ETH zürich

Übungslektion 4 – Arbeiten mit Daten

Informatik II

10. März 2026

Willkommen!

Polybox

Passwort: jschul

Personal Website

`https://jschultev.github.io/personal_website/`

Heutiges Programm

Matplotlib

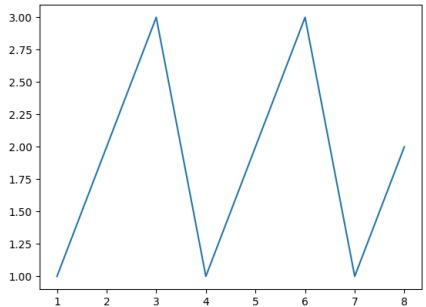
Pandas

Hausaufgaben

1. Matplotlib

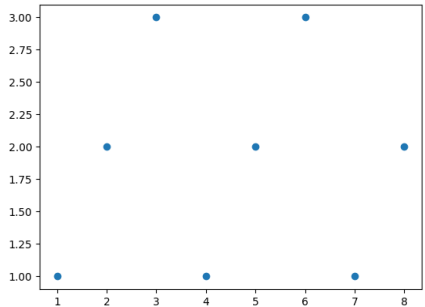
Matplotlib: Liniendiagramm

```
import matplotlib.pyplot as plt

X = range(1, 9)
Y = [1, 2, 3, 1, 2, 3, 1, 2]

fig = plt.figure()
ax = fig.add_subplot()

ax.plot(X, Y)
plt.show()
```



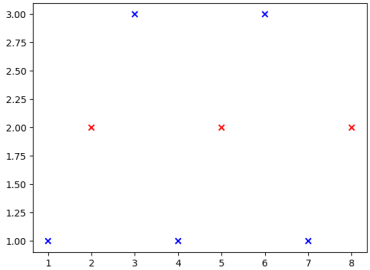
Matplotlib: Streudiagramm

```
import matplotlib.pyplot as plt

X = range(1, 9)
Y = [1, 2, 3, 1, 2, 3, 1, 2]

fig = plt.figure()
ax = fig.add_subplot()

ax.scatter(X, Y)
plt.show()
```



Matplotlib: Farbe und Marker

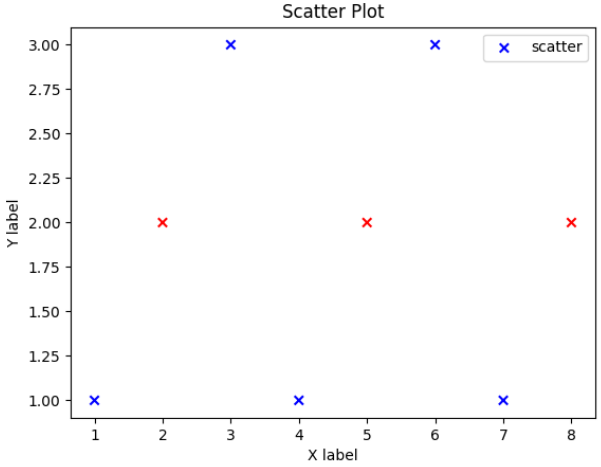
```
import matplotlib.pyplot as plt

X = range(1, 9)
Y = [1, 2, 3, 1, 2, 3, 1, 2]

fig = plt.figure()
ax = fig.add_subplot()

cols = ["red" if y == 2 else "blue" for y in Y]
ax.scatter(X, Y, marker = "x", c = cols)
plt.show()
```



Matplotlib: Beschriftung, Titel, Legende

```
import matplotlib.pyplot as plt
X = range(1, 9)
Y = [1, 2, 3, 1, 2, 3, 1, 2]
fig = plt.figure()
ax = fig.add_subplot()
cols = ["red" if y == 2 else "blue" for y in Y]
ax.scatter(X, Y, marker = "x", c = cols, label="scatter")
ax.set_xlabel('X label') # add an x-label to the X axes
ax.set_ylabel('Y label') # add a y-label to the Y axes
ax.set_title("Scatter Plot") # add a title
ax.legend() # add a legend
plt.show()
```

