

# 磁盘管理与常用软件安装！

## 1. 磁盘管理，分区，扩容

在此电脑中查看磁盘占用情况



鼠标右键windows开始，点击磁盘管理



可以看到我的电脑现在有四个分区，因为我安装了双系统（150GB用于Ubuntu），在Windows下只有C盘和D盘两个区，前面的100mb用于系统的启动。

我认为比较良好的分区方式就是C+D盘模式。

从左到右，随着分区的逐渐增多，电脑对其的读写速度也会相应减慢，C>D>E>F.....，不过一般情况下感知不明显，除非玩大型游戏，需要加载很多资源数据文件。

C盘用于存放核心软件，占用较大的软件。比如

通常安装在此文件夹下面：

Program Files	2023/9/26 17:20	文件夹
Program Files (x86)	2023/9/20 19:59	文件夹

为了更好的管理个人文件，或者减少C盘的压力，虽然现在大家的空间都很大，但是将所有软件安装在一个盘，查询起来相对不方便。

此时，我们便可以将一些非核心软件，比如QQ，百度云等放入D盘，该有自己的私人文件，比如上课的PPT，要交的Word文档。

📁 OBS	2023/8/17 21:57	文件夹
📁 QQ	2023/8/28 17:46	文件夹
📁 R	2023/8/19 21:16	文件夹
📁 Rpro	2023/8/19 23:27	文件夹
📁 stable-diffusion-webui	2023/9/2 23:56	文件夹
📁 tableau	2023/9/21 16:59	文件夹
📁 Typora	2023/8/25 16:36	文件夹
📁 VALL-E-X	2023/8/26 11:01	文件夹
📁 vlc	2023/8/14 15:09	文件夹
📁 wolai	2023/8/28 17:52	文件夹
📁 wps	2023/8/12 17:48	文件夹
📁 百度云	2023/8/12 13:54	文件夹

分区，即从我们现存的盘符中，根据我们的需要，进一步细化我们的磁盘，比如C盘存核心文件，D盘存非核心软件，此时，你还想分一个E盘出来存放你的个人照片，视频等。

首先，我们要知道我们所分出来的一块新的分区，是会紧靠于原分区的右侧，例如我们在C盘的基础上分出20G的空间(前提是你的C盘还有超过20G的空间)

右键C盘，点击，压缩卷



此时，会显示我们的C盘的空间占用情况

压缩前总计大小即我们的C盘目前的总空间：

255536MB/1024=249GB

可压缩空间：

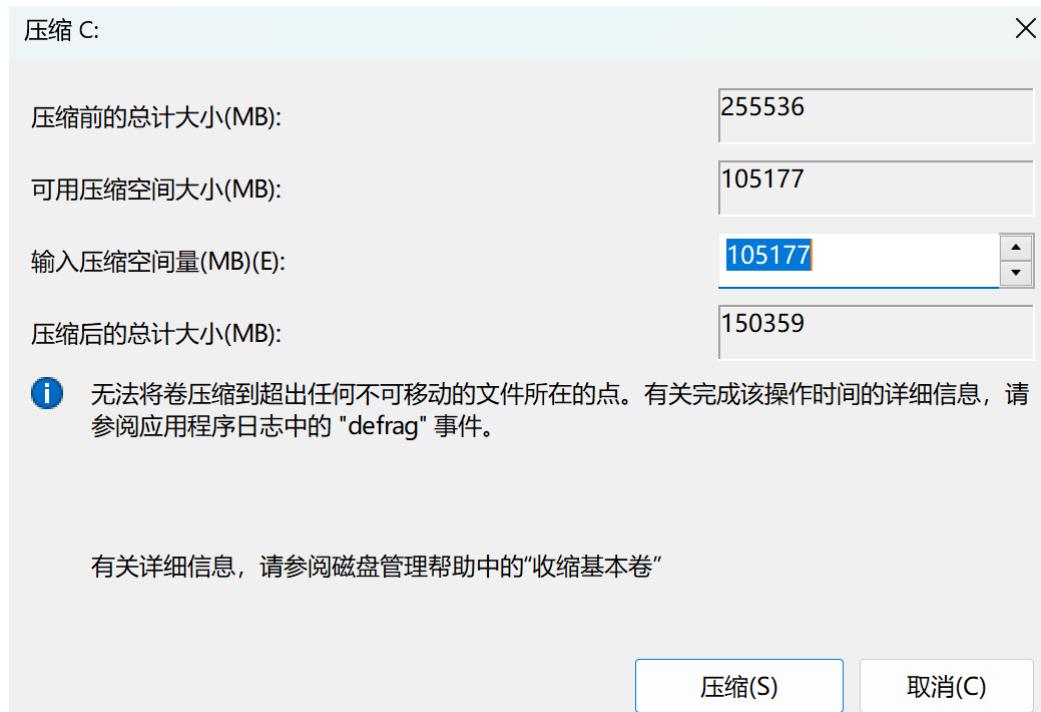
105177MB/1024=102GB

**我们需要压缩20GB**

**20x1024=20480MB**

**故我们在输入压缩空间量输入20480**

点击压缩。



而后，便会有会有一个未分配区域，20G

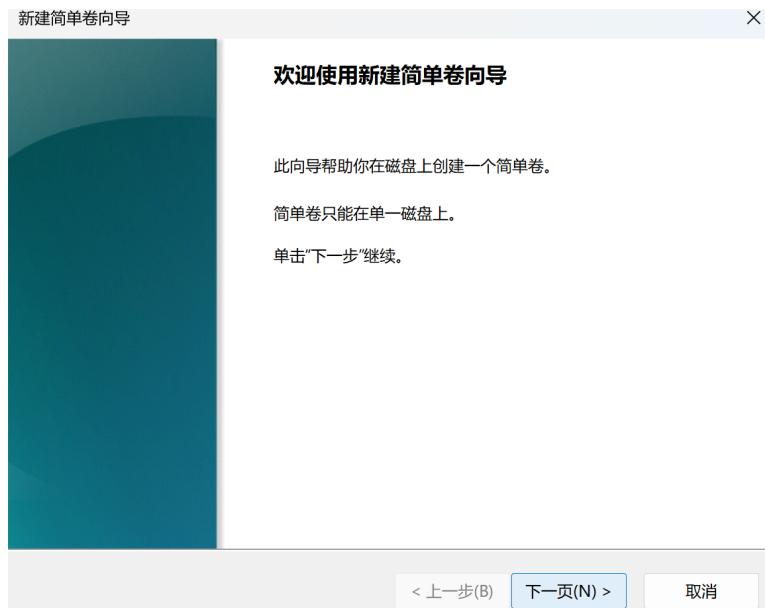


此时可以看到我们的C盘也从原来的249GB变成了229GB。

此时我们便可以使用这20G去创建一个新盘符，鼠标右键未分配区域，点击新建简单卷



点击下一页



点击下一页。



默认即可，但是可以自定义你的卷标，比如可以设置为pics

## 格式化分区

要在这个磁盘分区上储存数据，你必须先将其格式化。

选择是否要格式化这个卷；如果要格式化，要使用什么设置。

不要格式化这个卷(D)

按下列设置格式化这个卷(O):

文件系统(F):

NTFS

分配单元大小(A):

默认值

卷标(V):

新加卷

执行快速格式化(P)

启用文件和文件夹压缩(E)

< 上一步(B)

下一页(N) >

取消

点击完成

已选择下列设置:

卷类型: 简单卷  
选择的磁盘: 磁盘 0  
卷大小: 204.79 MB  
驱动器号或路径: E:  
文件系统: NTFS  
分配单元大小: 默认值  
卷标: pics  
快速格式化: 是

若要关闭此向导，请单击“完成”。

< 上一步(B)

完成

取消

我们就可以看到此时我们的新添加的一个分区pics了。



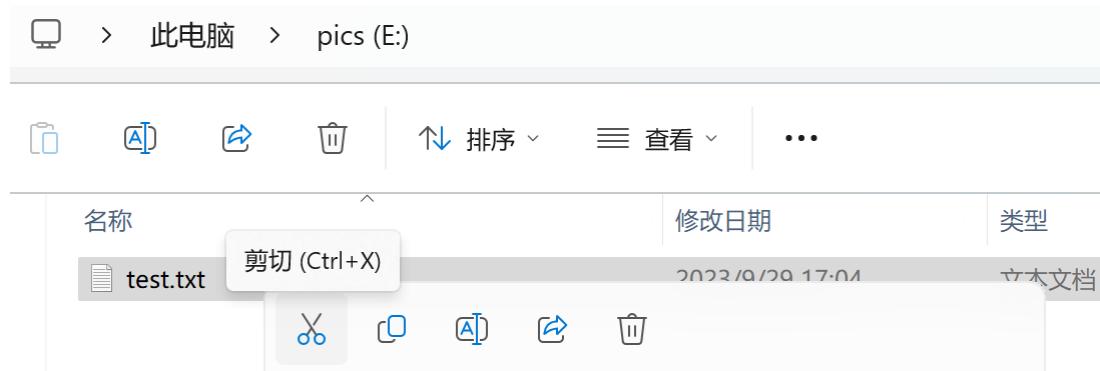
假设我不想要这么多分区，我想把原来的20G重新放入我的C盘该怎么办呢？

此时我们就需要做**扩展卷**，可以看到此时的扩展卷是不可被选择的，因为**扩展卷需要在被扩展的盘符的右侧存在未分配区域**，才可进行扩展。



我们刚刚从C盘压缩了20G出来，我们现在需要将这20G给扩展到C盘，首先需要把原盘符的文件备份到其他盘符

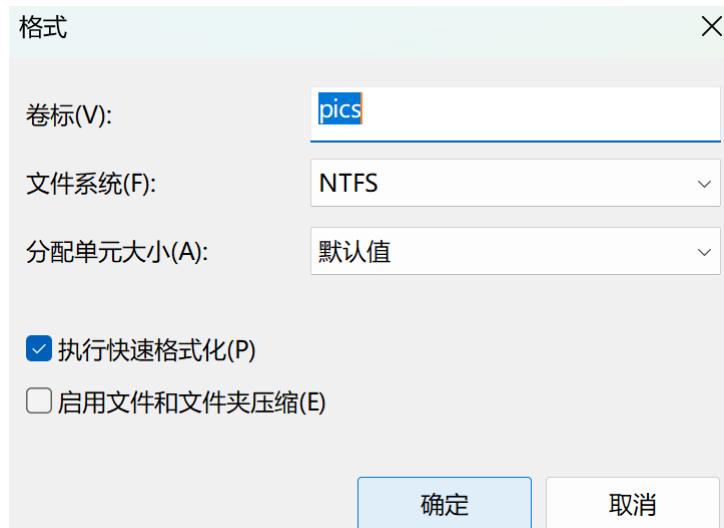
假设我的pics里面有一个重要文件



你可以直接将其剪切，并粘贴到其他盘符，**尽量确保你的此时这20G的盘符是一个空盘符**，如果存在一些不需要备份的数据文件，你可以打开磁盘管理，右键20G磁盘，直接格式化该盘符，



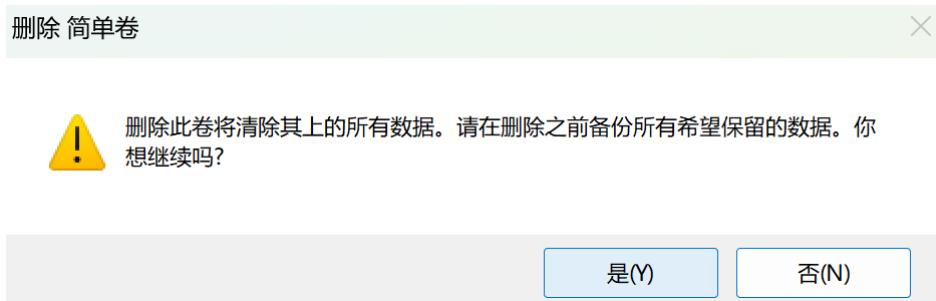
点击确定



格式化完成后，此时便是一个空盘符，右键该盘符，点击删除卷



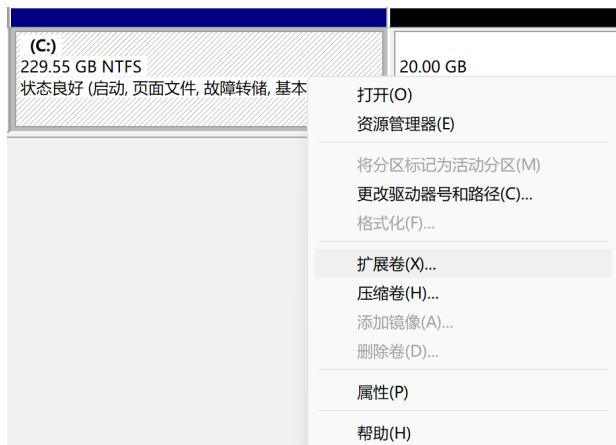
点击是



此时我们可以看到我们的磁盘管理中出现了那20G未分配空间

(C:)	229.55 GB NTFS 状态良好 (启动, 页面文件, 故障转储, 基本数据分区)	20.00 GB 未分配	D (D:)	76.57 GB NTFS 状态良好 (基本数据分区)

此时右键C盘，扩展卷便可选，点击扩展卷



点击下一页



$20479/1024=19.999\ldots=20$ , 20479便是我们的原来的20G的未分配

我们在选择空间量处选择20479, 即全部将未分配空间分配给C盘, 点击下一页

**选择磁盘**  
你可以用至少一个磁盘上的空间来扩展卷。

你只能将此卷扩展到如下所示的可用空间，因为不能将磁盘转换为动态磁盘，或者被扩展的卷是启动卷或系统卷。

可用(V):

已选的(S):

磁盘 0 20479 MB
---------------

卷大小总数(MB): 255535

最大可用空间量(MB): 20479

选择空间量(MB)(E): 20479

< 上一步(B) 下一步(N) > 取消

点击完成

已选择下列设置:

选择的磁盘: 磁盘 0 (20479 MB)

若要关闭此向导, 请单击“完成”。

< 上一步(B) 完成 取消

此时我们可以看到我们的磁盘管理中C盘又变成249GB了, 原来的未分配区域也不在了。

100 MB 状态良好 (EFI 系统分区)	(C): 249.55 GB NTFS 状态良好 (启动, 页面文件, 故障转储, 基本数据分区)	D (D): 76.57 GB NTFS 状态良好 (基本数据分区)
---------------------------	---	--

