Chem 361: Machine Learning in Chemistry

Xuhui Huang

Spring 2025

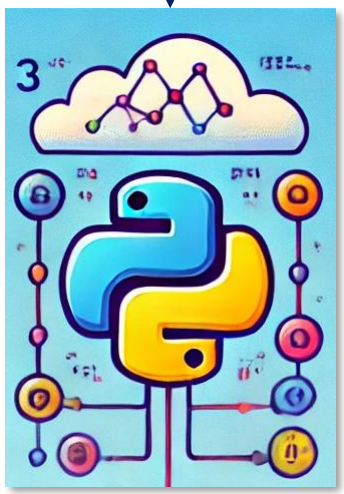
Tutorial:
set up your Python environment for data analysis and ML

Introduction to Python

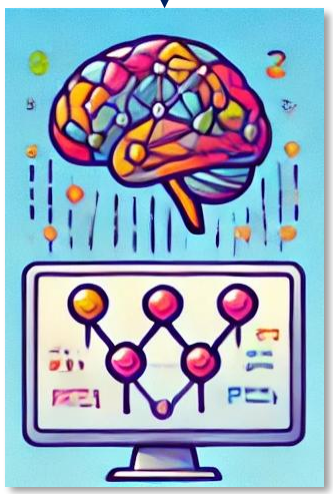
Python is one of the major programming languages with wide applications today

- Simplicity and versatility
- Readability
- Extensive ecosystem

Data Analysis



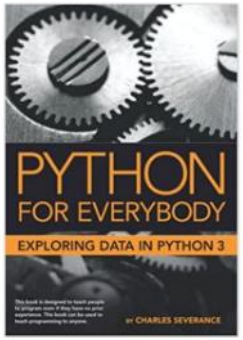
Web Development

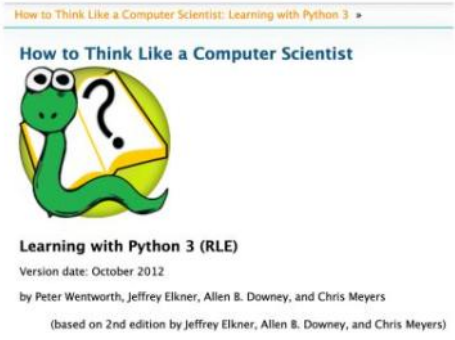


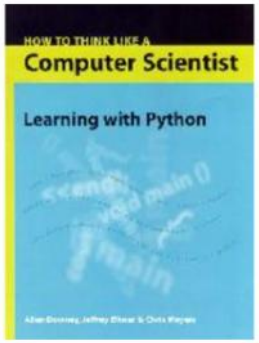
Machine Learning

Free books on using Python

https://bcrf.biochem.wisc.edu/2022/08/23/free-data-science-books/







Package management in Python

(C dependent on B-v1)

Task A: Package A + Package B-v1 → Package C

Task B: Package A + Package D + Package E

Task C: Package G + Package B-v2 → Package F

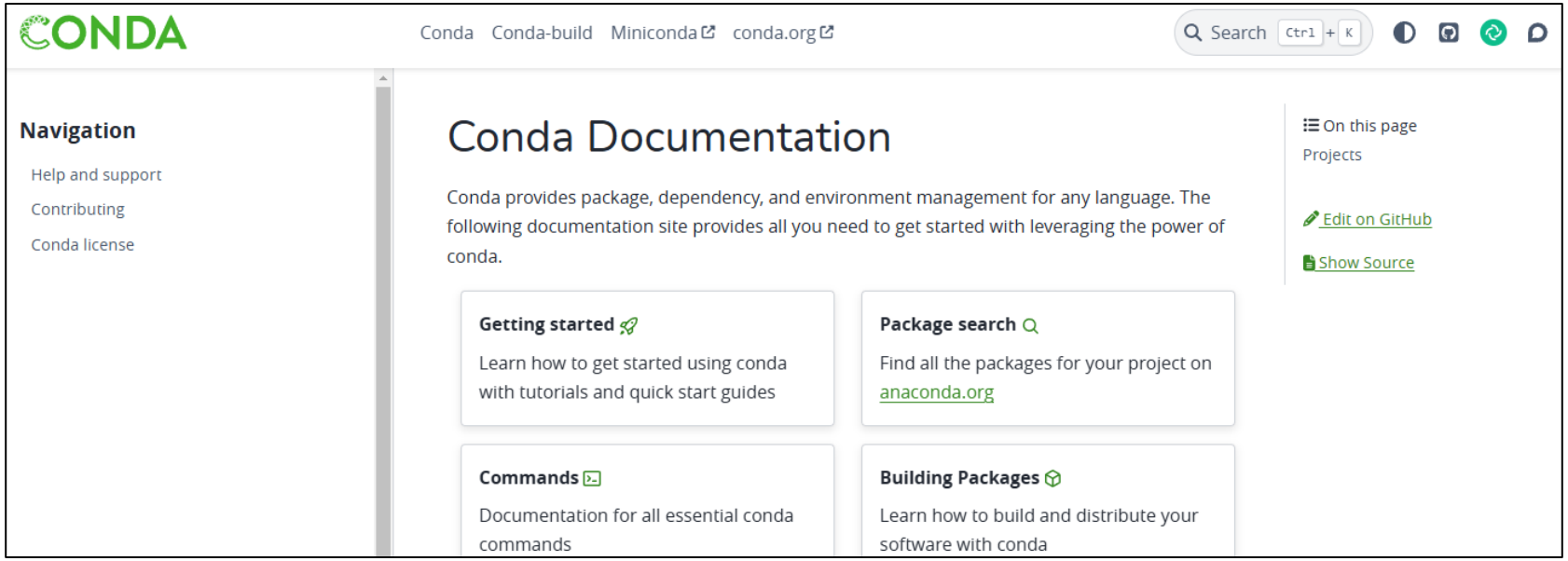
Conflict!

- **Instead …**

We aim to create multiple isolated Python environments on our local desktop, each with its own dedicated packages installed for specific tasks.

Conda: A platform for Python package management and creating isolated Python environments

- Conda is a powerful command line tool for package and environment management that runs on Windows, macOS, and Linux



