|Sustainable water management

Python for Hydrology

Session 1 – Anaconda Interface

Objective:

Explore the Conda commands, tools, and widgets inside the Anaconda interface and the Jupyter Lab tool

Conda

In this course, we are not dealing with the full Anaconda distribution; we are dealing with the "compressed" version called Miniconda, but regardless of the distribution, all come with the "conda" build.

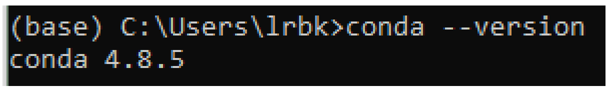
To update your Conda, open an Anaconda Prompt, and type the following

```
conda upgrade conda
```

Typically all the Conda commands take some time to run; it has its repository too, which contains lots of packages and GUI as Jupyter Lab that we previously installed, and do not think that Miniconda/Anaconda is exclusive to Python, you can install almost any programming language and use it with Jupyter Lab for example

If you want to see your conda version, type the following command to see in the kernel your current version. You do not need to update frequently because most of the packages change for most advanced features that you may not use.

```
conda --version
```



```
(base) C:\Users\lrbk>conda --version
conda 4.8.5
```

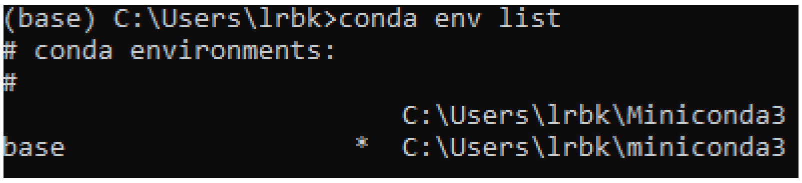
What you may want is to get the most up-to-date features of Jupyter Lab. It can quickly get the most up-to-date version with the following script. It may ask you if you want to update the packages, type "y" for yes.

```
conda update jupyterlab
```

Another useful tool of Conda is creating multiple separate environments, but what is an environment? An environment is like an isolated installation of Python, in which you can have packages. For example, you can have installed package "A" in one environment, but it may not be in "B"; it depends on your needs, or if you want to test an application or package.

To list the environments you have, you can use the following command:

```
conda env list
```

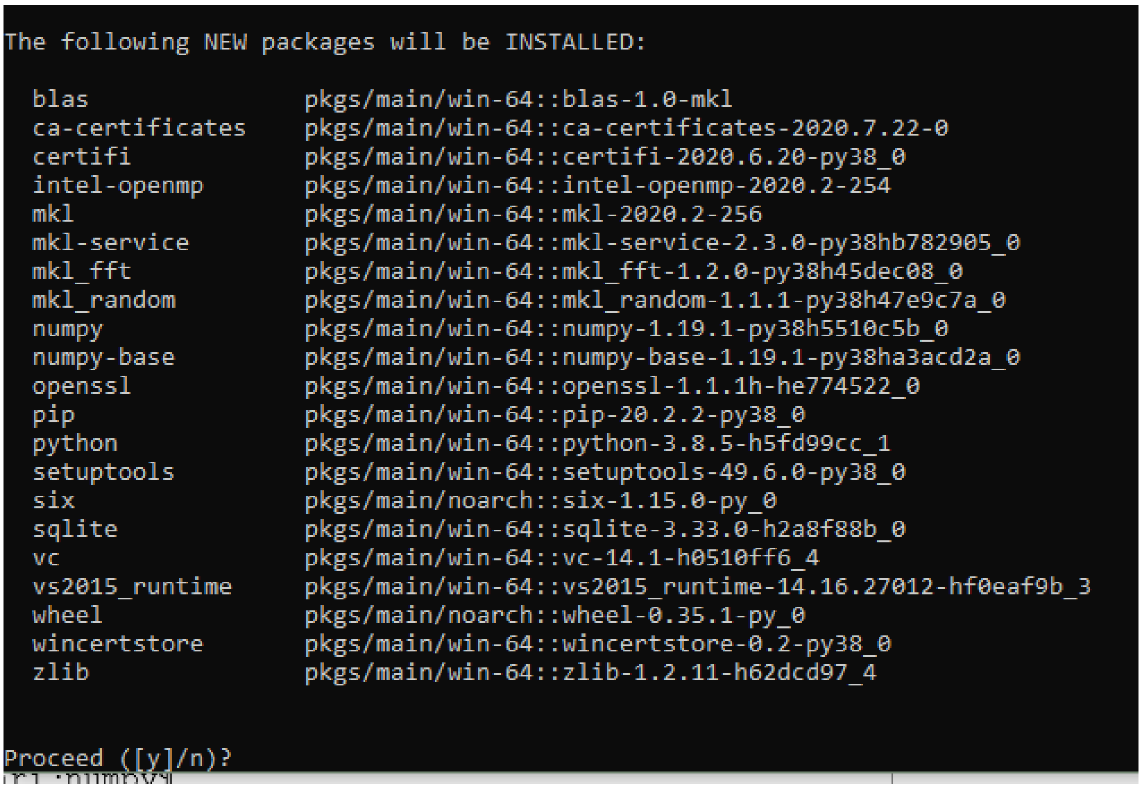


```
(base) C:\Users\lrbk>conda env list
# conda environments:
#
                         C:\Users\lrbk\Miniconda3
base                  *  C:\Users\lrbk\miniconda3
```

If you want to create a new environment called "hatari" with the python package "numpy" inside, type the following command, but beware because it installs Numpy from the "conda repository" it is useful because it is not attached to a programming language, you can install many features of different programming languages with it

```
conda create --name hatari numpy
```

