Class Preparation
Introduction to Data Science for Economists

Before we begin class, you need to install Anaconda and some Python packages. This guide will tell you how to do it, step by step.

Contents

1. Folder setup

1. Create a folder ‘Data_Science’ on the desktop. This folder will be used to store data, code, and resources for this class.
    a. Mac users: /Users/[[Your username]]/Desktop/Data_Science
    b. Windows users: C:\Users\[[Your username]]\Desktop\Data_Science
2. Save material for the class in this folder.
    a. You can rename or move this folder after the class ends but we will keep a consistent location and name for now so everyone is working on the same base directory.
3. Create sub-folders for each class session to keep your data and code organized.

2. Install Anaconda

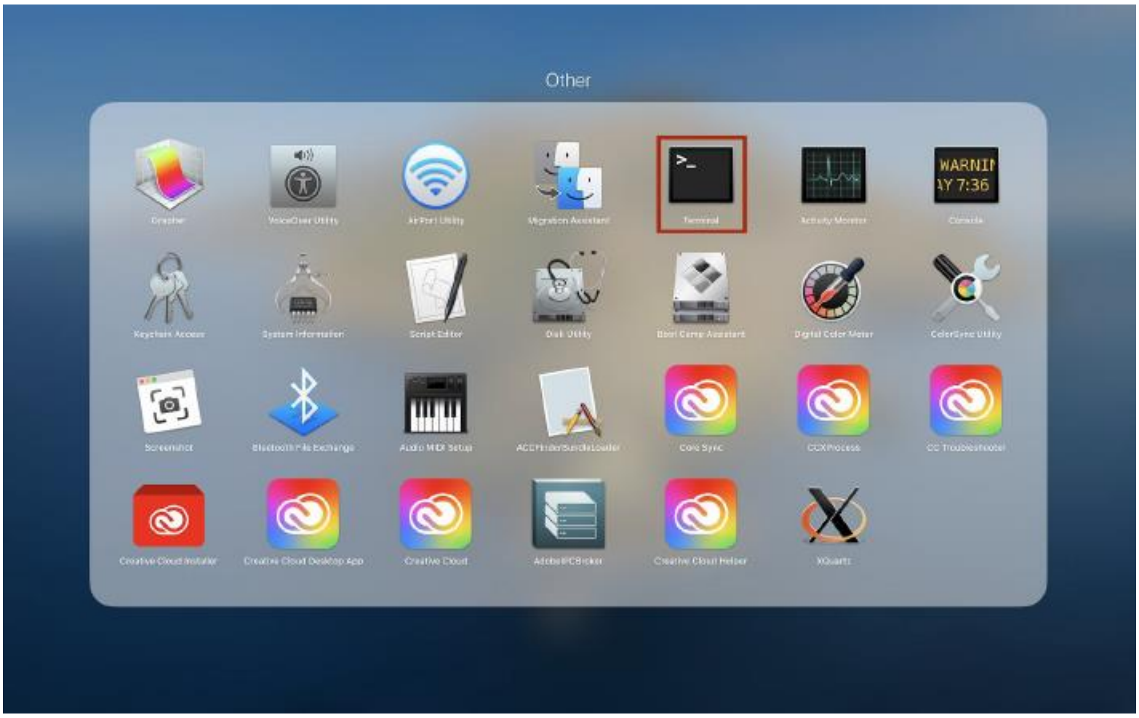
1. What is Anaconda?
    a. **Anaconda** is an open source data science and artificial intelligence distribution platform for Python and R programming languages. It is the most popular way to learn and use Python.
    b. It hosts **Conda**, an open-source, cross-platform, language-agnostic package manager and environment management system. It was originally developed to solve package management challenges faced by Python data scientists, and today is a popular package manager for Python and R. It comes pre-loaded with a large number of libraries and packages.
        i. What is a ‘**package’**? Package are sets of software or code that allow you to do different operations in Python or R. You have to download and load them to be able to use them in your code. ‘**Libraries’** are similar. You can use Conda to install, remove, and upgrade packages.
        ii. What is an ‘**environment’**? An environment is a directory that contains all the files needed for a particular application, such as Python interpreter, packages, and configuration files. It is essentially the set of software and packages you are using to write code. You can use Conda to create separate environments for different projects with different package versions. Note that an environment can also refer to the set of objects you have loaded or created in your code.
    c. What are the advantages of using Anaconda for coding?

        i. It is easy to install and relatively easy to use for installing and managing packages.
        ii. It includes many features and libraries you can use for your projects.
        iii. You can get started quickly with coding using the included Jupyter Notebook.
        iv. You can do many operations through the Anaconda Navigator graphical interface, avoiding the need to use the command line interface.

