```python
# This script is used to plot the data in different ways.

# Authors:
# Christopher Mahn
# Silas Teske
# Joshua Wolf
# Maria Riegel

# Import of libraries
import os
import main as settings
import lib_analysis as analysis
import matplotlib.pyplot as plt
import numpy as np

# ------------------------------------------------------------------------

# Functions

def import_csv(filename):
    """
    This function imports the csv-files and returns them as sensor-streams".
    """
    with open(os.path.join("data_converted", filename), "r") as file:
        data = file.readlines()
    converted_data = []
    num_of_entries = 0
    sensor_streams = []
    for line in data:
        line = line.split(";")
        converted_line = []
        for entry in line:
            entry = float(entry.strip())
            converted_line.append(entry)
        converted_data.append(converted_line)
        if(len(converted_line) > num_of_entries):
            num_of_entries = len(converted_line)
        del converted_line
    for i in range(num_of_entries):
        sensor_streams.append([])
    for line in converted_data:
        for i, entry in enumerate(line):
            sensor_streams[i].append(entry)
    return(sensor_streams)


def terminate():
    """
    This function terminates the program.
    """
    print("[INFO] The program has been terminated.")
    exit()


# ------------------------------------------------------------------------

# Classes

# ------------------------------------------------------------------------
```

```
60
61 # Beginning of main program
62
63 if(__name__=='__main__'):
64     # Import the data
65     timeserieses = []
66     for dataset in settings.datasets_converted:
67         sensor_streams = (import_csv(dataset))
68         for sensor_stream in sensor_streams:
69             timeserieses.append(np.array(sensor_stream))
70
71     if(len(timeserieses) == 23):
72         print("[INFO] The data has been imported successfully.")
73     else:
74         print("[ERROR] The data has not been imported successfully.")
75         terminate()
76
77     aramis_belasten_entlasten_1 = {'zeit':timeserieses[0], 'kraft':timeserieses[1],
'dms_hinten':timeserieses[2], 'dms_vorn':timeserieses[3]}
78     aramis_belasten_entlasten_2 = {'index':timeserieses[4],
'dehnung':timeserieses[5], 'kraft':timeserieses[6]}
79     belastung_entlastung_7000n = {'zeit':timeserieses[7], 'kraft':timeserieses[8],
'dms_hinten':timeserieses[9], 'dms_vorn':timeserieses[10]}
80     kabelkompensation_aus_magnet_heissluft = {'zeit':timeserieses[11],
'kraft':timeserieses[12], 'dms_hinten':timeserieses[13], 'dms_vorn':timeserieses[14]}
81     netzkabel_neben_messleitung = {'zeit':timeserieses[15], 'kraft':timeserieses[16],
'dms_hinten':timeserieses[17], 'dms_vorn':timeserieses[18]}
82     schwingung = {'zeit':timeserieses[19], 'kraft':timeserieses[20],
'dms_hinten':timeserieses[21], 'dms_vorn':timeserieses[22]}
83
84     # Plotting of data
85
86     # Dehnung im Verhältnis zur Kraft für Aramis Belasten Entlasten 1
87     analysis.plot_xy([[1000*aramis_belasten_entlasten_1['kraft'],
aramis_belasten_entlasten_1['dms_hinten']/1000],
88                       [1000*aramis_belasten_entlasten_1['kraft'],
aramis_belasten_entlasten_1['dms_vorn']/1000]],
89                      ["Aramis Belasten Entlasten 1 (hinten)",
90                       "Aramis Belasten Entlasten 1 (vorn)"],
91                      "Kraft [N]", "Dehnung [mm]", "Aramis Belasten Entlasten 1")
92
93     # Torsion im Verhältnis zur Kraft für Aramis Belasten Entlasten 1
94     analysis.plot_xy([[1000*aramis_belasten_entlasten_1['kraft'],
(aramis_belasten_entlasten_1['dms_hinten']-
aramis_belasten_entlasten_1['dms_vorn'])/1000]],
95                      ["Aramis Belasten Entlasten 1 (hinten-vorn)"],
96                      "Kraft [N]", "Dehnung [mm]", "Aramis Belasten Entlasten 1")
97
98     # Dehnung im Verhältnis zur Kraft für Aramis Belasten Entlasten 2
99     analysis.plot_xy([[aramis_belasten_entlasten_2['kraft'],
