Terminal and GitHub Basics, Python Refresher

Programming Psychology Experiments (CORE-1)

Barbu Revencu & Maxime Cauté

Session 1 | 10 September 2025

Who we are

Barbu Revencu

Cognitive scientist (previously linguistics)

Postdoctoral fellow at NeuroSpin (Université Paris-Saclay) & Collége de France

Research interests: Symbols, language of thought, cognitive development

Maxime Cauté

Cognitive scientist (previously mathematics and computer science)

PhD student at NeuroSpin (Université Paris-Saclay)—almost done

Research interests: Mathematical cognition, cognitive development

Reminder about CORE-1

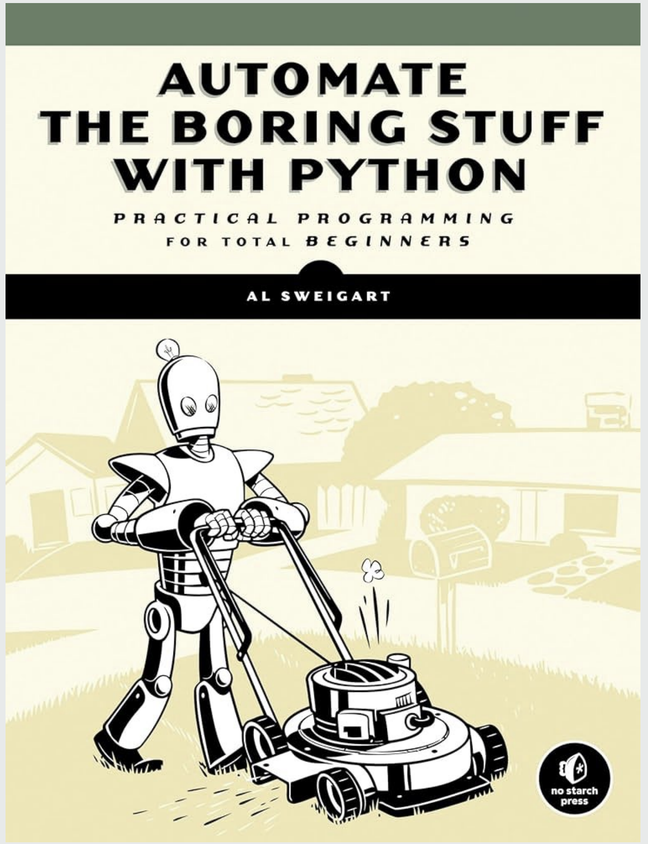
CORE-1 is a **mandatory course** toward the completion of the program, but you can also take it next semester (w/ Sylvain Charron)

Those who scored **≤ 12 points** on Friday's quiz must take PROG-1 first, before enrolling in CORE-1 next semester

Those whose scores were **between 13 and 17 points** are encouraged to take PROG-1 now and CORE-1 in the next semester

Those who **plan to take PROG-2** next semester must take CORE-1 now

Recommended reading



https://automatetheboringstuff.com/

The plan for today

1. Gentle introduction to **Terminal/Git Bash** commands
2. Finish setting up the connection between your computer and **GitHub**
3. **Python refresher**: Loops, lists, dictionaries, functions

Terminal & Git Bash

Terminal basics

The **Terminal** (on Mac and Linux) and **Git Bash** (on Windows) are applications that allow you to interact with your operating system by **typing commands** instead of **visualizing folders + clicking**

In this class, you will need the Terminal to:

- Run python scripts

```
Barbu@Mac % python script.py
```

- Link your computer to GitHub

```
Barbu@Mac % git add .
Barbu@Mac % git commit -m "Changed X"
Barbu@Mac % git push origin
```

Terminal basics: `pwd`

The Terminal always opens in **a working directory** (by default, the user directory), just like your GUI File Explorer

To find out which directory you are currently in, type in `pwd` (**p**rint **w**orking **d**irectory)

```
Barbu@Mac % pwd
/Users/Barbu/
```

Terminal basics: `ls`

You can **check the contents** of the current working directory by typing in `ls` (**list** directory contents)

```
Barbu@Mac % ls
Applications     Desktop     Music     Calibre Library
Documents        Library     Pictures  Downloads
Movies           Public      PPE
```

Confirm this by opening the home folder in your Finder/File Explorer (**MAC**: Finder > Go > Home | **WINDOWS**: C:/Users/*your-username*)

Terminal basics: `cd`

If you have a ‘PPE’ folder in your home directory, type in:

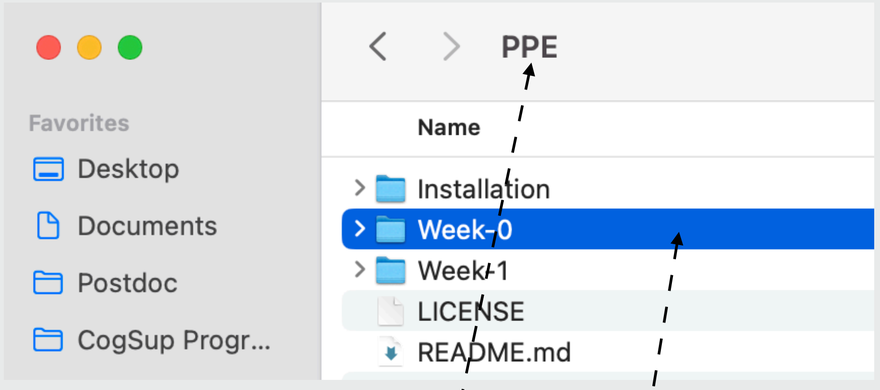
```
Barbu@Mac % cd PPE
Barbu@Mac % git pull
```

Else, type in:

```
Barbu@Mac % git clone https://github.com/barburevencu/PPE
Barbu@Mac % cd PPE
```

Terminal basics: `cd`

The GUI way (Finder/File Explorer)



The Terminal way (`cd` = **c**hange **d**irectory):

```
Barbu@Mac PPE % cd Week-0
```

Terminal basics: `cd`, `ls`, `pwd`

Use `ls` (**l**ist directory contents) and `pwd` (**p**rint **w**orking **d**irectory) in the new working directory

```
Barbu@Mac PPE % ls
Session 0. Presentation.pdf
Barbu@Mac PPE % pwd
/Users/Barbu/PPE/Week-0
```

Terminal basics: `cd ..`

Going back to the parent directory of the current working directory

```
Barbu@Mac Week-0 % cd .. # /PPE/Week-0
Barbu@Mac PPE % cd Installation
Barbu@Mac Installation % cd ../Week-0 # ≡ cd .. ↵ cd Week-0
Barbu@Mac Week-0 % cd ..
Barbu@Mac PPE %
```

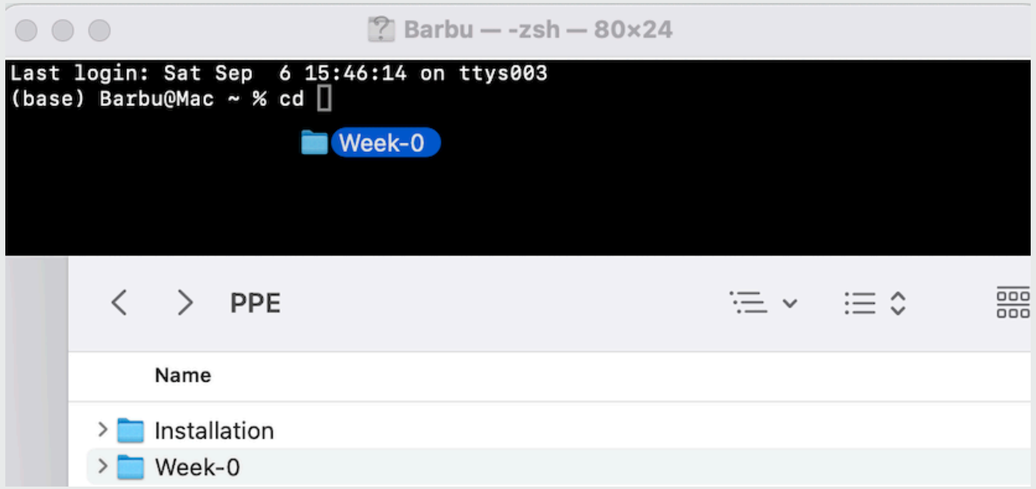
Terminal basics: The current directory

Every file-related command takes into account the current directory, because the paths you specify are relative to it

```
Barbu@Mac PPE % cd Installation # ≡ cd /Users/…/PPE/Installation
Barbu@Mac Installation % ls
installation-guide.md     installation-test.md      Test
Barbu@Mac Installation % cd # Takes you back to the home folder
Barbu@Mac % cd Installation
cd: no such file or directory
```

Terminal basics: `cd` + drag & drop

Instead of typing the entire path, you can write `cd` followed by `space`, then drag and drop the folder you want to set as your current working directory from your file explorer app



Terminal basics: ↑ ↓

Instead of retyping recent commands, you can use UP and DOWN arrow keys to toggle through the history of your commands

```
Barbu@Mac % cd /Users/Barbu/PPE/Week-0
Barbu@Mac Week-0 % cd
Barbu@Mac % cd /Users/Barbu/PPE/Week-0
# Instead of typing the whole path again, press ↑ twice to fill in
the whole command, then press RETURN to execute it
```

