Getting prepared

Tools

The tools used in this course (Spyder and the Jupyter Notebook server) can be found in the Anaconda distribution. Once you have installed this distribution (which should already be present on your company notebook), you can start these tools by:

- selecting the individual tool from the Anaconda3 menu in the Start menu,
- going to the Anaconda Navigator and use it to start them,
- selecting the Anaconda Prompt in the Anaconda3 menu.

  The last option opens a Command Line window with the proper environment setting. In there you can cd to the folder in which you have placed our notebooks and can start the Jupyter Notebook there by typing: jupyter notebook . If you are being asked for a token, check this Anaconda Prompt window for it.

Course notes

We will be using two notebooks:

For a quick overview of Python: https://github.com/jakevdp/WhirlwindTourOfPython

For an intro to the packages we will use (Numpy, Pandas, Matplotlib):

https://github.com/jakevdp/PythonDataScienceHandbook

Both of these are also available as pdf's:
http://www.oreilly.com/programming/free/a-whirlwind-tour-of-python.csp
https://jakevdp.github.io/PythonDataScienceHandbook

The DataScience book is only a year old, but things move fast in the Python world. Here you can find the edited version as used in my lectures.

Using the tools

Even though the tools are basic and generally easy to use, it will take some time to get used to them. The first chapters (notebooks) of the DataScience Handbook provide some basic information. I would read those first and then take your time to explore both IPython and the notebooks themselves. You may also want to compare using plain Python versus Ipython.

Debugging

You can write your Python scripts in any text editor and then keep running and modifying them until they no longer produce errors and actually deliver the results you were aiming for. Syntactic errors are rarely hard to find, but semantic errors (the script runs, but gives unexpected results or crashes for certain inputs) are often much harder to isolate. The most basic tool at hand is a rich sprinkling of print statements at points of interest in your code.

A debugger is a tool that basically makes this an easier and interactive process: you get to put breakpoints in your code, which when reached will cause execution to halt and allow you the opportunity to print the current value of any name (variable) bound at that point. You can then step through the remaining code line by line, pausing to repeat these inspections, or you can continue the execution up to the next breakpoint, if present or else finish the script.

The Python standard library comes with a basic debugger that allows you to do just this: pdb

When you import pdb, you can place pdb.set_trace() statements where you want your code to “break”. The pdb debugger will be started and show its prompt, waiting for further instructions. For an easier to read intro to pdd, look here.

IDE’s such as Eclipse with Pydev or PyCharm have built very nice debugging interfaces on top of this basic debugger. If you are used to these tools, using the debugger in Spyder will be a let-down as it does little else besides giving you an alternative option to introduce breakpoints by clicking in the margin to the left of the line you want to stop at, and by giving you a few buttons as an alternative to typing the corresponding (i)pdb commands directly to its prompt in your IPython console.