



UNIVERSITY OF TWENTE.

CENTER OF EXPERTISE IN BIG GEODATA SCIENCE

Introduction to JupyterLab

dr. ing. Serkan Girgin MSc

s.girgin@utwente.nl

Training Content

- Interactive Notebooks
- Project Jupyter
- JupyterLab
 - Installation
 - Kernels
 - Code Consoles
 - Text Editor
 - Terminals
 - Notebooks
 - User interface
 - Workspaces
 - File management
 - Debugging
 - Real-time collaboration
 - User settings
 - Extensions

Center of Expertise in Big Geodata Science

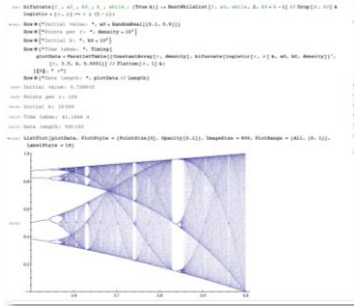
- **Center of Expertise in Big Geodata Science (CRIB)** is a horizontal facility established in **March 2020** to enable the better use of **geospatial cloud computing and big data** technologies in education, research, and institutional strengthening activities at ITC.
- Our mission is to collect, develop, and share operational know-how on cloud computing and big data technologies to solve large-scale geospatial problems.
- Our vision is to position UT/ITC as a globally renowned center of excellence in geospatial cloud computing and big data science.

<https://itc.nl/big-geodata>

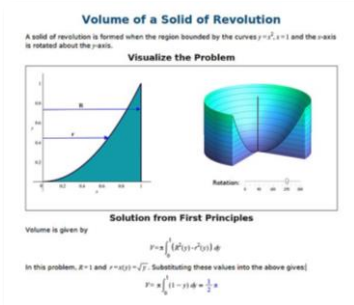
CRIB

Interactive Notebooks

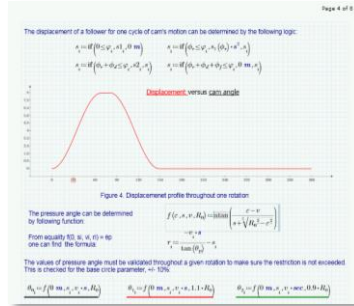
Mathematica



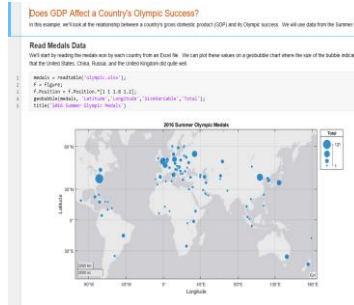
Maple



Mathcad



Matlab Live Editor



Open-Source Projects

Jupyter

Apache Zeppelin

Apache Spark Notebook

SageMath

Polynote

Org Mode

Starboard

...

...

IPython

- IPython (Interactive Python) is a command shell for interactive computing, first developed by [Fernando Perez](#) in 2001
 - An enhanced interactive Python shell (e.g. introspection, shell syntax, tab completion, history, etc.)
 - A **decoupled two-process communication** model, which allows for multiple clients to connect to a computation kernel
 - An architecture for interactive parallel computing (now part of [ipyparallel](#))
- It supported additional interactive shells, a browser-based notebook interface, and interactive data visualization
- IPython 3.x was the last monolithic release and in 2014 the language-agnostic parts of the project have been moved to **Project Jupyter**

```
Python 3.6.3 | packaged  
Type 'copyright', 'credi  
IPython 7.0.0.dev -- An
```

```
In [1]: from numpy.fft i  
...: from numpy impor  
...: a = arange(32)  
...: A = fft(a)  
...: f = fftfreq(32)
```

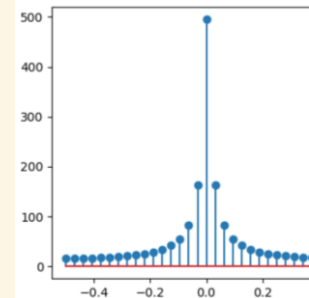
```
In [2]: %matplotlib tk
```

```
In [3]: from matplotlib.
```

