Environment Setup for "The Flaws of Others: An LLM-driven Framework for Scientific Knowledge Production"

By Juan B. Gutiérrez. juan.gutierrez3@utsa.edu -
Professor, Department of Mathematics. University of Texas at San Antonio
Updated on September 2, 2025

This document provides a structured walkthrough for setting up a local environment to run large language model (LLM) interfaces for scientific tasks. It is part of the manuscript "*A Mathematical Theory of Discursive Networks*" arXiv:2507.06565 DOI: 10.48550/arXiv.2507.06565

These instructions centered on Windows but adaptable for Mac and Linux. It begins with configuring environment variables to securely store API keys for OpenAI, Anthropic, and Groq. The reader is then guided through the installation of Python, emphasizing proper PATH configuration and optional tools like Visual Studio Code (VSC), Git, and Windows Terminal. Additional sections cover the setup of virtual environments both within and outside VSC, the installation of essential Python libraries, and optional tools such as LaTeX and C/C++ compilers. The setup culminates in verifying functionality by cloning a GitHub repository and running a helper script, ensuring the environment supports modular, reproducible project development with minimal dependency conflicts.

1) Required: Setting up environment variables for API keys:

You will receive an API key for OpenAI and Anthropic during our first session. You can get your own API for free at Groq. You need to create three API key variables following the procedure described below: OPENAI_API_KEY, ANTHROPIC_API_KEY, GROQ_API_KEY. Test the program agentGroq.py, agentGPT.py and/or agentClaude.py from the Github repository described below in this document.

On Windows, execute the following three commands one at a time for each API key. The example for `OPENAI_API_KEY` is as follows:

```
setx OPENAI_API_KEY "[sk-... API key]"
Exit
echo %OPENAI_API_KEY%
```

In the code above, replace `[sk-... API key]` with your actual API key. The first command sets the environment variable permanently for your user account. The second command closes the Command Prompt to ensure the new environment variable is registered. The third command, run in a new Command Prompt window, prints the stored value to verify it was set correctly.

On Mac/Linux execute the following three commands one at a time for each API key.

The example for OPENAI_API_KEY is as follows:

```
echo "export OPENAI_API_KEY='[sk-... API key]'" >> ~/.zshrc
source ~/.zshrc
echo $OPENAI_API_KEY
```

In the code above, replace '[sk-... API key]' with your actual API key.

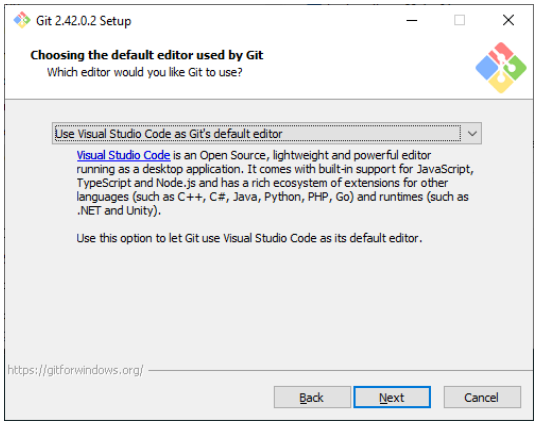
As reported by Bryan Fowler, if you experience errors trying to add your environmental variables in a Mac/Linux, the most likely problem is ownership of the ~/.zshrc file. In this case, execute

```
sudo chown $(whoami) ~/.zshrc
```

And try to create the environmental variables again.

As reported by Drew Stephen regarding instructions for Mac, "*apparently it doesn't work on the default c shell but switching to the zsh lets it run.*"You might need to close and reopen the terminal window from where you started for changes to become effective.

2) **Required**: Install Git for Windows. For Mac, install Git for Mac. Linux also has a version available. In the second screen, select Visual Studio Code as your Git editor. Use all other default options.