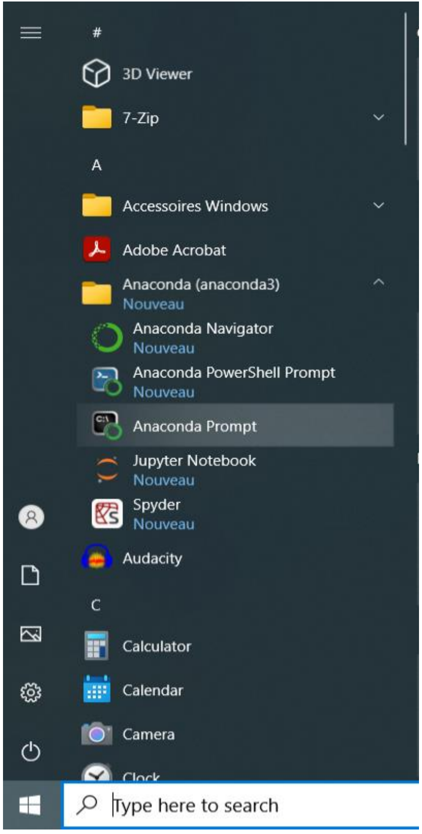
2. Refer to the installation guides:
    a. Mac: https://docs.anaconda.com/anaconda/install/mac-os/
    b. Windows: https://docs.anaconda.com/anaconda/install/windows/
3. Download installers
    a. Go to https://www.anaconda.com/products/individual and click 'DOWNLOAD.'
    b. Download an appropriate graphical installer according to the OS you are using (Mac or Windows). If you are a Windows user, you can check whether your machine is 32-bit or 64-bit following this guide: https://support.microsoft.com/en-us/help/15056/windows-32-64-bit-faq
    c. After completing the download, you can follow the installation guide.
4. Note that Anaconda installs Python automatically! You do not need to install Python separately.

3. Create a new environment

1. Why create a new environment?
    a. You can think of an environment as a workspace for a specific purpose (e.g. a class, a research project, etc.) The reason we recommend you to make a new environment for this class is that there can be some version conflicts among packages you might use for your future projects. Have a set environment ensures consistency across what everyone is working with.
2. We will primarily use a command-line interface for package installation, environment management, and launching python editors.
    a. What is a **command-line interface**? A 'CLI' is a means of interacting with a computer program by inputting lines of text called command-lines. Most computer users rely on graphical user interfaces ("GUIs") instead of CLIs. However, many programs and operating system utilities lack GUIs, and are intended to be used through CLIs.
    b. It may seem daunting but you will only use it for specific tasks with clearly indicated commands.
3. Open a command-line interface.
    a. On Mac: Open Launchpad, then click the Terminal icon

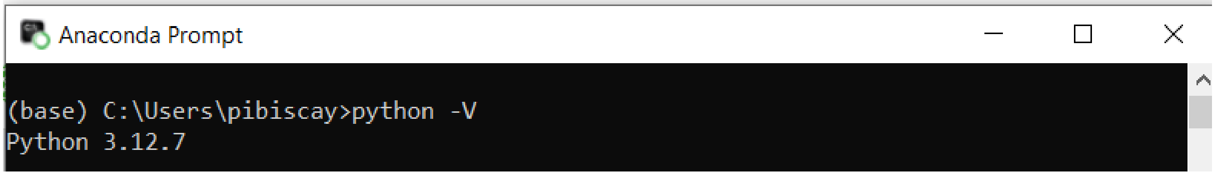


b. On Windows: From the Start menu, search for an open ‘Anaconda Prompt.’



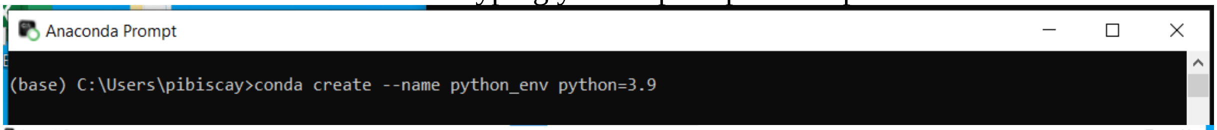
4. You will now see a command interface as below. By default, we are in an environment named ‘base.’ You will also see the default directory.
    a. On Mac, a ‘%’ symbol indicates where you can enter commands at the end of a command line.
    b. On Windows, a ‘>’ symbol indicates where you can enter commands.

5. As a first test of the command line, type *python -V* and click enter to display what version of Python was installed with your Anaconda installation. It will display the response on the next line, before showing another command line.
    a. Note that certain libraries and packages in python that we will use in this class do not work in the newest version of python. For this class, we will use python 3.9 to ensure all packages we want are installable.

