Créer des sites web Jupyter interactifs avec JupyterLite

AlpOSS 2024

Jérémy Tuloup

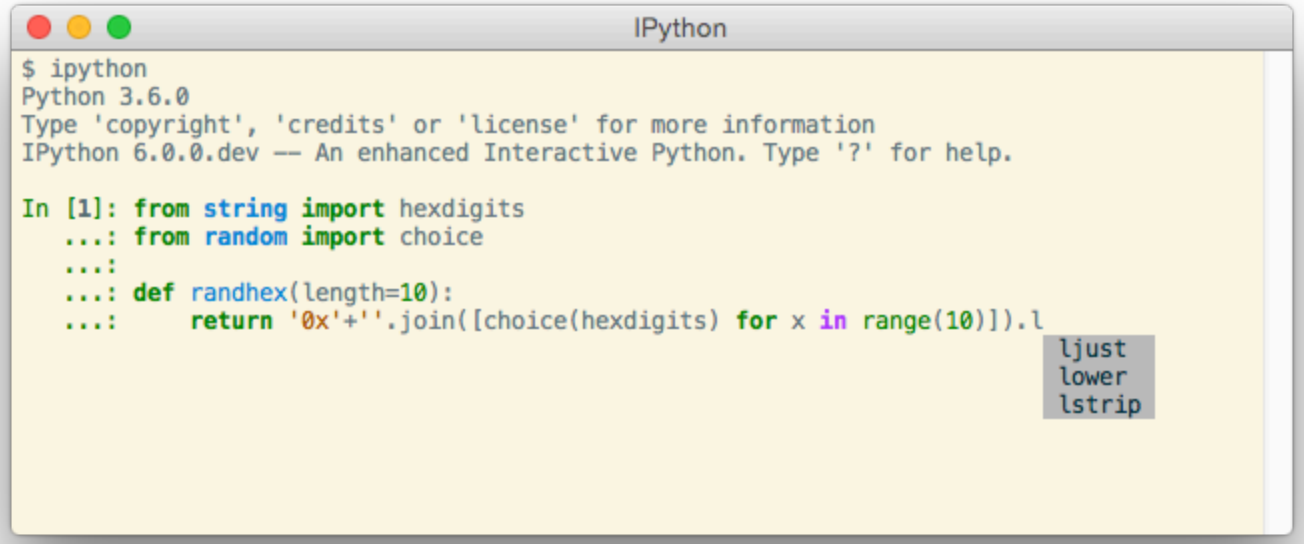
https://jtp.io/alposs-2024

Jérémy Tuloup

- Directeur Technique à QuantStack
- Jupyter Distinguished Contributor
- Jupyter Frontends SSC (Steering Software Council) representative
- Contributeur à de nombreux projets Jupyter
- Créateur de JupyterLite

Historique rapide de Jupyter

IPython dans le terminal



REPL: Read Eval Print Loop

Pourquoi les notebooks sont populaires ?

- Le workflow REPL
- Avec en plus:
  - **narration**
  - **mémoire**
  - **reproductibilité**
  - **communication**

IPython Notebook

IPython Notebook | Spectrogram | Save | Idle

Notebook

Actions: New | Open | Download | ipynb | Print

Cell

Actions: Delete

Format: Code | Markdown

Output: Toggle | ClearAll

Insert: Above | Below

Move: Up | Down

Run: Selected | All

Autoindent: ☑

Kernel

Actions: Interrupt | Restart

Kill kernel upon exit: ☐

Help

Links: Python | IPython | NumPy | SciPy | MPL | SymPy

Shift-Enter : run selected cell

Ctrl-Enter : run in terminal mode

Ctrl-m h : show keyboard shortcuts

Simple spectral analysis

An illustration of the Discrete Fourier Transform

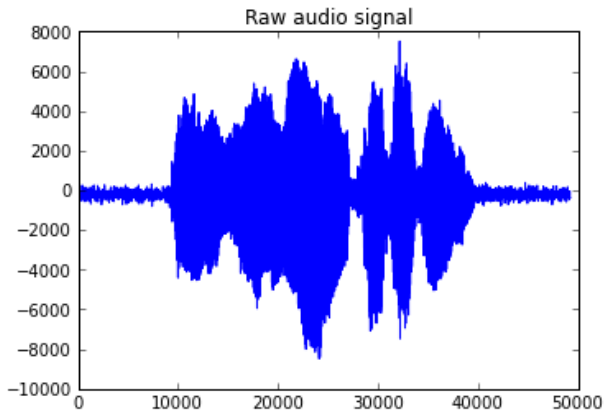
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

