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

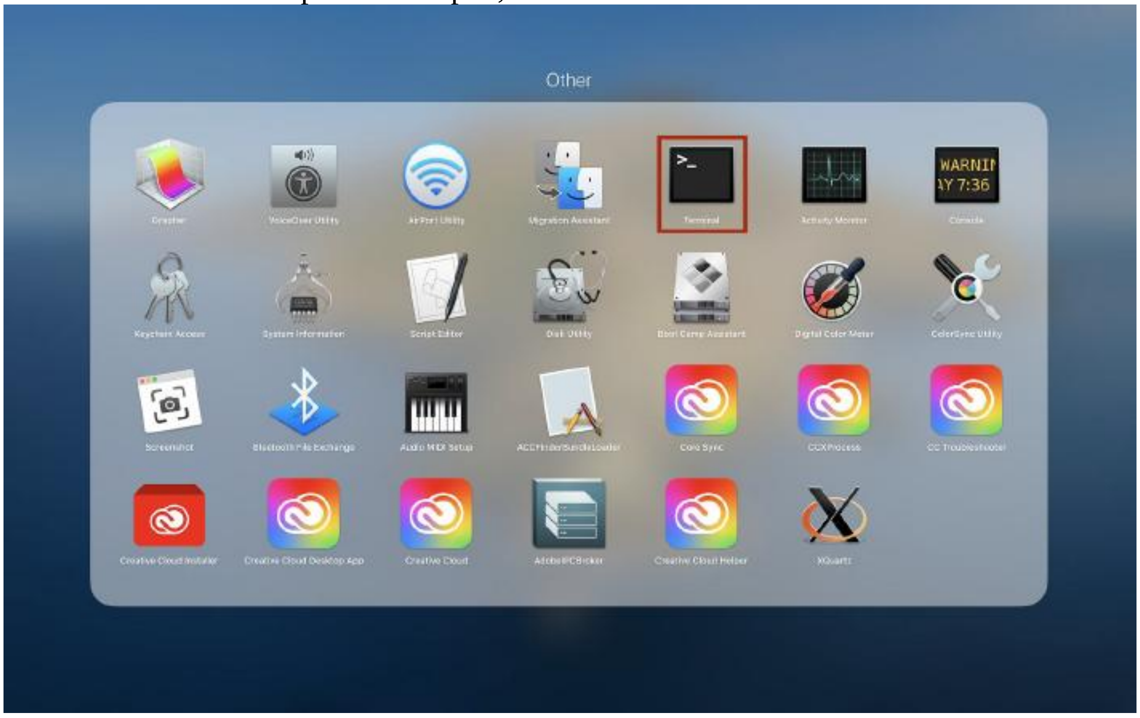
1. Install Anaconda

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.’

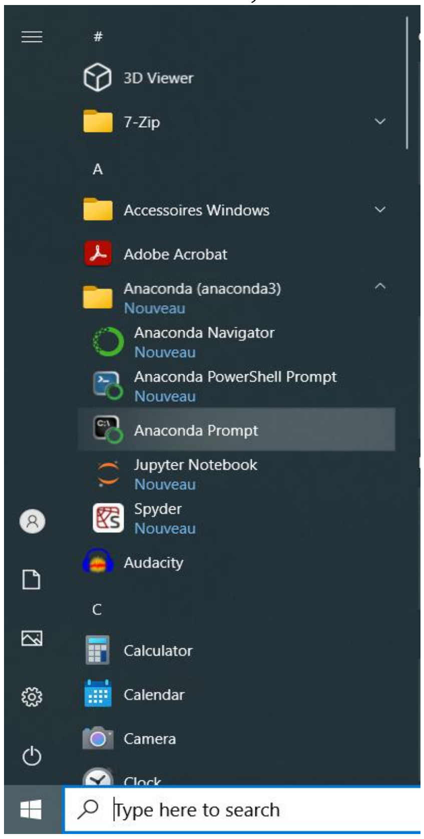
   b. Download an appropriate graphical installer according to the OS you are using (Mac or Windows). If you are a Windows user, you can check whether your machine is 32-bit or 64-bit following this guide: https://support.microsoft.com/en-us/help/15056/windows-32-64-bit-faq
   c. After completing the download,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

