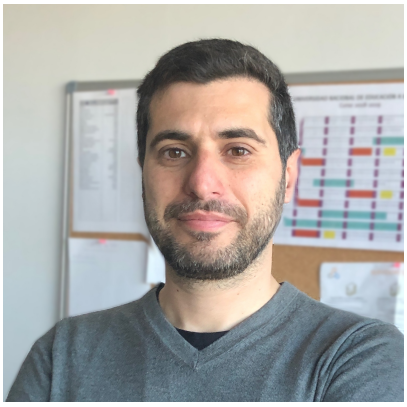
DESIR Winter School: Shaping new approaches to data management in arts and humanities

Open Research Notebooks

10-13 Dec 2019 Lisbon (Portugal)

About me



Javier de la Rosa, *versae@gmail.com* (mailto:versae@gmail.com), *@versae* (https://twitter.com/versae)

Postdoctoral Researcher in NLP at UNED (ERC POSTDATA Project), Spain
PhD in Hispanic Studies (Digital Humanities), University of Western Ontario, Canada
Master in Artificial Intelligence, Universidad de Sevilla, Spain

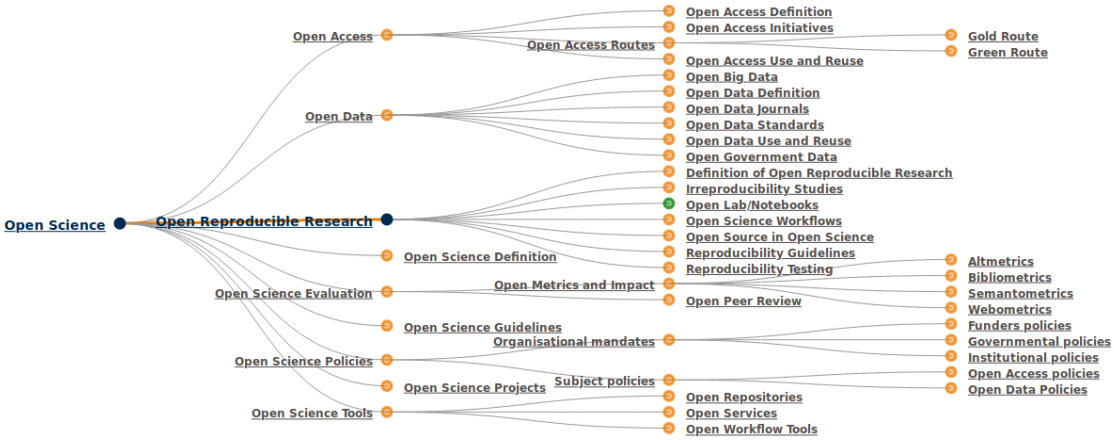
Ex-Research Software Engineer at Stanford University, California
Ex-Technical Lead at the CulturePlex Lab, University of Western Ontario, Canada

Introduction

New approaches to data management in arts and humanities:

- What is data in the humanities?
- Data and Software citation practices, PIDs
- **Open Research Notebooks**
- IPR and licensing
- Data Management Plans (DMP)

Introduction



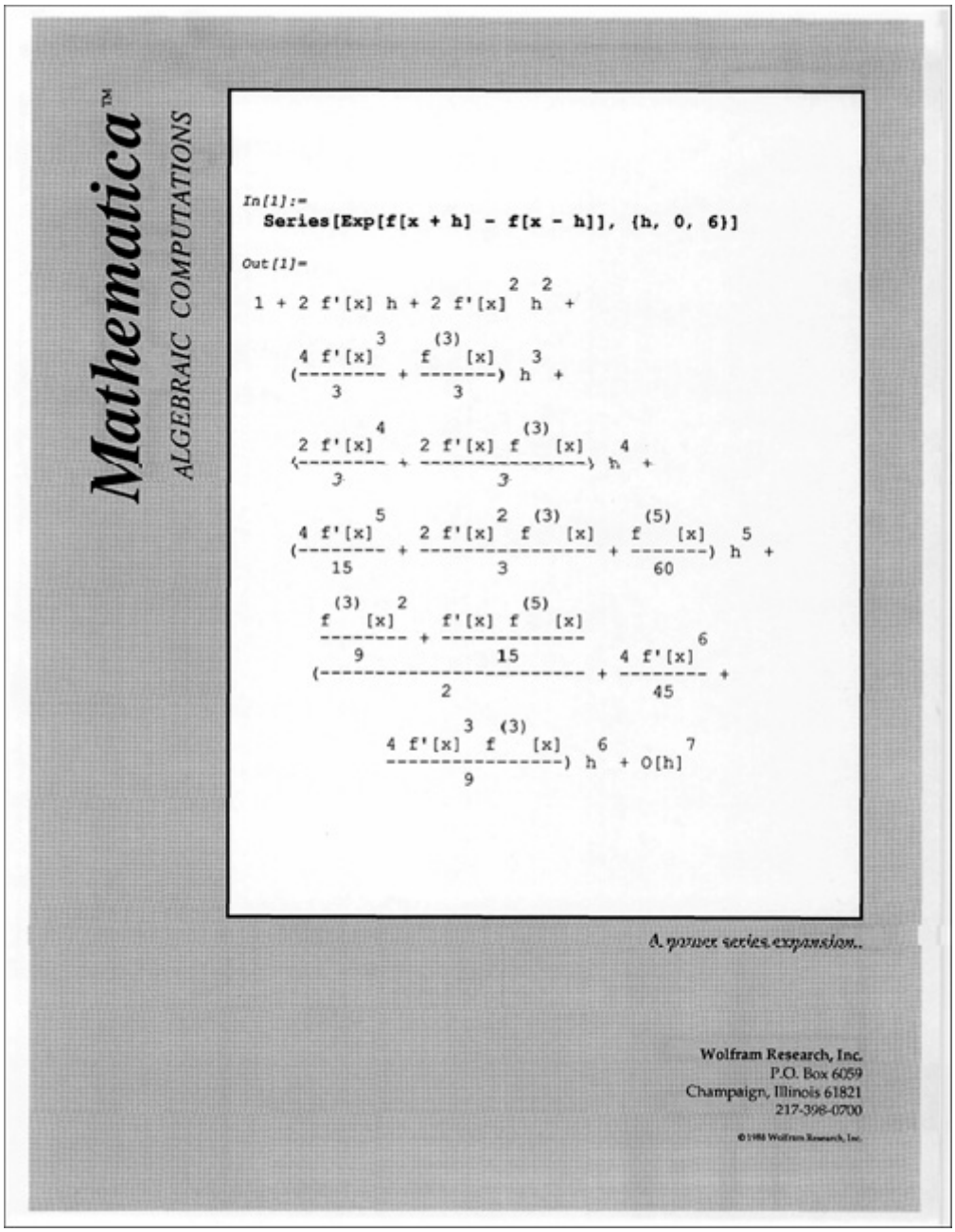
Pontika, Nancy, et al. "Fostering open science to research using a taxonomy and an eLearning portal." Proceedings of the 15th international conference on knowledge technologies and data-driven business. ACM, 2015.

Notebooks

- Virtual environment
- Literate programming (Donald Knuth, 1983)
  - Weaving: Generating a comprehensive document about the program and its maintenance.
  - Tangling: Generating machine executable code
- Document **research procedures**, data, calculations, and findings
- Track methodology to make it easier to **reproduce** results

History

- 1988, Wolfram Mathematica 1.0 (Demo! (https://www.wolfram.com/broadcast/video.php?sx=&p=63&v=741))



The Mathematica Story: A Scrapbook https://www.wolfram.com/mathematica/scrapbook/

History

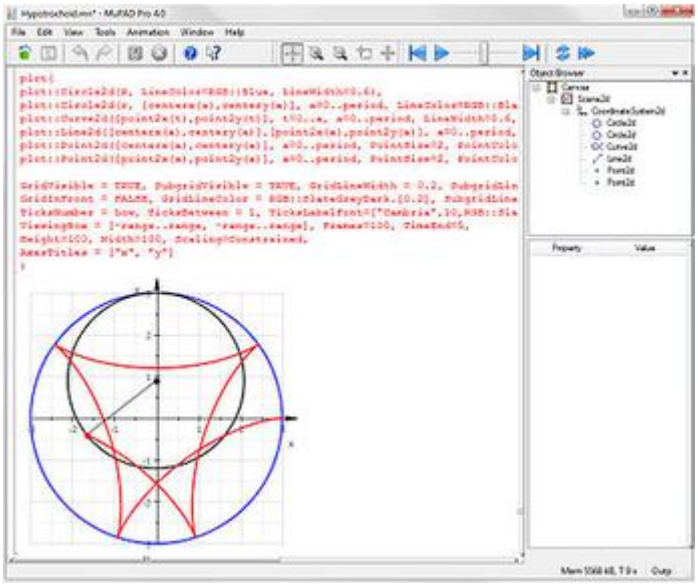
- 1989, Maple 4.3

```
aquila!33> maple
    |\^/|     Maple 6 (IBM INTEL LINUX22)
._|\|   |/|_. Copyright (c) 2000 by Waterloo Maple Inc.
 \  MAPLE  / All rights reserved. Maple is a registered trademark of
 <____ ____> Waterloo Maple Inc.
      |      Type ? for help.
> read `dist.maple`;
Distributed Maple V1.1.7 (c) 1998-2001 Wolfgang Schreiner (RISC-Linz)
See http://www.risc.uni-linz.ac.at/software/distmaple
> dist[initialize]([[virgo,linux], [andromeda,octane]]);
connecting virgo...
connecting andromeda...
                                    okay

> t1 := dist[start](int, x^n, x):
> t2 := dist[start](int, x^n, n);
> dist[wait](t1) + dist[wait](t2);
                                    (n + 1)     n
                                x           x
                                -------- + -----
                                 n + 1     ln(x)

> dist[terminate]();
                                    okay

> quit;
```

Schreiner, Wolfgang, Christian Mittermaier, and Karoly Bosa. "Distributed Maple: Parallel computer algebra in networked environments." Journal of Symbolic Computation 35.3 (2003): 305-347.