**! 注意, 磁盘的压缩与扩展都要在其右侧**

**压缩20G新区域, 会放置于原盘符的右侧,**

**扩展某盘符(如C盘), 需要在被扩展盘符(C盘)的右侧需要存在未分配区域**

**我们不能直接在D盘分配一部分空间然后再扩展到C盘**

但是可以间接操作,

以我的电脑为例, 假设我们需要再给扩展C盘20GB空间

现在我的D盘已经使用了 $76.5 - 51.5 = 25\text{GB}$ ,

我可以从D盘中, 分配50G创建一个新盘符, 将原来D盘的数据复制到新盘符

然后格式化D盘, 并删除D盘的卷, 此时就可以将剩余的 $76.5 - 50 = 26.5\text{GB}$ 扩展C盘。

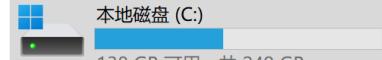
设备和驱动器



WPS云盘  
双击进入WPS云盘



百度网盘  
双击运行百度网盘



本地磁盘 (C):  
138 GB 可用, 共 249 GB



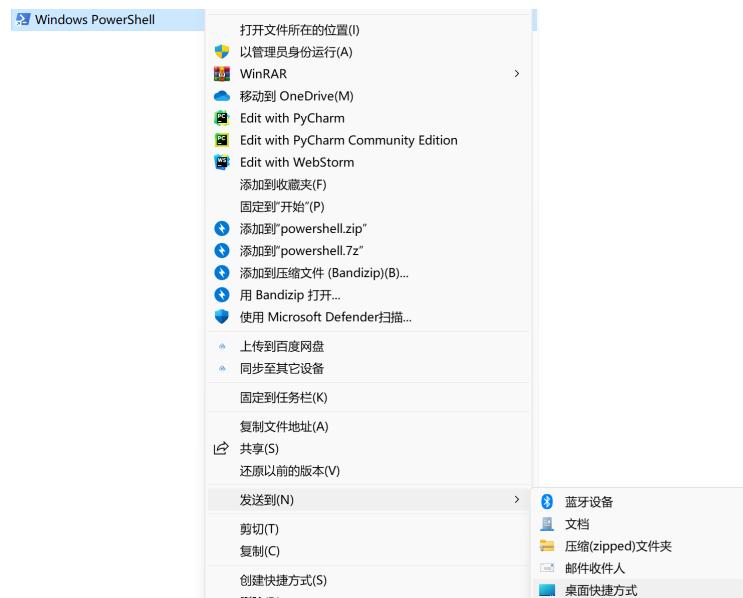
D (D):  
51.5 GB 可用, 共 76.5 GB

## 2.软件安装

python,anaconda,pycharm,git

之后我们会经常使用到power shell这个工具， power shell算是一种终端模拟器，

power shell在这个目录下， user代表用户， justin是我的用户名， 你打开的时候是你的用户名  
C:\Users\justin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Windows  
PowerShell



可以将其发送到桌面快捷方式，然后固定到任务栏

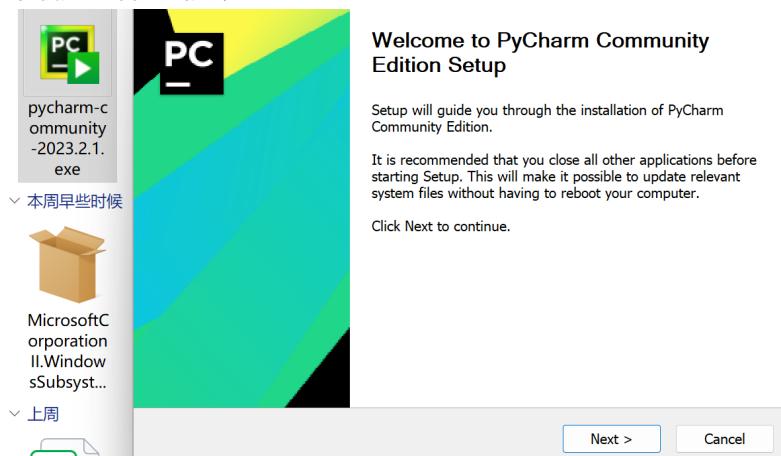
我的电脑已经配置好这些环境了，但是秉着认真负责的态度，我还是决定给卸载并重装一次！

### 0. pycharm

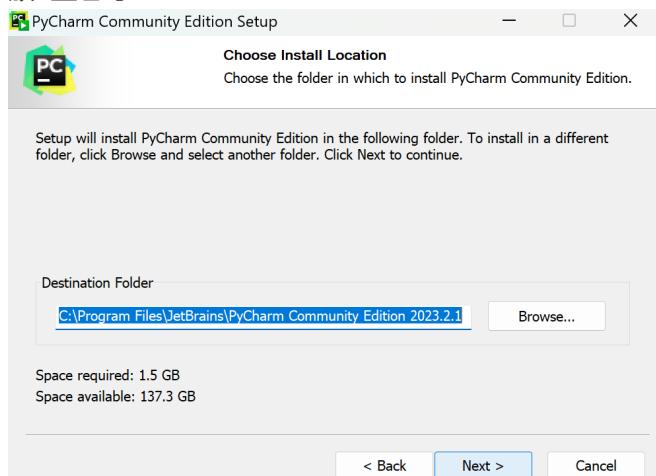
pycharm是一个集成开发环境，就是几乎所有的事情都可以在里面做。[pycharm](#)，选择社区版就可以了



下载完成后点击安装



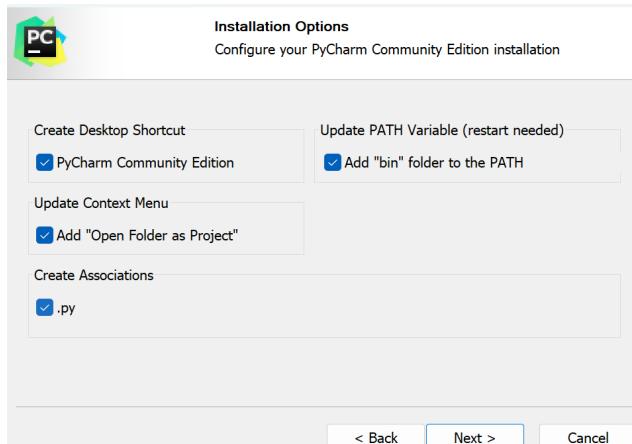
放C盘也可



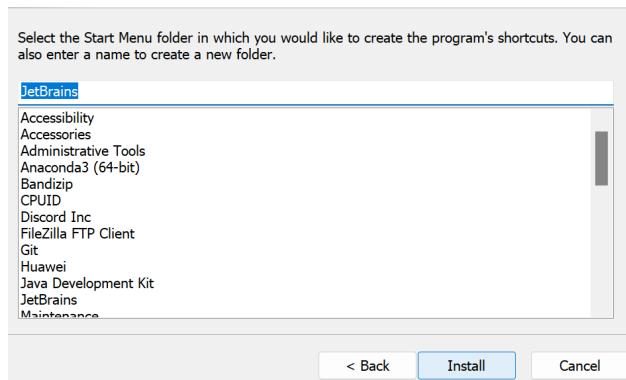
全选，next

第一个创建桌面快捷方式，第二个是配置环境变量

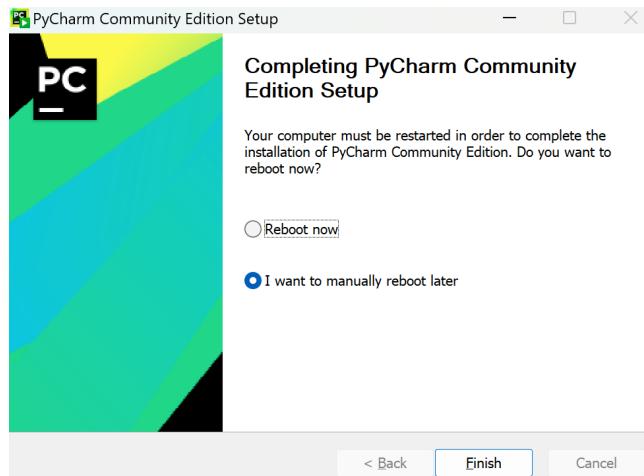
第三个是可以以项目形式打开文件夹，最后一个是创建链接到.py，即python文件默认用pycharm打开。