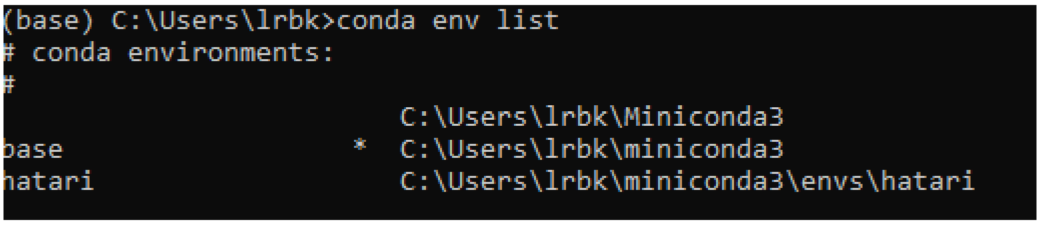
The previous command asks you for confirmation for the installation. What it wants to install are all the packages you would need to run Numpy, so pres "y"



```
The following NEW packages will be INSTALLED:

  blas               pkgs/main/win-64::blas-1.0-mkl
  ca-certificates    pkgs/main/win-64::ca-certificates-2020.7.22-0
  certifi            pkgs/main/win-64::certifi-2020.6.20-py38_0
  intel-openmp       pkgs/main/win-64::intel-openmp-2020.2-254
  mkl                pkgs/main/win-64::mkl-2020.2-256
  mkl-service        pkgs/main/win-64::mkl-service-2.3.0-py38hb782905_0
  mkl_fft            pkgs/main/win-64::mkl_fft-1.2.0-py38h45dec08_0
  mkl_random         pkgs/main/win-64::mkl_random-1.1.1-py38h47e9c7a_0
  numpy              pkgs/main/win-64::numpy-1.19.1-py38h5510c5b_0
  numpy-base         pkgs/main/win-64::numpy-base-1.19.1-py38ha3acd2a_0
  openssl            pkgs/main/win-64::openssl-1.1.1h-he774522_0
  pip                pkgs/main/win-64::pip-20.2.2-py38_0
  python             pkgs/main/win-64::python-3.8.5-h5fd99cc_1
  setuptools         pkgs/main/win-64::setuptools-49.6.0-py38_0
  six                pkgs/main/noarch::six-1.15.0-py_0
  sqlite             pkgs/main/win-64::sqlite-3.33.0-h2a8f88b_0
  vc                 pkgs/main/win-64::vc-14.1-h0510ff6_4
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.16.27012-hf0eaf9b_3
  wheel              pkgs/main/noarch::wheel-0.35.1-py_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py38_0
  zlib               pkgs/main/win-64::zlib-1.2.11-h62dcd97_4


Proceed ([y]/n)?
```

If we recheck the list of environments, we could see the new one added, the symbol "*" indicates the one that is active

```
conda env list
```

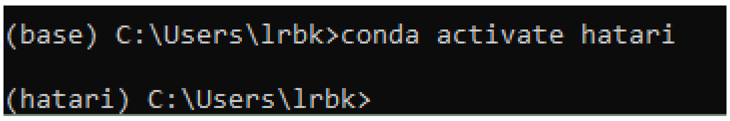


```
(base) C:\Users\lrbk>conda env list
# conda environments:
#
                         C:\Users\lrbk\Miniconda3
base                  *  C:\Users\lrbk\miniconda3
hatari                   C:\Users\lrbk\miniconda3\envs\hatari
```

To change of environment, type the following command:

```
conda activate hatari
```

And instead of "base" we could see that we are in the "hatari" environment



```
(base) C:\Users\lrbk>conda activate hatari

(hatari) C:\Users\lrbk>
```

Being here, if we want to open “Jupyter lab” we are not going to do be able to do so.

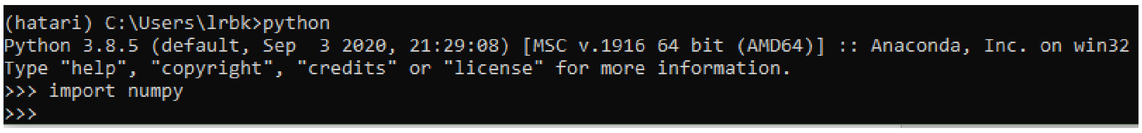
```
jupyter lab
```

As we installed Numpy, we should be able to use it, so open Python (this works for any environment)

```
python
```

And import Numpy. As you can see, we don’t get any error because Conda has installed the package correctly.

```
import numpy
```