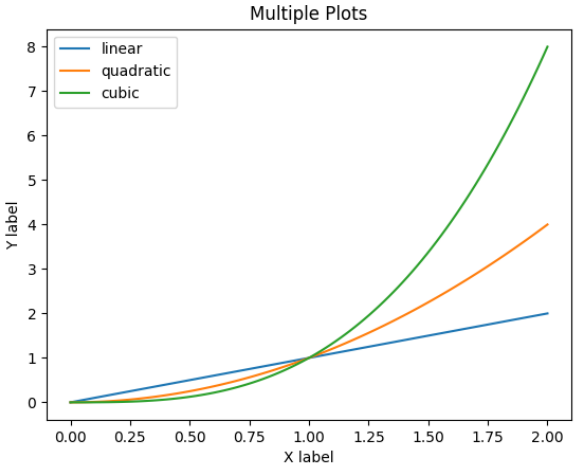
Matplotlib: Beschriftung, Titel, Legende



Matplotlib: Mehrere Plots

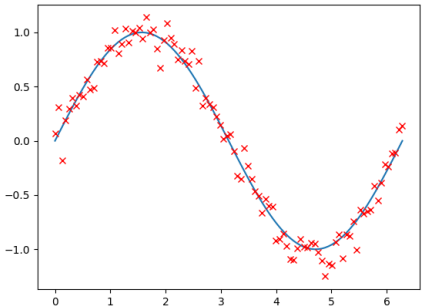
```
import matplotlib.pyplot as plt
import numpy as np
X = np.linspace(0, 2, 100)
fig = plt.figure()
ax = fig.add_subplot()
ax.plot(X, X, label='linear') # plot a linear line.
ax.plot(X, X**2, label='quadratic') # plot a quadratic line.
ax.plot(X, X**3, label='cubic') # plot a cubic line.
ax.set_xlabel('X label')
ax.set_ylabel('Y label')
ax.set_title("Multiple Plots")
ax.legend()
plt.show()
```

Matplotlib: Mehrere Plots



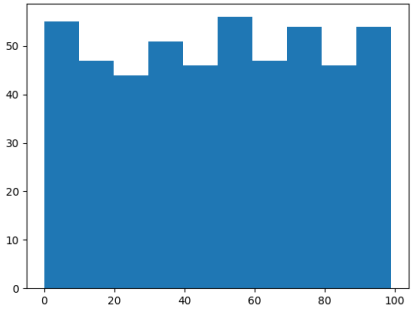
Matplotlib: Zeichnen Sie eine Funktion

```
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot()

X = np.linspace(0, 2*np.pi, 100)
Y = np.sin(X)
ax.plot(X, Y)

Z = Y + np.random.normal(0, 0.1, 100)
ax.plot(X, Z, "rx") # "rx" means red color and x marker
plt.show()
```



Matplotlib: Histogrammdiagramm

```
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.add_subplot()

X = np.random.randint(0, 100, 500)
# low: 0, high: 100, size: 500
ax.hist(X, bins=10)
plt.show()
```



Matplotlib: Informationen mit „Hilfe“ abrufen

```
help(plt)
help(plt.figure)
help(plot.axes)
help(ax.plot)
help(ax.scatter)
help(ax.hist)
help(ax.set_xlabel)
...
```

Matplotlib Kosmetik

```
plt.plot(x, y, linestyle="--") # Strichlinie
plt.plot(x, y, linestyle=":") # Gepunktet
plt.plot(x, y, linestyle="-") # Durchgehend (Standard)
plt.plot(x, y, linestyle="-.") # Strich-Punkt
```

```
plt.plot(x, y, color="r")    # Rot
plt.plot(x, y, color="blue") # Blau
plt.plot(x, y, color="#FF5733") # Hex-Code
```

```
plt.plot(x, y, marker="o") # Punkte
plt.plot(x, y, marker="s") # Quadrate
plt.plot(x, y, marker="x") # Kreuze
```

