软件部署说明

V1.0

**一、本项目在windows上运行，项目启动前的必要准备**

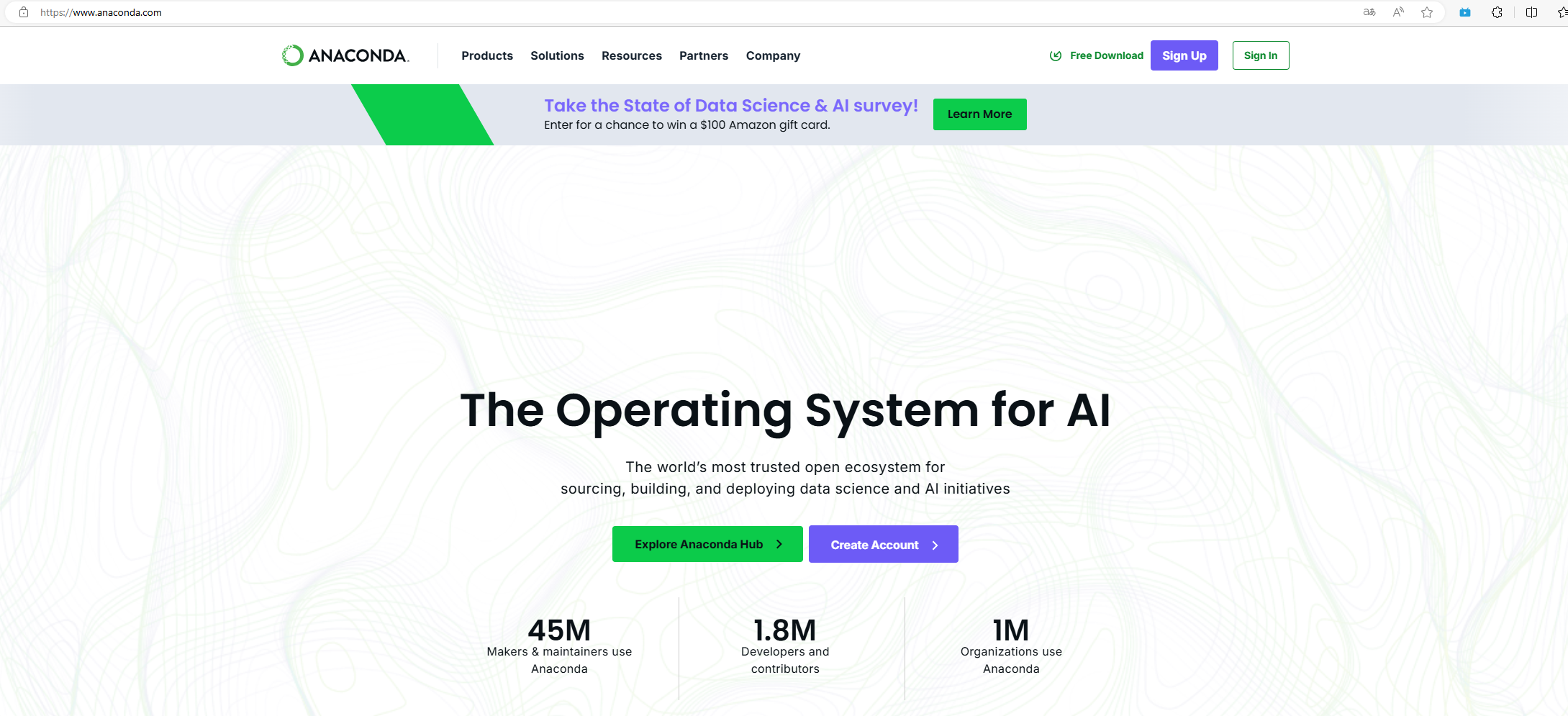
- 安装windows版anaconda 2023-09-0(64-bit)

- 安装windows版redis 5.0.14.1数据库

- 安装windows版ollama 0.1.44

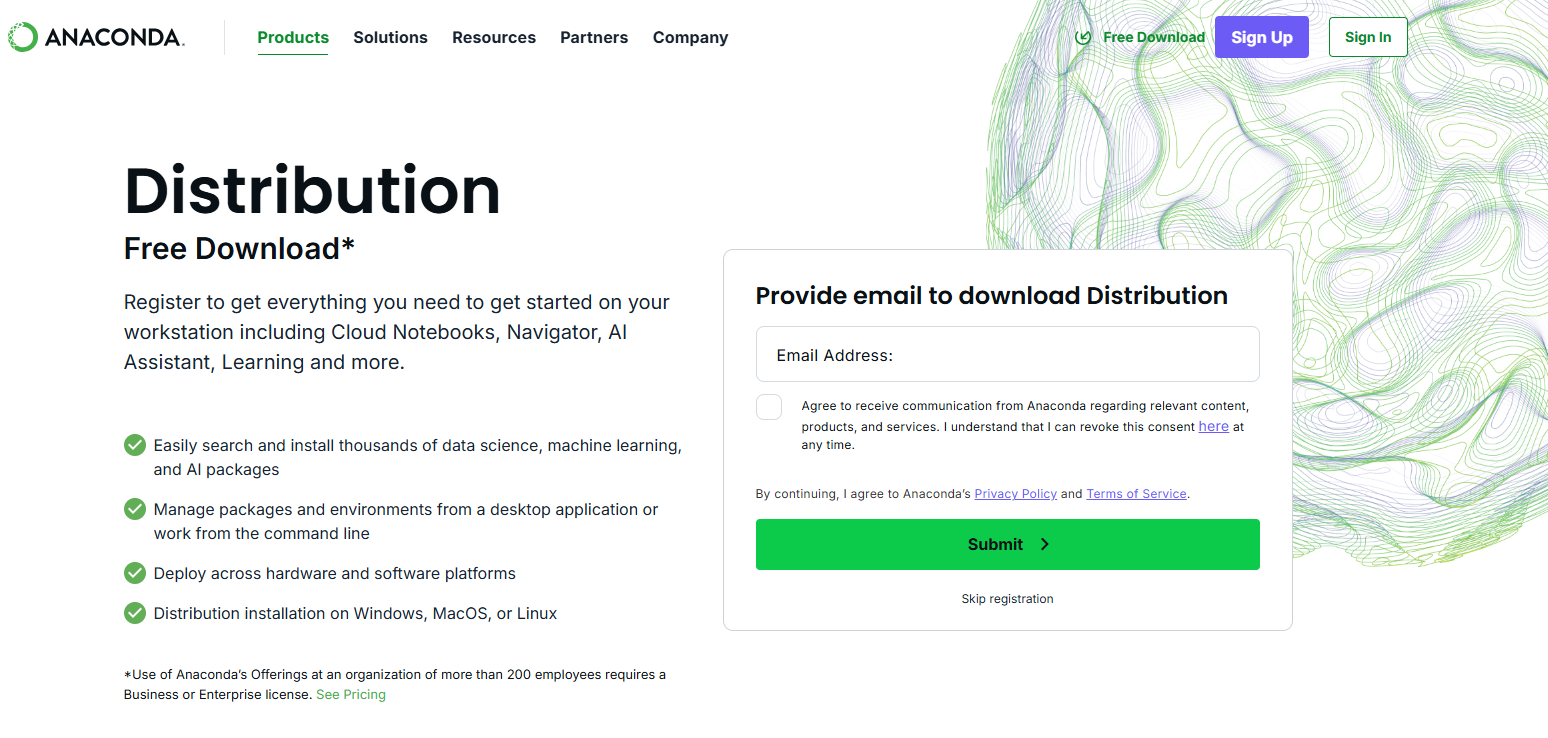
**1.1安装anaconda**

进入anaconda官网<https://www.anaconda.com/>



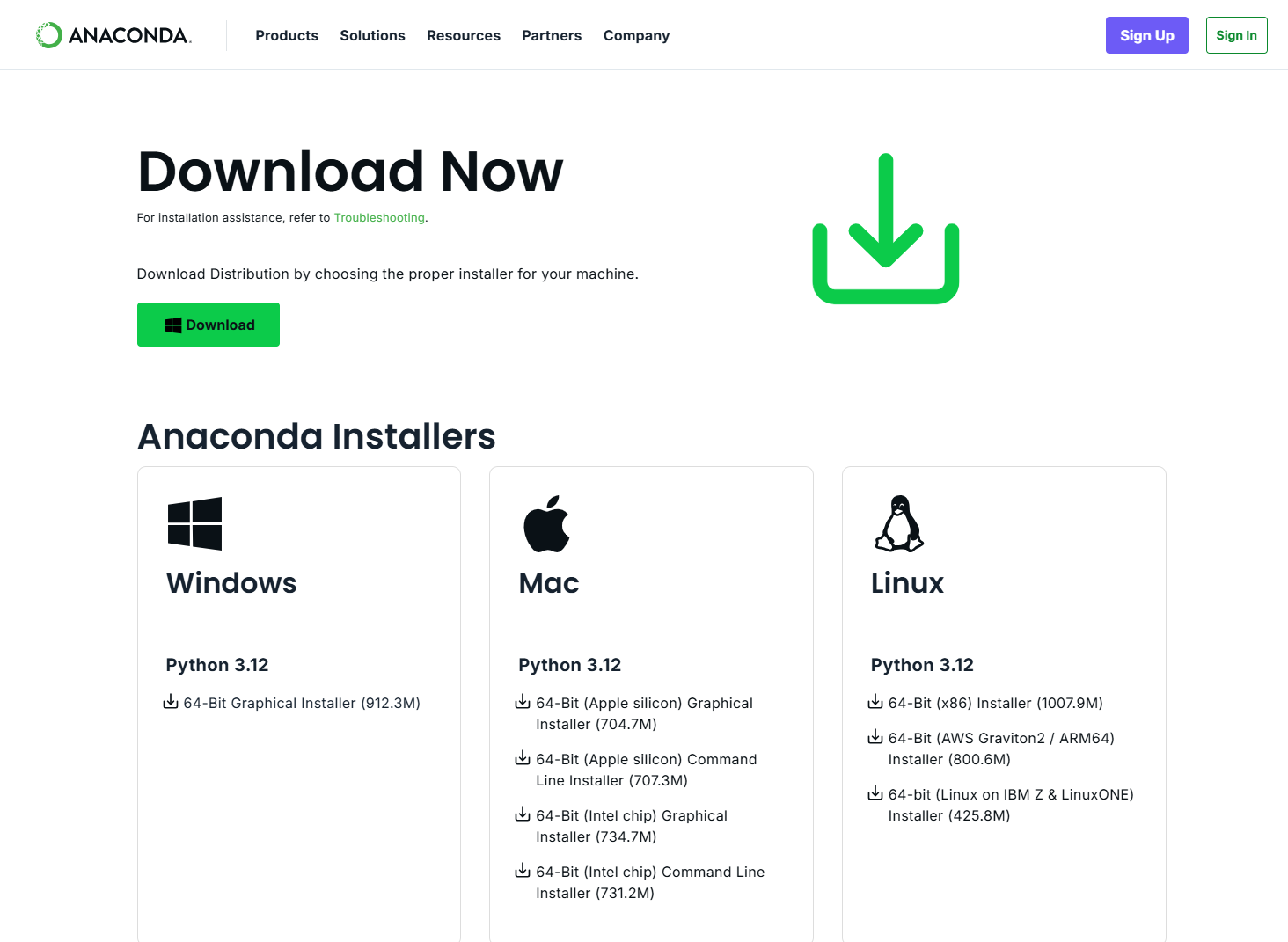
单击**Free Download**，选择**Skip registration，**跳过注册