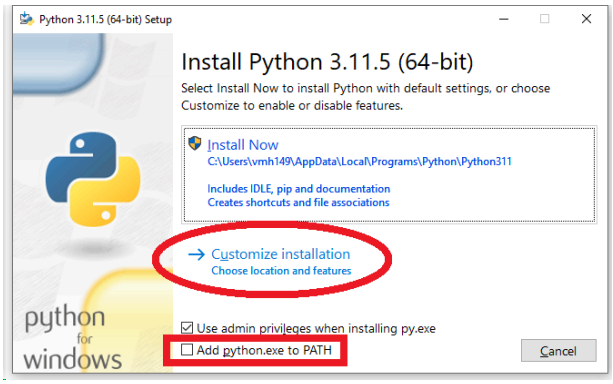
3) **Required**: Open Git Bash (or any terminal). The command line allows you to control you computer by typing instructions, providing precise access to files and system features. For example, typing `ls` on a Unix-like system such as Linux or macOS lists the files and directories in the current location, such as `ls /home/user/Documents` to display the contents of the Documents folder. On Windows, the `dir` command serves the same function, so `dir C:\Users\Alice\Desktop` shows what is on the Desktop. To move between directories, the `cd` command is used; `cd Downloads` changes the working directory to Downloads, while `cd ..` moves one level up. These commands allow

efficient navigation and inspection of the file system through the terminal.

Now that you installed git, you should **go to the folder you want to work in** using command line, and run the following command:

git clone https://github.com/DiscursiveNetworks/FOO_QtPy.git

4) **Required: Install Python**: https://www.python.org/downloads/. Install any recent Python version. Go to your downloads and double click on the install file to start installation. By default the Python installer for Windows places its executables in the user's AppData directory, so that it doesn't require administrative permissions. This works for most scenarios. If you're the only user on the system, you might want to place Python in a higher-level directory (e.g. C:\Python or /usr/local/bin) to have a shorter path to the binaries (sometimes you will need that). Depending on your preferences, either select "*Install Now*" or "*Customized installation*" (my preference). Please make sure you select "*Add python.exe to path*"; this will save you a few headaches later.

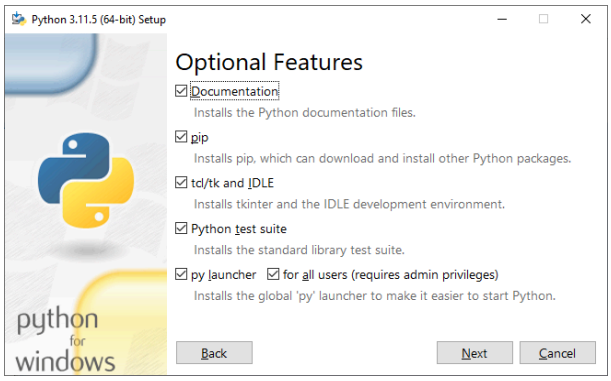


For PC: If you did not add `python.exe` to the PATH during installation, wait until the installation is complete. Then open File Explorer and right-click on "This PC." Select "Properties" from the context menu. In the System window that opens, click on "Advanced system settings" in the left-hand sidebar. In the new window, ensure the "Advanced" tab is selected and click the "Environment Variables" button near the bottom. Under the "System variables" section, find and double-click on the variable named "Path." This will open a window where you can add the path to the Python executable manually.

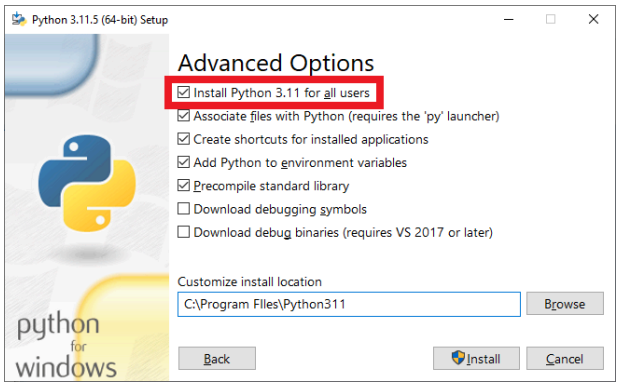
For Mac: If you did not ensure Python was added to your system's PATH during installation, you can manually do so using the following steps on a Mac. After installation is complete, open the Terminal application. Enter the command `which python3` or `which python` to find the path to the installed Python binary. Copy that path. Then open your shell configuration file in a text editor—this will typically be `~/.zshrc` if you are using the Zsh shell (default on newer versions of macOS) or `~/.bash_profile` if

you are using Bash. Add the line `export PATH="/path/to/python:$PATH"`, replacing `/path/to/python` with the path you copied. Save the file and close the editor. In the Terminal, run `source ~/.zshrc` or `source ~/.bash_profile` depending on the file you edited, so the new PATH is loaded into your environment.

