Python, Jupyter Notebooks and Jupyter Hub in a teaching setting

Method · September 2020

DOI: 10.13140/RG.2.2.31928.57605

1 author:

Jana Lasser
Complexity Science Hub Vienna
16 PUBLICATIONS 39 CITATIONS
SEE PROFILE

Python, Jupyter Notebooks and Jupyter Hub in a teaching setting

In the lecture described in the main article, we use Python in combination with Jupyter Notebooks and a Jupyter Hub as a technical infrastructure to (a) teach students how to program and (b) work on data science applications. In the following supplement, we will detail the technical implementation and setup of this teaching environment and how the three components - Python, Jupyter Notebooks and the Jupyter Hub - worked together in practice in our lecture.

Python

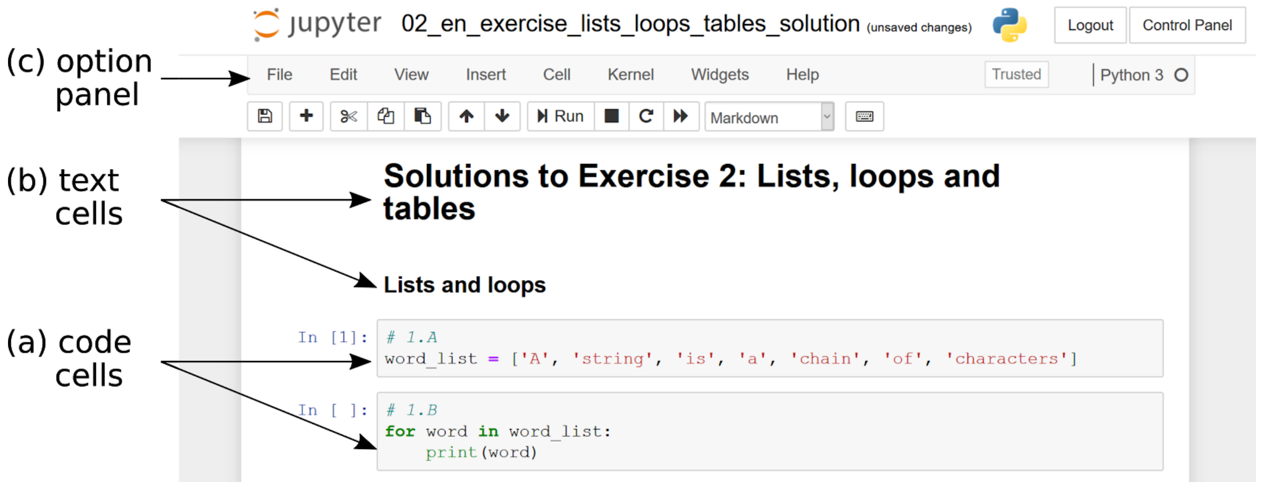
Python is a programming language that is widely used in research and industry. Two main properties make it a fitting candidate for use in a teaching environment: (1) Python is an interpreted language, allowing for quick prototyping and feedback without an additional compilation step. (2) the syntax is relatively close to the English language and allows for an intuitive understanding of most of the code.
In the described language, we use Python 3.7.3, together with three widely used third-party libraries:

- The library Pandas supplies the basic “data frame” class, which is used for data storage and manipulation.
- The library Matplotlib is used for all visualization tasks.
- SciPy is used for statistical analysis tasks such as regression

All of these libraries are well-documented and well-maintained and available through standard package distribution platforms for Python such as PyPi and Anaconda.

Jupyter Notebook

While an easy-to-learn programming language is a crucial ingredient for a successful entry into the world of programming, a suitable editor is very important as well. For the purposes of our lecture, we relied on Jupyter Notebooks for that purpose. Jupyter Notebooks can contain programming code as well as structured text and can display figures within the document. The documents under the hood are composed of HTML and can therefore be displayed by any modern browser. On a higher level, a Jupyter Notebook is composed of *cells.* Cells can contain code that can be *run* to execute the code and display any output such as prints to the command line or plots (see fig. 1 (a)). Cells can also contain text that can be formatted using Markdown (fig. 1 (b)). As a programming environment, Jupyter Notebooks are easy to use for beginners. They do not contain too much complex functionality (such as for example Eclipse) and all necessary functionality can also be executed using the mouse and interactive options at the top of the document (fig. 1 (c)). This makes it easier for beginners to continue practising, even if they have not memorized all handy keyboard shortcuts yet. Nevertheless, Jupyter Notebooks are extensible through plugins and are widely used in production environments both in research and industry. Moreover, Jupyter Notebooks can interface with different programming languages via their *Kernel.* Next to a Python kernel, Jupyter Notebooks have kernels for R, scala, haskell and SAS (full kernel list here). Therefore they are not purely educational software and learning how to work with Jupyter Notebooks is a valuable skill for a future career.



Solutions to Exercise 2: Lists, loops and tables

Lists and loops

```
In [1]: # 1.A
        word_list = ['A', 'string', 'is', 'a', 'chain', 'of', 'characters']

In [ ]: # 1.B
        for word in word_list:
            print(word)
```

Figure 1. A Jupyter Notebook document including code cells (a), text cells (b) and an options panel that allows for the execution of action using a mouse.