1

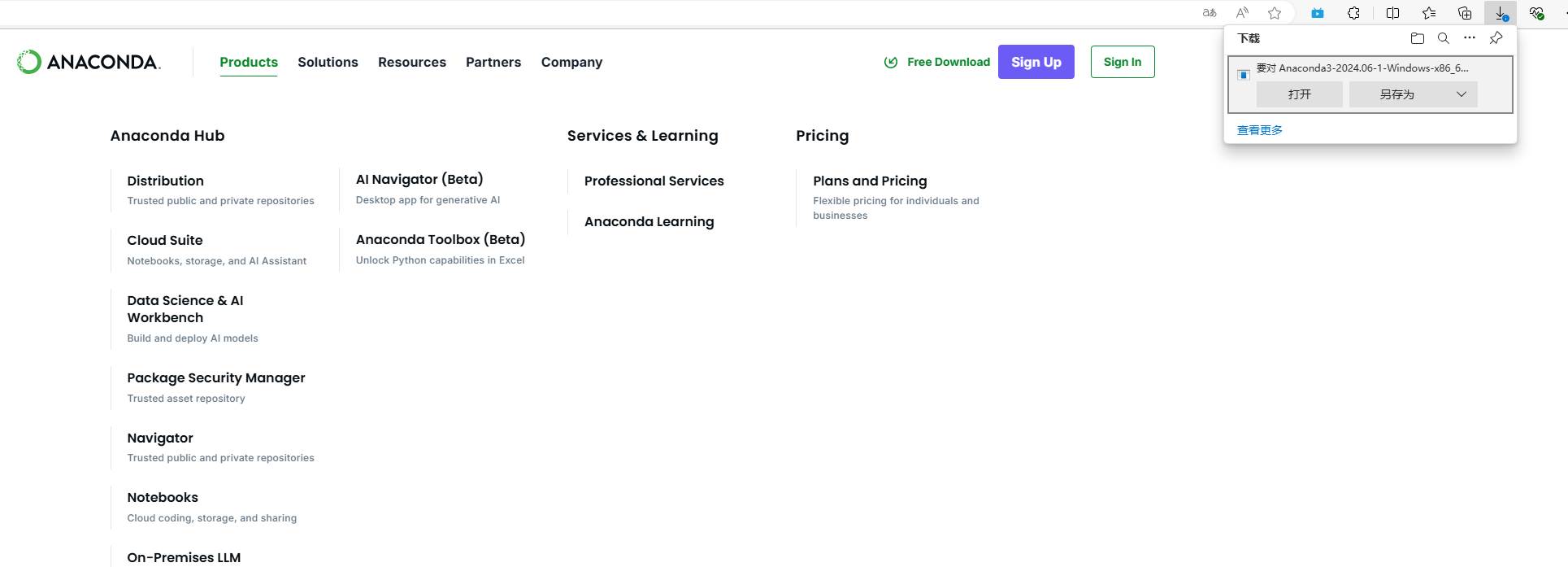


2

选择windows版本安装

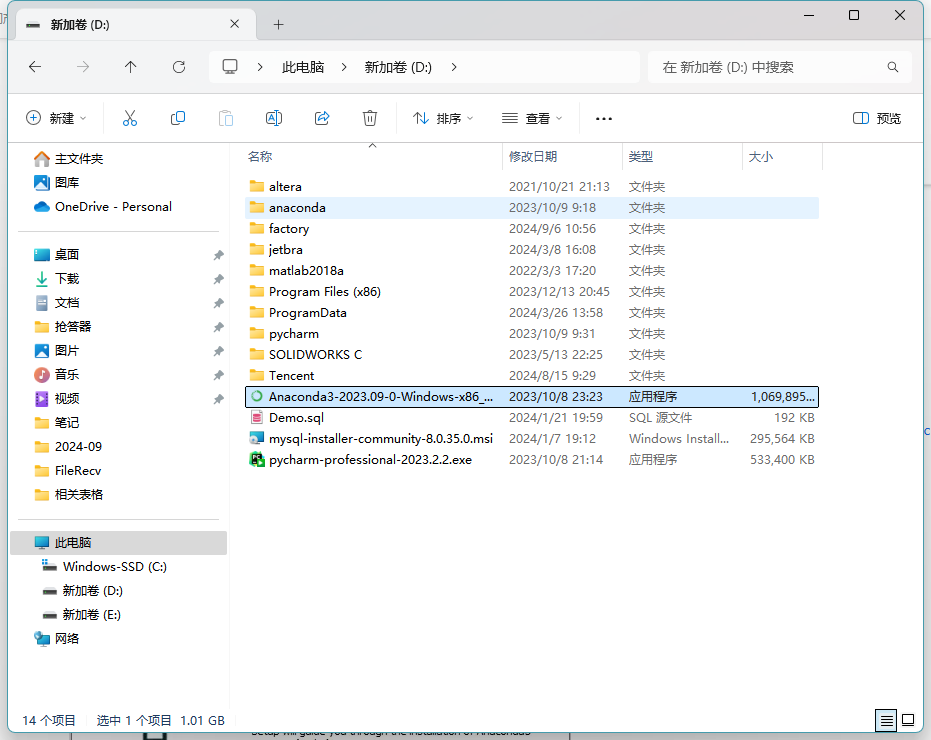


在弹出的对话框中，单击「**另存为**」选择下载到D盘根目录，并在D盘根目录位置进行anaconda的安装

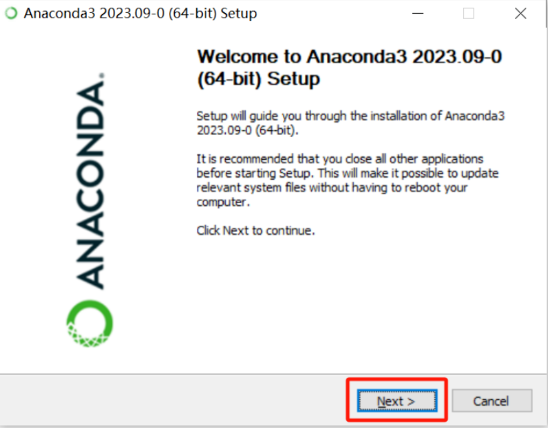


等待下载完成，下载好的安装程序如下图所示

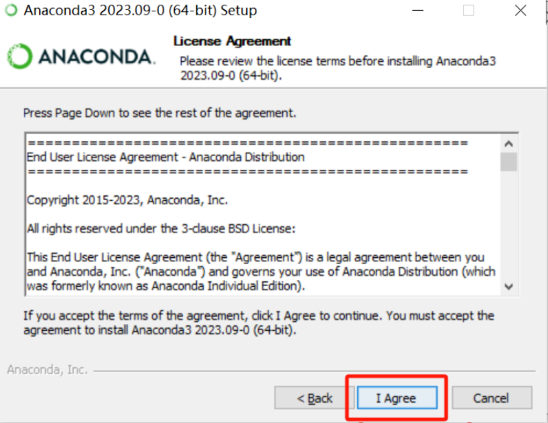
双击下载好的安装包文件，开始安装



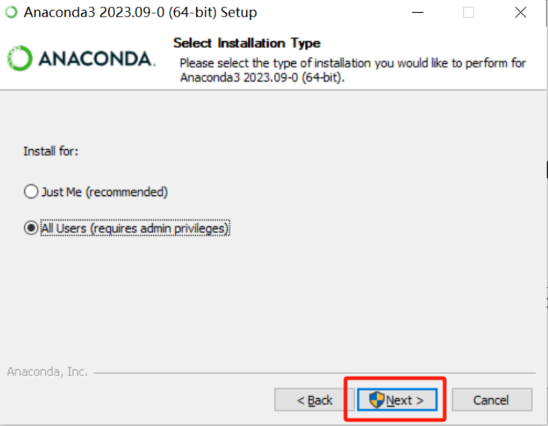
在弹出的对话框中单击「**Next >**」按钮



选择「**I Agree**」

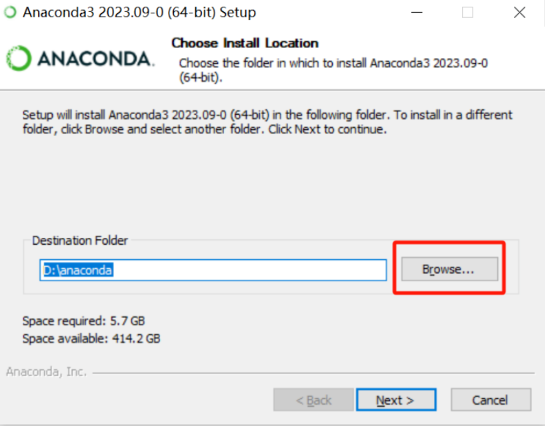


勾选**All Users(requires admin privileges)** 然后单击「**Next >**」进入下一步。

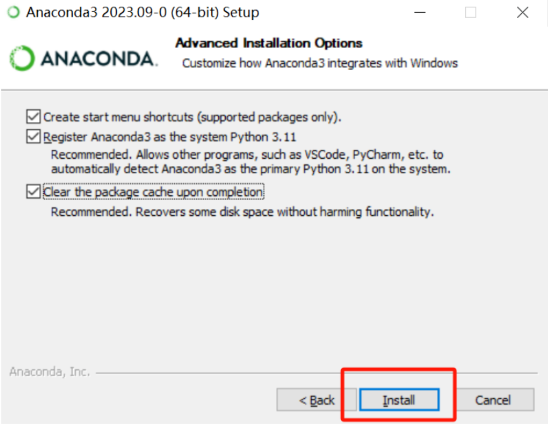


点击「**Browse…**」更改Anaconda安装路径。选择安装到D:\anaconda目录下。

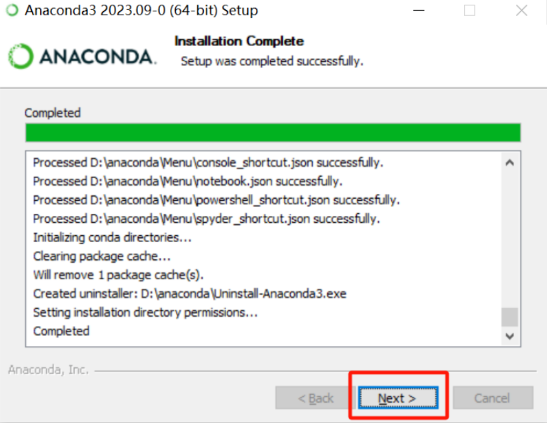
注：路径名称尽量不使用中文或者空格、特殊字符。



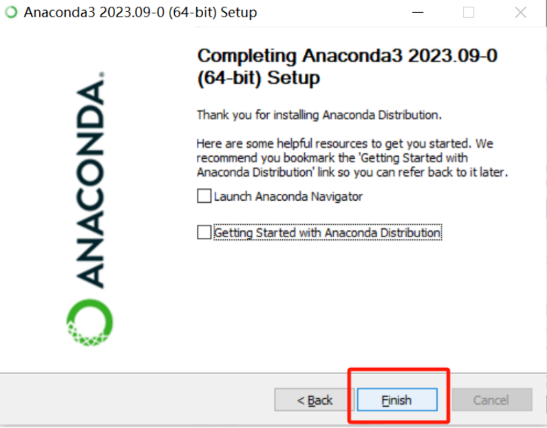
勾选所有内容后，点击「**Install**」安装



等待安装。之后一直「**Next**」。

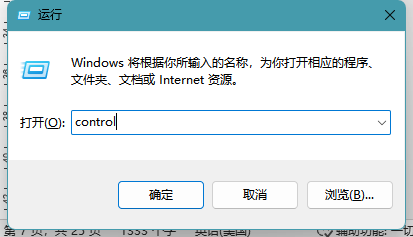


最后点击「**Finish**」完成安装。

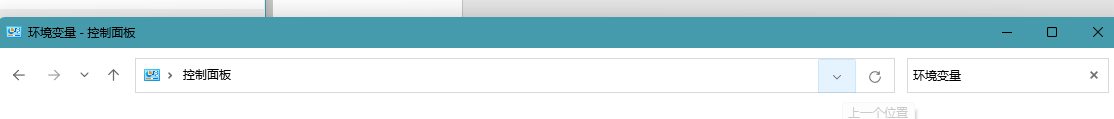


**1.2 为anaconda添加系统环境变量**

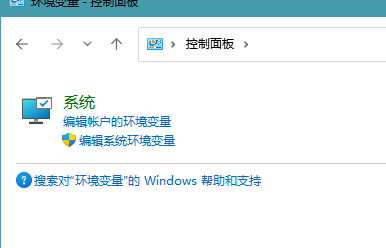
打开控制面板（键盘按住Win + R，在弹出的对话框中输入control） 在弹出的控制面板中搜索**环境变量**



