Introduction to the course

Monday, September 16

(https://colab.research.google.com/github/arnauddyevre/Python-for-Social-Scientists/blob/master/intro_to_programming/introduction/introduction.ipynb)

Contents

1. Logistics

1.1. Organisation of the course

Each class will be divided into two parts: a lecture, and a computer exercise. Each part will be approximately 80 minutes long and there will be a 20 minute break in-between. Attending the second part of the course is not compulsory, but we encourage you to do so as the exercises will be directly relevant to research-related applications. Moreover, the class teacher will be able to provide direct feedback.

1.2. Computer exercises

The exercises require that you use a laptop. You can bring your own PC or Mac. Linux machines can also be used, but they will sometimes require different softwares, and the teacher will not always be able to assist.

The PhD Academy has 22 MacBooks that can be borrowed for the duration of the class. They will need to be borrowed at the beginning of each day, and be returned at the end. Importantly, they cannot leave the PhD training room. The Anaconda distribution is already installed on the PhD Academy laptops, and it contains the main piece of software we will be using to write Python code: Jupyter. The laptops are also equipped with GitHub desktop. So you will not need to install anything else for the whole course if you use these machines.

Be careful, the PhD Academy laptops get their memory wiped when you put them back in their lockers, so you if you want to save your work somewhere, you will need to save it onto a personal drive or SD memory card. You can also save it directly to a dedicated GitHub repository; this would actually be very helpful to get familiar with version control.

1.3. Class material

All Notebooks, documents, datasets and the syllabus are available on the course GitHub repository (https://github.com/arnauddyevre/Python-for-Social-Scientists).

2. Technical details

During the course, we will use the following applications and services:

- **Jupyter Notebooks**, accessed through the Anaconda distribution
- **GitHub Desktop**
- **Amazon Web Services**, accessed directly through your web browser
- **Google Colab**, used via your browser as well

2.1. Jupyter Notebooks

Jupyter Notebooks are interactive Python scripts that can contain code and text. They support $\LaTeX$ and Markdown. As such, they are great ressources to learn Python.

This document is a Jupyter Notebook: it supports equations

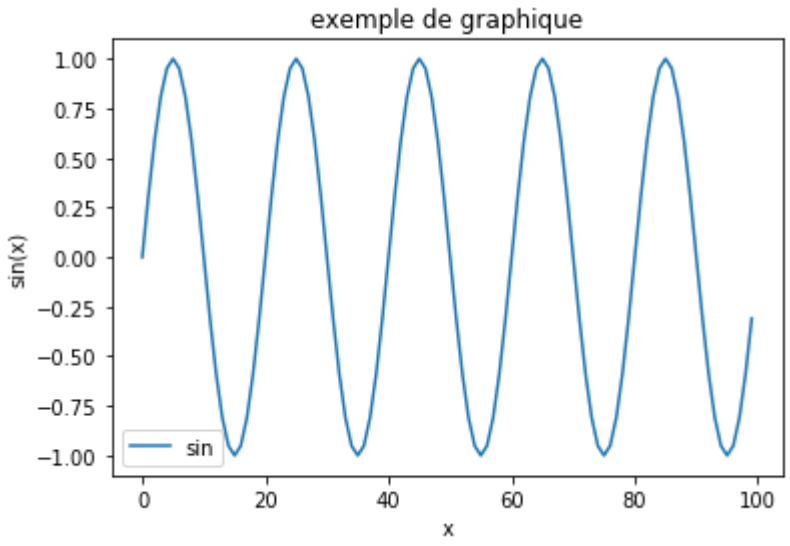
$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k$$

and it can run bits of code such as the snippet below, which simply outputs the graph of the sine function. After having selected the cell with Python code, you can run the code by pressing `Shift + Enter` in Windows and `Cmd + Enter` in Mac.

In [1]:

```
import numpy as np #package for scientific computing
import matplotlib.pyplot as plt #package for graphs
import math #package for mathematical functions
%matplotlib inline

#exemple of graph
x=range(100)
y=[math.sin(xi*math.pi/10) for xi in x]
plt.plot(x,y)
plt.title('exemple de graphique')
plt.legend(['sin'])
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.draw()
plt.show()
```