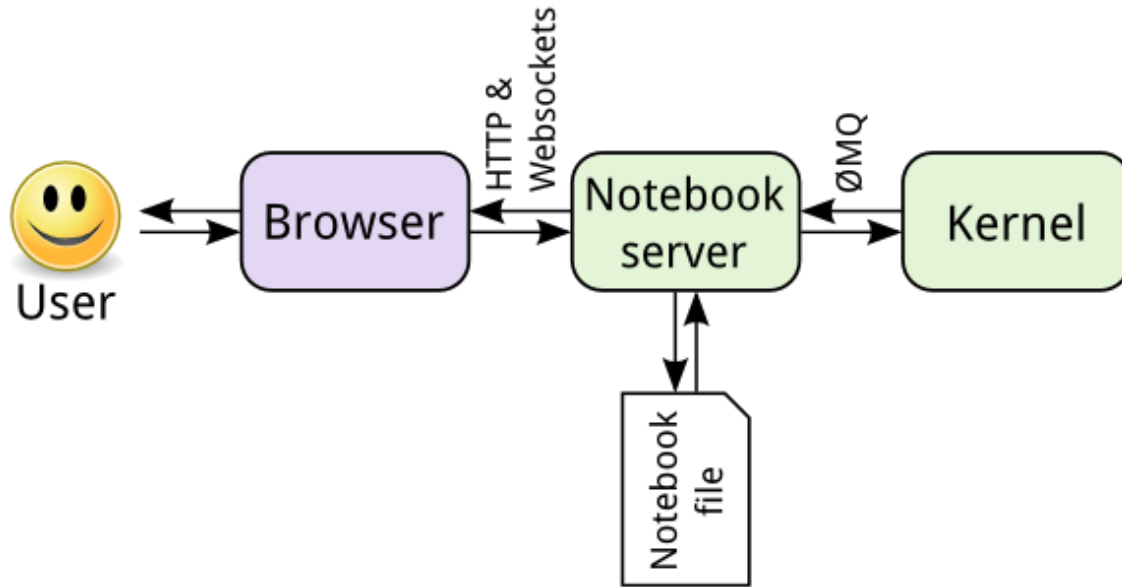
```
In [4]: stem(f, abs(A))
```

```
Out[4]: <Container objec
```

```
In [5]: _.  
add_callback  
baseline  
count()
```



Decoupled Two-process Communication



Project Jupyter

- Project Jupyter is a non-profit, open-source project that will always be 100% open-source software, free for all to use
- It is an ecosystem of a set of software applications and standards
 - [IPython](#) (interactive computing in Python)
 - [Jupyter Server](#) (main server application, server extensions, and related projects)
 - [Jupyter Widgets](#) (interactive widgets for the Jupyter Notebook)
 - [Jupyter Notebook](#) (Jupyter Interactive Notebook)
 - [Jupyter Console](#) (terminal-based console for interactive computing)
 - [Jupyter QtConsole](#) (Qt application for interactive computing with rich output)
 - [JupyterLab](#) (web-based IDE for Jupyter notebooks, code, and data)
 - [JupyterHub](#) (Multi-user server for Jupyter notebooks)
 - [nbviewer](#) (web application to render notebooks as HTML pages)
 - [nbconvert](#) (converts notebooks to other formats)
 - **and more...!**



Open Standards for Interactive Computing

- [Notebook Document Format](#)
 - Open document format based in JSON to contain user sessions including code, narrative text, equations, and rich output
- [Interactive Computing Protocol](#)
 - Open network protocol to communicate with computational kernels based on JSON data over ZMQ and WebSockets.
- [Kernel API](#)
 - Open specifications and API for developing custom kernels

JupyterLab

- JupyterLab is the new web-based user interface for Project Jupyter
- It enables to work with documents and activities (e.g. notebooks, editors, terminals, and other tools in a flexible, integrated, and extensible way
- It can be run on a local desktop requiring no internet access or can be installed on a remote server and accessed through the Internet

The screenshot displays the JupyterLab web interface. The top menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows a file browser with a list of files and folders, including Data.ipynb, Fasta.ipynb, Julia.ipynb, Lorenz.ipynb (selected), R.ipynb, iris.csv, lightning.json, and lorenz.py. The main workspace is divided into several panes:

- Notebook:** Contains text explaining the Lorenz system of differential equations, the equations themselves, and a code cell. The equations are:
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$
- Code Cell:** Contains the following Python code:

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```
- Output View:** Shows a plot of the Lorenz attractor with sliders for parameters sigma (10.00), beta (2.67), and rho (28.00).
- Terminal 1:** Shows the execution of the code cell.
- Console 1:** Shows the output of the code cell, including the Lorenz attractor plot.
- lorencz.py:** Shows the definition of the solve_lorenz function:

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
```







JupyterLab Release History

Initial Release	Release Data	Latest Release	Release Date
4.0.0.a0	27/07/2021	4.0.0.a7	01/09/2021
3.0.0	19/12/2020	3.1.10	01/09/2021
2.0.0	28/02/2020	2.3.2	05/08/2021
1.0.0	28/06/2019	1.2.21	05/08/2021
0.0.1	15/12/2015	0.35.6	05/05/2019

- Source code available at <https://github.com/jupyterlab/jupyterlab>
- **393** contributors, **23 442** releases, **1 865** open issues
- Major version upgrades were challenging for the community
(e.g. some extensions are still not compatible with 3.x)
- Some extensions are still not compatible with JupyterLab
(e.g. [nbgrader](#))

Try Without Installation

- [Try Jupyter](#)
- [JupyterLite](#)
- [Google Colaboratory](#)*
- [Azure Machine Learning Workspace](#)*
- [Amazon SageMaker](#)*
- [Microsoft Planetary Computer](#)*
- [UT Computing Platform](#)*
(not public, i.e. through VPN)
- [ITC Geospatial Computing Platform](#)*

<p>Try Classic Notebook</p>  <p>A tutorial introducing basic features of Jupyter notebooks and the IPython kernel using the classic Jupyter Notebook interface.</p>	<p>Try JupyterLab</p>  <p>JupyterLab is the new interface for Jupyter notebooks and is ready for general use. Give it a try!</p>
<p>Try Jupyter with R</p>  <p>A basic example of using Jupyter with R.</p>	<p>Try Jupyter with C++</p>  <p>A basic example of using Jupyter with C++</p>
<p>Try Jupyter with Julia</p>  <p>A basic example of using Jupyter with Julia.</p>	<p>Try Jupyter with Scheme</p>  <p>Explore the Calysto Scheme programming language, featuring integration with Python</p>

ITC Geospatial Computing Platform



Interactive

Interactive **Jupyter** notebooks provide easy to use and user friendly data analysis and visualization environment.

Powered by **JupyterLab**



Scalable

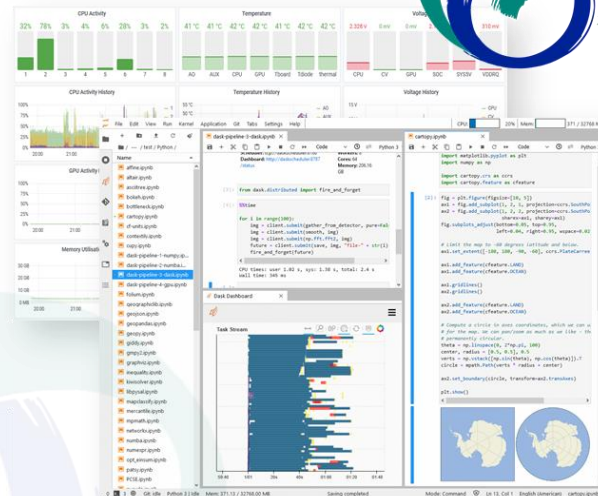
All units in the platform and their **GPUs** are available for distributed out-of-core geospatial data analysis purposes.

Powered by **Dask and Apache Spark**



Multilingual

Multiple programming languages can be used to access platform resources and perform geospatial computations.



Containerized

Each working environment is containerized and isolated from each other and also the host unit to ensure privacy.

Powered by **Docker**



Distributed

Built on a cluster of computing units, the platform scales automatically and balances workload among the units.

Powered by **Docker Swarm**



Replicated

Your assets are protected against hardware failures through replicated storage with minimum two copies.

Powered by **GlusterFS and ZFS**



Powerful

Each computing unit has 8-core ARM v8.2a 64-bit CPU, 512-core Volta GPU with Tensor Cores, and 32GB 256-bit LPDDR4 RAM.

Powered by **Jetson AGX Xavier**



Ready to Use

Software packages are ready to use out-of-the-box, without any further setup required.



Up to Date

Software packages are kept current to allow the use of latest, state of the art features.



Optimized

Software packages are fine-tuned for best performance utilizing high-performance, multi-threaded libraries.



Energy Efficient

Computing units operate at 10–30W ensuring low energy footprint, albeit high performance.



<https://crib.utwente.nl>

Installation

- JupyterLab can be installed using [conda](#), [mamba](#), [pip](#), [pipenv](#), and [Docker](#)
- Basic steps:
 - Install **Python**
 - [Download](#) and run installer
 - Install **node.js**
 - [Download](#) and run installer
 - Install **JupyterLab**
 - [Follow installation notes](#)
 - e.g: `pip install jupyterlab`
- Detailed installation steps for Windows are available at CRIB website: <https://crib.utwente.nl/training/jupyterlab-installation.pdf>

Kernels

- Kernels are processes that run interactive code in a (programming) language and return output
- Kernels also respond to tab completion and introspection requests
- Kernels run until they are terminated (i.e. shutdown)
 - Closing a document or activity does not terminate related running kernel
- Core kernels
 - [IPython](#)
 - [IRKernel](#)
 - [IJulia](#)
- More than 150 [community maintained kernels](#)
 - Some languages have multiple kernels
 - Not all of them are functional (e.g. prototype, not complete)
 - Not all of them have full kernel functionality (e.g. code completion)
 - Some of them are outdated

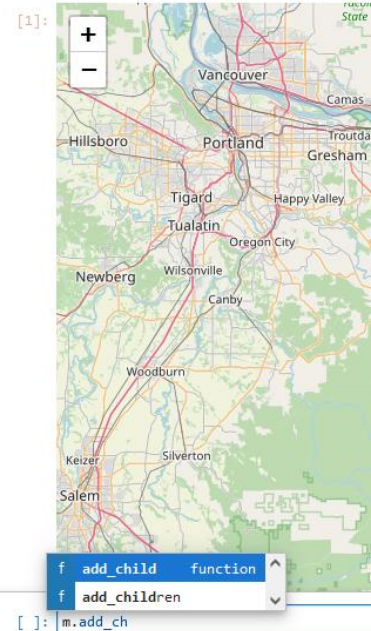


Code Consoles

- Code consoles allow to run code interactively in a kernel
- The cells of a code console show the order in which code was executed in the kernel
- Code consoles can display rich output
- Code consoles support code completion and tooltips
- Code consoles can be connected to running kernels
 - Allows interactively inspect and run code without disturbing other connected clients (e.g. notebooks)
 - Acts as a log of computations in the kernel
- If you close a console it continues to run in the background
 - You can access to the same console again
 - But you lose the console content

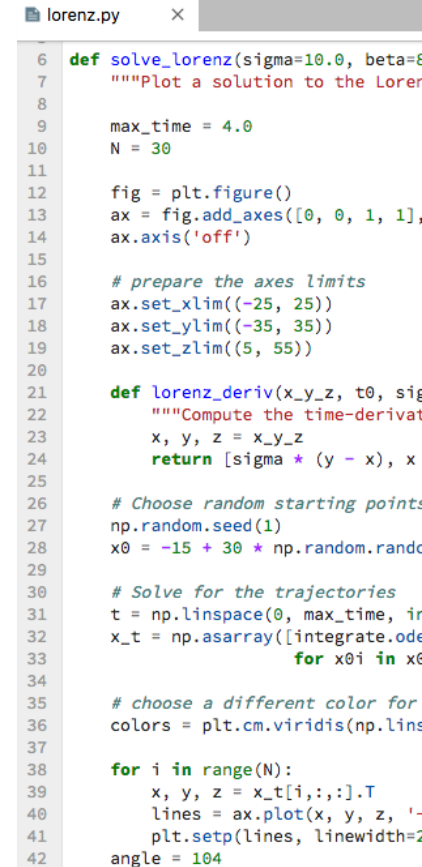
```
Python 3.8.10 (default, Jun 2 2021, 10:49:15)  
Type 'copyright', 'credits' or 'license' for more details  
IPython 7.27.0 -- An enhanced Interactive Python
```