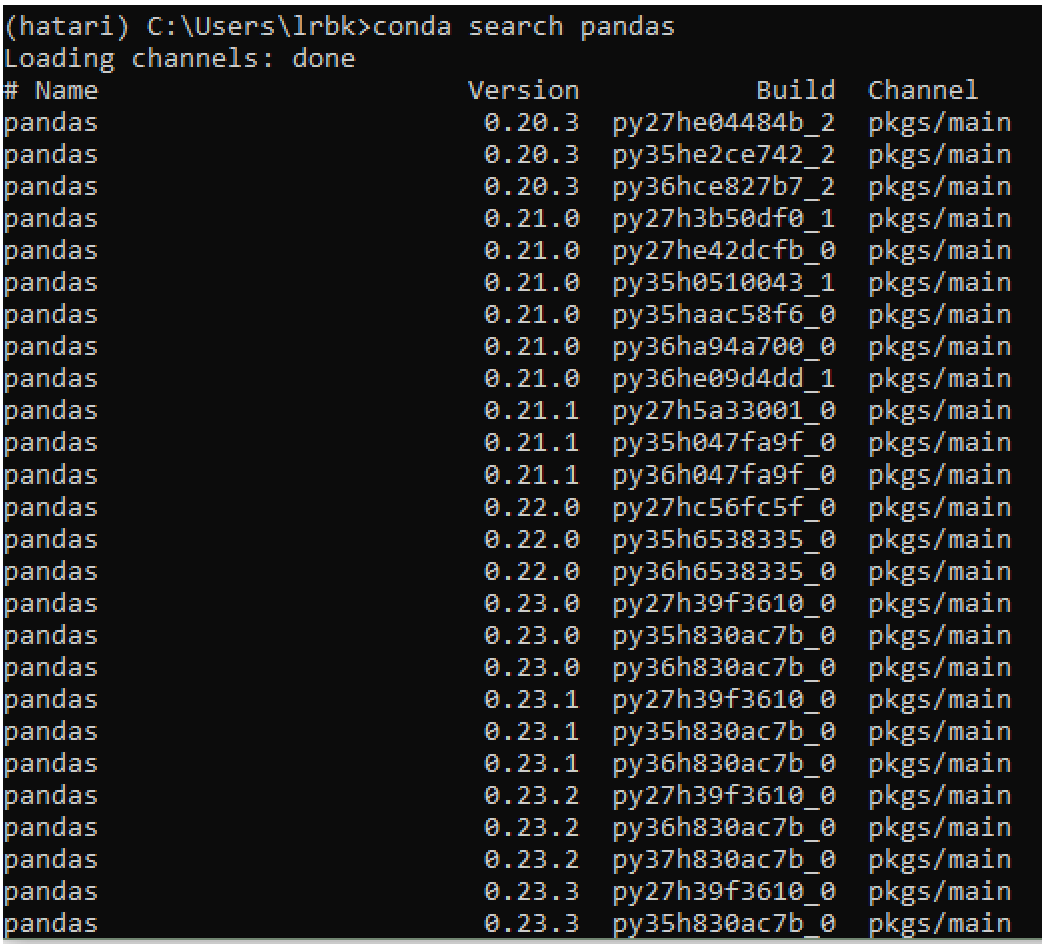


```
(hatari) C:\Users\lrbk>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>>
```

To go out of Python, press “Ctrl+Z” and the “enter” button from your keyboard

Conda allows looking for a variety of packages it has. For example, if we want to know if a package is available in Conda, type the following command, which returns you with a list of all the versions of Pandas available inside Conda with their Python version.

```
conda search pandas
```



```
(hatari) C:\Users\lrbk>conda search pandas
Loading channels: done
# Name                       Version           Build  Channel
pandas                        0.20.3  py27he04484b_2  pkgs/main
pandas                        0.20.3  py35he2ce742_2  pkgs/main
pandas                        0.20.3  py36hce827b7_2  pkgs/main
pandas                        0.21.0  py27h3b50df0_1  pkgs/main
pandas                        0.21.0  py27he42dcfb_0  pkgs/main
pandas                        0.21.0  py35h0510043_1  pkgs/main
pandas                        0.21.0  py35haac58f6_0  pkgs/main
pandas                        0.21.0  py36ha94a700_0  pkgs/main
pandas                        0.21.0  py36he09d4dd_1  pkgs/main
pandas                        0.21.1  py27h5a33001_0  pkgs/main
pandas                        0.21.1  py35h047fa9f_0  pkgs/main
pandas                        0.21.1  py36h047fa9f_0  pkgs/main
pandas                        0.22.0  py27hc56fc5f_0  pkgs/main
pandas                        0.22.0  py35h6538335_0  pkgs/main
pandas                        0.22.0  py36h6538335_0  pkgs/main
pandas                        0.23.0  py27h39f3610_0  pkgs/main
pandas                        0.23.0  py35h830ac7b_0  pkgs/main
pandas                        0.23.0  py36h830ac7b_0  pkgs/main
pandas                        0.23.1  py27h39f3610_0  pkgs/main
pandas                        0.23.1  py35h830ac7b_0  pkgs/main
pandas                        0.23.1  py36h830ac7b_0  pkgs/main
pandas                        0.23.2  py27h39f3610_0  pkgs/main
pandas                        0.23.2  py36h830ac7b_0  pkgs/main
pandas                        0.23.2  py37h830ac7b_0  pkgs/main
pandas                        0.23.3  py27h39f3610_0  pkgs/main
pandas                        0.23.3  py35h830ac7b_0  pkgs/main
```

To deactivate the environment "hatari" use the following command:

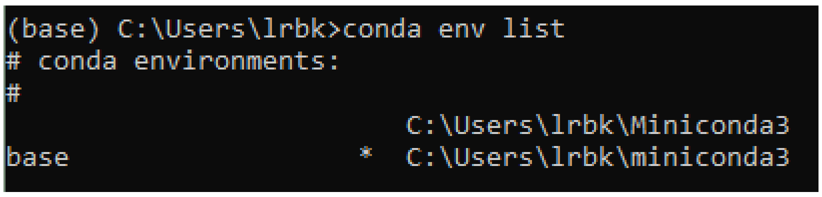
```
deactivate
```

If we want to remove it, use the following command, it is going to ask you for confirmation, so choose "y"

```
conda remove --name hatari --all
```

Check the environment list again

```
conda env list
```



```
(base) C:\Users\lrbk>conda env list
# conda environments:
#
                         C:\Users\lrbk\Miniconda3
base                  *  C:\Users\lrbk\miniconda3
```

So until this point, you can see that Conda is a package manager for our installation; it is useful and allows us to have easy installations of Python inside any operating system.