1



找到系统 --- 然后单击「**编辑系统环境变量**」

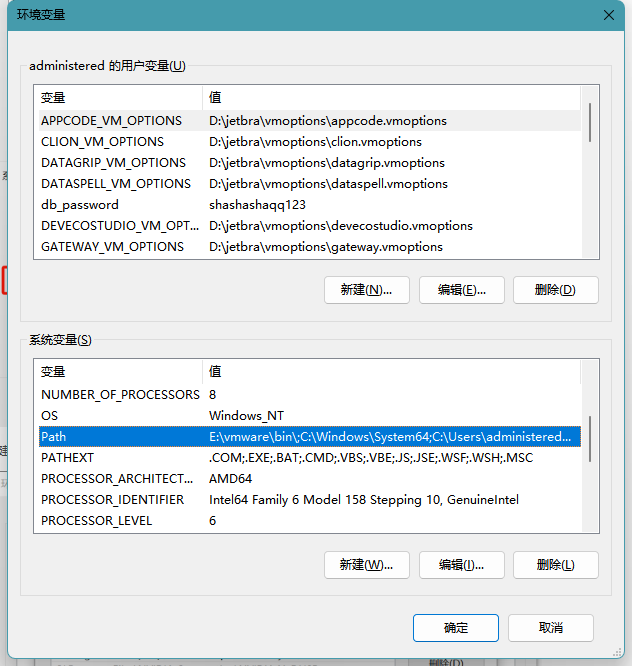


2

在弹出的系统属性对话框中点击「**环境变量(N)...** 」进行环境变量设置



在弹出的环境变量对话框中，双击点开**系统变量**中的「**Path**」



在弹出的编辑环境变量对话框中，点击「**新建(N)** 」创建4个空白的环境变量，并依次粘贴以下四个路径

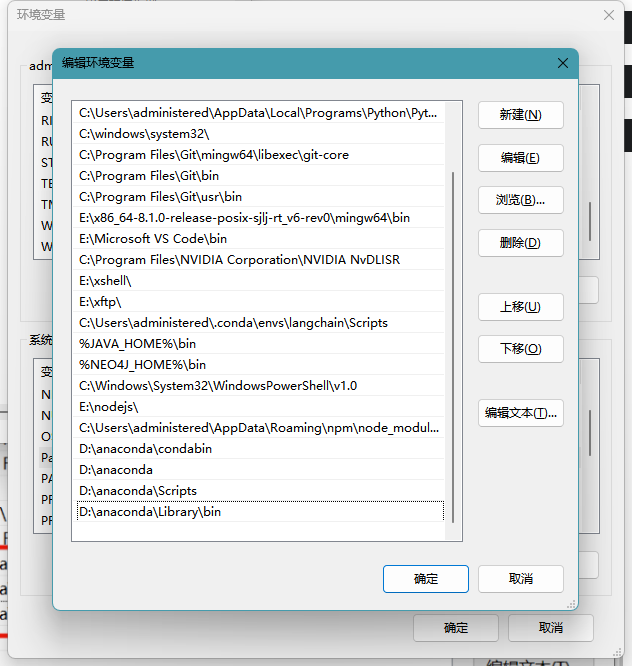
D:\anaconda\condabin，

D:\anaconda

D:\anaconda\Scripts

D:\anaconda\Library\bin

然后单击「**确定**」



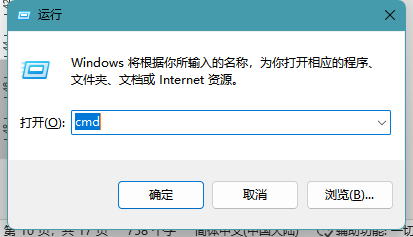
3

1

2

**1.3测试Anaconda是否安装成功**

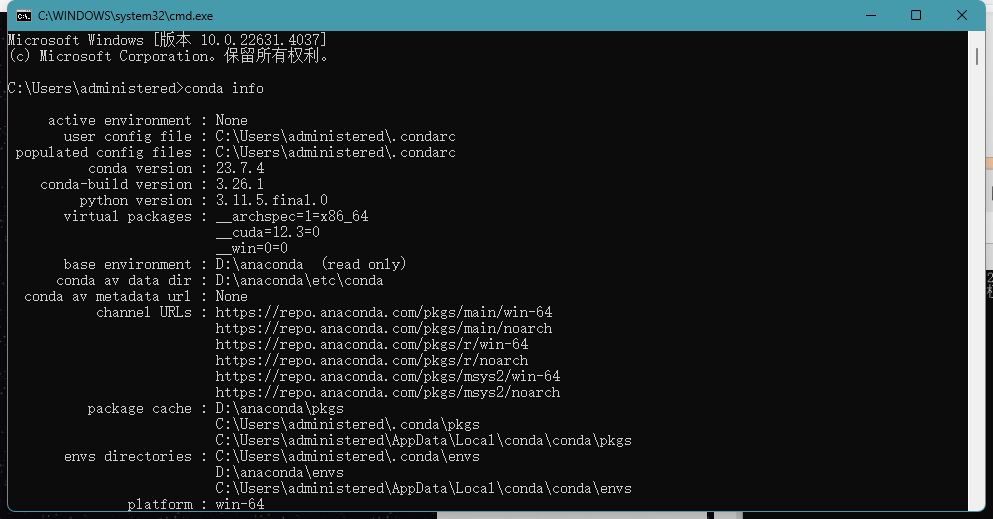
键盘上按住**Win+R**，在弹出的对话框中输入cmd，单击「**确定**」按钮



