**Jupyter Notebook – features & benefits**

From my short testing with Jupyter Notebook I came to a conclusion that it is a useful tool at running code “interactively” within a web browser. Because of using markdown text to describe the data that we’re looking at jn a Jupyter Notebook, it reminded me a lot of writing README files to GitLab repositories.

For someone who’s not from a programming background, a JN could be really useful in explaining how some results have been gotten and in reproducing those same results. Since JNs are dynamic visualizations, a user can often times change variables at will and see how they affect the results.

I bet a JN could be useful in teaching an order of events in a Python script, since every cell that contains code, can be run separately without running the entire program every time. That way there’s no need to either delete or comment out some existing methods. It is also possible to edit the execution order without cutting and pasting lines of code here and there by just dragging and reordering different cells in the Jupyter GUI.

Because of the freedom of ticking which cells to run in a JN, it could also be used for optimizing code performance and seeing first-hand which part of your code is slowing your whole program down. And that’s not all, since I think that while many non-programmers that are just starting out with Python would probably like to learn a few things about charts, they have the freedom of running only the cells containing the necessary code for generating the data for the chart and leaving everything else alone when it’s not needed.

Even if a JN creator isn’t that familiar with markdown text, they can use so-called “magic commands” to display the data they want in HTML form. These magic commands work either per line or per cell so again there’s the freedom of cherry-picking which parts of your code you would like to test as a HTML cell for example. A JN has multiple export options so that even when by nature JNs are interactive and dynamic, they can be exported as PDF for example, which could be useful if someone wanted a version of a notebook in which the data can’t be tampered with.

When working with Pandas and data analysis in Python, it’s generally a good idea to at least try out a Jupyter Notebook when visualizing data and mainly dataframes in case of Pandas. It’s not mandatory but using a JN makes the rows and columns much more readable than trying to get a grasp of them all in an editor and maybe using CSV reader like MS Excel side-by-side with the editor.

A kernel is needed when creating a JN and in this case, picking a kernel basically means picking a programming language that’s used as a basis of a JN when writing and running code. I see this pretty useful since this means that you could e.g switch from a Python version to another just by clicking a button, provided you have that another version of Python installed. Maybe you want to learn more about the inner workings of Python and run some code cells using Python 2? That and more is possible when using multiple kernels.

One last useful aspect of using JN that comes to mind is the fact that for some people who would like to learn useful algorithms and understand how to code some mathematical functions, the interactivity of using a JN helps a lot when you can try writing a formula and see results of that writing almost in real time.