```
[1]: import folium  
  
m = folium.Map(location=[45.5236, -122.6750])  
m
```



Text Editor

- The editor allows editing text files directly and supports:
 - Language-specific syntax highlighting
 - Theming
 - Key maps (e.g. vim, emacs, Sublime Text)
 - Configurable indentation
 - Auto close brackets
- Additional capabilities can be added by extensions (e.g. code completion, spellchecking)
- A file can be open simultaneously in multiple editors and viewers and they will remain in sync
 - Views may not be updated if file content is invalid
- Text files can be connected to code consoles or kernels
 - Code from a text file can be run interactively easily
 - Code blocks are automatically detected and run entirely



```
lorenz.py x
6 def solve_lorenz(sigma=10.0, beta=8
7     """Plot a solution to the Loren
8
9     max_time = 4.0
10    N = 30
11
12    fig = plt.figure()
13    ax = fig.add_axes([0, 0, 1, 1],
14                    ax.axis('off')
15
16    # prepare the axes limits
17    ax.set_xlim((-25, 25))
18    ax.set_ylim((-35, 35))
19    ax.set_zlim((5, 55))
20
21    def lorenz_deriv(x_y_z, t0, sig
22        """Compute the time-derivat
23        x, y, z = x_y_z
24        return [sigma * (y - x), x
25
26    # Choose random starting points
27    np.random.seed(1)
28    x0 = -15 + 30 * np.random.rand
29
30    # Solve for the trajectories
31    t = np.linspace(0, max_time, in
32    x_t = np.asarray([integrate.ode
33                    for x0i in x0
34
35    # choose a different color for
36    colors = plt.cm.viridis(np.lins
37
38    for i in range(N):
39        x, y, z = x_t[i, :, :].T
40        lines = ax.plot(x, y, z, '-
41        plt.setp(lines, linewidth=2
42    angle = 104
```

Terminals

- Jupyter provides terminals for the system shells (e.g. bash, zsh on Linux/Mac and PowerShell on Windows)
- Any text-based program can be run by using a terminal
- You can copy-paste content from/to a terminal
 - Check the documentation for [OS-specific details](#)
- Terminals run with the privileges of your user on the system where Jupyter server is running
 - You may have limited privileges on remote systems
- If you close a terminal it continues to run in the background
 - You can access to the same terminal again
 - But you lose the screen content



The image shows a terminal window with a file browser interface. The window has a title bar with 'Left', 'File', 'Command', 'Options', and 'Right' buttons. The main area displays a list of files and directories under the current directory. The files listed are: ., .astrophy, .cache, .config, .cupy, .dbus, .glue, .gnupg, .grass7, .ipynb_checkpoints, .ipython, .java, .julia, .jupyter, .jupyter-php, .keras, .local, .netlogo, .npm, .rstudio-desktop, .snap, .ssh, .subversion, .virtual_documents, .vnc, .wine32, .wine64, Desktop, /artifacts, /compiled, /fragstats, /globetrotter, /grass, /ilwisdata, /mlruns, /packages, -private, -public, /pysal_data, /qemu, -shared, /tools, .ICEauthority, .Rhistory, and .Xauthority. At the bottom of the terminal, there is a prompt 'UP--DIR' and a hint: 'Hint: Want your plain shell? Press C-o, and get joyvan@2a524fb00da4:~\$'. The bottom status bar shows '1 |elp', '2 |menu', and '3 |view'.

Notebooks

- Jupyter notebooks are documents combining live runnable code with narrative text, equations, images, interactive visuals and other output
- Notebooks are composed of multiple cells with different types (i.e. code, narrative, and raw) and related outputs
- Each notebook is connected to a kernel (e.g. Python, R, Julia) that run code cells and return their outputs
- Cells can be run in any order, but the execution order is displayed explicitly

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
pd.__version__
```

```
Out[1]: '0.24.1'
```