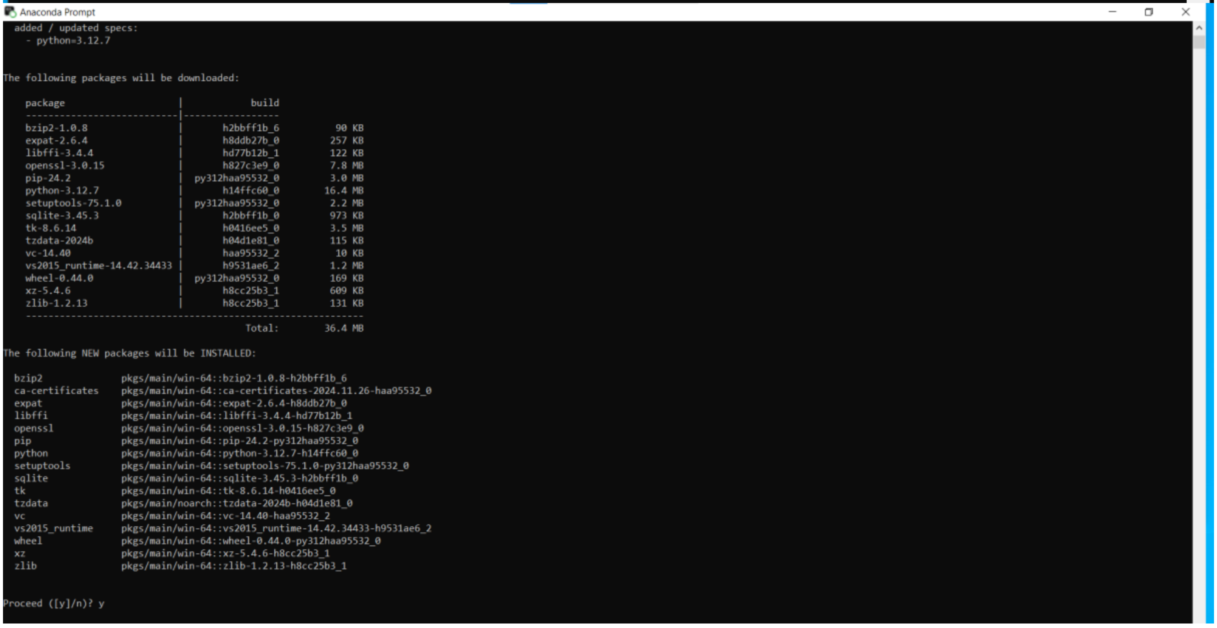


```
(base) C:\Users\pibiscay>python -V
Python 3.12.7
```

6. Create a new environment named 'python_env' by entering the following commands. Type each command at the end of a command line, then press enter and wait for the interface to complete the operation.
    a. *conda create --name python_env python=3.9*
        i. This line creates the environment we will use for this class, and specifies the target python version.
        ii. Proceed with typing y when prompted and press enter

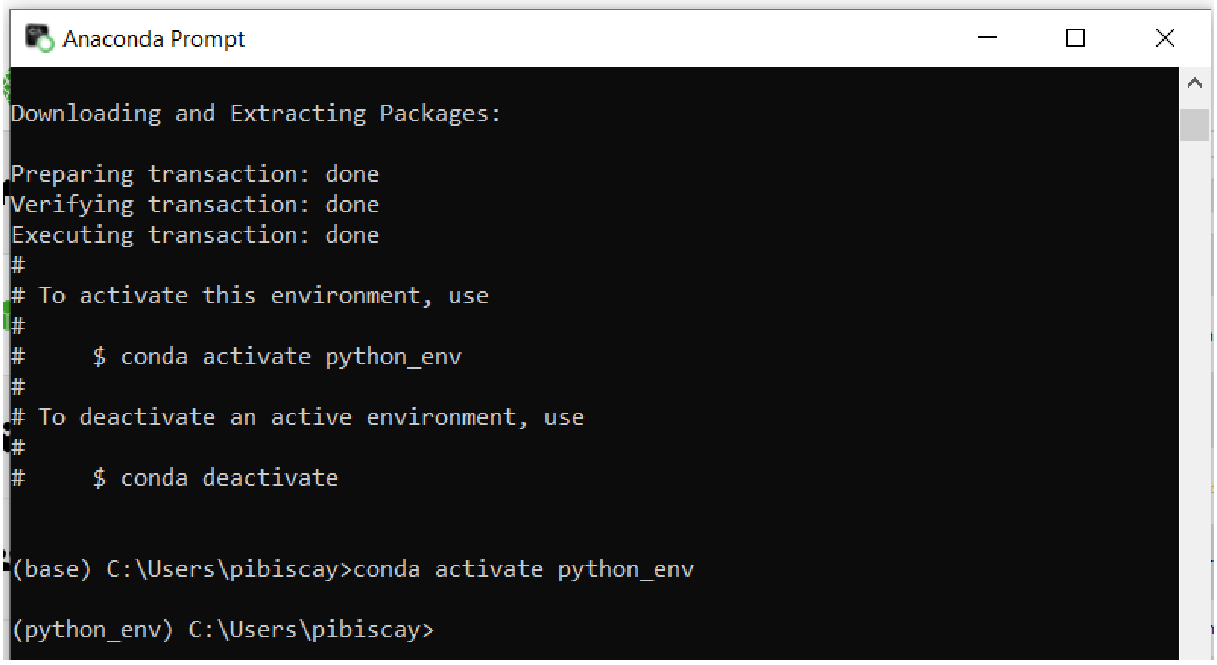


```
(base) C:\Users\pibiscay>conda create --name python_env python=3.9
```