History

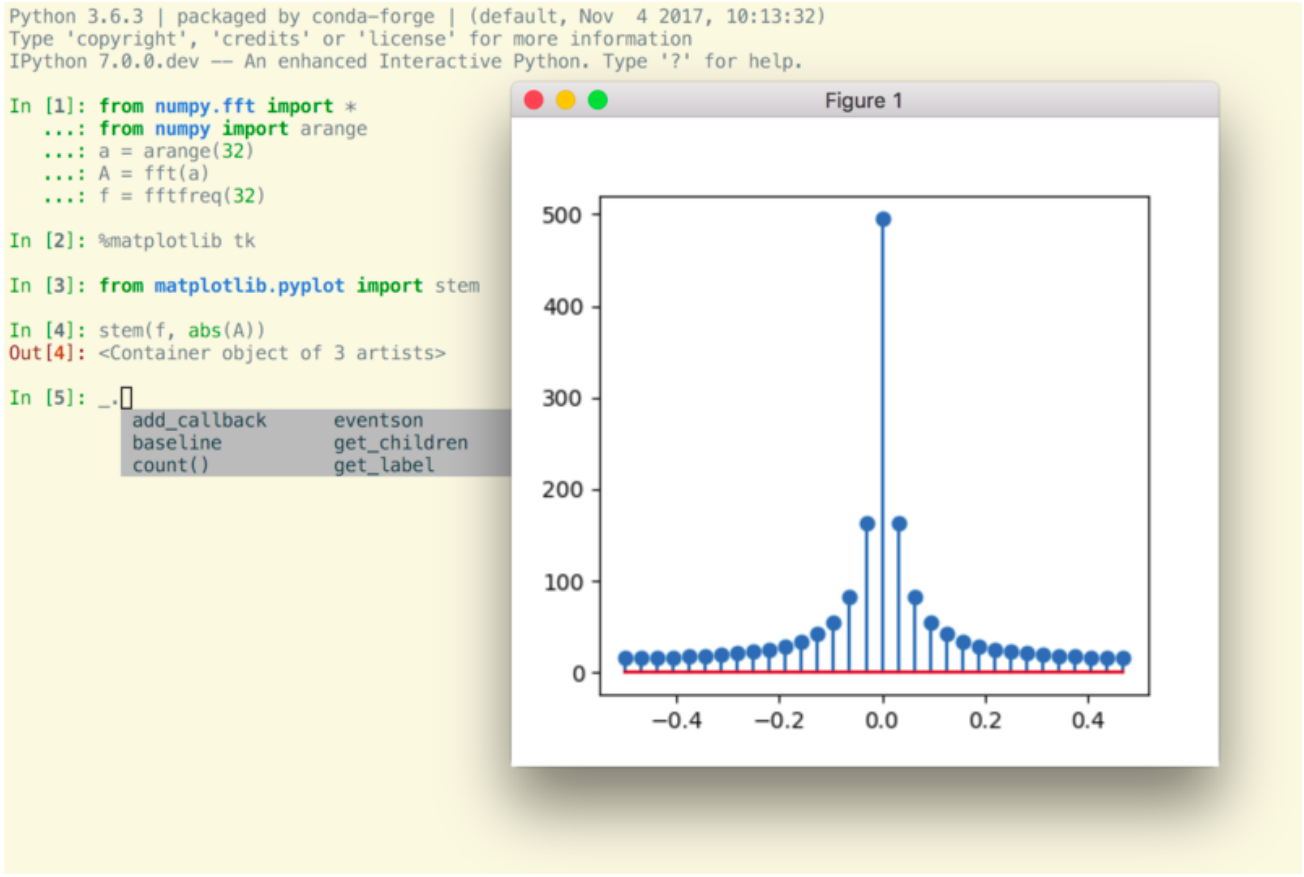
- 1997, MuPAD (acquired by Matlab in 2008)



By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=15024533

History

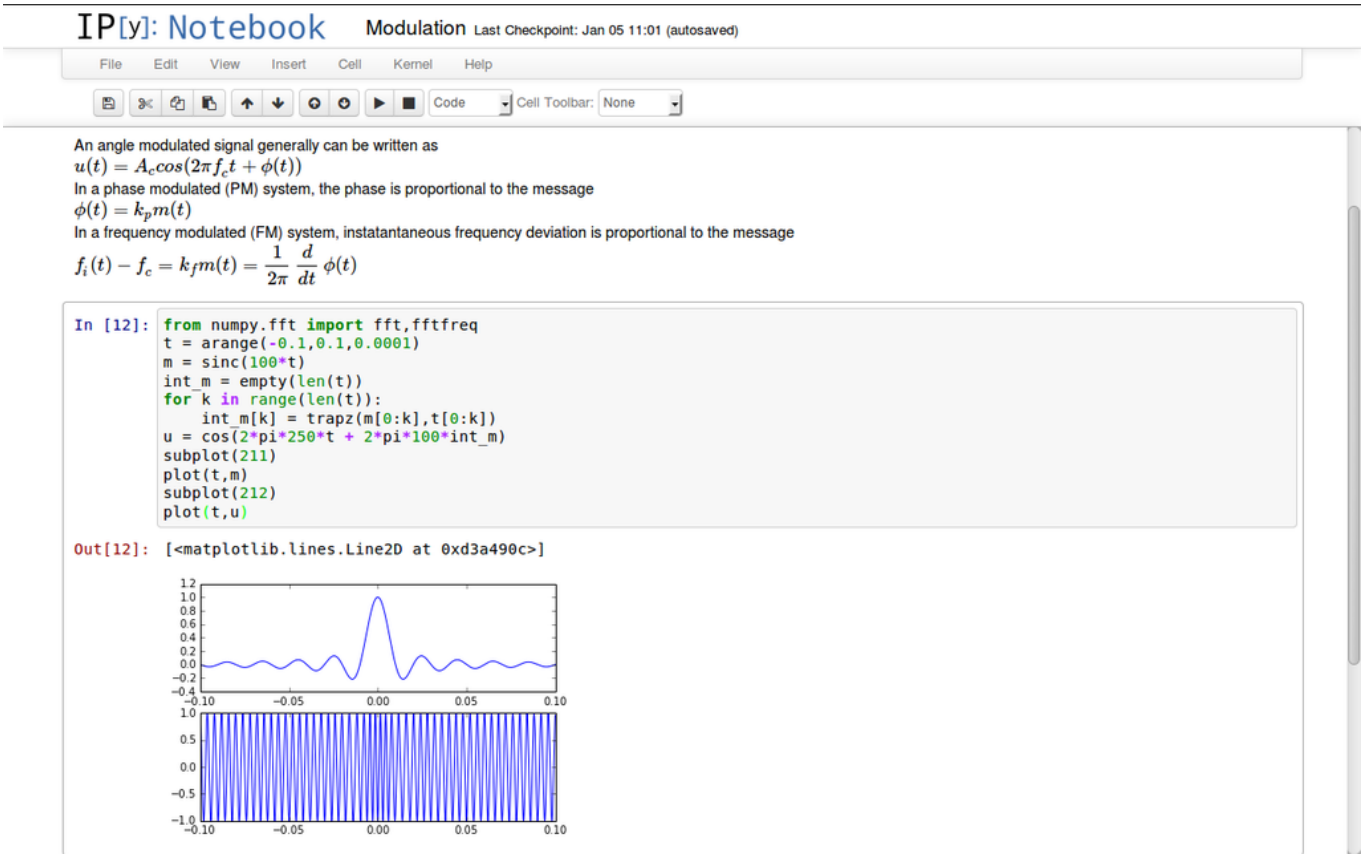
- 2001, IPython



By Mbussonn - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68247365 (https://commons.wikimedia.org/w/index.php?curid=68247365)