Dataset: Stanford


```
In [2]: # ri stands for Rhode Island
ri = pd.read_csv('poli')
```

```
In [3]: # what does each row represent?
ri.head()
```

```
Out[3]:
```

	stop_date	stop_time	count
0	2005-01-02	01:55	
1	2005-01-18	08:15	
2	2005-01-23	23:15	
3	2005-02-20	17:15	
	2005-02		

Output Formats

- JupyterLab provides a unified architecture for viewing and editing data in different formats
 - For files, the data format is detected by the extension of the file
 - For outputs, the data format is set by Jupyter APIs using MIME types
- Formats supported by default
 - [Markdown](#) (.md)
 - [Images](#) (.bmp .gif .jpeg .png .svg)
 - [Delimiter-separated values](#) (.csv)
 -  Maximum size depends on the web browser (e.g., 1.04GB for Firefox)
 - [JSON](#) (.json)
 - [HTML](#) (.html)
 - [LaTeX*](#) (.tex)
 - [PDF*](#) (.pdf)
 - [Vega/Vega-Lite](#) (.vl .vg.json)
 - [Virtual DOM](#) (.vdom .json)
- Extensions can add support for other formats

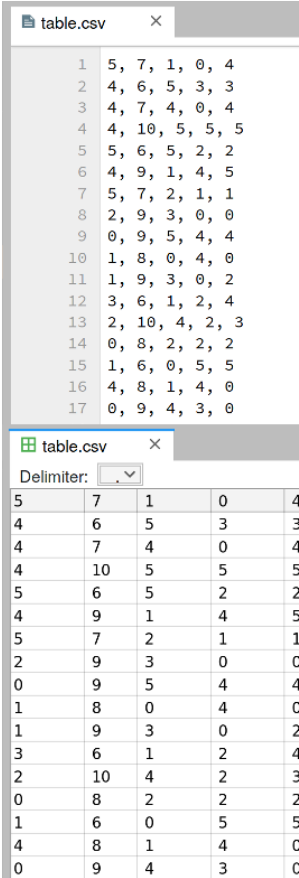


table.csv

```
1 5, 7, 1, 0, 4
2 4, 6, 5, 3, 3
3 4, 7, 4, 0, 4
4 4, 10, 5, 5, 5
5 5, 6, 5, 2, 2
6 4, 9, 1, 4, 5
7 5, 7, 2, 1, 1
8 2, 9, 3, 0, 0
9 0, 9, 5, 4, 4
10 1, 8, 0, 4, 0
11 1, 9, 3, 0, 2
12 3, 6, 1, 2, 4
13 2, 10, 4, 2, 3
14 0, 8, 2, 2, 2
15 1, 6, 0, 5, 5
16 4, 8, 1, 4, 0
17 0, 9, 4, 3, 0
```

table.csv

Delimiter: ,

5	7	1	0	4
4	6	5	3	3
4	7	4	0	4
4	10	5	5	5
5	6	5	2	2
4	9	1	4	5
5	7	2	1	1
2	9	3	0	0
0	9	5	4	4
1	8	0	4	0
1	9	3	0	2
3	6	1	2	4
2	10	4	2	3
0	8	2	2	2
1	6	0	5	5
4	8	1	4	0
0	9	4	3	0