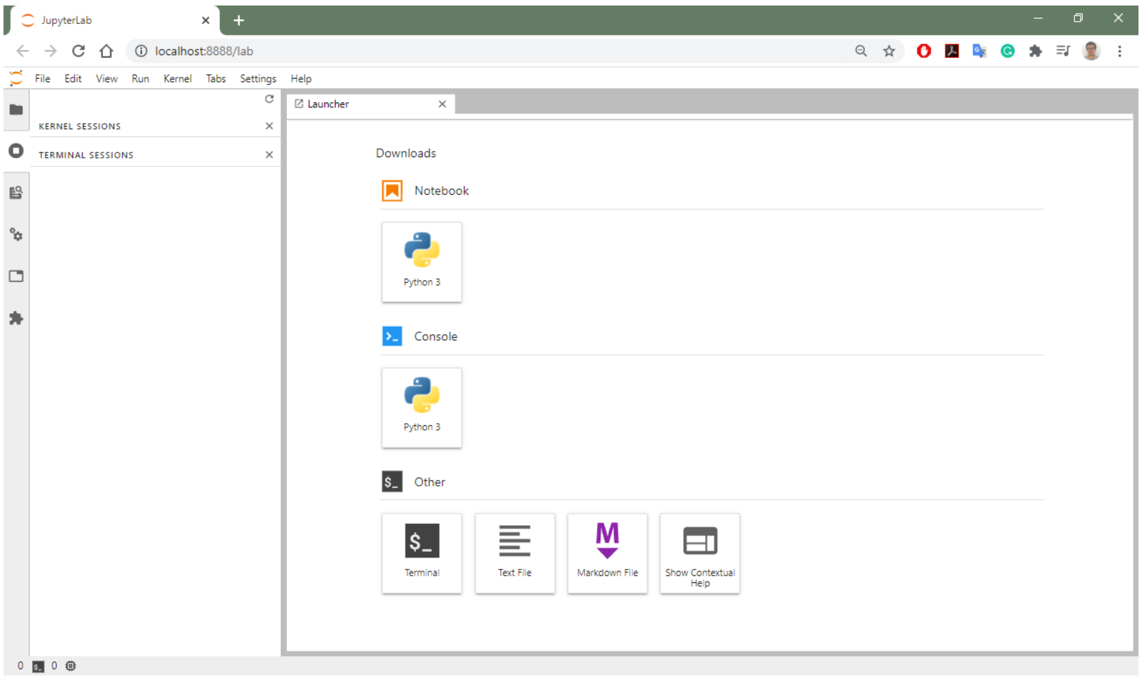
Jupyter Lab

Jupyter Lab is one of the most and best-used notebooks for developing code in Python, principally because of its ease of having fast visualizations of what we are doing.

To open Jupyter Lab open an Anaconda Prompt and write the following text:

```
jupyter lab
```

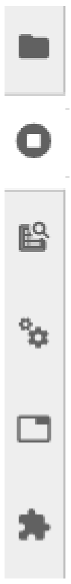
We get the following visualization



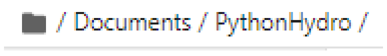
As a reference to the structure of Jupyter Lab, let's go through the parts of it.

At the left of Jupyter, you could see a column with icons, which are from top to bottom:

- File Browser
- Running terminals and kernels
- Commands
- Property Inspector
- Open Tabs
- Extension Manager

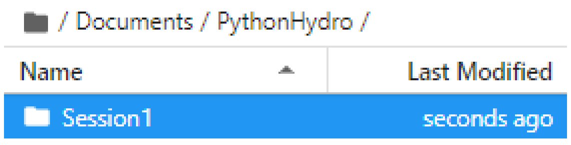


Go to the **File Browser**, and as default, it would appear in your Downloads, get familiar with it and look for the '**Documents**" path, here create a new folder called "**PythonHydro"** and go inside this folder, from now we are going to work in this folder for all the sessions.



/ Documents / PythonHydro /

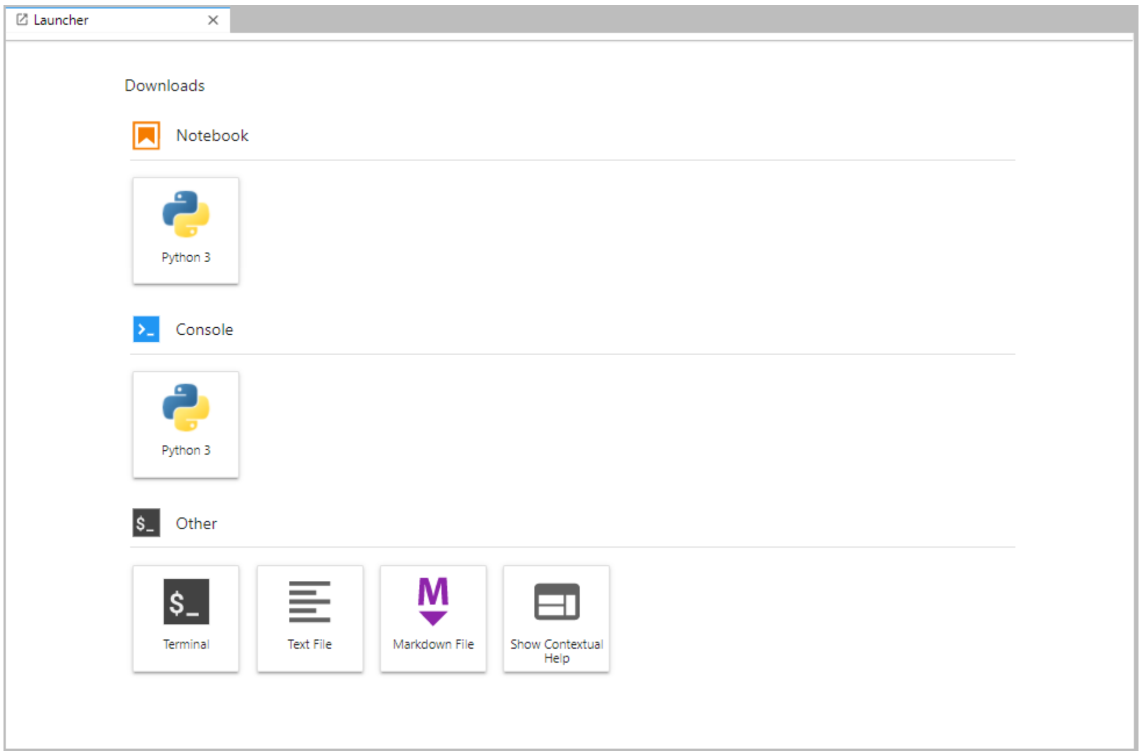
Create a new folder called "**Session1**". Go inside Session1.