In the window “Optional Features” select all features



Select “*Install Python X.XX for all users*” if you can and want. This requires administrative privileges. Select the folder of your choice for the install location.



5) **Optional**: If you’d like to code in C or C++, you have several options, including GCC, the GNU Compiler Collection, but it requires some effort to install the prerequisites. From Microsoft there is Visual Studio Community Edition (VSCE), which encapsulates the complexity of installing multiple compilers (including the .NET framework SDK). I recommend you install VSCE with all “Workloads” in “Web & Cloud” (except Python) and “Desktop & Mobile”.

6) **Optional**: If you intend to produce PDF documents using LaTeX, install MikTeX.

7) **Strongly recommended: If you are new to Python, install Visual Studio Code (VSC)**: There are many options to write Python code, and you could have a favorite one. I must pick one to deliver instruction, and experience has shown me that VSC offers the

least amount of friction.

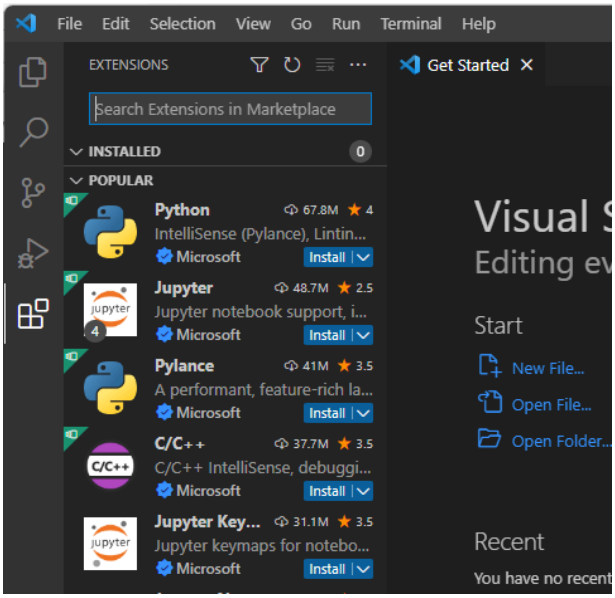
In VSC, there are two main options: "*User Installer*"and "*System Installer*". Use the User Installer if you want to install only for the current user. For all users, use System installer; I prefer this choice because it installs VSC in C:\Program Files\Microsoft VS Code

This little giant has become a darling of coders for many good reasons. It is available for Mac, Linux and Windows. There is an open source identical alternative called VSCodium, available at https://vscodium.com/ I personally use VSCode.

8) **If you installed VSC: Required**: Select the Extensions icon on the left and install the following Visual Studio Code libraries

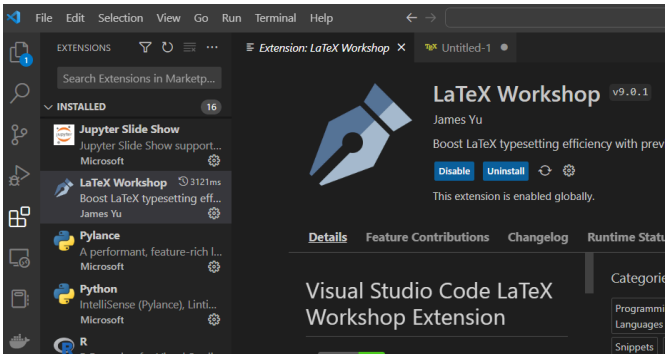
   a) **Required**: Python extension for Visual Studio Code.

   b) **Optional**: Also install Jupyter (this also installs Pylance, Jupyter Keymap, Jupyter Notebook Renderers, Jupyter SlideShow). Pay attention to the publisher of the extension; use the ones by Microsoft.



   c) **Optional**: Install C/C++ for Visual Studio Code, C/C++ Themes & C/C++ Extension Pack.

   d) **Optional**: Install the extension "**LaTeX Workshop**" by James Yu. You will never use another LaTeX editor :)



9) **Optional**: For Windows users, install Windows Terminal. This Microsoft app allows you to use a multi document window with tabs for different command consoles like PowerShell, Ubuntu, DOS, Git, etc. You will need this if you want to complete the steps related to Linux in the next section of this document.