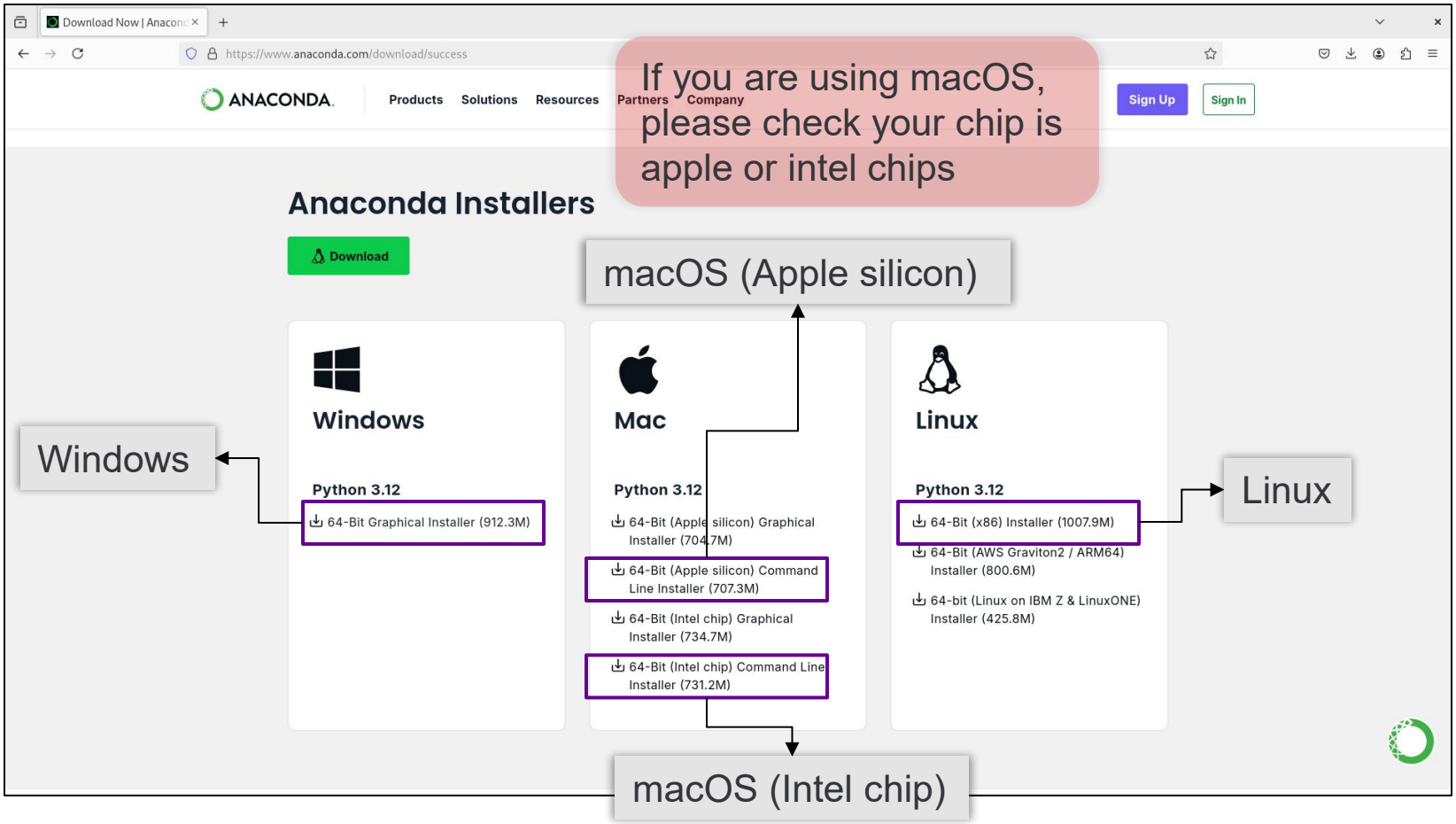
https://docs.conda.io/projects/conda/en/stable/user-guide/getting-started.html

https://docs.conda.io/en/latest/

Part 1:
Install Anaconda, a distribution that includes conda

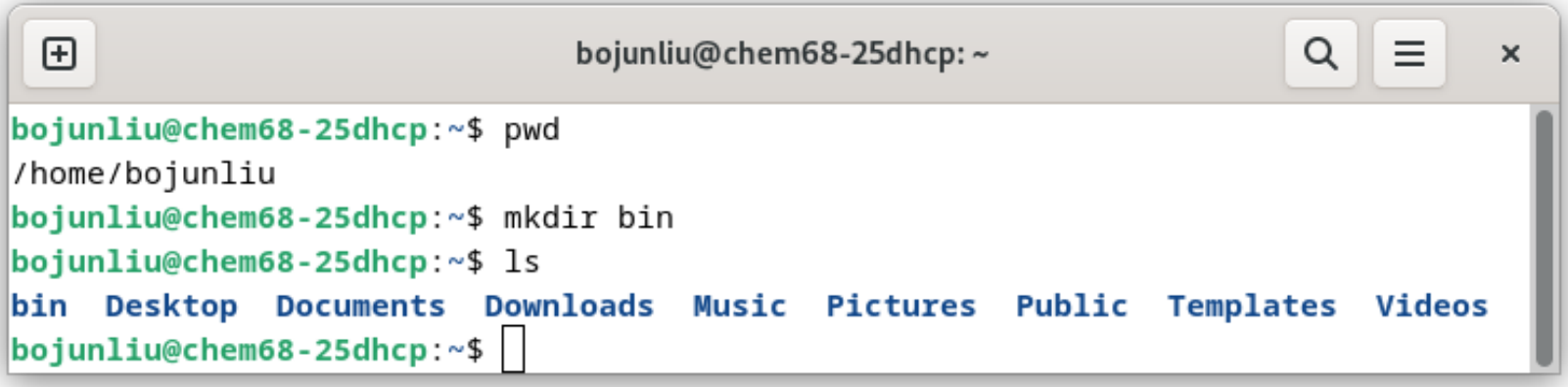
Step1: Download Anaconda installers

- Visit the website https://www.anaconda.com/download/success, and download the appropriate installer for your computer's operating system

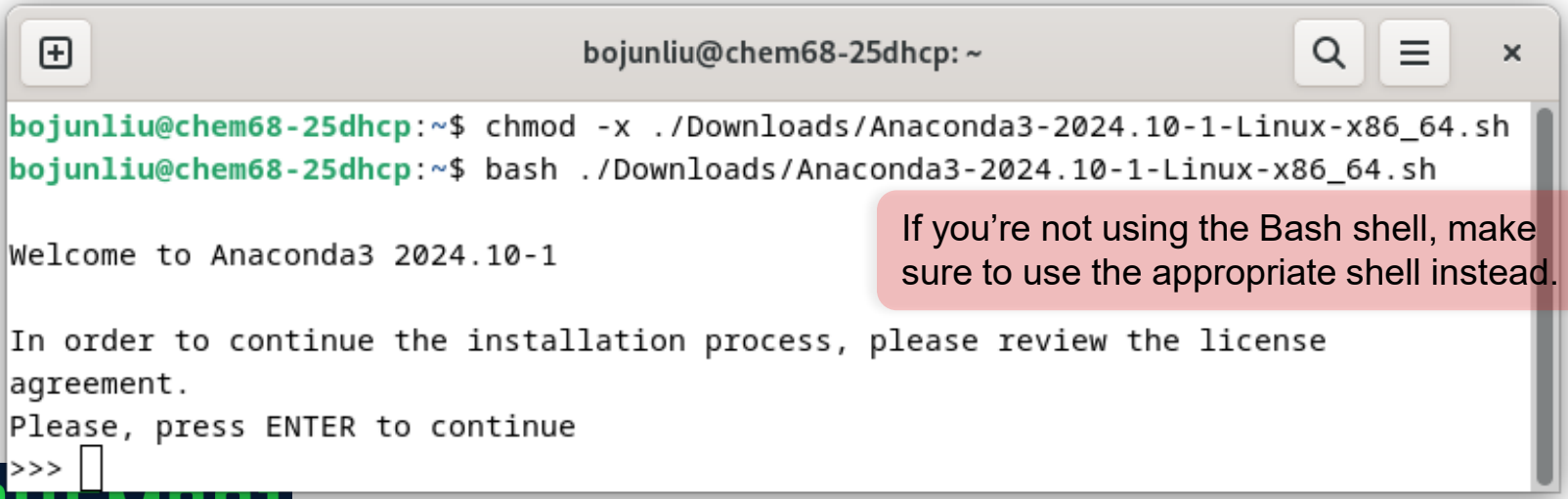


Step2: Install Anaconda - Linux

- In your “/home/your_account_name” directory, create a “bin” folder for installing Anaconda later. (This step is optional, depending on your preference)