Matplotlib ZF (Max Schaldach)

Matplotlib ist eine Bibliothek zur Erstellung von Grafiken

```
import matplotlib.pyplot as plt
X, Y = [0,1,2,3],[0,2,4,6]
fig = plt.figure()
ax = fig.add_subplot()
ax.plot(X, Y)  # lineare Funktion
plt.show()
```

```
fig = plt.figure()
ax = fig.add_subplot()
ax.scatter(X, Y, marker = 'x', c = 'r')  # Scatterplot
plt.show()
```

`marker` gibt das Symbol und `c` die Farbe der Datenpunkte an

```
import numpy as np
values = np.random.randint(1,100,1000)
fig = plt.figure()
ax = fig.add_subplot()
ax.hist(values, bins = 100)  # Histogramm
plt.show()
```

`bins` legt fest wie viele Säulen dargestellt werden

Matplotlib: In-class Exercise

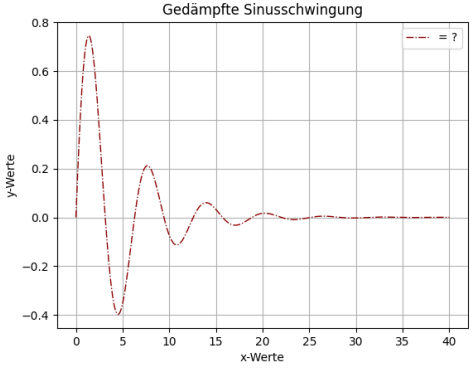
Normalerweise gestellte Aufgabe. Diese ist aber viel zu schwer und auch auf keinen Fall prüfungsrelevant. Ich habe deshalb selbst Aufgaben erstellt für euch zum üben und verstehen.

Matplotlib: Aufgabe 1

Scatterplot mit `pandas` und `matplotlib`:

1. Erzeugt folgende Variablen: np.random.uniform(low, high, size)
   - `x` mit 50 Werten zwischen 0 und 10.
   - `y` mit 50 Werten zwischen 0 und 10.
2. Erstellt einen **Scatter Plot**.
3. Verwendet **blaue Kreuze** als Marker.
4. Beschriftet die Achsen und gebt dem Diagramm eine **Legende**.
5. Fügt dem diagramm ein Gitter hinzu.

Matplotlib: Aufgabe 2



Matplotlib Aufgabe 3

Linienplot mit einer quadratischen Funktion:

1. Erzeugt folgende Variablen:
   - `x`: Werte von 0 bis 4.5 (mindestens 100 gleichmäßig verteilte Punkte).
   - `y`: Die Werte der Funktion $y = x^3 - 6x^2 + 9x - 2$.
2. Plotten der Daten als **Linienplot** mit folgenden Eigenschaften:
   - blaue Kreise als Marker (markerfacecolor)
   - Linienstärke = 5.
3. Fügt eine **Legende** und **Achsenbeschriftungen** hinzu.
4. Setzen Sie den **Plot-Titel** auf Quadratische Funktion.
5. Benutzt folgenden Befehl: `plt.style.use('dark_background')`
6. Setzt alles auf die Farbe "Deeppink" was ihr finden könnt. Wenn ihr nicht wisst wie, geht ins Internet, fragt AI, aber make it pink! (Ausser den Hintegrund und die Marker)

2. Pandas

Pandas: CSV-Daten

CSV steht für **`Comma Separated Values`**.
Eine CSV-Datei enthält eine Tabelle von Daten
CSV ist ein textuelles Format. Zeilen sind durch Zeilensprungzeichen voneinander getrennt
Spalten sind durch Kommas oder ein anderes vereinbartes Zeichen (oftmals Tabulatorzeichen **`\t`**) getrennt.

