Machine learning in computational chemistry: foundations and applications - “hands-on”

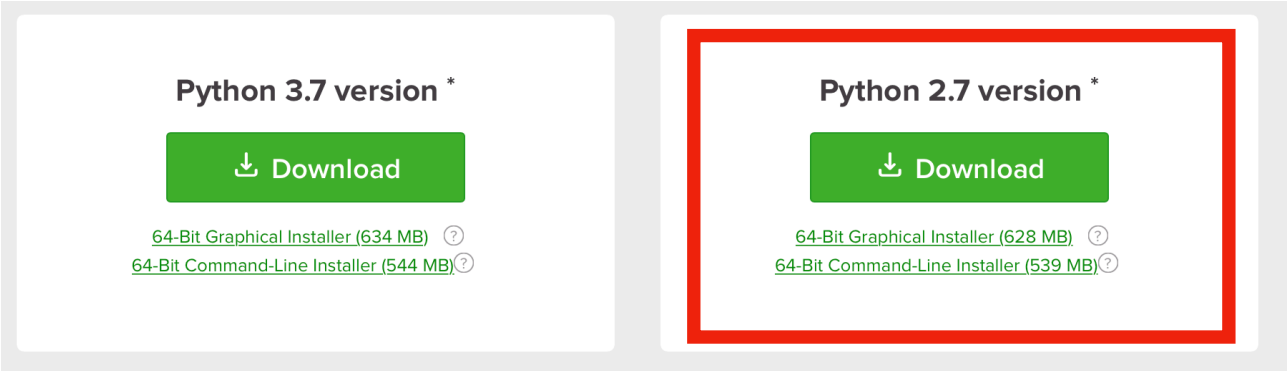
Jörg Meyer, j.meyer@chem.leidenuniv.nl

At first glance this might look like a lot of installation requirements, but you might in fact already have used Python and its continuously growing “ecosystem for scientific purposes” before.
If not, then even if machine learning and/or computational chemistry is not for you, this might still be something useful for your daily work, for example when analysing scientific data.

We are going to use Jupyter for the computer exercises. If you already have this installed together with numpy, scipy and matplotlib based on Python 2.7, you can directly jump to point 2.

1. Python environment for scientific purposes: Anaconda ***(use Python 2.7 version!)***

The Anaconda Python distribution is available for all current major platforms (Windows, MacOS and Linux). Make sure that you chose the version based on Python 2.7, since some of the examples below might not work with Python 3.



https://www.anaconda.com/download

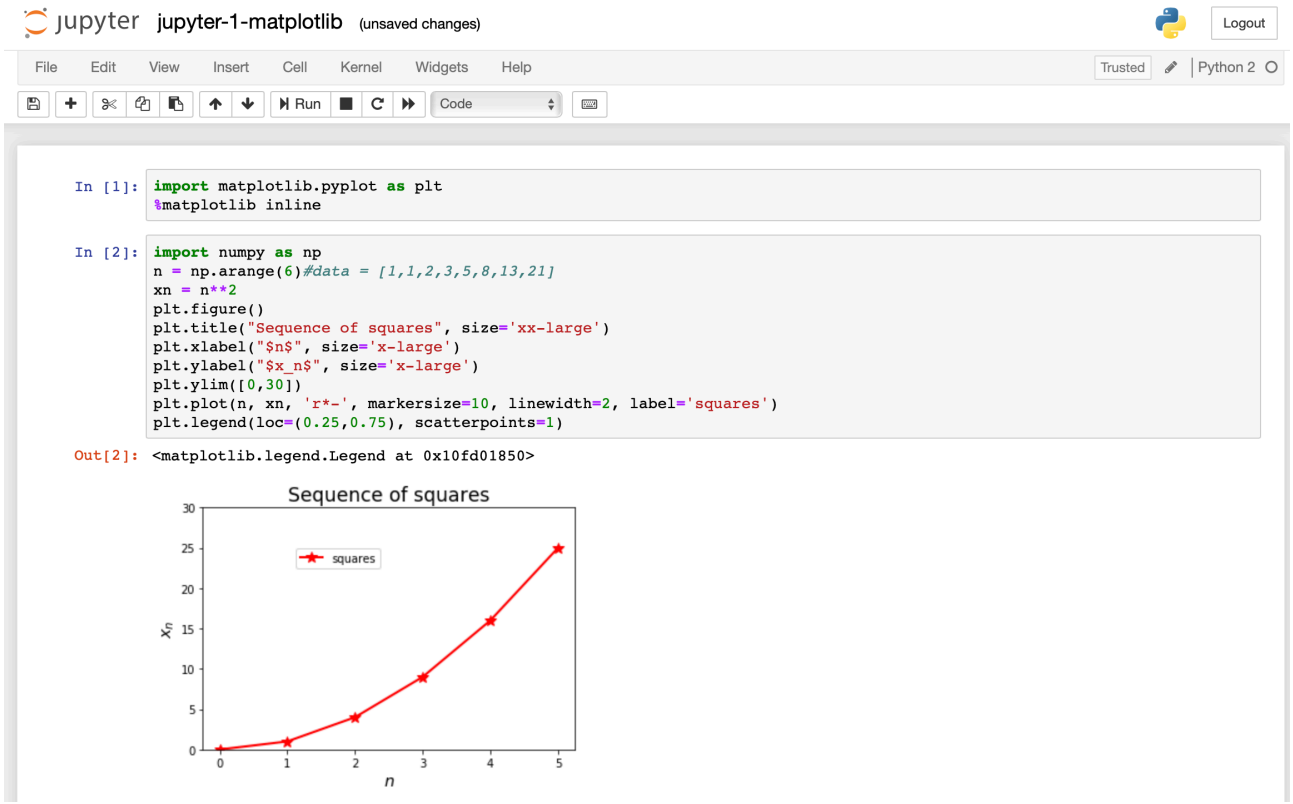
Anaconda allows to conveniently install Python 2.7 together with the aforementioned scientific packages (numpy, scipy, matplotlib and Jupyter) in one go.