Creating high-quality educational content including figures, videos, links, lists, headings and tables in Jupyter Notebooks is straight forward as text can be formatted using the markdown text markup language. Next to the use of Jupyter Notebooks as programming environments for the students, we also use Jupyter Notebooks to create content for the lecture and formulate the exercise sheets (see openly available exercise sheets). This has the added benefit of exposing students to the software in many different contexts of the lecture. Using and reading Jupyter Notebooks quickly becomes easy and natural for them. In addition, students do not need to handle multiple file formats or platforms to access all lecture content. Furthermore, it becomes easy to supply code snippets either in the lecture or exercise sheets that students can re-use and expand to work on case studies.

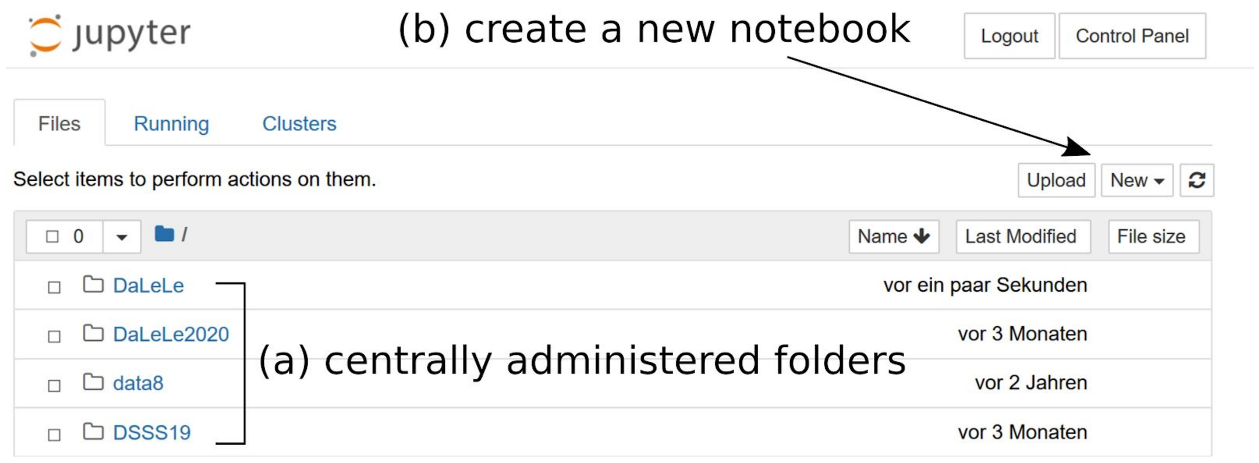
Jupyter Hub

A common problem in many lectures involving programming is the setup of the programming environment, i.e. the installation of the programming language and the editor. Setting up the programming environment on diverse hardware with diverse operating systems is time-consuming and requires deep technical knowledge from the lecture tutors and assistants. This is especially true in settings where no standardized university hardware such as PCs in a computer room are used for the hands-on parts of the lecture. Our lecture relied on the students to bring their own devices. This had both practical and educational reasons. Firstly, we encouraged students to bring laptops to the lecture to follow the live-coding on their own machines and the university did not have a scheme to borrow laptops. Secondly, the lecture is aimed at teaching the students workflows that are as close to a professional setting as possible. In this context, it makes sense to ask the students to use their own devices, as these will be their tools for any application outside the scope of the lecture as well.

To remedy this problem, we made use of a Jupyter Hub. The Jupyter Hub is a centralised installation of the Jupyter Notebook software on a university server. Students can log into the hub using their university access credentials and create and use Jupyter Notebooks there. The hub has a centrally administrated installation of Python and all necessary third-party packages as well as central read-only storage accessible to all students. This way, exercise sheets and data can be supplied and administered centrally and are accessible to all students at the same time (see fig. 2 (a)). Nevertheless, students also have their own persistent storage on the hub and can install additional third-party packages locally. Files can of course also be downloaded from the Hub and stored on local machines.

To set up the Jupyter Hub for our university we worked with our IT department which adapted existing installation guides to connect the hub to the university's identity and data management systems. To date, the Jupyter Hub is used in many other lectures at our university. It has been expanded to enable access to the university's high-performance computing facility, demonstrating the use of Jupyter Notebooks in a professional research context. By now Jupyter Hub installations are also available from vendors such as Google's colab and AWS. An important learning from the three years in which we have used the Jupyter Hub installation for teaching is the necessary virtualization and monitoring of computing resources for the Hub. Load on the Hub can spike quickly when 100 students try to download and process a large data set and a single inexperienced programmer can jeopardize the whole installation if they are allowed to use all available RAM. Therefore active load balancing and automated functionality to shut down notebooks that consume too many resources (for example through an infinite while-loop) are advisable.

In practice, we uploaded Jupyter Notebooks containing the exercises to the central storage of the Hub every week. Exercise sheets contained text instructions and code snippets intended to be used as a basis to solve the exercises. Students logged into the Hub either from home or during the tutorials to work on the exercises. They then created their own notebook (fig. 2 (b)), into which they could copy code snippets from the exercise sheets, to work on the exercises. One week after we deployed an exercise sheet, we also deployed a full solution to the exercise. This was necessary since exercises were partly designed as a continuing case study, building on exercises from the previous week(s). Providing solutions enabled students to follow the exercises, even if they had missed the previous week.



Select items to perform actions on them.

Name	Last Modified
DaLeLe	vor ein paar Sekunden
DaLeLe2020	vor 3 Monaten
data8	vor 2 Jahren
DSSS19	vor 3 Monaten

Figure 2. Landing page of the Jupyter Hub with (a) centrally administered folders containing exercises and lecture materials. New notebooks can be created via the "new" button (b) on the top right.