Introduction to Python

Objective: This exercise is an *optional* pre-exercise. You can use the exercise to check that you have the sufficient coding skills, as well as ensuring that you have your integrated development environment (IDE) setup.

The objective is to setup your Python IDE. Additionally, the objective is to get you acquainted with Python, as well as the exercise format of the course. Upon completing this exercise it is expected that you:

- Have setup your Python IDE.
- Can write and execute basic Python code.
- Understand how data can be represented as vectors and matrices in numerical Python (NumPy)
- Understand basic operations and plotting in Python.
- Understand the format of the exercises.

Material: For this exercise, you will need the 02450 toolbox. See the optional section in the end of the document for more material and tutorials on coding.

Discussion forum: You can get help on our online discussion forum:
https://learn.inside.dtu.dk/d2l/le/79650/discussions/List

0.1 How to do the exercises

The exercises in the course are structured such that you go through an exercise document like this one.

This exercise (Exercise 0) is an optional exercise that you can use to make sure have the sufficient coding skills and have setup the your IDE—that is: the exercise is not introducing any course material, and is meant as a help in preparing for starting with the course and ensuring that you fulfill the course requirements. If you encounter any problems in doing this exercise—in setting up your IDE or installing the 02450 Toolbox—make sure to get in touch with a teaching assistant after the first lecture at the exercises. The first real course exercise (Exercise 1) will be done after the first lecture. Note that if you have difficulties following the scripts, catching up on your

progamming skills are to be done as a self-study exercise (see optional material in the end of this document).

The exercises are centered around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and help you understand what is going on in them. You wont have time to code everything from scratch for the exercises, and so it is important that you get familar with this workflow centered around the existing scripts. The scripts that you go through in the exercises provide the basis for your work the project you will work on, where you will be able to re-use large parts of the code you have worked with in the exercises. However, for the reports you will have to code more yourself and tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex0_4_3.py` and corresponds to the section 0.4.3 in this document.

0.2 Getting started with Python

We strongly recommend installing the **Anaconda** Python distribution. To install Anaconda, go to `https://www.anaconda.com/download/` and download the most recent version for your operating system. If asked, ensure you select the latest python version in the Anaconda installer and accept the default settings.

Anaconda will amongst other things install Spyder, which offers many tools characteristic for mature IDEs, and thus will enhance your programming experience. These include: context help, text completion, syntax highlighting, enhanced editor, integrated interactive console, project navigator and more. Working with Spyder is quite similar to Matlab environment. See: `spyder-ide.org`. In the following it will be assumed Spyder is used to run python commands and edit Python files. To install other packages, note that you can open the **"Conda Command Prompt"** and use `easy_install <packagename>`.

0.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Python. These are distributed as a toolbox which need to be installed.

0.3.1 Download and unzip the 02450 Toolbox for Python, `02450Toolbox_Python.zip`. It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Python/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Python/Data/      # Datasets directory
<base-dir>/02450Toolbox_Python/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_Python/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

0.3.2 To finalize the installation you need to update your path. In Spyder, go to `Tools -> PYTHONPATH Manager` and press `Add path`. Navigate to `<base-dir>/02450Toolbox_Python/Tools/` and press `Select folder`. Restart Spyder for changes to take effect. Test your path by typing:

```
import toolbox_02450
print(toolbox_02450.__version__)
```

If a revision with a date is printed the path to the toolbox is correctly set up.

0.4 Basic operations in Python

0.4.1 *Console* The IDE provides both a console and an editor. The console can be used to execute code fast and interactively, and is often used when debugging your code. You can define and display variables, plot data, and even define functions on-the-fly using interactive console. It is useful when you debug your code or experiment with small chunks of code, mathematical expressions, etc. In Spyder, find the IPython Console (the console) and try typing: `a=9`. The variable `a` should then appear in your Variable explorer If you write `a` in the console, the value of `a` is displayed.