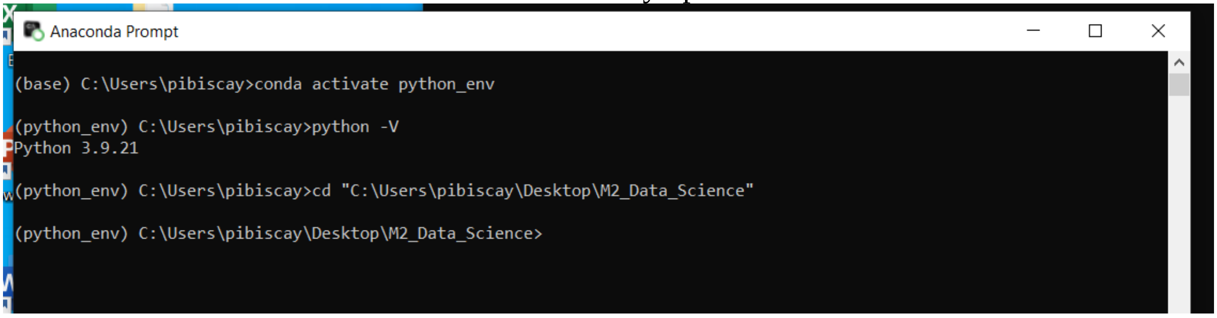
```
  added / updated specs:
    - python=3.12.7


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    bzip2-1.0.8                |       h2bbff1b_6          90 KB
    expat-2.6.4                |       h8ddb27b_0         257 KB
    libffi-3.4.4               |       hd77b12b_1         122 KB
    openssl-3.0.15             |       h827c3e9_0         7.8 MB
    pip-24.2                   |  py312haa95532_0         3.0 MB
    python-3.12.7              |       h14ffc60_0        16.4 MB
    setuptools-75.1.0          |  py312haa95532_0         2.2 MB
    sqlite-3.45.3              |       h2bbff1b_0         973 KB
    tk-8.6.14                  |       h0416ee5_0         3.5 MB
    tzdata-2024b               |       h04d1e81_0         115 KB
    vc-14.40                   |       haa95532_2          10 KB
    vs2015_runtime-14.42.34433 |       h9531ae6_2         1.2 MB
    wheel-0.44.0               |  py312haa95532_0         169 KB
    xz-5.4.6                   |       h8cc25b3_1         609 KB
    zlib-1.2.13                |       h8cc25b3_1         131 KB
    ------------------------------------------------------------
                                           Total:        36.4 MB

The following NEW packages will be INSTALLED:

  bzip2              pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates    pkgs/main/win-64::ca-certificates-2024.11.26-haa95532_0
  expat              pkgs/main/win-64::expat-2.6.4-h8ddb27b_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.15-h827c3e9_0
  pip                pkgs/main/win-64::pip-24.2-py312haa95532_0
  python             pkgs/main/win-64::python-3.12.7-h14ffc60_0
  setuptools         pkgs/main/win-64::setuptools-75.1.0-py312haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                 pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata             pkgs/main/noarch::tzdata-2024b-h04d1e81_0
  vc                 pkgs/main/win-64::vc-14.40-haa95532_2
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.42.34433-h9531ae6_2
  wheel              pkgs/main/win-64::wheel-0.44.0-py312haa95532_0
  xz                 pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
  zlib               pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1


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

    b. *conda activate python_env*
        i. This enters the newly created environment. You will see (python_env) at the start of the command line instead of (base).
        ii. Check the python version is 3.9 as desired.



