Jupyter

Salvo Nicotra

What is Jupyter ?

> Free software, open standards, and web services for interactive computing across all programming languages

https://jupyter.org/

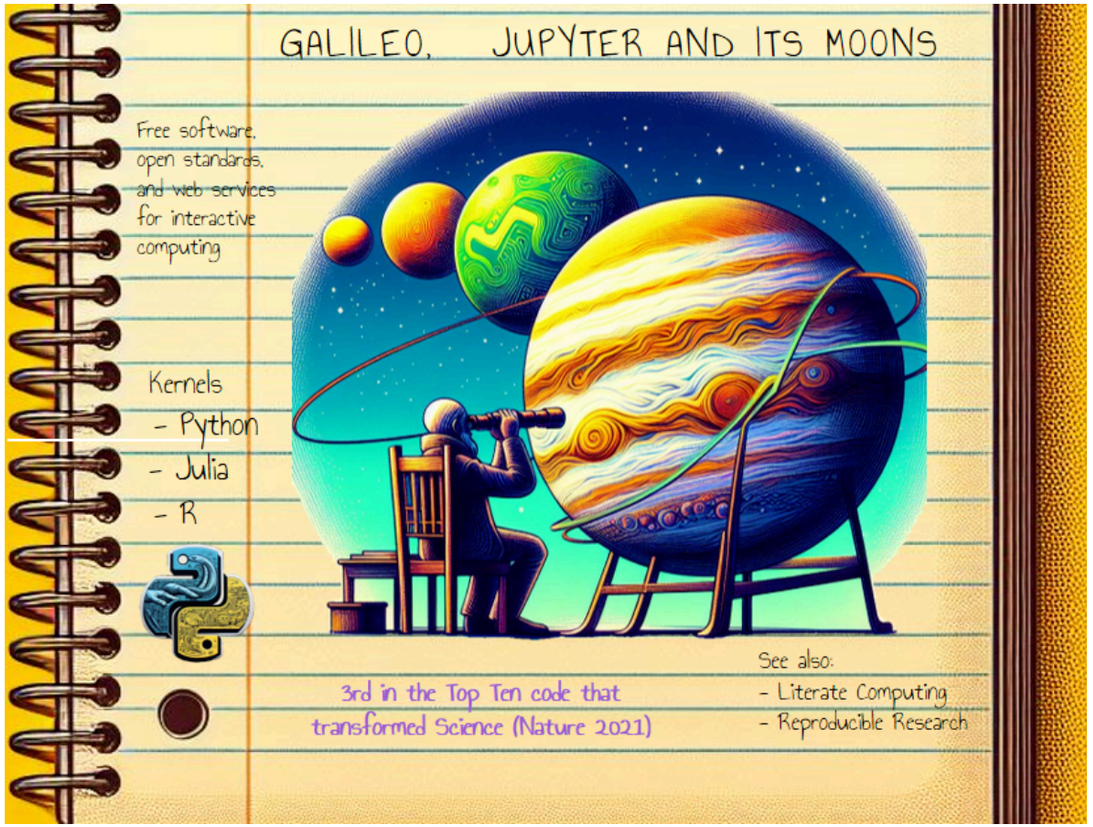
History

Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use

The **name** is refers to first three programming language supported (Julia, Python, R)

The **logo** is the Jupiter planet with its main moons representing the 3 language

It’s a tribute to Galileo



Impacts

In 2021, Nature added iPy as one the “Ten computer code that transfored science”
https://www.nature.com/articles/d41586-021-00075-2

Data explorer: IPython Notebook (2011)

Fernando Pérez was a graduate student “in search of procrastination” in 2001 when he decided to take on a core component of Python.

Python is an interpreted language, which means programs are executed line by line. Programmers can use a kind of computational call-and-response tool called a read–evaluate–print loop (REPL), in which they type code and a program called an interpreter executes it. A REPL allows for quick exploration and iteration, but Pérez noted that Python’s wasn’t built for science. It didn’t allow users to easily preload modules of code, for instance, or keep data visualizations open. So Pérez wrote his own version.

The result was IPython, an ‘interactive’ Python interpreter that Pérez unveiled in December 2001. A decade later, physicist Brian Granger, working with Pérez and others, migrated that tool to the web browser, launching the IPython Notebook and kick-starting a data-science revolution.