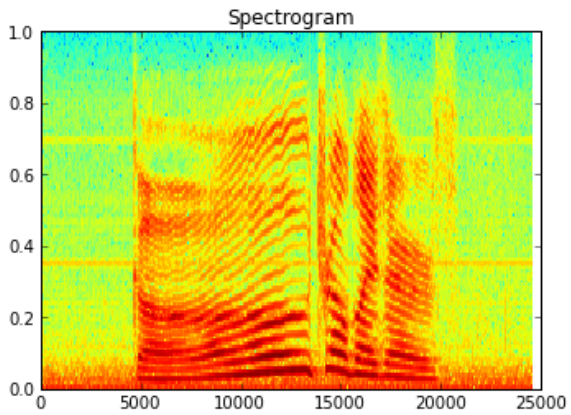
using windowing, to reveal the frequency content of a sound signal.
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('/home/fperez/teach/py4science/book/examples/test_mono.wav')
```

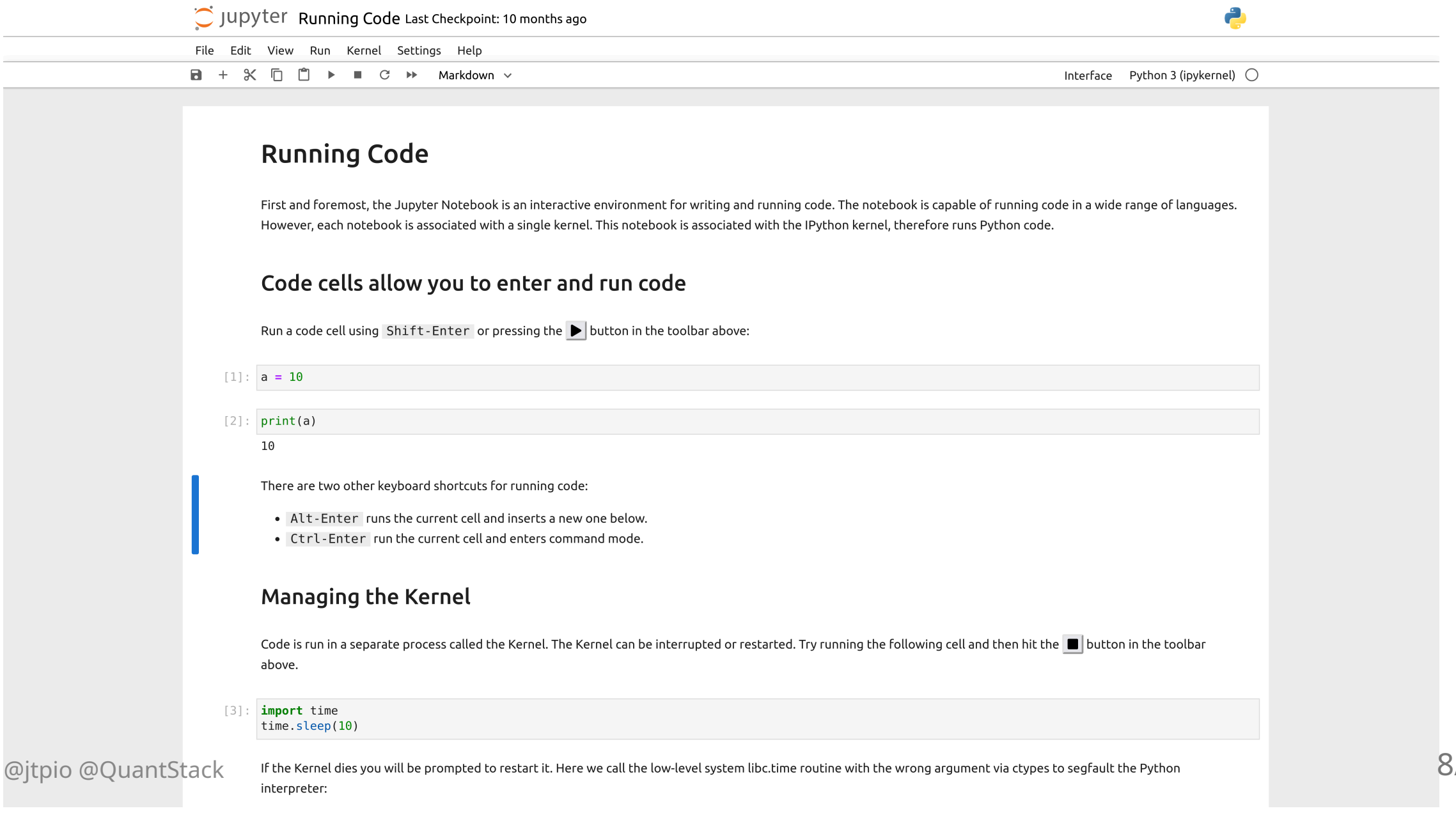
And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

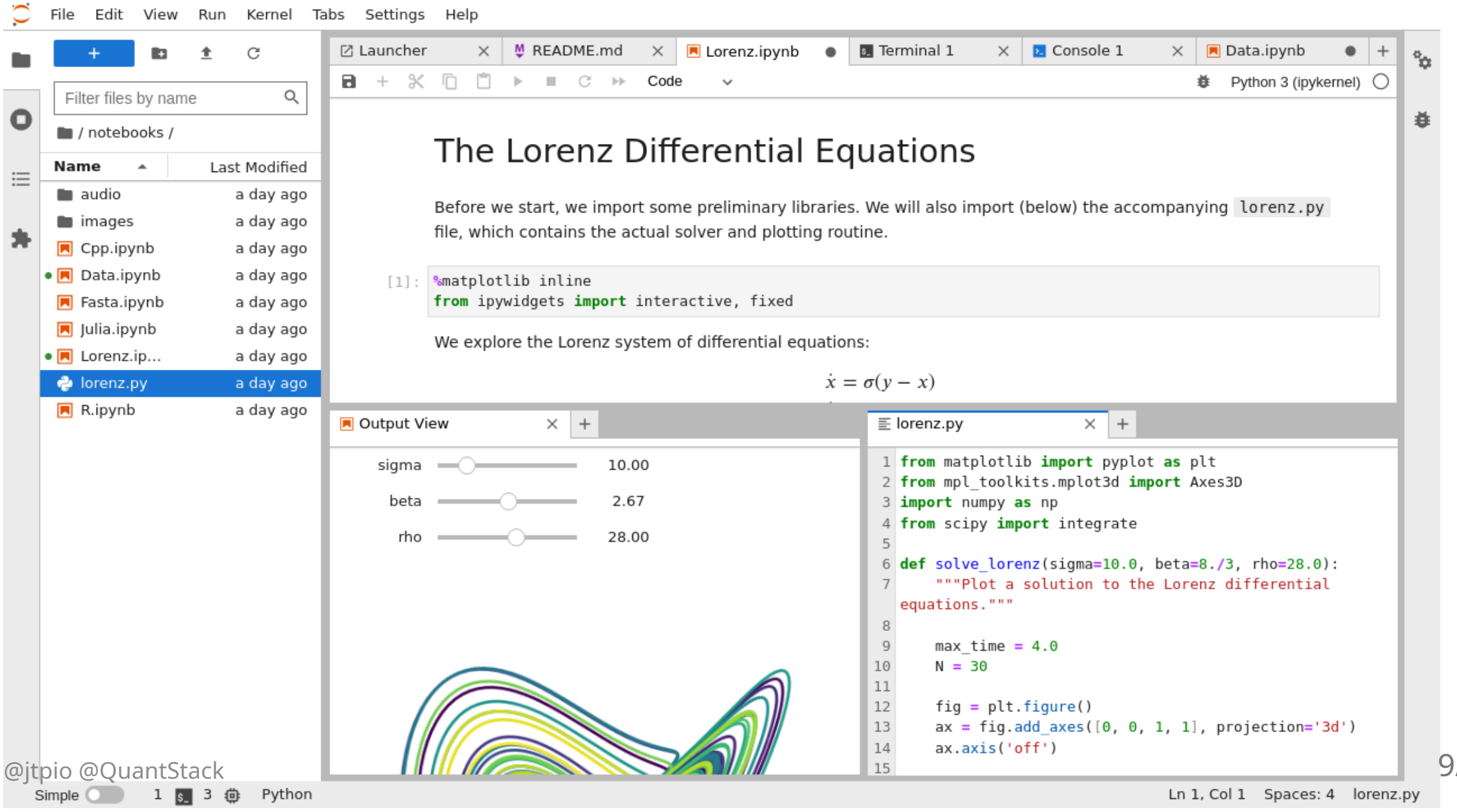




Jupyter Notebook

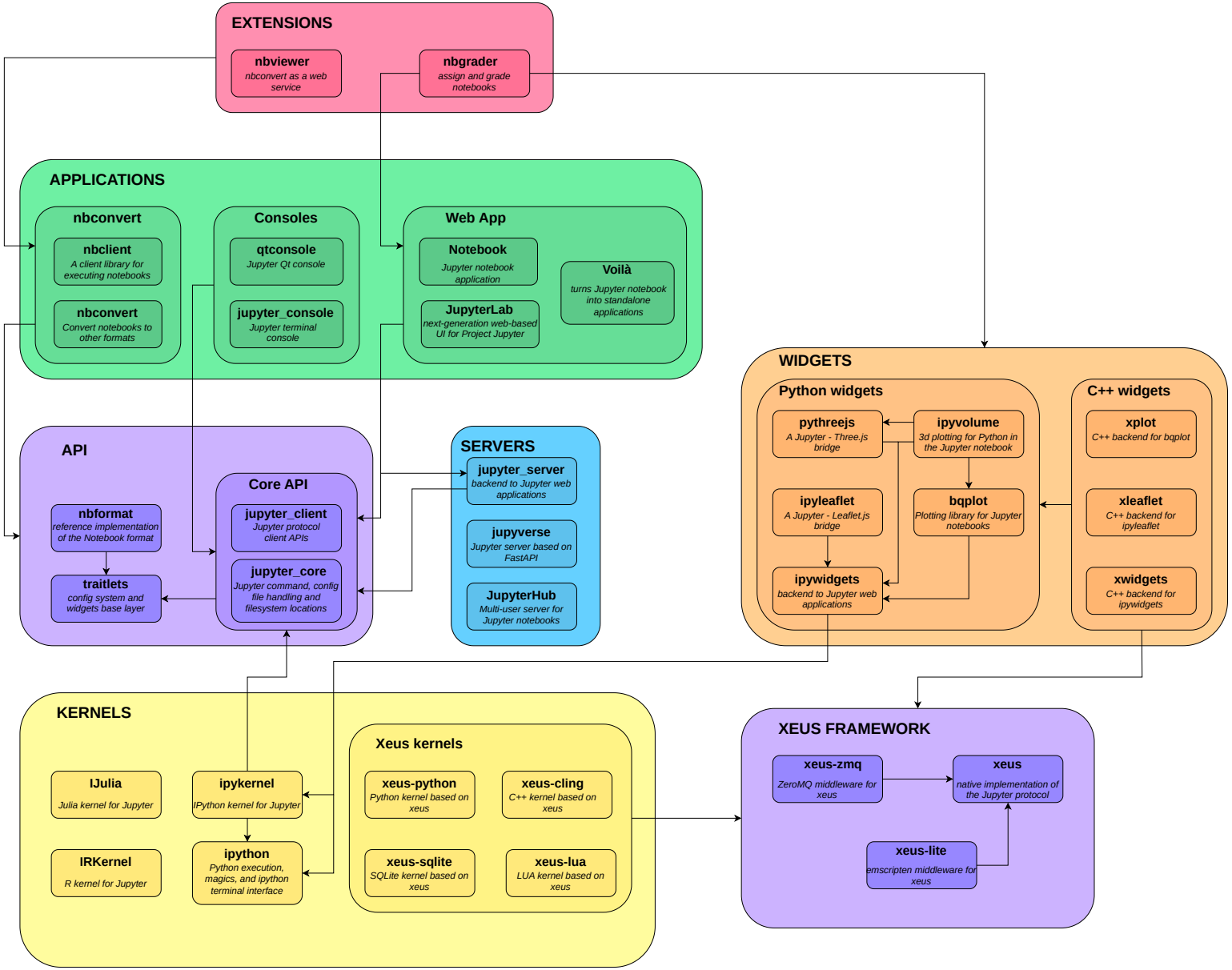


JupyterLab


File Edit View Run Kernel Tabs Settings Help
Launcher
README.md
Lorenz.ipynb
Terminal 1
Console 1
Data.ipynb
Code
Python 3 (ipykernel)
Filter files by name
/ notebooks /
Name
Last Modified
audio a day ago
images a day ago
Cpp.ipynb a day ago
Data.ipynb a day ago
Fasta.ipynb a day ago
Julia.ipynb a day ago
Lorenz.ip... a day ago
lorenz.py a day ago
R.ipynb a day ago
The Lorenz Differential Equations
Before we start, we import some preliminary libraries. We will also import (below) the accompanying lorenz.py file, which contains the actual solver and plotting routine.