```
Downloading and Extracting Packages:

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


(base) C:\Users\pibiscay>conda activate python_env

(python_env) C:\Users\pibiscay>
```

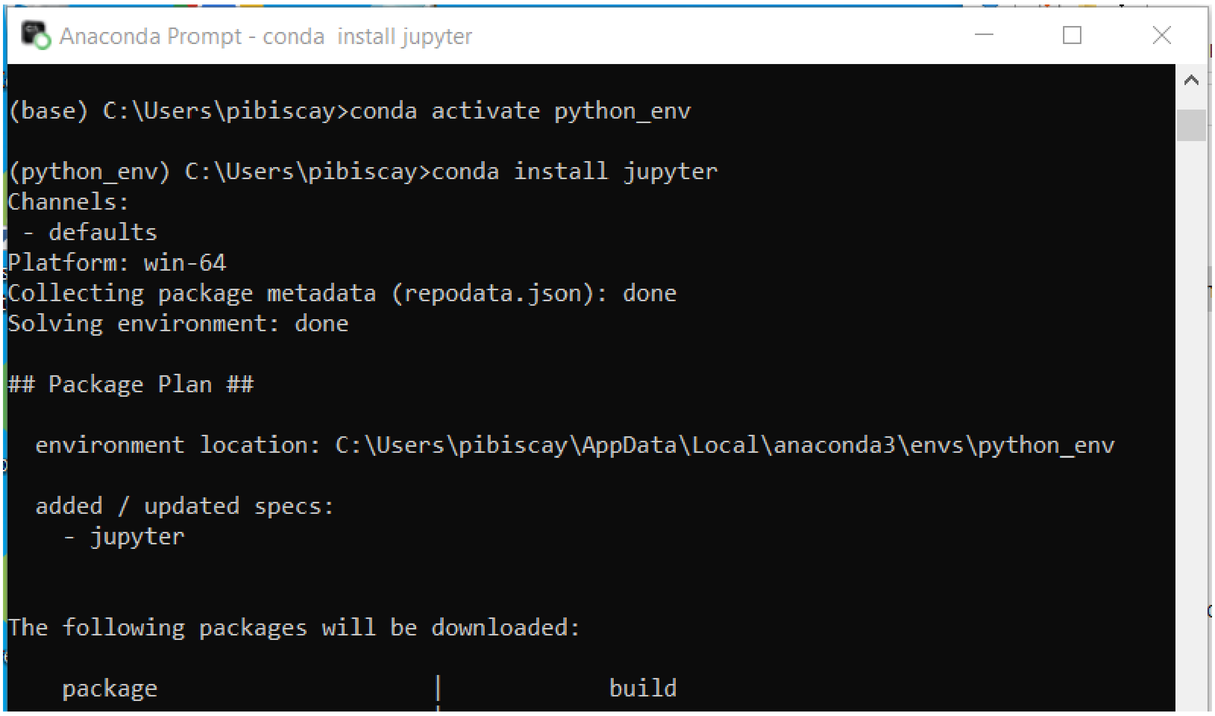
7. Set the working directory to the class folder you set up in step 1
   a. *cd "[[your path]]"*
   b. For example, [[your path]]= C:\Users\[[Your username]]\Desktop\M2_Data_Science
      i. You should see the directory update in the command line



```
(base) C:\Users\pibiscay>conda activate python_env

(python_env) C:\Users\pibiscay>python -V
Python 3.9.21

(python_env) C:\Users\pibiscay>cd "C:\Users\pibiscay\Desktop\M2_Data_Science"

(python_env) C:\Users\pibiscay\Desktop\M2_Data_Science>
```

4. Install some libraries/packages

1. Type the following code to install selected libraries/packages we will use in this class. We will install more packages later on. As with the above, type the code in the command line and press Enter, wait for the operation to proceed, and type 'y' then Enter whenever prompted to proceed.
   a. Note that installing packages uses Conda, the package manager that comes pre-loaded with Anaconda.
   b. Make sure to activate your environment before installing the packages, so that they are all saved to 'python_env' for future use.
2. *conda install jupyter*
   a. This package is useful for working in Jupyter Notebooks, which we will use in this class.



```
(base) C:\Users\pibiscay>conda activate python_env

(python_env) C:\Users\pibiscay>conda install jupyter
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\pibiscay\AppData\Local\anaconda3\envs\python_env

  added / updated specs:
    - jupyter


The following packages will be downloaded:

    package                    |            build
```

3. Additional libraries.
    a. Make sure to run these **one line at a time**! Be careful to check for any typos.
    b. Only run the next line after the given line has completed the installation, including saying “y” when prompted.
    c. These may take some time to all install, while some others may say all requested packages have already been installed (they may be required for earlier installations, for example).
    d. If errors are returned, Google “conda install [library_name]” and look for information for the specific working to use to call the installation in the command prompt interface.
    e. There will likely be other libraries to install during the class. Following similar language to install them to your class python environment.