History

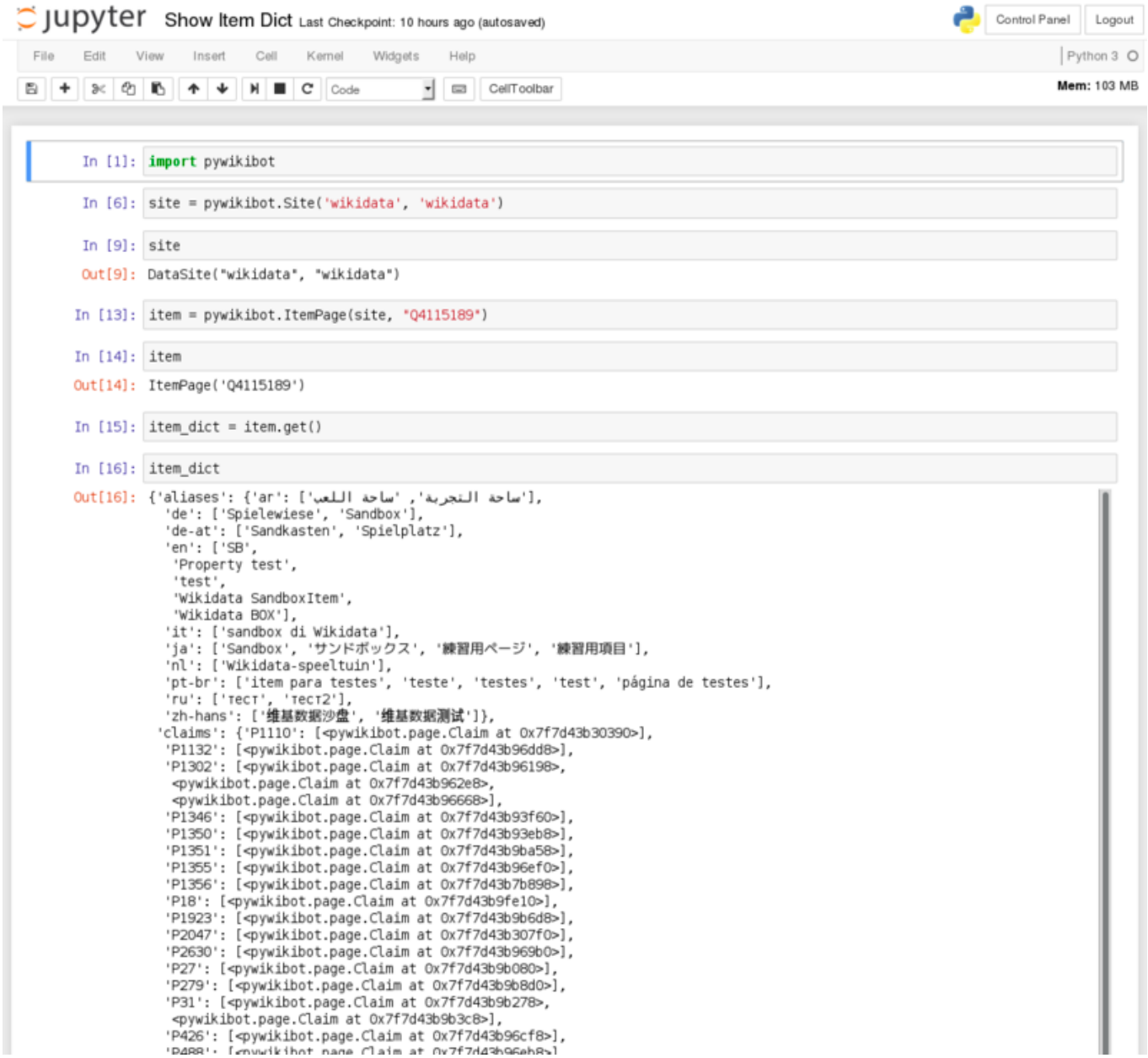
- 2011, IPython 0.12 (IPython Notebook)



By Shishirdasika - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=30482405

History

- 2014, Jupyter



Notebooks

Libre

- Apache Zeppelin (https://zeppelin.apache.org/)
- **Jupyter Notebook** (https://jupyter.org/) (formerly IPython)
- JupyterLab (https://jupyterlab.readthedocs.io/en/stable/)
- Mozilla Iodide (https://iodide.io)
- SageMath (http://www.sagemath.org/)
- R Markdown (https://rmarkdown.rstudio.com/)

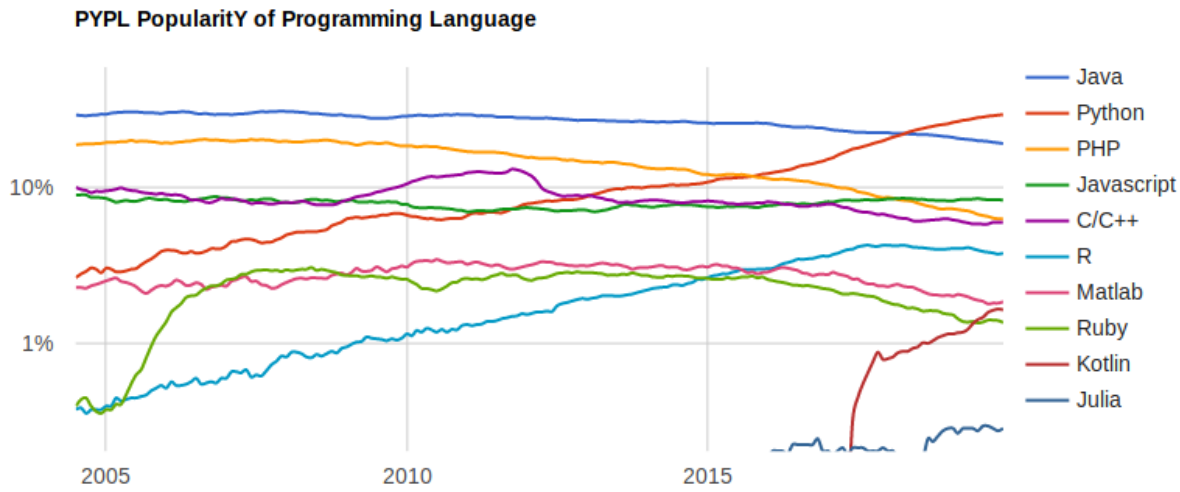
Notebooks

Not libre

- Mathematica
- Maple
- Matlab
- SMath Studio
- Carbide
- Databricks cloud
- Observable

Jupyter

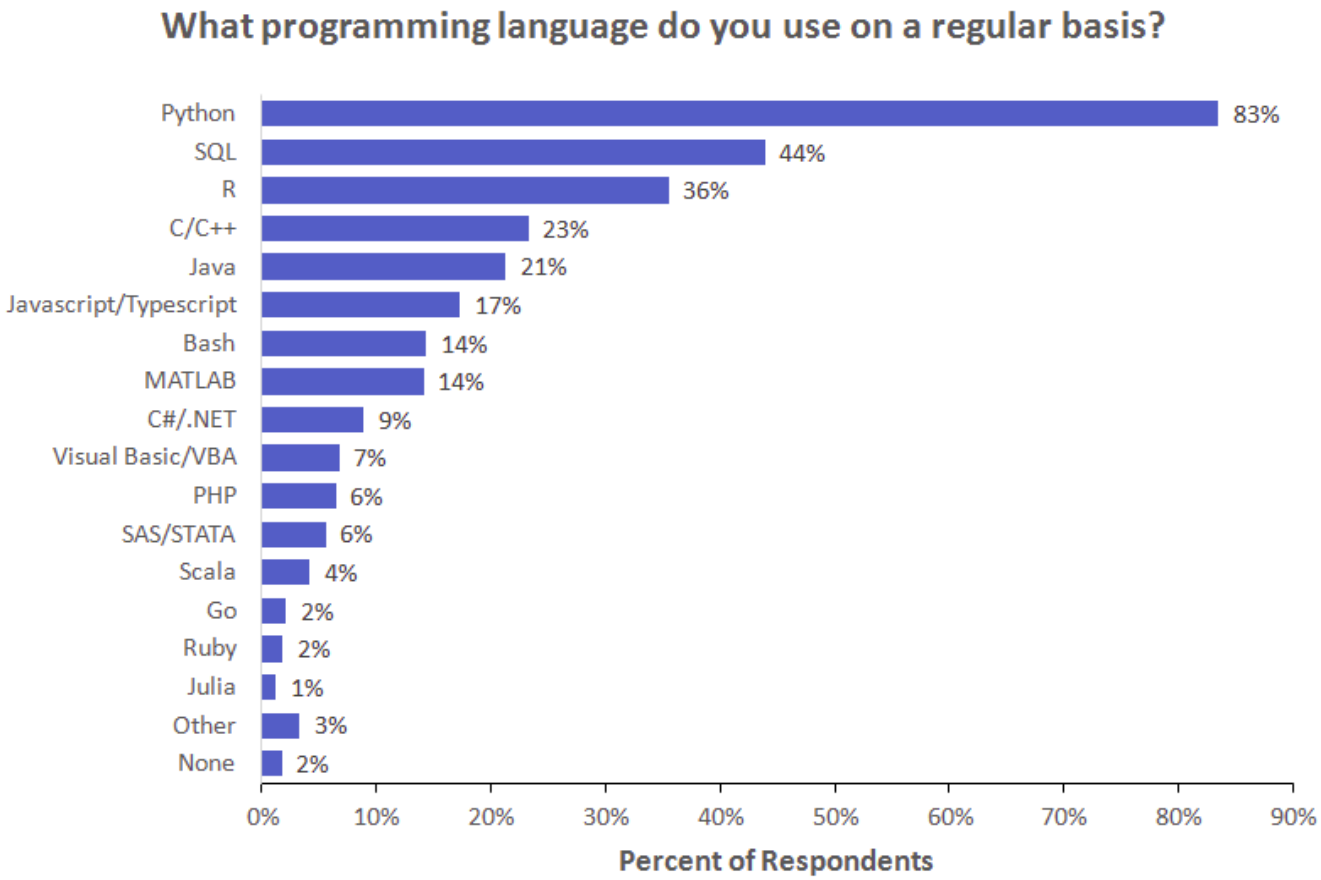
Why Python though? Trends worldwide...