10) If you installed VSC: Required: To start VSC, use the command line tool for your operating system and go to the folder CommandLineLLM. After Step 9, the command will be

```
cd CommandLineLLM
```

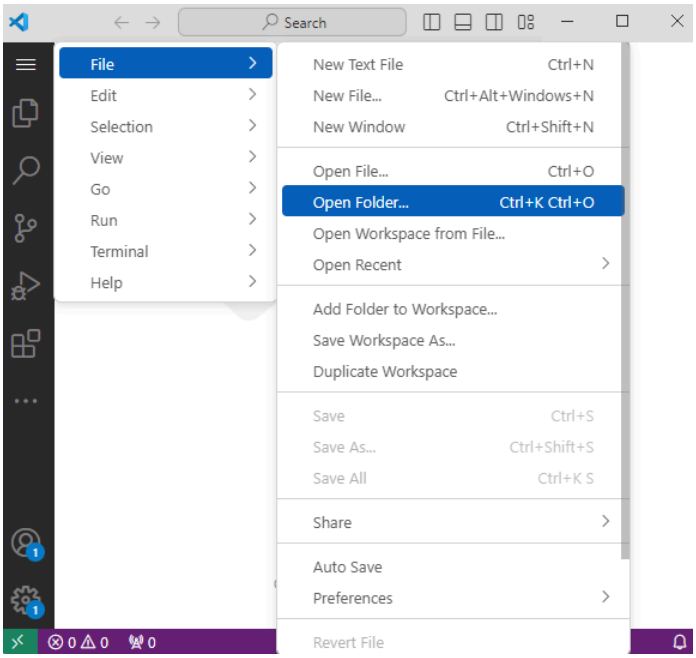
Inside this folder, execute the command

```
code .
```

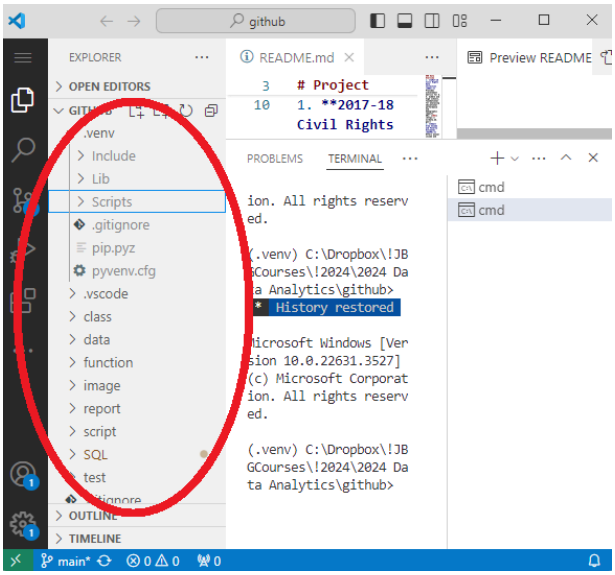
Note the period after the command code; it instructs VSC to use the current folder as the working folder.

11) **Partially Required (this or 12)**: Create virtual environments ***inside*** Visual Studio.We will start by creating a virtual environment; the reason for this is that sometimes specific versions of different libraries might be incompatible. You should create an environment for each project you work on.

a) Open visual studio. Select the option *File* > *Open Folder…* Choose the folder you want to use for your project.