JupyterLab Interface

The screenshot displays the JupyterLab interface with ten numbered callouts pointing to specific components:

- 1. Main menu (File, Edit, View, Run, Kernel, Application, Git, Diagram, Tabs, Settings, Help)
- 2. Right sidebar (Navigation icons)
- 3. Left sidebar (File browser showing a directory tree)
- 4. Status bar (Simple, 0, 4, Fully initialized, Python | Idle, Mode: Command, Ln 1, Col 1, English (United States), 03.12-Performance-Eval-and-Query.ipynb)
- 5. File browser (Search and list view of files)
- 6. Launcher (Grid of application icons: Notebook, Python, Bash, Go, Intel Assembly, Java, JavaScript, Julia)
- 7. Notebook (Code editor with a Python notebook titled 'Motivating query() and eval(): Compound Expressions')
- 8. Code Console (Terminal window showing Python code execution and output)
- 9. Log Console (Log viewer showing execution logs)
- 10. Command Palette (Searchable menu for actions)

1. Main menu
2. Right sidebar

3. Left sidebar
4. Status bar

5. File browser
6. Launcher

7. Notebook
8. Code Console

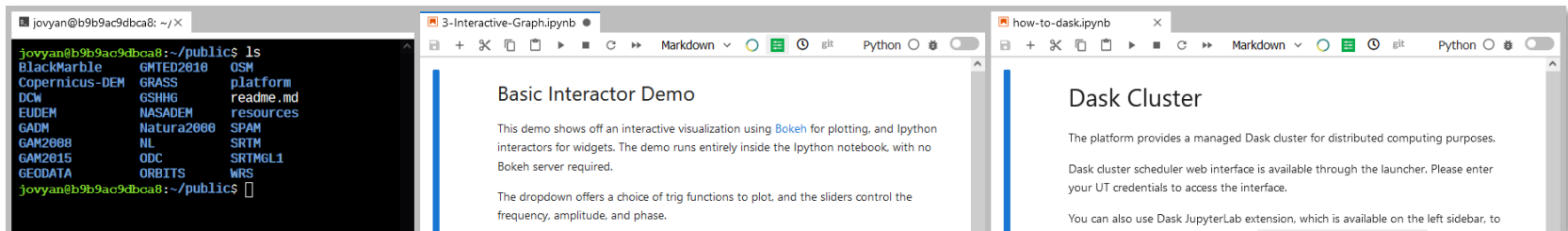
9. Log Console
10. Command Palette

JupyterLab Interface


- Menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, Help menus
 - Extensions may create additional menus
- Left sidebar contains common tasks (e.g. file browser, list of running kernels, etc.)
 - Extensions may create additional sidebar tabs
- Main work area allows to arrange documents and activities (e.g., terminals, code consoles) into panels that can be resized and subdivided
 - Simple mode toggles single- and multi-activity layouts
- Context menu can be accessed by right mouse click
 - Native context menu can be accessed by `Shift` + right mouse click
- Keyboard shortcuts can be used to perform actions
 - Command palette can be accessed by `Ctrl` + `Shift` + `C`
- User interface language can be set by the user (new in JupyterLab 3.0)
 - Language pack of the language should be installed to display a language

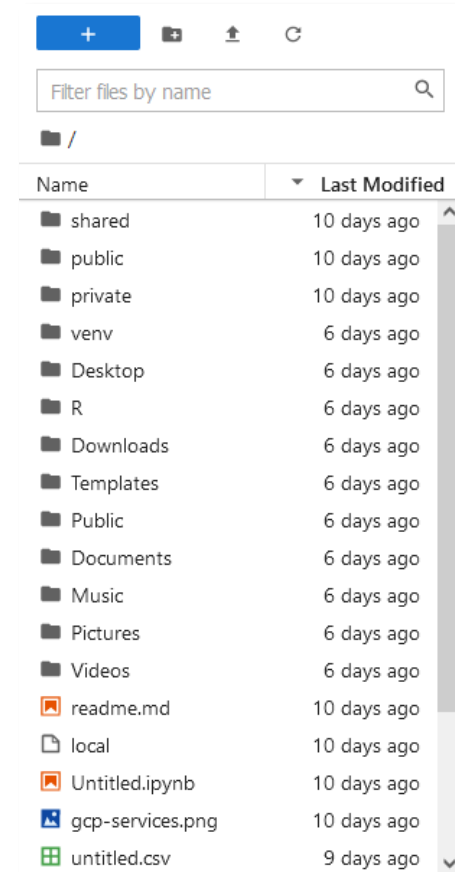
Workspaces

- Each JupyterLab session resides in a workspace that contains its state (e.g. open files, layout of the open tabs, etc.)
- Terminals, code consoles, text editors, and notebooks can be layout
- Workspaces can be saved and restored another time
 - Default workspace is saved and restored automatically
- Workspaces can be shared between multiple users or browsers as long as they have access to the same server
- A workspace should only be open in a single browser tab at a time
 - Simultaneous use results in a new workspace assignment