0.4.2 *Editor, scripts and functions* The editor is used to write longer scripts and functions. In Spyder, write the following lines in the script in the editor:

```
a = 3
b = 4
print(a*b)
```

and save the file as `myscript.py`. Run the script from the console by typing `import myscript` in the console and observe the results. The current file can be run in Spyder by pressing `F5` (or the green arrow), the current line or selection can be run by pressing `F9`. You can clear all variables by pressing eraser-like icon in the righthand corner of the console. You can interrupt a running script by pressing `CTRL+C`. The Python kernel can be restarted by pressing `CTRL + .` in the console (needed e.g. if it has crashed).

In many cases you will rather write scripts, and use console only as a supporting/debugging tool. Python script is a file with `.py` extension, which contains instructions, functions, class definitions. Try to write the following code in the Spyder editor, and save it as `mymodule.py` (notice that tabs are important after the first line):

```
def myfunction(n=int):
  """myfunction will return an array of values ranging
  from 1 to the input integer n."""
  x = range(1,n+1)
  return list(x)
```

Note that the triple quotes sign `"""` indicates multiline comments. To use the function, you import the function from the module. You can do this in your script, but also from the console like: `from mymodule import myfunction`. You can then use the function by writing `y = myfunction(9)`. Try writing `help(myfunction)` and see the result. Also, try calling the function a non-integer value and see the result (e.g. `'test'` or `3.1` instead of `9`).

You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

0.4.3 *Making sequences* We often need to use a sequence of numbers. Observe the various results obtained when running `ex0_4_3.py`.

0.4.4 *Indexing* We often need to retrieve or assign a value to a certain part of a vector or matrix. Inspect `ex0_4_4.py` to see how to do indexing in Python.

0.4.5 *Matrix operations* We will use various matrix operations extensively throughout the course. Inspect `ex0_4_5.py` to see how to do some common ones in Python.

0.5 Basic plotting

It is important to visualize the results you obtain. In this part of the exercise, we will make two plots, a simple, and a more elaborate example. Going forwards, try to keep the elements of the figure made in 1.5.2 in mind as a model for minimum required considerations when making a figure.

0.5.1 Inspect `ex0_5_1.py` to see how to do basic plotting in Python. Try modifying the script to display a cosine curve for values in the range $[0, \pi]$.

0.5.2 Now, let us consider a slightly more advanced plot. Inspect `ex0_5_2.py`. We simulate measurements from two sensors (sensor 1 and sensor 2) for a period of 10 seconds, and we say that the sensors output measurents in millivolt. Since the two measurements are done at the same time, we want to do plot them in the same figure. Furthermore, we want to make sure that the axis are labelled correctly both with a name and a designation of the unit of measurement. We also need to make sure that it is easy to see which curves comes from which sensor. Lastly, we ensure that the axis are readable (large enough), both when looking at them in the IDE, but also in an exported version that we might use in a report about the sensors.

0.6 OPTIONAL

The following online materials are recommended:

- `docs.python.org/tutorial` - Introduction into python environment, syntax and data structures. Recommended reading - sections 1, 2, 3, 4 and 5.
- `docs.scipy.org/doc/numpy-1.15.1/user/quickstart.html` - Tutorial introducing the scientific computing in Python, array and matrix operations, indexing and slicing matrices.

- `docs.scipy.org/doc/numpy-1.15.0/user/numpy-for-matlab-users.html` - Useful reference to scientific computing in Python if you have previous experience with Matlab programming.
- `matplotlib.sourceforge.net` - Documentation and examples related to matplotlib module, which we shall use extensively through the course to visualize data and results.
- `spyder-ide.org` - Overview of Spyder IDE for Python code editing/debugging.
- `https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_` - Series of video tutorials covering basics of Python programming.

Especially the Python tutorial (sections 1-5) and NumPy tutorial are recommended. The more you get acquainted with Python now, the easier it will be for you to solve machine learning problems in the following weeks. You will benefit from this course most if you try to implement the solutions on your own, before checking the correct answers. Nevertheless, if you run into problems, the guidelines and the correct scripts will be always provided for your reference.

References