Pierre Carbonnelle, 2019. http://pypl.github.io/PYPL.html

Jupyter

Why Python though? In Data Science...

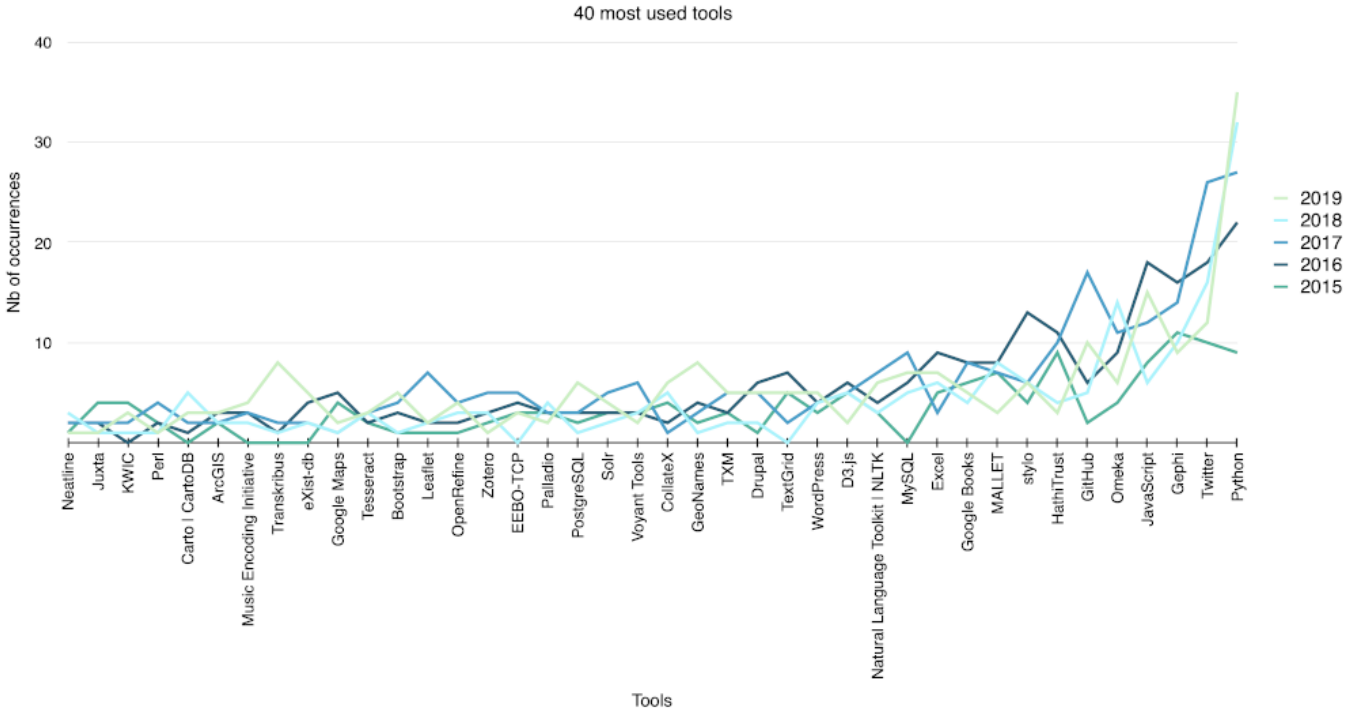


Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

Jupyter

Why Python though? In **Digital Humanities**...



Laure Barbot et. al. Tools mentioned in the proceedings of the annual ADHO conferences (2015–2019) https://lehkost.github.io/tools-dh-proceedings/index.html

What is Jupyter?

Jupyter notebook, formerly known as IPython (or Interactive Python), is a flexible and powerful open source research tool that can help you keep a narrative of your coding process. The name Jupyter is an acronym of the three core languages it was designed for: **JU**lia, **PYT**hon, and **R**. Project Jupyter supports interactive data science and scientific computing across more than 40 programming languages.

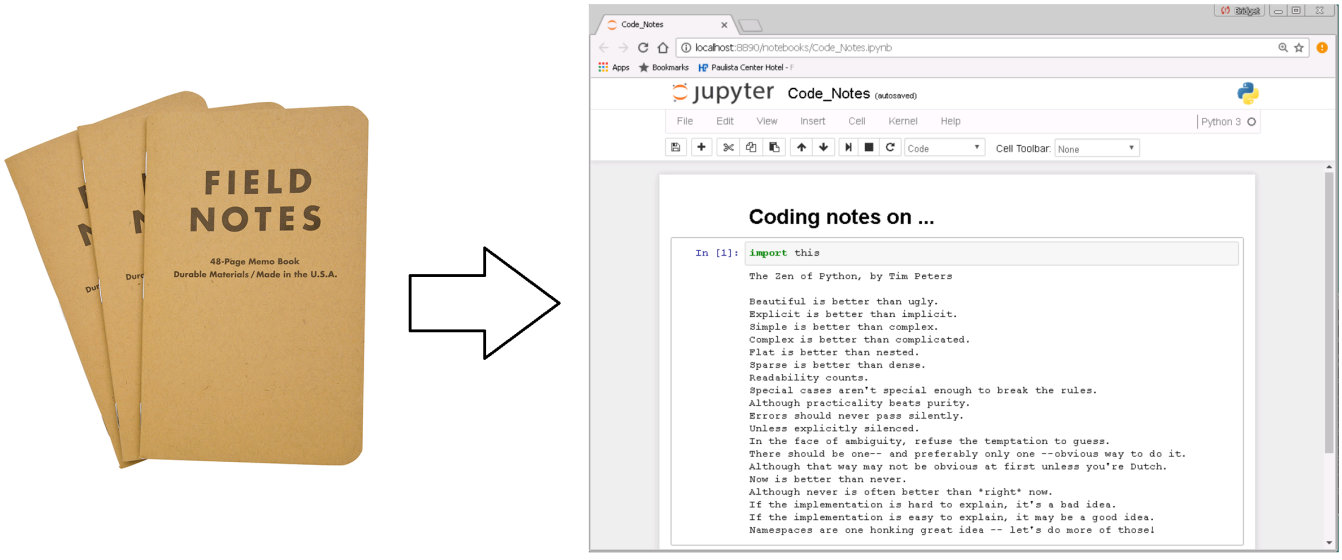
Open source, interactive data science and scientific computing across over 40 programming languages.

https://jupyter.org/

Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

Jupyter as Coding Diary

You can think of the notebook as a **lab or field diary** that keeps a detailed record of the steps you take as you develop scripts and programming workflows. Just as you would with a field notebook, it is important to develop good note-taking habits.



Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

Jupyter as *Literate Computing*

> A literate computing environment is one that allows users not only to execute commands but also to **store in a literate document format the results of these commands** along with figures and free-form text that can include formatted mathematical expressions. In practice it can be seen as a blend of a command-line environment such as the Unix shell with a word processor, since the resulting documents can be read like text, but contain blocks of code that were executed by the underlying computational system

-- Fernando Pérez

Millman, KJ and Fernando Perez. 2014. “Developing open source scientific practice”. In Implementing Reproducible Research, Ed. Victoria Stodden, Friedrich Leisch, and Roger D. Peng. https://osf.io/h9gsd/

Jupyter Architecture

On the **front-end**, the user will work with the:

1. **Web Application:** Browser-based tool for interactive development of notebook documents
2. **Notebook Document:** A representation of all content visible in the web application (internally stored as JSON files with the .ipynb extension)

Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

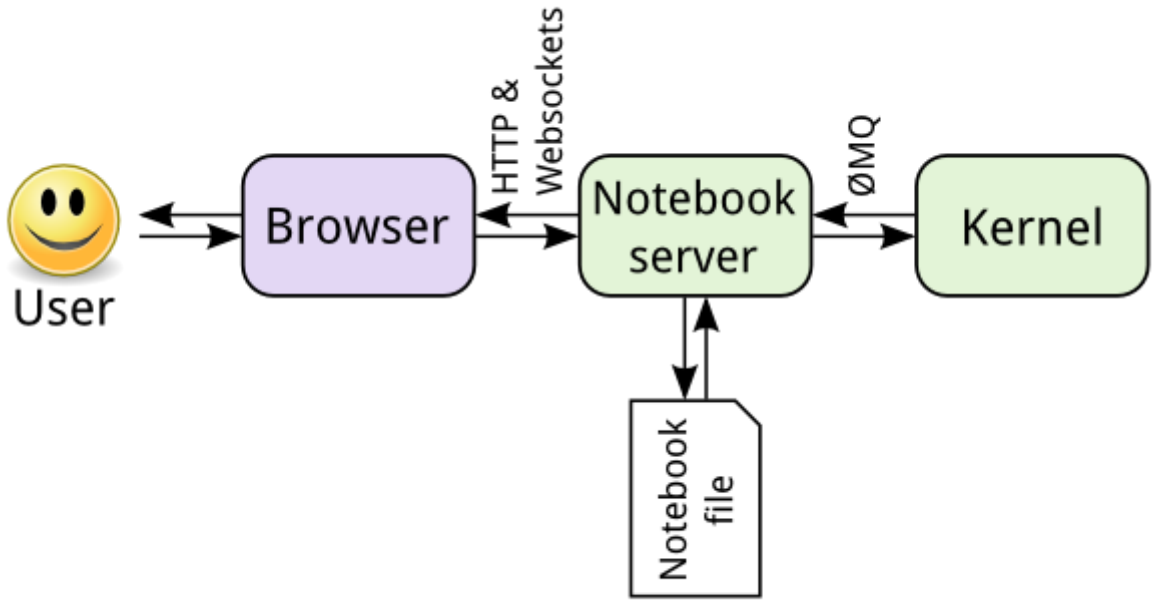
Jupyter Architecture

On the **back-end** (see more at [1 (http://jupyter-notebook.readthedocs.io/en/latest/notebook.html), 2 (http://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html)]):

1. **Kernel:** A separate process responsible for running user code. We will be working on Python kernels, although Jupyter is capable of interfacing with other programming languages as well.
2. **Notebook Server:** Communicates with kernel and routes the Python programming language to the web browser.

Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

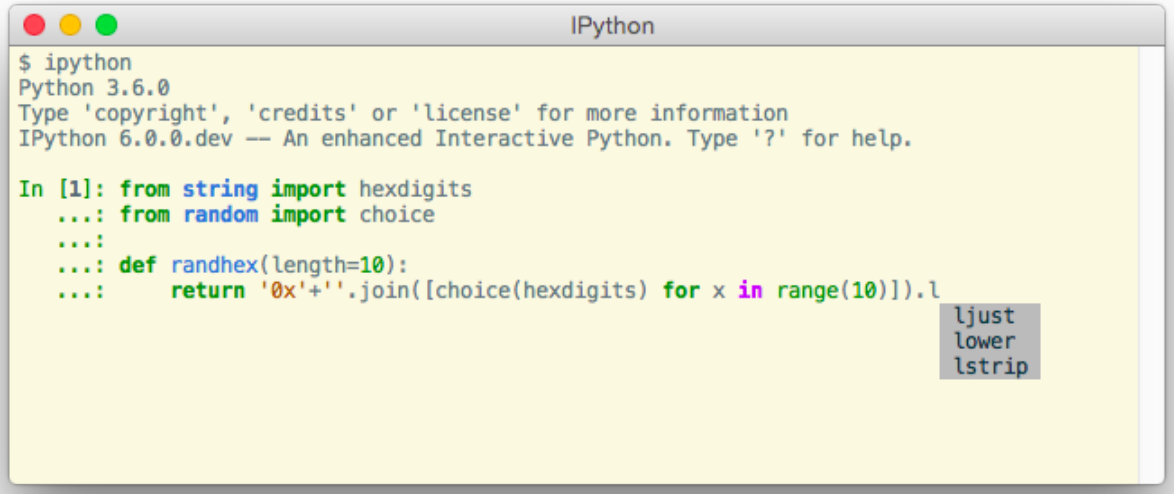
Jupyter Architecture



How IPython and Jupyter Notebook work https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Jupyter and the IPython Kernel

At its core, Jupyter works as a frontend to IPython.



IPython https://ipython.readthedocs.io/en/stable/

In [1]:

```
import secrets; from IPython.display import display, Markdown; from notebook import notebookapp
; display(Markdown(f"[File → Open → New → Terminal]({list(notebookapp.list_running_servers())
[0]['base_url']}terminals/{secrets.token_hex(8)})"))
```

File → Open → New → Terminal (/terminals/94788ac49d3b0563)

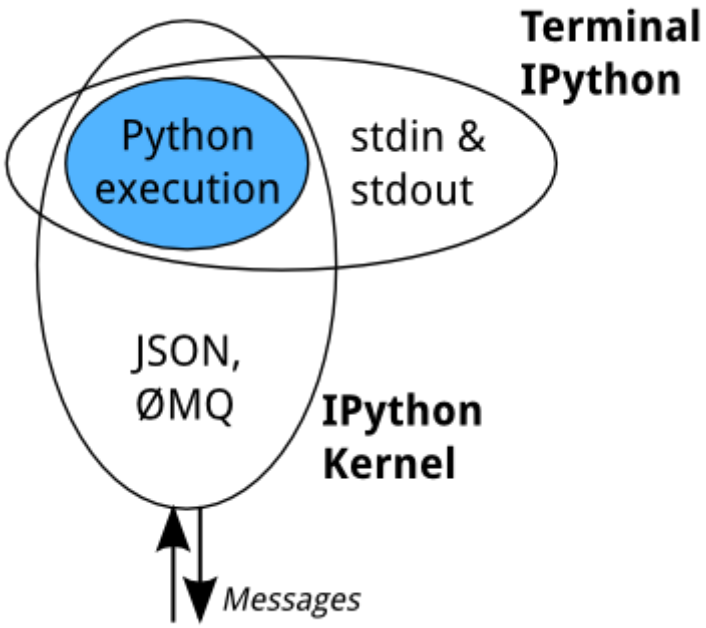
Jupyter and the IPython Kernel

The REPL (Read-Eval-Print-Loop):

```
while True:
    code = input(">>> ")
    exec(code)
```

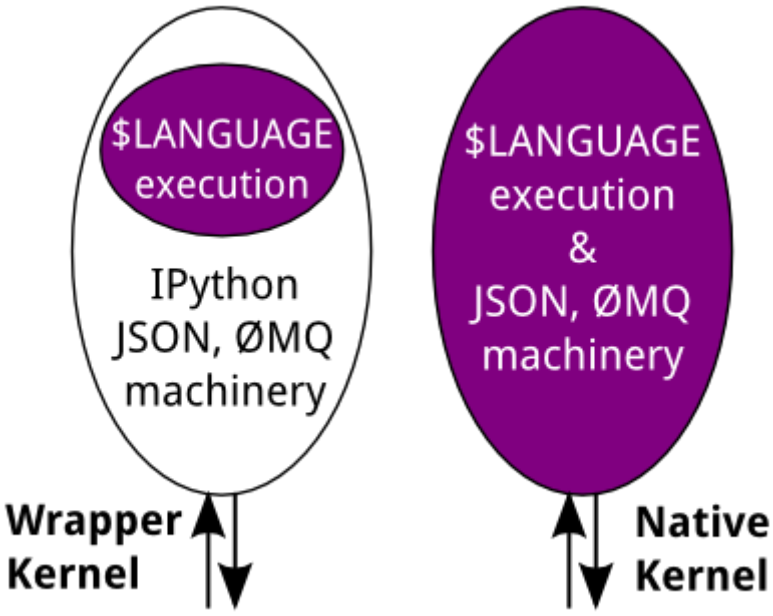
How IPython and Jupyter Notebook work https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Jupyter Kernels



How IPython and Jupyter Notebook work https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Jupyter Kernels



How IPython and Jupyter Notebook work https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

Jupyter Kernels

Name	Jupyter/IPython Version	Language(s)
Agda kernel (https://github.com/lclem/agda-kernel)		
Dyalog Jupyter Kernel (https://github.com/Dyalog/dyalog-jupyter-kernel)		APL
Coarray-Fortran (https://github.com/sourceryinstitute/jupyter-CAF-kernel)	Jupyter 4.0	Fortran 20
Ansible Jupyter Kernel (https://github.com/ansible/ansible-jupyter-kernel)	Jupyter 5.6.0.dev0	An
sparkmagic (https://github.com/jupyter-incubator/sparkmagic)	Jupyter >=4.0	Pyspark (Python 2 & 3), Spark Sp
sas_kernel (https://github.com/sassoftware/sas_kernel)	Jupyter 4.0	pytho
IPyKernel (https://github.com/ipython/ipykernel)	Jupyter 4.0	python 2.7
IJulia (https://github.com/JuliaLang/IJulia.jl)		juli
IHaskell (https://github.com/gibiansky/IHaskell)		gh
IRuby (https://github.com/SciRuby/iruby)		rub
tslab (https://github.com/yunabe/tslab)		Typescript 3.7.2, JavaScript
IJavascript (https://github.com/n-riesco/ijavascript)		nodejs
ITypeScript (https://github.com/nearbydelta/itypescript)		Typescrip
jpCoffeescript (https://github.com/n-riesco/jp-coffeescript)		coffeescrip
jp-LiveScript (https://github.com/p2edwards/jp-livescript)		livescrip
ICSharp (https://github.com/zabirauf/icsharp)	Jupyter 4.0	
IRKernel (http://irkernel.github.io/)	IPython 3.0	
SageMath (http://www.sagemath.org/)	Jupyter 4	
pari_jupyter (https://github.com/jdemeyer/pari_jupyter)	Jupyter 4	PARI/G
IFSharp (https://github.com/fsprojects/IfSharp)	Jupyter 4	
lgo (https://github.com/yunabe/lgo)	Jupyter >= 4, JupyterLab	G
iGalileo (https://github.com/cascala/igalileo)	Jupyter >= 4, JupyterLab	Galileo
gopherlab (https://github.com/fabian-z/gopherlab)	Jupyter 4.1, JupyterLab	G
Gophernotes (https://github.com/gopherdata/gophernotes)	Jupyter 4, JupyterLab, nteract	G
IGo (https://github.com/takluyver/igo)		G
IScala (https://github.com/mattpap/IScala)		
almond (old name: Jupyter-scala) (https://github.com/almond-sh/almond)	IPython>=3.0	Scal
IErlang (https://github.com/robbielynch/ierlang)	IPython 2.3	