2

1

在弹出的终端窗口输入conda info命令，出现如下信息，说明环境配置成功，anaconda配置完成。



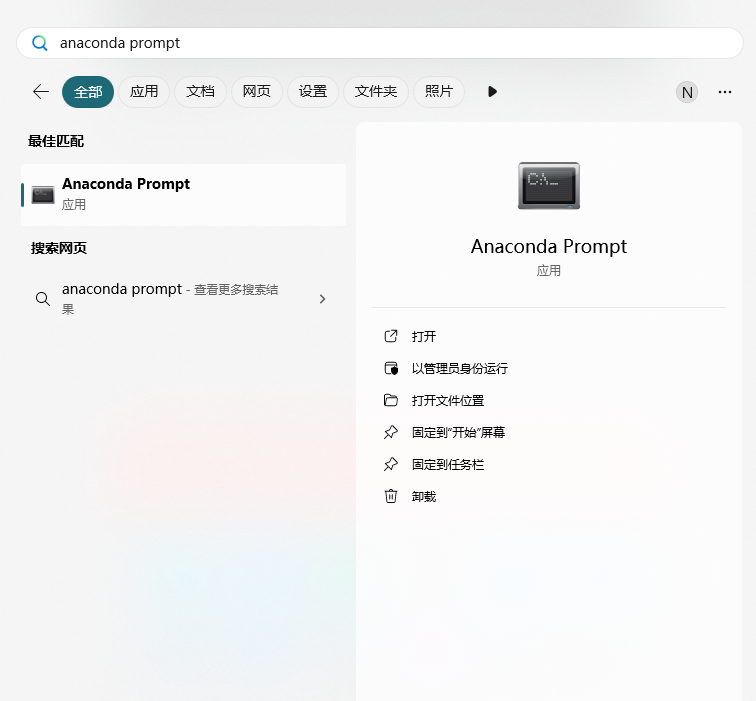
**1.4创建虚拟环境**

先将下载的HITL\_Project.zip解压到桌面

键盘上单击Win键，在弹出的菜单中搜索anaconda prompt



单击Anaconda Prompt应用



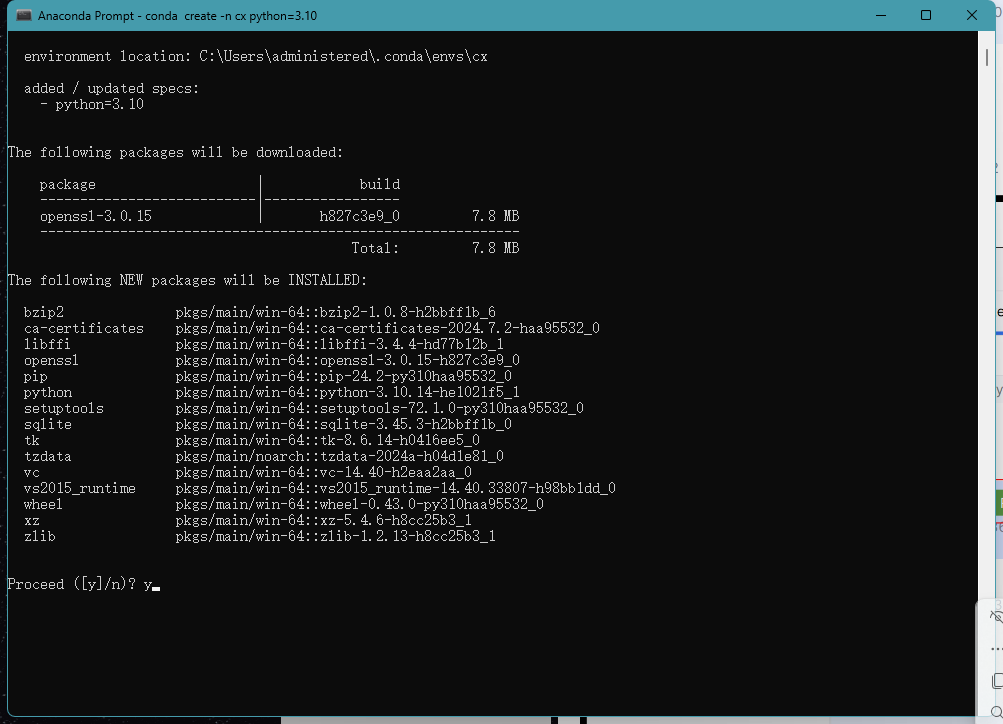
1

2

弹出Anaconda Prompt窗口中，在窗口中输入conda create -n au python=3.10命令，创建au虚拟环境



输入y



创建完成后，在窗口中继续输入conda activate au命令，进入au虚拟环境，此时前面的base环境变成了au环境



然后在窗口中继续输入

cd C:\Users\administered\Desktop\HITL\_Project命令进入到HITL\_Project文件夹下

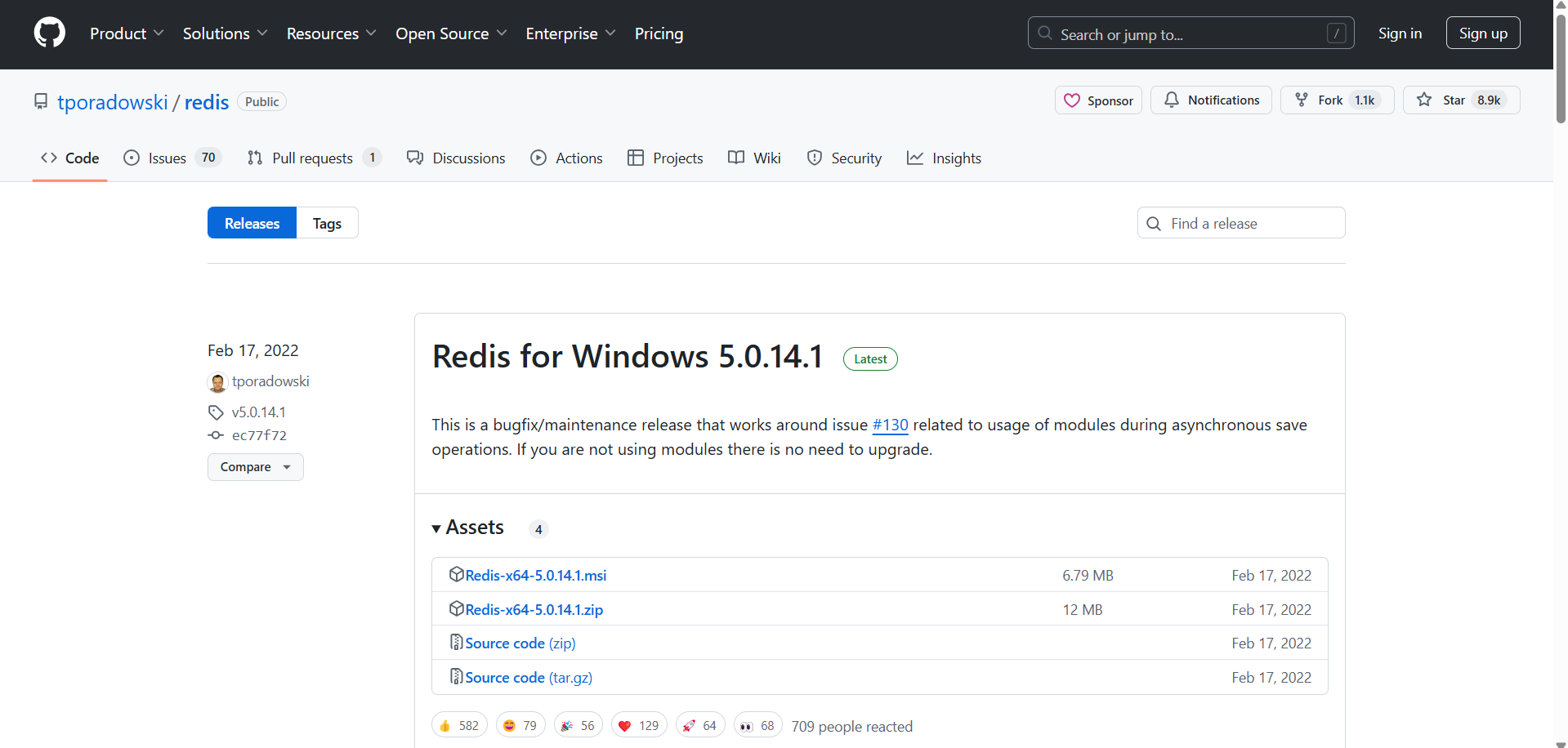
在窗口中继续输入 pip install -r requirements.txt命令安装项目的依赖包，等待所有安装包安装完成，到此au虚拟环境创建完成。



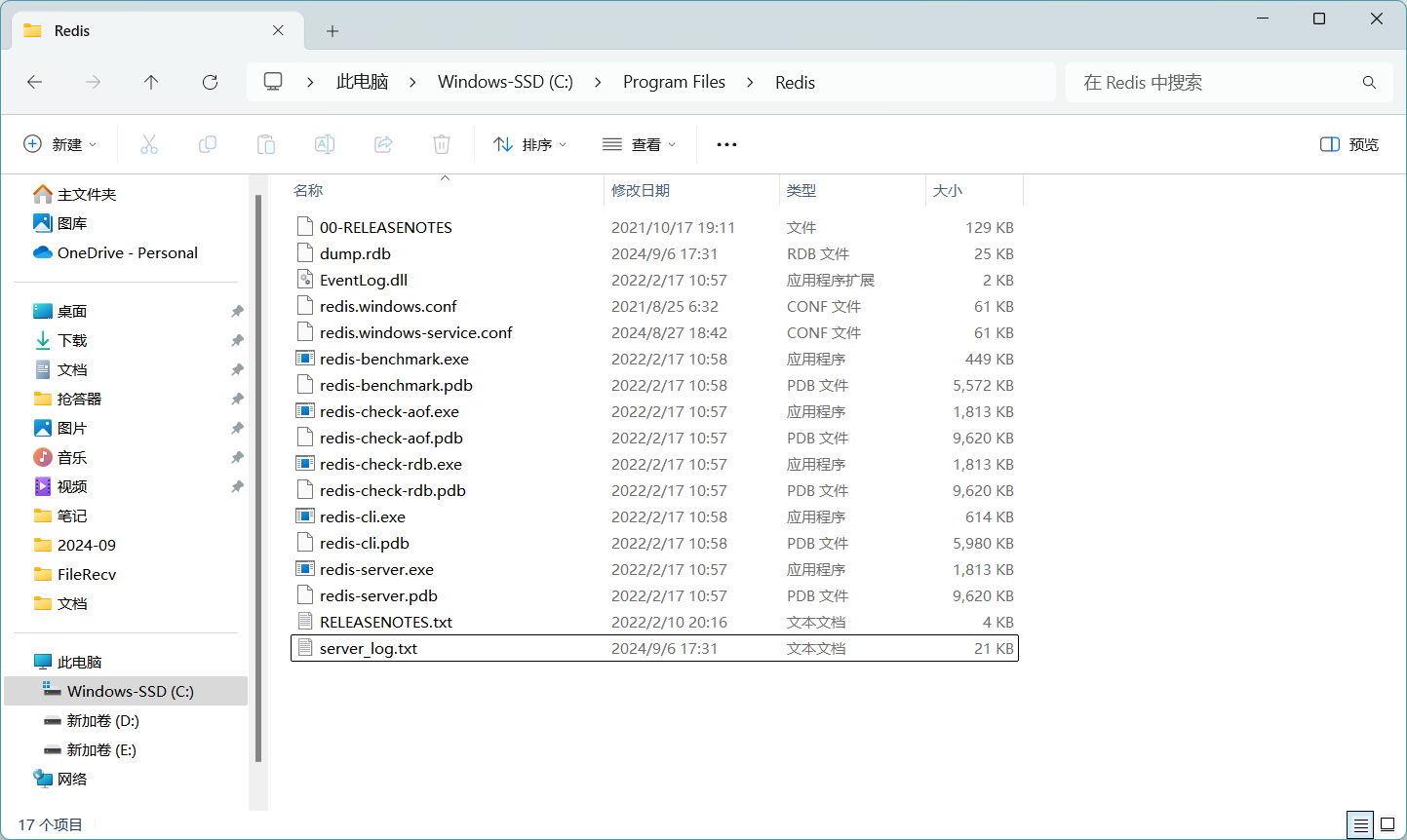
**2.1安装windows版本的redis 5.0.14.1**

首先访问<https://github.com/tporadowski/redis/releases> (需要科学上网)

单击Redis-x64-5.0.14.1.zip链接，下载Redis-x64-5.0.14.1.zip安装包

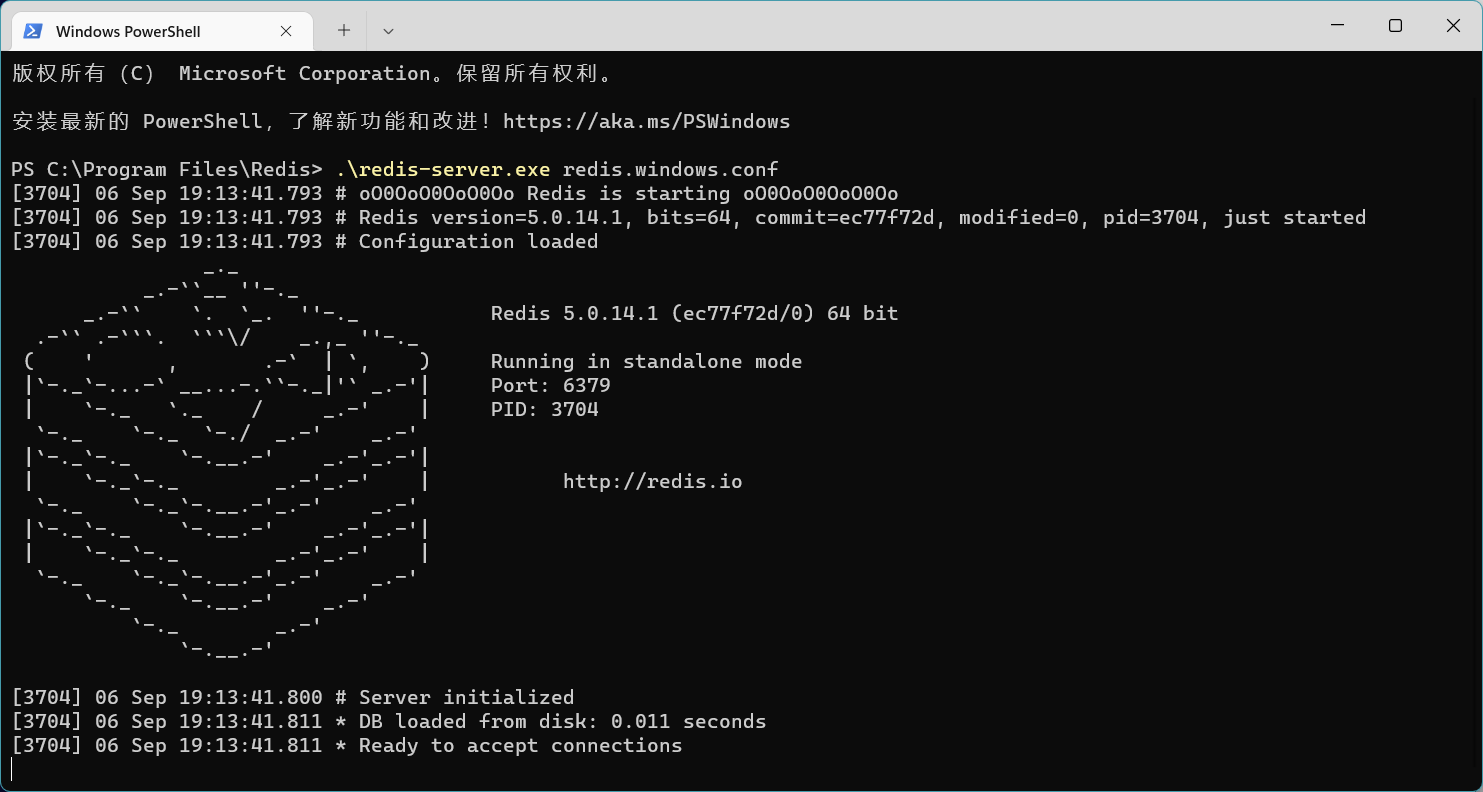


将下载的压缩包解压到C:\Program Files\Redis文件夹中，解压过后的内容如下



在Redis的安装目录(如上图所示)右击，点击**在终端中打开**，然后在弹出的Windows PowerShell中输入

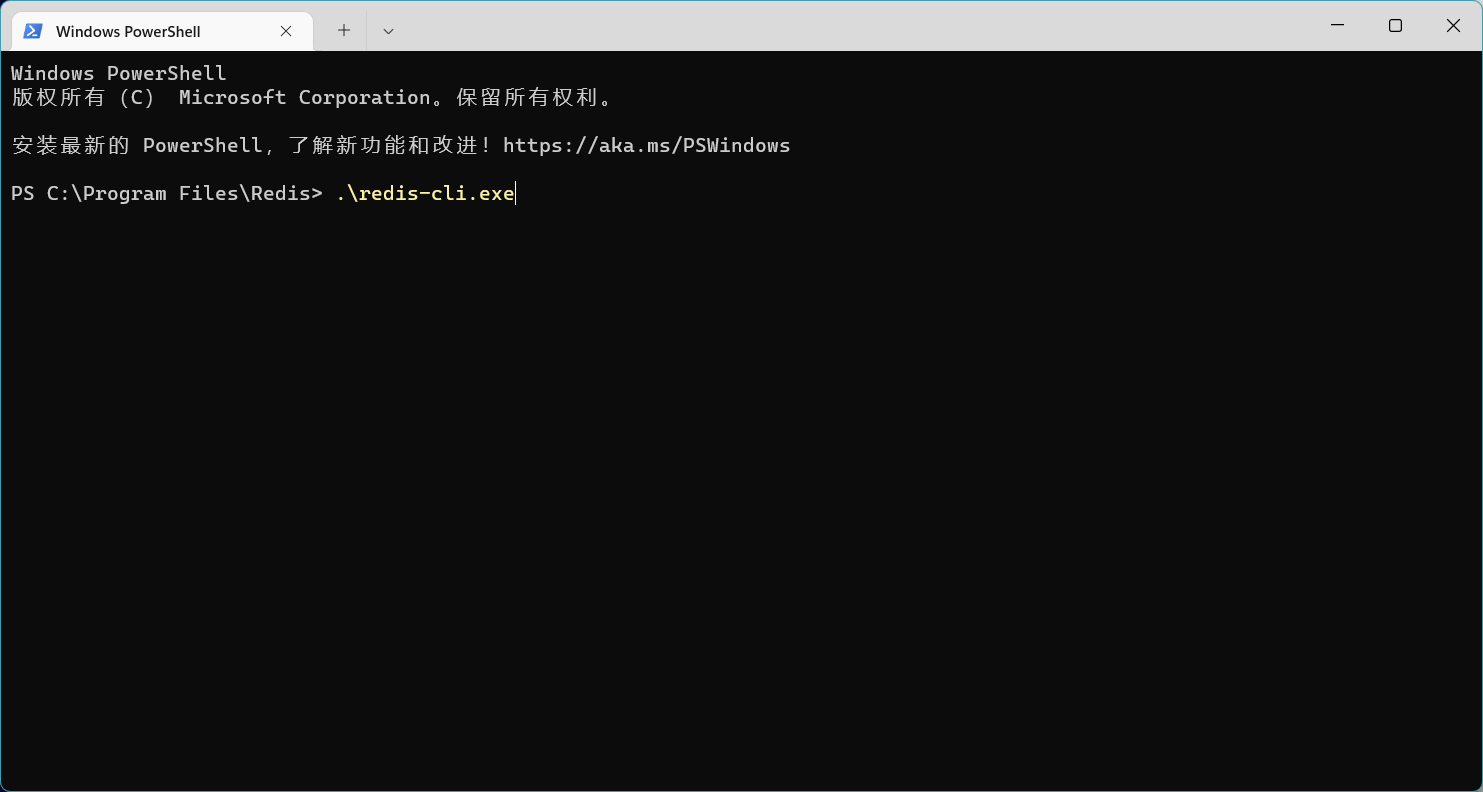
.\redis-server.exe redis.windows.conf 命令来启动redis服务