Item	noah	liame	emma	mia
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–
…				

Pandas: CSV-Daten lesen

```
import pandas as pd
```

Daten aus der CSV-Datei lesen

```
data = pd.read_csv("Expense.csv", sep="\t", index_col="Item")
```

Item	noah	liam	emma	mia
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–

Pandas: CSV-Daten lesen

Lesen Sie hartcodierte Daten

```
import io
csv_file = io.StringIO("""
Item;noah;liame;emma;mia
Food;270;140;188;200
Insurance;72;310;88;72
Fun;410;-;60;130
Salary;-;1510;-;-
Tuition;720;820;720;720
Rent;-;430;390;-
""")
data = pd.read_csv(csv_file, sep=";", index_col="Item")
```

Pandas: Spalten umbenennen

Verwenden Sie die Funktion „Umbenennen“

```
data = data.rename(columns={
        "noah": "Noah", "liame": "Liam", \
        "emma": "Emma", "mia": "Mia"})
```

	noah	liam	emma	mia
Item				
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–

→

	Noah	Liam	Emma	Mia
Item				
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–

Pandas: Spalten umbenennen

Spaltennamen direkt festlegen

```
data.columns = ["Noah", "Liam", "Emma", "Mia"]
```

	noah	liam	emma	mia
Item				
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–

→

	Noah	Liam	Emma	Mia
Item				
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	–	60	130
Salary	–	1510	–	–
Tuition	720	820	720	720
Rent	–	430	390	–

Pandas: Umgang mit ungültigen Daten

Normalerweise erkennt Pandas Spalten mit Zahlen und setzt den Datentyp der Spalte automatisch auf **Numerisch**.
Leider sind die Spalten nicht numerisch, da sie das Zeichen „-“ haben.

```
pd.Series([1, 2, 3]).dtype # check default data type
```

```
dtype('int64')
```

```
data["Noah"].dtype, data["Liam"].dtype, \
data["Emma"].dtype, data["Mia"].dtype
```

```
(dtype('O'), dtype('O'), dtype('O'), dtype('O')) # not numeric
```

Pandas: Umgang mit ungültigen Daten

Wandeln Sie die Spalten in numerische Spalten um

```
for column_name in data.columns:
    data[column_name] = pd.to_numeric(\
    data[column_name], errors="coerce")
    # if 'coerce', then invalid parsing will be set as NaN.
```

	noah	liam	emma	mia
Item				
Food	270	140	188	200
Insurance	72	310	88	72
Fun	410	-	60	130
Salary	-	1510	-	-
Tuition	720	820	720	720
Rent	-	430	390	-

→

	Noah	Liam	Emma	Mia
Item				
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	NaN	60.0	130.0
Salary	NaN	1510.0	NaN	NaN
Tuition	720.0	820.0	720.0	720.0
Rent	NaN	430.0	390.0	NaN

Pandas: Umgang mit ungültigen Daten

Löschen Sie die Zeilen mit NaN-Werten

```
data = data.dropna(how="any")
# how="any" -> If any NaN is present, drop the row.
```

	Noah	Liam	Emma	Mia
Item				
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	NaN	60.0	130.0
Salary	NaN	1510.0	NaN	NaN
Tuition	720.0	820.0	720.0	720.0
Rent	NaN	430.0	390.0	NaN

→

	Noah	Liam	Emma	Mia
Item				
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Tuition	720.0	820.0	720.0	720.0

Pandas: Umgang mit ungültigen Daten

Füllen Sie die NaN-Werte mit einem anderen Wert, z. B. 0

```
data = data.fillna(0)
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	NaN	60.0	130.0
Salary	NaN	1510.0	NaN	NaN
Tuition	720.0	820.0	720.0	720.0
Rent	NaN	430.0	390.0	NaN

→

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	0.0	60.0	130.0
Salary	0.0	1510.0	0.0	0.0
Tuition	720.0	820.0	720.0	720.0
Rent	0.0	430.0	390.0	0.0

Pandas: Datenfilterung