2. Using the command-line interface

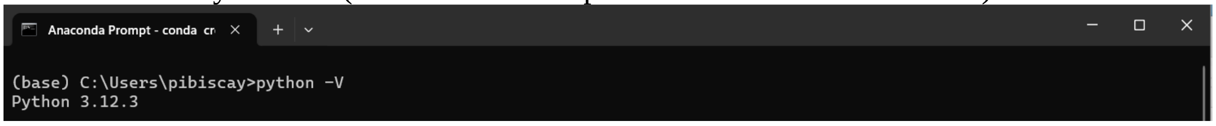
1. We will primarily use a command-line interface for package installation, environment management, and launching python editors.
   a. What is a **command-line interface**? A 'CLI' is a means of interacting with a computer program by inputting lines of text called command-lines. Most computer users 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.
2. Open a command-line interface.
   a. On Mac: Open Launchpad, then click the Terminal icon



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



3. 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.
4. 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 3.11** to ensure all packages we want are installable.
    b. As of December 2025, Python 3.11 has stable, native support for everything library we are working with and is faster and with better error messages than Python 3.9 (which we used in previous versions of the course)



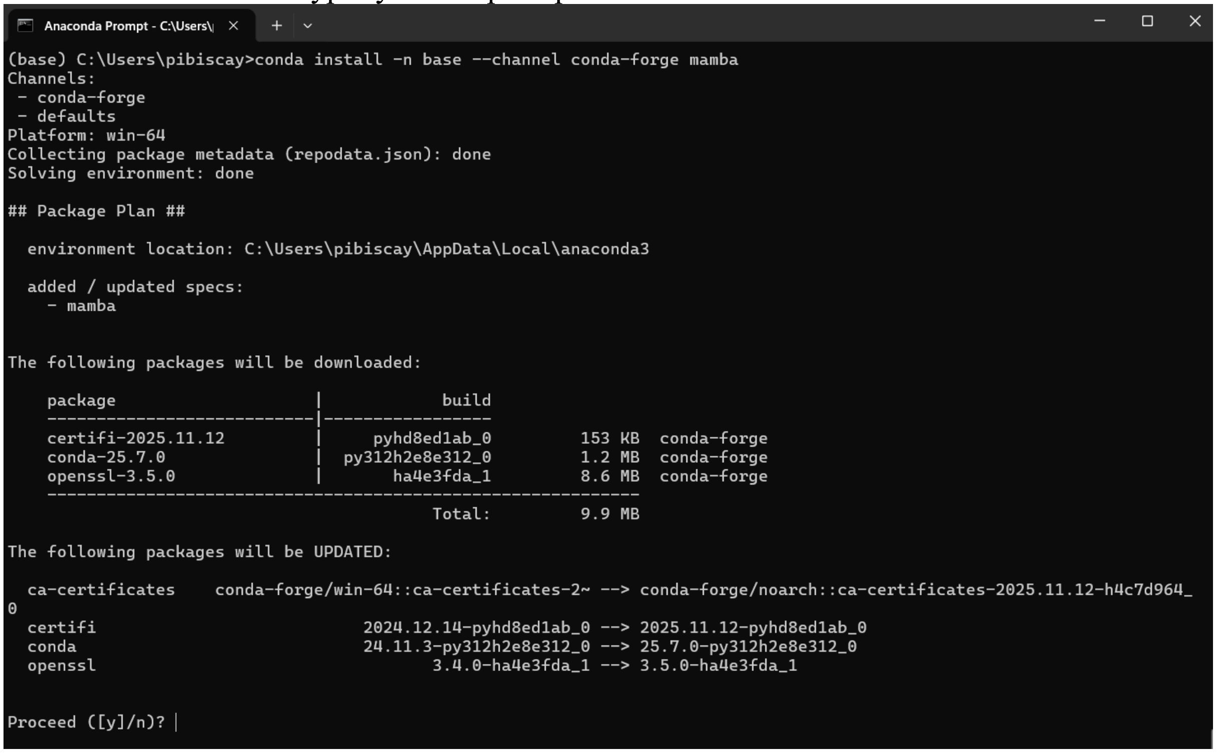
```
(base) C:\Users\pibiscay>python -V
Python 3.12.3
```

3. Create a new environment and install required libraries/packages

Why create a new **environment**?

