LEGAL DATA ANALYSIS

COURSE OVERVIEW

This course will introduce students to basic and common data analysis methods and tools, as applied in particular to legal data and legal knowledge.

This is a course for beginners: no pre-requisite is required, as we will start from the ground up. Students will only need a computer and an internet connection. Students who already know how to code in Python need not apply.

The course is aimed at all students interested in data analytics, since the skills learned on the basis of legal datasets are transferable to *any* kind of data. The lessons will elaborate on why mastering these skills are critical for their future career in the digital economy.

The course should be of particular interest for students who want to work in tech start-ups (especially legal techs), law firms, academia, or in devising public policy. Students will also learn how data can be used in a variety of business applications.

At the end of the course, students should be able to:

1. Identify and constitute a dataset;

2. Develop measures, metrics, and categories to explore and explain the data; and

3. Identify technological "needs" and devise a proof of concept.

The course is designed as a gradual introduction to Python and relevant methods of legal data analysis, within a 24-hour schedule. The first 12/16 hours are meant to be a sufficient introduction for the purpose of the Final Presentation; the latter hours are reserved for more specialised methods and uses.

Each course will be built around three elements:

- Some opening considerations of a theoretical nature;
- Practical teaching, based on python scripts that students will run on their computer. Students will be invited to fiddle with the pre-inputted code and discover by themselves its limits; and
- A number of exercises based on the material covered.

Given the intensive schedule, the course can accommodate only a limited number of students.

SUGGESTED READING

- Data Science For Lawyers (available at https://www.datascienceforlawyers.org/);
- The Programming Historian (available at https://programminghistorian.org/);

- Al Sweigart, *Automate the Boring Stuff with Python: Practical Programming for Total Beginners* (No Starch Press 2015);
- Folgert Karsdorp, Mike Kestemont, Allen Riddell, *Humanities Data Analysis: Case Studies with Python* (Princeton University Press 2021);
- *AT&T Archives: The UNIX Operating System*, 1982, video available on Youtube; and
- Damien Charlotin & Wolfgang Alschner, 'Data Mining, Text Analytics, and Investor-State Arbitration' *forthcoming in* Pietro Ortolani et al. (eds.), *International Arbitration and Technology* (Wolters Kluwer 2022).

COURSE DETAIL

For each course, students will be provided with PyCharm material, available on a GitHub repository.

1. INTRODUCTION TO PYTHON

A) *Course 1 – Basics*

Theory : We'll start by introducing programming languages in general, and what distinguishes them. Students will learn what makes Python a good language for most tasks, but also how to find more resources to go further.

Practice : After introducing the basic commands and uses of PyCharm and the Console, we'll review: (i) Basic Operations; (ii) Variables; (iii) Functions; (iv) Syntax (if/else, loops); and (v) List Comprehension.

B) *Course 2 – Files and Data*

Theory : The challenges facing lawyers – and professionals in general – arising from the automation of their tasks and the increasing use of machines and robots.

Practice : Introduction to the following concepts: (i) Manipulating Files; (ii) Encoding; (iii) Data Storage; and (iv) Regexes.

C) *Course 3 – Dataframes*

Theory : We'll discuss version control methods and tools such as Git and Github.

Practice : The practical part of the course will introduce students to pandas and dataframes in general. Students will be invited to perform their first data analyses over a legal dataset, mapping the numbers of CADA requests over time and space. We'll also check if the number and type of decisions is altered as local and presidential elections loom.

D) *Course 4 – Basic NLP*

Theory : History and theory of Natural Language Processing (NLP), and its applications in day-to-day applications.

Practice : Introduction to NLP tools and methods. Exercises will focus on a dataset of contracts, as students will be asked to identify verbs, subjects, and different kind of provisions on that basis.

2. Collect and Scrap Data

E) Course 5 – Formatted Data

Theory : We'll start by discussing xml, html, and formatted data in general: its uses and logic in the context of sharing and analysing data. This will segue into a discussion of the Open Data movement, including in the context of French law.

Practice : Students will be introduced to the xPath syntax, and try their hand on a number of exercises based on XML docs released by the Conseil d'Etat's Open Data Initiative.

F) Course 6 – Scraping

Theory : Short considerations as to the ethics of scraping, and possible alternatives. We'll also discuss the distinction between static and dynamic websites.

Practice : Starting with requests, we'll scrap the (static) website of the Constitutional Council. Switching to Selenium, we'll scrap judgments from the Conseil d'Etat's (dynamic) Ariane database.

Time allowing, we'll introduce APIs, with examples (e.g., Google Doc; JudiLibre).