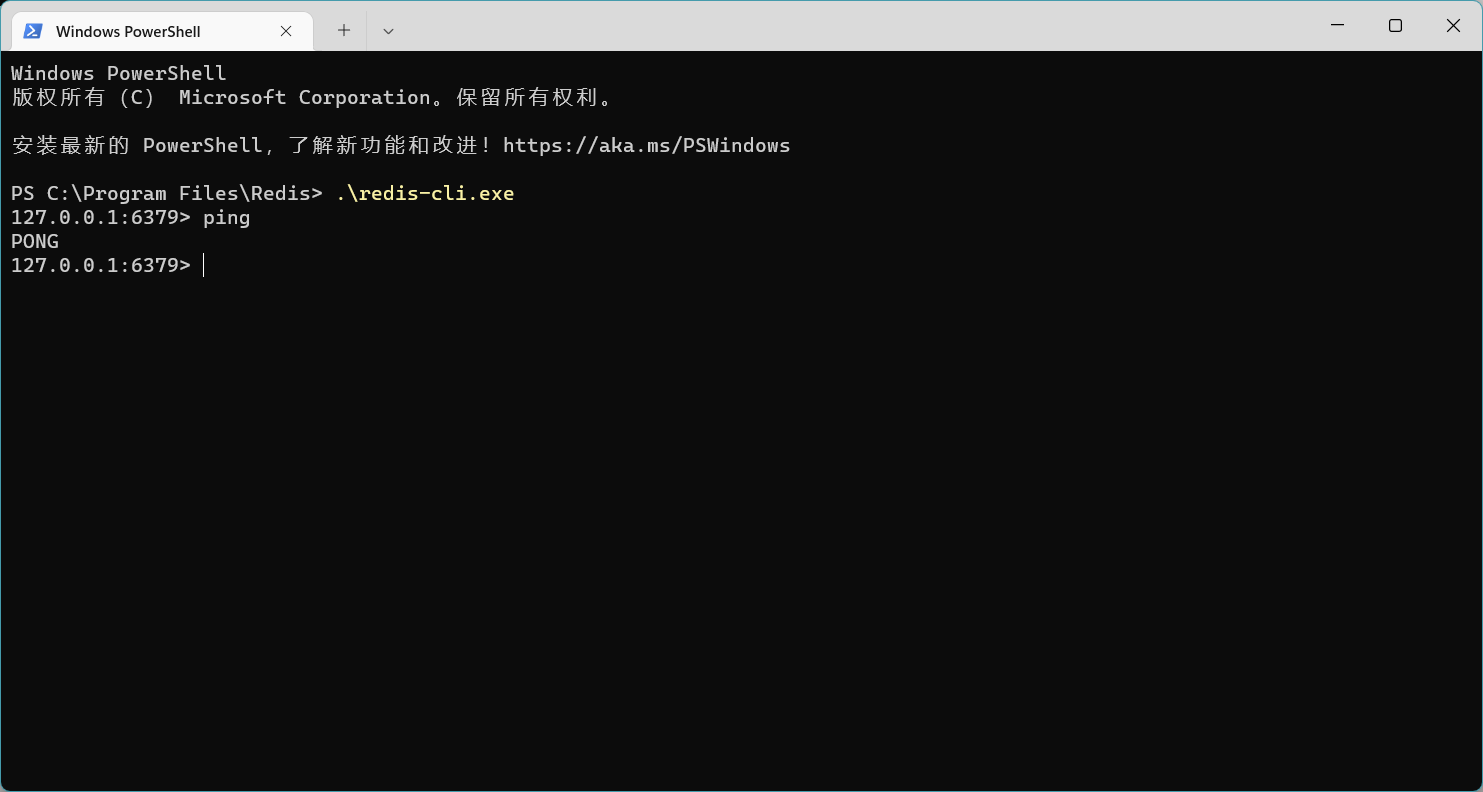
出现Ready to accept connections表示redis服务已经启动。

**2.2测试redis服务连通性**

为了测试redis是否能正确访问，我们在刚才的Redis的安装目录右击，选择**在终端中打开**，然后输入.\redis-cli.exe 命令来启动redis客户端程序