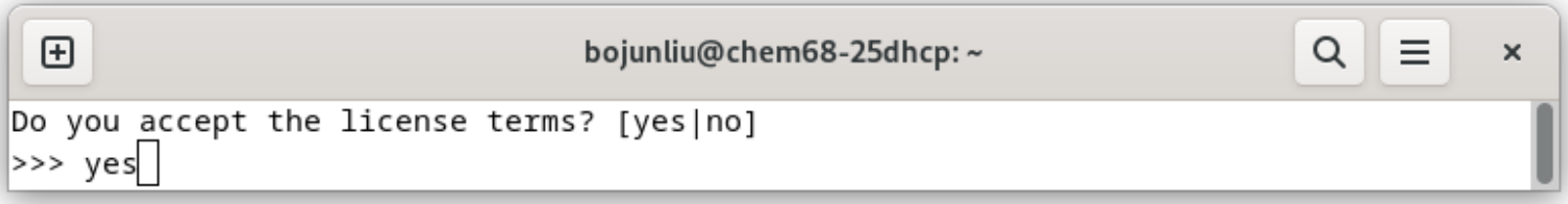
- Find your downloaded Anaconda installer (for me, its in my “./Downloads” directory) and execute the following commands.



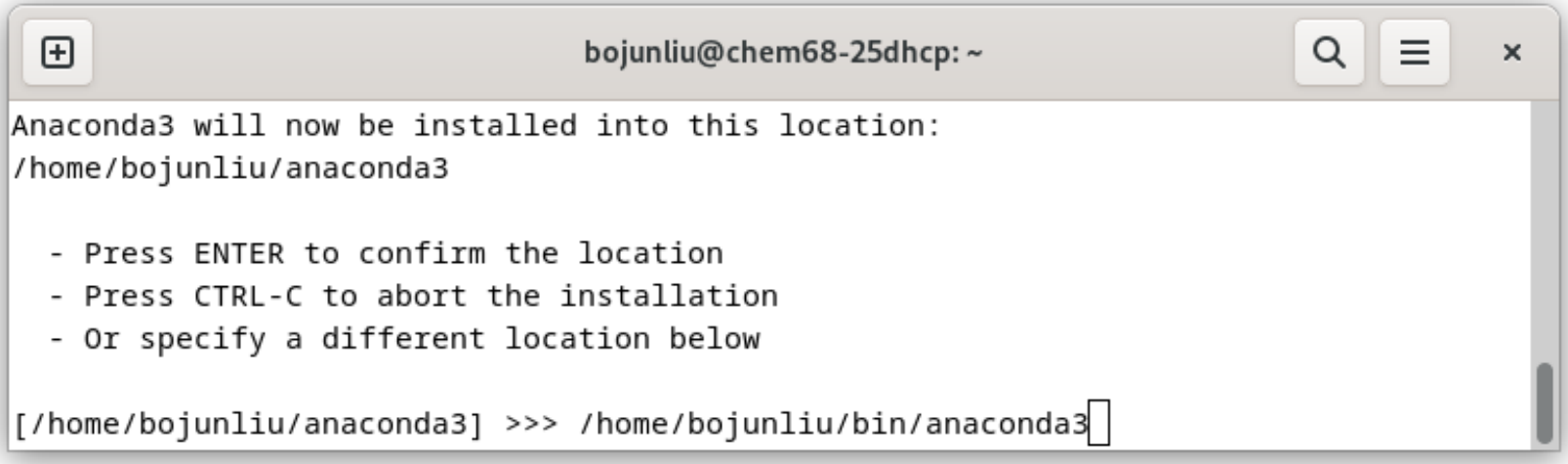
If you’re not using the Bash shell, make sure to use the appropriate shell instead.

Step2: Install Anaconda - Linux

- Follow the instructions (press “q” to continue when you finish reading the license”).

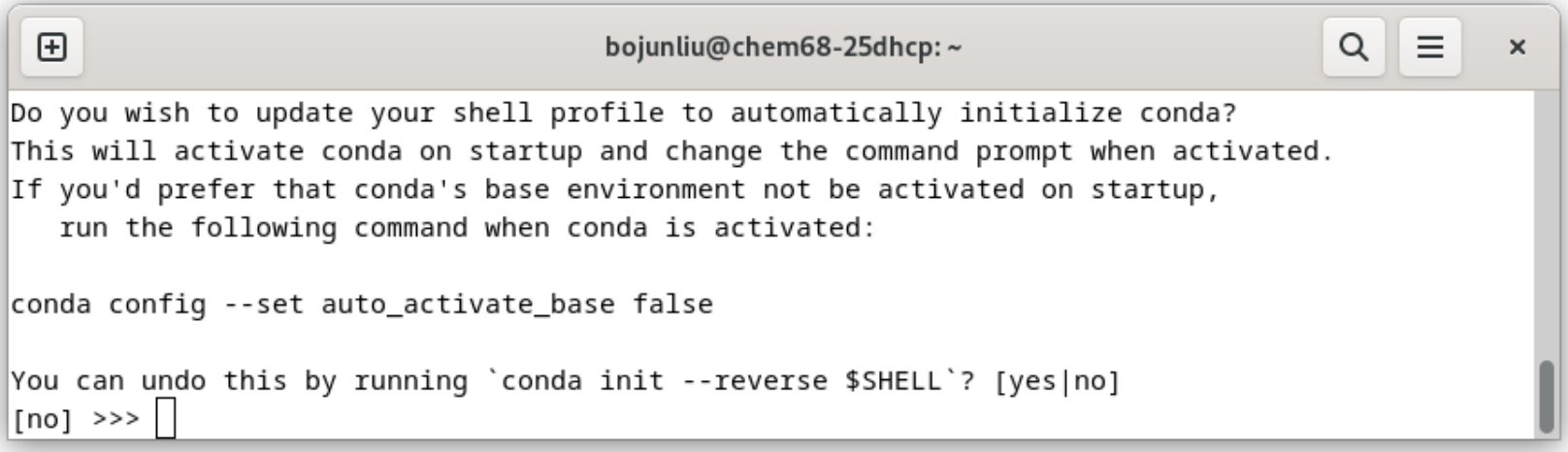


- By default, Anaconda is installed in your “/home/your_account_name” directory. Previously, we have created a “bin” folder, and we can specify our installation location in that folder.

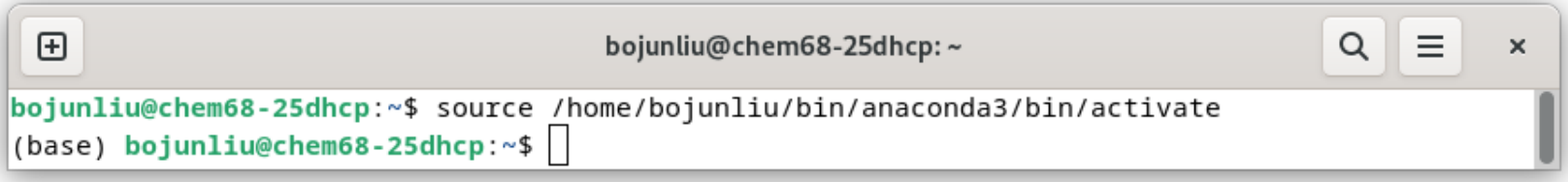


Step2: Install Anaconda - Linux

- Last step:

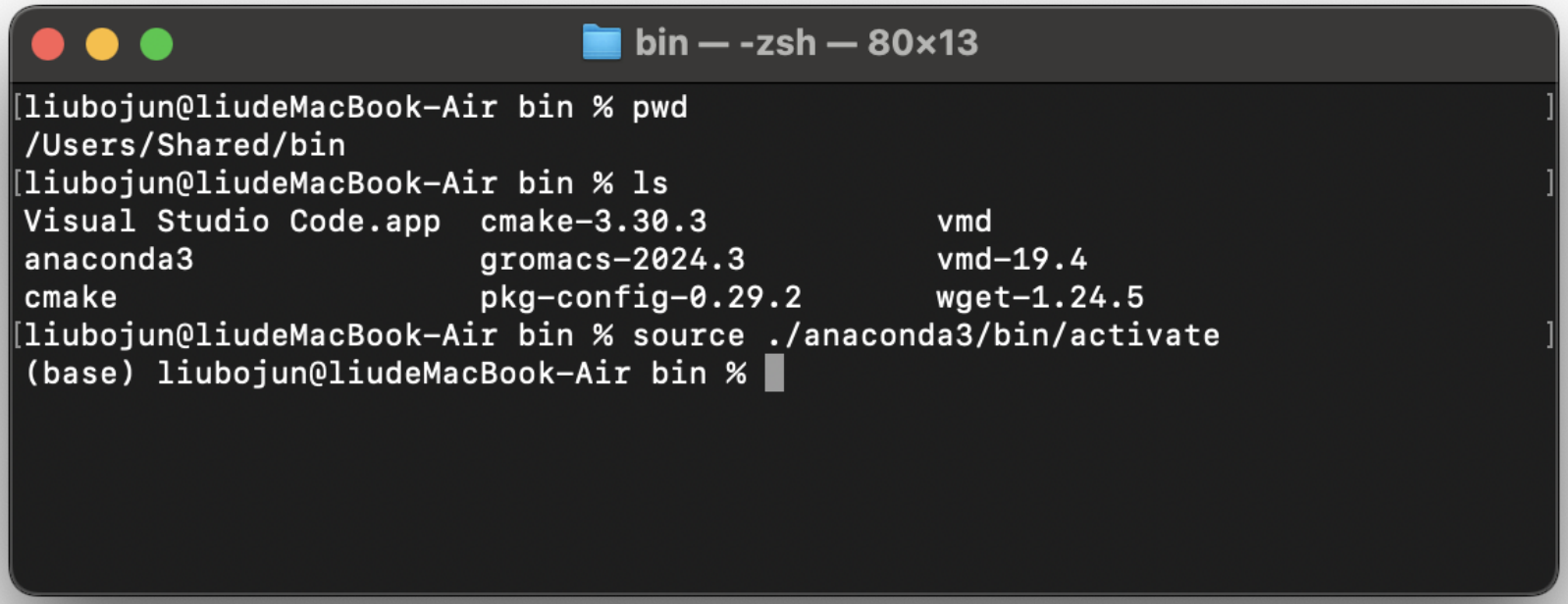


- If you press yes, you are all set! Conda will automatically initialize every time you open a shell.
- If you press no, you will need to activate conda manually using the **source** command. See below. Be sure to adjust the path based on your anaconda3 directory.



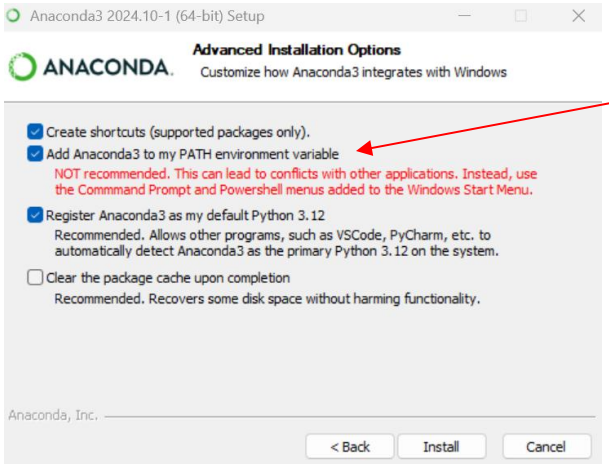
Step2: Install Anaconda - macOS

- On macOS, you can follow the same steps in the Terminal as on Linux. The only difference is where you install Anaconda. Personally, I create a “Shared” folder in the “/Users” directory, then a “bin” folder inside it, and install Anaconda there. This is optional and depends on your preference.



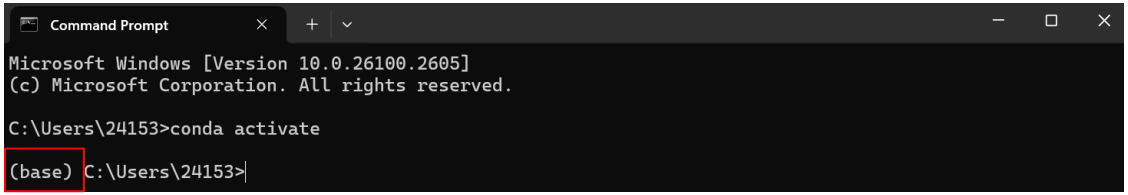
Step2: Install Anaconda - Windows

- Double click the downloaded executive file “Anaconda3-2024.10-1-Windows-x86_64.exe” for installation
- Proceed installation but pay attention to the following step:
  - Select “Add Anaconda3 to my PATH environment variable”, especially when new to Python.



PLEASE SELECT !!!

- After installation: open a Command Prompt terminal (via typing “cmd” in your search bar), and enter “conda activate” in terminal.

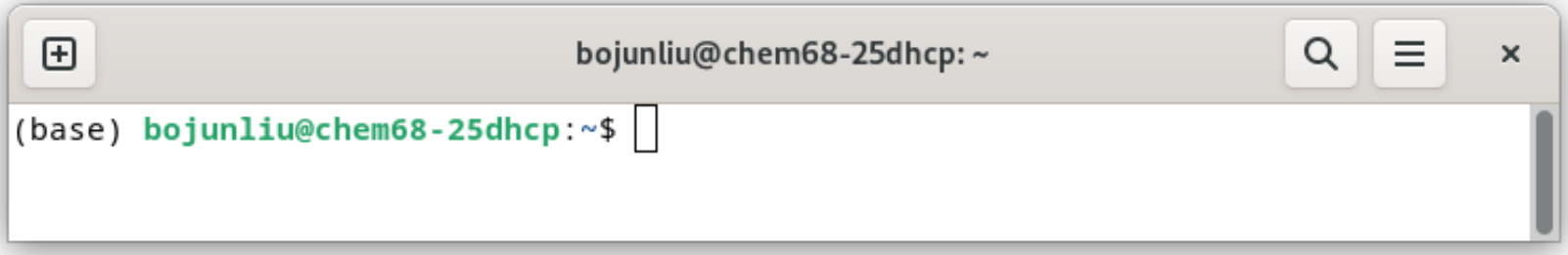


“base” environment will be activated after the “conda activate” command, if installed correctly. Otherwise, let the TA know.