```
conda install gdal
conda install pandoc
conda install pygeos --channel conda-forge
conda install numpy pandas matplotlib shapely geopandas -c conda-forge
conda install -c conda-forge nodejs
conda install -c conda-forge jupyterlab
conda install -c conda-forge rasterstats
conda install -c plotly plotly
conda install --channel conda-forge esda
conda install geopy -c conda-forge
conda install contextily -c conda-forge
conda install mapclassify -c conda-forge
conda install networkx -c conda-forge
conda install conda-forge::r-stargazer
conda install seaborn
conda install statsmodels
conda install xarray
conda install netCDF4
conda install rasterio
```

4. Some libraries have different dependencies. This is particularly the case for some of the libraries we will use for machine learning activities. To prevent issues across libraries, we will create a **separate environment** for machine learning libraries.
    a. Create an environment python_ml: *conda create --name python_ml python=3.9*
    b. Activate the environment
    c. Install libraries; many other libraries are installed as part of these libraries due to internal dependencies

```
conda install jupyter
conda install scipy
conda install scikit-image
conda install numpy pygeos shapely pandas geopandas -c conda-forge
conda install tpot
conda install seaborn
conda install scikit-learn
conda install plotly::plotly
conda install plotly::plotly_express
pip install NLTK
pip install transformers
pip install spaCy
python -m spacy download en_core_web_sm
pip install genism
pip install torch torchvision
```

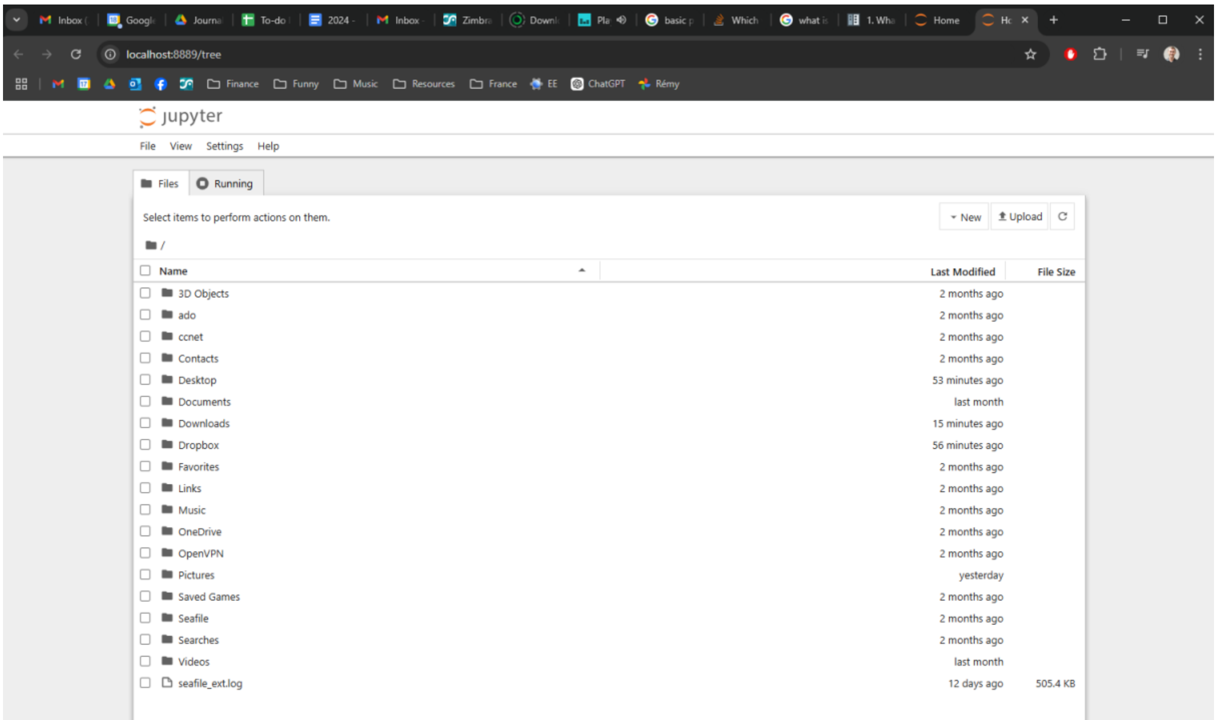
5. Launch Jupyter Notebook

1. What is **Jupyter Notebook**?
    a. Notebook documents (or "notebooks", all lower case) are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc…). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc..) as well as executable documents which can be run to perform data analysis.
    b. The **Jupyter Notebook App** is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.
    c. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a "**Dashboard**" (Notebook Dashboard), a "control panel" showing local files and allowing to open notebook documents or shutting down their kernels.
        i. The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels

        (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files.

    d. A notebook **kernel** is a "computational engine" that executes the code contained in a Notebook document. The ipython kernel, which we will use in this class, executes python code. Kernels for many other languages exist.
    e. When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results.

