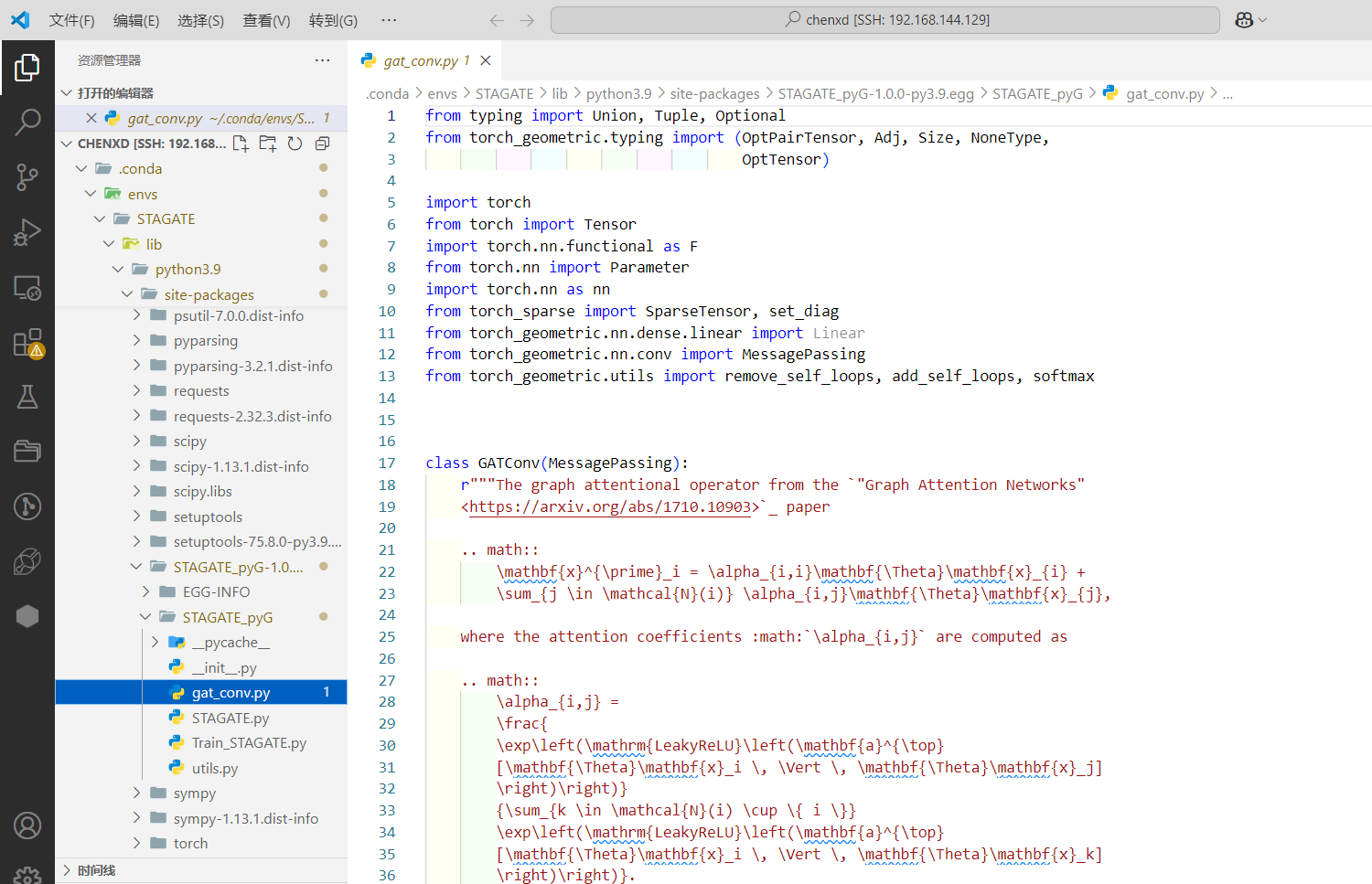


linux一般源码路径：

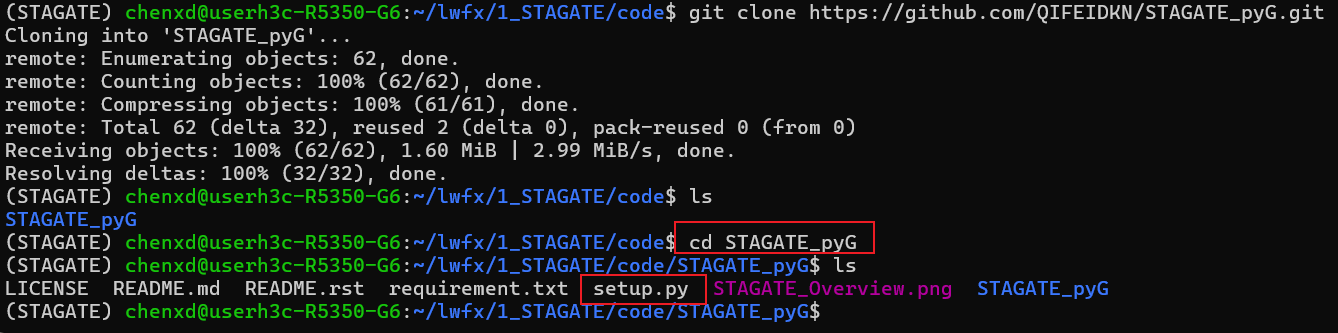
.conda/envs/<环境名>/lib/python<版本号>/site-packages/



