Übungsblatt 1 (Hallo Python)

Aufgabe 1:

Installation und Inbetriebnahme von Python.

a) Installieren Sie von der Homepage www.python.org die aktuelle Version von Python 3.x.

b) Öffnen Sie nach der Installation durch die Eingabe des Suchbegriffs **cmd** im Windows-Suchfeld eine Windows Eingabeaufforderung (Console).

- Starten Sie eine Python-Shell durch Eingabe von **python** (wenn Sie bei der Installation py-lancher angewählt haben, genügt die Eingabe von **py**).
- Sie erkennen die Shell an den drei spitzen Klammern (**>>>**) Testen Sie die Shell mit einer einfachen Ausgabe wie z.B. **`print("Hallo Python")`**.
- Durch die Eingabe von **`help()`** erreichen Sie die Help-Shell von python. Geben sie zum Testen **`print`** ein (Sie können auch in der Python-Shell direkt **`help(print)`** eingeben um der Hilfefunktion das Suchwort mitzugeben).
- Öffnen Sie das genannte Tutorial https://docs.python.org/3.x/tutorial/ und machen Sie sich kurz mit der Homepage vertraut.
- Verlassen Sie die Help-Shell durch Eingabe von **STRG+z** oder **`quit`**.
- Nun sind Sie wieder in der Python-Shell. Verlassen Sie diese durch Eingabe von **STRG+z** oder **`quit()`**.

Aufgabe 2:

Fingerübungen in der Python-Shell

a) Benutzen Sie die Python-Shell um die Grundrechenarten, sowie die Potent **, die Ganzzahl-Division mit **//** und den Modulo-Operator **%** anzuwenden.

b) Finden Sie mit Hilfe des Internets eine Erklärung für folgendes Verhalten:

```
 22 //  8 = +2
-22 //  8 = -3
-22 // -8 = +2
 22 // -8 = -3
```

Finden Sie eine Erklärung für folgendes Verhalten:

```
 22  %  8 = +6
-22  %  8 = +2
-22  % -8 = -6
 22  % -8 = -2
```

c) Beschäftigen Sie sich mit Komplexen Zahlen in der Python-Shell z.B. durch die Eingabe von:

```
1j * 1j
1j / 1j
1j + 1J
1j – 1j
```

Definieren Sie sich zwei Variablen mit je einer komplexe Zahl z.B.:

```
c1 = 1 + 2j   und   c2 = 2 - 1j
```

Wenden Sie nun die Grundrechenarten aus Teil a) auf die komplexen Zahlen an!

d) Suchen Sie im Internet wie Sie den Sinus von 0, 30, 90, 150, 180, 210, 270, 330, 360 Grad korrekt ausgeben können!?

Aufgabe 3:

Einbinden von Erweiterungsmodulen in ihre Python-Umgebung:

a) Aktualisieren Sie pip (das ist ein Paketverwaltungsprogramm für Python-Pakete aus dem Python Package) durch Eingabe folgender Zeile (z.B. in der Windows-Shell in Ihrem Python-Verzeichnis):
`python -m pip install --upgrade pip` (https://pypi.org/project/pip/)
(evtl. muss pip zunächst installiert werden: https://www.youtube.com/watch?v=c_qNC1IL4qA)

b) Öffnen Sie eine Windows-Eingabeaufforderung und gehen Sie in den Unterordner **`\Scripts`**, im Verzeichnis Ihrer Python-Installation (z.B. **`c:\Tools\Python39\Scripts`**) für die Sie Module hinzufügen wollen. Verwenden Sie nun **pip**, um die Erweiterungsmodule **numpy** (Numerical Python), **`scipy`** (Scientific Python), **`matplotlib`** (MatrixPlotLibrary) und **`pandas`** (Python and data analysis) einzubinden (vgl. https://www.python-kurs.eu/numpy.php):

```
python -m pip install numpy        #Erweiterung für Arrays und Matrizen
python –m pip install scipy        #Erweiterung für z.B. Regression u. FFT
python -m pip install matplotlib   #Erweiterung für Matlab-ähnliche Plots
python –m pip install pandas       #Erweiterung für Tabellen u. Zeitreihen
```