File Management

- File browser allows opening, creating, deleting, copying, renaming, downloading, and sharing files and directories
- The directory you start JupyterLab matters; it becomes the root folder
- Directory listing is updated automatically if changes occur
- Ease of file management depends on deployment type
 - Local JupyterLab: easy file handling, difficult file sharing
 - Remote JupyterLab: difficult file handling, easy file sharing
 - Upload and downloads are performed through the browser
 - Some extension may facilitate file management
- When you create a file it is stored by using a default name
- Checkpoints are created for notebook files
 -  Hidden checkpoint directories may pollute your system



Sharing Notebooks

- Notebook files are [JSON](#) files saved with the `.ipynb` extension
- They can be displayed or run using Jupyter or other compatible software
 - Not all output might be displayed properly (e.g. interactive widgets)
 -  **Check your notebook before sharing by opening it in another session**
 - Dependencies (i.e. packages, extensions) should be available to run them
 - External assets (e.g. files, datasets) should be shared separately
- They can be exported into other file formats

• AsciiDoc	<code>.asciidoc</code>	• PDF	<code>.pdf</code>
• HTML	<code>.html</code>	• ReStructured Text	<code>.rst</code>
• LaTeX	<code>.tex</code>	• Executable Script	<code>.py</code>
• Markdown	<code>.md</code>	• Reveal.js Slides	<code>.html</code>
- Exporting options depend on [nbconvert](#) configuration
 - Some formats may require additional software packages
 - Some formats may allow customization by using cell metadata

Debugger (new in JupyterLab 3.0*)

- Notebooks, code consoles, and files can be [debugged directly](#) from JupyterLab
- A kernel with support for debugging is required for the debugger to be enabled and visible
- Kernels should implement [Jupyter Debugger Protocol](#) to support debugging
- Kernels with debugging support
 - [xeus-python](#)
 - [xeus-robot](#)
 - [ipykernel](#)

The screenshot displays the JupyterLab debugger interface with four main sections:

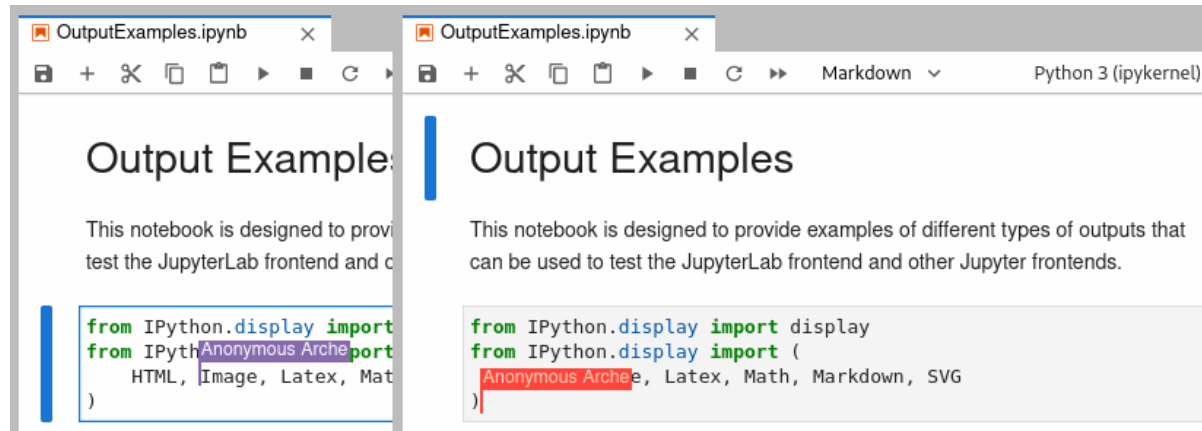
- VARIABLES:** Shows local variables with values: `Locals: 2`, `a: 1`, and `b: 2`.
- CALLSTACK:** Shows the current call stack with the active frame highlighted: `add at :2` and `<module> at :1`.
- BREAKPOINTS:** Shows a single breakpoint set at `/tmp/xpython_18660/2114632017.py`.
- SOURCE:** Shows the source code of the file `/tmp/xpython_18660/2114632017.py` with the following content:

```
1 def add(a, b):  
2     res = a + b  
3     return res
```

The line `res = a + b` is highlighted in orange, indicating the current execution point.