这样我们就可以通过vs code编辑源文件了。



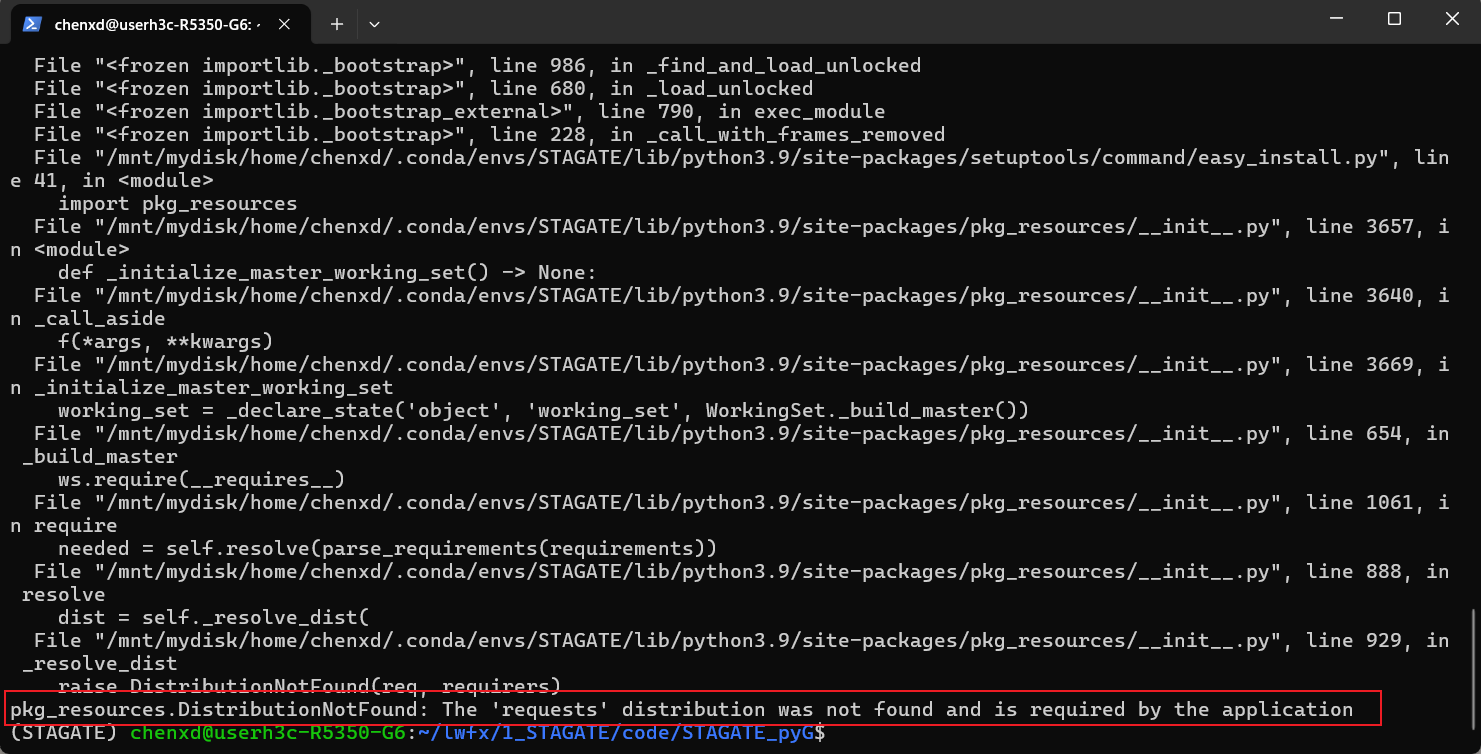
**step4：**cd 到setup.py的路径：



**step5：输入配置指令**

首先安装requests以防下方报错：

pip install requests



然后依次输入：

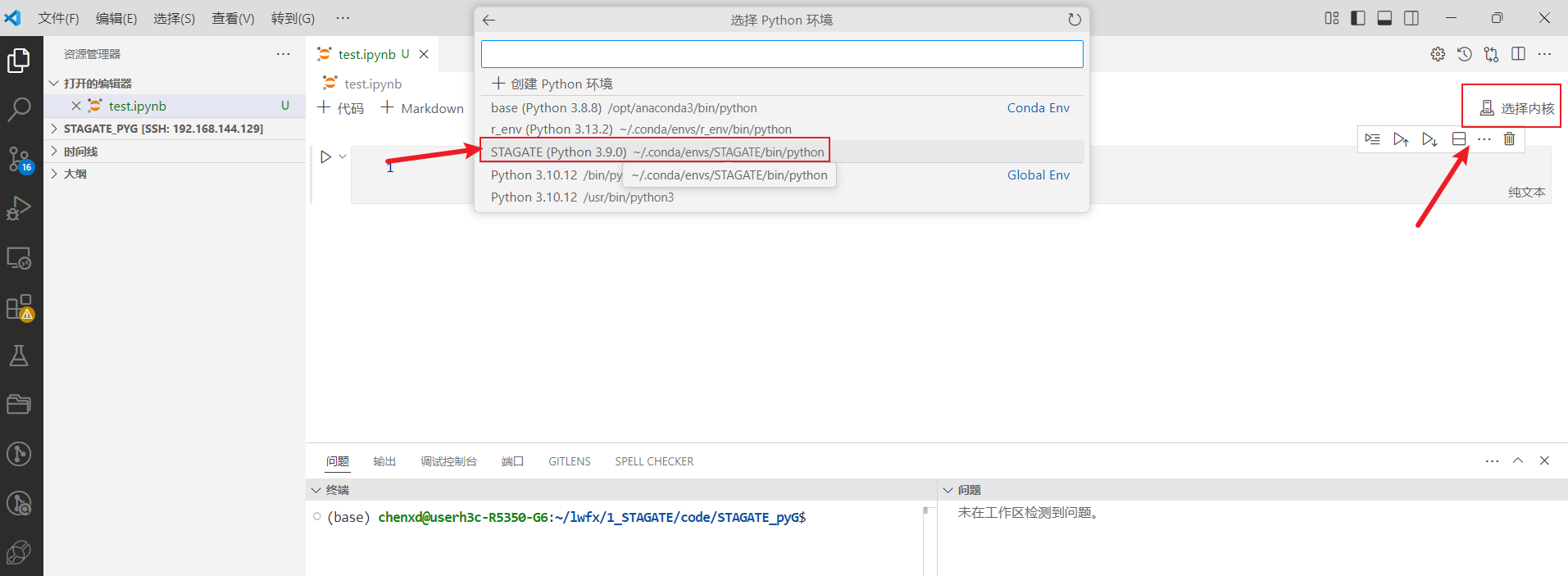