install,然后等待安装完成



finish, reboot later

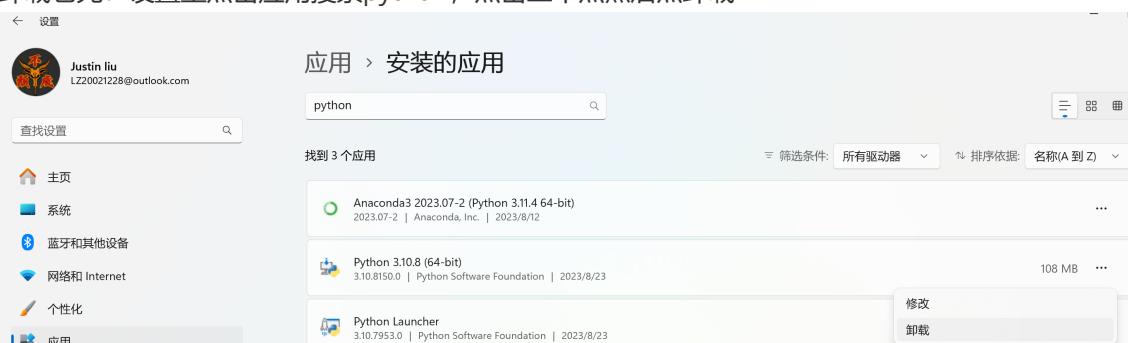


## 1. python

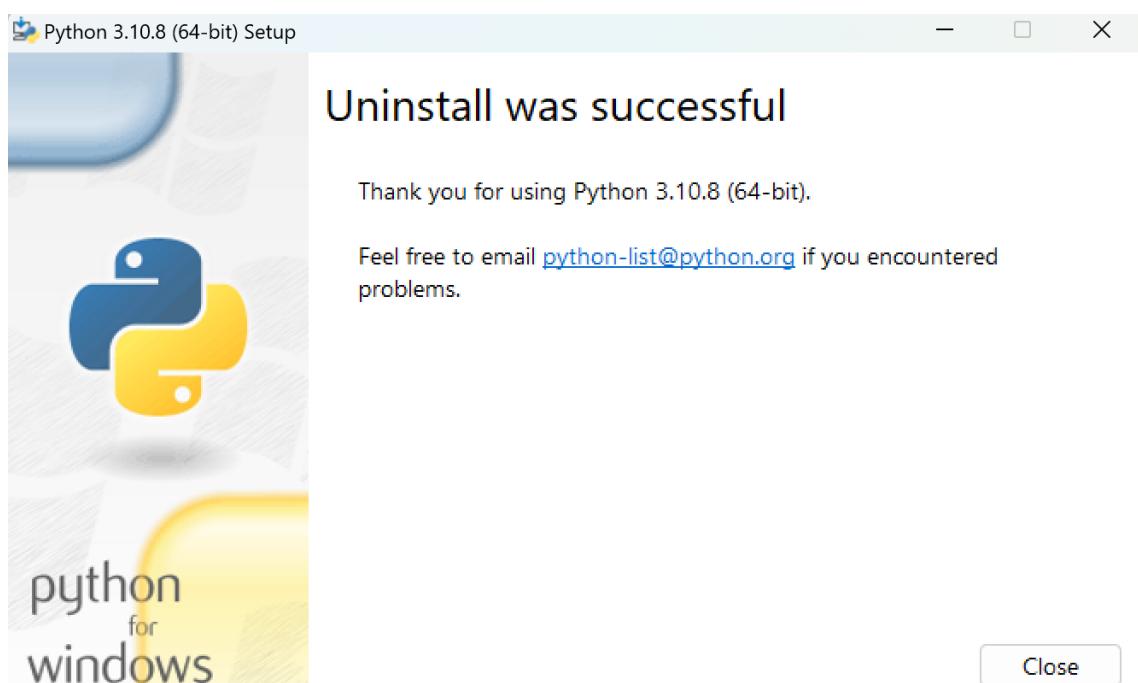
在power shell中输入python，我已经安装好了python3.10.8

```
PS C:\Users\justin> python
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

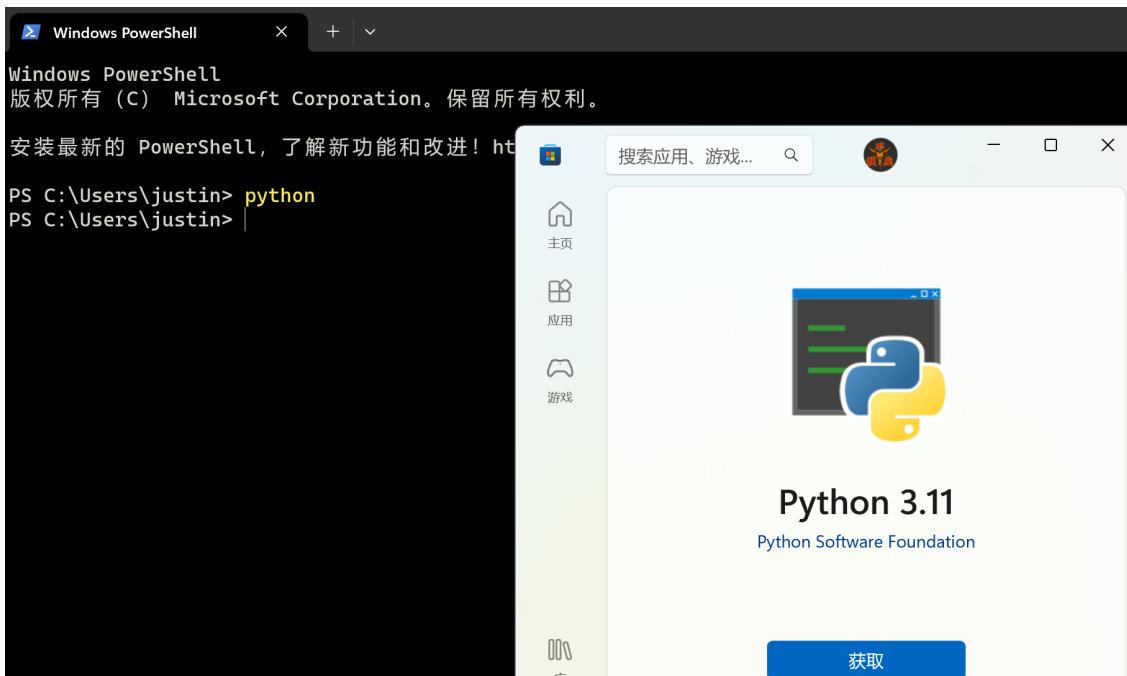
卸载它先！设置里点击应用搜索python，点击三个点然后点卸载



卸载成功！



此时在power shell中输入python  
便不会显示任何信息，而是推荐我们安装



但是我们不在微软商店安装，而是官网安装，python更新至今有很多很多版本

我最常用的是python3.10.8,因为现在很多AI项目要求python3.10.6，所以我们选择3.10.8兼容性更好一点

安装方法

在[python官网](#)

python官网，这是一个链接，你按住ctrl，然后鼠标左键点击即可跳到浏览器

在Downloads中点击All releases。

All releases

Source code

Windows

Download for Windows

Python 3.11.5

Note that Python 3.9+ cannot be used with Python 3.10.

在looking for a specific release中找到python3.10.8,点击download

Looking for a specific release?

Python releases by version number:

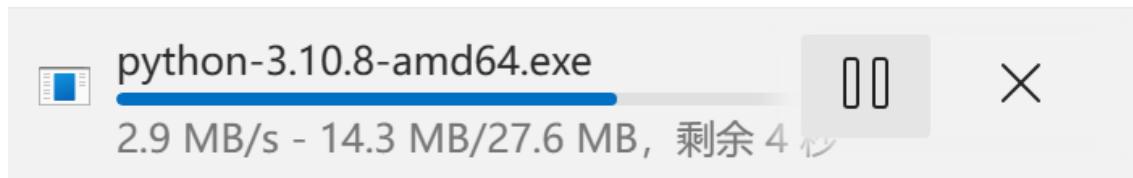
Release version	Release date	Click for more
Python 3.9.15	Oct. 11, 2022	Download <a href="#">Release Notes</a>
Python 3.8.15	Oct. 11, 2022	Download <a href="#">Release Notes</a>
Python 3.10.8	Oct. 11, 2022	Download <a href="#">Release Notes</a>
Python 3.7.15	Oct. 11, 2022	Download <a href="#">Release Notes</a>
Python 3.7.14	Sept. 6, 2022	Download <a href="#">Release Notes</a>
Python 3.8.14	Sept. 6, 2022	Download <a href="#">Release Notes</a>
Python 3.9.14	Sept. 6, 2022	Download <a href="#">Release Notes</a>

然后点击Windows installer64bit,

## Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore
Gzipped source tarball	Source release		fbe3fff11893916ad1756b15c8a48834	26015299	SIG	.sigstore
XZ compressed source tarball	Source release		e92356b012ed4d0e09675131d39b1bde	19619508	SIG	.sigstore
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	eb11b5816b1a37d934070145391eadfe	40883084	SIG	CRT
Windows embeddable package (32-bit)	Windows		e0dbe095e5963b26b8bf258fd2b9f41	7617241	SIG	CRT
Windows embeddable package (64-bit)	Windows		923be16c4cef2474b7982d16cea60ddb	8592015	SIG	CRT
Windows help file	Windows		0cbba41f049c8f496f4fb18d84430d9a	9379210	SIG	CRT
Windows installer (32-bit)	Windows		10efcd9a8777fe84f9a9c583d074e632	27820784	SIG	CRT
Windows installer (64-bit)	Windows	Recommended	308a3d095311fb82e5c696ab4036251	28978512	SIG	CRT

等待下载完成



下载完成后，我们打开其所在文件夹



然后勾选下面两个，选框，第一个表示安装时会使用管理员权限，第二个代表将其放入到环境变量（即我们在电脑的全局可使用python），并选择Customize installation(自定义安装)，



所有都勾选，

第一个代表会下载python的使用手册，很长很长，但是是官方的，可以看一看，虽然我没看过

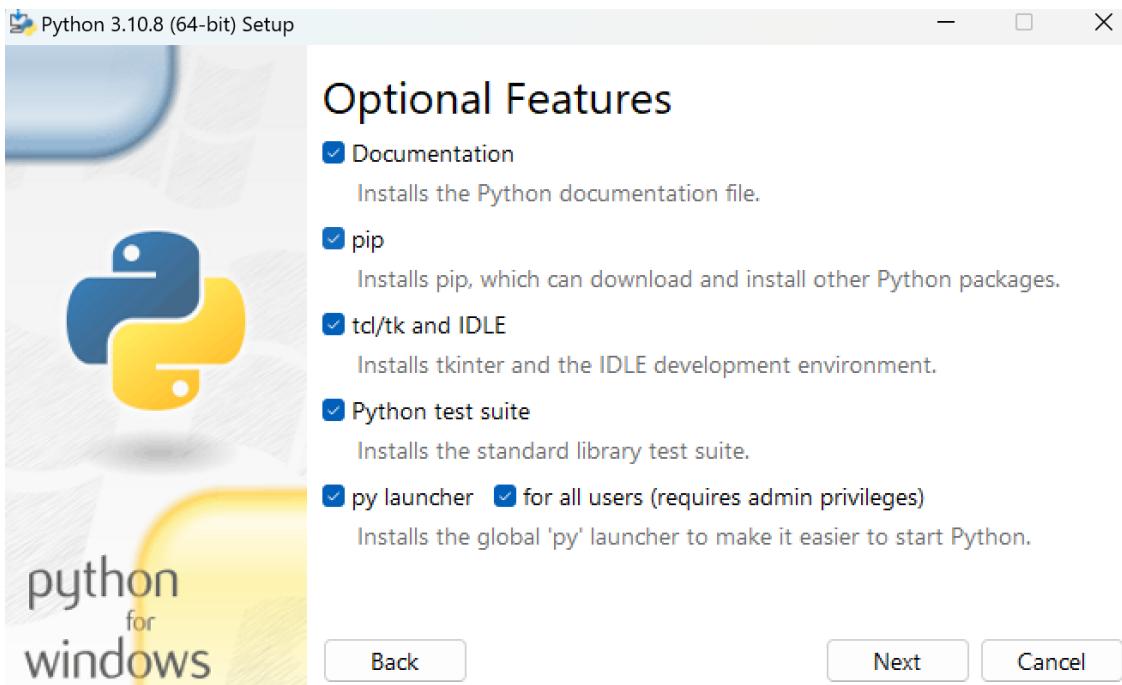
第二个代表安装pip用于安装第三方库，依赖之类的，以后配置虚拟环境会经常用到

第三个代表安装将tkinter安装到IDLE中，IDLE指的是集成开发环境，但是其实我们可以不需要这一个，因为我们待会要安装pycharm，不过也可以安装，等你厉害了可以用他官方的IDLE写代码的时候，也可能会用到这个tkinter，用于创建一些界面啊之类的

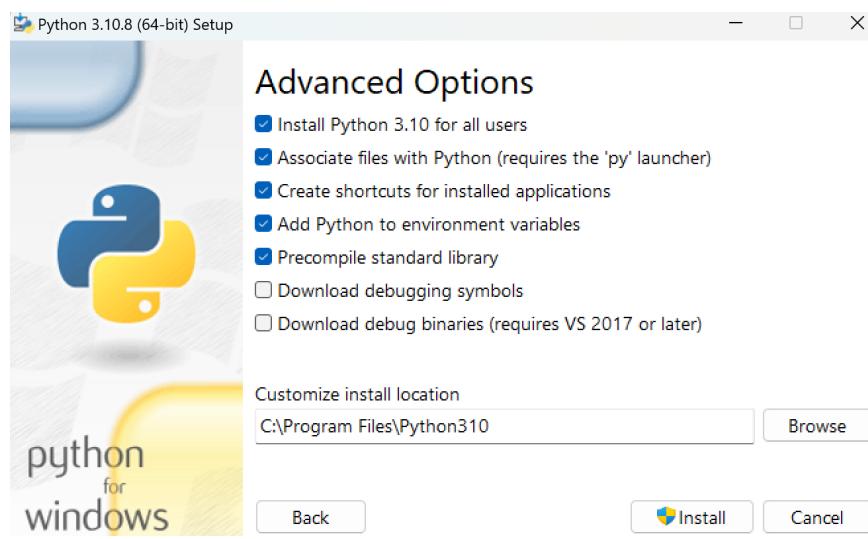
第四个代表测试框架，我对测试不太了解，但是还是安装上吧，万一以后用的上

第五个for all users 代表为所有用户安装，这个也安装上吧，虽然是全民pc的时代了，但是万一你之后想创建新的users就不用重新安装python了

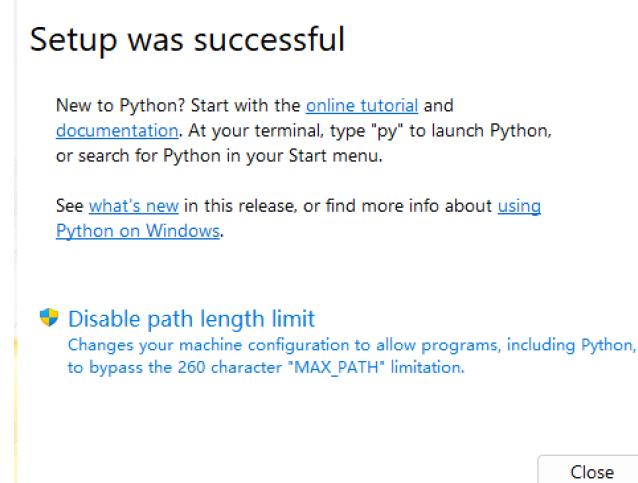
点击next



勾选如下，安装路径可以的话就放在C盘吧，因为python也算是相对比较重要的软件，点击install



安装完成，点击close

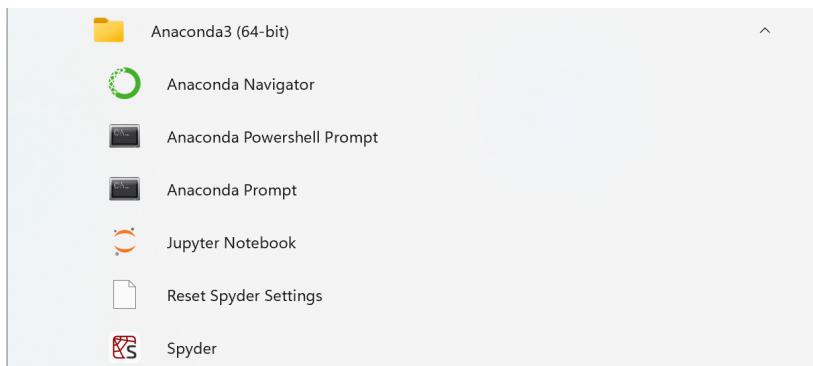


我们再次再powershell中输入python，此时就会显示python的版本信息。

```
PS C:\Users\justin> python
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

## 2. anaconda

anaconda是一个很强大的管理工具，帮助你管理python的版本，创建不同版本的虚拟环境，



还是先卸载：

应用 > 安装的应用

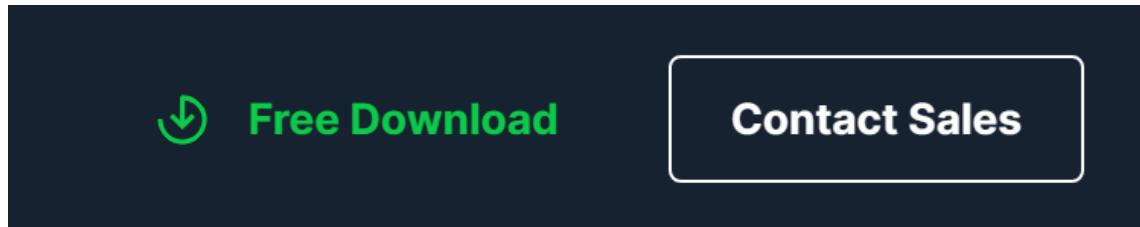
找到 1 个应用

筛选条件: 所有驱动器 排序依据: 名称(A 到 Z)

conda |

Anaconda3 2023.07-2 (Python 3.11.4 64-bit) 2023.07-2   Anaconda, Inc.   2023/8/12
--

然后去[官网](#),点击Free Download



选择windows

# Free Download

Everything you need to get started in data science on your workstation.