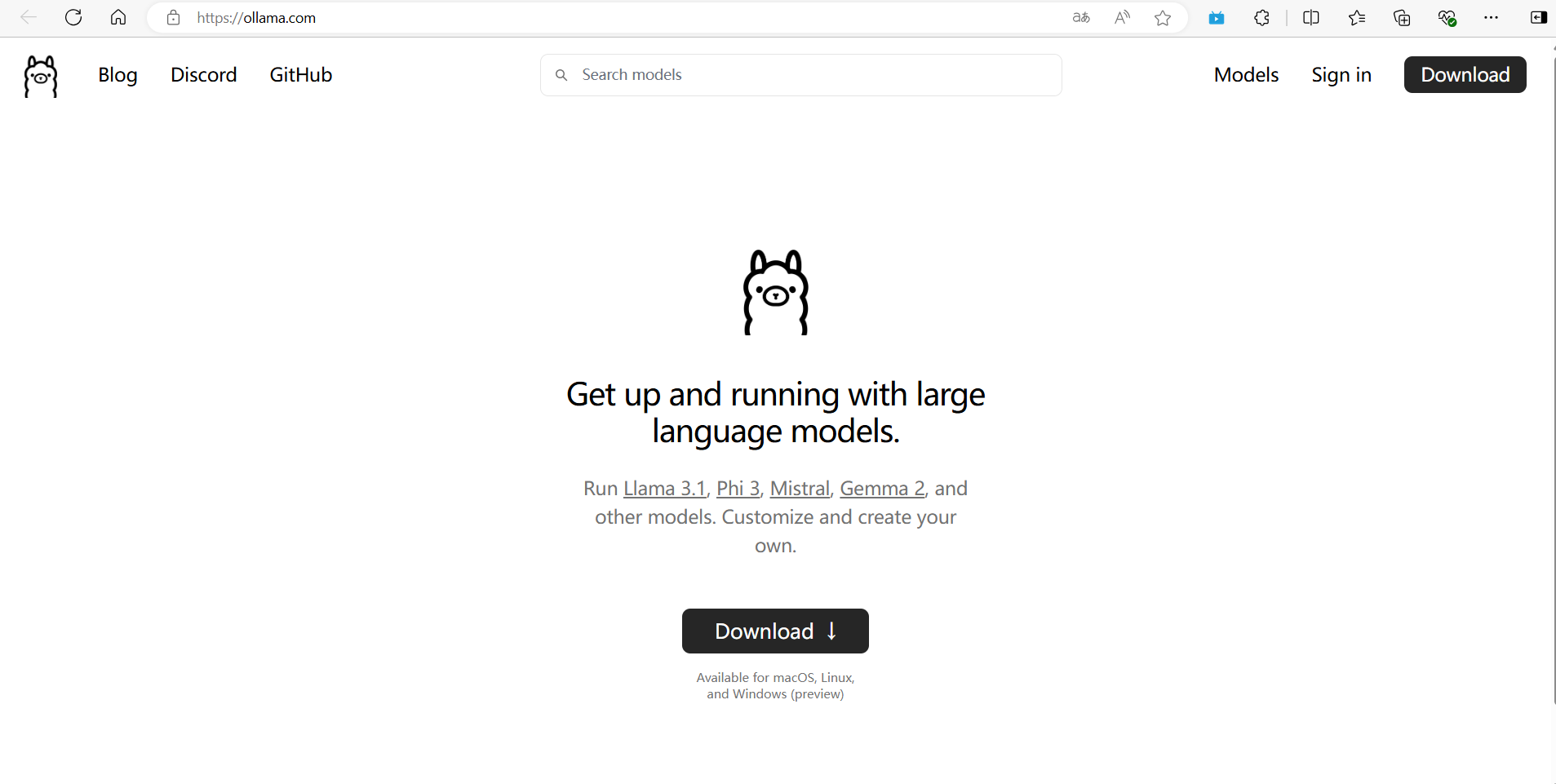


在弹出的PowerShell中输入**ping**命令来检测redis服务器与redis客户端的连通性，返回PONG则说明连接成功了。

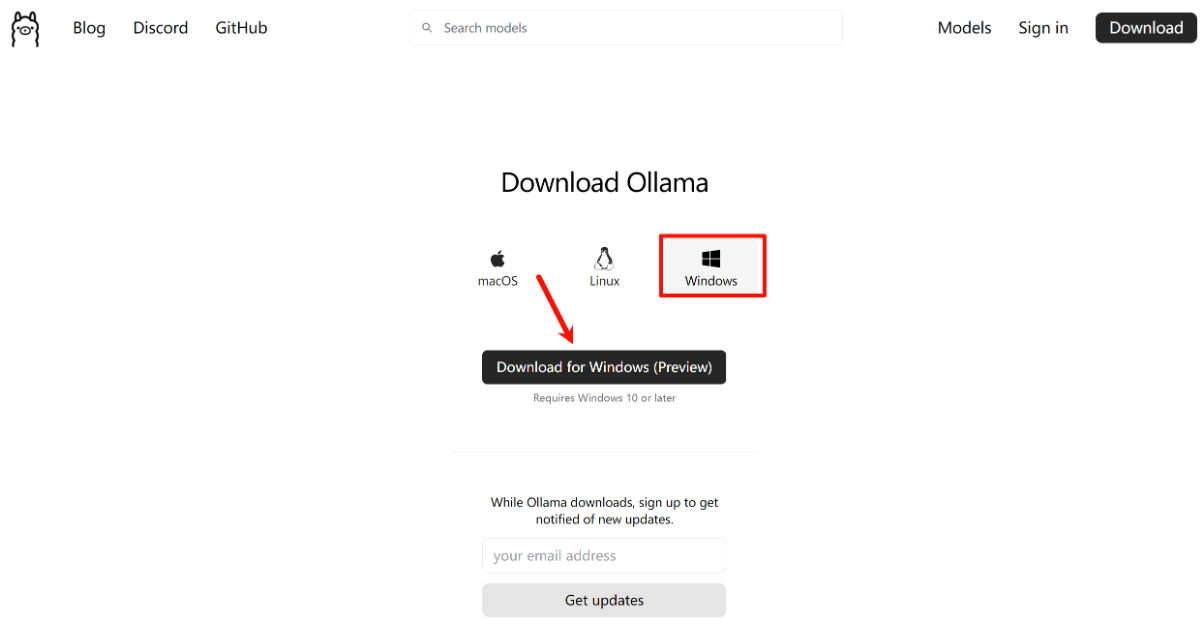


**3.1安装ollama（Ollama on Windows 需要 Windows 10 或更高版本）**

在浏览器输入<https://ollama.com/> 进入ollama官网，单击**Download**



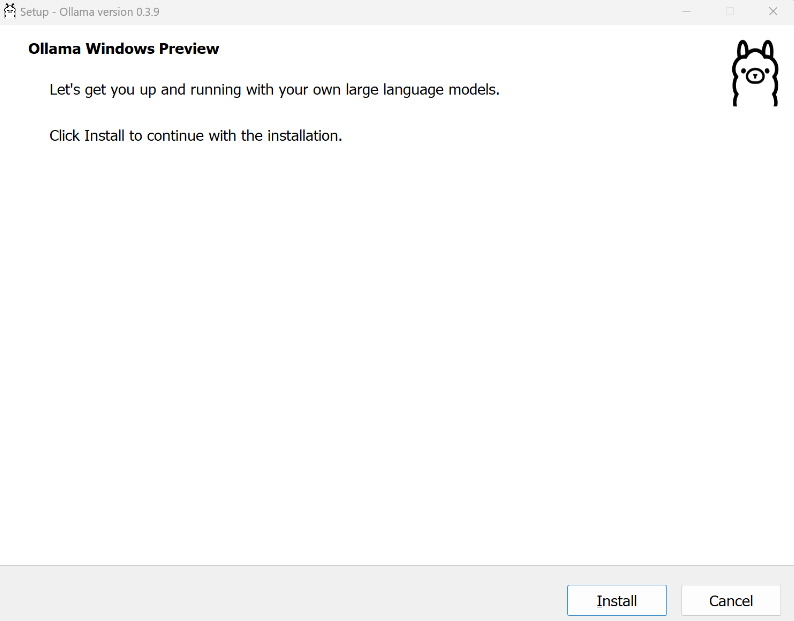
选择Windows版本，然后单击中间的Download for Windows(Preview)按钮，进行下载



1

2

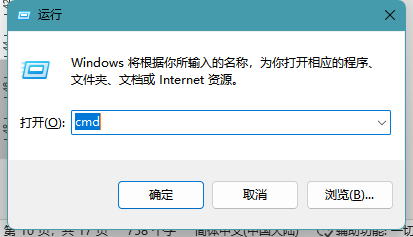
下载完成后，双击安装包文件，点击「Install」开始安装。



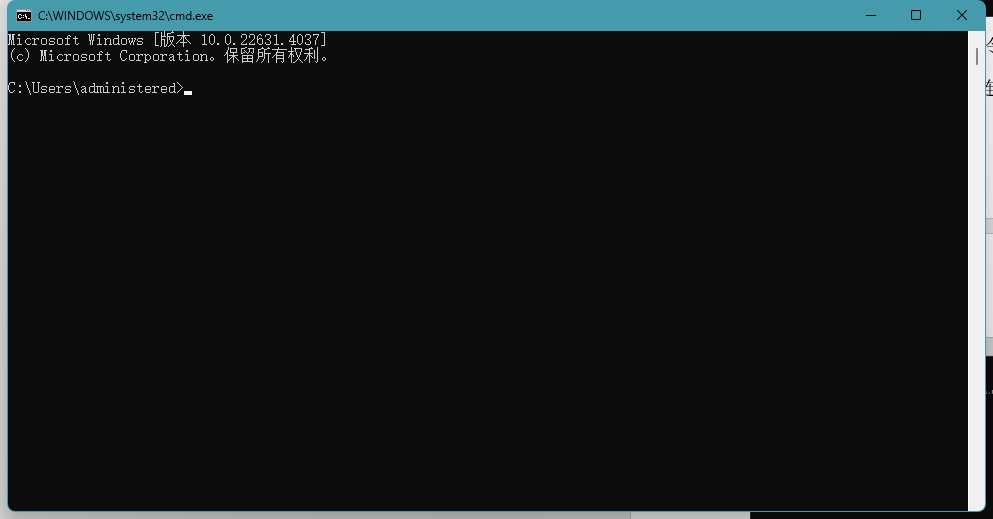
等待安装完成。

**3.2验证是否安装完成**

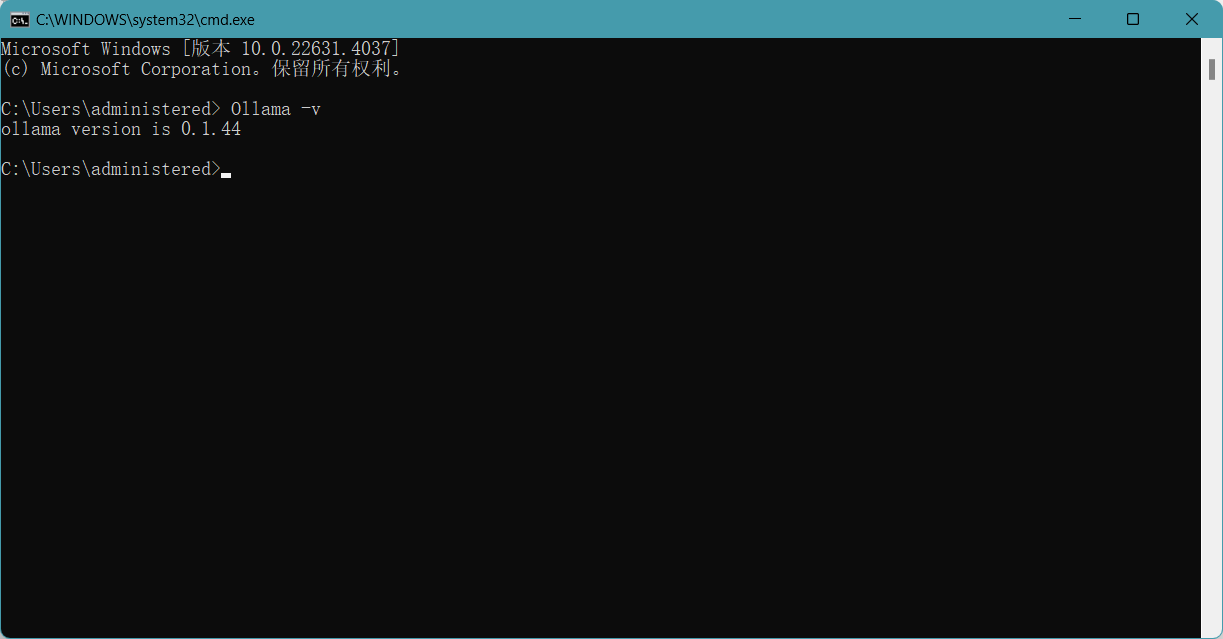
键盘上按住**Win+R，**在弹出的对话框中输入cmd



点击「**确定**」按钮

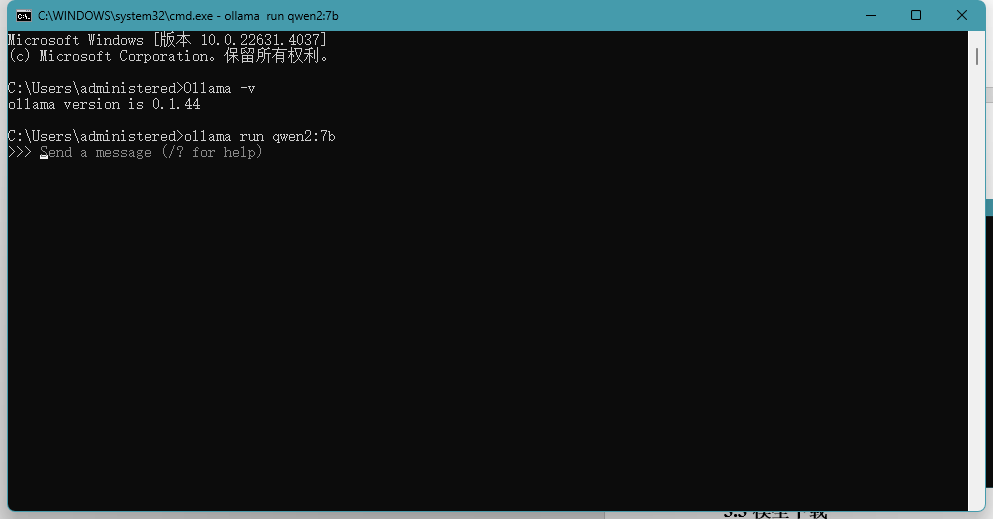


在弹出的终端窗口中输入Ollama -v命令，显示当前安装的ollama版本则表示安装成功。



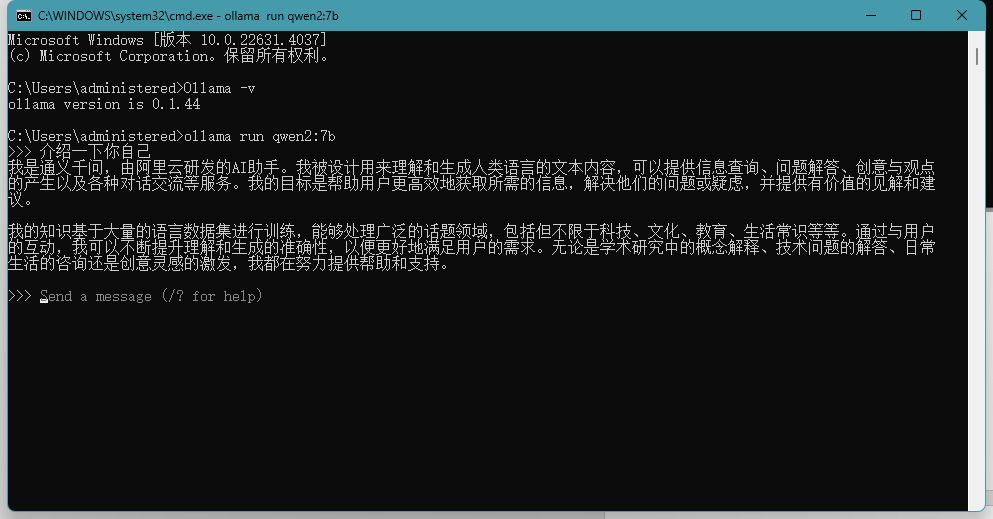
**3.3模型下载**

在刚才的终端窗口中输入ollama run qwen2:7b，直接运行qwen2:7b模型，如果是第一次使用，则会自动下载对应的模型文件。下载完成后，出现Send a message (/? for help) 表示模型下载成功