3. Organise, Clean, and Exploit a Dataset

G) Course 7 – Clean and Exploit a Dataset

Theory : Some considerations as to the turn to empirical study in legal practice & scholarship.

Practice : We'll go back to pandas and perform more sophisticated operations over a dataset of decisions from the Cour de cassation. In this context, we'll introduce basic plotting and statistical tools to investigate this jurisprudence.

H) Course 8 – Advanced NLP

Theory : Short discussion as to the use of NLP and other methods to boost litigation and arbitration skills.

Practice : We'll address similarity measures, vector-models, topic models. We'll apply these tools to the information collected from *LégiFrance*, and identify the evolution of French law over the years.

4. Advanced Topics

I) Course 9 – Networks

Theory : Introduction to network science and related methods and measures, and relevant classic legal scholarship. We'll also discuss uses of network analysis beyond case citation networks, such as cliques detection, recommendation algorithms, probability designs, contract/boilerplate analysis, visualisations, etc.

Practice : From a dataset of international court decisions (European Court of Human Rights or other international court or tribunal), we'll use regexes methods to create a dataset of nodes and edges.

We'll then study the resulting case citation network to identify sub-areas of jurisprudence, tally a relevant "study list" of important precedents, and compare this approach with classic textbook.

J) Course 10 – Machine Learning

Theory : What's behind the (overbroad) term "Artificial Intelligence", how to think about associated risk ("existential" or more mundane), what's the issue with so-called "algorithmic bias", etc. Given time we'll discuss ethics, the singularity, etc.

Practice : Introduction to supervised and unsupervised Machine Learning, with examples. Will discuss basic prediction models, random forests, fitting, etc.

Based on the dataset of Cour de cassation judgments, we'll feed a basic machine learning model with text data as well as outcome metadata (rejet du pourvoi/cassation), and see if we can create a model that scores better than chance (spoiler: we will).

Finally, we'll see how to review this model and evaluate it in terms of accuracy, robustness, etc.

K) Course 11 – Varia (Text Generation, APIs, etc.)

This last section will be used to review the entire course, deal with any code or script that had to be passed over, and answer any lingering questions.

Given enough time, we'll use GPT-3 or another language model (maybe JuriBert) to create contract or treaty provisions, or other legal documents, based on prompts, and/or to generate summaries.

L) Course 12 – Presentations

Final presentations by the students.

EVALUATION

- A presentation along the following lines:
    - Identify a dataset and a research question;
    - Collect and organise the data; and
    - Answer the research question.
- Students will be provided with examples and suggestions, but are invited to design their own analysis in any (legal) field that is of interest to them.
- The Python code used to perform the analysis will be taken into account for grading; elegance, efficiency, etc., are irrelevant (as long as it runs, it runs), but all students will be expected to be able to explain the logic behind coding choices and the input/output processes.

The presentation will (hopefully) take place in the offices of a law firm, legal department, or legal tech start-up.

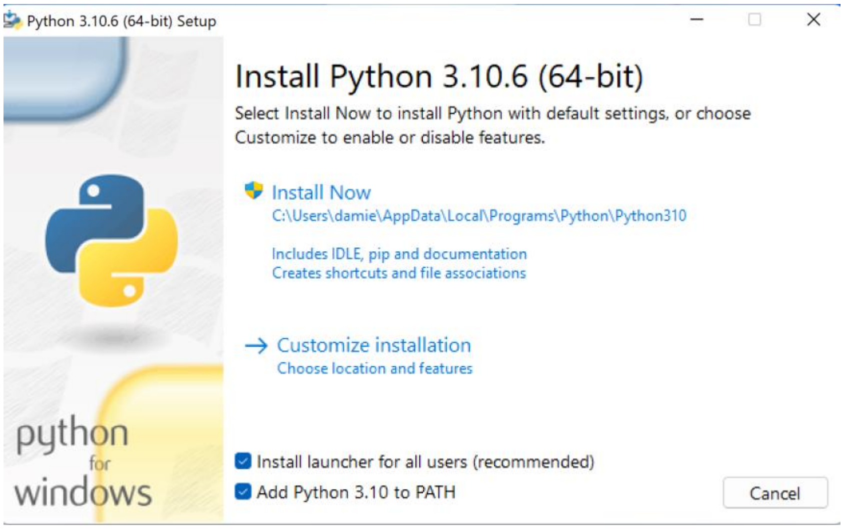
ANNEXES

ANNEX 1: INSTALL PYTHON ON YOUR COMPUTER

Follow this link:

https://www.python.org/downloads/

Select the version that works for you; if in doubt, the 64-bit version of your operating system should do. Download and install it, and make sure that the option “Add Python to PATH” is ticked (it is not by default). In “Customize Installation”, click on “Install for All Users”.



