[**NOTE**: you can download an `.ipynb` version of this file here]

Lab 1 - Getting up to speed

This lab has two main parts. In the first one, we will make sure you are signed up for all the services we will be using throughout the course. This way, when the time comes, you'll be ready to go and will be able to jump into the task right away. In the second part, we will introduce/refresh the Jupyter Notebook, a file format that is common in scientific computing and that allows us to combine formatted text with code and rich content from the web such as image, video, or interactive maps.

Sign up for relevant services

Much of the practical aspect of this course will rely on web products that provide functionality through the internet. Every service we will use in class has at least a free tier, but you will need to sign up before hand.

Here is a list of the services you will need to register:

- `CDRC Data`: we will use some of the data provided by the CDRC, so a (free) account with them will be necessary.
- `CARTO`: CARTO provides an online platform for web mapping and location intelligence. They have a free tier that you can try for a limited amount of time, but the Geographic Data Science Lab has a subscription that will allow you to use it beyond the basic allowance. Do not sign up for the free account but instead make sure you have received the invite from the module leader.
- `Mapbox`: Mapbox is one of the industry leaders in web mapping. Their free tier is rather generous so will more than suffice for what we will do within the course. You can sign up for a new (free) account here.

In addition, below is a list of software we will also rely on during the course. They are all free and open-source, so you should be able to install them on your own laptop as well:

- `QGIS`: the stable version (3.4 at the time of writing) is OK, any more recent version will also work. If you use a version older than 3.8, please make sure you also have installed the `XYZ Tiles` plugin.
- `Python`: at parts of the course, we will rely on Python to read, manipulate and write geospatial data. There are several libraries required, all of them shared with the `GDS'19` course so, if you have a stack installed for it, you should be fine. If not, checkout the install instructions.

- `R`: we will not require it in the course but, if you are a fan of the statistical platform R, we will explore a bit how R can be linked to web mapping. This is an optional install.

The Jupyter Notebook

```
from IPython.display import Image
```

Cells

The main building block of notebooks are cells. These are chunks of the same time of content which can be cut, pasted, and moved around in a notebook. Cells can be of two types:

- **Text**, like the one where this is written.
- **Code**, like the following one below:

```
# This is a code cell
```

For example, you can create a new cell by searching for "Insert Cell". By default, this will be a code cell, but you can change that on the `Cell` -> `Cell Type` menu. Choose `Markdown` for a text cell. Once a new cell is created, you can edit it by clicking on it, which will create the cursor bar inside for you to start typing.

Pro tip!: cells can also be created with shortcuts. If you press `<escape>` and then `b` (`a`), a new cell will be created below (above). There is a whole bunch of shortcuts you can explore by pressing `<escape>` and `h` (press `<escape>` again to leave the help).

Code and its output

A particularly useful feature of notebooks is that you can save, in the same place, the code you use to generate any output (tables, figures, etc.). As an example, the cell below contains a snipet of Python that returns a printed statement. This statement is then printed below and recorded in the notebook as output:

```
print("Hello world!!!")
```

```
Hello world!!!
```

Note also how the notebook has automatic syntax highlighting support for Python. This makes the code much more readable and understandable. More on Python below.

Markdown

Text cells in a notebook use the Github Flavored Markdown markup language. This means you can write plain text with some rules and the notebook renders a more visually appealing version of it. Let's see some examples:

- **BOLD**:

  `This is **bold**.`
  Is rendered:
  This is **bold**.

- **ITALIC**:

  `This is *italic*.`
  Is rendered:
  This is *italic*.

- **LISTS**:

  You can create unnumbered lists:

```
* Item 1
* Item 2
* ...
```

  Which will produce:

- Item 1
- Item 2
- …

  Or you can create numbered lists:

```
1. First element
1. Second element
1. ...
```

  And get:

1. First element
2. Second element
3. …

  Note that you don't have to write the actual number of the element, just using `1.` always produces a numbered list.

  You can also nest lists:

```
* First unnumbered element, which can be split into:

    1. One numbered element
    2. Another numbered element

* Second element.
* ...
```

- First unnumbered element, which can be split into:

  1. One numbered element
  2. Another numbered element

- Second element.
- ...

This creates many oportunities to combine things nicely.

- **LINKS**

  ```
  You can easily create hyperlinks, for example to [WikiPedia](https://www.wikipedia.org/).
  ```
  You can easily create hyperlinks, for example to WikiPedia.

- **HEADINGS**: including `#` before a line causes it to render a heading.

```
# This is Header 1
```
Turns into:

This is Header 1

```
## This is Header 2
```
Turns into:

This is Header 2

And so on...

You can see a more in detail introduction in the following links:

https://help.github.com/articles/markdown-basics/

https://help.github.com/articles/github-flavored-markdown/

Rich content in a notebook

Notebooks can also include rich content from the web. For that, we need to import the `display` module:

```
import IPython.display as display
```

This makes available additional functionality that allows us to embed rich content. For example, we can include a YouTube clip easily by passing it's ID:

```
display.YouTubeVideo('iinQDhsdE9s')
```

```
    <iframe
        width="400"
        height="300"
        src="https://www.youtube.com/embed/iinQDhsdE9s"
        frameborder="0"
        allowfullscreen
    ></iframe>
```

Or we can pass standard HTML code:

```
display.HTML("""<table>
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>""")
```

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Note that this opens the door for including a large number of elements from the web, as an `iframe` is also allowed. For example, interactive maps can be included:

```
osm = """
<iframe width="425" height="350" frameborder="0" scrolling="no" marginheight="0" marginwidth="0" src="h
"""
display.HTML(osm)
```

View Larger Map

Or sound content:

```
sound = '''
<iframe width="100%" height="300" scrolling="no" frameborder="no" allow="autoplay" src="https://w.sound
display.HTML(sound)
```

A more thorough exploration of them is available in this notebook.

Exercise to work on your own

Try to reproduce, using markdown and the different tools the notebook affords you, the following WikiPedia entry:

https://en.wikipedia.org/wiki/Chocolate_chip_cookie_dough_ice_cream

```
display.IFrame('https://en.wikipedia.org/wiki/Chocolate_chip_cookie_dough_ice_cream',
               700, 500)
```

```
<iframe
    width="700"
    height="500"
    src="https://en.wikipedia.org/wiki/Chocolate_chip_cookie_dough_ice_cream"
    frameborder="0"
    allowfullscreen
></iframe>
```

Do not over think it. Focuse and pay special attention to getting the bold, italics, links, headlines and lists correctly formated, but don't worry too much about the overall layout. Bonus if you manage to insert the image as well (it does not need to be properly placed as in the original page)!