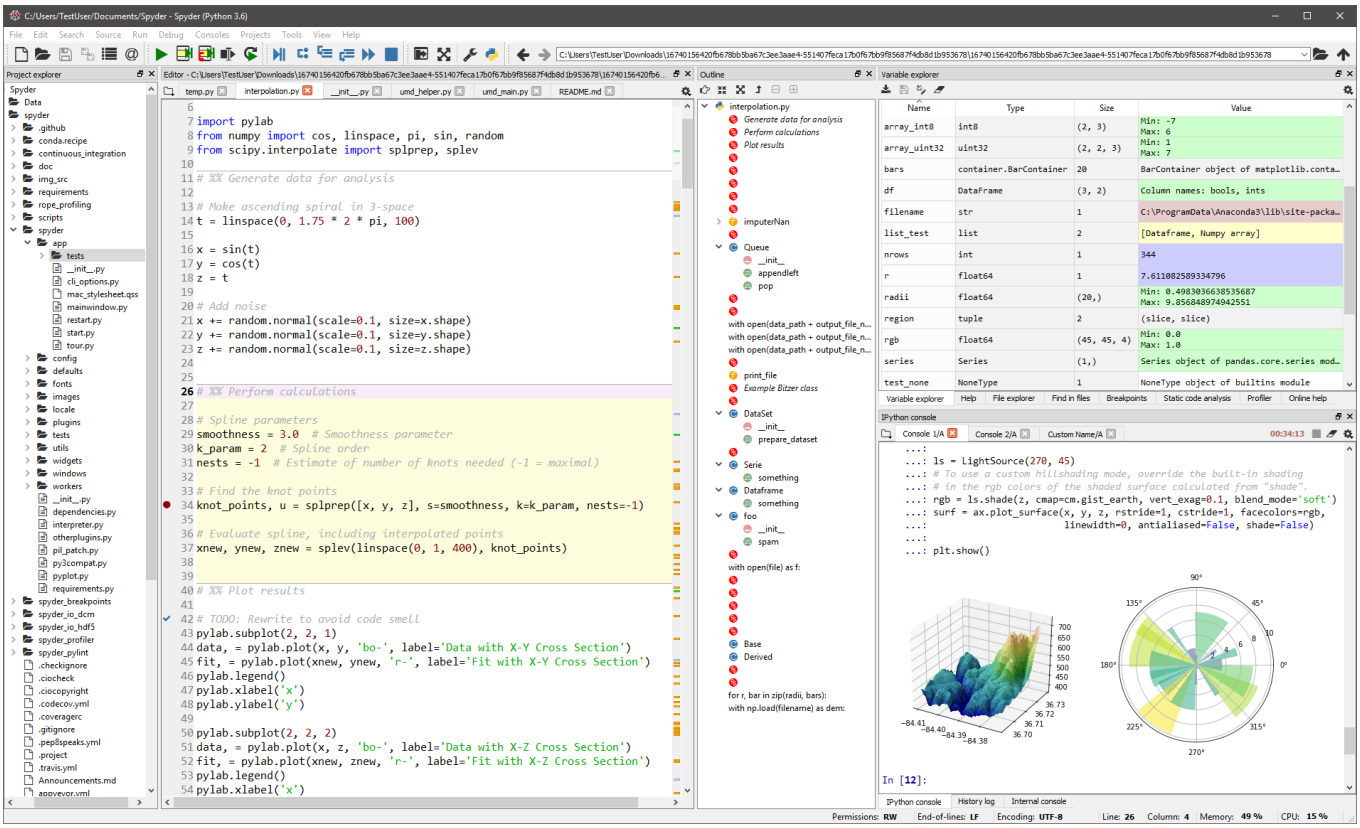


The numbers between blue brackets [ ] allow you to keep track of the order in which you have run the commands. This is helpful when you run many cells, and loose track of variables you have defined earlier. If you are really lost, it is often best to run the whole thing from the start. To do so, selct **Kernel** in the ribbon at the top, and choose **Restart**. All numbers in brackets and all outputs will clear. A **Restart** erases all the objects you have created.

Modifying a Notebook. You can modify a Notebook yourself. To add a code or a Markdown cell, click the **+** button: this adds a cell below the one you have selected. To delete a cell, simply select one and click on the scissors in the ribbon. You the need to choose whether you are typing code (`Code` in the dropdown menu) or text (`Markdown`). You can double click on a text cell to see how the Markdown formatting work. Simply run the Markdown cell after that, just like a code cell (`Shift` or `Cmd + Enter`).

We will create a Jupyter Notebook and get familiar with version control in GitHub as a class exercise the first day.

