

# Introduction to the course

Monday, September 16



([https://colab.research.google.com/github/arnaudyevre/Python-for-Social-Scientists/blob/master/intro\\_to\\_programming/introduction/introduction.ipynb](https://colab.research.google.com/github/arnaudyevre/Python-for-Social-Scientists/blob/master/intro_to_programming/introduction/introduction.ipynb))

## Contents

- [1. Logistics](#)
- [2. Technical details](#)
- [3. Outline of the course](#)
- [4. Recommended readings](#)

## 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/arnaudyevre/Python-for-Social-Scientists). (<https://github.com/arnaudyevre/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  $LAT_{EX}$  and Markdown. As such, they are great resources to learn Python.

This document is a Jupyter Notebook: it supports equations

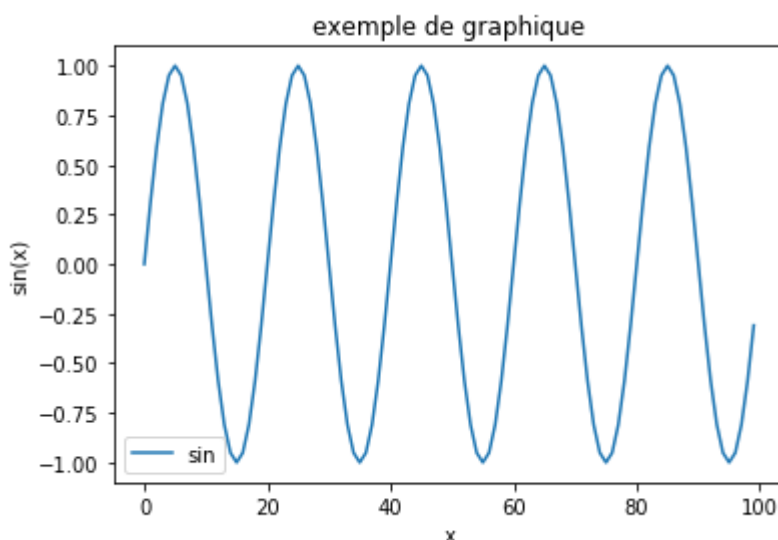
$$e^x = \sum_{k=0}^{\infty} \frac{1}{k!} x^k$$

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, select **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.

The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script with the following code:

```

6
7 import pylab
8 from numpy import cos, linspace, pi, sin, random
9 from scipy.interpolate import splprep, splev
10
11 # Generate data for analysis
12
13 # Make ascending spiral in 3-space
14 t = linspace(0, 1.75 * pi, 100)
15
16 x = sin(t)
17 y = cos(t)
18 z = t
19
20 # Add noise
21 x += random.normal(scale=0.1, size=x.shape)
22 y += random.normal(scale=0.1, size=y.shape)
23 z += random.normal(scale=0.1, size=z.shape)
24
25
26 # Perform calculations
27
28 # Spline parameters
29 smoothness = 3.0 # Smoothness parameter
30 k_param = 2 # Spline order
31 nests = -1 # Estimate of number of knots needed (-1 = maximal)
32
33 # Find the knot points
34 knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)
35
36 # Evaluate spline, including interpolated points
37 xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)
38
39
40 # Plot results
41
42 # TODO: Rewrite to avoid code smell
43 pylab.subplot(2, 2, 1)
44 data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
45 fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
46 pylab.legend()
47 pylab.xlabel('x')
48 pylab.ylabel('y')
49
50 pylab.subplot(2, 2, 2)
51 data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
52 fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
53 pylab.legend()
54 pylab.xlabel('x')

```

The left sidebar shows the Project Explorer with a file tree. The right sidebar shows the Variable Explorer with a table of variables and their types. The bottom panel shows the Python console with the execution of the script, including a 3D surface plot and a 2D polar plot.

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.

Current repository: desktop

Current branch: esc-pr #3972

Fetch origin: Last fetched 2 minutes ago

Changes | History

**Add event handler to dropdown component**

iAmWillShepherd and Markus Olsson committed • c79e71c 1 changed file

Co-Authored-By: Markus Olsson <niik@users.noreply.github.com>

app\src\ui\toolbar\dropdown.tsx

```

@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
    this.state = { clientRect: null }
  }
  145
  146
  147
  148 + private get isOpen() {
  149 +   return this.props.dropdownState === 'open'
  150 + }
  151 +
  148
  152 private dropdownIcon(state: DropdownState): OcticonSymbol {
  153   // @TODO: Remake triangle octicon in a 12px version,
  154   // right now it's scaled badly on normal dpi monitors.
  149
  153
  150
  154
  249
  253 }
  250
  254 }
  251
  255
  256 + private onFoldoutKeyDown = (event: React.KeyboardEvent<HTMLElement>) => {
  257 +   if (!event.defaultPrevented && this.isOpen && event.key === 'Escape') {
  258 +     event.preventDefault()
  
```

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 use 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>). 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.

The screenshot shows a Jupyter Notebook titled 'tf\_and\_colab.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for Code, Text, and Cell. The notebook contains two code cells. The first cell imports numpy, tensorflow, and matplotlib. The second cell checks the TensorFlow version using '!pip show tensorflow'. Below the code cells, there is a section titled 'Some Frequently Used Keyboard Shortcuts' with a list of shortcuts. At the bottom, there is a comment about deleting a cell and a print statement.

```
[1] import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

[2] # Check TensorFlow versions
!pip show tensorflow

Name: tensorflow
Version: 1.12.0
Summary: TensorFlow is an open source machine learning framework for everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: opensource@google.com
License: Apache 2.0
Location: /usr/local/lib/python3.6/dist-packages
Requires: termcolor, tensorboard, keras-preprocessing, numpy, absl-py, six, protobuf, gast, grpcio, wheel, keras-applications, astor
Required-by: stable-baselines, magenta, fancyimpute
```

**Some Frequently Used Keyboard Shortcuts**

- Run cell: "Shift + Enter"
- Delete cell: "Ctrl + M, then D"
- Undo: "Ctrl + Shift + Z"
- Open up the shortcut screen: "Ctrl + M, then H"
- Convert to code cell: "Ctrl + M, then Y"
- Convert to markdown cell: "Ctrl + M, then M"

```
# A random cell to be deleted
print("Colab -> is cool")

Colab -> is cool
```

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) (<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/) (<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) ([https://lectures.quantecon.org/\\_downloads/pdf/py/Quantitative%20Economics%20with%20Python.pdf](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) ([https://scholar.harvard.edu/files/ambell/files/python\\_for\\_economists.pdf](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) (<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) (<https://www.aeaweb.org/articles?id=10.1257/jep.31.2.87>)