GitHub Basics

Setting up the course folder

First, remove the PPE folder in your home directory (GUI/Terminal)

```
Barbu@Mac % cd # Go back to your home folder
Barbu@Mac % rm -r PPE # Remove the PPE folder and all subfolders
```

You will be asked for confirmation (possibly multiple times)—press `y` to confirm (or type `rm -rf PPE` to skip confirmation prompts)

Setting up the course folder

Depending on your organization preferences, create a folder dedicated to this class (e.g., …/`Documents/CogSup/Programming`)

```
Barbu@Mac % cd /Users/Barbu/Documents/
Barbu@Mac Documents % mkdir CogSup # Creates CogSup folder
Barbu@Mac Documents % cd CogSup # Go to CogSup folder
Barbu@Mac Cogsup % mkdir Programming # Create Programming folder
Barbu@Mac Cogup % cd Programming # Go to Programming subfolder
```

Setting up the course folder

The **Programming** folder will contain two further subfolders

The first subfolder will correspond to *our* GitHub repository, which will contain the course materials (slides, exercises, etc.)

```
Barbu@Mac Programming % git clone https://github.com/barburevencu/PPE
```

Typing in `ls` should now print the **PPE** folder

Recommended: Rename the **PPE** folder to ‘Materials’ to identify it faster

```
Barbu@Mac Programming % mv PPE Materials
```

Setting up the course folder

The second subfolder will correspond to *your* GitHub repository, which you will use to upload your solutions to the exercises

```
Barbu@Mac Programming % git clone https://github.com/YOUR-GITHUB-
USERNAME/cogsup-prog Assignments
# Change cogsup-prog if your repo has another name
# ≡ % git clone https://github.com/YOUR-GITHUB-USERNAME/cogsup-prog ↵
    % mv cogsup-prog Assignments
```

Use `ls` to confirm that **Assignments** is part of the Programming folder

GitHub basics: `pull`

Let's check that everything works as it should

To update the **PPE** (**Materials**) folder to the most recent version, go to the folder and *pull* our PPE repository

```
Barbu@Mac Programming % cd Materials
Barbu@Mac Materials % git pull
```

In plain language: Check if there are any new changes on GitHub that I don't have yet, download them, and update my local copy to include them

GitHub basics: `push`

To update your GitHub repository with the folders and files in your local folder, you need to *push* your local directory to GitHub

First, let's create a new folder called **Week-1**

```
Barbu@Mac Materials % cd ../Assignments
Barbu@Mac Assignments % mkdir Week-1
Barbu@Mac Assignments % ls
Week-0    Week-1
```