Python 3.10.6 (64-bit) Setup

Install Python 3.10.6 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\damie\AppData\Local\Programs\Python\Python310

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)

☑ Add Python 3.10 to PATH

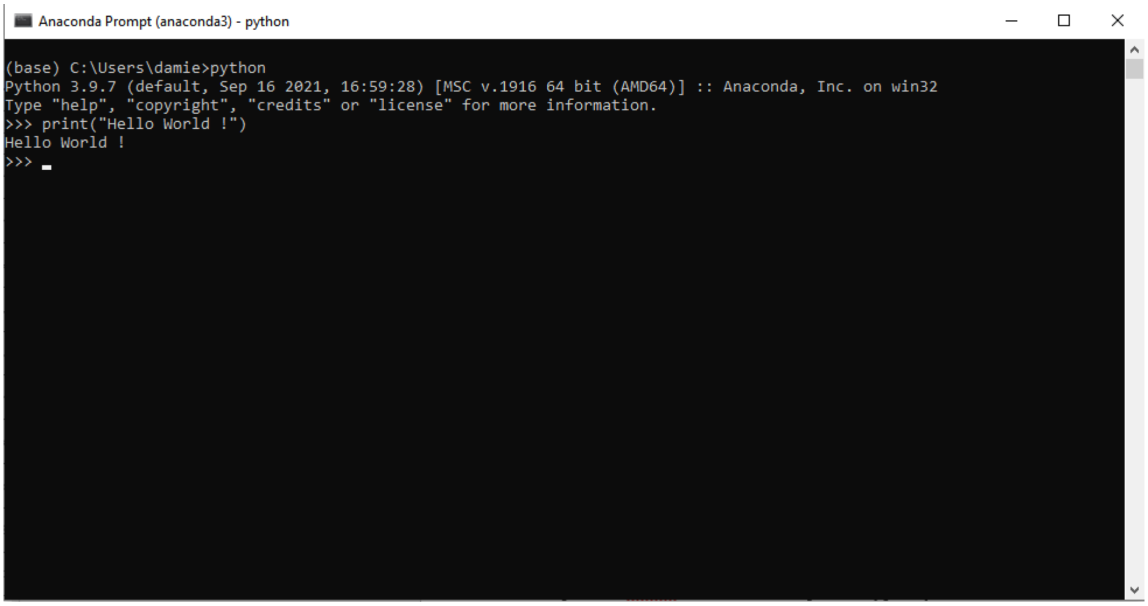
(For further info, Mac users can also follow the instructions here; Windows users here.)

Check that all is fine by opening “cmd” or “Command Prompt” (on Windows) or the “Terminal” (on Mac). Type “python”, and press Enter. If you see no error, you are good.[1] Then type (exactly) …

```
print(“Hello World !”)
```

… and check that it renders as follows:

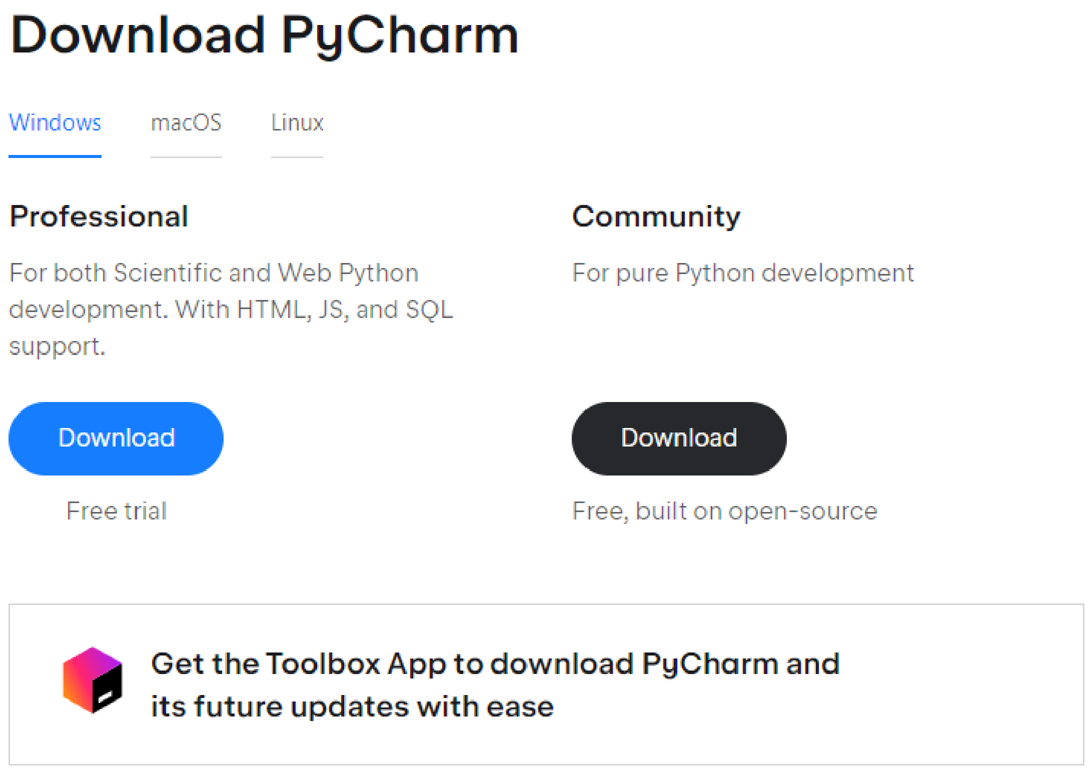
[1] Otherwise, there might be an issue with your environment variables, etc., which we’ll fix together.



```
(base) C:\Users\damie>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World !")
Hello World !
>>> 
```