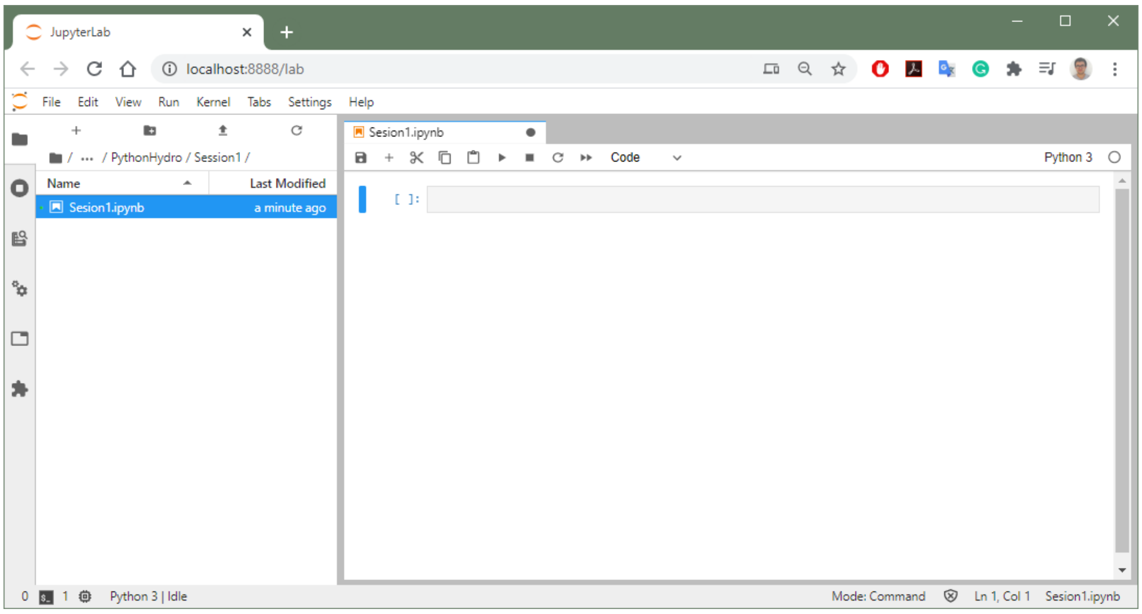
/ Documents / PythonHydro /

Name	Last Modified
Session1	seconds ago

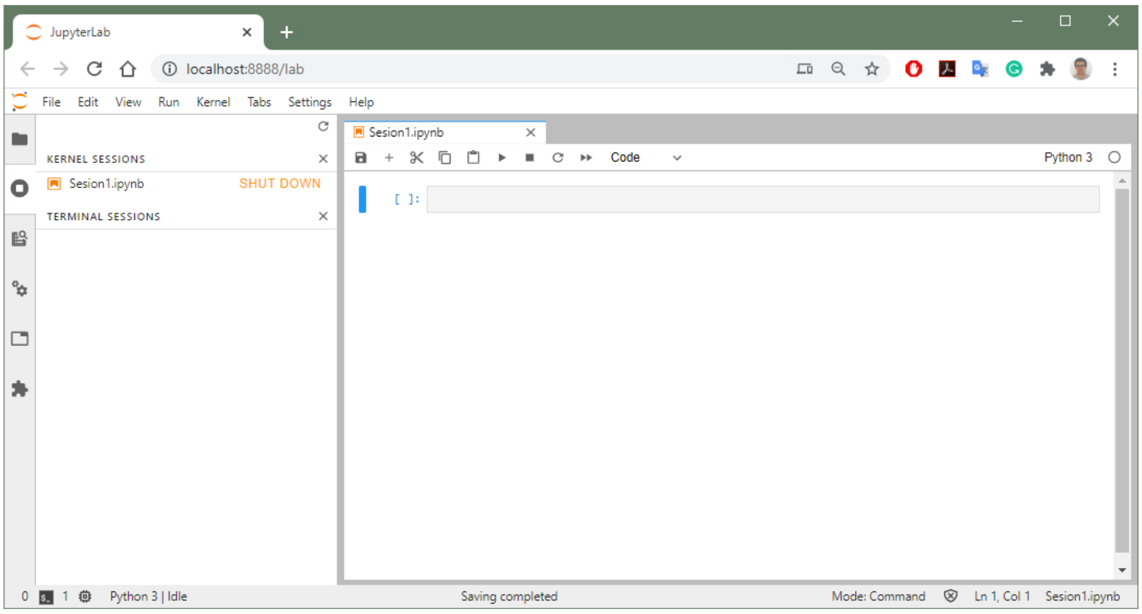
In the **launcher**, you can choose to create a new notebook, console, and other kinds of files