**3.4测试大模型回答**

在窗口中输入一些问题，与大模型进行对话

  
可以看到大模型在本地运行起来，并可以进行对话。

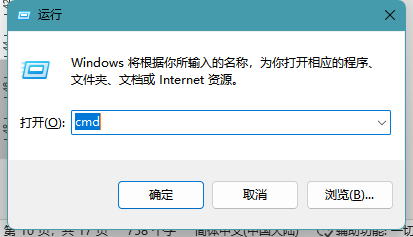
**至此项目必要的软件全部安装完成**

**二、项目启动**

**1将下载的HITL\_Project.zip解压到桌面**

**2启动redis**

键盘按住Win+R，在弹出的对话框中输入cmd



1

2

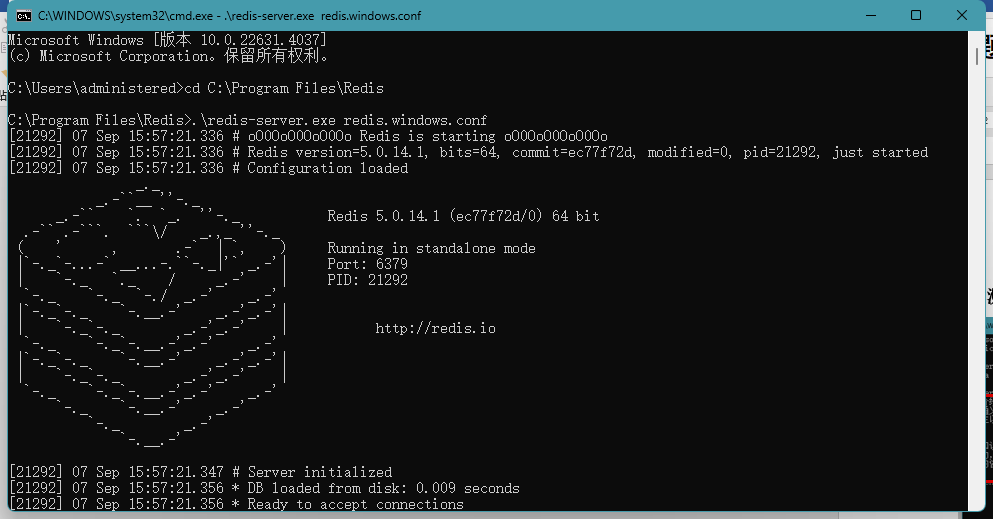
点击「**确定**」按钮

然后在弹出的终端窗口输入cd C:\Program Files\Redis命令进入redis安装目录

然后继续输入.\redis-server.exe redis.windows.conf命令启动redis服务

之后最小化redis启动的终端窗口，保持redis服务在后台运行

3

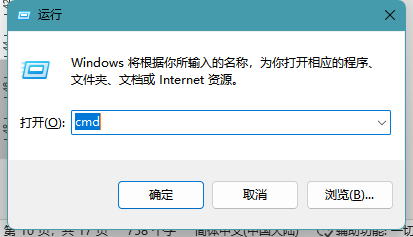


2

1

**3启动ollama**

键盘按住Win+R 在弹出的对话框中输入cmd



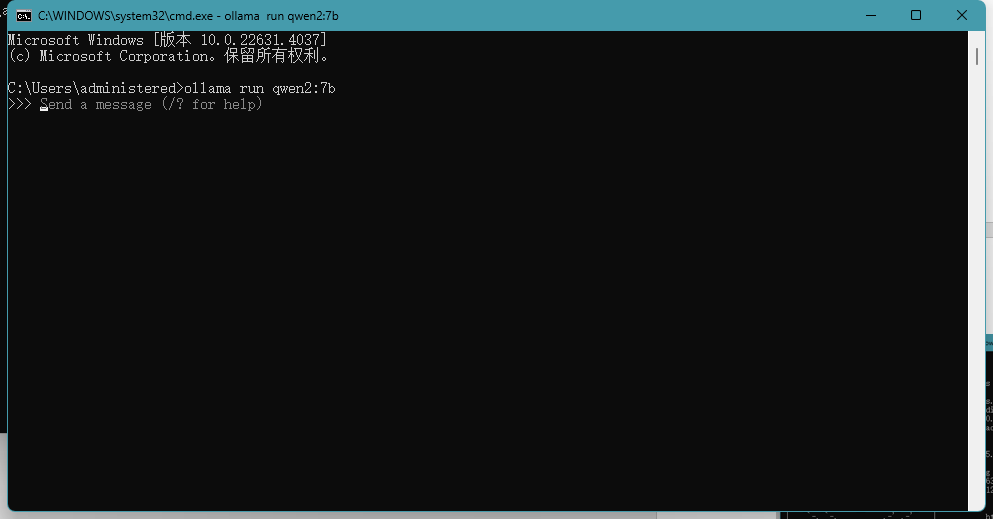
1

2

点击「**确定**」按钮

然后在弹出的终端窗口输入 ollama run qwen2:7b

出现如下界面后，最小化ollama启动的窗口，保持ollama在后台运行

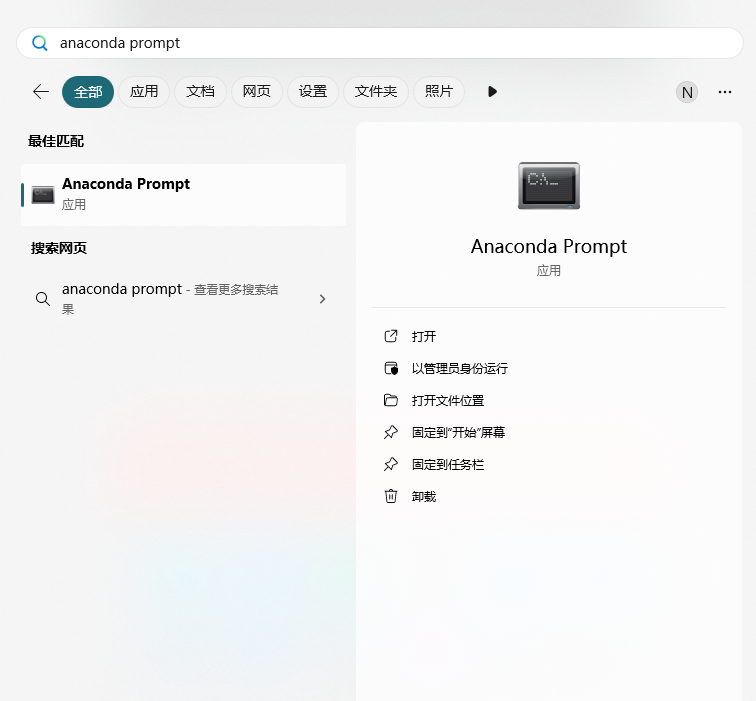


**4进入虚拟环境**

键盘上单击Win键，在弹出的菜单中搜索anaconda prompt



单击Anaconda Prompt应用



1

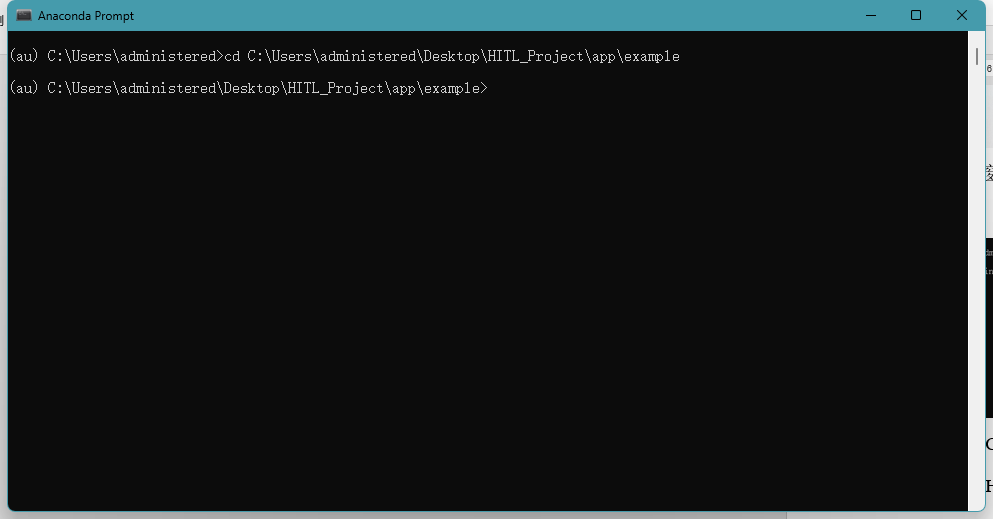
2

弹出如下Anaconda Prompt终端窗口，在Anaconda Prompt终端窗口中输入conda activate au命令进入au虚拟环境，成功进入au虚拟环境后，前面的**base**会变成**au**



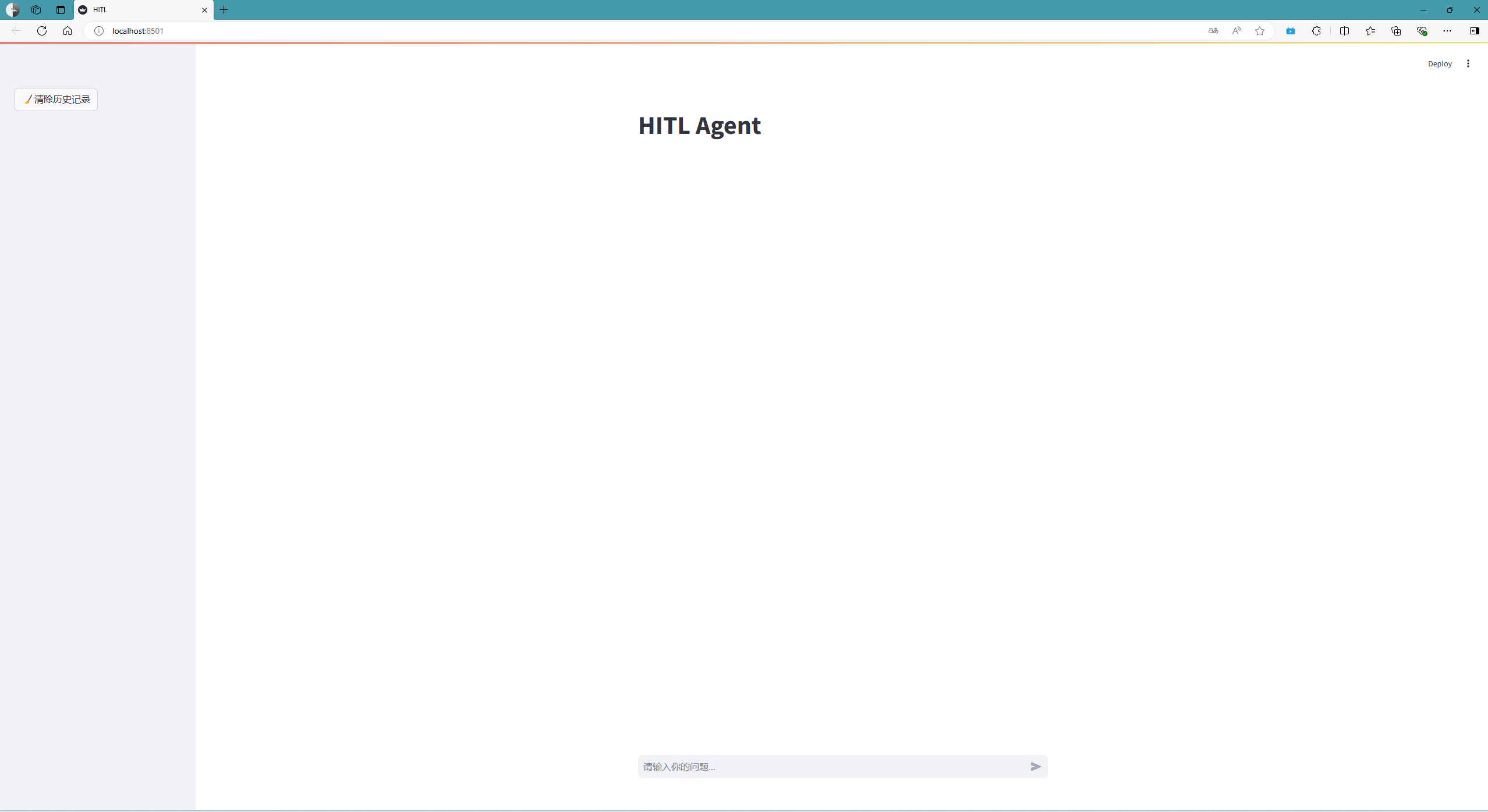
然后在Anaconda Prompt终端窗口中继续输入

cd C:\Users\administered\Desktop\HITL\_Project\app\example命令进入到HITL\_Project项目的example文件夹下



然后再继续输入streamlit run my\_agent.py命令启动my\_agent.py脚本文件，程序启动后会自动跳转到<http://localhost:8501>地址，出现如下界面，此时整个项目启动完成