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. Having a set environment ensures consistency across what everyone is working with.

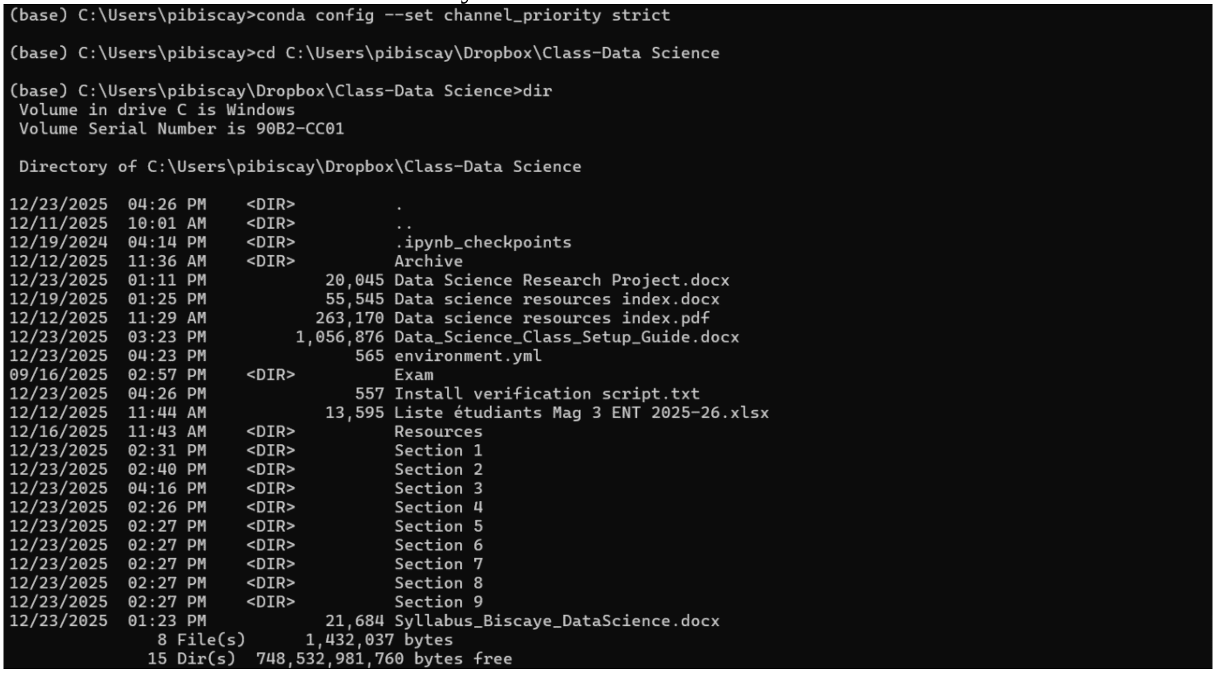
1. Install **mamba** as the package manager.
   a. Mamba is faster and more efficient package manager than Conda. It is fully compatible with Conda packages and environments and it is seamless to switch between the two.
   b. See https://iamdamilare13.medium.com/mamba-vs-conda-know-the-differences-and-similarities-be3ae94d2542
   c. We will type the following command, and then press enter
      i. *conda install -n base --channel conda-forge mamba*
      ii. Type 'y' when prompted and enter



```
(base) C:\Users\pibiscay>conda install -n base --channel conda-forge mamba
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

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

  added / updated specs:
    - mamba


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    certifi-2025.11.12         |     pyhd8ed1ab_0         153 KB  conda-forge
    conda-25.7.0               |  py312h2e8e312_0         1.2 MB  conda-forge
    openssl-3.5.0              |       ha4e3fda_1         8.6 MB  conda-forge
    ------------------------------------------------------------
                                           Total:         9.9 MB

The following packages will be UPDATED:

  ca-certificates    conda-forge/win-64::ca-certificates-2~ --> conda-forge/noarch::ca-certificates-2025.11.12-h4c7d964_
0
  certifi                          2024.12.14-pyhd8ed1ab_0 --> 2025.11.12-pyhd8ed1ab_0
  conda                            24.11.3-py312h2e8e312_0 --> 25.7.0-py312h2e8e312_0
  openssl                                 3.4.0-ha4e3fda_1 --> 3.5.0-ha4e3fda_1


