Introduction to Jupyter Hub- Basic Steps

Content

1. Login	2
2 Server Ontions	з
2. The Weylinks of Merline Frankrey and	
3. The workplace/working Environment	4
4. Working with Jupyter Hub	8
5. Help & Tutorials	13
6. Closing Jupyter Hub	14
7. BONUS	16
8. Further References	17

1. Login

https://jupyter.zmml.uni-bremen.de

Shibboleth Login: Enter your username and password you use for university online services



Figure 1: Sign in button (left) will lead to the shibboleth login from the Zentrum für Netze Uni Bremen (right)

2. Server Options

This allows choosing a profile with different programming languages, so-called kernels

	Server Options
۲	Minimal environment To avoid too much bells and whistles: Python.
0	Java/IJava ijava-jupyter-stack is a Jupyter Docker Stack image that's including the IJAVA Kernel to Jupyter notebook.
0	Datascience environment Python, R, Julia, C++ and Swift, some with code completion.
0	Tensorflow Jupyter Notebook Deep Learning Stack.
0	try-GMT Workspace with pre-installed GMT, pygmt
0	Multikernel R & Python Workspace with R & Python

Start

Figure 2: Choice of the different kernel options

3. The Workplace/Working Environment

Launcher

After setting the kernel options a launcher will open. The launcher allows choosing between Notebook or Console view or other file options. Further, different modules can be chosen that contain different languages /environments (e.g., SoS allows using different languages in one script).

A new launcher can be opened by the big blue plus sign (Fig. 3, blue arrows).



Figure 3: Left Sidebar / Workspace Overview. The launcher with different module options is found in the center. Blue arrows show the icons for starting a new launcher.

Left Sidebar/ Workspace Overview (Fig. 3)

On the left sidebar there are several workspace overviews options available:

1) File browser (folder icon):

shows last accessed folders, uploading documents (see Chap.3), creating your own folders

2) Running terminals and kernels (stop button):

shows open tabs, open kernels or terminals

3) Table of contents (list icon):

shows code and markdown cells in a script and allows to toggle between them

Launcher > Console (Fig.4)

A code can be run interactively in a kernel. The cells will appear in the order in which the code was executed.

Cells with codes can be deleted by right-clicking on the console window > "Clear Console Cells".



Figure 4: An example of the console

Launcher > Notebook (Fig.5)

A notebook environment allows the use of interactive elements. Further markdown (text) and code cells can be used together in one notebook. This can help the comprehensibility of the code through e.g., adding text explanations to the code.



Figure 5: An example of the Notebook

Launcher > Combining Notebook & Console (Fig.6)

The notebook and console can be used together. For example, when working with a notebook under "*file > new console for notebook*" an additional console can be started, which is linked to the code in the notebook.



Figure 6: Example of Notebook and Console combination. After defining the variables in the notebook, calculations with these can be done in the console

Debugger / Variable Overview (Fig.7)

Via "enable debugger" (bug icon), variables can be found either in table or tree view and callstack, breakpoints, and (kernel) sources are listed



Figure 7: The bug icon opens a tab on the right side, where defined variables are shown (blue circles)

Changing the server option

The server option can be changed by *"File > Hub Control Panel > Stop my Server > Start my Server"* This will close the running kernels and allows changing to a new server option.

4. Working with Jupyter Hub

Useful Shortcuts:

- Tab: Autocompletion, suggestion of functions/variables
- Shift & Tab: Shows documentation
- Shift & Enter: Execution of a cell
 (This shortcut can be changed under "settings > console
 - run keystroke")
- Double click on a cell: Shows the original formatting

Loading in Documents

Documents can be uploaded from a device either by the folder icon or directly by drag & drop (Fig.8)



Figure 8: Uploading documents via icon or drag & drop into the workspace (see blue arrows)

Setting Cell Format

A cell can either be marked as code, markdown or raw. This can be done via a dropdown menu in the top bar (Fig.9). Markdown allows including and formatting text paragraphs. The cell which is currently modified is visible by the blue mark on the left side.

Ø	Lau	ncher				×		Untit	ed1.ipynb		×	+														
8	-	+ %	D	Ľ	►		C	**	Code	~											ĕ	Ру	tho	n 3 ((ipyk	ernel)
I		[1]:	a=3						- Code											Þ	\uparrow	\downarrow	,	÷	Ŧ	Î
		[2]:	b=8						Markdowr	n																
		[3]:	a+b						Raw		J															
		[3]:	11																							
		[]:																								

Figure 9: Drop down menu at the top bar of the launcher allows choosing a cell format (code, markdown, raw)

Inserting / Changing Cells

General:

[] = an empty bracket shows that the cell has not been executed yet

[1] = the number in the brackets shows the order of execution

* = code in the cell is still running. If the cell is running too long, there might be an infinite loop or a disruption to the kernel connection. You may consider to interrupt the process then.