aramis_belasten_entlasten_2['dehnung']]],
100                     ["Aramis Belasten Entlasten 2"],
101                     "Kraft [N]", "Dehnung [%]", "Aramis Belasten Entlasten 2")
102
103    # Dehnung im Verhältnis zur Kraft für Belastung Entlastung 7000N
104    analysis.plot_xy([[1000*belastung_entlastung_7000n['kraft'],
belastung_entlastung_7000n['dms_hinten']/1000],
105                      [1000*belastung_entlastung_7000n['kraft'],
belastung_entlastung_7000n['dms_vorn']/1000]],
106                     ["Belastung Entlastung 7000N (hinten)",
```

```
                     "Belastung Entlastung 7000N (vorn)"],
                    "Kraft [N]", "Dehnung [mm]", "Belastung Entlastung 7000N")

    # Torsion im Verhältnis zur Kraft für Belastung Entlastung 7000N
    analysis.plot_xy([[1000*belastung_entlastung_7000n['kraft'],
(belastung_entlastung_7000n['dms_hinten']-
belastung_entlastung_7000n['dms_vorn'])/1000]],
                     ["Belastung Entlastung 7000N (hinten-vorn)"],
                     "Kraft [N]", "Dehnung [mm]", "Belastung Entlastung 7000N")

    # Dehnung im Verhältnis zur Kraft für Kabelkompensation Aus Magnet Heissluft
    analysis.plot_xy([[kabelkompensation_aus_magnet_heissluft['zeit'],
kabelkompensation_aus_magnet_heissluft['dms_hinten']/1000],
                      [kabelkompensation_aus_magnet_heissluft['zeit'],
kabelkompensation_aus_magnet_heissluft['dms_vorn']/1000]],
                     ["Kabelkompensation Aus Magnet Heissluft
(hinten)","Kabelkompensation Aus Magnet Heissluft (vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Kabelkompensation Aus Magnet
Heissluft")

    # Torsion im Verhältnis zur Kraft für Kabelkompensation Aus Magnet Heissluft
    analysis.plot_xy([[kabelkompensation_aus_magnet_heissluft['zeit'],
(kabelkompensation_aus_magnet_heissluft['dms_hinten']-
kabelkompensation_aus_magnet_heissluft['dms_vorn'])/1000]],
                     ["Kabelkompensation Aus Magnet Heissluft (hinten-vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Kabelkompensation Aus Magnet
Heissluft")

    # Dehnung im Verhältnis zur Kraft für Netzkabel Neben Messleitung
    analysis.plot_xy([[netzkabel_neben_messleitung['zeit'],
netzkabel_neben_messleitung['dms_hinten']/1000],
                      [netzkabel_neben_messleitung['zeit'],
netzkabel_neben_messleitung['dms_vorn']/1000]],
                     ["Netzkabel Neben Messleitung (hinten)","Netzkabel Neben
Messleitung (vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Netzkabel Neben Messleitung")

    # Torsion im Verhältnis zur Kraft für Netzkabel Neben Messleitung
    analysis.plot_xy([[netzkabel_neben_messleitung['zeit'],
(netzkabel_neben_messleitung['dms_hinten']-
netzkabel_neben_messleitung['dms_vorn'])/1000]],
                     ["Netzkabel Neben Messleitung (hinten-vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Netzkabel Neben Messleitung")

    # Dehnung im Verhältnis zur Kraft für Schwingung
    analysis.plot_xy([[schwingung['zeit'], schwingung['dms_hinten']/1000],
                      [schwingung['zeit'], schwingung['dms_vorn']/1000]],
                     ["Schwingung (hinten)","Schwingung (vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Schwingung")

    # Torsion im Verhältnis zur Kraft für Schwingung
    analysis.plot_xy([[schwingung['zeit'], (schwingung['dms_hinten']-
schwingung['dms_vorn'])/1000]],
                     ["Schwingung (hinten-vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Schwingung")

    # Amplitude von Frequenzen bei Schwingung
    fft_x, fft_y = analysis.fast_fourier_transform(schwingung['dms_vorn']-
schwingung['dms_hinten'], 2400)
    analysis.plot_xy([[fft_x, fft_y]],
```

```
                    ["Schwingung (hinten-vorn)"],
                    "Frequenz [Hz]", "Amplitude", "Schwingung")

    # Amplidute von Frequenzen bei Netzkabel Neben Messleitung
    fft_x, fft_y = analysis.fast_fourier_transform(netzkabel_neben_messleitung['dms_vorn']-netzkabel_neben_messleitung['dms_hinten'], 2400)