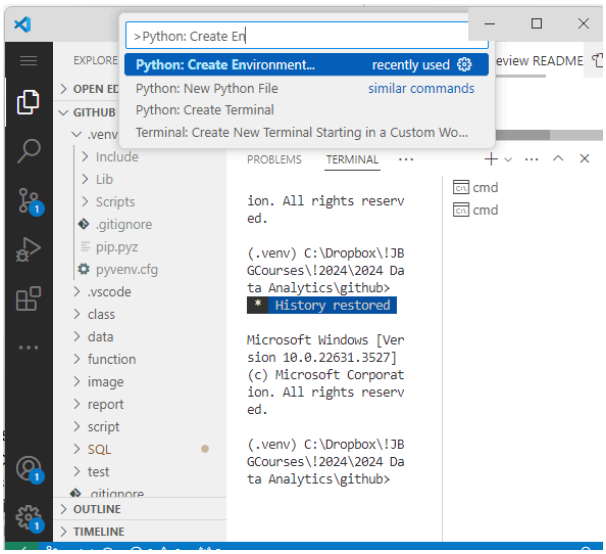


b) Once you select the folder, you will see the files related to the project you are working on in the Visual Studio File Explorer. Notice that the status bar changes color from purple to blue.

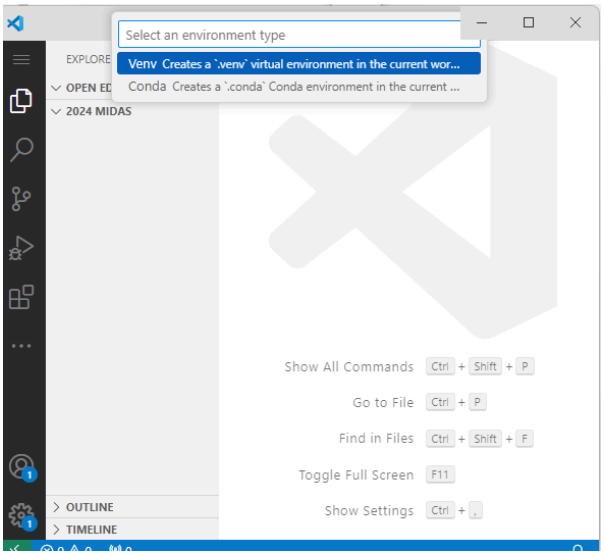


c) Now, hit the keys *SHIFT+CTRL+P*. This will allow you to type in the command palette the following instruction:

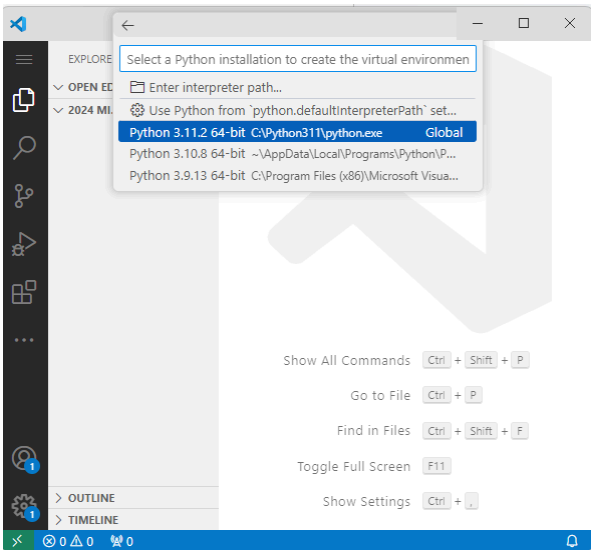
Python: Create Environment



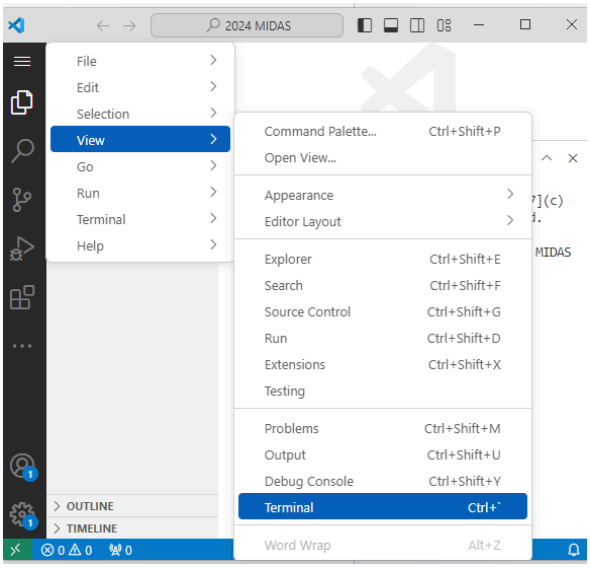
d) You will likely see two options: venv and conda. Choose venv.