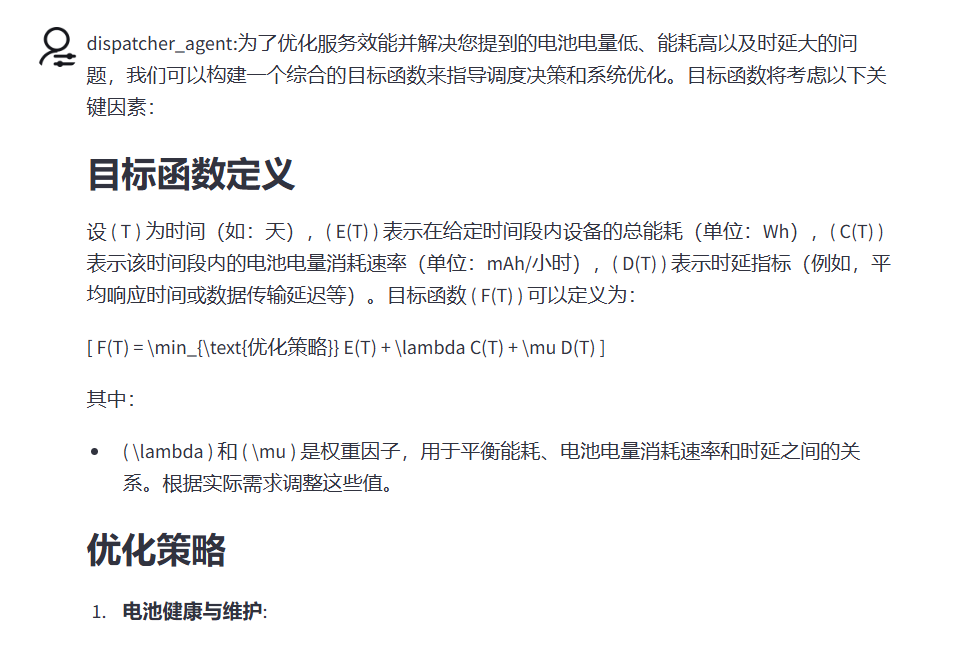
如果没有自动跳转，需要手动在浏览器上输入<http://localhost:8501>

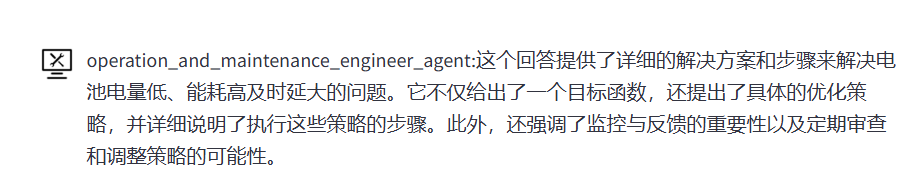


**二、测试样例**

样例1.输入完整信息：电池电量低，能耗高，时延大







样例2.输入不完整信息：电池电量低--->我希望能耗要低，时延要高

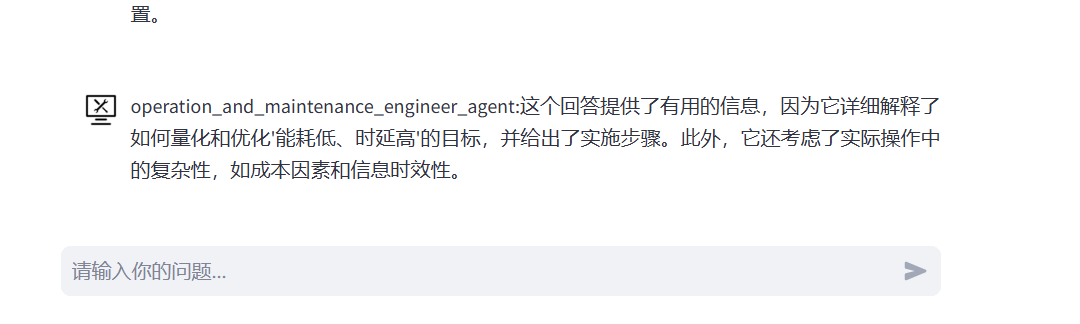
当输入信息不完整时，agent会询问关键词槽信息。此时我们继续在文本框中输入词槽的描述信息。



在电池电量低的场景下，时延信息、能耗信息，当词槽填充完整后，多智能体的交互会继续进行。





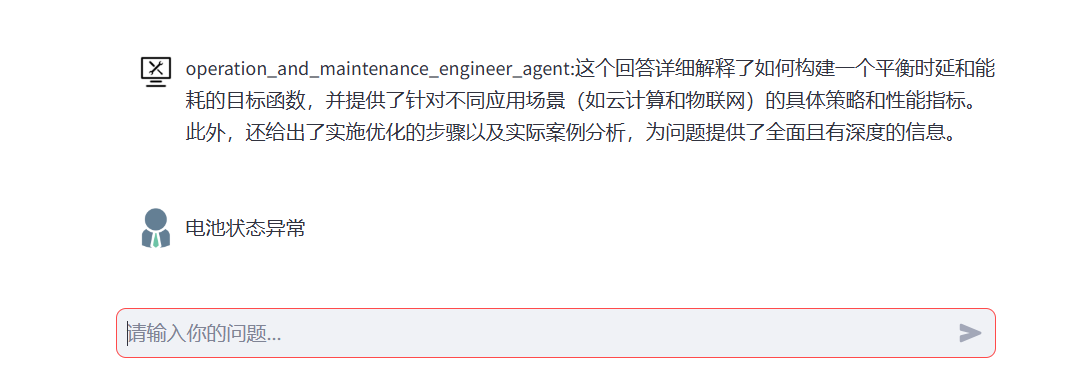


样例3.中途切换场景：先输入电池电量充足--->电池状态异常









样例4.输入不在设计范围内的场景：订机票



由于未在场景文件中设计定机票的场景，会提示未命中场景

**附场景文件**：

{

"battery\_status\_normal": {

"name": "电池状态正常",

"description": "当前场景下描述的是电池状态，且电池状态正常的",

"example": "JSON：[{'name': 'W(1,1)', 'desc': '电池状态正常时，时延的权重低。value应该是对该参数的描述', 'value': '时延低'}]\n输入：当前场景下，电池的状态是正常状态，能耗较高\n答：{'name': 'W(1,2)', 'value': '能耗较高'}",

"parameters": [

{

"name": "W(1,1)",

"desc": "电池状态正常时，对时延权重的描述。value应该是对该参数的描述",

"type": "string",

"required": true

},

{

"name": "W(1,2)",

"desc": "电池状态正常时，对能耗权重的描述，value应该是对该参数的描述",

"type": "string",

"required": true

}

]

},

"battery\_fully\_charged": {

"name": "电池能量充足",

"description": "当前场景下描述的是电池能量，且电池能量充足",

"example": "{'name':'W(2,1)','value':'xxx'},{'name':'W(2,2)','value':'xxx'}",

"parameters": [

{

"name": "W(2,1)",

"desc": "电池能量充足时，对时延权重的描述。",

"type": "string",

"required": true

},

{

"name": "W(2,2)",

"desc": "电池能量充足时，对能耗权重的描述。",

"type": "string",

"required": true

}

]

},

"battery\_low\_energy": {

"name": "电池能量不足",

"description": "当前场景下描述的是电池能量，且的电池能量不足",

"example": "{'name':'W(3,1)','value':'xxx'},{'name':'W(3,2)','value':'xxx'}",

"parameters": [

{

"name": "W(3,1)",

"desc": "电池能量不足时，对时延权重的描述。",

"type": "string",

"required": true

},

{

"name": "W(3,2)",

"desc": "电池能量不足时，对能耗权重的描述。",

"type": "string",

"required": true

}

]

},

" battery\_status\_abnormal": {

"name": "电池状态异常",

"description": "当前场景下描述的是电池状态，且电池状态异常",

"example": "{'name':'W(4,1)','value':'xxx'},{'name':'W(4,2)','value':'xxx'}",

"parameters": [

{

"name": "W(4,1)",

"desc": "电池状态异常时，对时延权重的描述。",

"type": "string",

"required": true

},

{

"name": "W(4,2)",

"desc": "电池状态异常时，对能耗权重的描述。",

"type": "string",

"required": true

}

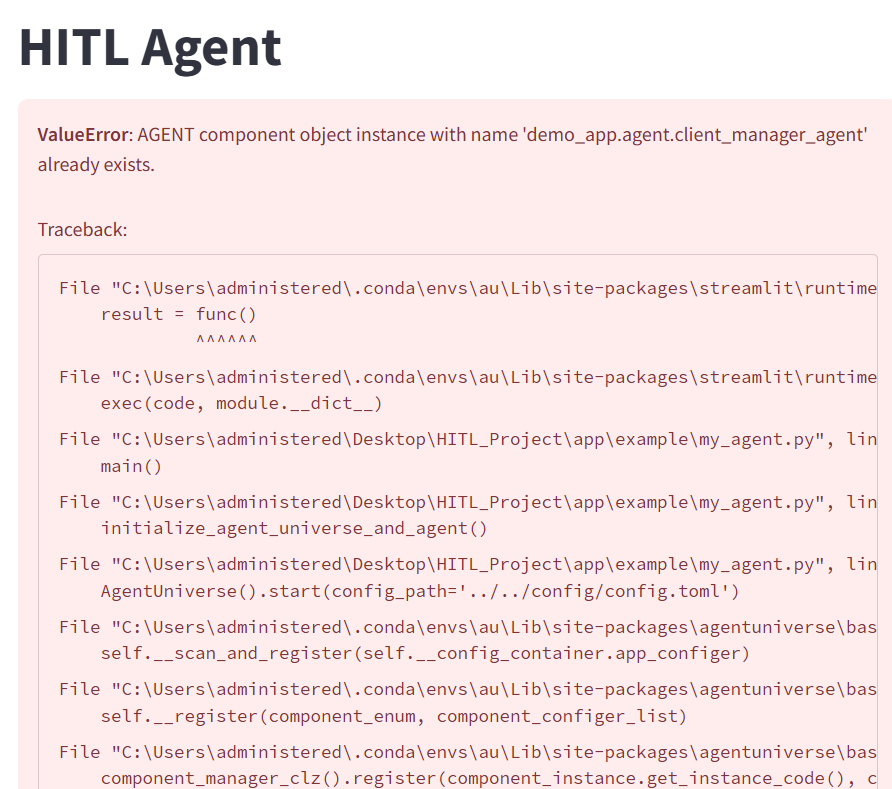
]

}

}

**三、注意事项：**

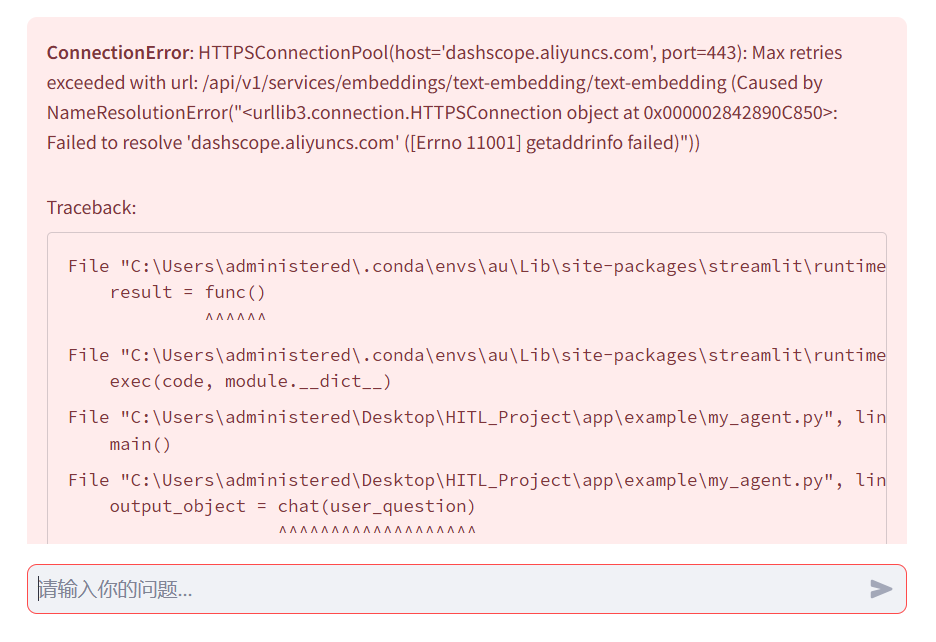
问题1.多次打开页面，导致重复注册app，出现如下ValueError错误。



解决办法：在anaconda终端按住键盘上的Ctrl+C 退出程序，然后重新在anaconda终端使用命令：streamlit run my\_agent.py运行项目。

问题2.出现ConnectionError

指出Failed to resolve 'dashscope.aliyuncs.com'，一般表明在使用dashscope API时，因网络问题无法连接dashscope服务器。



解决办法：检查网络设置，确保能正常访问互联网。