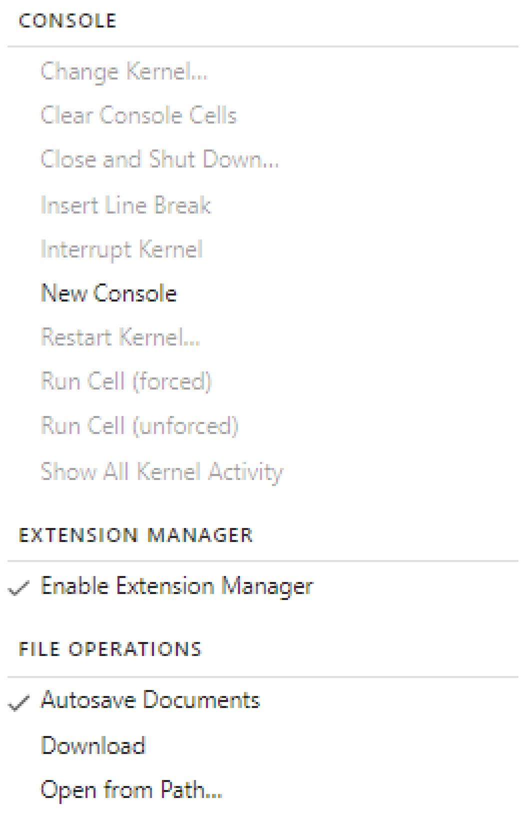
As we have only one programming language, start **creating** a new Python 3 Notebook. As a default, it has the name "Untitled," change it to "Session_1"



In the **Running Terminals and kernels tab**, you could see your **new notebook listed**, and the option to "Shut Down" if you click in Shut Down, the kernel becomes wholly closed, so you would have to rerun each cell of the notebook.

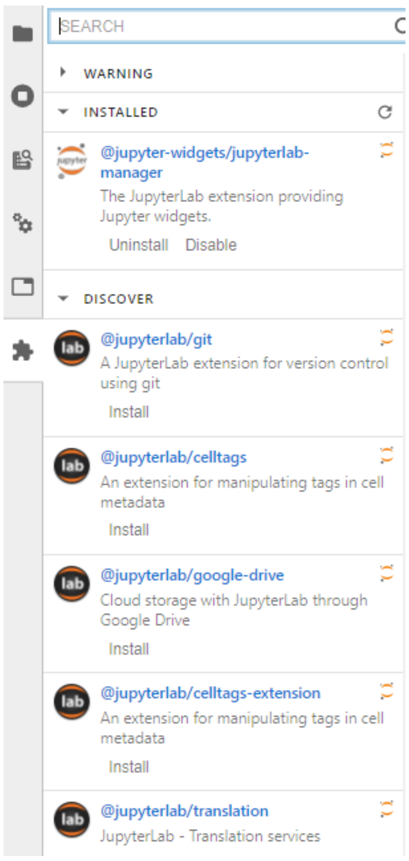


In the **Commands tab**, you can choose or change the different options that Jupyter Lab has.



In the installation guide, we went quickly through some features regarding Jupyter, like installing "ipywidgets."

Go to the Widgets tab, and you could see the one we have installed and a list of widgets we can install and add to our workflow.


SEARCH

WARNING

INSTALLED

@jupyter-widgets/jupyterlab-manager
The JupyterLab extension providing Jupyter widgets.
Uninstall Disable

DISCOVER

@jupyterlab/git
A JupyterLab extension for version control using git
Install

@jupyterlab/celltags
An extension for manipulating tags in cell metadata
Install

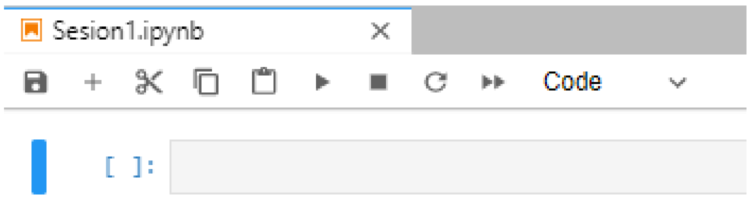
@jupyterlab/google-drive
Cloud storage with JupyterLab through Google Drive
Install

@jupyterlab/celltags-extension
An extension for manipulating tags in cell metadata
Install

@jupyterlab/translation
JupyterLab - Translation services

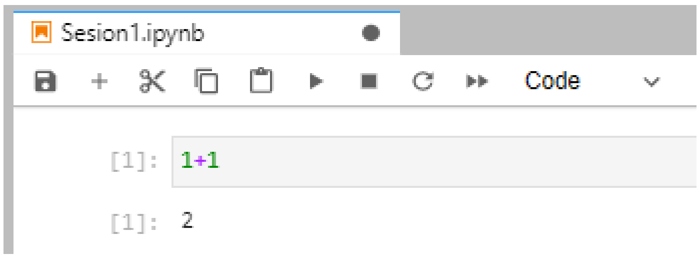
In our **notebook**, we can see the different options that the notebook has, which are from left to right:

- ✓ Save the notebook
- ✓ Insert a cell below
- ✓ Cut cell
- ✓ Copy cell
- ✓ Past cell
- ✓ Run the code of the cell and advance
- ✓ Stop the code of the selected cell
- ✓ Restart the kernel
- ✓ Restart the kernel and rerun the complete notebook.
- ✓ And the final allows us to change between Code and Markdown.



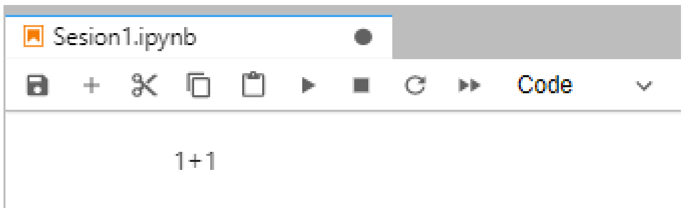
If we want to **write code**, you can type the following command and press "Shift+Enter" in your keyboard

```
1+1
```



```
[1]: 1+1
[1]: 2
```

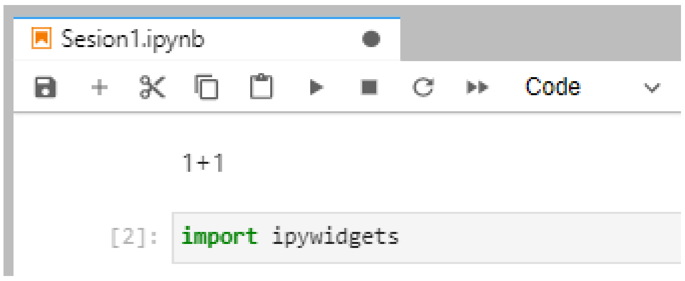
If we want to use it as a **Markdown text**, most commonly used for annotations, we can select the cell with the code and change it to Markdown.