ANNEX 2 : INSTALL PYCHARM

PyCharm can be found here. Be sure to install the "Community" (free) version.



Download PyCharm

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Free trial

Community

For pure Python development

Free, built on open-source

Get the Toolbox App to download PyCharm and its future updates with ease

Click Download (right button), then follow the installation process.

Annex 3 : Install a Source Code Editor [Optional]

This is optional, though could prove handy as we study structured data in XML files.

Windows users may want to install NotePad++ (https://notepad-plus-plus.org/), which is a better version of the basic notepad, and allows you to see and investigate coding scripts in more details.

Mac OS users can refer to this article. https://setapp.com/how-to/alternative-to-notepad++-for-mac. Out of the alternatives proposed, Sublime Text and Atom seem the most relevant.

Annex 4 : Download the Course

You will need:

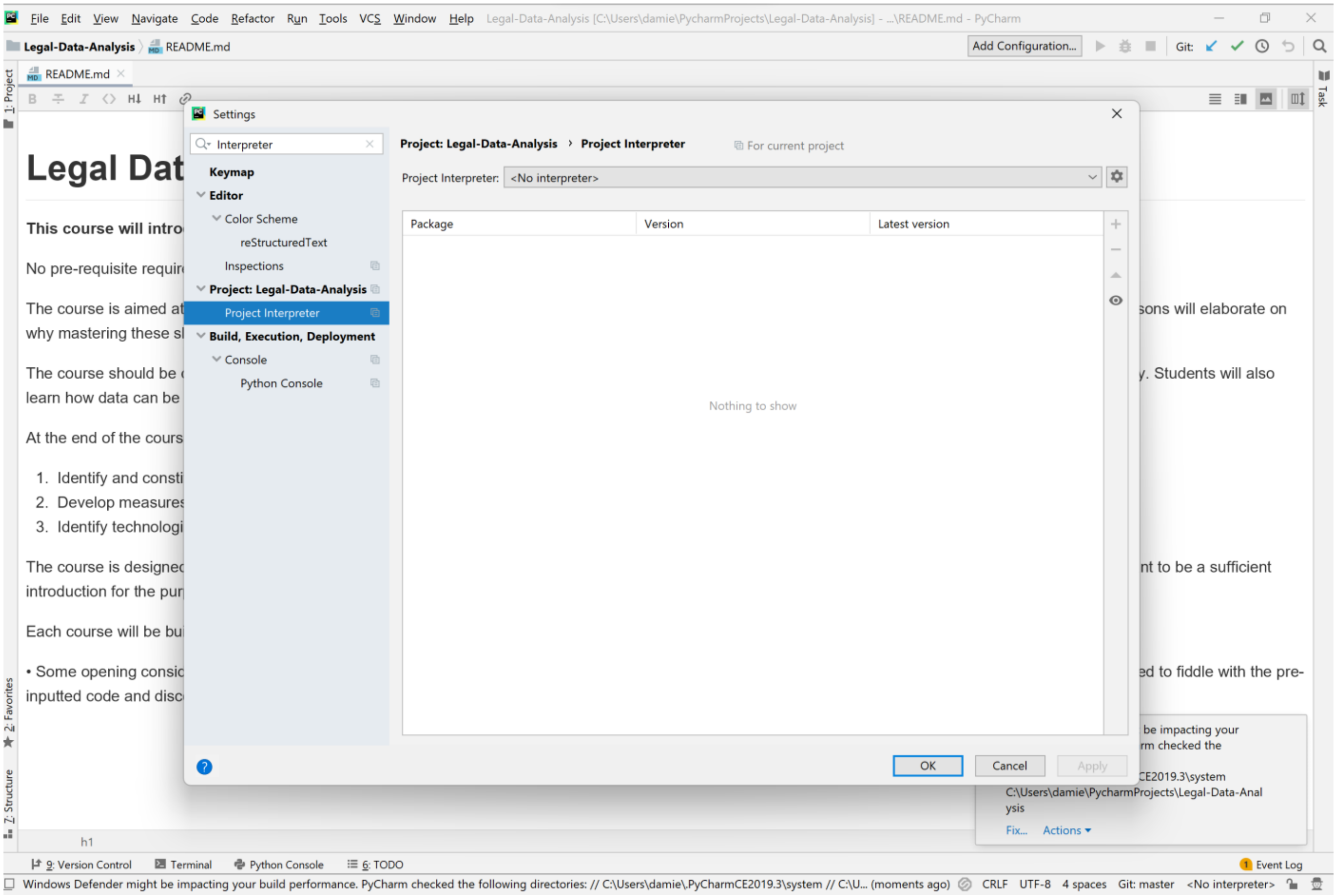
1. To create a GitHub account here.
2. Send your GitHub account name to the instructor.
3. Wait for the invitation to join the GitHub Repository holding the course archive.