python setup.py build

python setup.py install

最后我们可以测试一下是否配置成功

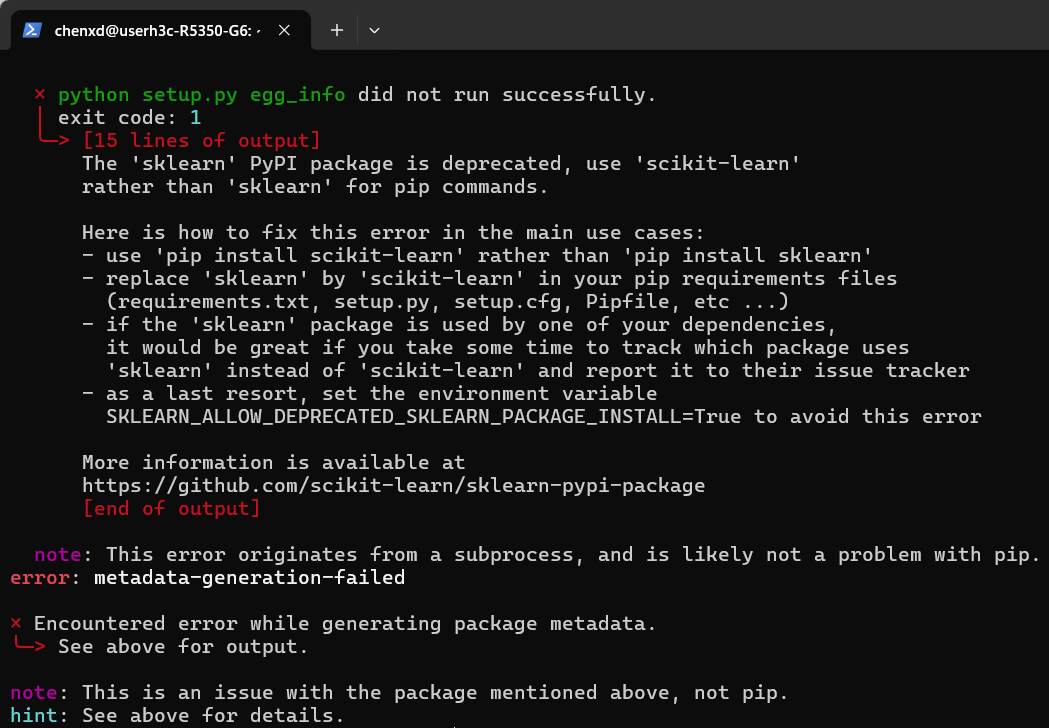


可以看到已经成功配置，上面那个报错只是因为当前虚拟环境缺少这个模块，直接pip 安装就好啦！好了，接下来你就可以在vs code中使用这个虚拟环境去运行项目代码了。



**1.22 虚拟环境调试**

当我们在安装模块的时候出现下列这种报错“xxxx，not pip”（**这个报错的原因是因为包的名字写错了！**），一般都是虚拟环境中各个模块之间版本不匹配导致装包不成功：



容易版本不匹配的包：Scanpy、stlearn（这个包要注意，我们通过pip安装的是scikit-learn，import的是sklearn）、numpy、numba等；那么应该怎么去安装版本匹配的版本呢？**直接在一起pip即可**，pip会自动帮助我们筛选各个相适应的版本：