	Name	Jupyter/IPython Version	Language(s)
	ITorch (https://github.com/facebook/iTorch)	IPython >= 2.2 and <= 5.x	Torch 7
	IElixir (https://github.com/pprzetacznik/IElixir)	Jupyter >= 4.0	Elix
	ierl (https://github.com/filmor/ierl)	Jupyter >= 4.0	Erlang >= 19, Elixir >= 1.4,
	IAldor (https://github.com/mattpap/IAldor)	IPython >= 1	
	IOCaml (https://github.com/andrewray/iocaml)	IPython >= 1.1	OCaml
	OCaml-Jupyter (https://github.com/akabe/ocaml-jupyter)	Jupyter >= 4.0	OCaml
	IForth (https://github.com/jdfreder/iforth)	IPython >= 3	
	peforth (https://github.com/hcchengithub/peforth)	IPython 6/Jupyter 5	
	IPerl (https://metacpan.org/release/Devel-IPerl)		
	Perl6 (https://github.com/gabrielash/p6-net-jupyter)	Jupyter >= 4	
	IPerl6 (https://github.com/timo/iperl6kernel)		
	Jupyter-Perl6 (https://github.com/bduggan/p6-jupyter-kernel)	Jupyter	
	IPHP (https://github.com/dawehner/ipython-php)	IPython >= 2	PH
	Jupyter-PHP (https://github.com/Litipk/Jupyter-PHP)	Jupyter 4.0	PHP
	IOctave (https://github.com/calysto/octave_kernel)	Jupyter	
	IScilab (https://github.com/calysto/scilab_kernel)	Jupyter	
	MATLAB Kernel (https://github.com/calysto/matlab_kernel)	Jupyter	
	Bash (https://github.com/takluyver/bash_kernel)	IPython >= 3	
	Z shell (https://github.com/danylo-dubinin/zsh-jupyter-kernel)	IPython >= 3	zs
	Pharo Smalltalk (https://github.com/jmari/JupyterTalk)	IPython >= 3	M
	PowerShell (https://github.com/vors/jupyter-powershell)	IPython >= 3	V
	CloJupyter (https://github.com/roryk/clojupyter)	Jupyter	Clojur
	CLJ-Jupyter (https://github.com/achesnais/clj-jupyter)	Jupyter	
	jupyter-kernel-jsr223 (https://github.com/fiber-space/jupyter-kernel-jsr223)	Jupyter>=4.0	Cl
	Hy Kernel (https://github.com/bollwyvl/hy_kernel/)	Jupyter	
	Calysto Hy (https://github.com/Calysto/calysto_hy)	Jupyter	
	Redis Kernel (https://github.com/supercoderz/redis_kernel)	IPython >= 3	
	jove (https://www.npmjs.com/package/jove)		
	jp-babel (https://www.npmjs.com/package/jp-babel)	Jupyter	
	ICalico (http://wiki.roboteducation.org/ICalico)	IPython >= 2	
	IMathics (http://nbviewer.ipython.org/gist/sn6uv/8381447)		
	IWolfram (https://github.com/mmatera/iwolfram)		Wolfram Math
	Lua Kernel (https://github.com/neomantra/lua_ipython_kernel)		

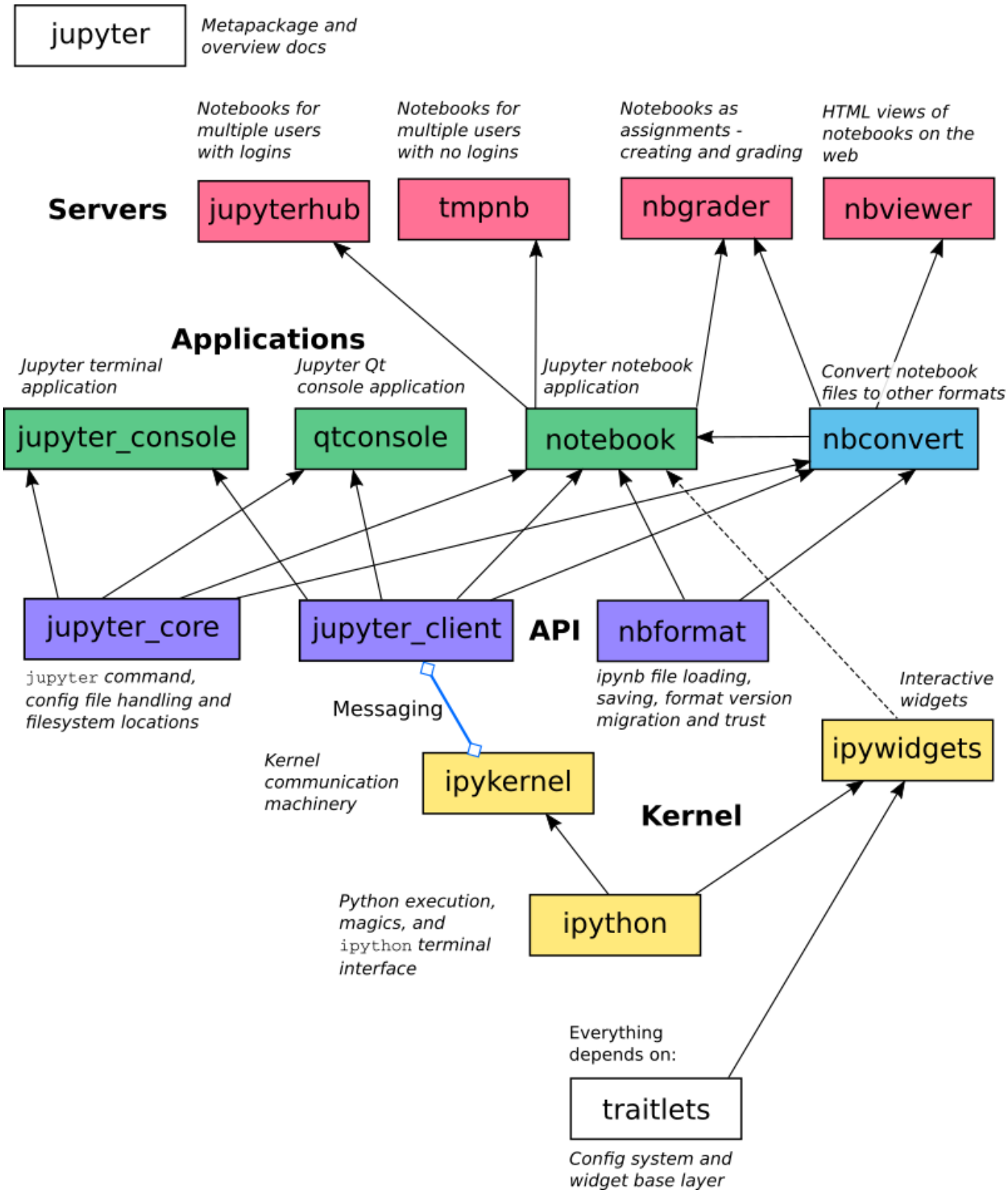
Name	Jupyter/IPython Version	Language(s)
IPurescript (https://github.com/Eoksni/ipurescript)		Pu
IPyLua (https://github.com/pakozm/IPyLua)		
ILua (https://github.com/guysv/ilua)		
Calysto Scheme (https://github.com/Calysto/calysto_scheme)		
Calysto Processing (https://github.com/Calysto/calysto_processing)		Processin
idl_kernel (https://github.com/lstagner/idl_kernel)		
Mochi Kernel (https://github.com/pya/mochi-kernel)		
Lua (used in Splash) (https://github.com/scrapinghub/splash/tree/master/splash/kernel)		
Apache Toree (formerly Spark Kernel) (https://github.com/apache/incubator-toree)	Jupyter	Scala, P
Skulpt Python Kernel (https://github.com/Calysto/skulpt_python)		Skulp
Calysto Bash (https://github.com/Calysto/calysto_bash)		
MetaKernel Python (https://github.com/Calysto/metakernel/tree/master/metakernel_python)		
IVisual (https://pypi.python.org/pypi/IVisual)		'
IBrainfuck (https://github.com/robbielynch/ibrainfuck)		E
KDB+/Q Kernel (IKdbQ) (https://github.com/jvictorchen/IKdbQ)	IPython >= 3.1	
KDB+/Q Kernel (KdbQ Kernel) (https://github.com/newtux/KdbQ_kernel)	Jupyter	
ICryptol (https://github.com/GaloisInc/ICryptol)		
cling (https://github.com/root-mirror/cling)	Jupyter 4	
xeus-cling (https://github.com/QuantStack/xeus-cling)	Jupyter >= 5.1	
Xonsh (https://github.com/calysto/xonsh_kernel)		
Prolog (https://github.com/Calysto/calysto_prolog)		
SWI-Prolog (https://github.com/madmax2012/SWI-Prolog-Kernel)	Jupyter >=4.0	SW
cl-jupyter (https://github.com/fredokun/cl-jupyter)	Jupyter	Comr
common-lisp-jupyter (https://github.com/yitzchak/common-lisp-jupyter)	Jupyter	Comr
Maxima-Jupyter (https://github.com/robert-dodier/maxima-jupyter)	Jupyter	
Calysto LC3 (https://github.com/Calysto/calysto_lc3)		
Yacas (https://github.com/grzegorzmazur/yacas_kernel)		
IJython (https://github.com/suvarchal/IJython)		Jy
ROOT (https://github.com/root-mirror/root/tree/master/bindings/pyroot/JupyROOT)	Jupyter	C++
Gnuplot Kernel (https://github.com/has2k1/gnuplot_kernel)		
Tcl (https://github.com/rpep/tcl_kernel)	Jupyter	