The course's repository is here (can be accessed only after invitation).

Annex 5 : Bring it All Together

Once you have been added to the course repository:

- Start PyCharm;
- In the "Projects" window, click on "Get from VCS/Version Control"; from this, either:
  - Input your GitHub details, and you should be able to see the course's repo; or
  - Input the course's repository URL, and click on "Clone".
- Go to Settings, and in the search box type "Interpreter":
  - There should be two results, named "Python Interpreter" and "Console Interpreter".
  - Make sure **both** interpreters refer to a path name that ends with "python.exe".
    - If there is nothing there, click on the right-hand-side wheel, then "Add…", and then you should be able to find a "python.exe" path name in "System Interpreter" (on the left-hand side).
    - After clicking "Apply", wait a few minutes, or even restart PyCharm, to be able to use the Console (see below).



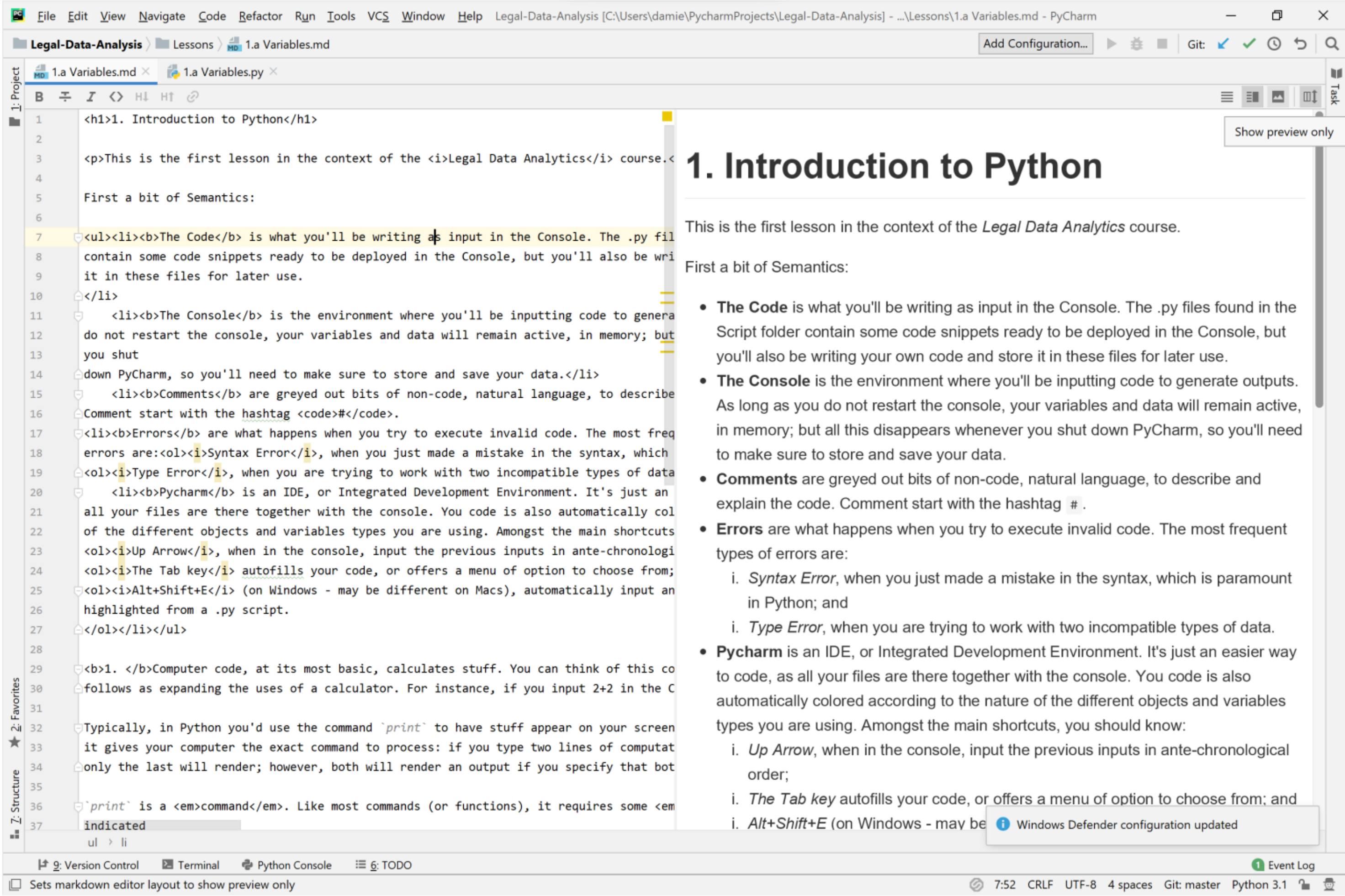
Project Interpreter: <No interpreter>

```
Windows Defender might be impacting your build performance. PyCharm checked the following directories: // C:\Users\damie\.PyCharmCE2019.3\system // C:\U... (moments ago)
```