2. Why use Jupyter Notebook?
    a. You may run Python codes using Mac terminal, Anaconda prompt, etc., however, Jupyter notebook might be one of the most convenient ways for coding in Python.
    b. The fact that it allows for both human-readable text and code and shows analysis outputs make it very easy to navigate and read.
    c. It is similar in many ways to R Markdown but even more easy to use (though with some differences in functionality, e.g., whether you want to hide certain chunks of code/results). It also incorporates Latex for editing math and special characters. Further, it allows you to easily export your code and results as a readable PDF.
3. How to launch Jupyter Notebook?
    a. From the Anaconda prompt/Terminal
        i. Open the command line interface
        ii. *cd "[[your class folder directory]]"*
        iii. *conda activate python_env*
        iv. *jupyter notebook*
    b. From the Anaconda Navigator
        i. Go to the Environments tab and click on python_env
        ii. Go to Home, find Jupyter Notebook, and click Launch
    c. Both approaches will open a new tab on your default web browser.
    d. You will see the Notebook Dashboard, which will allow you to navigate your folders and create a new notebook.

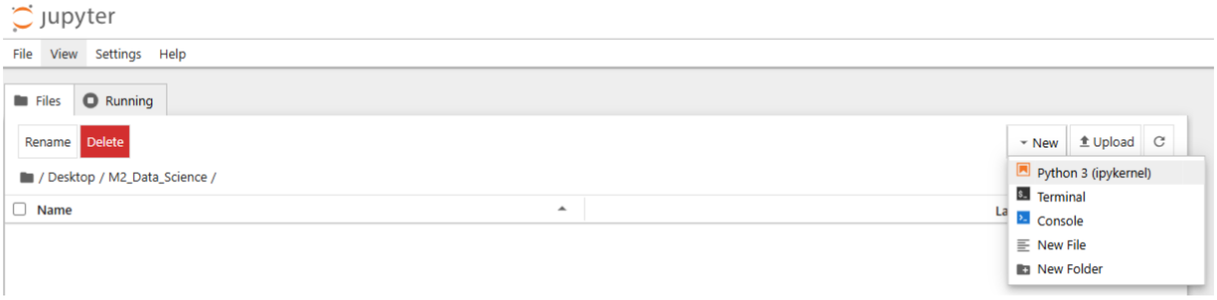


Select items to perform actions on them.

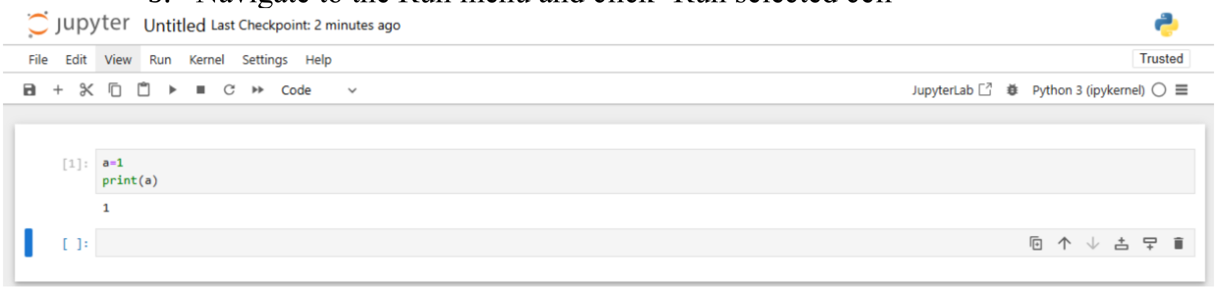
■ /

Name	Last Modified	File Size
3D Objects	2 months ago	
ado	2 months ago	
ccnet	2 months ago	
Contacts	2 months ago	
Desktop	53 minutes ago	
Documents	last month	
Downloads	15 minutes ago	
Dropbox	56 minutes ago	
Favorites	2 months ago	
Links	2 months ago	
Music	2 months ago	
OneDrive	2 months ago	
OpenVPN	2 months ago	
Pictures	yesterday	
Saved Games	2 months ago	
Seafile	2 months ago	
Searches	2 months ago	
Videos	last month	
seafile_ext.log	12 days ago	505.4 KB

4. Create a New Jupyter Notebook in the class folder you created in Step 1 of this document
    a. Click on the 'Desktop' folder, then on 'M2_Data_Science'
    b. On the top right, click on New, and then Python 3 (ipykernel) to create a new Jupyter Notebook using the python kernel.
        i. Note that you can also run R in Jupyter Notebook with an R kernel, but you will have to install this using Conda first.



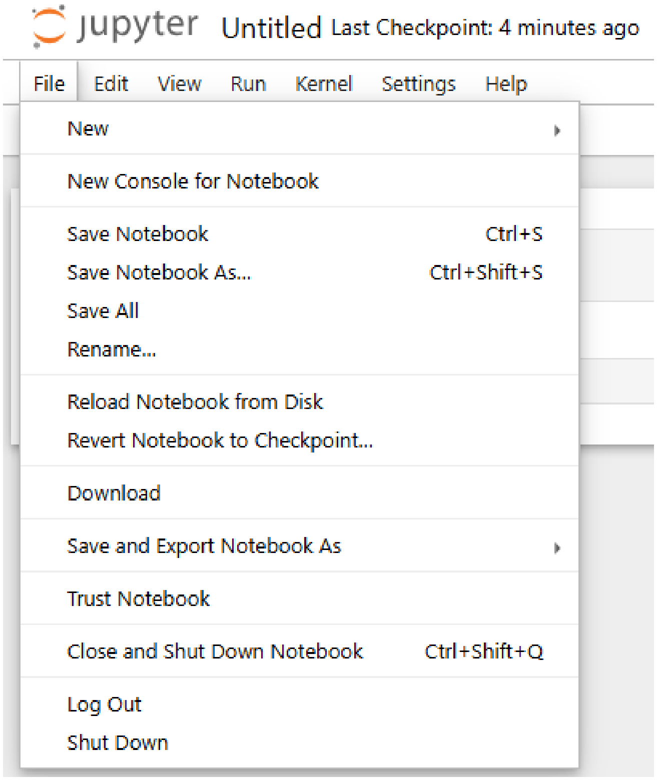
    c. A new tab will open with a blank cell.
5. Let's try our first lines of code!
    a. Click inside the cell and type the following:
        i. *a=1*
        ii. *print(a)*
    b. Navigate to the Run menu and click 'Run selected cell'


jupyter Untitled Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

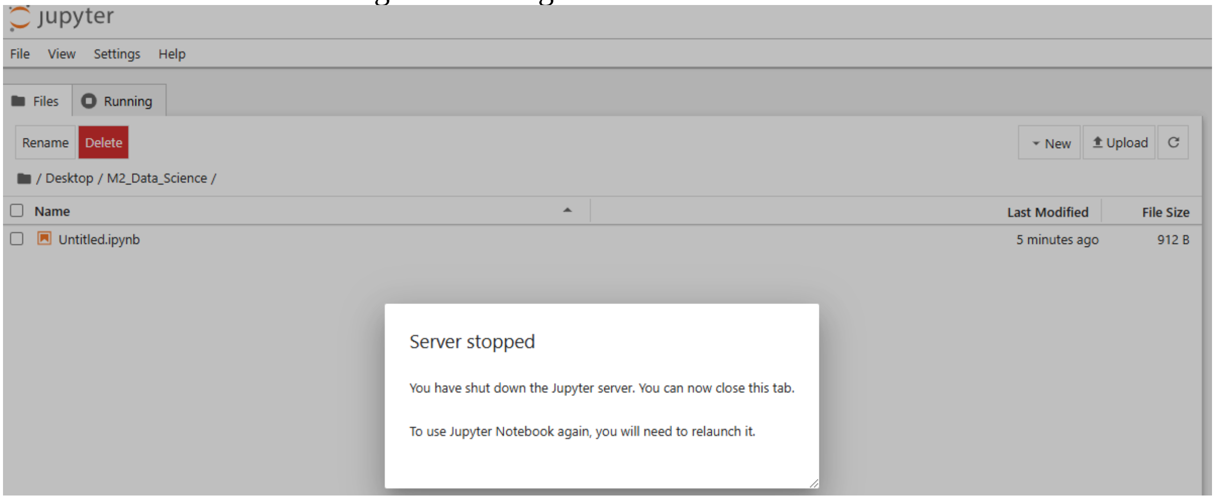
```
[1]: a=1
     print(a)