1+1

Now let's play with the **Widgets**. The widgets are applied as a python function (we would go into detail of functions in the next sessions). To import the widgets to the notebook, we import them as importing a python library like Numpy.

```
import ipywidgets
```

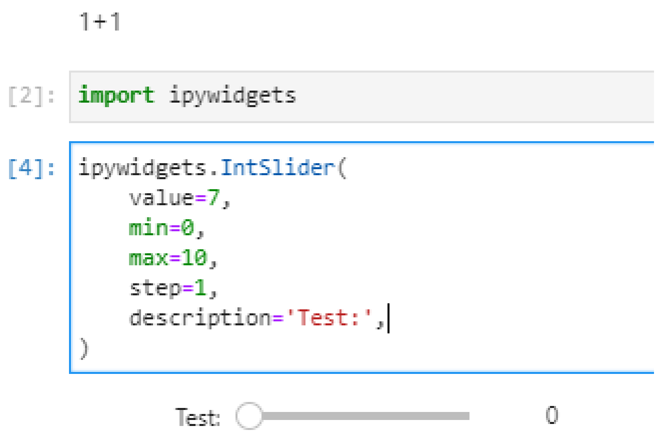


1+1

```
[2]: import ipywidgets
```

And to use a simple **slider widget**, we can run the following code, which returns an interactive slider.

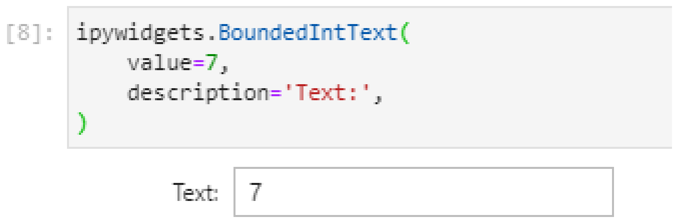
```
ipywidgets.IntSlider(
    value=7,
    min=0,
    max=10,
    step=1,
    description='Test:',
)
```



```
1+1

[2]: import ipywidgets

[4]: ipywidgets.IntSlider(
         value=7,
         min=0,
         max=10,
         step=1,
         description='Test:',
     )
```

You can **create a widget to input text**:

```
ipywidgets.BoundedIntText(
    value=7,
    description='Text:',
)
```



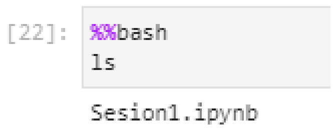
```
[8]: ipywidgets.BoundedIntText(
         value=7,
         description='Text:',
     )
```

In the cells, we can **run commands as we were in Ubuntu**. To do this, we have to start a line with the command:

```
%%bash
```

For example, if we want to see what is inside the current location

```
%%bash
ls
```

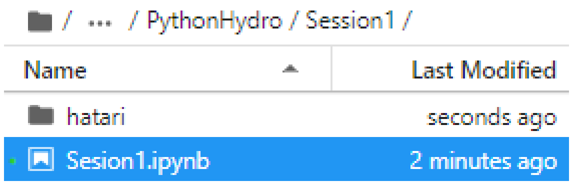


```
[22]: %%bash
      ls

      Sesion1.ipynb
```

You can **create a directory** too.

```
%%bash
mkdir hatari
```

If you **refresh** the File Browser, you would see the new folder created.



/ ... / PythonHydro / Session1 /

Name	Last Modified
hatari	seconds ago
Sesion1.ipynb	2 minutes ago

Voila

To this point, we have explored most of the functions related to Jupyter Lab, but it is growing fast.

As an example of the improvements made to Jupyter, the add-on Voila is to render the notebooks with its interactive widgets; this can be installed directly to our Conda distribution.

To install it, **open an Anaconda Prompt**, and run the following command:

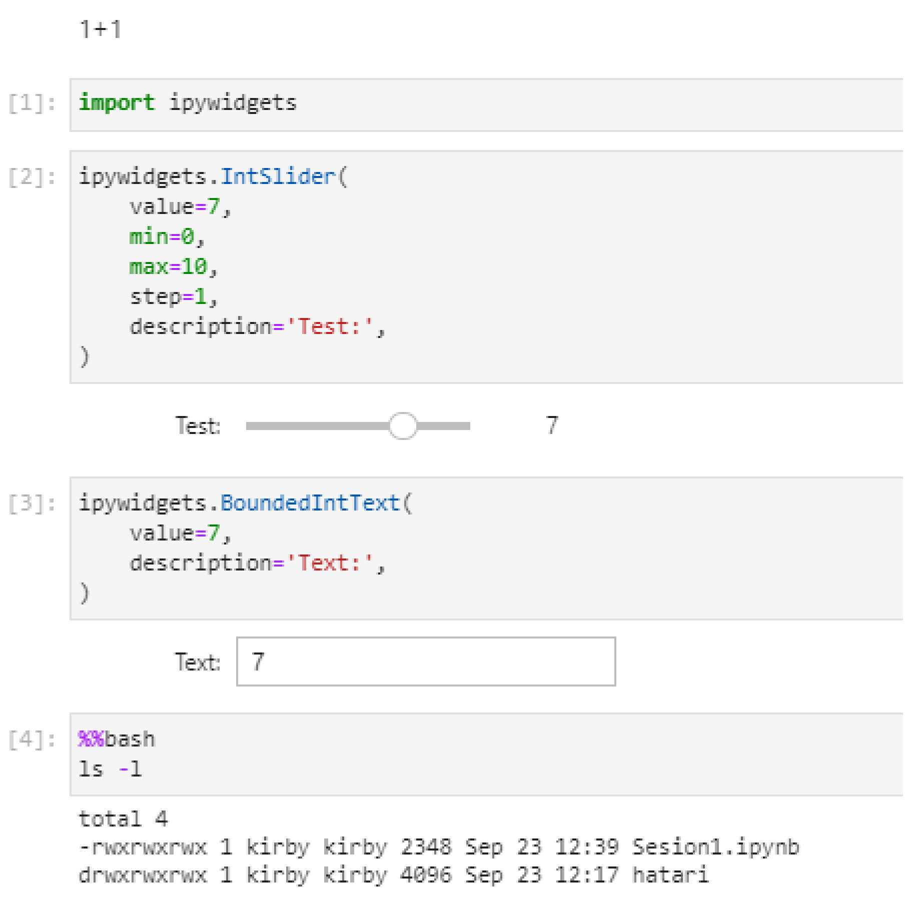
```
conda install -c conda-forge voila
```