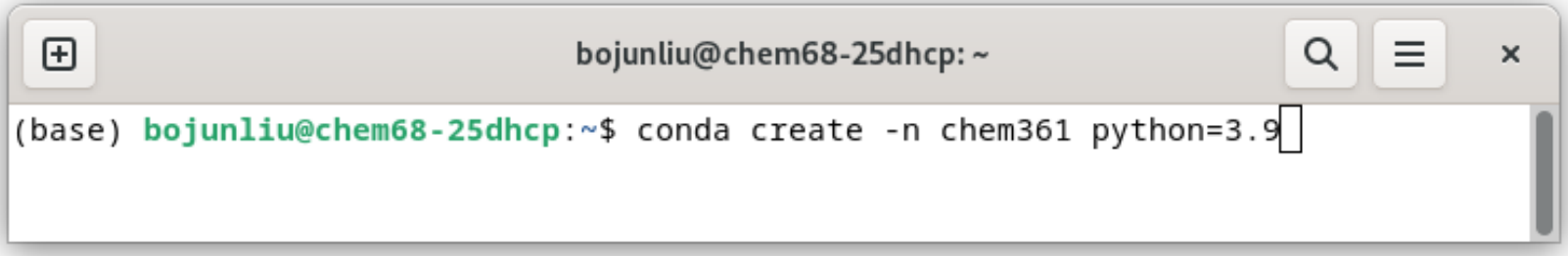
Part 2:
Set up Python environment using Conda

Step3: Create Python (conda) environment

- Now you have successfully installed anaconda3 and opened terminal with “conda” activated.

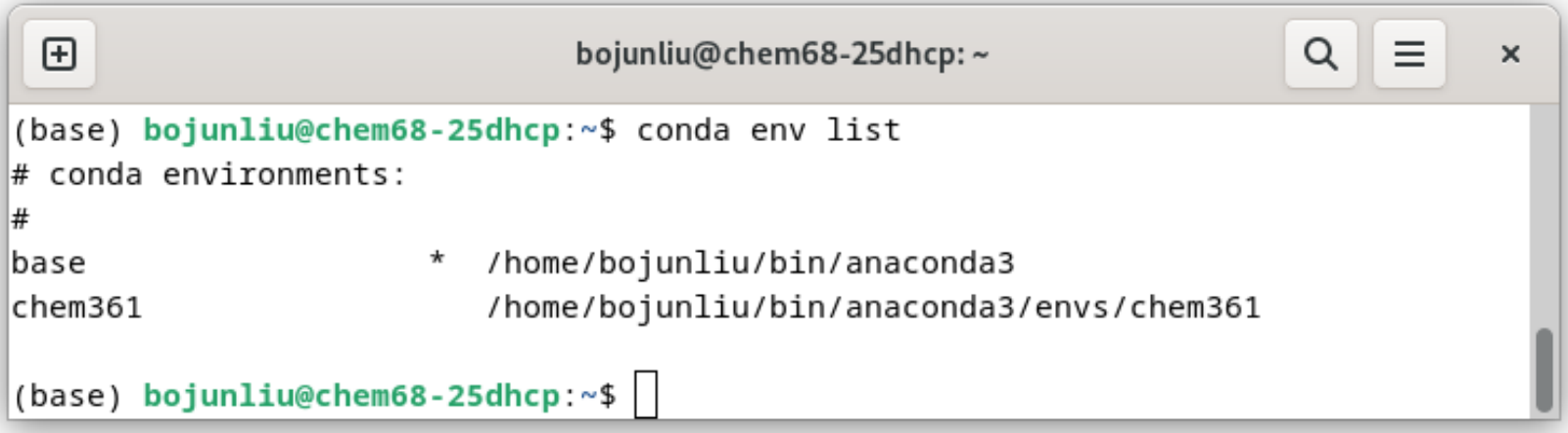


- “(base)” indicates you’re in the base environment. We typically avoid modifying the base environment directly and instead create new Python environments for different tasks.
- To create a Python environment, use the **conda create** command. You can specify the environment’s name with the **-n** flag and set the Python version. For example, the command below creates an environment named “chem361” with Python 3.9 in it.

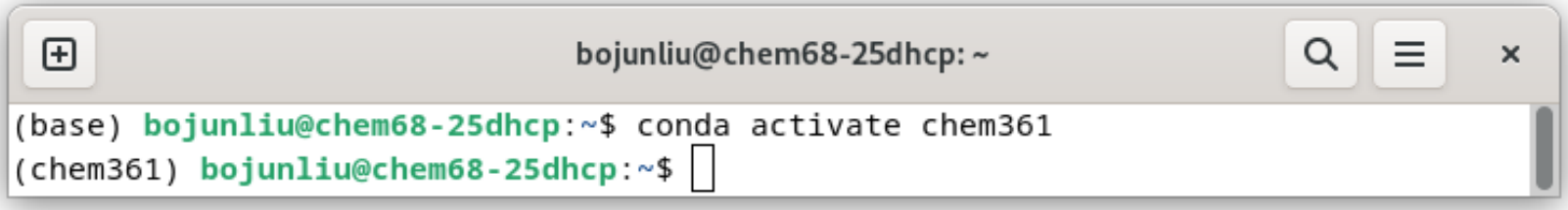


Step3: Create Python (conda) environment

- Now you have successfully created a new Python environment named “chem361”. You can use the **conda env list** command to see your newly created environment.



- However, you're still in the base environment. Run **conda activate chem361** to switch to the newly created environment “chem361”. The prompt “(chem361)” indicates you are now in that environment.



Step4: Install Python packages

- Now that you've created a new Python environment, it only includes Python by default. To use it effectively for data analysis or machine learning, you'll need to install additional Python packages. For example, …

Array-based computing

Scientific computing

Machine Learning

Deep Learning

Data Visualization

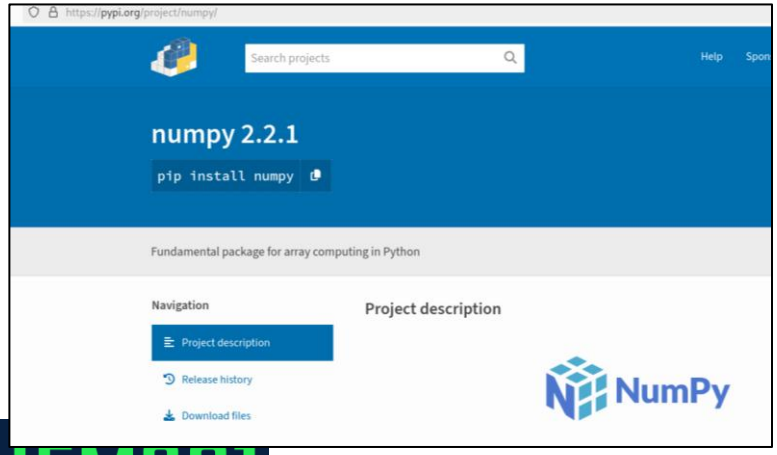
Chem-Informatics

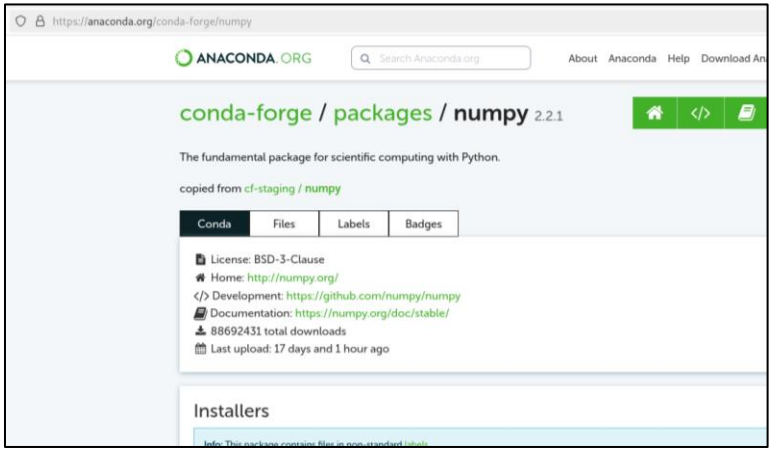
Step4: Install Python packages

- Before we begin, let me introduce two standard methods for installing Python packages.
  - **conda install <pkg_name> = <pkg_version>**
  - **pip install <pkg_name> == <pkg_version>**

If you don't specify the pkg_version, the newest version will be automatically selected.