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

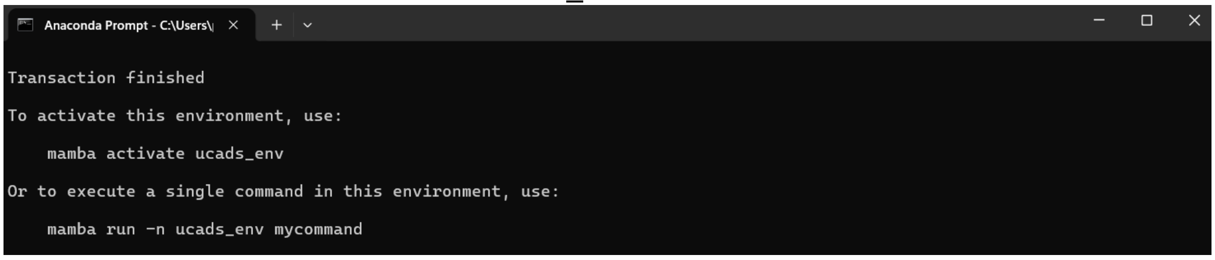
2. Set up an **environment.yml file**
   a. We will be installing a large number of libraries/packages. Doing this one-by-one can create major problems as the installation solver has to check relationships between all installed packages every single time. This is problematic especially for spatial analysis packages which can conflict with other basic packages.
   b. Instead we will install all core libraries using a single command. This allows Mamba to see the "big picture" and find a set of versions across all packages that work together from the start. This also has the advantage of being much faster.
   c. We will use a strict channel strategy, instructing mamba to download version from the **conda-forge channel** rather than the defaults channel (Anaconda). These two channels often fight over dependencies but conda-forge is more up-to-date and robust and is the data science industry standard.
   d. All of the necessary information is contained in the environment.yml file I have provided: environment name (*ucads_env*), channels to use, and dependencies to install.

3. Run the “solve” to **set up the environment**.
    a. First, set strict priority to avoid fighting over dependencies from packages installed from different channels
        i. *conda config --set channel_priority strict*
        ii. This forces the environment to be internally consistent, which means we will all be working from the same libraries.
    b. Then, navigate to where the yml file is stored, for example:
        i. *cd C:\Users\Name\Documents\UCA_DataScience*
        ii. Verify by typing *dir* and pressing enter. You should see environment.yml listed in the files.