pip install Scanpy scikit-learn numpy numba

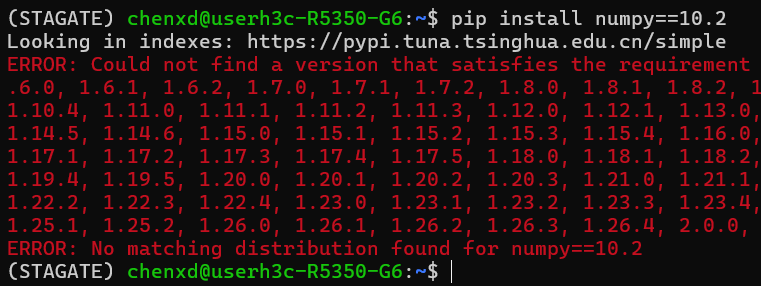
如果还是不行，可以先卸载pip uninstall 那些“钉子包”，然后再装。

如果需要更改某一个模块的版本，我们可以利用下列命令快速查找模块版本

pip install numpy ==

上列命令有些服务器终端不支持，可以随便写一个版本，如下：

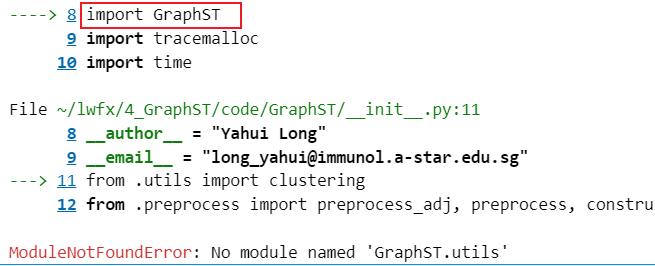
pip install numpy==10.2



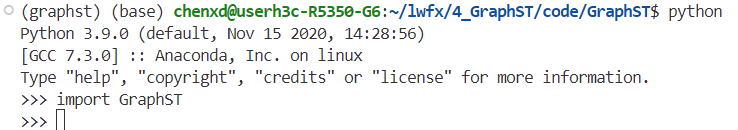
这样我们就可以根据对应包的版本一个个尝试了。

**1.23 已经安装了某个包import 还是报错解决方案**

问题描述，在vs code import报错：

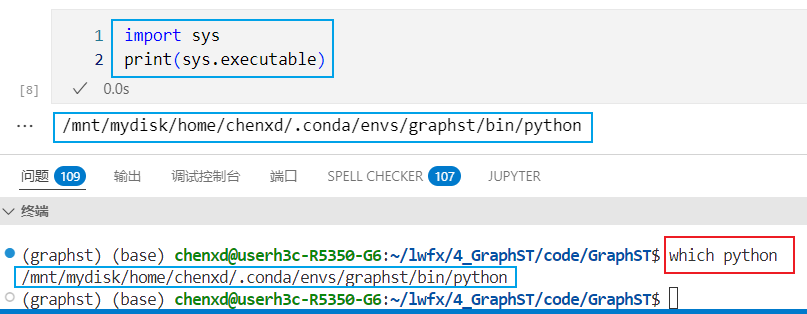


但是在终端却没有问题：



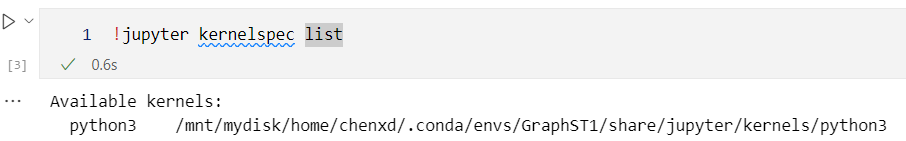
以下提供几种思路，大家自己对应去检查：

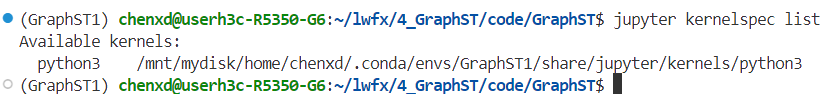
1.首先确保适用相同的虚拟环境，这个没啥好说的；

2.确保python解释器是一致的：  


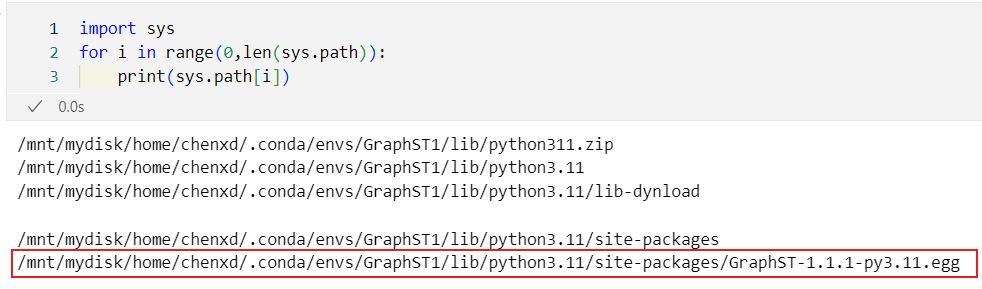
vs code是全自动的，所以不用太担心。如果是本地jupyter发生这种情况，需要切换虚拟环境内核，步骤有点繁琐，详情见b站视频“**已经装了某个包，jupyter notebook还是说No module named 'xx'解决方法**”；