In Notebook (Fig.10)

Cells can be modified by clicking on the cell itself.

The menu bar on top of the notebook allows modifying, stopping an starting the kernel. Note: All variables saved in the workspace will be deleted when restarting the kernel.

Cells can be splitted or merged via "Edit > Split Cell" or "Edit > Merge Cells"

Add Copy Run Restart Save Cut Paste Interrupt kernel Restart & re-run whole kernel *	
🖻 Unt tled.ipynb 🛛 🥆 🕂	
	 Python 3 (ipykernel)
[]:	
[]:	
[]:	
	copy cell &MoveAddDeletepaste it belowcellcellcell

Figure 10: Options for changing a cell or kernel within the Notebook. * (order of cell execution is set to [] zero)

In Console:

Cells can only be directly executed via "Shift & Enter" or "Run > Run Cell" in the general menu bar

Extensions & Widgets

Enable Extensions (Fig.11)

To use these widgets/applications, the extensions need to be enabled under "*Extension manager – Enable*".

After enabling a list of already installed extensions can be found and the option to browse through extensions. Extensions can be e.g., widgets, connection to git, or python libraries.



Figure 11: Blue arrows show the buttons to enable widgets

Ipywidgets (Fig.12)

In Jupyter Hub widgets can be used. Those are interactive graphical user interfaces such as scrollbars/sliders or toggle-buttons.

The widgets can be used in Python or C++ kernels.

The widgets can be installed via the *"extension manager"* (Fig) or using the command *"pip install ipywidgets"*



Figure 12: Example of including a slider in a Notepad

Maps (Fig.13)

Next to widgets, interactive map can be created. For this load the ipyleaflet library via executing "from ipyleaflet import Map" in one of the code cells or commando lines



Figure 13: Example of including an interactive map using ipyleaflet

see more under 7. BONUS

Saving & Export

Under "File" the workspace can be saved or exported to e.g., a PDF; HTML; LaTex file. The files/kernels are currently automatically saved when logging out. However, it is recommended to do a safety copy.

5. Help & Tutorials

Tutorials & References

Under the tab "*help*" certain "*references*" (e.g., Python Reference, NumPy Reference) can be found. These are a link to general information & tutorials about the specific programming language / program library used.

Contextual help (Fig.14)

Under the tab "*help > show contextual help*" a widget similar to a tooltip can be found. A separate window is opened, which shows general information about the content of a cell (e.g., the datatype & how to use it)



Figure 14. Example of the contextual help (right) referring to cell [10] in the notebook (left)

6. Closing Jupyter Hub

There are two options to log out:

Closing the kernel & Log Out (Fig.15, 16)

Go to: "File > Hub Control Panel > Stop my Server"

Then:" logout button" (top right corner)

This stops the server and allows to change the server option (see Fig.16, Step 3). If you want to select a different server option when logging in next time, it is currently needed to stop the server via Hub Control Panel after the last session and then logout. If you do not want to logout and want to return to your kernel, choose "my server" in the Hub Control Panel.



Figure 16: First, stop the server (3), second logout in order to select a new/different server option when logging in next time

<u>Log out</u>

Go to: "File > Log Out"

When logging out directly without stopping the kernel, the kernel & workspace will be reloaded when logging in next time.

7. BONUS

MAGICS

So-called magics are special commands that are on available in Jupyter. They support an easier work with data and address features specific for one language.

General:

% = applies magic to single-line command
%% = applies magic to an entire cell
Some magics can only be run with one % option

Examples:

%Ismagic will list all available magic commands %run <file_name> will run a python file that is included in Jupyter Notebook %%time will time the execution of the code %reset will delete defined variables %%R will allow using R-code in a Python environment

Python Libraries

Install additional software with "!pip-install" or via the "extension manager"

The most common and maybe useful libraries are:

- Numpy (http://www.numpy.org/) is a fundamental library for numerical and scientific computing with Python. It contains data structures for numerical arrays, tools for linear algebra, random number capabilities, and much more.
- SciPy (https://docs.scipy.org/) offers a varied set of functions for scientific computing, such as optimization, interpolation, statistics and signal processing. It also includes fundamental constants from many disciplines such as the speed of light as well as data structures for sparse matrices.
- Matplotlib (https://matplotlib.org/) is the core plotting library for Python and can be used inline in the notebook with the %matplotlib notebook or %matplotlib inline cell magics.
- Pandas (https://pandas.pydata.org/) provides resources for data analysis and a flexible data structures for labeled tabular data.

From Barba et al. 2019

8. Further References

Barba, L. A. et al. (2019). Teaching and Learning with Jupyter. Accessible via: https://jupyter4edu.github.io/jupyter-edu-book/