Introduction to Data Analysis with Python

Cecilia Graiff

October 23, 2025

ALMAnaCH, Inria Paris - École d'Affaires Publiques, SciencesPo
cecilia.graiff@sciencespo.fr

Recap of General Concepts

Coding and version control

- You write code **locally** on your computer.
- You use Visual Studio Code, which is **an editor**: think of it like a text editor, an app that you can use to **open, edit, and share your files**
- You use a **virtual environment** to manage packages
- You track your work with **Git**.
- You share and collaborate through **GitHub**.

Coding and version control

- You write code **locally** on your computer.
- You use Visual Studio Code, which is **an editor**: think of it like a text editor, an app that you can use to **open, edit, and share your files**
- You use a **virtual environment** to manage packages
- You track your work with **Git**.
- You share and collaborate through **GitHub**.

To sum up

Think of this as writing a report with your own research setup (Python), saving versions (Git), and uploading to a shared folder (GitHub).

Virtual Environments

What is a Virtual Environment?

- A **virtual environment** is like a private workspace for your project.
- It is a **folder** where you tell Python to download specific packages
- Often, two projects require different packages, or different versions of the same package: hence the **need for a virtual environment**
- It contains its own copy of Python and packages.
- This prevents version conflicts between projects.
- **Reproducibility is fundamental**: to execute your code, people need to **install the same versions of the same pacakages**

What is a Virtual Environment?

- A **virtual environment** is like a private workspace for your project.
- It is a **folder** where you tell Python to download specific packages
- Often, two projects require different packages, or different versions of the same package: hence the **need for a virtual environment**
- It contains its own copy of Python and packages.
- This prevents version conflicts between projects.
- **Reproducibility is fundamental**: to execute your code, people need to **install the same versions of the same pacakages**

Python interpreter

- When you install Python, an **interpreter** is downloaded: this interpreter **understands what you write in Python** and **runs it** (= makes the execution possible)
- Different **operating systems (OS)** call this interpreter differently
- Often, you need to add Python to the **Path variables** if you want to be able to use the standard commands

- Possible commands:
    - `python` usually refers to the latest Python, but if you have both `python3` and `python2`, it can refer to Python 2
    - `python3`: usually safest option
    - `py`: on Windows, Python comes with an interpreter called `py`. Using this commands solves eventual problems with the PATH variable (if you code on a regular basis, I recommend adding Python to PATH, but otherwise, it can be fine)

- Possible commands:
    - `python` usually refers to the latest Python, but if you have both `python3` and `python2`, it can refer to Python 2
    - `python3`: usually safest option
    - `py`: on Windows, Python comes with an interpreter called `py`. Using this commands solves eventual problems with the PATH variable (if you code on a regular basis, I recommend adding Python to PATH, but otherwise, it can be fine)

Please try to understand what your setup is, and what you need to type. You might have to change my code examples, where I usually use `python`.

Git and GitHub Basics

Why Use Git and GitHub?

- **Git** tracks versions of your code (like “Save As” with history).
- **GitHub** hosts those versions online.
- Together, they make your projects **reproducible and shareable**.

Git vs GitHub

- Git is a **tool for version control**
    - Therefore: you need to install Git (did at the beginning of the course)
- GitHub is a **website built around Git**. It is not the only existent one, but the most widely used.
    - Therefore: No need to install anything (although a tool called GitHub Desktop exists: it is a downloadable interface, but not required for this class: due to its limitation, most people use only GitHub).

Basic Git Workflow

1. Clone the repository:

```
git clone https://github.com/username/project.git
```

2. Track and save changes:

```
git add .
git commit -m "Describe your change"
```

3. Upload (push) to GitHub:

```
git push
```

Basic Git Workflow

1. **Clone the repository:**

```
git clone https://github.com/username/project.git
```

2. **Track and save changes:**

```
git add .
git commit -m "Describe your change"
```

3. **Upload (push) to GitHub:**

```
git push
```

Summary: Activate your environment, work locally, then commit and push.

Terminal vs Python commands

- Terminal: you write here what you need to tell **to your computer**
    - Example: git commands, pip install
    - Those are not commands specific to the Python file/Jupyter notebook you are using: When you install a package, you install it **in your virtual environment**, that you can use for any Jupyter notebook you want to create
    - The terminal can be opened with the dedicated app, or in VSC (Click on "Open terminal" in the menu, and it will be displayed in the lowest half of the screen)

Terminal vs Python commands

- Terminal: you write here what you need to tell **to your computer**
    - Example: git commands, pip install
    - Those are not commands specific to the Python file/Jupyter notebook you are using: When you install a package, you install it **in your virtual environment**, that you can use for any Jupyter notebook you want to create
    - The terminal can be opened with the dedicated app, or in VSC (Click on "Open terminal" in the menu, and it will be displayed in the lowest half of the screen)
- Python: Here you write commands specific to your Python file/Jupyter notebook, **with the Python syntax**
    - Example: import packages, functions for data analysis

Terminal vs Python commands

- Terminal: you write here what you need to tell **to your computer**
  - Example: git commands, pip install
  - Those are not commands specific to the Python file/Jupyter notebook you are using: When you install a package, you install it **in your virtual environment**, that you can use for any Jupyter notebook you want to create
  - The terminal can be opened with the dedicated app, or in VSC (Click on "Open terminal" in the menu, and it will be displayed in the lowest half of the screen)
- Python: Here you write commands specific to your Python file/Jupyter notebook, **with the Python syntax**
  - Example: import packages, functions for data analysis

Python is a programming language and as such has a **syntax**, which is different from the one of terminal commands.

Summary

Example Workflow

1. Open terminal and navigate to your project folder.
2. Activate your virtual environment.
3. Run or edit your notebooks/scripts.
4. Save and commit changes with Git.
5. Push to GitHub to back up and share.

Example Session:

```
cd myproject
(or you open the project directly in VSC and use the VSC termina
```

Example Session:

```
cd myproject
(or you open the project directly in VSC and use the VSC termina

source env/bin/activate

(install packages, modify your files, write code, etc)

git add .
git commit -m "Updated analysis"
git push
```

Questions?