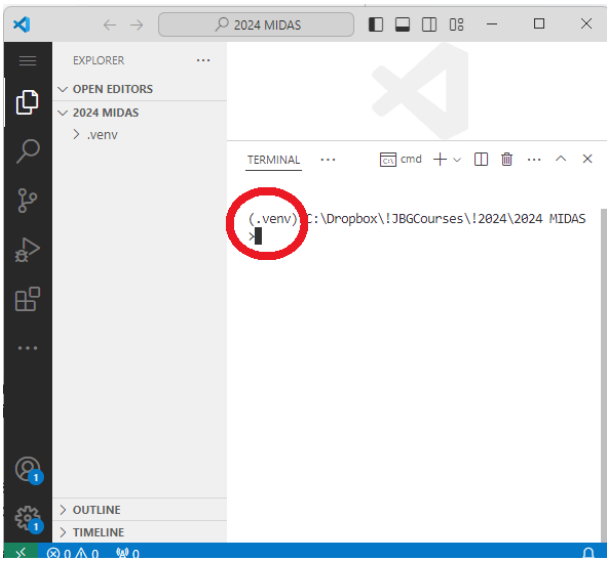
e) Once you choose venv, you will have to pick a Python installation. Pick the one we just installed (this could be the only one you see).



f) Now, go to the menu View > Terminal.



g) You will see that the command prompt now has the word (.venv). This means that the virtual environment has been installed and is now active. If it has not been activated, you can simply type activate. To inactivate the virtual environment, you simply need to type deactivate.



12) **Partially Required (this or 11)**: Create virtual environments ***outside*** Visual Studio. A comprehensive guide is here:

https://biomathematicus.me/working-with-python-virtual-environments-in-visual-studio-code/

Open a terminal window. Now you are ready to install Python packages. We will start by creating a virtual environment; the reason for this is that sometimes specific versions of different libraries might be incompatible. You should create an environment for each project you work on.

a) All related instructions about packages and virtual environments are at:

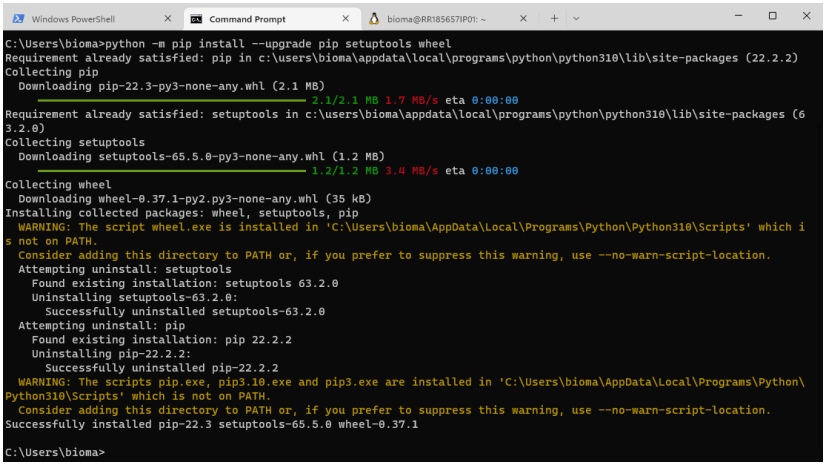
https://packaging.python.org/en/latest/tutorials/installing-packages/

b) Check your version of pip. It is the most popular tool for installing Python packages, and the one included with modern versions of Python. If the following command returns a version number, you are good to go. Otherwise, reinstall python (or bootstrap it).

```
python -m pip --version
```

c) Update the pip and setuptools

```
python -m pip install --upgrade pip setuptools wheel
```



d) I like to have an easy-to-access folder for virtual environments, thus I always create a folder like `c:\penv` for this purpose. To create this folder, run the following command in the terminal: `mkdir c:\penv` You can name this folder whatever you like. The importance of this will become evident in step 8. Also importantly, the folder in which you create your virtual environments does not need to be the folder in which your code resides.

e) Now **create** a virtual environment called venn (this is an example for a program about Venn diagrams; you can call it whatever name you prefer):

```
python -m venv c:\penv\venn
```

f) To **activate** this environment execute the following command in the terminal in VSC:

```
C:\penv\venn\Scripts\activate
```

Now the command line will show the prompt named as the environment that was activated.

```
C:\Users\bioma>C:\penv\venn\Scripts\activate

(venn) C:\Users\bioma>
```

g) To **deactivate** a virtual environment, simply type `deactivate` in the command prompt.