Figure 1: What you see if no interpreter is set

Display the “Project” oane on the left-hand side. The written course can be found in the **Lessons** folder, with the corresponding code in the **Scripts** Folder. For ease of use, follow this process:

- Open both the .md and .py files with the relevant lessons;
- Click on “Preview Only” for the .md file (top right of the pane, third button);


File Edit View Navigate Code Refactor Run Tools VCS Window Help Legal-Data-Analysis [C:\Users\damie\PycharmProjects\Legal-Data-Analysis] - ...\Lessons\1.a Variables.md - PyCharm
Legal-Data-Analysis
Lessons
1.a Variables.md
Add Configuration...
Git:
1.a Variables.md
1.a Variables.py
1: Project
2: Favorites
7: Structure
Task
Show preview only
ul › li
9: Version Control
Terminal
Python Console
6: TODO
Event Log
Windows Defender configuration updated
7:52 CRLF UTF-8 4 spaces Git: master Python 3.1

```
<h1>1. Introduction to Python</h1>

<p>This is the first lesson in the context of the <i>Legal Data Analytics</i> course.<

First a bit of Semantics:

<ul><li><b>The Code</b> is what you'll be writing as input in the Console. The .py fil
contain some code snippets ready to be deployed in the Console, but you'll also be wri
it in these files for later use.
</li>
    <li><b>The Console</b> is the environment where you'll be inputting code to genera
do not restart the console, your variables and data will remain active, in memory; but
you shut
down PyCharm, so you'll need to make sure to store and save your data.</li>
    <li><b>Comments</b> are greyed out bits of non-code, natural language, to describe
Comment start with the hashtag <code>#</code>.
<li><b>Errors</b> are what happens when you try to execute invalid code. The most freq
errors are:<ol><i>Syntax Error</i>, when you just made a mistake in the syntax, which
<ol><i>Type Error</i>, when you are trying to work with two incompatible types of data
    <li><b>Pycharm</b> is an IDE, or Integrated Development Environment. It's just an
all your files are there together with the console. You code is also automatically col
of the different objects and variables types you are using. Amongst the main shortcuts
<ol><i>Up Arrow</i>, when in the console, input the previous inputs in ante-chronologi
<ol><i>The Tab key</i> autofills your code, or offers a menu of option to choose from;
<ol><i>Alt+Shift+E</i> (on Windows - may be different on Macs), automatically input an
highlighted from a .py script.
</ol></li></ul>

<b>1. </b>Computer code, at its most basic, calculates stuff. You can think of this co
follows as expanding the uses of a calculator. For instance, if you input 2+2 in the C

Typically, in Python you'd use the command `print` to have stuff appear on your screen
it gives your computer the exact command to process: if you type two lines of computat
only the last will render; however, both will render an output if you specify that bot

`print` is a <em>command</em>. Like most commands (or functions), it requires some <em
indicated
```

1. Introduction to Python

This is the first lesson in the context of the *Legal Data Analytics* course.