[1]:
%matplotlib inline
from ipywidgets import interactive, fixed
We explore the Lorenz system of differential equations:
$$\dot{x} = \sigma(y - x)$$
Output View
sigma 10.00
beta 2.67
rho 28.00
lorenz.py
1 from matplotlib import pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 import numpy as np
4 from scipy import integrate
5
6 def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
7 """Plot a solution to the Lorenz differential equations."""
8
9 max_time = 4.0
10 N = 30
11
12 fig = plt.figure()
13 ax = fig.add_axes([0, 0, 1, 1], projection='3d')
14 ax.axis('off')
15
Simple 1 3 Python
Ln 1, Col 1 Spaces: 4 lorenz.py

L'écosystème Jupyter est très vaste

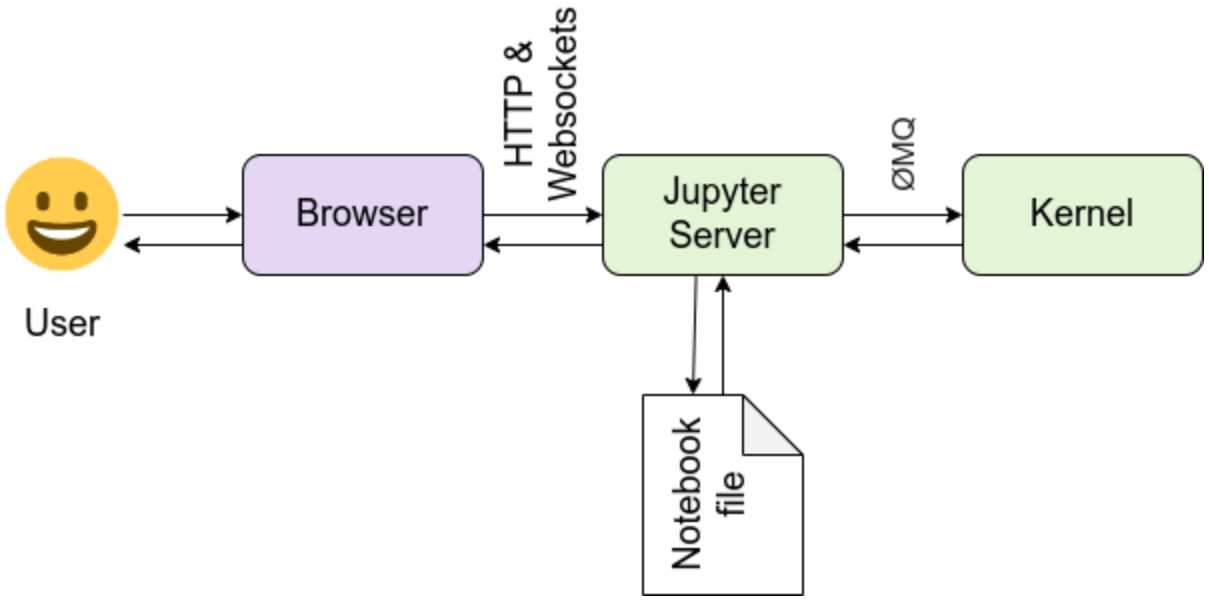
- Navigating the Jupyter Landscape
  - JupyterCon 2023 (Paris)
  - Jeremy Tuloup, Johan Mabille
  - https://www.youtube.com/watch?v=uWJ0-OPKTxI


EXTENSIONS
nbviewer
nbconvert as a web service
nbgrader
assign and grade notebooks
APPLICATIONS
nbconvert
nbclient
A client library for executing notebooks
nbconvert