     1

[ ]:
```

6. Close out of Jupyter Notebook.
    a. If you wish, give the file a name and save it for future practice.

b. Go to the File menu and click ‘Close and shut down notebook.’
   i. Click OK when prompted.
   ii. This will automatically close the Notebook tab.
   iii. It is important to always do this when you are done working with a notebook to ensure your work is saved and your kernel is shut down correctly. Failure to do this can lead to problems using the kernel in the future.


jupyter
Untitled Last Checkpoint: 4 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
New
New Console for Notebook
Save Notebook Ctrl+S
Save Notebook As... Ctrl+Shift+S
Save All
Rename...
Reload Notebook from Disk
Revert Notebook to Checkpoint...
Download
Save and Export Notebook As
Trust Notebook
Close and Shut Down Notebook Ctrl+Shift+Q
Log Out
Shut Down

c. Go to the Jupyter Dashboard Tab. Go to the File menu and click ‘Shut down’. Click “Shut down” again when prompted.
   i. This shuts down the server hosting your Jupyter Notebook session.
   ii. It is important to always go through this Shut down step, otherwise even if you close the Jupyter Dashboard tab the server will keep running in the background.



Server stopped

You have shut down the Jupyter server. You can now close this tab.

To use Jupyter Notebook again, you will need to relaunch it.

Sources

https://blog.hubspot.com/website/anaconda-python
https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
https://en.wikipedia.org/wiki/Command-line_interface
https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html