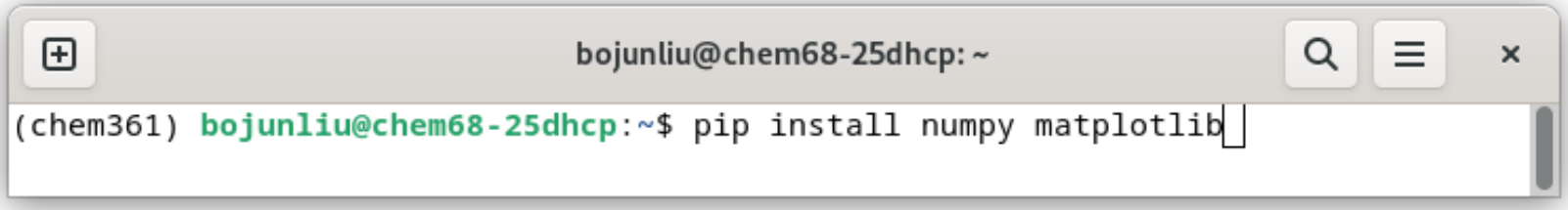
- Both methods can be used to install Python packages. While **pip install** is often simpler and faster, **conda install** can automatically handle dependencies within your environment.
- When you use **pip install**, packages are retrieved from PyPI at https://pypi.org. In contrast, **conda install** pulls packages from the Anaconda repository at https://anaconda.org/anaconda/repo. If you specify **conda install -c conda-forge**, it also searches for packages in the conda-forge channel at https://conda-forge.org/packages/.
- For instance, the package "numpy" can be found in both PyPI and the Anaconda repository.



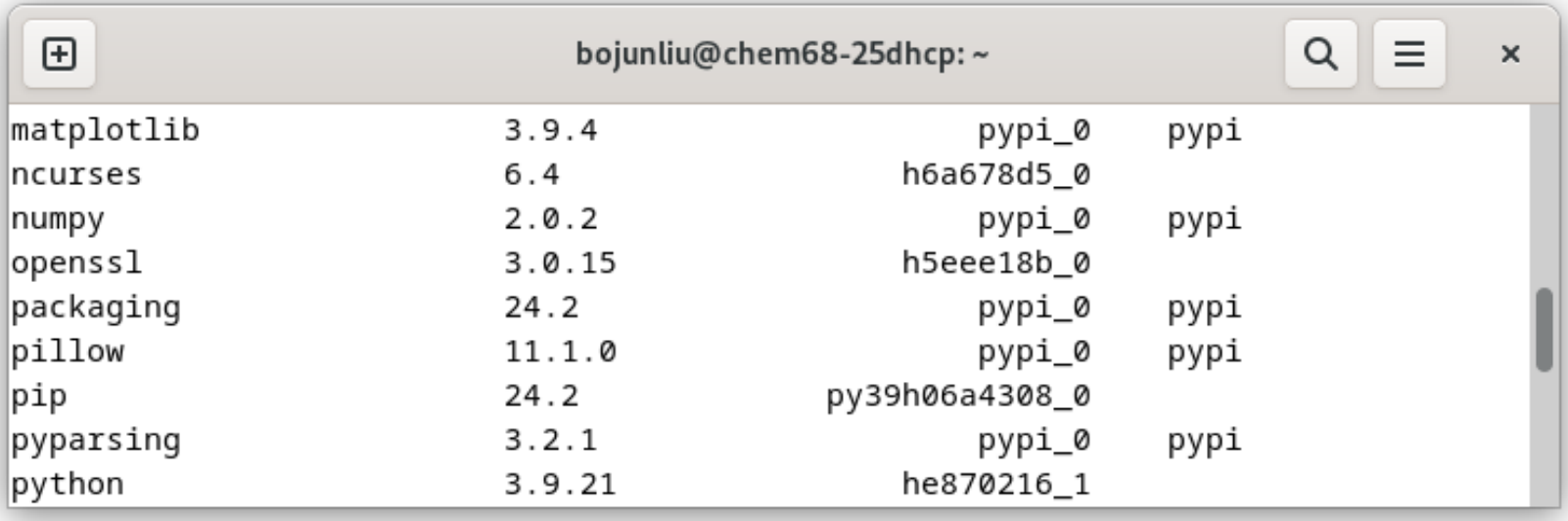


Step4: Install Python packages

- Now let's start installing Python packages. For example, "numpy" and "matplotlib":



- Use the command **conda list** , you will see "numpy" and "matplotlib" have been installed in your Python environment.

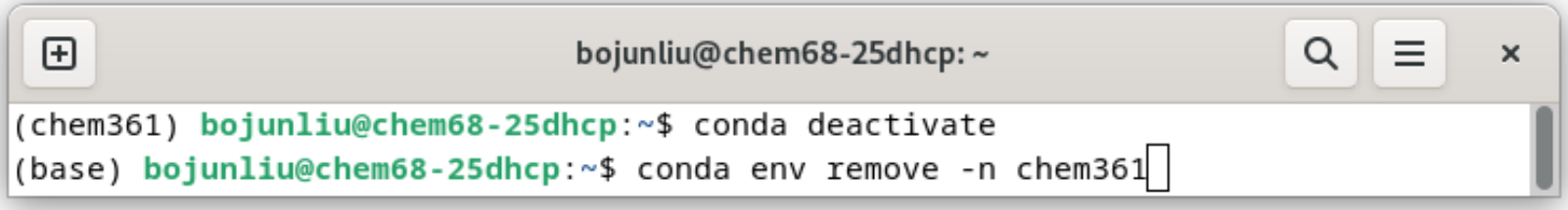


Step4: Install Python packages

- In the future, we will release the required Python dependencies for each homework project.

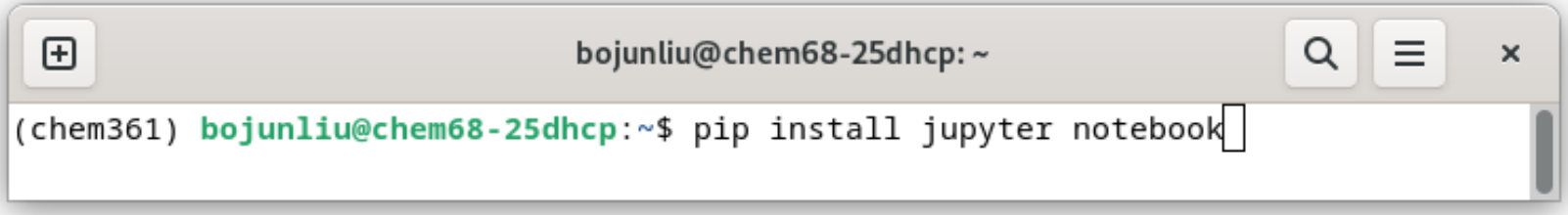
- Be mindful when setting up your Python environment to ensure compatibility among different packages. If you encounter any issues, you can use **pip uninstall <pkg_name>** or **conda uninstall <pkg_name>** (depending on how the package was installed). However, we recommend creating a new environment if problems persist, as this approach often simplifies resolving conflicts.

- When creating another Python environment, remember to run **conda deactivate** to return to the base environment. To remove an environment you previously created, use **conda env remove -n <env_name>** .