Convert notebooks to other formats
Consoles
qtconsole
Jupyter Qt console
jupyter_console
Jupyter terminal console
Web App
Notebook
Jupyter notebook application
JupyterLab
next-generation web-based UI for Project Jupyter
Voilà
turns Jupyter notebook into standalone applications
WIDGETS
Python widgets
pythreejs
A Jupyter - Three.js bridge
ipyvolume
3d plotting for Python in the Jupyter notebook
ipyleaflet
A Jupyter - Leaflet.js bridge
bqplot
Plotting library for Jupyter notebooks
ipywidgets
backend to Jupyter web applications
C++ widgets
xplot
C++ backend for bqplot
xleaflet
C++ backend for ipyleaflet
xwidgets
C++ backend for ipywidgets
API
nbformat
reference implementation of the Notebook format
traitlets
config system and widgets base layer
Core API
jupyter_client
Jupyter protocol client APIs
jupyter_core
Jupyter command, config file handling and filesystem locations
SERVERS
jupyter_server
backend to Jupyter web applications
jupyverse
Jupyter server based on FastAPI
JupyterHub
Multi-user server for Jupyter notebooks
KERNELS
IJulia
Julia kernel for Jupyter
IRKernel
R kernel for Jupyter
ipykernel
IPython kernel for Jupyter
ipython
Python execution, magics, and ipython terminal interface
Xeus kernels
xeus-python
Python kernel based on xeus
xeus-cling
C++ kernel based on xeus
xeus-sqlite
SQLite kernel based on xeus
xeus-lua
LUA kernel based on xeus
XEUS FRAMEWORK
xeus-zmq
ZeroMQ middleware for xeus
xeus
native implementation of the Jupyter protocol
xeus-lite
emscripten middleware for xeus

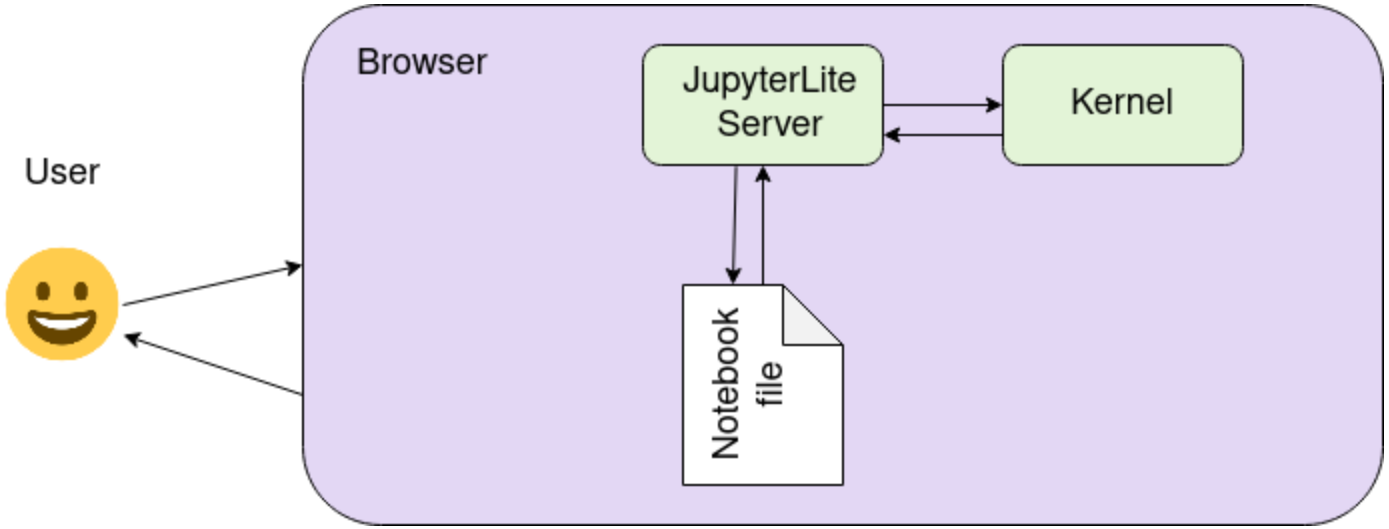
JupyterLite

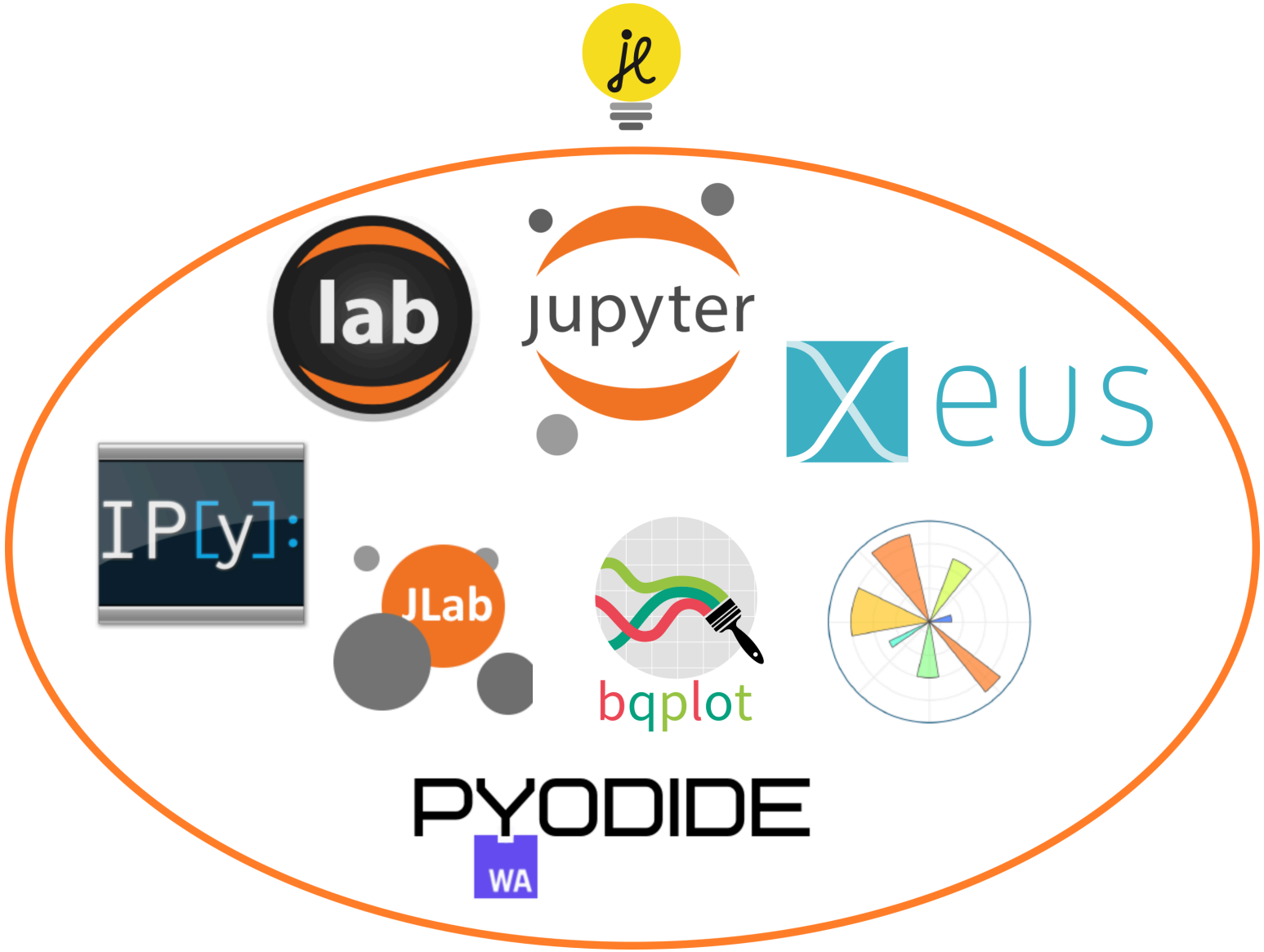
- Tout tourne dans le navigateur web via WebAssembly
- Se base sur la stack Jupyter existante:
  - Les noyaux Pyodide et Xeus exécutent le code dans le navigateur
  - Interfaces web JupyterLab et Jupyter Notebook
  - Voici pour faire des applications web et dashboards

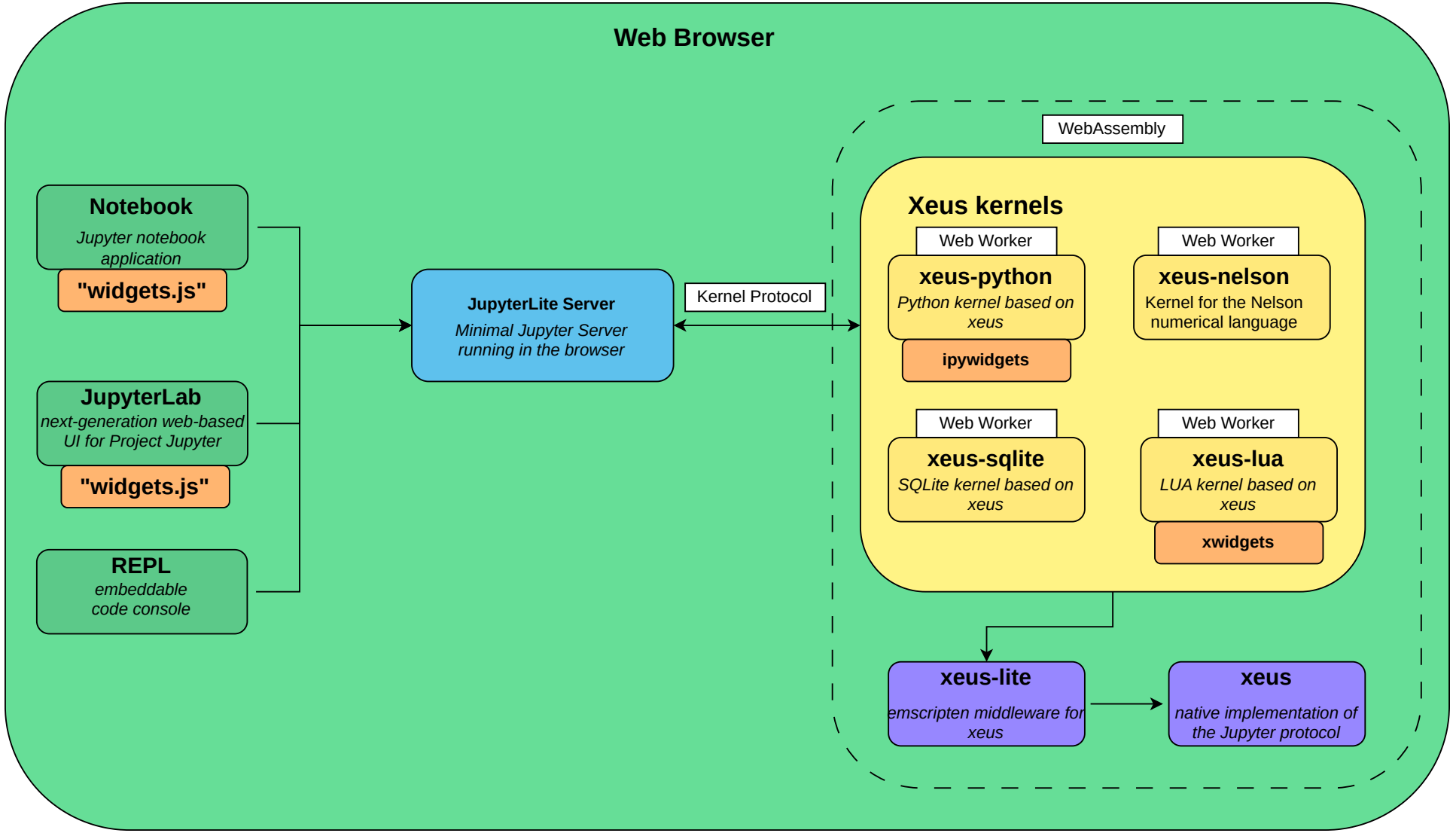
Jupyter


HTTP &
Websockets
ØMQ
Browser
Jupyter
Server
Kernel
User
Notebook
file

JupyterLite



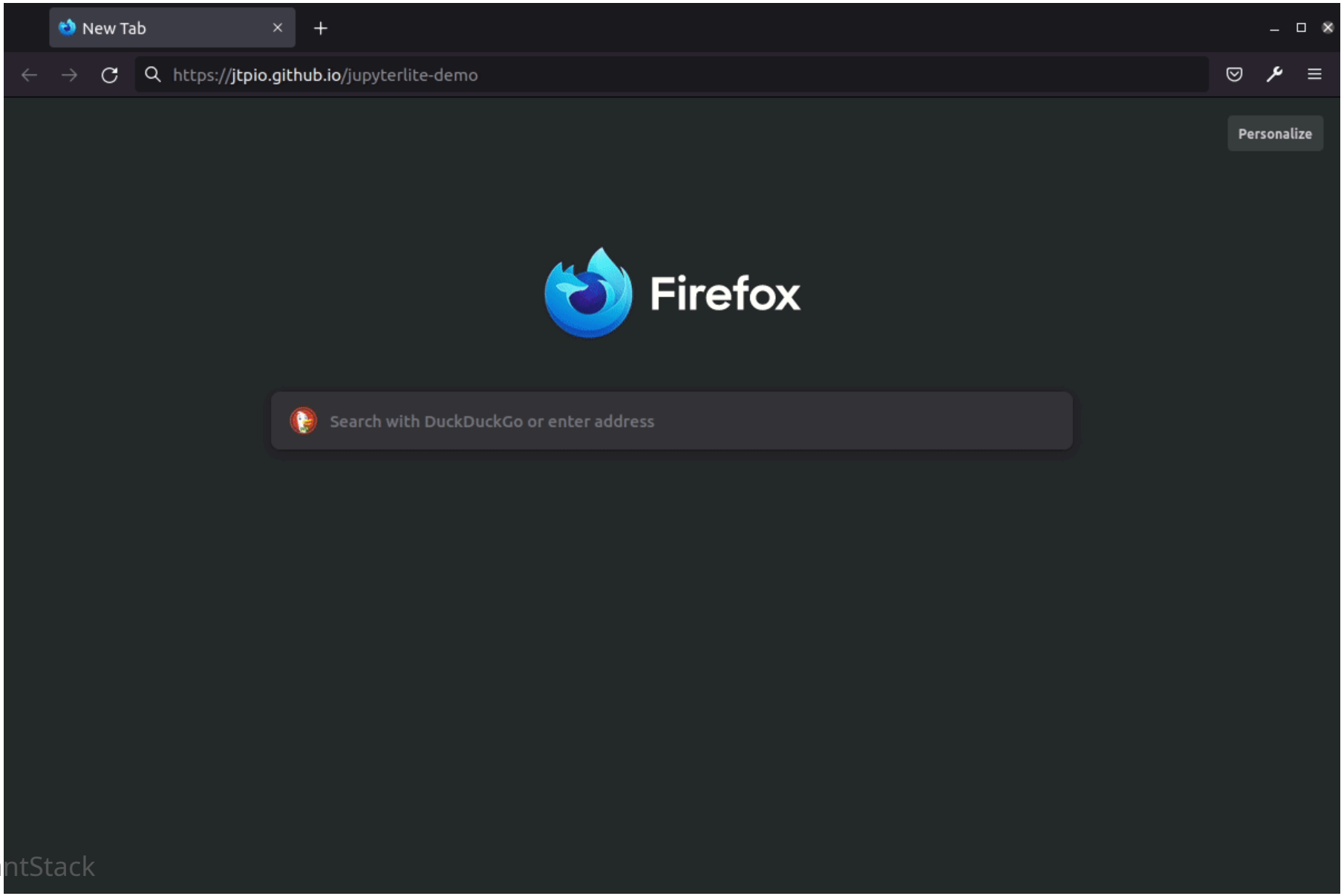