```
data["Noah"] > 0
```

```
Item
Food          True
Insurance     True
Fun           True
Salary       False
Tuition       True
Rent         False
Name: Noah, dtype: bool
```

```
data[data["Noah"] > 0]
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	0.0	60.0	130.0
Tuition	720.0	820.0	720.0	720.0

Pandas: Datenfilterung

Filtern Sie Daten mit mehreren Bedingungen

```
data[(data["Noah"] > 0) & (data["Liam"] > 0) & \
    (data["Emma"] > 0) & (data["Mia"] > 0)]
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Tuition	720.0	820.0	720.0	720.0

Pandas: Datenfilterung

Wählen Sie bestimmte Zeilen aus

```
data[data.index != "Fun"]
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Salary	0.0	1510.0	0.0	0.0
Tuition	720.0	820.0	720.0	720.0
Rent	0.0	430.0	390.0	0.0

```
data.loc[["Food", "Rent"], :]
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Rent	0.0	430.0	390.0	0.0

Pandas: Datenänderung

Fügen Sie eine neue Spalte hinzu

```
data["Total"] = data["Noah"] + data["Liam"] + \
                data["Emma"] + data["Mia"]
```

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	0.0	60.0	130.0
Salary	0.0	1510.0	0.0	0.0
Tuition	720.0	820.0	720.0	720.0
Rent	0.0	430.0	390.0	0.0

→

Item	Noah	Liam	Emma	Mia	Total
Food	270.0	140.0	188.0	200.0	798.0
Insurance	72.0	310.0	88.0	72.0	542.0
Fun	410.0	0.0	60.0	130.0	600.0
Salary	0.0	1510.0	0.0	0.0	1510.0
Tuition	720.0	820.0	720.0	720.0	2980.0
Rent	0.0	430.0	390.0	0.0	820.0

Pandas: Datenänderung

Entfernen Sie eine Spalte

```
data = data.drop(columns="Total")
```

Item	Noah	Liam	Emma	Mia	Total
Food	270.0	140.0	188.0	200.0	798.0
Insurance	72.0	310.0	88.0	72.0	542.0
Fun	410.0	0.0	60.0	130.0	600.0
Salary	0.0	1510.0	0.0	0.0	1510.0
Tuition	720.0	820.0	720.0	720.0	2980.0
Rent	0.0	430.0	390.0	0.0	820.0

→

Item	Noah	Liam	Emma	Mia
Food	270.0	140.0	188.0	200.0
Insurance	72.0	310.0	88.0	72.0
Fun	410.0	0.0	60.0	130.0
Salary	0.0	1510.0	0.0	0.0
Tuition	720.0	820.0	720.0	720.0
Rent	0.0	430.0	390.0	0.0

Pandas: Datenzusammenfassung

```
data.sum()
```

```
Noah    1472.0
Liam    3210.0
Emma    1446.0
Mia     1122.0
dtype: float64
```

```
data.max()
```

```
Noah    720.0
Liam   1510.0
Emma    720.0
Mia     720.0
dtype: float64
```

```
data["Noah"].sum()
```

```
1472.0
```

```
data["Noah"].max()
```

```
720.0
```

Pandas: Datenzusammenfassung

```
data.agg(["max", "sum"]) # use a list of aggregate functions
```

	Noah	Liam	Emma	Mia
max	720.0	1510.0	720.0	720.0
sum	1472.0	3210.0	1446.0	1122.0

Pandas: Datenzusammenfassung

```
data.T.describe() # get statistical information
```

Item	Food	Insurance	Fun	Salary	Tuition	Rent
count	4.000000	4.000000	4.000000	4.0	4.0	4.00000
mean	199.500000	135.500000	150.000000	377.5	745.0	205.00000
std	53.674948	116.577585	181.291662	755.0	50.0	237.27621
min	140.000000	72.000000	0.000000	0.0	720.0	0.00000
25%	176.000000	72.000000	45.000000	0.0	720.0	0.00000
50%	194.000000	80.000000	95.000000	0.0	720.0	195.00000
75%	217.500000	143.500000	200.000000	377.5	745.0	400.00000
max	270.000000	310.000000	410.000000	1510.0	820.0	430.00000

Pandas: Datengruppierung