	Name	Jupyter/IPython Version	Language(s)
	J (https://github.com/martin-saurer/jkernel)	Jupyter Notebook/Lab	J 805-807 (J9
	Jython (https://github.com/fiber-space/jupyter-kernel-jsr223)	Jupyter>=4.0	Jythor
	C (https://github.com/brendan-rius/jupyter-c-kernel)	Jupyter	
	TaQL (https://github.com/tammojan/taql-jupyter)	Jupyter	
	Coconut (http://coconut-lang.org/)	Jupyter	
	SPARQL (https://github.com/paulovn/sparql-kernel)	Jupyter 4	Python 2.7
	AIML chatbot (https://github.com/paulovn/aiml-chatbot-kernel)	Jupyter 4	Py
	IArm (https://github.com/DeepHorizons/iarm)	Jupyter 4	ARMv6
	SoS (https://github.com/vatlab/SOS)	Jupyter 4	Pythc
	jupyter-nodejs (https://github.com/notablemind/jupyter-nodejs)	Jupyter, iPython 3.x	NodeJS, Babel, Cloj
	Pike (https://github.com/kevinior/jupyter-pike-kernel)	IPython >= 3	Pik
	imatlab (https://github.com/imatlab/imatlab)	ipykernel >= 4.1	MATLAB >
	jupyter-kotlin (https://github.com/ligee/kotlin-jupyter)	Jupyter	Kotlin 1.1-M
	jupyter_kernel_singular (https://github.com/sebasguts/jupyter_kernel_singular)	Jupyter	Singu
	spylon-kernel (https://github.com/maxpoint/spylon-kernel)	ipykernel >=4.5	python >= 3.5, scala
	mit-scheme-kernel (https://github.com/joeltg/mit-scheme-kernel)	Jupyter 4.0	MIT Sch
	elm-kernel (https://github.com/abingham/jupyter-elm-kernel)	Jupyter	
	SciJava Jupyter Kernel (https://github.com/hadim/scijava-jupyter-kernel)	Jupyter 4.3.0	Java + 9 scripting lar
	Isbt (https://github.com/ktr-skmt/Isbt)	Jupyter 4.3.0	sbt
	BeakerX (http://beakerx.com/)		
	MicroPython (https://github.com/goatchurchprime/jupyter_micropython_kernel/)	Jupyter	ESP8266
	IJava (https://github.com/SpencerPark/IJava)	Jupyter	
	Guile (https://github.com/jerry40/guile-kernel)	Jupyter 5.2	Guil
circuitpython_kernel (https://github.com/adafruit/circuitpython_kernel)		Jupyter	Circu (https://github.com/adafruit/circui
	stata_kernel (https://github.com/kylebarron/stata_kernel)	Jupyter >=5	
	iPyStata (https://github.com/TiesdeKok/ipystata)	Jupyter	
	IRacket (https://github.com/rmculpepper/iracket)	IPython >= 3	Racket
	jupyter-dot-kernel (https://github.com/laixintao/jupyter-dot-kernel)	Jupyter >= 4.0	dot/
	Teradata SQL kernel and extensions (https://teradata.github.io/jupyterextensions/)	JupyterLab >= 0.34	
	HiveQL Kernel (https://github.com/EDS-APHP/HiveQLKernel)	Jupyter >= 5	(https://en.wikipedia.org/wiki/Apach

	Name	Jupyter/IPython Version	Language(s)
	EvCxR Jupyter Kernel (https://github.com/google/evcxr/tree/master/evcxr_jupyter)	Jupyter 4, JupyterLab, nteract	Rust >
	StuPyd Kernel (https://github.com/StuPyd/demo-kernel)	Jupyter >= 4	StuPyd Programming La (https://github.com/StuPyd/stup
	coq_jupyter (https://github.com/EugeneLoy/coq_jupyter)	Jupyter 5	
	Cadabra2 (https://github.com/kpeeters/cadabra2/blob/master/JUPYTER.rst)	Jupyter 5	Cadabra2 (https://cadabra.
	iMongo (https://github.com/gusutabopb/imongo)		M
	jupyter_kernel_chapel (http://github.com/krishnadey30/jupyter_kernel_chapel)	Jupyter	Chapel (https://github.com lang
	A Jupyter kernel for Vim script (https://github.com/mattn/vim_kernel)	Jupyter	Vim script (https://github.com/\
	SSH Kernel (https://github.com/NII-cloud-operation/sshkernel)	Jupyter	
	GAP Kernel (https://gap-packages.github.io/JupyterKernel/)	Jupyter	GAP
	Wolfram Language for Jupyter (https://github.com/WolframResearch/WolframLanguageForJupyter)		Wolfram Engine, i.e., a Wolfram or Mathematica ins `wolframscript` is opt recom
	GrADS kernel (https://github.com/ykatsu111/jupyter-grads-kernel)		GrAD
	Bacatá (https://github.com/cwi-swat/bacata)	Jupyter	Java & Rascal (https://rascal-
	nelu-kernelu (https://github.com/3Nigma/nelu-kernelu)	Jupyter	Nc
	IPolyglot (https://github.com/hpi-swa/ipolyglot)	Jupyter	JavaScript, Ruby, Python, R, a (https://www.graalvm.org/docs/re manual/p
	Emu86 Kernel (https://github.com/gcallah/Emu86/tree/master/kernels)	Jupyter	Intel Assembly La

Jupyter Kernels, https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

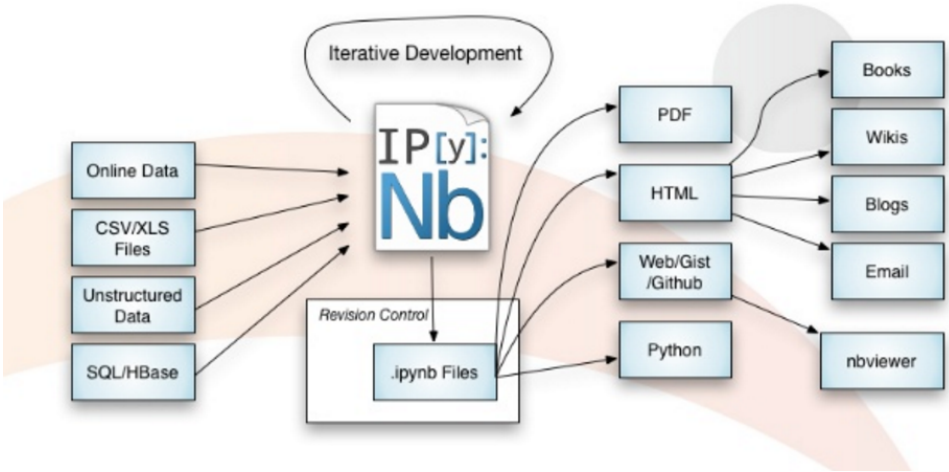
Jupyter Ecosystem



A Visual Overview of Projects https://jupyter.readthedocs.io/en/latest/architecture/visual_overview.html

Jupyter Integrations

There has been considerable development by both Project Jupyter and external collaborators that have yielded a multitude of options for Jupyter users.



Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

Jupyter Notebooks as a Powerful Tool for Reproducible Research?

Jupyter Notebooks are great because they **facilitate**:

- **Documentation and literate programming** by combining rich-text narrative *concepts* & machine-readable *code*. The notebeook itself is a data-structure with metadata that can be easily read and parsed.
- **Exploration & development**: Intermediate steps are saved in a clean, well documented format

Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

Jupyter Notebooks as a Powerful Tool for Reproducible Research?