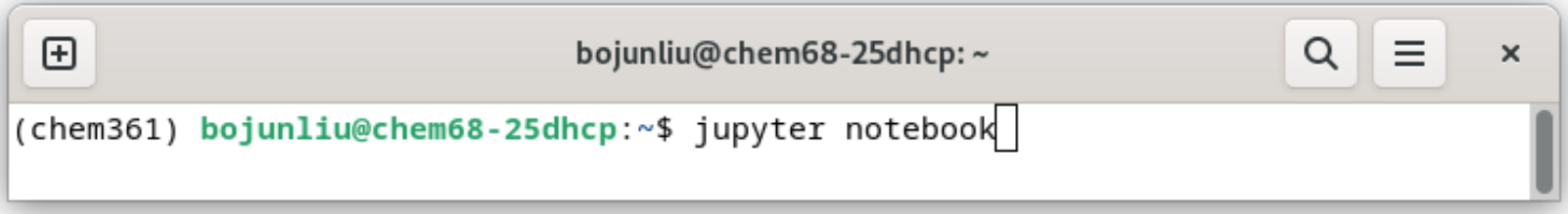


Step5: Run Python code: jupyter notebook

- We recommend using jupyter notebooks to write and run Python code for your projects.
- Firstly, you need to install jupyter, notebook in your Python environment.



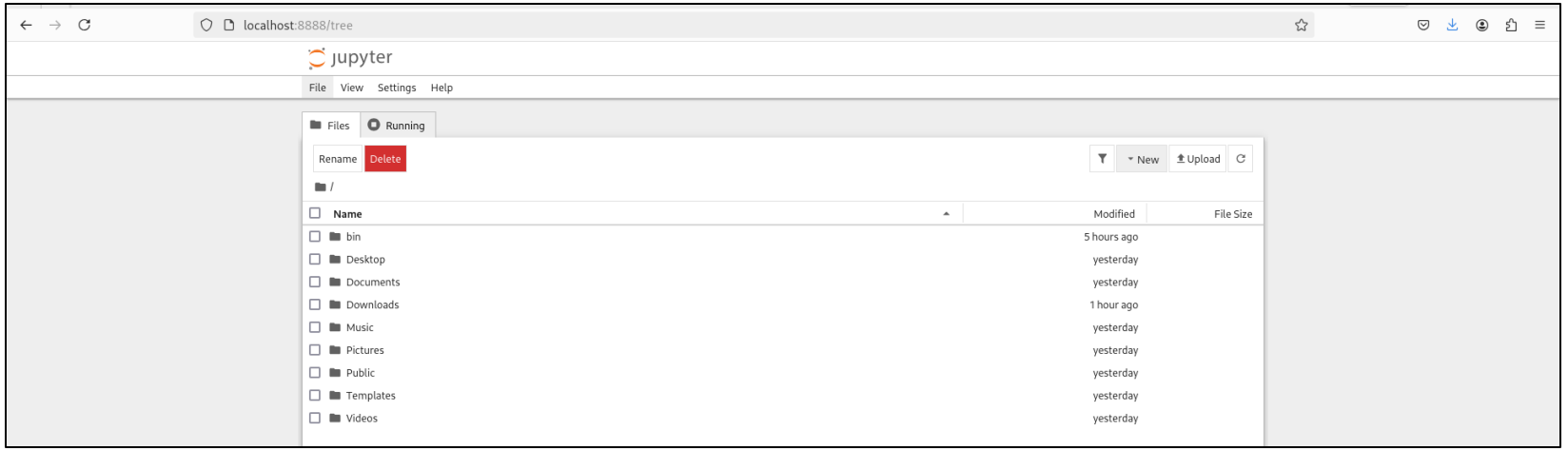
- Then, type **jupyter notebook**.



- The Jupyter server will open in your web browser (press **Ctrl + C** in your terminal to stop it).

Step5: Run Python code: jupyter notebook

- Your home directory will be shown here.



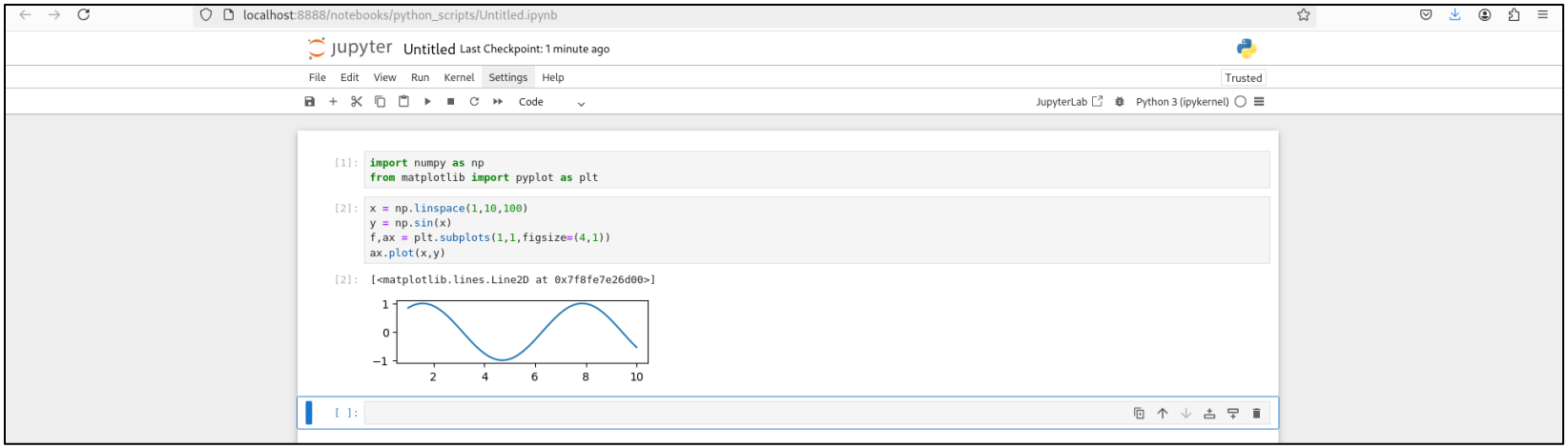
- Create a Python 3 (ipykernel) file.

Step5: Run Python code: jupyter notebook

- Your can start running Python code in this jupyter notebook!