GitHub basics: `push`

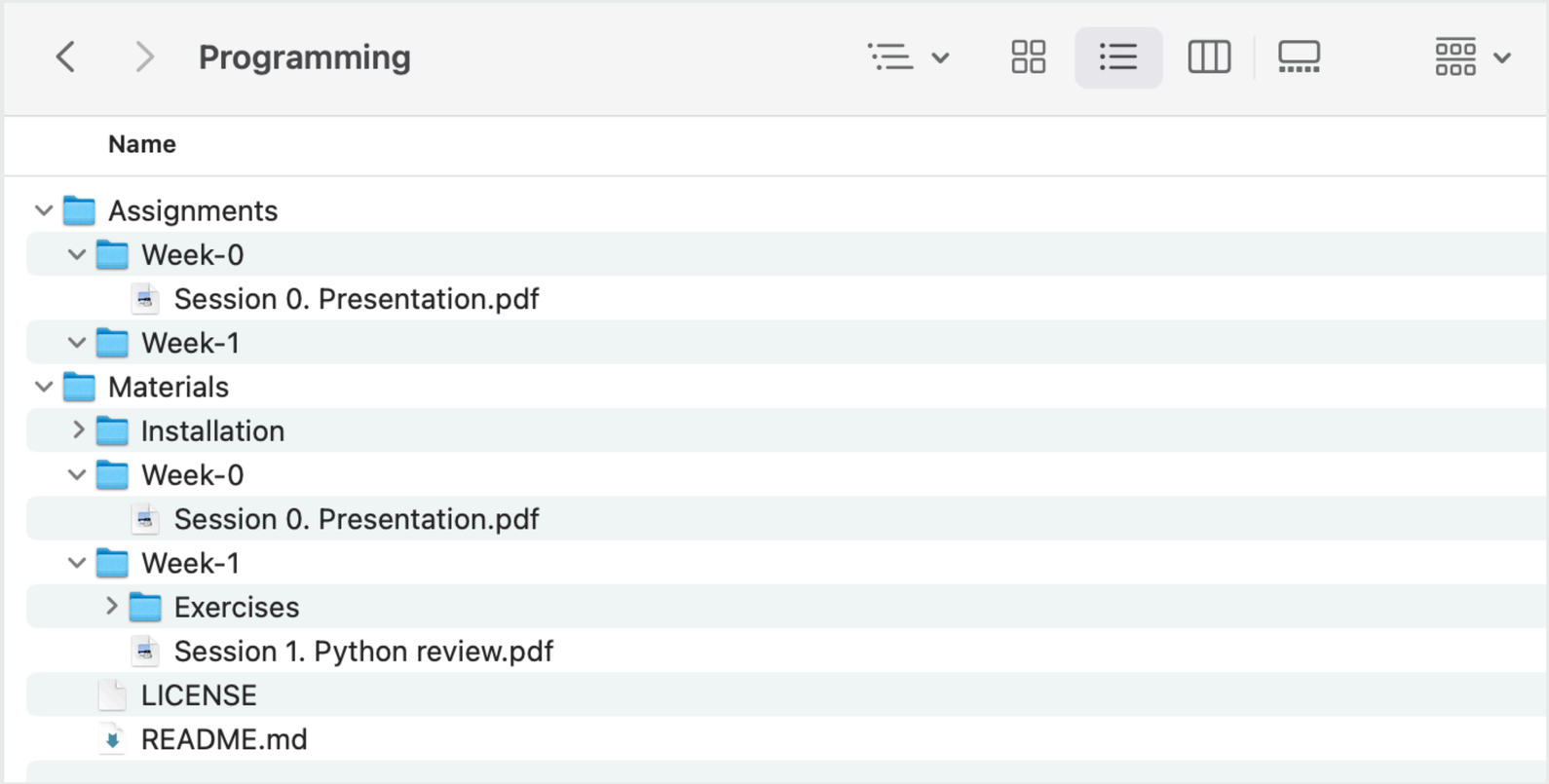
Now let's upload it to GitHub

```
Barbu@Mac Assignments % git add .
Barbu@Mac Assignments % git commit -m "Created Week-1 folder"
Barbu@Mac Assignments % git push origin
```

In plain language:

1. Take a snapshot of all the files I've changed
2. Save changes under a summery label to keep track of what changed
3. Upload changes so they're stored online

How your folders should look like



Let us know if this hasn’t worked for you.

Python Refresher

Preparing the Assignment folder

Let’s move all the materials for Week 1 to your repository

```
Barbu@Mac Assignments % cd Week-1
Barbu@Mac Week-1 % cp -R ../../Materials/Week-1/* .
```

In plain language:

1. `cp`: **copy**
2. **`-R`: r**ecursively (include subfolders)
3. `../../Materials/Week-1/`*****: Everything in this folder
4. `.`: To the current working directory

The first in-class exercise

Type `ls` to confirm that file copying took place

```
Barbu@Mac Week-1 % ls
Exercises    Session 1. Python review.pdf
```

Go to **Exercises** and run `Exercise-1.1.py`

```
Barbu@Mac Week-1 % cd Exercises
Barbu@Mac Week-1 % python Exercise-1.1.py # Or python3
```

In plain language: Open the file called `Exercise-1.1.py` and interpret its contents using `python`

Solve Exercise 1.1

`Push` Exercise 1.1 to GitHub

Go back to the parent **Assignment** folder (which is linked to *your* GitHub)

```
Barbu@Mac Exercises % cd ../..
Barbu@Mac Assignment % git add .
Barbu@Mac Assignment % git commit -m "Added answers to Exercise 1.1"
Barbu@Mac Assignment % git push origin
```

Open your GitHub repository in a browser and check if the changes are reflected there as well. If yes, you're good to go; if not, let us know!

Exercises 1.2–1.5

Continue Exercise 1 by running `python Exercise 1.x.py` in the Terminal for `x` in `[2, 3, 4, 5]`

When done, `push` the **Assignments** folder to GitHub

Exercise 2 and beyond

In your file explorer, go to the **Assignments/Week-1/Exercises** folder, then open **Exercise-2.py** in Visual Studio Code (VS Code)

For each sub-exercise (2.1–2.8), you will have to write some code using **loops**. When done, push your work to GitHub

For questions, get our attention and we'll come to help you

Repeat for Exercises 3–7

Your Feedback

Difficulty of in-class assignments

Fill in the form at https://forms.gle/TPDjfrC3Ejww1q26A