Like other computational notebooks, IPython Notebook combined code, results, graphics and text in a single document. But unlike other such projects, IPython Notebook was open-source, inviting contributions from a vast developer community. And it supported Python, a popular language for scientists. In 2014, IPython evolved into Project Jupyter, supporting some 100 languages and allowing users to explore data on remote supercomputers as easily as on their own laptops.

“For data scientists, Jupyter has emerged as a de facto standard,” *Nature* wrote in 2018 (ref. 9). At the time, there were 2.5 million Jupyter notebooks on the GitHub code-sharing

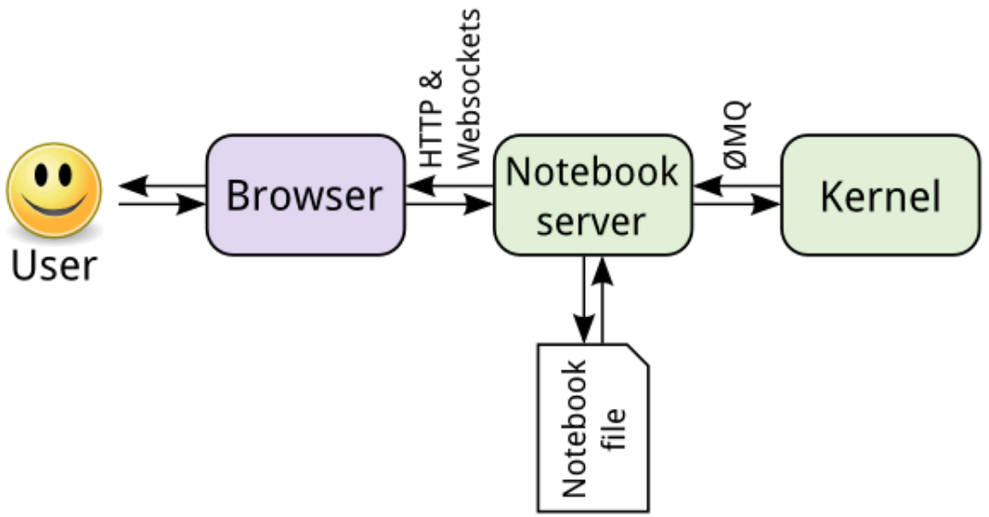
Juypter Notebooks

https://jupyter-notebook.readthedocs.io/

Components of Jupyter Notebooks

- **Notebook web application**: An interactive web application for writing and running code interactively and authoring notebook documents.
- **Kernels**: Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application.
- **Notebook documents**: Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects.

Component diagram



The notebook web application

Dashboard The Dashboard it’s the initial page and lists the files availabile in the directory and the running notebook.

Please note that the scope depends on the path where the application is lanched.