First a bit of Semantics:

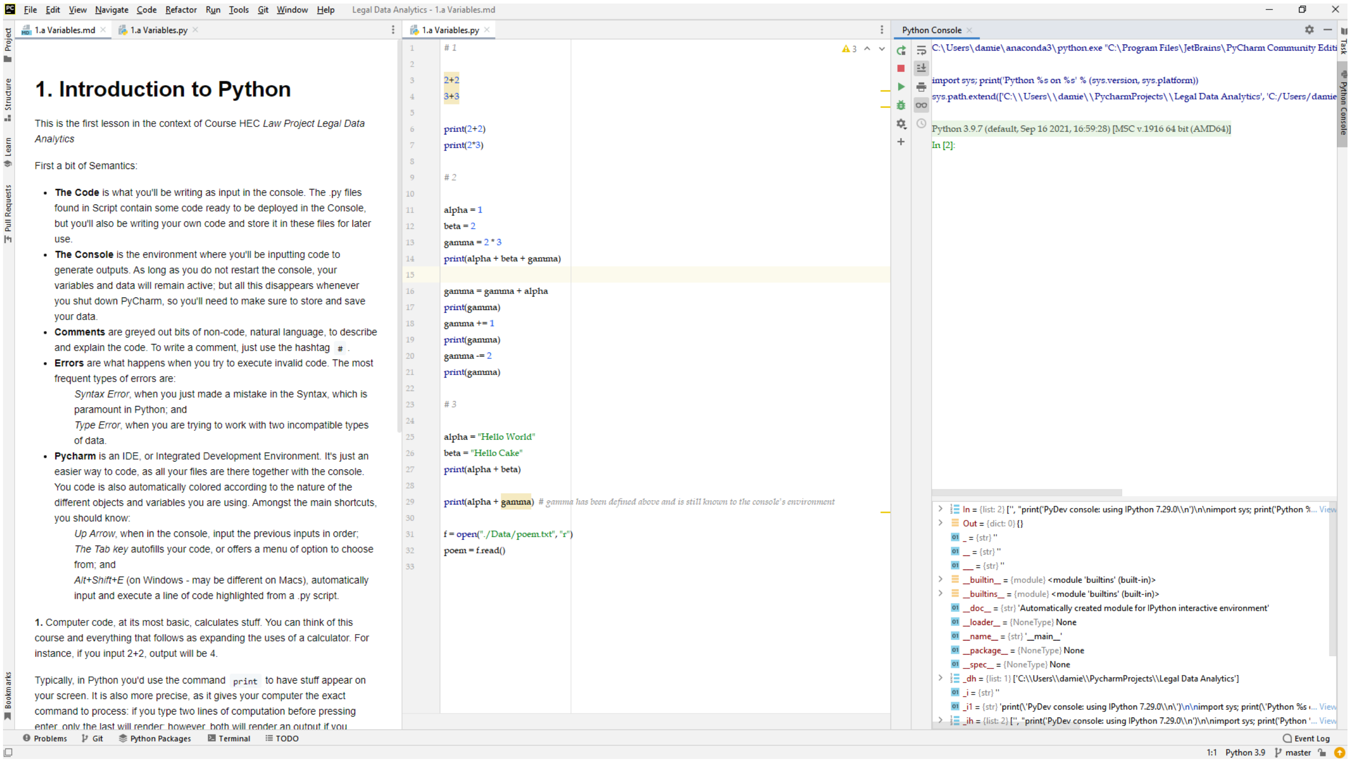
- **The Code** is what you'll be writing as input in the Console. The .py files found in the Script folder contain some code snippets ready to be deployed in the Console, but you'll also be writing your own code and store it in these files for later use.
- **The Console** is the environment where you'll be inputting code to generate outputs. As long as you do not restart the console, your variables and data will remain active, in memory; but all this disappears whenever you shut down PyCharm, so you'll need to make sure to store and save your data.
- **Comments** are greyed out bits of non-code, natural language, to describe and explain the code. Comment start with the hashtag `#`.
- **Errors** are what happens when you try to execute invalid code. The most frequent types of errors are:
  i. *Syntax Error*, when you just made a mistake in the syntax, which is paramount in Python; and
  i. *Type Error*, when you are trying to work with two incompatible types of data.
- **Pycharm** is an IDE, or Integrated Development Environment. It's just an easier way to code, as all your files are there together with the console. You code is also automatically colored according to the nature of the different objects and variables types you are using. Amongst the main shortcuts, you should know:
  i. *Up Arrow*, when in the console, input the previous inputs in ante-chronological order;
  i. *The Tab key* autofills your code, or offers a menu of option to choose from; and
  i. *Alt+Shift+E* (on Windows - may be

Sets markdown editor layout to show preview only

Figure 2: Show Preview Only, on top right; Project pane on top left

- From the top menu, select Window > Editor Tab > Split Vertically or Split Right, until you can see both the .md and .py files;
- Close the Project pane on the left-hand side; and
- Click on PyCharm's "Python Console" pane, which by default is at the bottom. Dock it on the "Right-Top" of the IDE, by clicking on the Wheel > Move To.