```
(base) C:\Users\pibiscay>conda config --set channel_priority strict

(base) C:\Users\pibiscay>cd C:\Users\pibiscay\Dropbox\Class-Data Science

(base) C:\Users\pibiscay\Dropbox\Class-Data Science>dir
 Volume in drive C is Windows
 Volume Serial Number is 90B2-CC01

 Directory of C:\Users\pibiscay\Dropbox\Class-Data Science

12/23/2025  04:26 PM    <DIR>          .
12/11/2025  10:01 AM    <DIR>          ..
12/19/2024  04:14 PM    <DIR>          .ipynb_checkpoints
12/12/2025  11:36 AM    <DIR>          Archive
12/23/2025  01:11 PM            20,045 Data Science Research Project.docx
12/19/2025  01:25 PM            55,545 Data science resources index.docx
12/12/2025  11:29 AM           263,170 Data science resources index.pdf
12/23/2025  03:23 PM         1,056,876 Data_Science_Class_Setup_Guide.docx
12/23/2025  04:23 PM               565 environment.yml
09/16/2025  02:57 PM    <DIR>          Exam
12/23/2025  04:26 PM               557 Install verification script.txt
12/12/2025  11:44 AM            13,595 Liste étudiants Mag 3 ENT 2025-26.xlsx
12/16/2025  11:43 AM    <DIR>          Resources
12/23/2025  02:31 PM    <DIR>          Section 1
12/23/2025  02:40 PM    <DIR>          Section 2
12/23/2025  04:16 PM    <DIR>          Section 3
12/23/2025  02:26 PM    <DIR>          Section 4
12/23/2025  02:27 PM    <DIR>          Section 5
12/23/2025  02:27 PM    <DIR>          Section 6
12/23/2025  02:27 PM    <DIR>          Section 7
12/23/2025  02:27 PM    <DIR>          Section 8
12/23/2025  02:27 PM    <DIR>          Section 9
12/23/2025  01:23 PM            21,684 Syllabus_Biscaye_DataScience.docx
               8 File(s)      1,432,037 bytes
              15 Dir(s)  748,532,981,760 bytes free
```

    c. Finally, create the environment from the yml file
        i. *mamba env create -f environment.yml*
        ii. This will likely take some time. Have this proceed in the background while you work on something else.
        iii. If the installation gets stuck or throws a "Conflict" error, the cleanest fix is to delete the half-finished environment and try again: conda env remove --name ucads_env



```
Transaction finished

To activate this environment, use:

    mamba activate ucads_env

Or to execute a single command in this environment, use:

    mamba run -n ucads_env mycommand
```

    d. As an additional step, register the python kernel to make it visible inside Jupyter Notebooks (which we will discuss below).
        i. *python -m ipykernel install --user --name ucads_env --display-name "Python (UCA DS Base)"*
        ii. This lets you access this environment within Jupyter Notebook without first having to activate the environment and then launch Jupyter from

within the environment. This can be very useful for environment switching for advanced users.

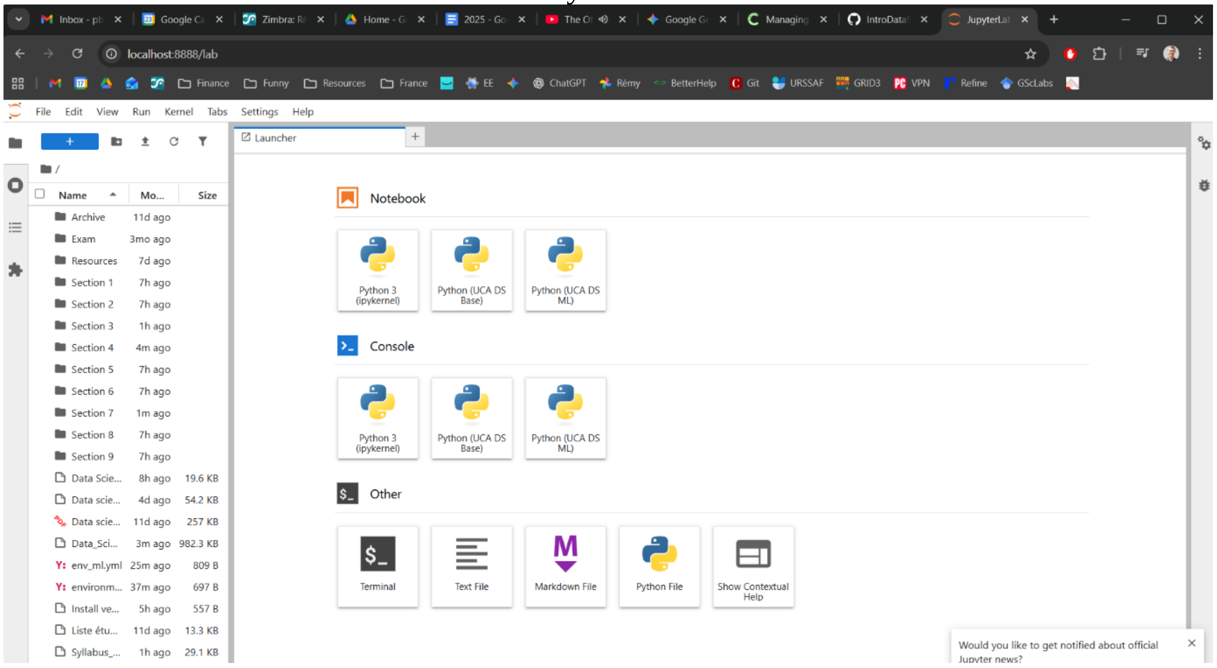
5. Some libraries have different dependencies. This is particularly the case for some of the libraries we will use for machine learning activities. To prevent issues across libraries, we will create a **separate environment for machine learning** libraries.
    a. *mamba env create -f env_ml.yml*
    b. *python -m ipykernel install --user --name ucads_ml --display-name "Python (UCA DS ML)"*
    c. Note that some of these packages are larger than in the basic course environment, and may take longer to install.

4. 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 kernels we created for the environments we set up execute python code. Kernels for many other languages exist.
    e. When you open a Notebook document, you have to choose the associated kernel. 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.

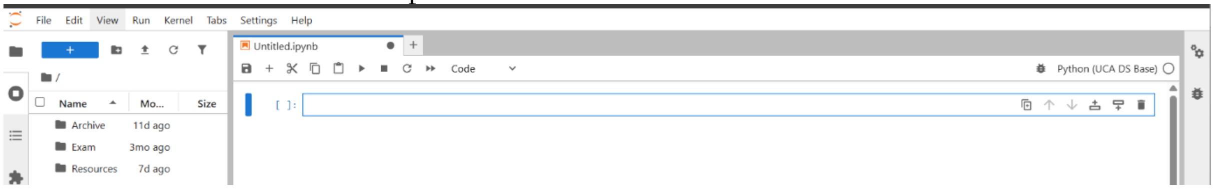
3. How to launch Jupyter Notebook?
    a. From the Anaconda prompt/Terminal
        i. Open the command line interface
        ii. *conda activate ucads_env* (or *ucads_ml* for that part of the course)
        iii. *jupyter lab*
    b. From the Anaconda Navigator
        i. Go to the Environments tab and click on the appropriate environment to load it
        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.
        i. Notice the two new Python kernels we created are there!



Name	Mo...	Size
Archive	11d ago	
Exam	3mo ago	
Resources	7d ago	
Section 1	7h ago	
Section 2	7h ago	
Section 3	1h ago	
Section 4	4m ago	
Section 5	7h ago	
Section 6	7h ago	
Section 7	1m ago	
Section 8	7h ago	
Section 9	7h ago	
Data Scie...	8h ago	19.6 KB
Data scie...	4d ago	54.2 KB