3.查看jupyter 内核是否和终端一致：





4.将模块所在位置添加到工作目录中sys.path.append：

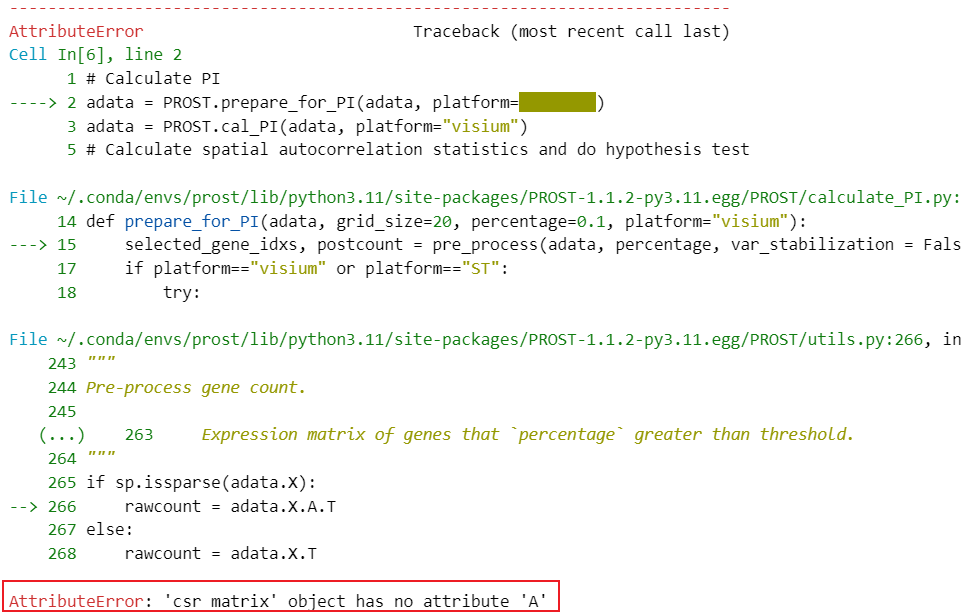


5.其他冲突（大家一般用不上）

根目录名包含了一个 \_\_init\_\_.py 文件，这个真的把我坑死了。虽然 Python 默认不会将带有非法字符（如连字符 -）的目录识别为包，但某些编辑器（如 VS Code、PyCharm）或开发工具会尝试将项目根目录隐式添加到 sys.path，导致 Python 试图将其作为包导入，从而引发混乱。

终端正常但编辑器报错：本质是编辑器隐式添加了项目根目录到 sys.path，而根目录的 \_\_init\_\_.py 和非法名称导致路径冲突。

**1.24 去修改github源码解决报错**

今天遇到了这样一个问题，明明几个月前都能运行的github包，但是今天来运行都报错：  




把错误复制到豆包，给的反馈是因为版本更新了，一些方法不再支持了：



以下提供两种解决方法，第一种就是把报错的那个软件包安装旧版本的，这种是非常不建议的，我们可以pip show一下，可以看到其牵连的包非常多，往往修改一个就要修改多个。



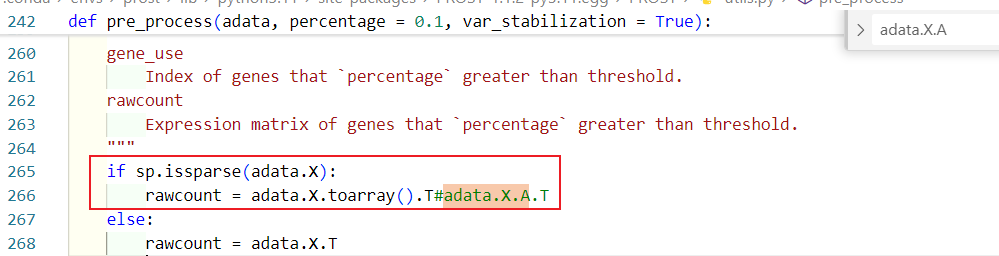
那么接下来，我们就采纳豆包的建议，去源码里面修改。首先找到源码所在位置，一般linux的位置在：