Die Module sollten nun alle im Unterverzeichnis **`\Lib\site-packages`** erscheinen.
Mit diesen Modulen ist Ihre Python-Installation jetzt ähnlich gut gerüstet wie **Matlab !**

c) Öffnen Sie die IDLE-Shell über das Windows-Startmenü und erzeugen Sie mit File->New File ein Editor-Fenster. Speichern Sie das File in einem beliebigen Arbeitsverzeichnis für Ihre Python Programme (z.B. **`c:\<Arbeitsverzeichnis>\<Uebungsblatt>\powerspectrum.py`**
Schreiben Sie folgende Anweisungen in diese Datei um die Erweiterungsmodule zu testen:

```
# import modules and give them abbreviations (for scipy just import fft)
import numpy as np
from scipy.fft import fft
import pandas as pd
import matplotlib.pyplot as mpl

# set number of samples, sample frequency and sample time
k = 1000
fs = 1000
ts = 1/fs
# use numpy to create array of time samples and sinus signal
x = np.linspace(0,ts*k,k)
y = np.sin(100*2*np.pi*x) + 0.5*np.sin(180*2*np.pi*x)
# use scipy to get fft and power spectrum as pandas time series
spec = fft(y)
freq = np.linspace(0,fs/2,k//2)
power = pd.Series(2/k*np.abs(spec[0:k//2]),freq)
# use matplotlib to create figure and 1st subplot to plot numpy arrays
mpl.figure()
mpl.subplot(3,1,1)
mpl.title('Composed Sinus Signal')
mpl.xlabel('Time')
mpl.ylabel('Signal')
mpl.plot(x,y)
# create 2nd subplot and plot power spectrum from pandas series
mpl.subplot(3,1,3)
mpl.title('Power Spectrum of Signal')
mpl.xlabel('Frequency')
mpl.ylabel('Power')
mpl.plot(power)
# show figure on screen. This will block the execution until figure is closed
mpl.show()
```

d) Starten Sie den Python-Interpreter durch Run->Run Module. Sie können das Python-Script auch in einer Windows-Eingabeaufforderung in Ihrem Arbeits- bzw. Übungsverzeichnis durch Eingabe von **`python powerspectrum.py`** oder der Python-Shell ausführen.

Aufgabe 4 (freiwillige Zusatzaufgabe):

Ein Python Virtual Environment schafft eine isolierte Python-Arbeitsumgebung, in der Sie Python-Modul-Abhängigkeiten ohne Einfluss auf globale Python-Module installieren können. Zudem können Sie ein Projekt aus mehreren Modulen organisieren. Sie richten eine virtuelle Python-Umgebung folgendermaßen ein:

a) Öffnen Sie eine Windows-Eingabeaufforderung und gehen Sie mit Hilfe von **`cd c:\...`** in das Verzeichnis in dem Sie Ihre Python-Version installiert haben, von der Sie eine virtuelle Umgebung erstellen wollen (z.B. c:\Tools\Python39). Verwenden Sie **venv,** um nun die virtuelle Arbeitsumgebung einzurichten:

`python –m venv c:\<Arbeitsverzeichnis>\<Umgebungsname>\myenv`

Nun können Sie im Verzeichnis **`c:\<Arbeitsverzeichnis>\<Umgebungsname>`** diverse Python-Scripte speichern, die sich gegenseitig referenzieren und sich alle auf diese virtuelle Python-Umgebung beziehen.

b) In diese virtuelle Arbeitsumgebung können Sie auch alle notwendigen Module integrieren, um so ein vollständig lauffähiges Softwarepaket zu erzeugen:
Gehen Sie dazu in das Unterverzeichnis **`\Scripts`**, im Verzeichnis Ihrer virtuellen Arbeitsumgebung **`c:\<Arbeitsverzeichnis>\<Umgebungsname>\myenv`** und installieren Sie mit pip alle notwendigen Module (siehe Aufgabe 3b). Eventuell ist zuvor auch hier die Version von pip zu aktualisieren!

c) Um zu entscheiden, welche virtuelle Arbeitsumgebung bei der ausführung von Python-Skripten verwendet werden soll ist die entsprechende Umgebung zu aktivieren:
Gehen Sie dazu in das Unterverzeichnis **`\Scripts`**, im Verzeichnis Ihrer virtuellen Arbeitsumgebung **`c:\<Arbeitsverzeichnis>\<Umgebungsname>\myenv`** und starten Sie die Batch-Datei **`acivate`**. Der Shell-Prompt wechselt zu **`(myenv) C:\…`**
Damit sich IDLE ebenfalls auf diese Arbeitsumgebung bezieht ist IDLE jetzt im Verzeichnis **`c:\<Arbeitsverzeichnis>\<Umgebungsname>`** mit folgendem Aufruf zu starten:
`python -m idlelib.idle`.

d) Jetzt können Sie mit IDLE Python-Skripte schreiben, die sich ausschließlich auf die Module in dieser Umgebung beziehen. Zudem lässt sich jetz diese Arbeitzumgebung kopieren und somit lauffähig and andere weitergeben.