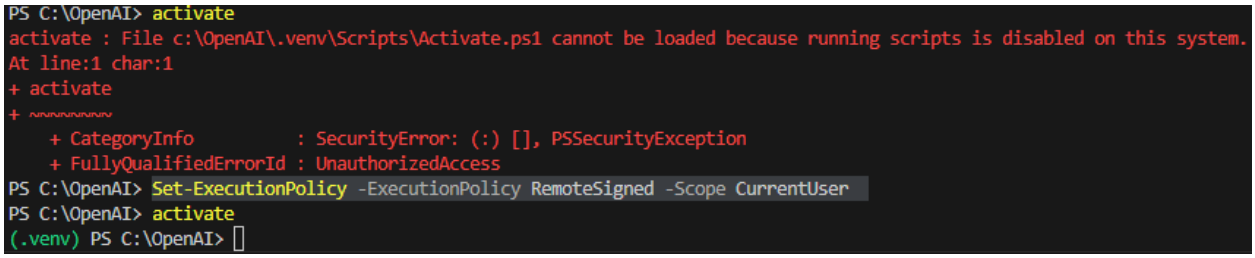
h) The ideal solution is to use the ActivateEnv.bat file described in the appendix (add c:\venv to the PATH environment variable). Call it from a .bat or a .sh file as in the following example:

```
ActivateEnv DiNet "C:\Dropbox\!JBGResearch\!!!DiNet\FOO_QtPy"
```

13) **Troubleshoot:**
As reported by Lorena Roa de La Cruz, you could see an error when trying to activate a virtual environment. This is due to configuration of permissions. If this happens, run the command:

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

After that, run the script '`activate`'



14) **Required (after completing 11 or 12)**: We will install libraries for the python environment for this project environment. The most important libraries are:

OpenAI
Anthropic
Langchain
PyQt5
NumPy
Matplotlib
SciPy
Jupyter
Pandas

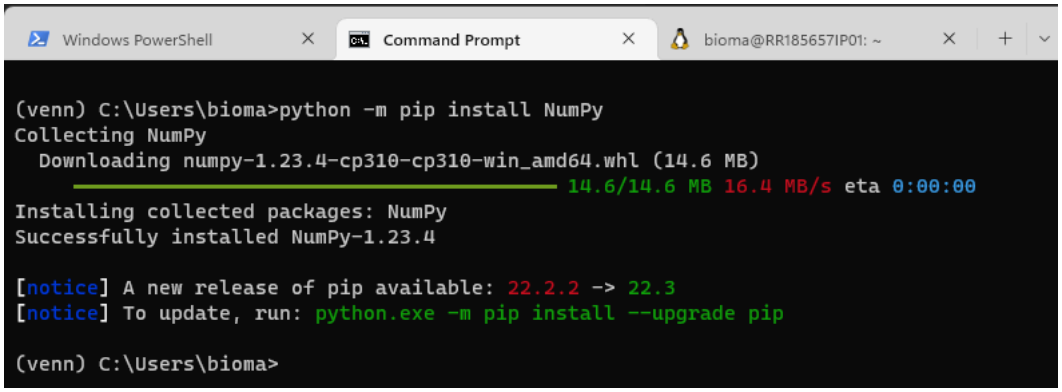
For each one of them type the following command: python -m pip install [library name]
For example:

python -m pip install NumPy

Alternatively, if you are using git bash, you can simply type:

pip install NumPy

Repeat for the libraries listed above. You should see something like this:



It is possible that the code fails; the source code repository changes over time. In that case the error will be something like:

ModuleNotFoundError: No module named 'library_name'

Where 'library_name' stands for the library name that is causing the error. In that case, simply follow the procedures above to install the missing library.