- ✓ Free distribution install
- ✓ Thousands of the most fundamental DS, AI, and ML packages
- ✓ Manage packages and environments from desktop application
- ✓ Deploy across hardware and software platforms

[Code in the Cloud](#) [Download](#)

Get Additional Installers

Windows | Apple | macOS

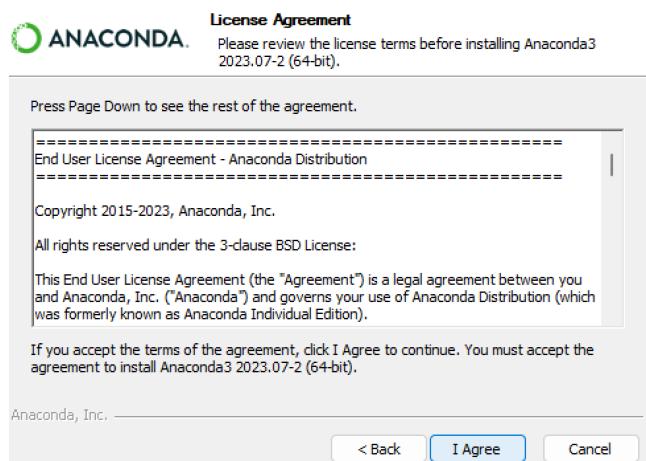
下载好后，双击



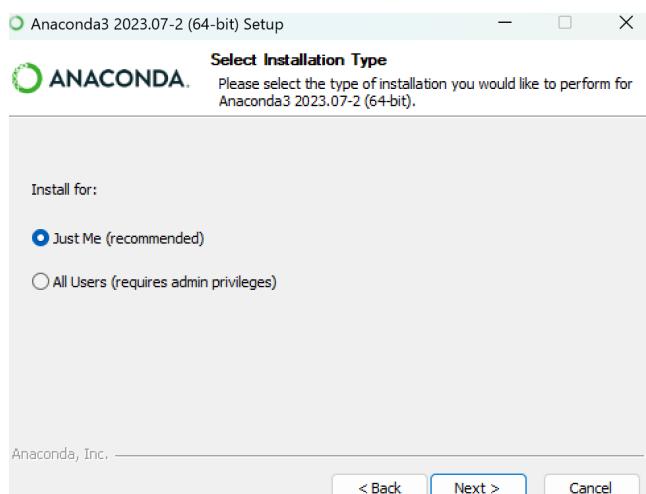
next



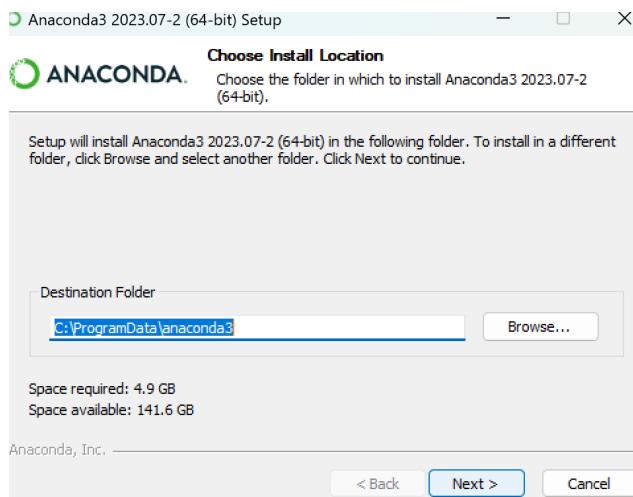
I agree



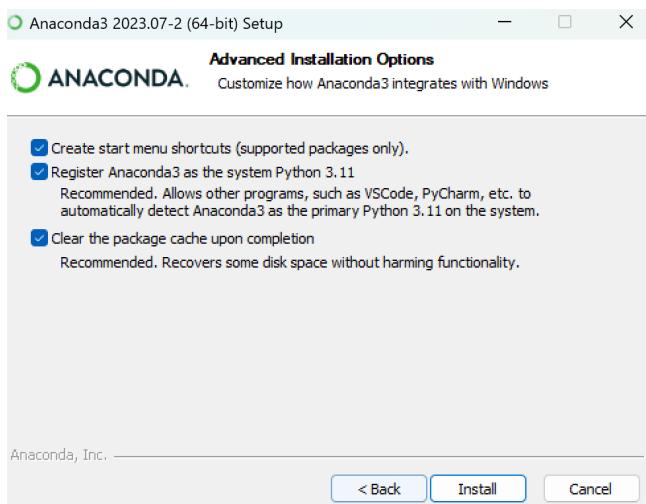
all users吧，需要管理员授权



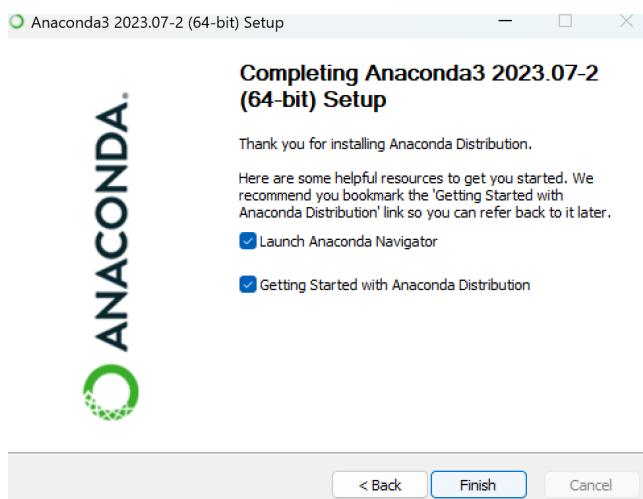
放在C盘可以的，如果空间够的话



全部勾选

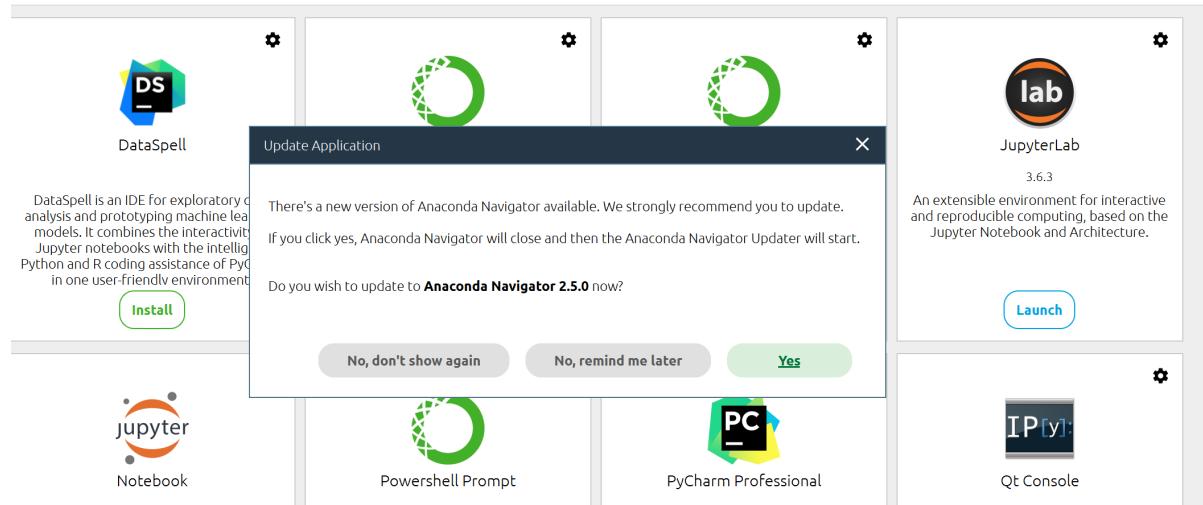


等待安装完成，即可。点击Finish

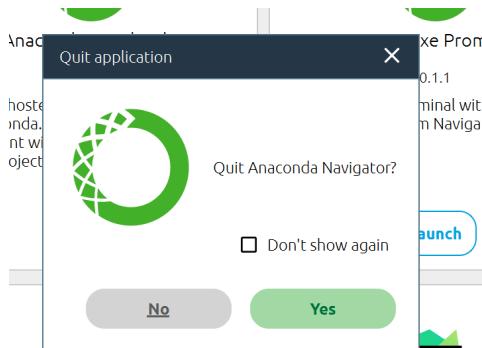


然后它可能会跳出来一个绿色的圈圈，一会就会打开其主页，我们什么都不用管，

点击No,don't show again，然后直接退出

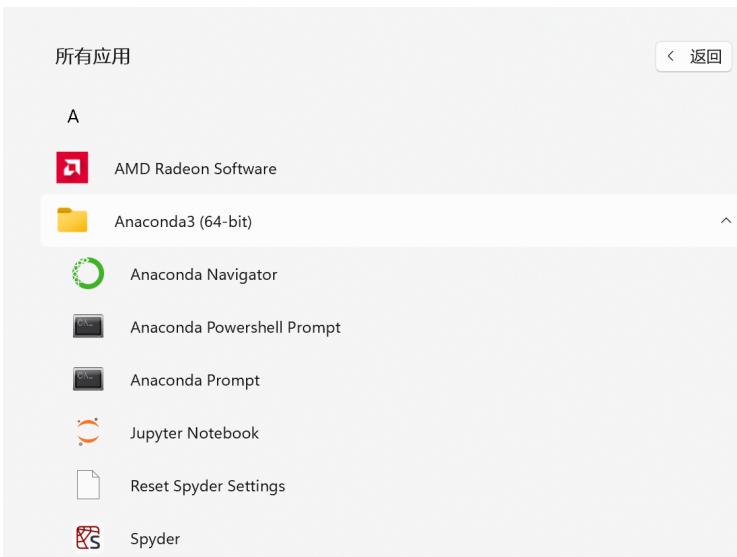


点击Yes



此时你可以在应用里看到，Anaconda3这个文件夹

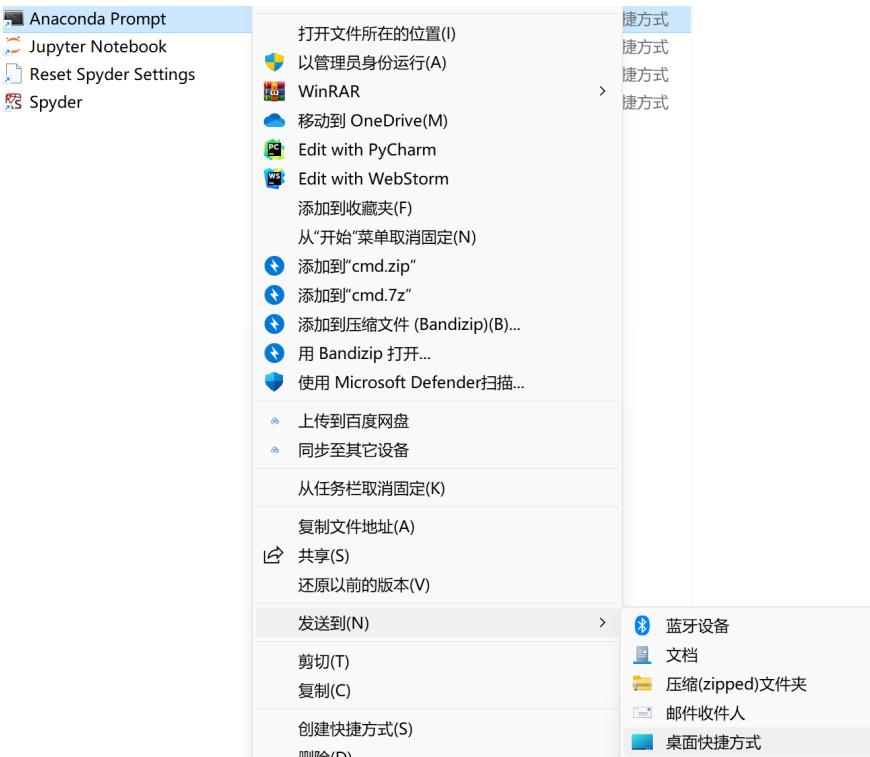




右键Anaconda Prompt，更多，点击打开文件位置，此后，我们使用的最多的便是这个类似于专属于anaconda的终端模拟器



将其快捷方式发送到桌面快捷方式。



然后将其拖到任务栏，直到出现链接，即可



我们在任务栏中打开它并输入

```
conda env list
```

```
(base) C:\Users\justin>conda env list
# conda environments:
#
base          * C:\ProgramData\anaconda3
CI
chatgpt
dow
flask
fsas
gradii
pydml
urp
```

The screenshot shows the Anaconda Prompt window with the command 'conda env list' run. It lists several virtual environments: base, CI, chatgpt, dow, flask, fsas, gradii, pydml, and urp. The 'base' environment is marked with an asterisk (\*) and is located at 'C:\ProgramData\anaconda3'. Other environments are located at 'C:\Users\justin\.conda\envs\CI', 'C:\Users\justin\.conda\envs\chatgpt', etc.

就会显示我们系统中现有的虚拟环境，你初次安装应该是只有base这个虚拟环境

ok，接下来我给你稍微讲一下什么是虚拟环境(可能不太准确)，就是我们在开发项目的过程中，或者说去运行别人的项目的过程中，每个人所需要的python版本都不一样，然后需要的第三方库，依赖也不一样

我们创建并配置我们的虚拟环境来满足项目虚拟环境要求。

在虚拟环境中包括了python版本，python解释器们还有python的第三方库等。

对于虚拟环境就大概讲这么多，就是为了帮我们的项目满足实现我们依赖，版本等要求。

为什么使用anaconda？因为anaconda是一个很好python的虚拟环境管理的软件，其实有很多创建虚拟环境的方式，但是相比起来，使用anaconda来管你虚拟环境是最为方便快捷的。

那么接下来，我们便可以使用使用简单的密令来创建一下虚拟环境。

比如我们刚开始学习python我们创建了一个项目叫做base\_python。

我们希望我们的虚拟环境的python版本是3.10.6，我们便可通过如下密令创建虚拟环境

```
#代表创建一个新的名叫base_learn, python版本为3.10.6的虚拟环境  
conda create -n base_learn python==3.10.6
```

#打开固定到任务栏的Anaconda prompt, 输入上述密令并回车, 期间会出现一些调试信息, 不用管

```
(base) C:\Users\justin>conda create -n base_learn python==3.10.6  
Retrieving notices: ... working ... DEBUG:urllib3.connectionpool:Starting  
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.ana  
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): mirrors.  
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.ana
```

直到它出现, 下图, 输入y,并回车

```
The following NEW packages will be INSTALLED:
```

bzip2	pkgs/main/win-64::bzip2-1.0.8-he774522_0
ca-certificates	pkgs/main/win-64::ca-certificates-2023.08.22-haa95532_0
libffi	pkgs/main/win-64::libffi-3.4.4-hd77b12b_0
openssl	pkgs/main/win-64::openssl-1.1.1w-h2bbff1b_0
pip	pkgs/main/win-64::pip-23.2.1-py310haa95532_0
python	pkgs/main/win-64::python-3.10.6-hbb2ffb3_1
setuptools	pkgs/main/win-64::setuptools-68.0.0-py310haa95532_0
sqlite	pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0
tk	pkgs/main/win-64::tk-8.6.12-h2bbff1b_0
tzdata	pkgs/noarch::tzdata-2023c-h04d1e81_0
vc	pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime	pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel	pkgs/main/win-64::wheel-0.41.2-py310haa95532_0
xz	pkgs/main/win-64::xz-5.4.2-h8cc25b3_0
zlib	pkgs/main/win-64::zlib-1.2.13-h8cc25b3_0

```
Proceed ([y]/n)? y
```

便可以看到一些基础的包, 和python3.10.6正在安装

```
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/xz-5.4.2-h8cc25b3_0.conda HTTP/1.1" 200  
| 6064332.13 | 113 KB | ##### | 100%  
| wheel-0.41.2 | 127 KB | ##### | 100% D  
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/ca-certificates-2023.08.22-haa95532_0.c  
onda HTTP/1.1" 200 126244  
| bzip2-1.0.8 | 113 KB | ##### | 100% D  
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/pip-23.2.1-py310haa95532_0.conda HTTP/1  
| 1" 200 2982252 | 592 KB | ## | 3%  
| | | | D  
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/openssl-1.1.1w-h2bbff1b_0.conda HTTP/1.  
| 1" 200 5763830  
  
| python-3.10.6 | 13.8 MB | ##### | 35%  
| xz-5.4.2 | 592 KB | ##### | 14%  
| pip-23.2.1 | 2.8 MB | ##9 | 4%  
| setuptools-68.0.0 | 934 KB | ##### | 100%  
| xz-5.4.2 | 592 KB | ##### | 100%  
| pip-23.2.1 | 2.8 MB | ##### | 77%
```