lab
jupyter
eus
IP[y]:
JLab
bqplot
PYODIDE
WA


Web Browser
Notebook
Jupyter notebook application
"widgets.js"
JupyterLab
next-generation web-based UI for Project Jupyter
"widgets.js"
REPL
embeddable code console
JupyterLite Server
Minimal Jupyter Server running in the browser
Kernel Protocol
WebAssembly
Xeus kernels
Web Worker
xeus-python
Python kernel based on xeus
ipywidgets
Web Worker
xeus-nelson
Kernel for the Nelson numerical language
Web Worker
xeus-sqlite
SQLite kernel based on xeus
Web Worker
xeus-lua
LUA kernel based on xeus
xwidgets
xeus-lite
emscripten middleware for xeus
xeus
native implementation of the Jupyter protocol

Jupyter et Python dans le navigateur

- ✅ pas de serveur Python
- ✅ pas de ligne de commande pour les utilisateurs
- ✅ pas besoin d'installer Python et autres paquets
- ✅ peut être hébergé comme site statique

Un site Jupyter accessible en quelques secondes



Générateur de site statique

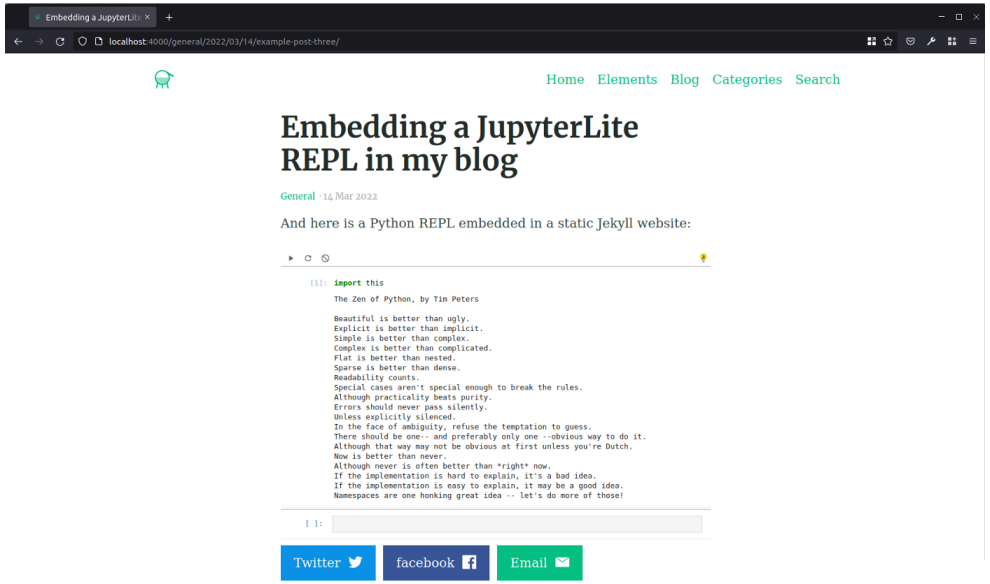
```
pip install jupyterlite-core