Final result should look like this:



1. Introduction to Python

This is the first lesson in the context of Course HEC *Law Project Legal Data Analytics*

First a bit of Semantics:

- **The Code** is what you'll be writing as input in the console. The .py files found in Script contain some code ready to be deployed in the Console, but you'll also be writing your own code and store it in these files for later use.
- **The Console** is the environment where you'll be inputting code to generate outputs. As long as you do not restart the console, your variables and data will remain active; but all this disappears whenever you shut down PyCharm, so you'll need to make sure to store and save your data.
- **Comments** are greyed out bits of non-code, natural language, to describe and explain the code. To write a comment, just use the hashtag `#`.
- **Errors** are what happens when you try to execute invalid code. The most frequent types of errors are:
  Syntax Error, when you just made a mistake in the Syntax, which is paramount in Python; and
  Type Error, when you are trying to work with two incompatible types of data.
- **Pycharm** is an IDE, or Integrated Development Environment. It's just an easier way to code, as all your files are there together with the console. You code is also automatically colored according to the nature of the different objects and variables you are using. Amongst the main shortcuts, you should know:
  Up Arrow, when in the console, input the previous inputs in order;
  The Tab key autofills your code, or offers a menu of option to choose from; and
  Alt+Shift+E (on Windows - may be different on Macs), automatically input and execute a line of code highlighted from a .py script.

1. Computer code, at its most basic, calculates stuff. You can think of this course and everything that follows as expanding the uses of a calculator. For instance, if you input 2+2, output will be 4.

Typically, in Python you'd use the command `print` to have stuff appear on your screen. It is also more precise, as it gives your computer the exact command to process: if you type two lines of computation before pressing enter, only the last will render; however, both will render an output if you

```python
# 1

2+2
3+3

print(2+2)
print(2*3)

# 2

alpha = 1
beta = 2
gamma = 2 * 3
print(alpha + beta + gamma)

gamma = gamma + alpha
print(gamma)
gamma += 1
print(gamma)
gamma -= 2
print(gamma)

# 3

alpha = "Hello World"
beta = "Hello Cake"
print(alpha + beta)

print(alpha + gamma) # gamma has been defined above and is still known to the console's environment

f = open("./Data/poem.txt", "r")
poem = f.read()

```

```
C:\Users\damie\anaconda3\python.exe "C:\Program Files\JetBrains\PyCharm Community Editi

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\damie\\PycharmProjects\\Legal Data Analytics', 'C:/Users/damie

Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
In [2]:
```

```
In = {list: 2} ['', "print('PyDev console: using IPython 7.29.0\\n')\n\nimport sys; print('Python %... View
Out = {dict: 0} {}
_ = {str} ''
__ = {str} ''
___ = {str} ''
__builtin__ = {module} <module 'builtins' (built-in)>
__builtins__ = {module} <module 'builtins' (built-in)>
__doc__ = {str} 'Automatically created module for IPython interactive environment'
__loader__ = {NoneType} None
__name__ = {str} '__main__'
__package__ = {NoneType} None
__spec__ = {NoneType} None
_dh = {list: 1} ['C:\\Users\\damie\\PycharmProjects\\Legal Data Analytics']
_i = {str} ''
_i1 = {str} 'print(\'PyDev console: using IPython 7.29.0\\n\')\n\nimport sys; print(\'Python %s ... View
_ih = {list: 2} ['', "print('PyDev console: using IPython 7.29.0\\n')\n\nimport sys; print('Python '... View
```

Figure 3: Final Result, you are ready to Code

Annex 6 : Install iPython for greater ease

One last step:

- Follow the instructions to install pip that you can find here. You will need to open the Command Prompt (for Windows) or Terminal (for Mac);
- Still in the Prompt/Terminal, type "pip install ipython"; and
- If installation was successful, restart PyCharm, and then the Console. Instead of chevrons (">>>"), the Console should indicate "In [1]:".

If you encounter any issue prior to the course starting, email me at: damien.charlotin@sciencespo.fr.