安装完成

```
Preparing transaction: done  
Verifying transaction: done  
Executing transaction: done  
#  
# To activate this environment, use  
#  
#     $ conda activate base_learn  
#  
# To deactivate an active environment, use  
#  
#     $ conda deactivate
```

```
#输入 conda env list即可看到我们刚创建的base_learn虚拟环境 ()
```

```
(base) C:\Users\justin>conda env list
# conda environments:
#
base          * C:\ProgramData\anaconda3
CI
base_learn
chatgpt
dow
flask
fsas
gradii
pydml
urp
```

```
输入cls 并回车，来清屏
```

```
(base) C:\Users\justin>cls|
```

```
#进入虚拟环境（进入虚拟环境来管理第三方库）
```

```
conda activate base_learn
```

```
#可以看到前面的base变成了base_learn，前面的()代表我们目前在那个虚拟环境下
```

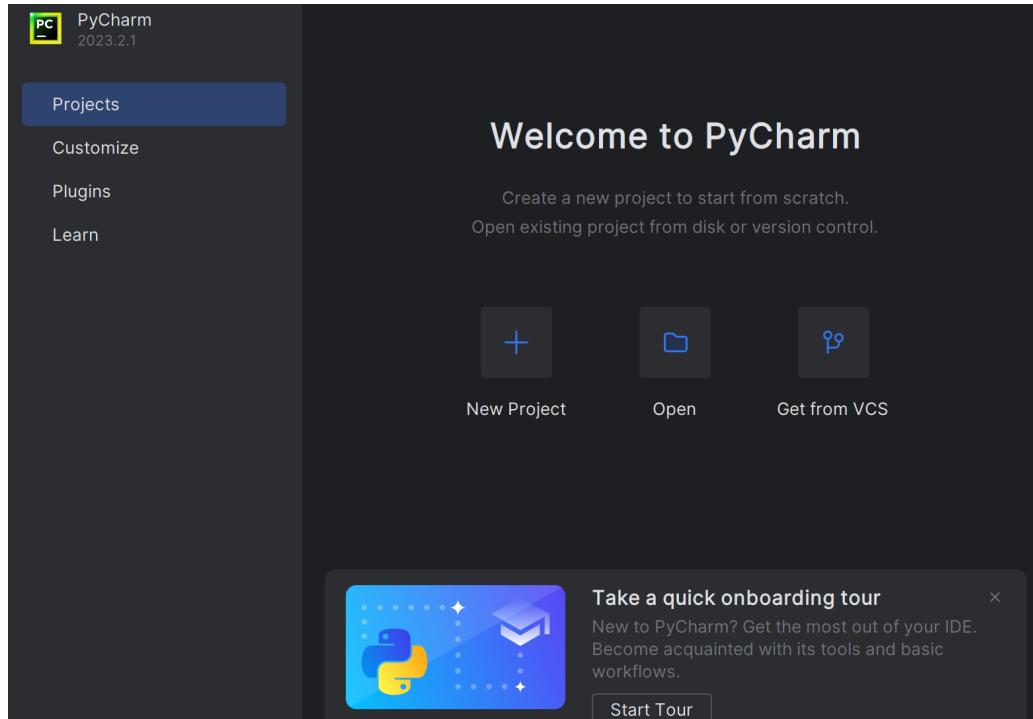
```
(base) C:\Users\justin>conda activate base_learn
(base_learn) C:\Users\justin>|
```

```
使用conda deactivate 退出当前虚拟环境
```

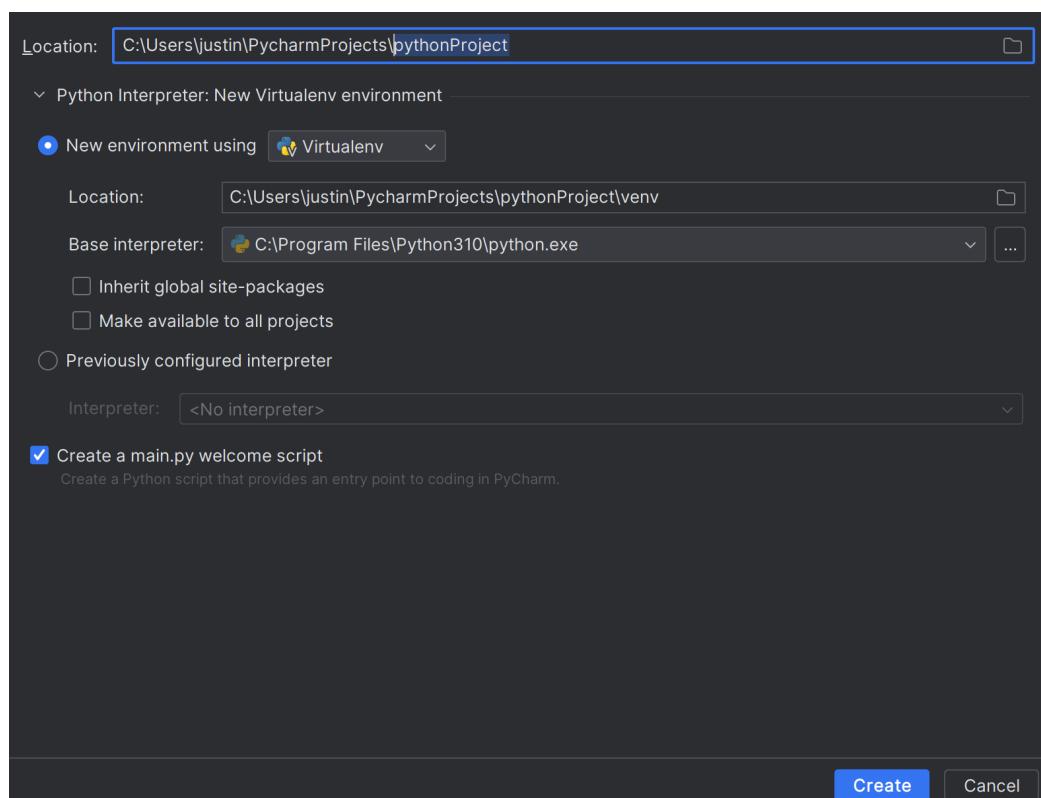
```
(base_learn) C:\Users\justin>conda deactivate
(base) C:\Users\justin>|
```

3. 创建一个项目

打开pycharm, 点击 New Project



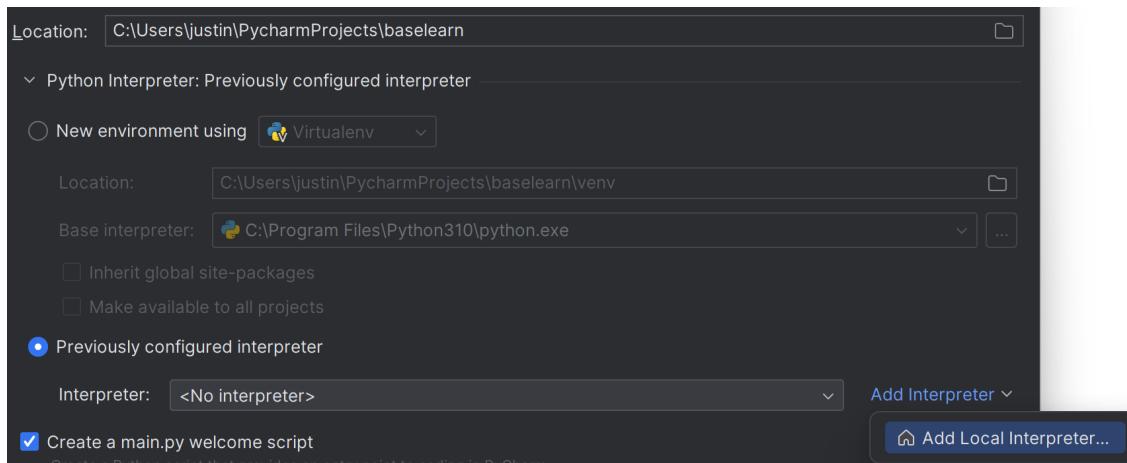
location代表项目位置最后的pythonProject代表项目名称



我们设置项目名称为baselearn

选择虚拟环境为Previousls configured interpreter(之前配置的解释器 (即我们之前创建的虚拟环境))

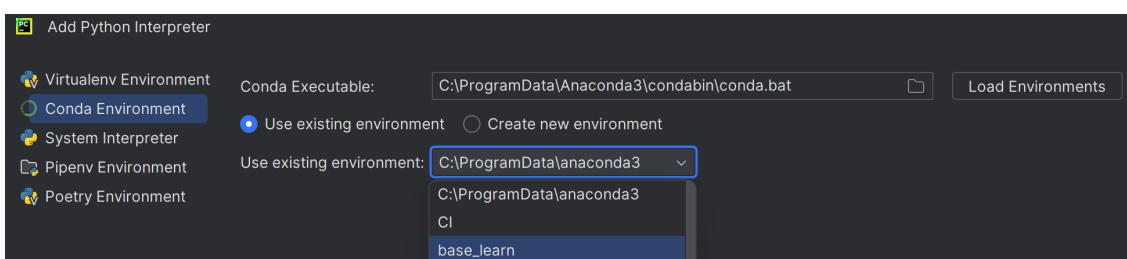
点击add local interpreter



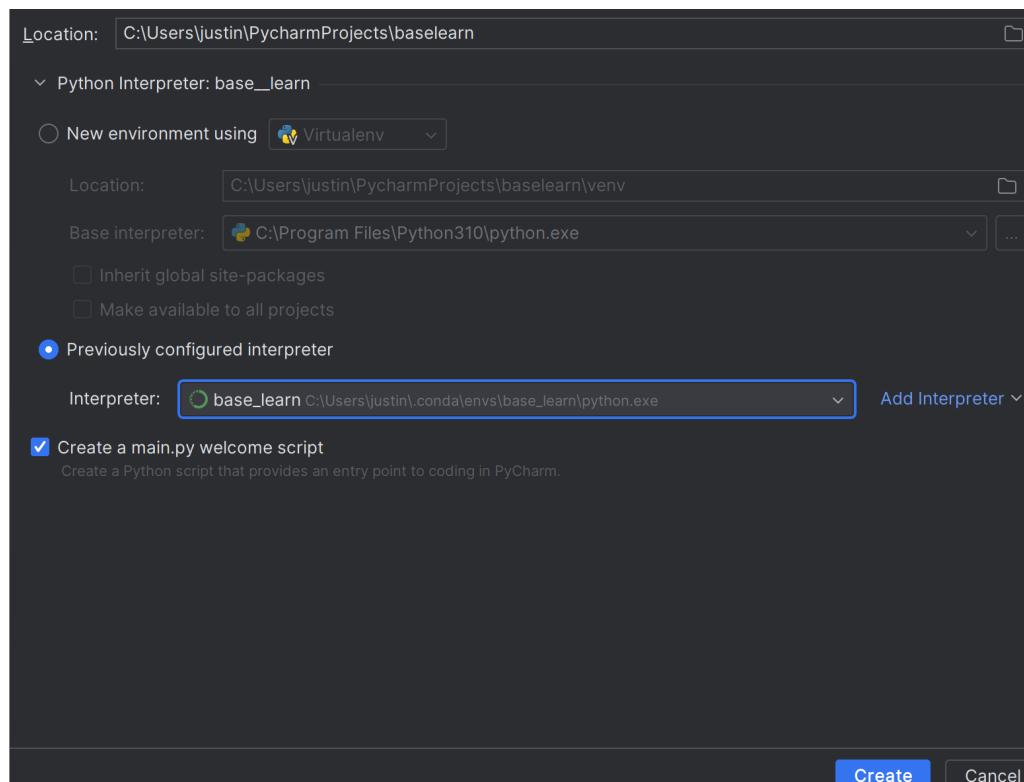
点击conda enviroment

选择 using existing enviroment 然后找到我们之前创建的base\_learn

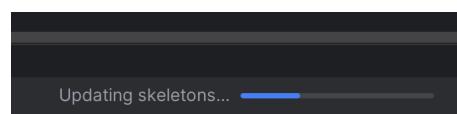
点击ok



点击create



等待下面进度条加载完成，表示我们的虚拟环境被读取完成



左下角这几个从上到下分别是

控制台：主要用于程序的调试，**debug**

包：运行代码的第三方库

服务：比如网页服务，**jupyter**这种在线代码编辑

终端：这个应该是之后用的最多的，安装第三方库啊之类的，**git**创建版本啊，都在这里

报错：程序问题

版本控制(**git**)：显示版本信息，这个也很有用！我会在接下来的教程教会你如何使用**git**和**github**



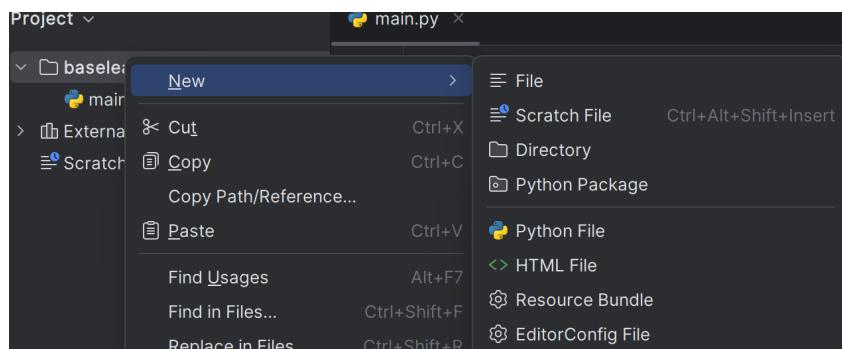
#### 4. 项目管理之良好的文件管理方式

通常，我们的项目有如下文件，文件夹

```
project_name
    --src(源代码)
        --train.py
        --test.py
        .....
    --data(数据文件，根据自己的项目来创建)
        --example.csv
        --train
            --a.png
            --b.png
            --c.png
            .....
        --test
            --d.png
            --e.png
            --f.png
            .....
    --assets(通常存放一些媒体文件，比如图标啊，图片啊，readme.md,介绍文档中的图片之类的)
        --icon.svg
        --pics
        --ui.svg
        .....
main.py/run.py/app.py(一般情况下，这三个文件用于整个项目的启动)
README.md(项目介绍文档，使用markdown语言编写，markdown是一种轻量级的标记语言，可以
用于记笔记，撰写文档等多种用途。)
requirement.txt(依赖文档，列出了该项目中所用到的第三方库)
```