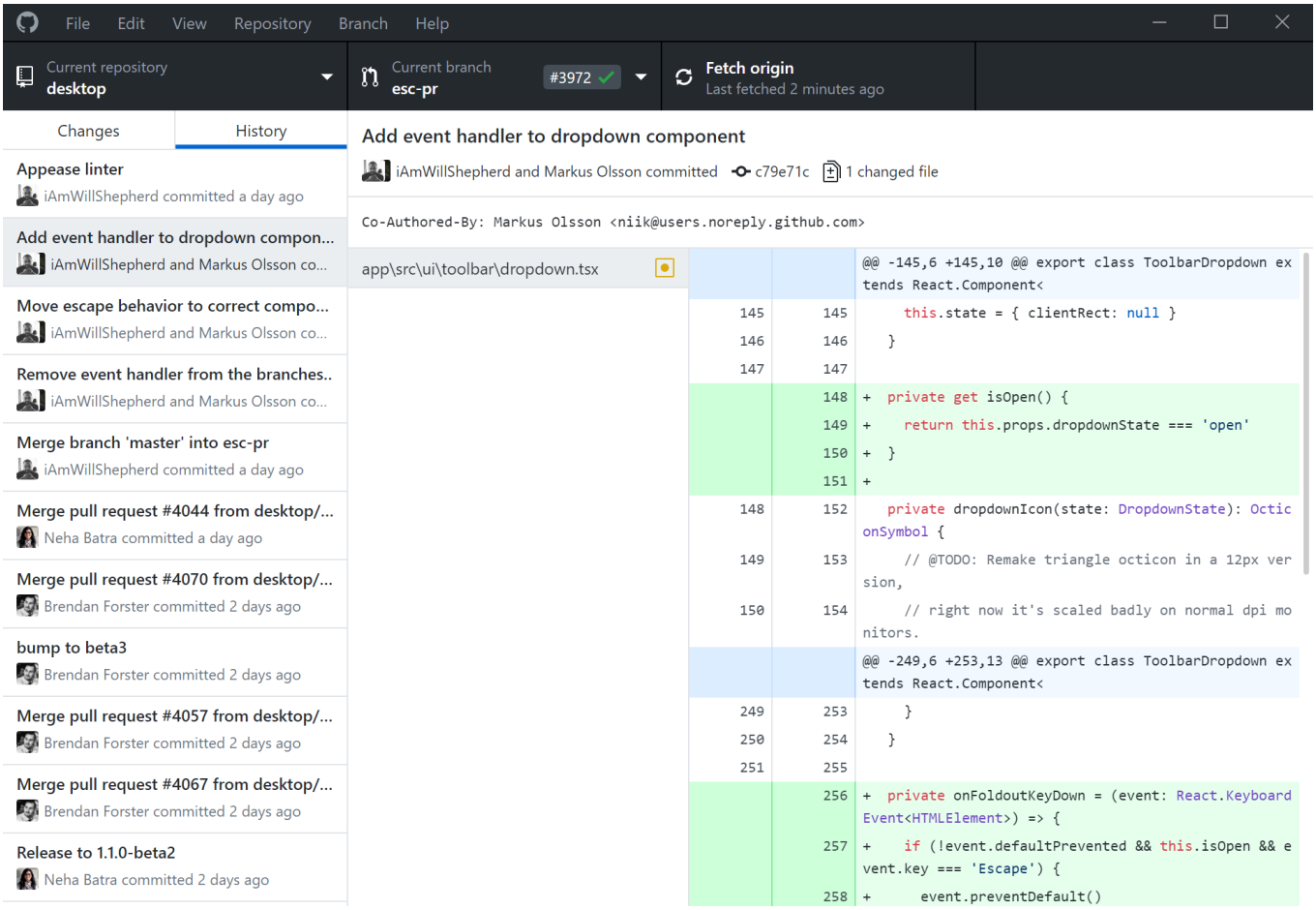
It is worth mentioning **Spyder** at this stage. Spyder is a Stata-like environment for data analysis in Python. While we will not use it during the class for empirical exercises, it is a useful resource to be aware of. It also comes with Anaconda. See a screenshot of Spyder's interface below.



Graph from [spyder-ide.org (https://docs.spyder-ide.org)](https://docs.spyder-ide.org)

2.2. GitHub Desktop

This is the prime interface for non-command line version control. The whole first exercise in Version Control is dedicated to it. See below a snapshot of its interface.



Graph from [desktop.github.com (https://desktop.github.com)](https://desktop.github.com)