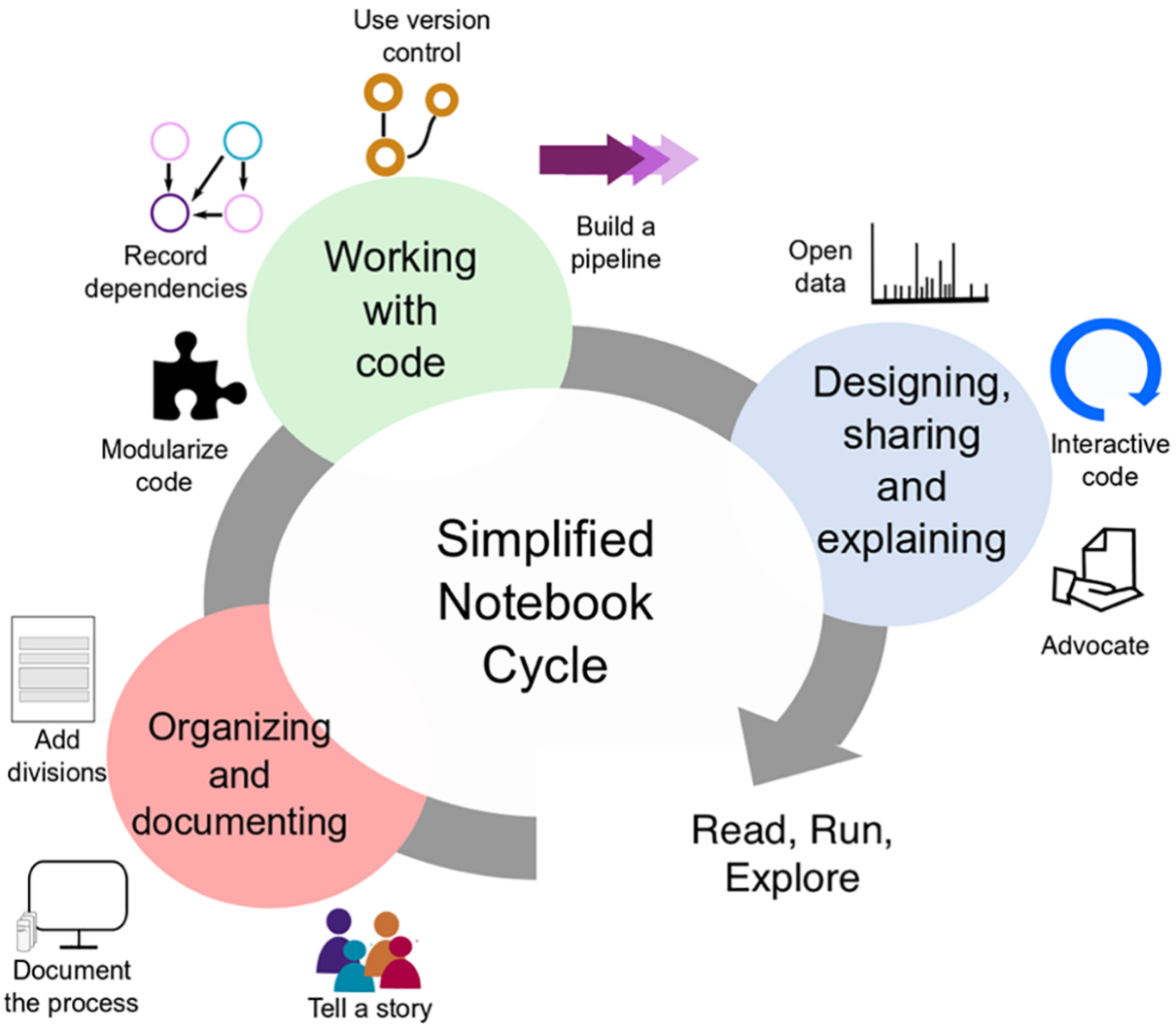
Jupyter Notebooks are great because they **facilitate**:

- **Communication/Collaboration**: sharing research with peers, collaborators, reviewers, public
- **Publishing**: It is simple and quick switch between the development & publishing stage

Software Carpentry and Data Carpentry's ["Reproducible Research using Jupyter Notebooks"](https://reproducible-science-curriculum.github.io/workshop-RR-Jupyter/)

The 10 Rules of Jupyter

Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks



Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 1: Tell a story for an audience

- Explanatory text to tell a compelling story (introduction to the topic, description of steps, and intepretation of the results.
- Not just what you did but why you did it.
- The story will depend on your goal and audience (your primary audience will most likely be your future self!)

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 2: Document the process, not just the results

- Make sure to document all your explorations (even those that led to dead ends!).
- Don't wait until the end of an analysis to add explanatory text.
- Clean, organize, and annotate your notebook (e.g., publication-ready images).

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 3: Use cell divisions to make steps clear

- One cell ~ one meaningful step of the analysis.
- **Modularize your code by cells and label the cells with markdown (https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html).**
- Avoid long cells (anything over 100 lines or one page is too long).
- Organize your notebook into sections.
- Split long notebooks into a series of notebooks and keep a top-level index notebook.

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 4: Modularize code

- Avoid duplicate code
- Wrap code in functions, modules, packages, or libraries.
- It saves space, supports maintenance, eases debugging and interactivity (ipywidgets, https://ipywidgets.readthedocs.io/en/stable/ (https://ipywidgets.readthedocs.io/en/stable/)).

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 5: Record dependencies

- Manage your dependencies using a package or environment manager like `pip` or `Conda` .
- Generate files such as Conda's `environment.yml` or pip's `requirements.txt` (./requirements.txt).
- Print out your dependencies (e.g., using watermark, https://github.com/rasbt/watermark (https://github.com/rasbt/watermark))

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

In [2]:

```
import sys
!!{sys.executable} -m pip install watermark
%reload_ext watermark
%watermark -vim -p requests,jupyter,numpy,rise,pandas
```

```
2019-12-11T15:45:22+01:00

CPython 3.7.4
IPython 7.8.0

requests 2.22.0
jupyter 1.0.0
numpy 1.17.2
rise 5.6.0
pandas 0.25.1

compiler   : GCC 7.3.0
system     : Linux
release    : 5.0.0-36-generic
machine    : x86_64
processor  : x86_64
CPU cores  : 8
interpreter: 64bit
```

Rule 6: Use version control

- Version control is critical (fixing bugs, new versions of code, etc.).
- Git and GitHub are two commonly used solutions for this (templates exist, for example, http://drivendata.github.io/cookiecutter-data-science/ (http://drivendata.github.io/cookiecutter-data-science/)).
- Jupyter uses JSON for serialization, making diffing difficult (use nbdime instead, https://github.com/jupyter/nbdime (https://github.com/jupyter/nbdime)).

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 7: Build a pipeline

- A well-designed notebook can be generalized into a pipeline.
- Place key variable declarations at the top of the notebook.
- Perform preparatory steps, like data cleaning, directly in the notebook and avoid manual interventions.
- Try restarting your kernel and rerunning all cells.
- Notebooks can be parameterized (e.g., papermill, https://github.com/nteract/papermill (https://github.com/nteract/papermill))
  - Code quality
  - Testing with Continuous Integration systems (for example, https://travis-ci.org/ (https://travis-ci.org/)).

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 8: Share and explain your data

- Make your data or a sample of your data publicly available along with the notebook
- Host public copies of your data (for example, figshare (https://figshare.com/), Zenodo (https://zenodo.org/)).
- Include Digital Object Identifiers (DOIs) when possible.

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 9: Design your notebooks to be read, run, and explored

Support others' reuse of your notebooks (add README and LICENSE (https://opensource.org/licenses) files).

- Read:
  - Leave static HTML/PDF versions of all notebooks stored.
  - Use Nbviewer (https://nbviewer.jupyter.org/ (https://nbviewer.jupyter.org/)) to provide static views

Rule 9: Design your notebooks to be read, run, and explored

Support others' reuse of your notebooks (add README and LICENSE (https://opensource.org/licenses) files).

- Run:
  - Use Binder (https://mybinder.org/ (https://mybinder.org/)) to provide a zero-install environment to run your notebooks in the cloud
  - Create a portable containerized environment, such as a Docker image (https://docs.docker.com/ (https://docs.docker.com/)), or a dependency description file.

Rule 9: Design your notebooks to be read, run, and explored

Support others' reuse of your notebooks (add README and LICENSE (https://opensource.org/licenses) files).

- Explore:
  - Consider using ipywidgets (https://ipywidgets.readthedocs.io/ (https://ipywidgets.readthedocs.io/))
  - Consider using Voalà (https://voila.readthedocs.io/ (https://voila.readthedocs.io/))

Rule, Adam, et al. "Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks." PLoS computational biology 15.7 (2019).

Rule 10: Advocate for open research

Become an advocate of this methodology in your lab or workplace!

Working with Jupyter

Let's now see an introduction on how to actually use Jupyter (./overview.ipynb).