```
# add a new column "method" used for grouping
data["method"] = ["credit card", "transfer", "transfer", \
           "cash", "credit card", "cash"]
```

	Noah	Liam	Emma	Mia	method
Item					
Food	270.0	140.0	188.0	200.0	credit card
Insurance	72.0	310.0	88.0	72.0	transfer
Fun	410.0	0.0	60.0	130.0	transfer
Salary	0.0	1510.0	0.0	0.0	cash
Tuition	720.0	820.0	720.0	720.0	credit card
Rent	0.0	430.0	390.0	0.0	cash

Pandas: Datengruppierung

```
data.groupby("method")
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fca90646d30>
```

```
data.groupby("method").sum()
```

method	Noah	Liam	Emma	Mia
cash	0.0	1940.0	390.0	0.0
credit card	990.0	960.0	908.0	920.0
transfer	482.0	310.0	148.0	202.0

```
data.groupby("method").max()
```

method	Noah	Liam	Emma	Mia
cash	0.0	1510.0	390.0	0.0
credit card	720.0	820.0	720.0	720.0
transfer	410.0	310.0	88.0	130.0

Pandas: Datengruppierung

```
data.groupby("method").describe() # get statistical information
```

	Noah								Liam	...
	count	mean	std	min	25%	50%	75%	max	count	...
method										...
cash	2.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	2.0	...
credit card	2.0	495.0	318.198052	270.0	382.5	495.0	607.5	720.0	2.0	...
transfer	2.0	241.0	239.002092	72.0	156.5	241.0	325.5	410.0	2.0	...

[3 rows x 32 columns]

Eigene Pandas Aufgaben

1. Ladet den Climate.csv Datensatz herunter (meine Website oder Kurswebsite)
2. Ladet es in euer Jupyterlab und lest die Datei ein.
3. Führt folgende Befehle aus:
    - 3.1 "time" als index festlegen
    - 3.2 Alle Spaltennamen ausgeben
    - 3.3 Alle Spalten von 1868
    - 3.4 Alle Jahre im März
    - 3.5 Alle Daten 1910 bis 1920
    - 3.6 Temperatur im Januar 2020
4. Plottet (mit matplotlib) die Daten von Sommer und Winter gegen alle Jahre

Pandas: In-class Exercise

Code-Experte — CSVs & Pandas: Verwenden Sie Pandas, um die CSV-Datei (auseinsteiger.csv) zu verarbeiten und benötigte Informationen abzurufen.

Schreibe ein Python Programm welches:

- Benennen Sie die Spalten um (Aufgabe 1).
- Entfernen Sie ungültige Zeilen (Aufgabe 2).
- Drucken Sie die zusammenfassenden Informationen aus (Aufgabe 3, 4).
- Geben Sie die Anzahl der Daten aus, die die Bedingungen erfüllen (Aufgabe 5, 6).
- Drucken Sie die zusammenfassenden Informationen gruppierter Daten aus (Aufgabe 7).

Eine ausführliche Aufgabenbeschreibung finden Sie in Code Expert.

3. Hausaufgaben

Übung 3: Python III

Auf `https://expert.ethz.ch/mycourses/SS25/mavt2/exercises`

Exercise 3: Python III

- Processing Earthquake Data
- Working with Data
- Bar Charts with matplotlib

KEINE HARDCODIERUNG

Fragen?

Feedback

Ich bin euch sehr dankbar für jegliche Rückmeldung, damit ich die Übungsstunde für **Euch** besser gestalten kann. Seien es Anmerkungen zu:

- **Aufbau**
- **Inhalt**
- **Redetempo**
- **Beispielen**
- **Website/Polybox**

oder anderem. Ich freue mich über **alles!**

`https://jschultev.github.io/personal_website/Feedback`