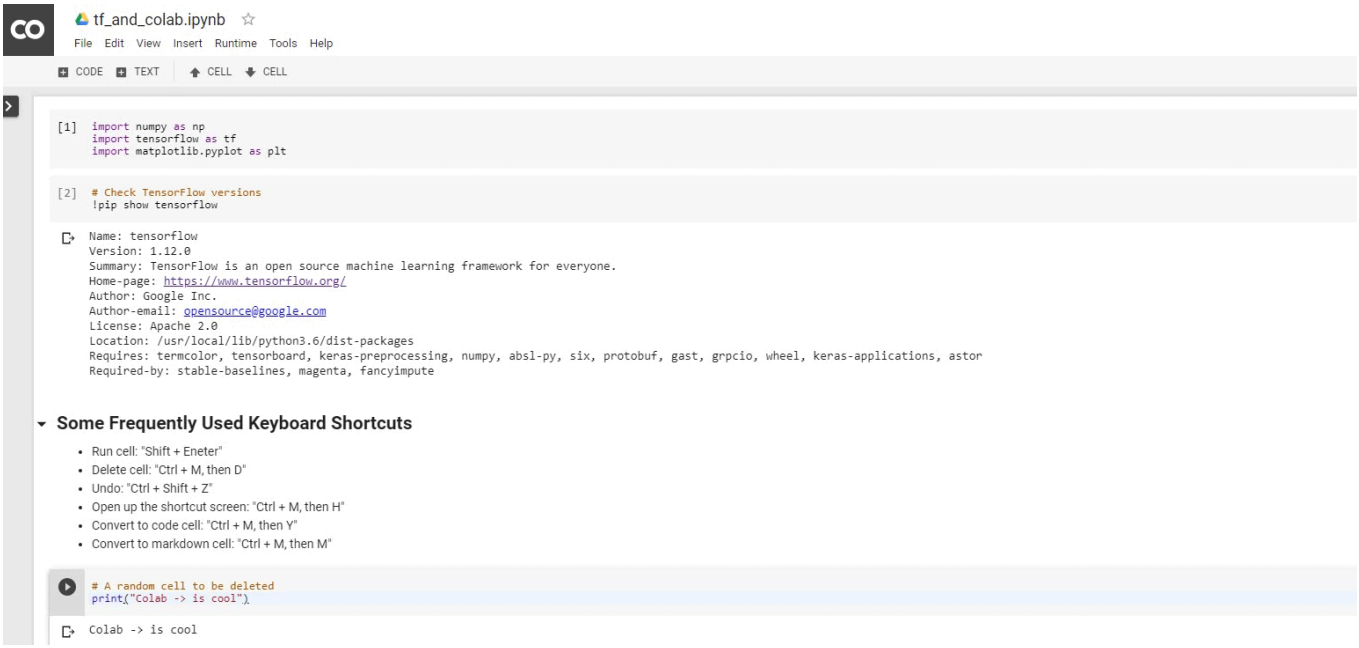
2.3. Amazon Web Service

Amazon Web Service is Amazon's Cloud computing platform. It allows us to execute computationally intensive and memory-demanding tasks online. Its versatility and generous computing resources make it a tool of choice for big data projects. We will us it during the lecture on Cloud Computing.

Notable alternatives to AWS are **Google Cloud Platform** and **Microsoft Azure**.

2.4. Google Colab

All Notebooks are available online through Google's own version of the Jupyter Notebook: **[Google Colab (https://colab.research.google.com/notebooks/welcome.ipynb)](https://colab.research.google.com/notebooks/welcome.ipynb)**. The great thing with Colab is that it executes the code online, so there is no need to install any software on our local machines. The computing power offered by Google Colab is also well beyond what we will need for this course. It can be interesting to start data projects directly in Colab.



GIF from [towarddatascience.com (https://towardsdatascience.com/getting-started-with-tensorflow-in-google-colaboratory-9a97458e1014)](https://towardsdatascience.com/getting-started-with-tensorflow-in-google-colaboratory-9a97458e1014)

3. Outline of the course

The course consists of three parts:

- Introduction to coding in Python, version control, and cloud computing (3 days)
  - (1) Version control
  - (2) Cloud computing
  - (3) Introduction to Python
- Data analysis for the social sciences (4 days)
  - (4) Data handling
  - (5) Linear models
  - (6) MLE and discrete choice models
  - (7) Time series
- Introduction to machine learning (2 days)
  - (8) Machine learning 1
  - (9) Machine learning 2

4. Recommended readings

Best practices

- Gentzkow, M. and J. M. Shapiro (2014). “Code and data for the social sciences: A practitioners guide”. *University of Chicago mimeo* link (https://web.stanford.edu/~gentzkow/research/CodeAndData.pdf)

Version control

- GitHub Guides (2017). “Understanding the GitHub flow”, accessed August 5, 2019 link (https://guides.github.com/introduction/flow/)

Data analysis in Python

- Sargent, T. J. and J. Stachurski (2019). “Lectures in quantitative economics with Python”. *mimeo* link (https://lectures.quantecon.org/_downloads/pdf/py/Quantitative%20Economics%20with%20Python.pdf)
- Bell, A. (2016). “Python for Economists”. *mimeo* link (https://scholar.harvard.edu/files/ambell/files/python_for_economists.pdf)

Introduction to machine learning

- Athey, S., and Imbens, G. W. (2019). “Machine learning methods that economists should know about” *Annual Review of Economics*, 11. link (https://www.annualreviews.org/doi/abs/10.1146/annurev-economics-080217-053433)
- Mullainathan, S., and Spiess, J. (2017). “Machine learning: an applied econometric approach” *Journal of Economic Perspectives*, 31(2), 87-106. link (https://www.aeaweb.org/articles?id=10.1257/jep.31.2.87)