Technische Universität Chemnitz
Faculty of Mathematics

Prof. Dr. O. Ernst
J. Blechschmidt

8th October 2018

Introduction to Data Science

Sheet 1: Introduction to Python

Exercise 1: Introduction – Python

This exercise is intended to get you comfortable with some essential functions of python. The workflow is always the same, i.e., you open a shell and activate the python environment with

```
source /LOCAL/Software/DataScience2018/setup_env
```

Make sure that the output of the command `python --version` is `Python 3.6.6`. You should also check the path of `python` with the command `which python` It should be

```
/LOCAL/Software/DataScience2018/miniconda3/envs/DS2018/bin/python
```

Start an interactive python shell

```
ipython
```

The command `python` starts a plain python shell without coloration and other nice features.

(a) We start with the command

```
a = [1, 2, 3]
```

With

```
type(a)
```

we find out that `a` is of type "list". The expression

```
a[1]
```

yields the **second** element in the list `a`. In contrast to other programming languages such as MATLAB, python starts its enumeration with index `0`. The command

```
len(a)
```

returns the length of the list. Now, define a second list, e.g.,

```
b = [4, 5, 6]
```

and try the command

```
a + b
```

What happens? What happens if you multiply a list by a scalar?

(b) To find out what else we can do with a list, we may type `a.` and press the Tab-Key. In `ipython`, a listing will be displayed with the available methods of the class "list". You can, for example, append the number `1` to the list `a` by

```
a.append(1)
```

You can always type `? a.append` in order to see the built-in documentation of a function. Depending on how extensively the function/method is commented this can be more or less helpful. Now, you should sort the list `a` in descending order. From

```
? a.sort
```

we conclude, that the method has two optional inputs, namely `key` with default value `None` and `reverse` with default value `False`. You can close the help dialogue by typing 'q'.

Since we want to sort the list in descending order, we might set

```
a.sort(reverse = True)
```

(c) Insert the number `5` at the second position of `a`. To achieve this, look for a suitable method and become comfortable with it using `?`. Finally, you should have `a = [3, 5, 2, 1, 1].`

Plain python without additional modules is not capable of handling advanced mathematical problems. Packages extend the functionalities of python in different ways. We will introduce some of them in the upcoming exercises.

Exercise 2: Introduction – Numpy

As already indicated, packages are extremly important in python. Undoubtedly, the most useful package for scientific computing is `numpy`, which is imported by typing

```
import numpy as np
```

Please familiarize yourself with the tutorial
https://docs.scipy.org/doc/numpy/user/quickstart.html
For those who have previously worked with MATLAB, the following overview is also

worth a short look
https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html.

Short note: It is not common to import all functions of a package via `from numpy import *`. Instead, one uses `import numpy` or `import numpy as np`, which provides access to the package's functions via `numpy.ndarray` or `np.ndarray`, resp.

(a) Create the matrix

$$A = \begin{pmatrix} 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

through direct input using the function `np.matrix`.

(b) Create the matrix

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

using the commands `np.arange`, `np.reshape` and `np.asmatrix`. Beware of the difference between a `matrix` and an `array`.

(c) Compute

$$C = A\,B^\top$$

using `A * B.T` and `A.dot(B.T)`. What is the difference between `C**2` and `np.power(C,2)`?

(d) Now create arrays A and B instead of matrices. What happens if you try to compute $D = A\,B^\top$? What is the output of `D**2` and `np.power(D,2)`?

(e) The command

```
np.random.randint(5, size=(2, 4))
```

generates a 2×4 array of random integers between `0` and `4`. Now, create a 40×4 array `X` of standard-normal distributed random variables using `np.random.randn`. Compute the columnwise mean and variance of `X` using the commands `np.mean` and `np.var`. Since you generate random numbers, the result will be slightly different each time you run your code.

For debugging and testing purposes you should always initially set the seed of the random number generator to a fixed value, e.g.,

```
np.random.seed(0)
```

Set the random number seed to `0` and repeat the generation of `X` as well as the computation of the mean and the variance. Compare your results with those of your colleagues.

Exercise 3: Introduction – Plots

This exercise introduces to you some important plot routines. The package `matplotlib` is the most popular module in this context. It provides similar functionality as MATLAB, and is fairly user-friendly. You can import pyplot by

```
import matplotlib.pyplot as plt
```

You can find more information and a nice tutorial under
https://realpython.com/python-matplotlib-guide/

(a) Create a vector `x` (here as a numpy array)

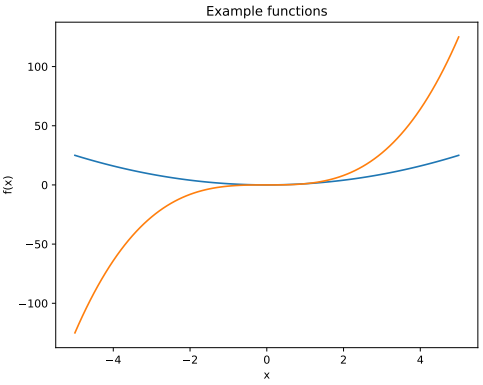
```
x = np.linspace(-5,5,101)
```

that partitions the interval $[-5, 5]$ into 101 equidistant points. Compute $y = x^2$ as well as $z = x^3$. The commands

```
plt.plot(x,y)
plt.show()
```

create a simple plot of the function $f(x) = x^2$. In contrast to MATLAB, python does not create a new figure each time you call `plt.plot`. Python tries to plot everthing into one figure until one finally calls `plt.show()`.

Try to plot both functions $y(x)$ and $z(x)$ in one figure. Change the labels of the x and y axes as well as the title using the functions `plt.xlabel`, `plt.ylabel` and `plt.title`. It is important to do this before you finally call `plt.show()`



(b) Now we want to plot the bivariate function

$$f(x, y) = e^{-(x^2+y^2)}$$

Execute the following code line-by-line and try to figure out what it is doing.

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
z = np.linspace(-3, 3, 201)
x, y = np.meshgrid(z, z)
f = np.exp(-(np.power(x,2) + np.power(y,2)))
```

```
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_surface(x,y,f, cmap=cm.coolwarm)
plt.show()
```

(c) Create a contour plot of the function

$$g(x, y) = \sin(3\,x)\,\cos(2\,y)$$

using the function `plt.contour` or `plt.contourf`.

Remark: With `plt.ion()` and `plt.ioff()` you can create figures in an interactive mode. As the name suggests, you see directly what you have typed., i.e., before calling `plt.show()`.