Karlsruher Institut für Technologie SS 2022
Institut für Theorie der Kondensierten Materie
Dr. Andreas Poenicke **12.04.2022**
`http://comp.physik.kit.edu/Lehre/ERA/` era@physik.kit.edu

Exercise Sheet 5 for "'Einf"uhrung in das Rechnergest"utzte Arbeiten"'

The exercise are supposed to be done on the Jupyter server of the faculty. Prerequisite is only a browser and an account for the pool.
The jupyter server is available under `https://jupytermachine.etp.kit.edu`.
A short overview of the most important Python directives and functions is summarized on the Python Spickzettel.

1 NumPy

NumPy is a Python module, that simplifies numeric calculations with vectors and matrices considerably. The functionality is heavily oriented on that of Matlab. Likewise it uses highly optimized numeric libraries for high performance calculations.

NumPy defines a new data type called NumPy array (`ndarray`) that is the basis for most methods. Since the mathematical functions from `math` can usually only handle scalars, the analog functions are implemented from scratch in NumPy. You can also find mathematical constants like π in here.

1. Import the module `numpy`.
2. Calculate a `x_max` $= 4\pi$. Generate a NumPy array `x` of $N = 101$ values between 0 and `x_max`.
3. Calculate (in one step) $\sin(x)^2 + \cos(x)^2$ for all these values.
4. NumPy has many functions that calculates properties of NumPy arrays. This way you can calculate e.g. the sum of all elements of an array with one function.
   Calculate another array y=sin(x) and use this summation (i.e. no for-loops!) and the array `y` to approximately calculate the integral over the sin function form 0 to x_{max}.
   Use as a simple approximation for the integral trapezoidal rule
   $$\int_a^b f(x) \approx \frac{a-b}{N-1}\left(\frac{f(x_{N-1}) + f(x_0)}{2} - \sum_{n=0}^{N-1} f(x_n)\right) .$$
   Where N is the number of equidistant(!) values x_n in the interval $[a, b]$.
   Hint: If you calculate for a NumPy array `x` the function $\mathtt{y} = f(\mathtt{x})$, then for each element $\mathtt{y}_n$ of the array `y` holds $\mathtt{y}_n = f(\mathtt{x}_n)$

$\boxed{2}$ Matplotlib

Matplotlib is the second important module, that is regularly used in Python programs in physics. This library offers simple ways to create scientific graphs. Matplotlib can handle with NumPy arrays without problem, so the two packages are often used together.

A comprehensive overview of the features of the module can be found at the Matplotlib gallery.

For all examples there the program to create them is shown as well.

1. Inlude the `pyplot` submodule from `matplotlib`.
2. Transfer your `my_sin(x)` function from the last sheet (or use the one from the example solution) and calculate `y2 = my_sin(x)`[1].
3. Compare the result with the one from `math.sin(x)` by plotting them in a single graph.
4. Add labels to the x and y axes and a legend.
5. As a last step, save the graph to the file `comparison.pdf`.

$\boxed{3}$ SciPy - Curvefit

SciPy is a library with an extensive collection of numerical methods.[2] In the following we will only use `curve_fit` to fit a theoretical function to experimental data.

1. Import only `curve_fit` from the `scipy.optimize` module.
2. Download (like shown in the lecture) the data from the columns of the `data.txt` file in separate arrays `x` and `y`.
3. Define a function `linear(x, a, b)` that calculates and returns $y = a \cdot x + b$.
4. Use `curve_fit()` to fit this function to the data.

   Extract a and b and their respective standard derivation.
5. Plot the data together with the fitted curve in one graph. Use the `+` symbol for the data and a line for the fit function.

   Note: To calculate the fitted data you can still use your function `linear(x, a, b)`.

[1] x should still be the array from exercise 1 with 101 values between 0 and `x_max`

[2] Under `https://docs.scipy.org/doc/scipy/reference/` you can find the documentation of the methods and for many also an introduction with examples.