```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

1 Schreiben und Lesen von Dateien

1.1 Schreiben von Daten in eine Datei

1.1.1 1. Variante

```
z1 = 'Auf dem Seerosenblatt der Frosch\n'
z2 = 'aber was macht er\n'
z3 = 'für ein Gesicht?\n'
# \n ist Zeilenumbruch
```

Datei öffnen für schreibenden Zugriff (w für write)

```
datei_hd = open('haiku.txt', 'w')
datei_hd.write(z1)
datei_hd.write(z2)
datei_hd.write(z3)
print('\n \t (Kobayashi Issa)', file=datei_hd) # alternativ
# \t Tab
datei_hd.close()
```

Am Ende die Datei wieder schließen

1.1.2 2. Variante mit with

```
import numpy as np
x = np.array([1, 2, 3])
y = x**2

with open('data.txt', 'w') as daten:
    for _x, _y in zip(x, y):
        daten.write('{0} \t {1} \n'.format(_x, _y))
```

Die Datei wird automatisch durch with geschlossen.

1.2 Lesen von Daten aus einer Datei

```
x = np.array([])
y = np.array([])
```

1.2.1 Variante 1: Standardpythonstyle

Datei öffnen für lesenden Zugriff (r für read)

```
f = open('data.txt', 'r')

for zeile in f.readlines():  #Line ist ein string
    #Teilt den string bei "whitespace", Rückgabe ist eine Liste mit strings
    zeile = zeile.strip()
    numbers_str = zeile.split()
    #strings in floats umwandeln
    numbers_float = [float(eval(_x)) for _x in numbers_str]
    #die neuen Werte zu x und y hinzufügen
    x = np.append(x, numbers_float[0])
    y = np.append(y, numbers_float[1])
```

Am Ende die Datei wieder schließen

```
f.close()
```

1.2.2 Variante 2: mit with

```
x2 = np.array([])
y2 = np.array([])
with open('data.txt', 'r') as f:
    for zeile in f:
        zeile = zeile.strip()    # entfernt den Zeilenumbruch
        zeile = zeile.split()
        zeile = [float(x) for x in zeile]
        x2 = np.append(x2, zeile[0])
        y2 = np.append(y2, zeile[1])
```

die Datei wird automatisch durch Verwendung von 'with' geschlossen.

1.2.3 Variante 3: Numpystyle

Numpy hat Funktionen zum bequemen Ein- und Auslesen von Daten aus Textdateien hierfür muss in jeder Zeile die gleiche Anzahl von durch Leerzeiche/whitespace getrennte Zahlen stehen

```
import numpy as np
import matplotlib.pyplot as plt
x3, y3 = np.loadtxt('data.txt', unpack=True)
```

`np.loadtxt()` lädt die Daten (durch whitespace/Leerzeichen getrennt) in ein(!) array. Die Option 'unpack' teilt dieses array direkt auf, so dass ohne Umwege in 'x3' und 'y3' gespeichert werden kann.

x3 und y3 sind in diesem Fall np.arrays

Der Inhalt von x und x2 und x3 (bzw. y und y2 und y3) ist gleich:

```
print(' Variante 1 und 2 : {0} {1} \n\
 Variante 1 und 3 : {2} {3}'.format(\
        np.allclose(np.array(x), x2),\
        np.allclose(np.array(y), y2),\
```

```
        np.allclose(np.array(x), x3),\
        np.allclose(np.array(y), y3)))
```

```
Variante 1 und 2 : True True
 Variante 1 und 3 : True True
```

Weitere Infos zum Datei Ein- und Auslesen unter

https://www.python-kurs.eu/numpy_dateien_lesen_schreiben.php

2 Lineare Ausgleichsrechnung (linear least squares)

```
t = np.array([0, 1, 2, 3])
y = np.array([-1, 0.2, 0.9, 2.1])


A = np.vstack([t, np.ones(len(t))]).T
print(A)
```

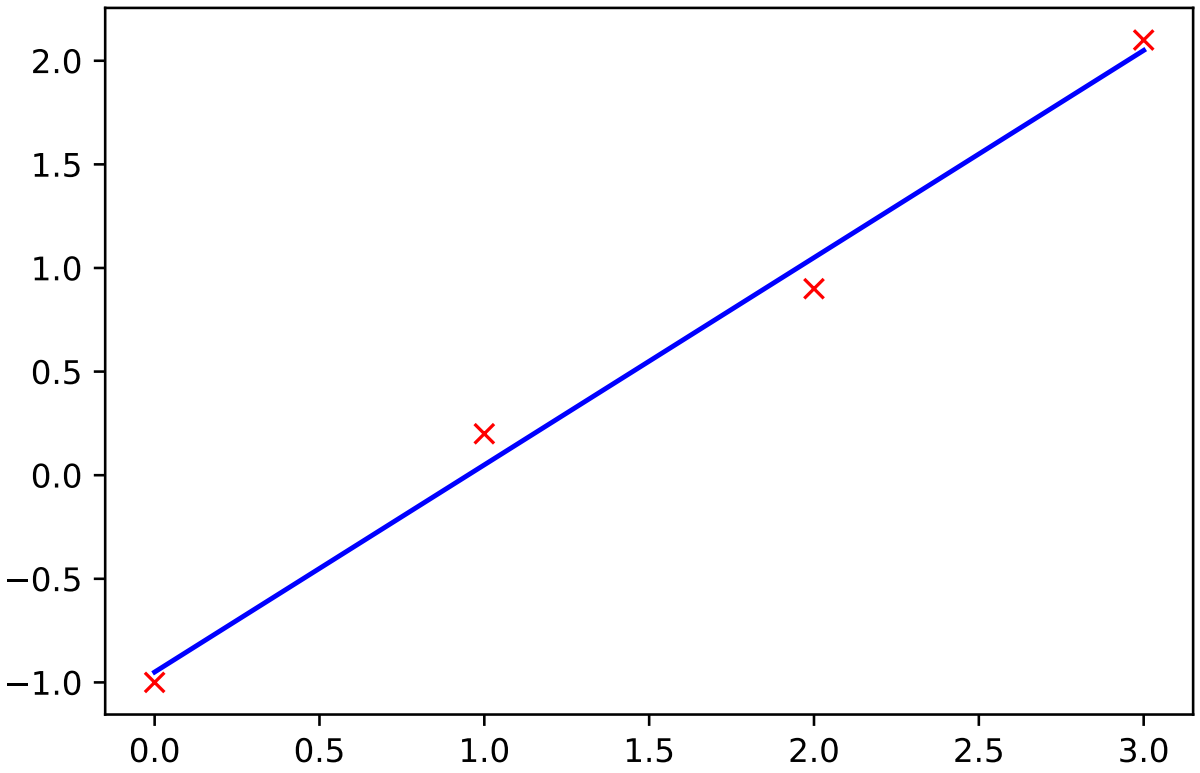
```
[[0. 1.]
 [1. 1.]
 [2. 1.]
 [3. 1.]]
```

Lösung mit Gaußnormalengleichungen

```
x = np.linalg.solve(A.T@A,A.T@y)

plt.figure()
plt.plot(t,y,'rx')
plt.plot(t,x[1]+x[0]*t,'b')
```

```
[<matplotlib.lines.Line2D at 0x7ff8a5f07da0>]
```



spezielles Verfahren für kleinste Fehlerquadrate (engl. least square)

```
xb,res,rank,s = np.linalg.lstsq(A,y,rcond=None)
```