#在**pycharm**中右键项目选择**new**来创建对应的文件，或者文件夹

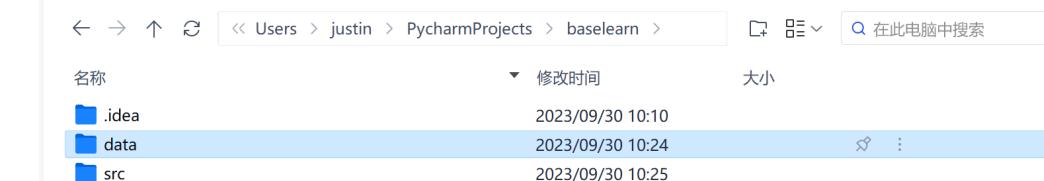
比如我们现在要创建一个能够将两个**csv**文件合并的项目



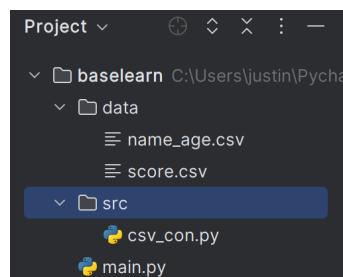
创建一个表格，在文件类型处选择csv（逗号分隔符）文件类型



直接存放到我们的项目文件的data项目中，同理再创建一个csv文件，命名最好不要带中文，经常会出现编码问题，最好使用英文字母



此时可以看到我们的项目文件结构如下，是不是相对清晰明了



ok,我们现在需要合并这两个csv文件，解决方案有很多，  
在此教程中，我们选择使用pandas这个第三方库中的pd.concat来完成，代码如下：

```
import pandas as pd #导入第三方库pandas并实例化一个对象叫pd

#定义一个函数名叫con_csv,接受两个参数, file1, file2
def con_csv(file1,file2):
    # 读取两个CSV文件
    df1 = pd.read_csv(file1)
    df2 = pd.read_csv(file2)

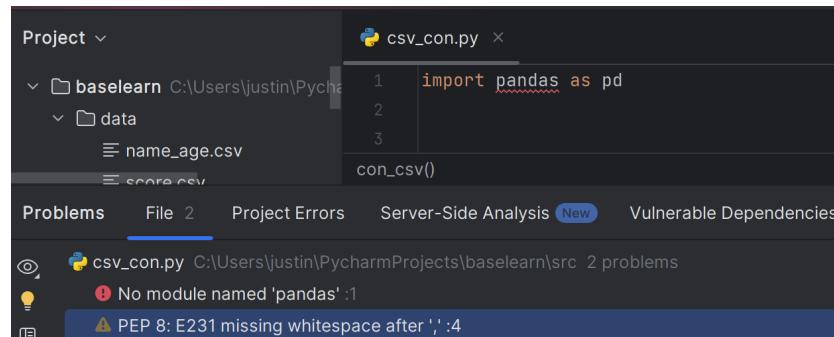
    # 使用concat函数合并两个数据框
    merged_df = pd.concat([df1, df2],axis=1)

    # 将合并后的数据框保存为新的CSV文件
    merged_df.to_csv('merged_file.csv', index=False)
```

我们将代码复制到csv\_con.py文件中,发现提示no module named 'pandas',表示我们没有pandas这个第三方库

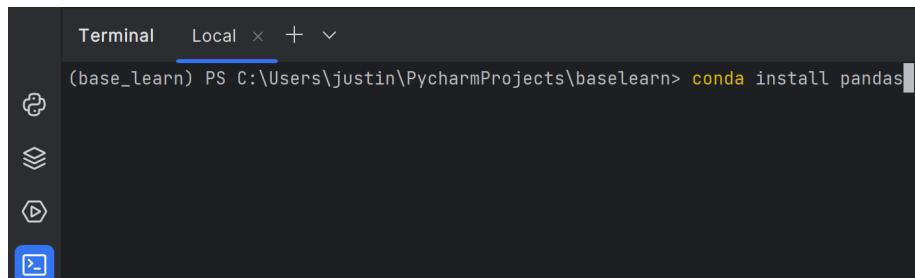
下面的PEP 8: E231 missing whitespace after ',' :4 表示在第四行的,后面确实一个空格。

(这种是代码美观性的提示,可以不予管理,但是你要是有强迫症,可以右键它,选择 show quick fixes 选择reformat the file)



在pycharm的左下角,点击终端模拟器

输入conda install pandas,回车,在我们的虚拟环境中安装pandas



同样会输出很多调试信息(),直到最后提示你输入y/n 输入y回车即可

```
\ DEBUG:urllib3.connectionpool:https://mirrors.tuna.tsinghua.edu.cn:443 "GET /anaconda/pkgs/free/noarch/repo/repodata.json HTTP/1.1" 200 None
\ DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/noarch/current_repodata.json HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/msys2/noarch/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/r/win-64/current_repodata.json HTTP/1.1" 304 0
DEBUG:urllib3.connectionpool:https://repo.anaconda.com:443 "GET /pkgs/main/win-64/current_repodata.json HTTP/1.1" 200 None

mkl-service      pkgs/main/win-64::mkl-service-2.4.0-py310h2bbff1b_1
mkl_fft          pkgs/main/win-64::mkl_fft-1.3.8-py310h2bbff1b_0
mkl_random        pkgs/main/win-64::mkl_random-1.2.4-py310h59b6b97_0
numexpr           pkgs/main/win-64::numexpr-2.8.4-py310h2cd9be0_1
numpy              pkgs/main/win-64::numpy-1.26.0-py310h055cbcc_0
numpy-base         pkgs/main/win-64::numpy-base-1.26.0-py310h65a83cf_0
pandas             pkgs/main/win-64::pandas-2.0.3-py310h4ed8f06_0
python-dateutil    pkgs/main/noarch::python-dateutil-2.8.2-pyhd3eb1b0_0
python-tzdata       pkgs/main/noarch::python-tzdata-2023.3-pyhd3eb1b0_0
pytz                pkgs/main/win-64::pytz-2023.3.post1-py310haa95532_0
six                 pkgs/main/noarch::six-1.16.0-pyhd3eb1b0_1
tbb                  pkgs/main/win-64::tbb-2021.8.0-h59b6b97_0

Proceed ([y]/n)?
```

等待安装成功,显示done,然后pycharm右下角会自动更新虚拟环境

```
pandas-2.0.3           | 9.8 MB    |
EBUG:urllib3.connectionpool:Starting new HTTPS connection (1): repo.anaconda.co
mkl_random-1.2.4       | 227 KB     |
done
(base_learn) PS C:\Users\justin\PycharmProjects\baselearn>
```

Background Tasks

Updating Python interpreter

Scanning installed packages...

Hide processes (1) 4:18 CRLF UTF-8 4 spaces base\_learn

然后我们在终端中输入`conda list`即可看到我们安装的第三方库  
此时就可以看到我们的pandas版本是2.0.3，可以使用`ctrl`键+`l`清屏

```
(base_learn) PS C:\Users\justin\PycharmProjects\baselearn> conda list
# packages in environment at C:\Users\justin\.conda\envs\base_learn:
#
# Name          Version      Build  Channel
blas           1.0           mkl    defaults
bottleneck    1.3.5        py310h9128911_0  defaults
bz2            1.0.8         he774522_0  defaults
ca-certificates 2023.08.22   haa95532_0  defaults
intel-openmp   2023.1.0     h59b6b97_46319 defaults
libffi          3.4.4         hd77b12b_0  defaults
mkl             2023.1.0     h6b88ed4_46357 defaults
mkl-service    2.4.0         py310h2bbff1b_1  defaults
mkl_fft         1.3.8         py310h2bbff1b_0  defaults
mkl_random     1.2.4         py310h59b6b97_0  defaults
numexpr          2.8.4        py310h2cd9be0_1  defaults
numpy           1.26.0        py310h055cbcc_0  defaults
numpy-base      1.26.0        py310h65a83cf_0  defaults
openssl         1.1.1w        h2bbff1b_0  defaults
pandas          2.0.3        py310h4ed8f06_0  defaults
```

可以看到我们的代码此时已经不再报错



此时我们创建两个路径的字符串，并调用上述函数，即可给我们创建出合并后的CSV文件，并存放于代码文件的同级目录下

```
import pandas as pd

def con_csv(file1, file2):
    # 读取两个CSV文件
    df1 = pd.read_csv(file1)
    df2 = pd.read_csv(file2)

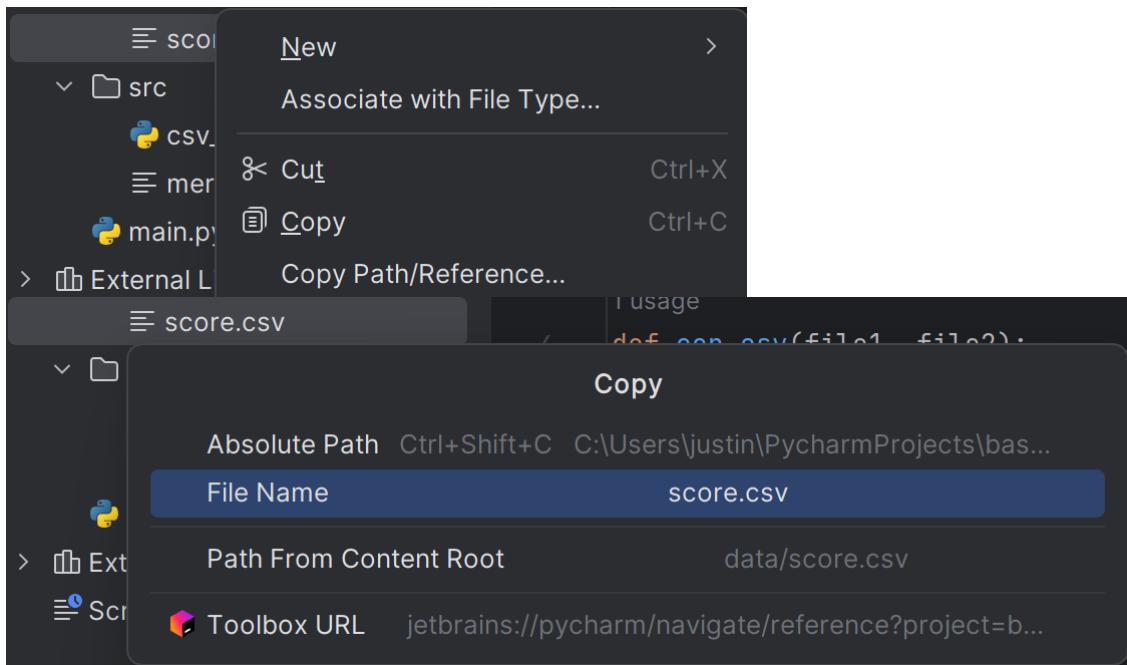
    # 使用concat函数合并两个数据框
    merged_df = pd.concat([df1, df2], axis=1)

    # 将合并后的数据框保存为新的CSV文件
```

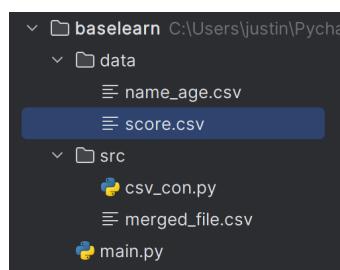
```
merged_df.to_csv('merged_file.csv', index=False)
```

```
path1 = "../data/name_age.csv"  
path2 = "../data/score.csv"  
con_csv(path1, path2)
```

ok, 这里再提一下关于文件路径的问题, 我们一般常用的是绝对路径和相对路径  
对于整个盘符来讲, 绝对路径就是从你这个盘的最初开始一步一步去检索  
在pycharm中, 我们可以, 右键某个文件, 或者文件夹, 点击copy path/Reference  
然后会跳出来三个选项, 第一个便是绝对路径, 从c盘开始检索, 一直到最后的文件名  
`C:\Users\justin\PycharmProjects\baselearn\data\score.csv`  
而相对路径则是从文件所在的文件夹开始检索  
`data/score.csv`  
在上述项目代码中, 我们的路径前面为什么要加../呢?  
`path1 = "../data/name_age.csv"  
path2 = "../data/score.csv"`  
是因为我们的代码文件在src文件夹下, ../代表的是上一级目录,



`csv_con.py`要调用`data`下的文件就必须使用`../`来访问,  
当然也可以使用绝对路径  
# 绝对路径  
`path1 = r"C:\Users\justin\PycharmProjects\baselearn\data\name_age.csv"  
path2 = r"C:\Users\justin\PycharmProjects\baselearn\data\score.csv"`  
但是在使用绝对路径的时候一定要加上r, 因为在python中有很多格式化输入与输出所用的特殊字符,  
比如\n, 用于换行输出,



不在前面加r就会在代码中显示出特殊字符

```
# 绝对路径
path1 = "C:\Users\justin\PycharmProjects\baselearn\data\name_age.csv"
path2 = "C:\Users\justin\PycharmProjects\baselearn\data\score.csv"
```

添加了r便不会

```
# 绝对路径
path1 = r"C:\Users\justin\PycharmProjects\baselearn\data\name_age.csv"
path2 = r"C:\Users\justin\PycharmProjects\baselearn\data\score.csv"
```

通常我比较喜欢使用相对路径，  
相对路劲较短，且检索项目文件比较好检索

ok 相信你已经运行成功，输出了最后的merged\_file.csv

name\_age.csv

```
csv_con.py      name_age.csv ×      score.csv      merged_file.csv
1  name,age
2  jiojio,20
3  justin,18
```

score.csv

```
csv_con.py      name_age.csv      score.csv ×      merged_file.csv
1  score_db,score_pe
2  100,98
3  100,98
```

合并后的merged\_file.csv

```
csv_con.py      name_age.csv      score.csv      merged_file.csv ×
1  name,age,score_db,score_pe
2  jiojio,20,100,98
3  justin,18,100,98
```

一般情况下，我们完整的项目的代码文件只需要一个文件去驱动，你会在main.py/run.py/app.py这些文件中看到很多import

此时我们也尝试将我们刚写的代码变成一个随处可调用的函数，将代码改成如下即可  
import pandas as pd

```
def con_csv(file1, file2):
    # 读取两个CSV文件
    df1 = pd.read_csv(file1)
    df2 = pd.read_csv(file2)
```

```

# 使用concat函数合并两个数据框
merged_df = pd.concat([df1, df2], axis=1)

# 将合并后的数据框保存为新的CSV文件
merged_df.to_csv('merged_file.csv', index=False)