    analysis.plot_xy([[fft_x, fft_y]],
                    ["Netzkabel Neben Messleitung (hinten-vorn)"],
                    "Frequenz [Hz]", "Amplitude", "Netzkabel Neben Messleitung")

    """
    # Low-Pass-Filter für Netzkabel Neben Messleitung
    analysis.plot_xy([[netzkabel_neben_messleitung['zeit'], analysis.low_pass_filter(netzkabel_neben_messleitung['dms_vorn']-netzkabel_neben_messleitung['dms_hinten'], int(2400))]],
                     ["Netzkabel Neben Messleitung (hinten-vorn)"],
                     "Zeit [s]", "Dehnung [mm]", "Netzkabel Neben Messleitung")

    # Amplitude von Low-Pass-Filter für Netzkabel Neben Messleitung
    fft_x, fft_y = analysis.fast_fourier_transform(analysis.low_pass_filter(netzkabel_neben_messleitung['dms_vorn']-netzkabel_neben_messleitung['dms_hinten'], int(2400)), 2400)
    analysis.plot_xy([[fft_x, fft_y]],
                    ["Netzkabel Neben Messleitung (hinten-vorn)"],
                    "Frequenz [Hz]", "Amplitude", "Netzkabel Neben Messleitung")
    """

    # Kraftverlauf bei Arammis Belasten Entlasten 1 Aramis Belasten Entlasten 2
    analysis.plot_xy([[((aramis_belasten_entlasten_1['zeit'])), aramis_belasten_entlasten_1['kraft']*1000],
                     [(aramis_belasten_entlasten_2['index']*0.5)+33, aramis_belasten_entlasten_2['kraft']]],
                     ["Aramis Belasten Entlasten 1", "Aramis Belasten Entlasten 2"],
                    "Zeit [s]", "Kraft [N]", "Aramis Belasten Entlasten")

    # Dehnung von Messdatei 2 in Zeitbezug von Messdatei 1 umrechnen
    # index*0.5+33=Zeit
    aramis = {'zeit': aramis_belasten_entlasten_1['zeit'], 'dms_vorn': aramis_belasten_entlasten_1['dms_vorn'], 'dms_hinten': aramis_belasten_entlasten_1['dms_hinten'], 'kraft': aramis_belasten_entlasten_1['kraft']*1000, 'kraft_alt': np.zeros(len(aramis_belasten_entlasten_1['zeit'])).tolist(), 'dehnung': np.zeros(len(aramis_belasten_entlasten_1['zeit'])).tolist()}
    for index, i_zeit in enumerate(aramis_belasten_entlasten_1['zeit']):
        index_neu = int(i_zeit*2-66)
        if(index_neu >= 0 and index_neu < len(aramis_belasten_entlasten_2['index'])):
            aramis['dehnung'][index] = aramis_belasten_entlasten_2['dehnung'][index_neu]
            aramis['kraft_alt'][index] = aramis_belasten_entlasten_2['kraft'][index_neu]
        else:
            aramis['dehnung'][index] = np.nan
            aramis['kraft_alt'][index] = np.nan


    # Dehnung im Verhältnis zur Kraft für Aramis Belasten Entlasten
    analysis.plot_xy([[aramis['kraft'], aramis['dehnung']]],
                    ["Aramis Belasten Entlasten"],
                    "Kraft [N]", "Dehnung [%]", "Aramis Belasten Entlasten")
```

```
196
197     # Dehnung im Verhältnis zur Kraft für Aramis Belasten Entlasten
198     analysis.plot_xy([[aramis['kraft'], aramis['dehnung']],
199                       [aramis['kraft_alt'], aramis['dehnung']]],
200                      ["Aramis Belasten Entlasten", "Aramis Belasten Entlasten (nicht
    korrigiert)"],
201                      "Kraft [N]", "Dehnung [%]", "Aramis Belasten Entlasten")
202
203     # Dehnung im Verhältnis zur Kraft für Aramis und beide DMS
204     analysis.plot_xy([[aramis['kraft'], aramis['dehnung']],
205                       [aramis['kraft'], aramis['dms_vorn']/1000],
206                       [aramis['kraft'], aramis['dms_hinten']/1000]],
207                      ["Aramis", "DMS Vorn", "DMS Hinten"],
208                      "Kraft [N]", "Dehnung [%]", "Aramis und DMS")
209
210     # Bestimmung von Steigung und y-Achsenabschnitt
211     kraft_no_nan = []
212     dehnung_no_nan = []
213     for i, e in enumerate(aramis['kraft']):
214         if(not np.isnan(e) and not np.isnan(aramis['dehnung'][i])):
215             kraft_no_nan.append(e/1000)
216             dehnung_no_nan.append(aramis['dehnung'][i])
217     steigung, offset = analysis.linear_regression(kraft_no_nan, dehnung_no_nan)
218     print(f'Steigung: {steigung:.3f} prozentuale Dehnung pro kN\nOffset: {offset:.6f}
    Prozent')
```