用户名/.conda/envs/虚拟环境名/lib/python3.XX/site-packages

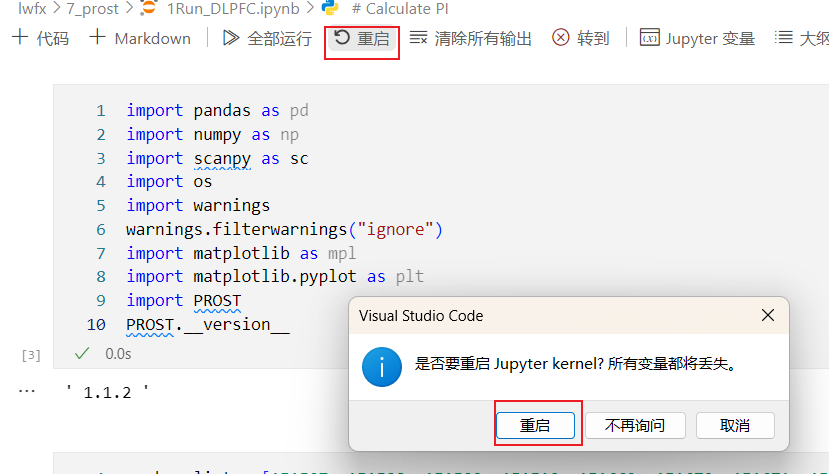
找到我们在github安装的包：



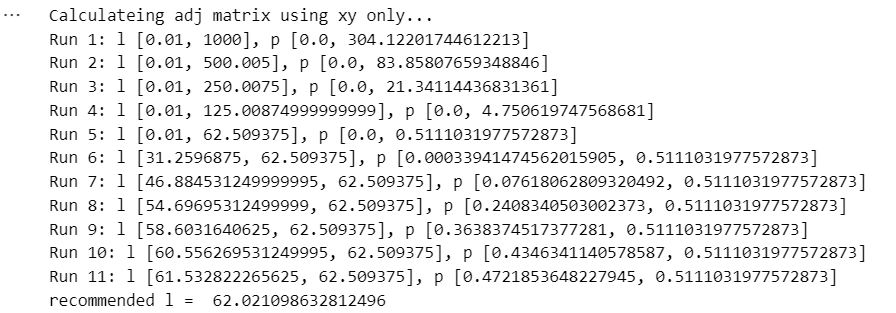
打开功效函数，进行修改把adata.X.A.T都替换成 rawcount = adata.X.toarray().T



我这边建议的是将源码进行注释修改，而不是直接删除，因为有时候我们一次修改不成功，可能还需要回来重新修改。每次更新源码后，我们都需要重启内核才能生效：

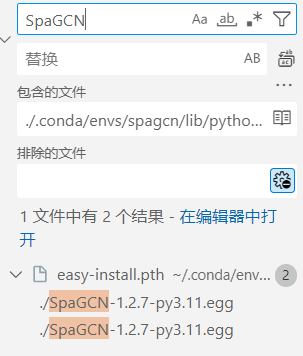


现在再次重新运行之前报错的代码：

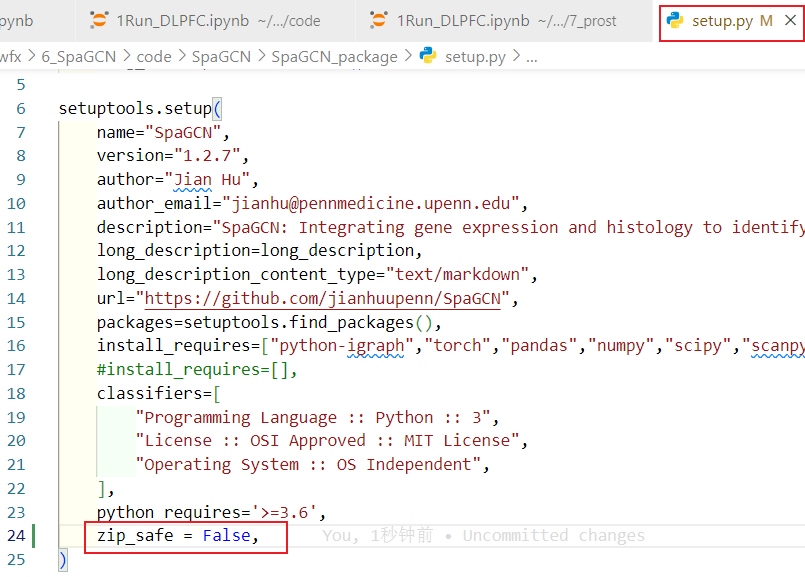


然后就好啦！

有的同学可能遇到下列这种情况，找不到包怎么办？然后搜索发现可以搜索到，但是不是文件夹：



这个问题是因为我们在安装github包的时候，setup.py文件没有添加这一行指令：



现在将这个命令加进去，重新安装这个模块。（安装的方式说了很多遍了，这里省略）。再回来我们就可以编辑这个文件了：

