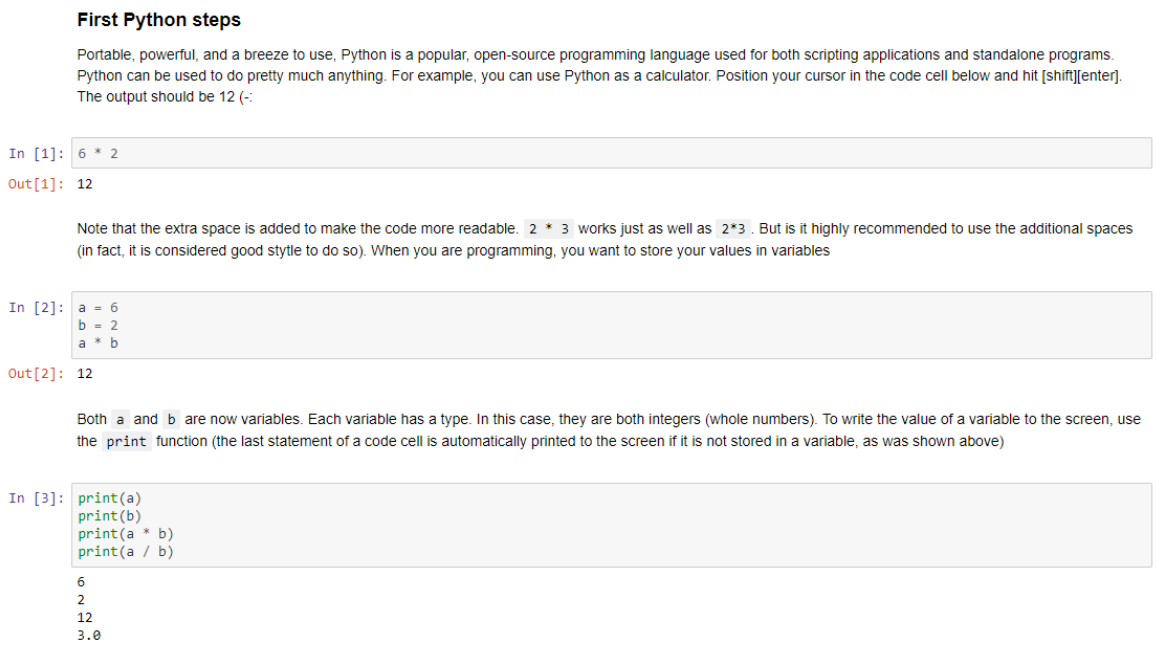
What are Notebooks?

I'll admit, the first time I heard the term computational notebook I had a mental image of a small laptop. I was, of course, wrong. Computational notebooks are a much more useful tool for both Learning and for Teaching. Specifically, I am speaking about Jupyter notebooks which are live documents that allow for interactive computing in several languages: Python, R, Julia, Sage, and more. These documents are broken up into cells which can contain code, markdown text or mathematical expressions (via LaTeX). These cells can then be run to compute the output within the document, no need to compile. Due to this ability to immediately see the output notebooks allow learners to play around with code, adapting or adding elements and observing the outcome. Rather than carrying on trying to explain this I've added a snippet below so you can get more of a feel for it. You can also head to the Jupyter page to try it out in your browser.



First Python steps

Portable, powerful, and a breeze to use, Python is a popular, open-source programming language used for both scripting applications and standalone programs. Python can be used to do pretty much anything. For example, you can use Python as a calculator. Position your cursor in the code cell below and hit [shift][enter]. The output should be 12 (-:

```
In [1]: 6 * 2
Out[1]: 12
```

Note that the extra space is added to make the code more readable. `2 * 3` works just as well as `2*3`. But is it highly recommended to use the additional spaces (in fact, it is considered good stytle to do so). When you are programming, you want to store your values in variables

```
In [2]: a = 6
        b = 2
        a * b
Out[2]: 12
```

Both `a` and `b` are now variables. Each variable has a type. In this case, they are both integers (whole numbers). To write the value of a variable to the screen, use the `print` function (the last statement of a code cell is automatically printed to the screen if it is not stored in a variable, as was shown above)

```
In [3]: print(a)
        print(b)
        print(a * b)
        print(a / b)
6
2
12
3.0
```

Being able to have text cells alongside code means that you can add context and create a computational narrative which is a powerful learning tool. A few years ago I started learning to write code, just basic JavaScript and some Ruby, the problem I had at the time was not being able to understand why I was doing what I was doing. With a computational narrative, you can explain and give context alongside the exercise which helps to reinforce the concept.

As you can see this is a very powerful tool for teaching both the mechanisms and the theory required for programming but there are even more good bits! Jupyter can be installed to run on your own machine but the really exciting part is that these document can be run online. By using a hosting service, such as Noteable, you can use notebooks without having to install anything on your own machine. Anyone who has tried to run a workshop before will know that being able to remove the need to pre-install software from your learner's machines is a godsend. This removal of one of the key barriers allows learners to engage more directly with the source material. With a little ingenuity, this can also become a very versatile tool. You could use a blank document to show the process of programming to a class, you could then share that document and allow learners to alter or add cells, you could present learners with a 'broken' cell and challenge them to fix it.

Have a look at this list on Github for more varied examples of what can be done.

If you use notebooks already then drop a comment below to say how you use them. If you are interested in using them at the University of Edinburgh then get in touch.