```

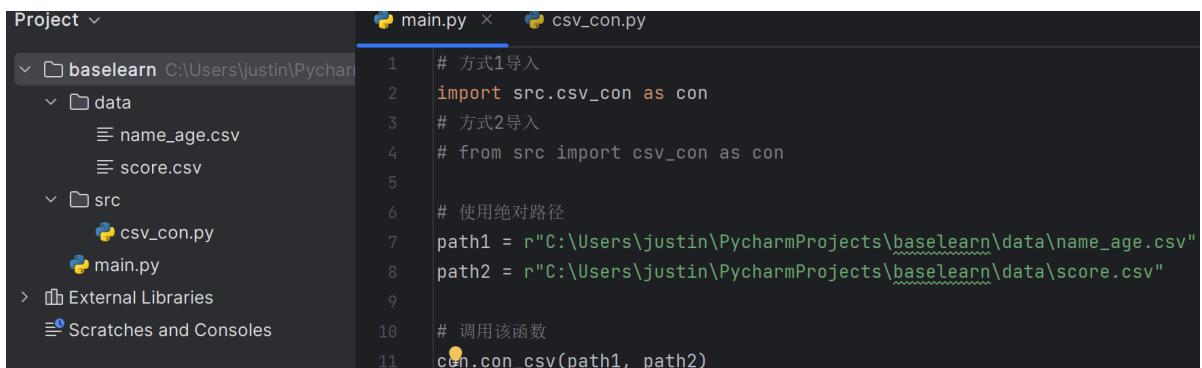
我们在我们的**main.py**中，调用该函数，导入**src**中的**csv\_con.py**模块，并重命名为**con**。此时我们就可以使用**con**来调用**csv\_con.py**中的所有函数。在我们**csv\_con.py**中只有一个函数，**con\_csv**，这个时候我们便可使用**con.con\_csv()**来调用该函数。

```

import src.csv_con as con

# 绝对路径
path1 = r"C:\Users\justin\PycharmProjects\baselearn\data\name_age.csv"
path2 = r"C:\Users\justin\PycharmProjects\baselearn\data\score.csv"
💡
con.con_csv(path1, path2)

```



运行**main.py**，你便可得到一个**CSV**文件于**main**在同一目录下。或者你也可以自行定义存放的输出文件的位置，比如我们在项目文件夹下创建一个**output**文件夹，并将**csv\_con.py**中最后一行中的，改为如下：`merged_df.to_csv('output/merged_file.csv', index=False)`

```

import pandas as pd

def con_csv(file1, file2):
    # 读取两个CSV文件
    df1 = pd.read_csv(file1)
    df2 = pd.read_csv(file2)

    # 使用concat函数合并两个数据框
    merged_df = pd.concat([df1, df2], axis=1)

    # 将合并后的数据框保存为新的CSV文件
    merged_df.to_csv('output/merged_file.csv', index=False)

```

此时我们再次运行**main.py**，就可以看到我们的输出的文件在**output**文件夹下。

```

1 # 方式1导入
2 import src.csv_con as con
3 # 方式2导入
4 # from src import csv_con as con
5
6 # 使用绝对路径
7 path1 = r"C:\Users\justin\PycharmProjects\baseLearn\data\name_age.csv"
8 path2 = r"C:\Users\justin\PycharmProjects\baseLearn\data\score.csv"
9
10 # 调用该函数
11 con.con_csv(path1, path2)

```

OK, 此时你已经学会了基础的**python**项目的创建, 配置, 管理, 也许在实际开发中还需要添加更多的其他的东西

比如项目的说明文档, 依赖需求, 添加更多的测试数据, 还有**docker**(我也还在学习中！！！).....等等,

很多时候我们不是做项目, 可能只是学习, 或者做一个简单东西不需要很复杂, 我们确实没有必要创建这么多东西,

通常一个项目文件夹下, 有几个**py**文件, 然后如果要进行数据操作, 甚至不创建**data**文件夹, 直接使用数据的文件名访问数据

为了方便也都是可以的, 但是如果你要做相对大型一点的项目, 此时我认为就应该有一个比较整洁的项目结构, 如第四点刚开始所展示的一样。

而且创建项目的文件和文件夹的命名也没有确定的格式或者要求,

但是最好是简洁, 明了, 易懂, 这三点不仅是针对自己, 同时也针对可能会访问到你的项目的人, 比如工作中同组的开发人员之类的。

lifeiteng Merge pull request <a href="#">lifeiteng#157</a> from guoao/guoao-patch-1 ...		
	a1b44ca on Aug 8	238 commits
📁 .github	update README.md	2 months ago
📁 docker	Create Dockerfile	last month
📁 docs	Update README.md	5 months ago
📁 egs	Add a tip and more detailed help in the script	2 months ago
📁 valle	[bin/infer.py] simplify model config	2 months ago
📄 .flake8	Import Reworked Conformer Tricks	5 months ago
📄 .gitignore	configure Fbank audio extractor	7 months ago
📄 .pre-commit-config.yaml	Import Reworked Conformer Tricks	5 months ago
📄 LICENSE	Initial commit	8 months ago
📄 README.md	update README.md	2 months ago
📄 README.zh-CN.md	Update README.md	5 months ago
➡️ examples	Update README.md	5 months ago
📄 setup.py	version 1.0.0	5 months ago
📄 test.sh	update models	5 months ago

最后, 我将向你展示如何去使用**git**管理自己的项目, 并且上传到自己的**github**

首先得安装**git**, 我的电脑已经安装好了, 现在我卸载重装一次。

```
Σ Windows PowerShell × + ▾  
PS C:\Users\justin> git --version  
git version 2.41.0.windows.3  
PS C:\Users\justin> |
```

1. 安装git

git官网[git](#)

点击download

The screenshot shows the official Git website. The top navigation bar includes links for 'About', 'Documentation' (which is currently selected), 'Chapters' (2nd Edition), 'Downloads', and 'Community'. The main content area features a large heading '1.5 起步 - 安装 Git' and a sub-section titled '安装 Git'. It explains that to start using Git, it needs to be installed on the computer, which can be done via a software package or by downloading the source code.

点击windows

The screenshot shows the 'Downloads' section of the Git website for the Windows platform. It offers three download options: 'macOS', 'Windows', and 'Linux/Unix'. Below these, a note states that older releases are available and the Git source repository is on GitHub. To the right, a graphic of a computer monitor displays the 'Latest source Release' information, showing '2.42.0' and a 'Release Notes (2023-08-21)' link, with a prominent 'Download for Windows' button.

选择Standalone installer中的64-bit，等待下载完成

# Other Git for Windows downloads

## Standalone Installer

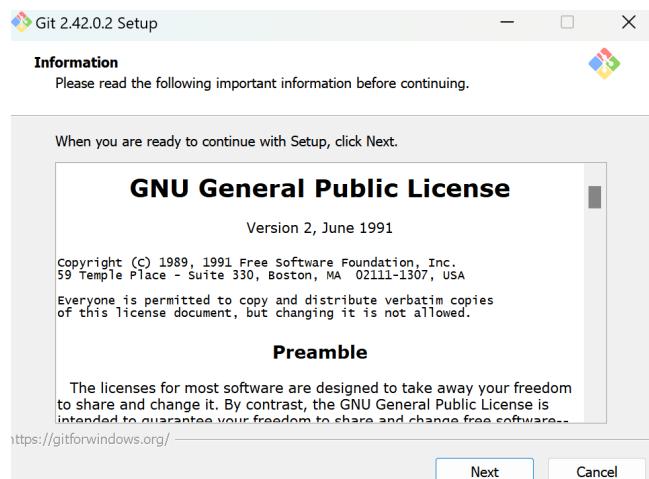
### 32-bit Git for Windows Setup.

### 64-bit Git for Windows Setup.

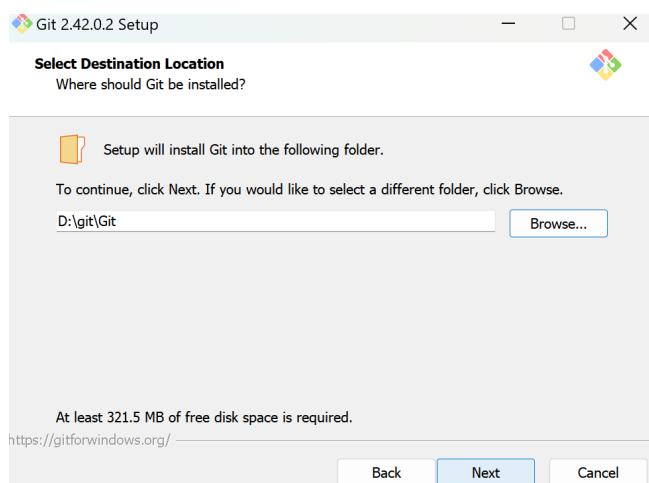
双击文件安装



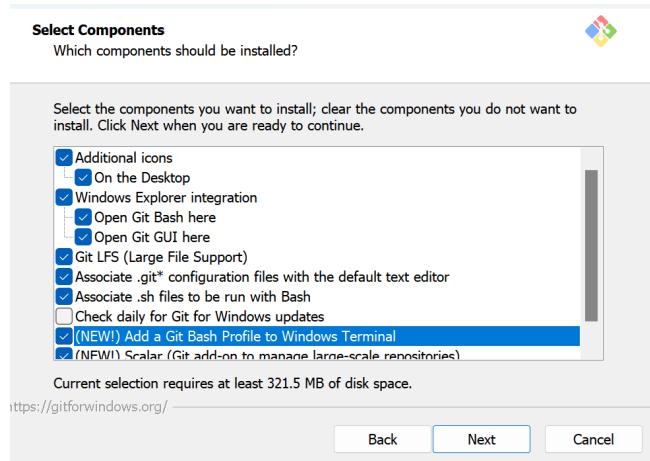
next



这个软件可以随意安装在一个位置



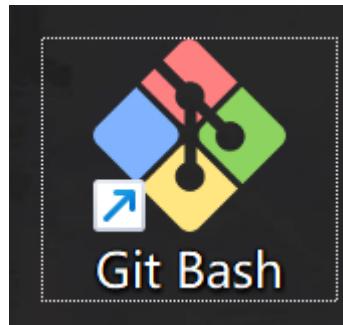
勾选情况如下



后面的全部默认即可，安装完成后，我们打开power shell，输入git --version，即可看到我们所安装的git版本

```
PS C:\Users\justin> git --version
git version 2.42.0.windows.2
```

同时它也会在桌面创建一个快捷方式，这power shell中使用git和git bash中使用git没有差别



ok,此时，我们再来介绍一下什么是git，git是一个版本管理工具，比如我们的项目，我们完成了一个功能，我们需要记录当前项目的状态，方便之后做了修改，想回到这个版本，或者其他用途，最为明显的应用场景就是你完成了某项目的某个功能，或者你即将做一个重大的修改，在修改前，你想保存一个当前项目的状态，方便做对比，或者害怕自己修改错误，此时git就派上了用场，我们可以是使用git 来创建一个版本，比如1.0 ， 使用git commit -m "version1.0"

创建版本之后，你可以随时回到这个版本，这个版本的项目状态就是你创建的时候的状态

git的使用很简单，但是我得先教你如何将其和github配合起来使用

github是全球最大的开源代码社区，你能在上面找到很多有意思的东西，比如什么AI相关的啊乱七八糟的什么都有。

注册一个github账号，可以使用qq邮箱，也可使用其他的邮箱，创建用户名的时候尽量简短一点

创建好github账号之后，我们就要配置git和github的链接,首先配置用户名和邮箱

#打开git bash，分别输入如下内容，

```
#配置用户名  
git config --global user.name "test"  
#配置邮箱  
git config --global user.email abc@163.com
```

比如我的就是

```
git config --global user.name "Justin-12138"  
git config --global user.email lz12138@mails.guet.edu.cn
```

最后，使用

```
git config --global --list  
查看你的配置信息
```

```
MINGW64:/c/Users/justin  
justin@Justin MINGW64 ~  
$ git config --global user.email lz12138@mails.guet.edu.cn  
  
justin@Justin MINGW64 ~  
$ git config --global --list  
user.name=Justin-12138  
user.email=lz12138@mails.guet.edu.cn
```

之后生成ssh密钥，在git bash中输入如下内容，

```
ssh-keygen -t rsa
```

第一次询问你密钥的保存位置，默认即可，回车

```
justin@Justin MINGW64 ~  
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/justin/.ssh/id_rsa):
```

默认回车

```
justin@Justin MINGW64 ~  
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/justin/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):
```

再次回车

```
justin@Justin MINGW64 ~  
$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/justin/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:
```

此时可以看到创建成功，使用ctrl+L清屏

```

Generating public/private rsa key pair.
Enter file in which to save the key (/c/users/justin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/users/justin/.ssh/id_rsa
Your public key has been saved in /c/users/justin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/2gpiimb+vmq2PezSiyuz8M5+YZ+R5KdRaov4couzi4 justin@Justin
The key's randomart image is:
+---[RSA 3072]----+
| . |
|   o |
| + .. |
| + .S + o |
| E o ... + = |
| o +.= O |
| . oo#*B.* |
| ooXO%@Oo. |
+---[SHA256]----+