Data scie...	11d ago	257 KB
Data_Sci...	3m ago	982.3 KB
env_ml.yml	25m ago	809 B
environm...	37m ago	697 B
Install ve...	5h ago	557 B
Liste étu...	11d ago	13.3 KB
Syllabus_...	1h ago	29.1 KB

Would you like to get notified about official Jupyter news?

4. Create a New Jupyter Notebook
    a. Navigate to the desired folder
    b. Under Notebook, click on the Python kernsel you want to use. Start with Python (UCA DS Base). This will create a new Jupyter Notebook using the python kernel associated with the targeted environment.
    c. A new tab will open with a blank cell.



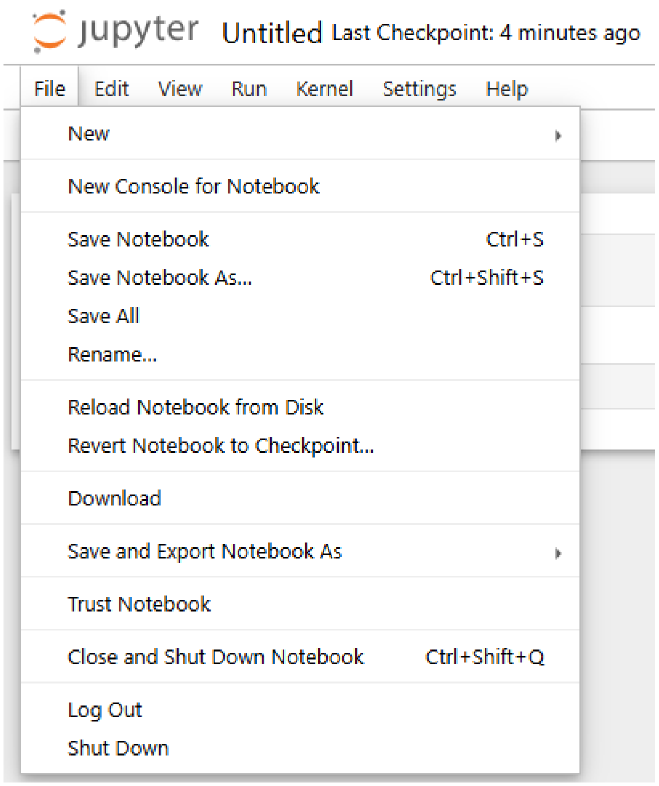
Name	Mo...	Size
Archive	11d ago	
Exam	3mo ago	
Resources	7d ago	

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'

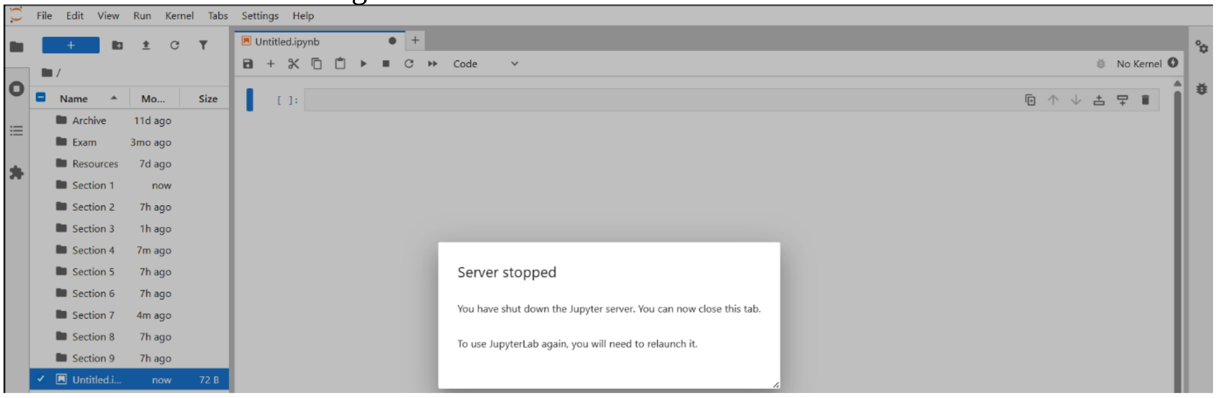
6. Let’s **verify the environment installation**.
    a. Copy the following code into your notebook (in a new cell), and run it.
        i. The * symbol next to the cell means it is working.
        ii. When it finishes, you will see a number appear next to the cell indicating the order in which it has been run.
    b. If you have any errors, try troubleshooting them with help from Google or an LLM. If still stuck, contact me.

```
# Verification Script
import pandas as pd
import geopandas as gpd
import shapely
import statsmodels.api as sm
import matplotlib.pyplot as plt

print(f"Pandas version: {pd.__version__}")
print(f"GeoPandas version: {gpd.__version__}")

# Test a spatial operation (The most likely point of failure)
point = shapely.geometry.Point(0, 0)
buffer = point.buffer(1)
print("Spatial dependencies are working!")

# Test a basic plot
plt.plot([1, 2, 3], [4, 5, 6])
plt.title("Matplotlib Check")
plt.show()
print("Graphics engine is working!")
```

7. **Close out** of Jupyter Lab.
    a. If you wish, give the file a name and save it for future practice.
        i. Note that by default it will be saved as Untitled.ipynb in your working directory.
    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 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 Lab 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 JupyterLab 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