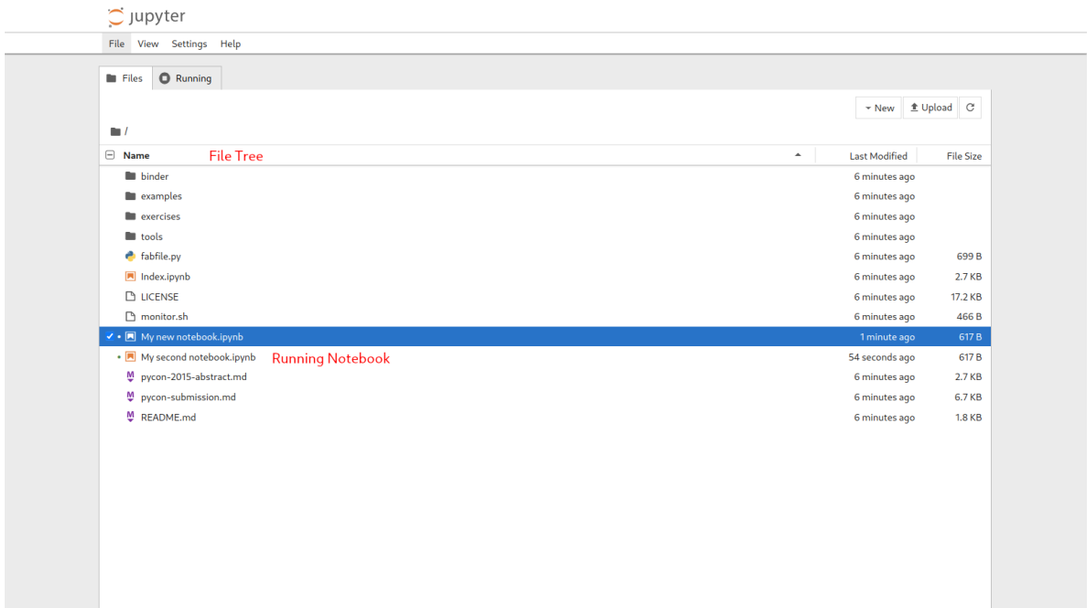


Image of the Notebook Dashboard from the Jupyter manual

Notebook Editor

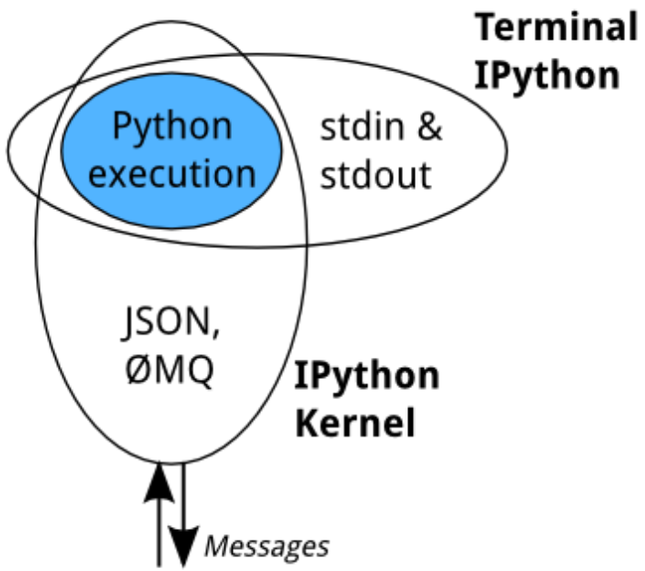
The editor of a notebook it’s “word” style with a menu on top

Main features

- Edit code in the browser, with automatic syntax highlighting, indentation, and tab completion/introspection.
- Run code from the browser, with the results of computations attached to the code which generated them.
- See the results of computations with rich media representations, such as HTML, LaTeX, PNG, SVG, PDF, etc.
- Create and use interactive JavaScript widgets, which bind interactive user interface controls and visualizations to reactive kernel side computations.
- Author narrative text using the Markdown markup language.
- Include mathematical equations using LaTeX syntax in Markdown, which are rendered in-browser by MathJax.

Kernels

- Through Jupyter's kernel and messaging architecture, the Notebook allows code to be run in a range of different programming languages.
- For each notebook document that a user opens, the web application starts a kernel that runs the code for that notebook.
- Each kernel is capable of running code in a single programming language and there are kernels available in the many languages https://github.com/jupyter/jupyter/wiki/Jupyter-kernels



Notebook documents

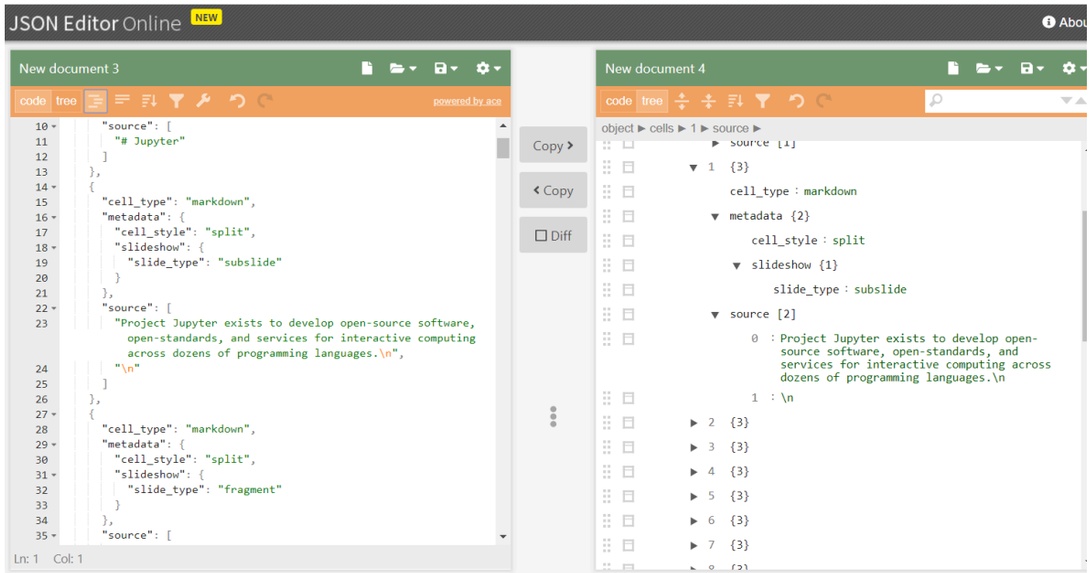
Notebook documents contain the inputs and outputs of an interactive session as well as narrative text that accompanies the code but is not meant for execution. Rich output generated by running code, including HTML, images, video, and plots, is embeddeed in the notebook, which makes it a complete and self-contained record of a computation.