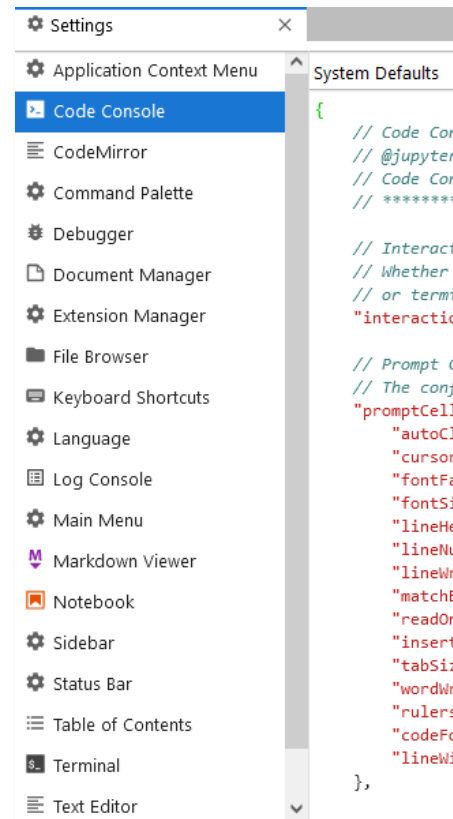
Real Time Collaboration (new in JupyterLab 3.1)

- Documents and notebooks have collaborative editing feature enabling collaboration in real-time between multiple clients **without user roles**
- Other users have access to the same environment you are working on
- Editors are not collaborative by default; to activate it, start JupyterLab with the `--collaborative` flag
- This feature is not well developed yet



User Settings

- Some settings can be changed by using menu items (e.g. main theme, text editor indentation, text editor key map)
- All settings can be changed by using **Advanced Settings Editor** (under Settings menu)
 - System defaults (Read-only JSON with comments)
 - User preferences (JSON)
- Extensions may also have settings which can be modified by the editor
- Some changes may require restart (e.g. logout / login) (usually the case with extensions)
- See JupyterLab Documentation for more information on [user settings directory and files](#)



Extensions

- JupyterLab is designed as an extensible environment
- Extensions can customize or enhance any part of JupyterLab (e.g. new themes, file viewers, editors, or renderers for outputs)
- Types of extensions
 - Source extension (requires a rebuild of JupyterLab)
 - Prebuilt extension (does not require a rebuild of JupyterLab)
- Installation methods
 - Package managers (e.g. pip, conda)
 - [Extension Manager](#) in JupyterLab
 - [jupyter labextension install](#) command
 - Some extensions may require additional software packages
 - ⚠ **Check extension documentation before installation**
 - ⚠ **Restart JupyterLab after installation**

DISCOVER



[@jupyterlab/apputils](#)

JupyterLab - Application Utili

Install



[@jupyter-widgets/jupyterlab](#)

The JupyterLab extension pr

Install



[@jupyterlab/translation](#)

JupyterLab - Translation ser

Install



[@jupyterlab/git](#)

A JupyterLab extension for v

Install



[@jupyterlab/geojson-exten](#)

GeoJSON renderer for Jupy

Install



[@jupyterlab/fasta-extension](#)

Fasta renderer for JupyterLa

Install

Popular Extensions



Search

Name	Description
dask-labextension	JupyterLab extension for Dask
jupyter-archive	Jupyter/JupyterLab extension to create, download and extract archive files
jupyterlab-tensorboard	Tensorboard extension for JupyterLab
jupyterlab-chart-editor	JupyterLab extension for Plotly's react chart editor
jupyterlab-drawio	A standalone embedding of the drawio / mxgraph package into JupyterLab
jupyterlab-git	A Git extension for JupyterLab
jupyterlab-lsp	Coding assistance for JupyterLab using Language Server Protocol
jupyterlab-spellchecker	Spellchecker for JupyterLab notebook markdown cells and file editor
jupyterlab-spreadsheet-editor	JupyterLab spreadsheet editor for tabular data (e.g. csv, tsv)
jupyterlab-proxy-gui	A JupyterLab extension to control the configurable-http-proxy of JupyterHub

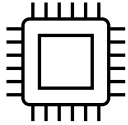
⚠ Check [npm repository](#) for more by using the query "keywords:jupyterlab-extension"

Further Resources

- <https://jupyter.org/>
- [Jupyter Project Documentation](#)
- [Jupyter Notebook Documentation](#)
- [JupyterLab Documentation](#)
- [IPython Documentation](#) 🧐
- [Jupyter Client Documentation](#) 🧐
- [Awesome Jupyter](#)
A curated list of awesome Jupyter projects, libraries and resources
- [Jupyter Discourse](#)
A Discourse Forum for a multitude of Jupyter topics
- [Jupyter GitHub](#) 🧐
A place where the community collaborates on the development of Jupyter software
- [Contribute to Jupyter](#)
It doesn't matter if you are new, Jupyter community welcomes you!



Contact CRIB



<https://crib.utwente.nl>



<https://itc.nl/big-geodata>



crib-itc@utwente.nl



[@BigGeodata](https://twitter.com/BigGeodata)



[Big Geodata Newsletter](#)



[Big Geodata Channel](#)