jupyter lite build
```

Fichiers HTML, CSS, JavaScript, Wasm

```
├── api
│   └── translations
│       ├── all.json
│       └── en.json
├── bootstrap.js
├── build
│   ├── 9507.1e6cc5d.js
│   ├── 9602.62bf0f1.js
│   ├── 9621.e2e8b5d.js
│   ├── ...
│   ├── repl
│   │   ├── bundle.js
│   ├── retro
│   │   ├── bundle.js
│   ├── schemas
│   │   ├── all.json
│   │   ├── @jupyterlab
│   │   │   ├── application-extension
│   │   │   │   ├── commands.json
│   │   │   │   ├── context-menu.json
│   │   │   │   ├── shell.json
│   │   │   │   └── sidebar.json
│   ├── themes
│   │   └── @jupyterlab
│   │       ├── theme-dark-extension
│   │       │   ├── index.css
│   │       │   └── index.js
...
```

Cas d'usage




```
<iframe
  src="https://jupyterlite.github.io/demo/repl/index.html?kernel=python&toolbar=1"
  width="100%"
  height="500px"
>
</iframe>
```

NumPy

https://numpy.org

Try NumPy

Use the interactive shell to try NumPy in the browser

```
"""
To try the examples in the browser:
1. Type code in the input cell and press
   Shift + Enter to execute
2. Or copy paste the code, and click on
   the "Run" button in the toolbar
"""

# The standard way to import NumPy:
import numpy as np

# Create a 2-D array, set every second element in
# some rows and find max per row:

x = np.arange(15, dtype=np.int64).reshape(3, 5)
x[1:, ::2] = -99
x
# array([[  0,   1,   2,   3,   4],
#        [-99,   6, -99,   8, -99],
#        [-99,  11, -99,  13, -99]])

x.max(axis=1)
# array([ 4,  8, 13])

# Generate normally distributed random numbers:
rng = np.random.default_rng()
samples = rng.normal(size=2500)
samples
```

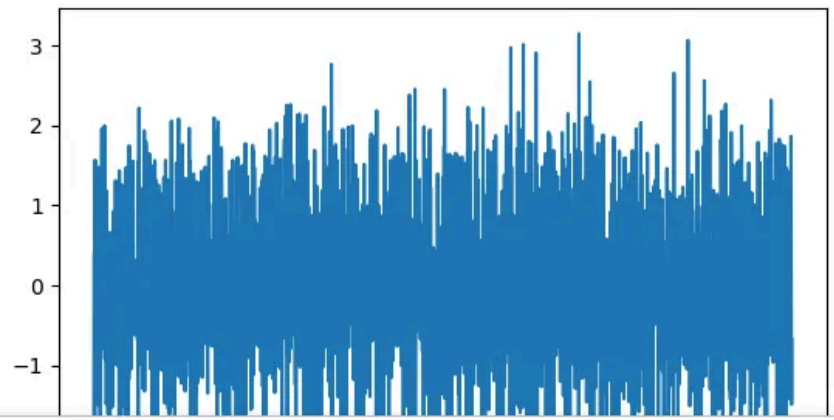
```
[4]: array([-2.37210593, -0.28975014,  0.40946174, ...,  0.56293026,
            -1.49251505, -0.67961604])

[5]: import matplotlib.pyplot as plt

[6]: plt.plot(samples)

[6]: [<matplotlib.lines.Line2D at 0x2e917d0>]