When you run the notebook web application on your computer, notebook documents are just files on your local filesystem with a `.ipynb` extension. This allows you to use familiar workflows for organizing your notebooks into folders and sharing them with others.

Structure of a notebook document

Internally, notebook documents are

- **JSON** https://en.wikipedia.org/wiki/JSON data
- with binary values **base64** https://en.wikipedia.org/wiki/Base64 encoded.
- This allows them to be read and manipulated programmatically by any programming language.
- Because JSON is a text format, notebook documents are version control friendly.



Cells

Notebooks consist of a linear sequence of cells. There are three basic cell types:

- **Code cells**: Input and output of live code that is run in the kernel
- **Markdown cells**: Narrative text with embedded LaTeX equations
- **Raw cells**: Unformatted text that is included, without modification, when notebooks are converted to different formats using nbconvert

Markdown Cell

> Markdown is a plain text format for writing structured documents, based on conventions for indicating formatting in email and usenet posts. It was developed by John Gruber (with help from Aaron Swartz) and released in 2004

As Gruber writes: > The overriding design goal for Markdown's formatting syntax is to make it as readable as possible. The idea is that a Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.
(http://daringfireball.net/projects/markdown/)

Markdown References

- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- https://www.markdownguide.org/

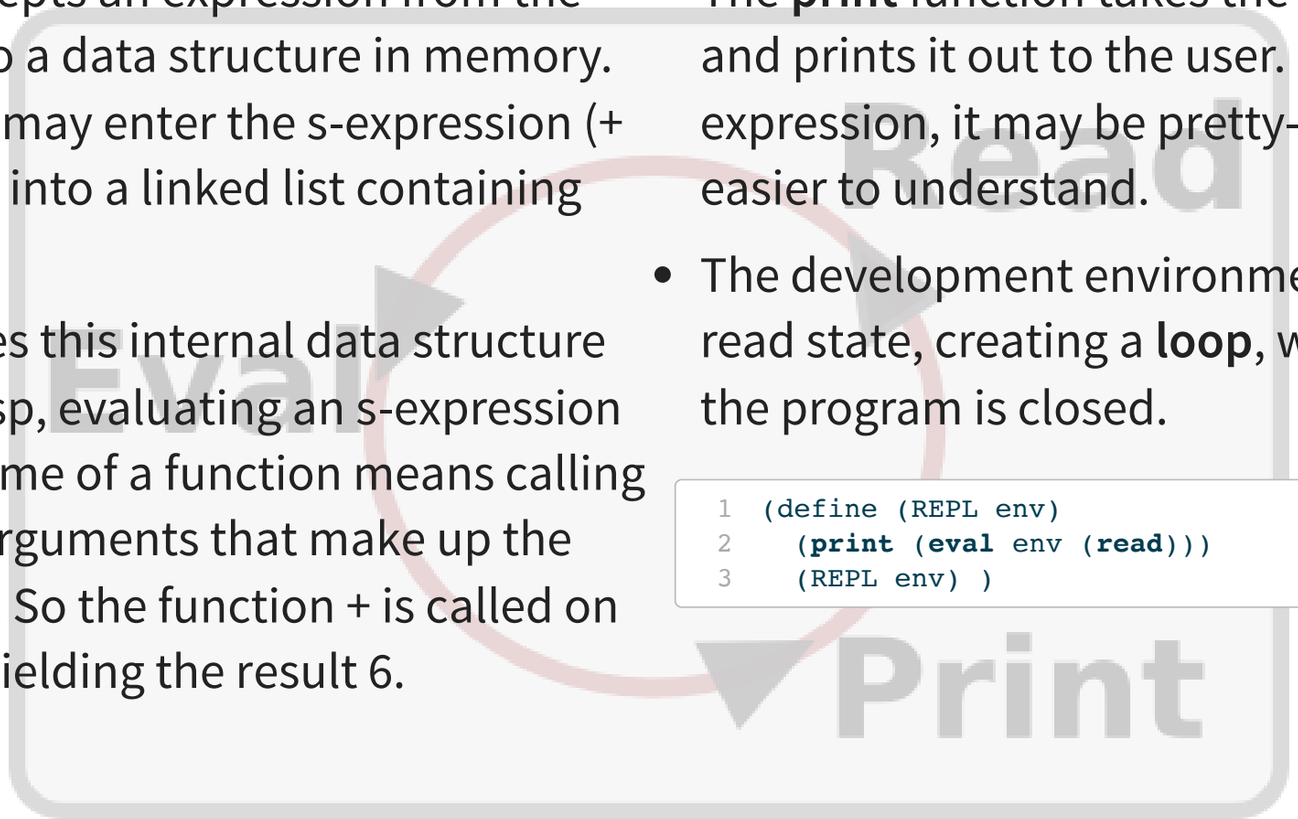
Code Cell

- Cell where it’s possibile to insert code associated to kernel of the notebook (i.e. Python)
- Code cell interact with the kernel using a protocol, i.e. the code is executed and output (if any) is printed after the cell
- Cell code are executed in order, i.e. no other cell code is effectively executed until the previosly completed

REPL

- The **read** function accepts an expression from the user, and parses it into a data structure in memory. For instance, the user may enter the s-expression (+ 1 2 3), which is parsed into a linked list containing four data elements.
- The **eval** function takes this internal data structure and evaluates it. In Lisp, evaluating an s-expression beginning with the name of a function means calling that function on the arguments that make up the rest of the expression. So the function + is called on the arguments 1 2 3, yielding the result 6.
- The **print** function takes the result yielded by eval, and prints it out to the user. If it is a complex expression, it may be pretty-printed to make it easier to understand.
- The development environment then returns to the read state, creating a **loop**, which terminates when the program is closed.

```
1  (define (REPL env)
2    (print (eval env (read)))
3    (REPL env) )
```



Edit / Command Mode

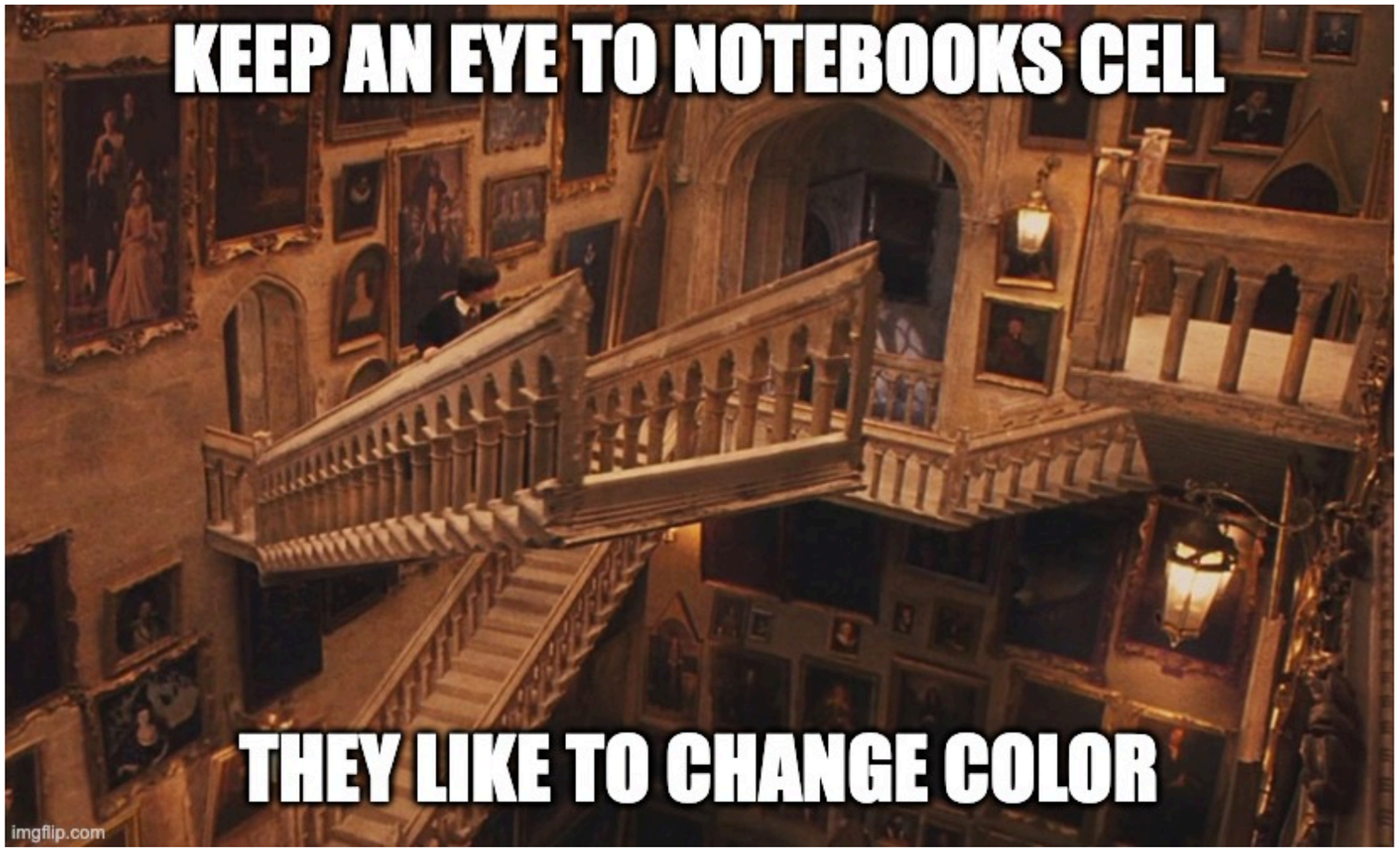
Notebooks have two modes:

Edit Mode (green) - it's possibile to write in the cell - code is not immediatetely executed

Some ide features (but it's not an IDE) - syntax highlighting - code completion

Command Mode: (blue): it's a "navigation mode", special keys to author the notebook

- Basic navigation: enter, shift-enter, up/k, down/j
- Saving the notebook: s
- Change Cell types: y, m, 1-6, t
- Cell creation: a, b
- Cell editing: x, c, v, d, z
- Kernel operations: i, 0 (press twice)


KEEP AN EYE TO NOTEBOOKS CELL
THEY LIKE TO CHANGE COLOR
imgflip.com

NicsMeme

Cell execution

- Cells can be executed clicking on the “Play” icon in the toolbar, via Cell -> Run in the menu or Shift-Enter
- Cells are executed in order, output appears asynchronously
- Markdown cells are immediately rendered
- Code cells sends the entire cell to the kerne, a [] box appers at the left with a * (waiting) and when completed the execution order

Rendering Notebooks

Nbviewer

https://nbviewer.org/

Any notebook document available from a public URL or on GitHub can be shared via nbviewer. This service loads the notebook document from the URL and renders it as a static web page. The resulting web page may thus be shared with others without their needing to install the Jupyter Notebook.

Examples - See this notebook using nbviewer

- Github automatically renders notebook See this notebook in GitHub

Export Notebook (NB Convert)

Using nbconvert enables: presentation of information in familiar formats, such as PDF. publishing of research using LaTeX and opens the door for embedding notebooks in papers. collaboration with others who may not use the notebook in their work. sharing contents with many people via the web using HTML. Format: - latex - pdf - html - slides

https://nbconvert.readthedocs.io

Slides

- There are different ways to create presentation of a notebook
- Each cell is “labeled” as Slide, Subslide or Fragment
- Typically they are based on reveal
- Rise: it’s an extension able to transform notebook in presentation inside the web application. The great advantage is that it’s possibile to execute cell “live”
- Voila: Voilà allows you to convert a Jupyter Notebook into an interactive dashboard that allows you to share your work with others.
- Quarto: Quarto can render Jupyter notebooks represented as plain text (.qmd) or as a normal notebook file (.ipynb). One benefit of using .ipynb is that you can use JupyterLab as your editor.

Jupyter Book

What is Jupyter Book

https://jupyterbook.org/en/stable/intro.html - Jupyter Book is an open source project for building beautiful, publication-quality books and documents from computational material. - Define Table of Content and Meta Data - Uses MyST Markdown: MyST makes Markdown more *extensible* & **powerful** to support an ecosystem of tools for computational narratives, technical documentation, and open scientific communication.

Criticism to the Notebook

I don’t like notebooks

https://docs.google.com/presentation/d/1n2RlMdmv1p25Xy5thJUhkKGvjtV-dkAIsUXP-AL4ffI/edit#slide=id.g362da58057_0_1