You need to **add the extension to Jupyter lab** running the following command in the Anaconda Prompt

```
jupyter labextension install @jupyter-voila/jupyterlab-preview
```

Once the installation finishes, **close your Anaconda Prompt running Jupyter Lab and reopen it.**

Reopen your Sesion1 notebook, run all the cells again, and you could see a new option in the menu of the notebook

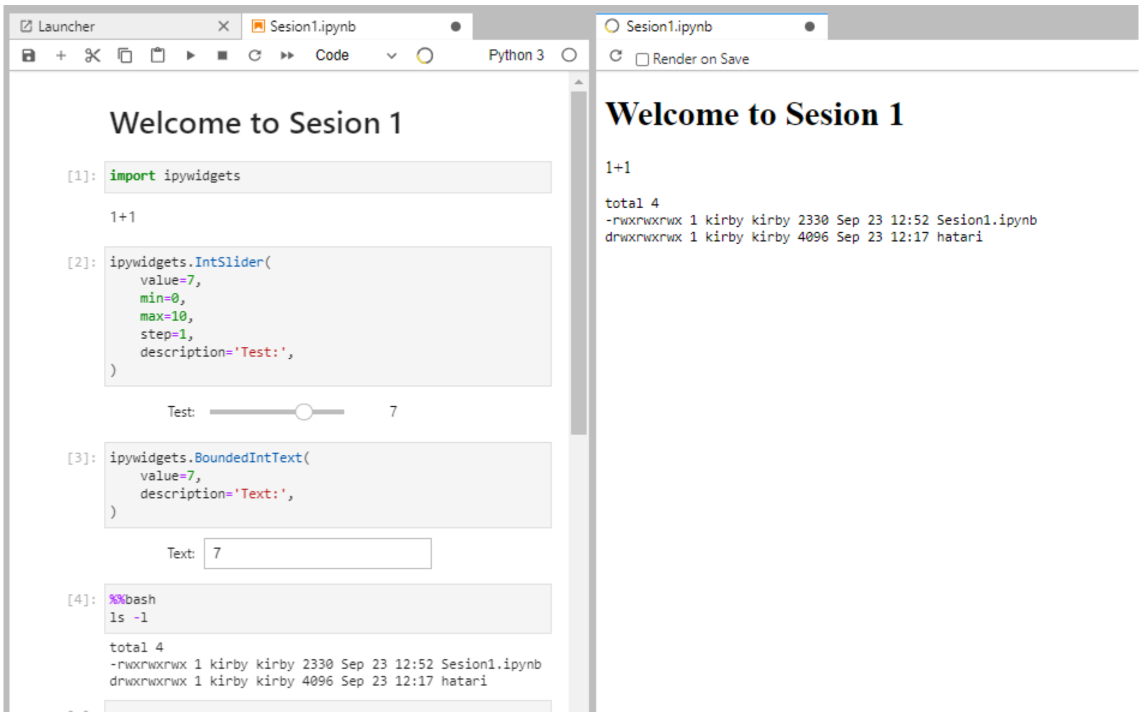


```
1+1

[1]: import ipywidgets

[2]: ipywidgets.IntSlider(
         value=7,
         min=0,
         max=10,
         step=1,
         description='Test:',
     )
```

```
[3]: ipywidgets.BoundedIntText(
         value=7,
         description='Text:',
     )
```

```
[4]: %%bash
     ls -l

total 4
-rwxrwxrwx 1 kirby kirby 2348 Sep 23 12:39 Sesion1.ipynb
drwxrwxrwx 1 kirby kirby 4096 Sep 23 12:17 hatari
```

Click on this new option called "**Render with Voila**". Another tab appears, rendering our notebook. As we have not assigned a plot with the widget, it is not rendered.



Welcome to Sesion 1

```
[1]: import ipywidgets
```

1+1

```
[2]: ipywidgets.IntSlider(
         value=7,
         min=0,
         max=10,
         step=1,
         description='Test:',
     )
```

Test: 7

```
[3]: ipywidgets.BoundedIntText(
         value=7,
         description='Text:',
     )
```

Text: 7

```
[4]: %%bash
     ls -l
```

```
total 4
-rwxrwxrwx 1 kirby kirby 2330 Sep 23 12:52 Sesion1.ipynb
drwxrwxrwx 1 kirby kirby 4096 Sep 23 12:17 hatari
```

Welcome to Sesion 1

1+1

```
total 4
-rwxrwxrwx 1 kirby kirby 2330 Sep 23 12:52 Sesion1.ipynb
drwxrwxrwx 1 kirby kirby 4096 Sep 23 12:17 hatari
```

Reference Links:

- https://www.anaconda.com/products/individual
- https://github.com/jupyterlab/jupyterlab
- https://github.com/voila-dashboards/voila