```

```

# 进入.ssh目录 在git bash中输入
cd .ssh
# 列出.ssh文件夹下的目录
ls
# 查看id_rsa.pub的文件内容，然后复制
cat id_rsa.pub

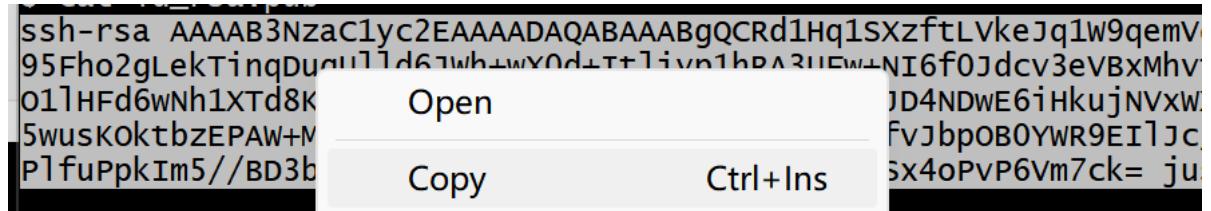
```

```

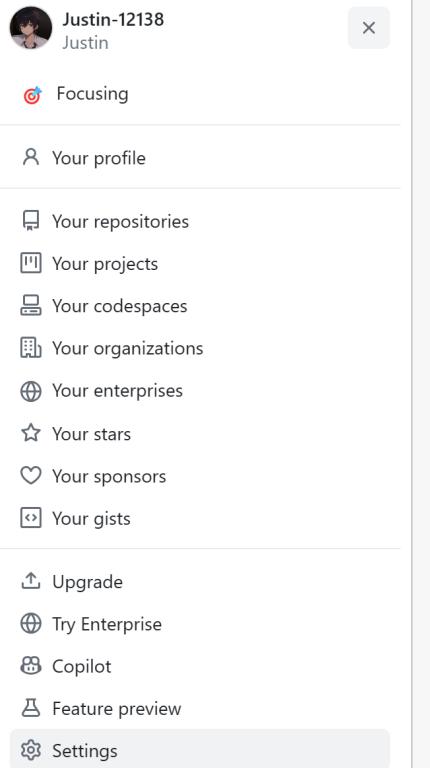
justin@justin MINGW64 ~
$ cd .ssh
justin@justin MINGW64 ~/ssh
$ ls
codespaces.jetbrains codespaces.jetbrains.pub  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
justin@justin MINGW64 ~/ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAABgQCRd1Hq1SXzftLVkeJq1w9qemVdnMpkhnwT4SA3dd5uqyIFcj99dlo106y1chotfpUMr98UDZLZTnof
95Fho2gLeiTinqDugU1ld6jwh+wX0d+It1iyplhRA3UFw+NI6f0Jdcv3eVBxMhv0TGwbv5mhhdIn9xViUHVkcJnqvD6voMzVkbxeJKH2QkBQcs3+Sa
o1HFd6Nh1XTd8kce7uJrcF8eRwxAQy0JPO7JVNNeJNHJD4NDwE6iHkujNVxwXiNPL3f3wbb0FodAMSA/GptEj0ysvwulj5rjt9w4gwaeazp1AfF3PN
5wusKoktbzEPaw+M9CEsPEfF7j4rmc/ShE8dB8QiPef1j6fvJbpob0YWR9EI1jc/901tRlwbhEqa0mP9g+uux21WTjMG28D+LpbclVHNQP40TyXHIs>
P1fuPpkIm5//BD3bpoCBpzruhAhdxLeFlieF1p3ynufysx4oPvP6Vm7ck= justin@justin

```

然后使用鼠标选中文件内容，右键并复制



之后打开github，点击你的头像，并选择settings



在SSH and GPG keys中选择点击New SSH key

A screenshot of the GitHub 'SSH keys' management page for the user 'Justin (Justin-12138)'. The left sidebar includes links for 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access' (with 'Billing and plans' expanded), 'Emails', 'Password and authentication', 'Sessions', and 'SSH and GPG keys', which is highlighted with a grey background. The main content area is titled 'SSH keys' and contains a sub-section 'Authentication Keys'. It lists two existing SSH keys: 'windows' (added on Aug 17, 2023, last used within the last week, Read/write) and 'ubuntu' (added on Sep 9, 2023, last used within the last week, Read/write). A green 'New SSH key' button is located at the top right of the 'SSH keys' section.

title随便取什么都可，然后将你刚复制的ssh key粘贴到key，点击add ssh key

## Add new SSH Key

Title

windows-ssh

Key type

Authentication Key

Key

ssh-rsa

```
AAAAAB3NzaC1yc2EAAAQABAAQgQCQd1Hq1SXzftLkq1W9qemVdnMpkhnWT4SA3dd5uqyIFcj99dlo106y1chOtf
pUMr98UDZLztNoFl7JMgbV95Fho2gLeiTinDugUlld6JWh+wX0d+Itliy1hRA3UFw+NI6f0Jdcv3eVBxMhvt0TGWbv5mHh
dln9xVlUHVKcJnqVd6voMzVkbxeJKH2QkBQcS3+SaGhbQvAVIO1IHfd6wNh1XTd8KcElv7ujrcF8eRwxAQy0JPO7JVNeJNHJ
D4NDwE6iHkujNVxWxiNPL3fbwbbOFQdAMSA/GptEj0ysvWulj5rjT9w4gWeaZp1AfF3PMHQ/T2d95wusKOktbzEPaw+M9
CEsPEfF7j4rmc/ShE8dBBQiPef1j6fvJbpOB0YWR9EIIj/c/9o1tRlwbhEQaOmp9g+Uux21WTjMG28D+LpbclVHNQP40xTyXHls
x88C0pcnPlfuPpkIm5//BD3bpoCZBpzsUhAHdXLeFIleFip3ynNufySx4oPvP6Vm7ck= justin@Justin
```

Add SSH key

然后输入密码确认即可



Confirm access

 Signed in as @Justin-12138

Password

.....  
[Forgot password?](#)

Confirm

然后我们就可以看到我们刚刚创建的ssh key了，这个ssh key 是用来我们方便拉取，并将代码上传到github的

它不同于其他的链接ssh链接更加稳定和安全，具体的传输原理你可以上网搜索一下，ok，到这里我们的配置算是完成了，

接下来就是实际应用了



windows-ssh

SHA256:/2gp1mb+Vmq2PezSyiuLz8M5+Yz+R5KdRaov4CoUZi4

SSH

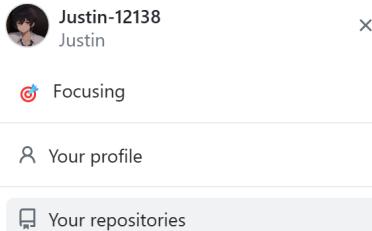
Added on Sep 30, 2023

Never used — Read/write

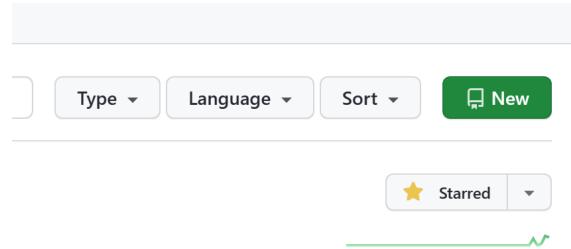
Delete

首先创建一个新仓库

点击your repositories



点击右上角的new



第一个是项目名称，精良也还是相对简短一点

**description**是项目的描述，

**public or private**就是公开或者私有，公开代表大家都可以看，但是**private**就只有你自己可以和你授权的人可以看

**add a readme.md**代表的是创建一个项目说明文档

**add.gitignore**，代表的是，在我们本地的项目中，会创建一些不必要的配置文件，我们就没有必要上传到github，

我们选择**python**代表的是我们这个项目大概率是**python**主要代码，然后**python**的一些无需上传的配置文件会帮我们自动忽略掉

**choose a license** 代表的是选择开源协议，这个开源协议还有点复杂，不同的开源协议有不同的要求，

**MIT**协议大概是你可以用我的这个项目去商用啊什么都可以，

当然也有一些严苛的开源协议，比如你可以商用，但是比必须开源，等等

最后点击**create repository**

Owner \*  Justin-12138 / Repository name \* testrepo  testrepo is available.

Great repository names are short and memorable. Need inspiration? How about [friendly-succotash](#) ?

Description (optional) firstpro

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore .gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

**Create repository**

创建完成后，我们的项目就如下图所示，现在我们要把刚刚的那个合并csv的文件，作为一个项目上传到github的这个testrepo仓库中

 **testrepo** 

 main  1 branch  0 tags   

 Justin-12138	Initial commit	2fb024b now	 1 commit
 .gitignore	Initial commit	now	
 LICENSE	Initial commit	now	
 README.md	Initial commit	now	

**README.md** 

## testrepo

firstpro

第一步克隆仓库到本地，点击code，选择ssh，点击右边的复制即可

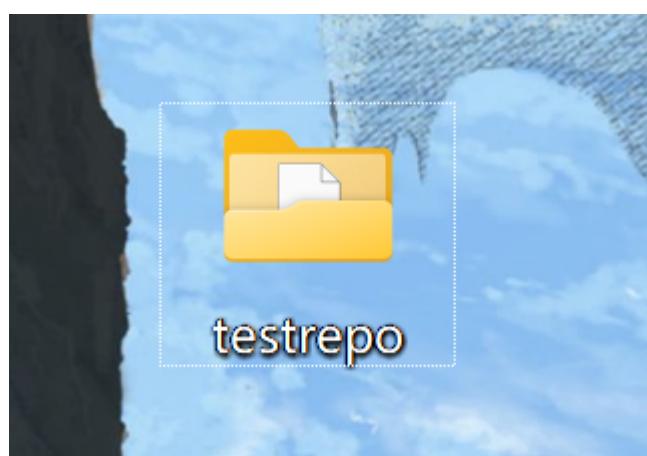


我们在桌面中右键，选择在终端中打开  
并在终端中输入  
`git clone` 我们刚复制的仓库地址  
比如，我的就是  
`git clone git@github.com:Justin-12138/testrepo.git`



回车并等待克隆完成，同时也可在桌面中看到我们多了一个**testrepo**的文件夹

```
PS C:\Users\justin\Desktop> git clone git@github.com:Justin-12138/testrepo.git
Cloning into 'testrepo' ...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
PS C:\Users\justin\Desktop> |
```



点击进入，可以看到现在这个仓库中除了多了一个`.git`，其他的文件都是我们创建仓库的时候所自带的

名称	修改日期	类型
.git	2023/9/30 13:08	文件夹
.gitignore	2023/9/30 13:08	txtfile
LICENSE	2023/9/30 13:08	文件
README.md	2023/9/30 13:08	Markdown

此时，我们便可以将之前的**python**的项目文件复制到该文件夹中  
并在终端中使用 `git status` 查看仓库状态  
可以看到它提示我们标红的文件和文件夹未追踪（**untracked**）  
这里就涉及到一个之前没讲到东西就是**git**如何进行版本管理  
我这里就稍微提供一下简单的教程，其实**git**还有很多功能  
比如我们现在向项目中添加，删除，更改了一些文件或者文件夹  
我们想更好的管理他们都需要把他们放入暂存区  
使用`git add *` 代表将所有文件都放入暂存区 你也可以使用`git add` 某文件或者某文件夹 将某文件或者某文件夹放入暂存区，  
又或者说追踪这些文件或者文件夹，也许你看到这里有点懵，但是只要记住，你想使用**git**更好的去管理你的项目，管理版本  
一般的步骤就是  
修改某个文件，之后，使用  
`git add` 某个文件  
  
创建版本  
`git commit -m "我修改了这个文件的某个地方"`  
  
`git push` 上传到**github**  
然后使用`git log`可以查看版本记录  
接下来，我们就基于我们当前的项目来做一次演示

Windows PowerShell  
版权所有 (C) Microsoft Corporation。保留所有权利。  
安装最新的 PowerShell，了解新功能和改进！<https://aka.ms/PSWindows>

```
PS C:\Users\justin\Desktop\testrepo> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    data/
    main.py
    output/
    src/

nothing added to commit but untracked files present (use "git add" to track)
```

```
git add *
git status #再次查看git状态
git commit -m "第一次提交" #创建一个版本
git log #查看git log版本日志
```

```
PS C:\Users\justin\Desktop\testrepo> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    data/
    main.py
    output/
    src/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\justin\Desktop\testrepo> git add *
PS C:\Users\justin\Desktop\testrepo> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file> ..." to unstage)
    new file:   data/name_age.csv
    new file:   data/score.csv
    new file:   main.py
    new file:   output/merged_file.csv
    new file:   src/csv_con.py

PS C:\Users\justin\Desktop\testrepo> git commit -m "第一次提交"
[main 8c981f9] 第一次提交
 5 files changed, 35 insertions(+)
 create mode 100644 data/name_age.csv
 create mode 100644 data/score.csv
 create mode 100644 main.py
 create mode 100644 output/merged_file.csv
 create mode 100644 src/csv_con.py
PS C:\Users\justin\Desktop\testrepo> git log
commit 8c981f944cdec8b5377f316b98d1f37990215bc5 (HEAD → main)
Author: Justin-12138 <lz12138@mails.guet.edu.cn>
Date:   Sat Sep 30 13:25:13 2023 +0800
```

第一次提交

```
git push #上传到github
```

```
PS C:\Users\justin\Desktop\testrepo> git log
commit 8c981f944cdec8b5377f316b98d1f37990215bc5 (HEAD → main)
Author: Justin-12138 <lz12138@mails.guet.edu.cn>
Date:   Sat Sep 30 13:25:13 2023 +0800

第一次提交

commit 2fb024b76384e0e83def3753955c7dbe294cce7e (origin/main, origin/HEAD)
Author: Justin <2094623381@qq.com>
Date:   Sat Sep 30 13:03:08 2023 +0800

  Initial commit
PS C:\Users\justin\Desktop\testrepo> git push
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 1.19 KiB | 1.19 MiB/s, done.
Total 10 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:Justin-12138/testrepo.git
  2fb024b..8c981f9  main → main
PS C:\Users\justin\Desktop\testrepo> |
```

```
刷新我们的github仓库，即可看到我们上传的文件
```



testrepo

Public

Pin

Unwatch 1

main ▾

1 branch

0 tags

Go to file

Add file ▾

Code ▾



Justin-12138 第一次提交

8c981f9 2 minutes ago

2 commits

data	第一次提交	2 minutes ago
output	第一次提交	2 minutes ago
src	第一次提交	2 minutes ago
.gitignore	Initial commit	25 minutes ago
LICENSE	Initial commit	25 minutes ago
README.md	Initial commit	25 minutes ago
main.py	第一次提交	2 minutes ago

README.md



## testrepo ↗

firstpro

ok

到这里也接近该教程的尾声了，本教程花费接近6小时完成，全程自己手打字完成，最后计数9000字有余，去掉其中的一些非手写字数应有5000字左右，我做该教程的初衷便是让更多没有什么基础的同学(eg:jiojio)能够尽快的入门一些基础知识

本教程还有很多不完善的地方，

markdown语言的基础教程，

常用Linux操作密令，

操作系统基础知识，

git详细教程，

python其他虚拟环境的创建方式

这些基础的知识在我的baselearn的仓库中有所体现，希望你能够从中学到一些东西，或者帮我完善这个教程！

最后如果觉得本教程不错帮忙点个star！！！