http://docs.anaconda.com/anaconda/install/

Note that some of these installation instructions focus on Python 3. Please also verify the installation according to the instructions provided here:

http://docs.anaconda.com/anaconda/install/verify-install/

Finally, you should also check that you can open and run the jupyter-1-matplotlib.ipynb, which is a Jupyter notebook in the form that we are going to use in the exercises.



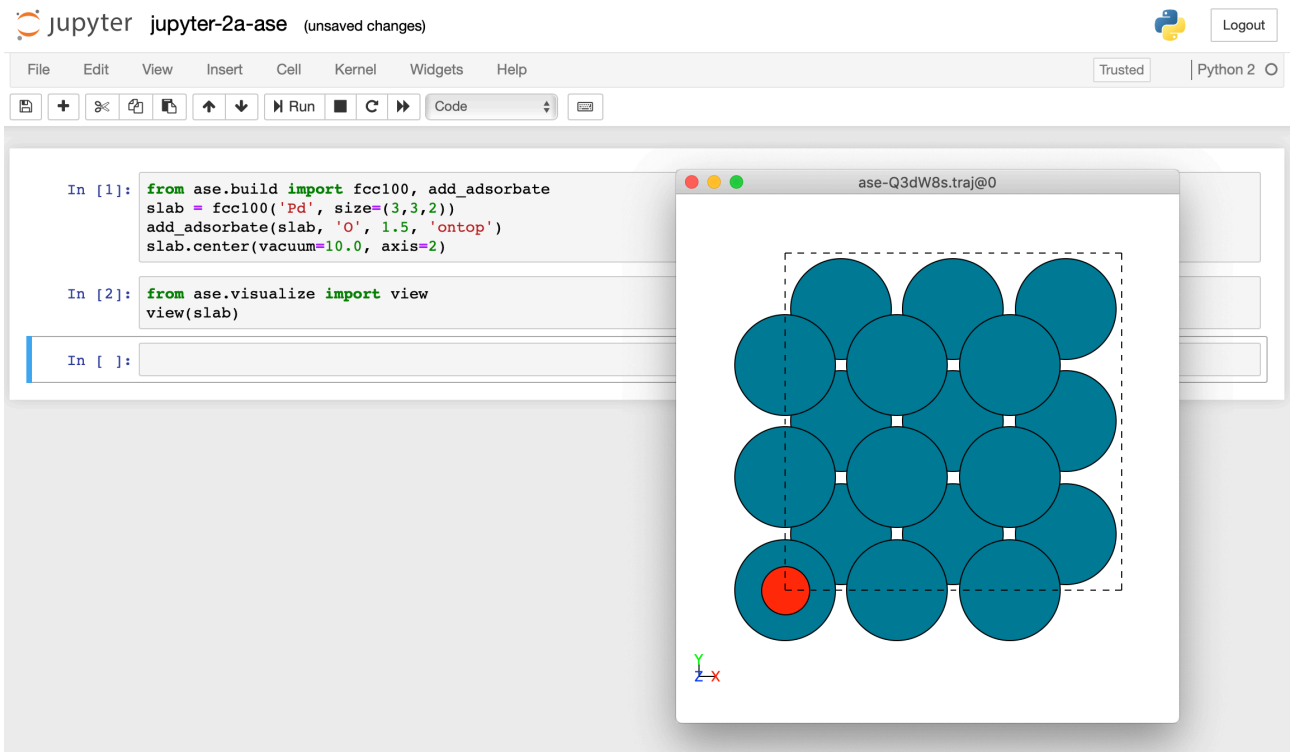
This will functionality should be sufficient for the first computer exercise.