[7]: plt.show()
```




```
[ ]:
```

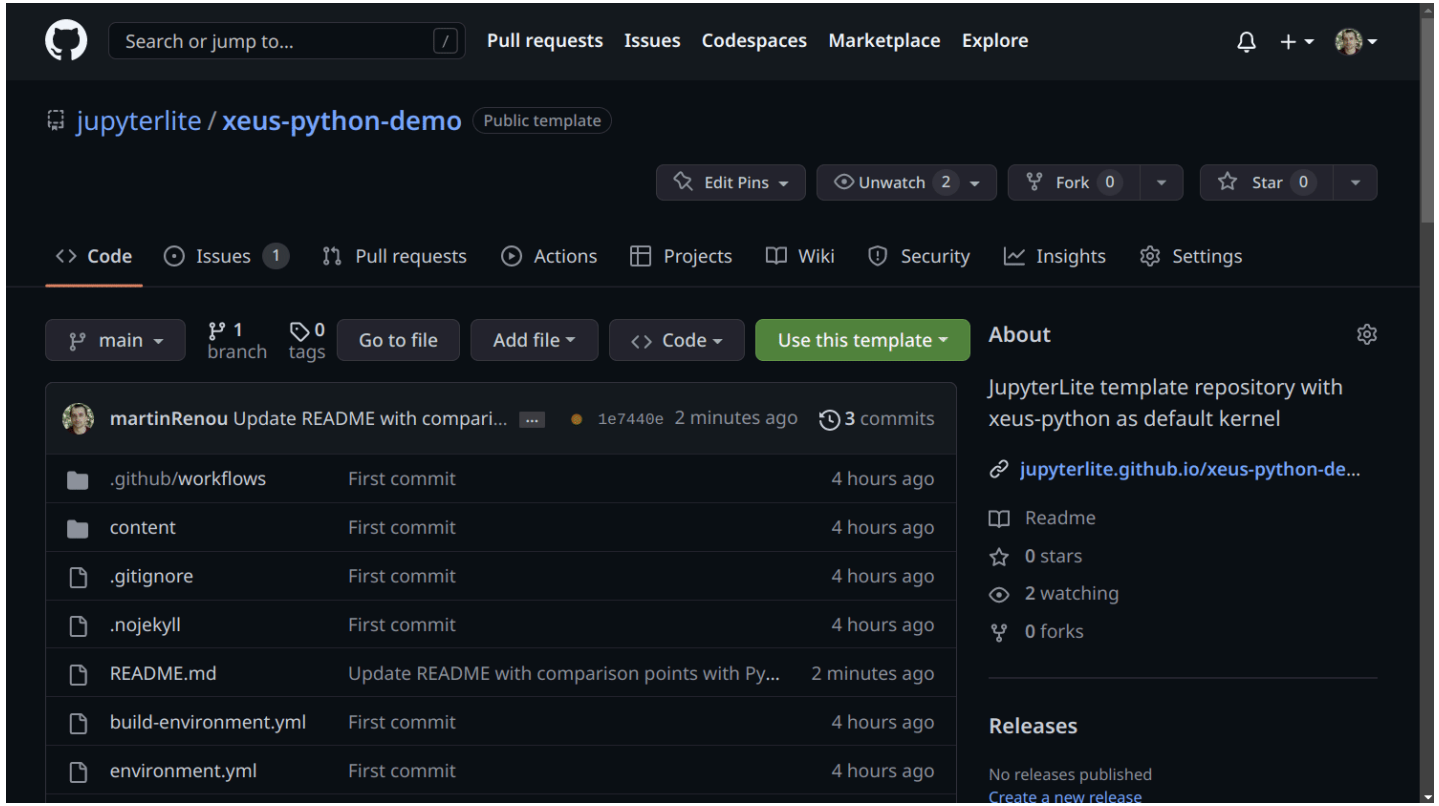
<video controls width="100%" height="600px"
src="https://github.com/jtpio/alposs-2024/assets/591645/9b5e247d-cb89-4b15-aba3-97e5ab381bb5">
</video>

Education

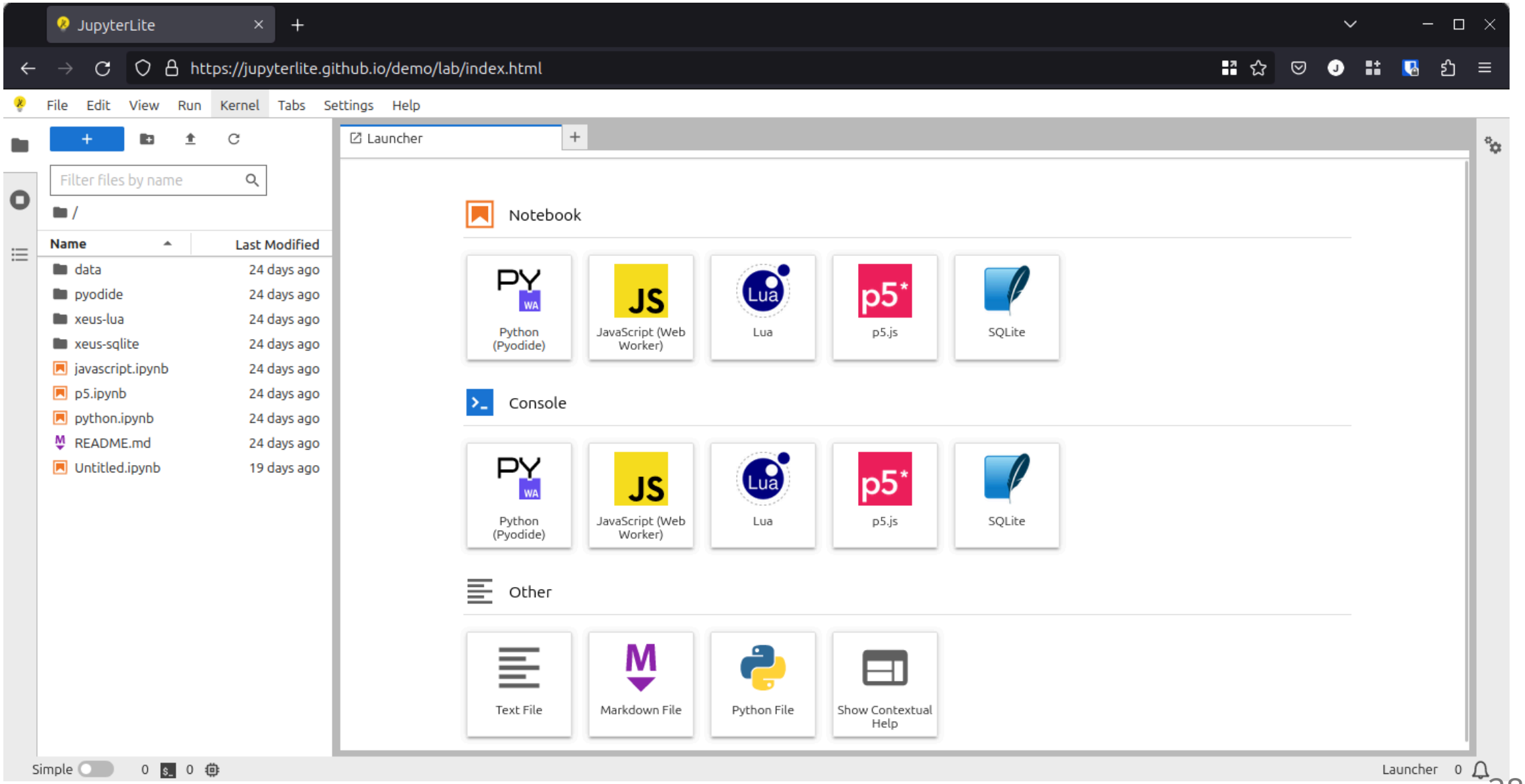
- Capytale: https://capytale.fr
- Paris Saclay: https://jupyter.gitlab.dsi.universite-paris-saclay.fr/tutoriel-jupyter/utiliser.html
- UC Berkeley

🚀 Deployez sur GitHub Pages

- https://github.com/jupyterlite/demo



https://jupyterlite.github.io/demo/



🔍 Références

- Documentation Jupyter: https://docs.jupyter.org
- Cette présentation:
  - Repo: https://github.com/jtpio/alposs-2024
  - Slides: https://jtp.io/alposs-2024

Merci !