An important suggestion that comes from experiencing frustration in the past is this: Keep the base environment clean, that is, with minimal libraries. Conflicts between libraries will arise once you install a number of them. This is why it is a good practice to create an environment for each project.

You know your environment is fully functional when you can execute the file agentGroq.py, agentGPT.py and/or agentClaude.py.

15) **Optional**: Now, create a Jupyter Notebook.

a) Abundant relevant information regarding Jupyter in VSC is available at https://code.visualstudio.com/docs/datascience/jupyter-notebooks

b) Activate the environment you want to use according to step 13.f

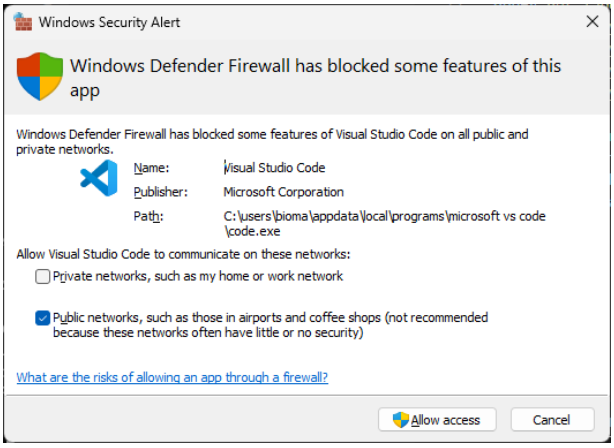
c) You need to attach the kernel of your virtual environment. For example, if I have a virtual environment called venn:

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

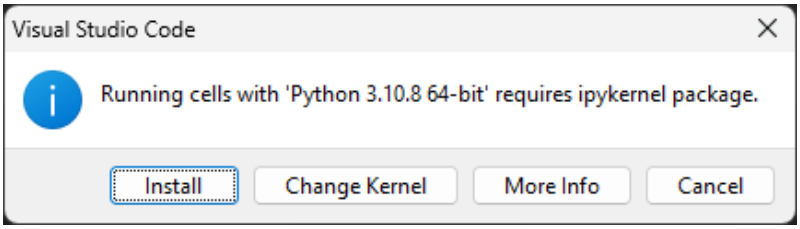
d) Execute step 13.a.

e) In VSC, run the command Create:New Jupyter Notebook from the Command Palette (Ctrl+Shift+P) or by creating a new .ipynb file in your workspace.

f) You might receive a Security Alert regarding the firewall. Allow access.



g) After you enter a command, you might see the following message:



Install it.

16) Finally, you can simply enter Python commands in the Terminal. Open a Command Prompt and type the following command:

```
python
```

This will allow you to enter python commands, e.g. 1+1

APPENDIX

ActivateEnv.bat to be placed in c:\venv to activate Python environments

```
@echo off
setlocal

:: Read parameters
set "ENV_NAME=%~1"
set "BASE_DIR=%~2"
echo Starting activation of the virtual environment '\%ENV_NAME%'...

:: Validate input
```

```
if "%ENV_NAME%"=="" (
    echo [ERROR] Environment name not specified.
    goto :eof
)

if "%BASE_DIR%"=="" (
    echo [ERROR] Base directory not specified.
    goto :eof
)

:: Check if base directory exists
if not exist "%BASE_DIR%" (
    echo [ERROR] Directory not found: %BASE_DIR%
    goto :eof
)

:: Change to working directory
cd /d "%BASE_DIR%"

:: Define the path to the venv under C:\penv
set "VENV_ROOT=C:\penv"
set "VENV_ACTIVATE=%VENV_ROOT%\%ENV_NAME%\Scripts\activate.bat"

:: Check for the activation script
if exist "%VENV_ACTIVATE%" (
    echo calling activation...
    call "%VENV_ACTIVATE%"
    echo activation was called...
) else (
    echo [ERROR] Virtual environment activation script not found:
%VENV_ACTIVATE%
    goto :eof
)

echo Activation of the virtual environment '\%ENV_NAME%' has ended

echo Launching VSC Starting
code .
echo Launching VSC Ended

endlocal
```