2. Atomic Simulation Environment (ASE) & Atomistic Machine Learning Package (AMP)

With anaconda at hands, you can now easily install the Atomic Simulation Package from here:

https://anaconda.org/conda-forge/ase

After that, running the Jupyter notebook jupyter-2a-ase.ipynb should work and produce the following output:



Installing the Atomistic Machine Learning Package written by

> Khorshidi & Peterson,
> "Amp: A modular approach to machine learning in atomistic simulations",
> Computer Physics Communications 207:3 10-324, 2016.
> DOI:10.1016/j.cpc.2016.05.010

might be the most difficult endeavour (and thus also the most real-world experience when it comes to scientific Python packages). Some instructions are given here:

https://amp.readthedocs.io/en/latest/installation.html

The installation via "pip" should work for an Anaconda installation on any platform when substituting the "pip3" command by "pip".

Although the package can work in "pure Python" mode, this is very slow. Therefore you should build the Fortran extensions which speed up several parts of the package significantly. Building these extensions should work without any problem on Mac and Linux platforms these days, but Windows can be a bit of a hassle. Michael Hirsch provides some excellent hints in his blog about scientific computing

https://www.scivision.co/f2py-running-fortran-code-in-python-on-windows/#install-f2py
https://www.scivision.co/f2py-running-fortran-code-in-python-on-windows/#windows-troubleshooting
https://www.scivision.co/install-windows-subsystem-for-linux/

In order to test the package ignore test section in the installation instructions (seems to be outdated) and use the notebook that I provide instead:

```
In [3]: train_test()
        energy = 9.395566978997334
        forces = [[-1.3140206996297782e-02 -1.8058217118811563e-02  6.9523995215112777e-01]
         [ 1.3140206996297867e-02 -1.8058217118811486e-02  6.9523995215112777e-01]
         [-1.3140206996297726e-02  1.8058217118811396e-02  6.9523995215112844e-01]
         [ 1.3140206996297832e-02  1.8058217118811365e-02  6.9523995215112844e-01]
         [ 4.7954167993790471e-16 -1.9253380950204106e-17 -7.4358462973296569e-01]
         [-4.3715031594615539e-16 -1.2896012340564930e-16 -4.8402579425977466e-01]
         [ 4.9322529667937688e-16  2.7279595063072373e-16 -7.1786406050823615e-01]
         [-5.2549002375190538e-16  1.4940908593261228e-16 -9.1264687369165287e-01]
         [ 5.0306980803327406e-17  1.5951864366762880e-16  1.5269955627770454e-01]
         [-1.2261036953962713e-22 -7.8755278899674138e-17 -7.5538006689587406e-02]]
        energy = 8.705074182978223
        forces = [[-7.0130441931991383e-01 -2.0383048572561843e-01  1.2901072174112955e+00]
         [ 7.3017443612171429e-01  6.9362763349910672e-01  1.2501349473892169e+00]
         [-4.9170618720475873e-01 -4.6775688596150798e-02  1.2977770515543352e+00]
         [ 2.3858807231140122e-01 -7.1336587923641082e-01  1.2964986546788990e+00]
         [ 6.3106026580342134e-01  3.1134633235865855e-01 -1.4732962372376444e+00]
         [ 1.1587018683471442e-01 -2.0071217410622350e-01 -8.2539740571344378e-01]
         [-6.1557828355481448e-01 -9.6086533555573728e-02 -1.6466283831408262e+00]
         [ 9.1920846723759983e-02  2.5838929488359791e-01 -1.0017665166990013e+00]
         [ 1.0095994857244736e-03 -3.3754047176674786e-03 -5.3535191698377116e-02]
         [-3.4517201248723817e-05  7.8290519628178398e-04 -1.3389413654445501e-01]]
```

Using ASE & AMP, in the second computer exercise we are going to revisit some parts of the neural network related chapter of my PhD thesis, which you can already download from here:

https://refubium.fu-berlin.de/handle/fub188/12261

Further reading

Following the links for any of the above packages plus additional googling brings up a lot of material. Perhaps the following two blog entries by Daniel Seita provide some very concise impressions of "cool features" of Python based scientific computing and data visualization:

https://danieltakeshi.github.io/2013/07/05/ten-things-python-programmers-should-know/
https://danieltakeshi.github.io/2016-01-16-ipython-jupyter-notebooks-and-matplotlib/