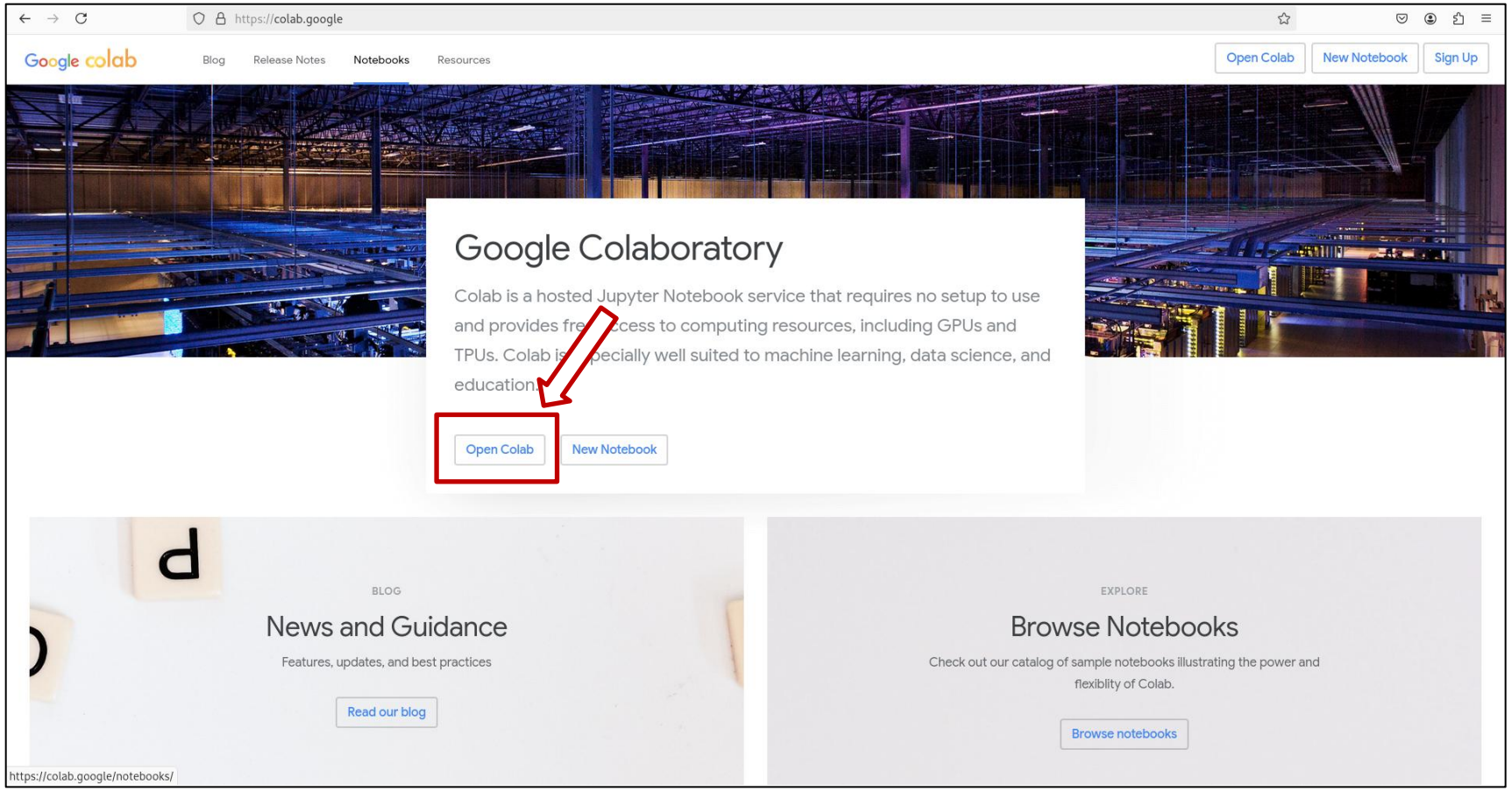
- Visit the official Jupyter Notebook documentation at https://jupyter-notebook.readthedocs.io/en/latest/notebook.html to learn about its full range of features.

- Let's try it out by plotting a sine function. Press **Shift + Enter** to execute each cell.



Part 3:
Use Python in Google Colab

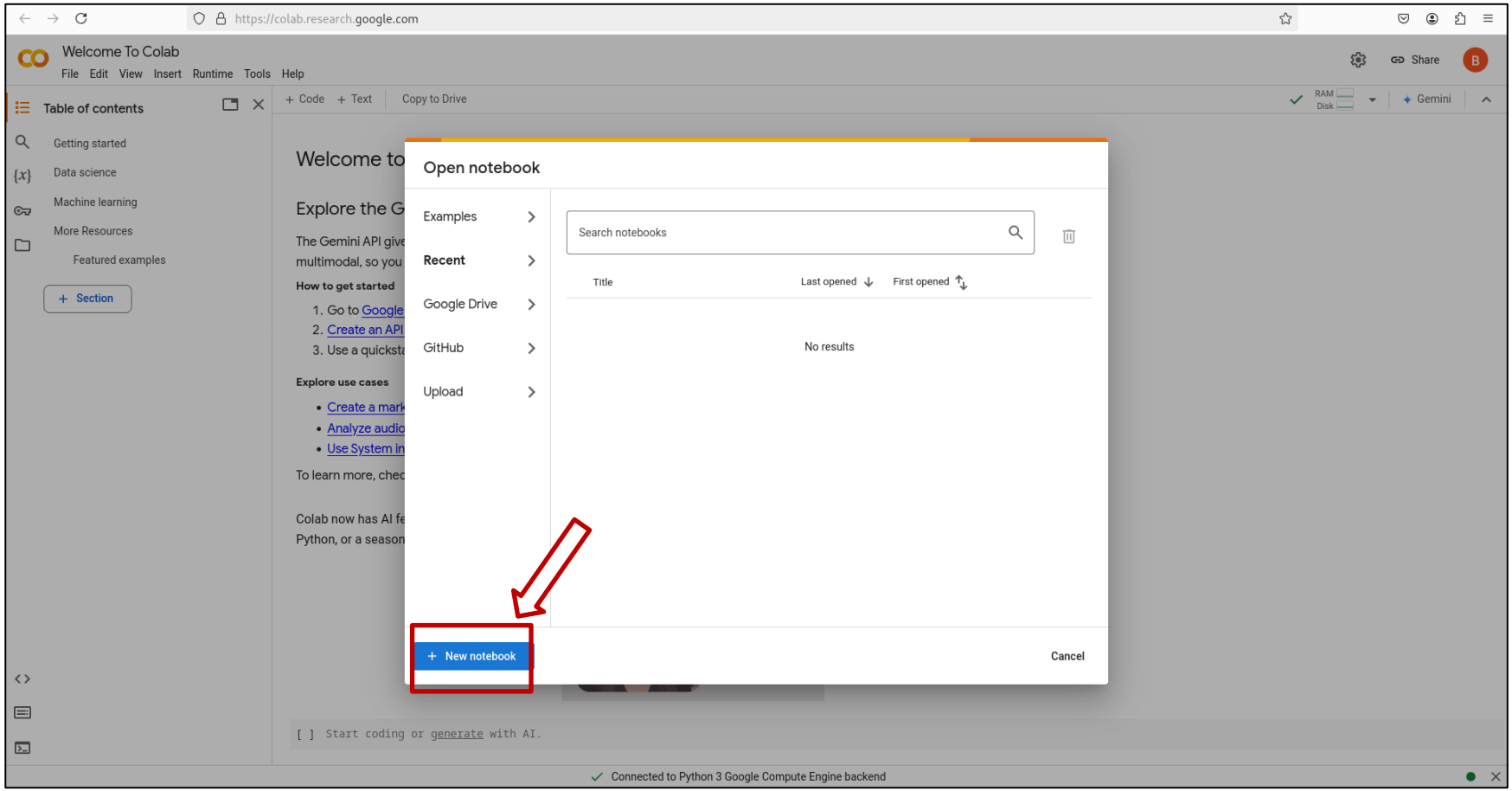
Google Colab: A cloud-based platform for running Python code and collaborating on data science projects in Jupyter notebooks



https://colab.google/

Open a notebook

- Open new notebook.



Check the available Python packages

- The Google Colab server usually comes pre-installed with a variety of Python packages. You can use **!pip list** command to check the already installed Python packages. For example, you can see numpy and matplotlib in our previous example are already installed!

- If you want to install Python packages not being included, you can use **!pip install <pkg_name>** command. The "!" at the beginning is specific to Jupyter Notebook or IPython environments. It allows you to run shell commands directly from a notebook cell.

- You can try our previous example to plot a sine function in Google Colab!