Informatik II Woche 4

Aliasing Recap, Matplotlib, Pandas

Website: n.ethz.ch/~kvaratharaja/

Die Slides basieren auf den offiziellen Übungsslides der Kurswebsite: https://lec.inf.ethz.ch/mavt/informatik2/2025/

Heute

1. **Aliasing Recap**
2. **Matplotlib**
3. **Pandas**
4. **Hausaufgaben**

Python None Keyword

- Wird verwendet um ein null Wert, bzw. kein Wert zu definieren
- None ist ein eigener Datentyp -> **nicht** das gleiche wie "None", 0, False oder ein leerer String ""!

```
x = None

if x == "":
    print('None is the same as ""!')
elif x == False:
    print('None is the same as False!')
elif x == 0:
    print('None is the same as 0!')
else:
    print('None is not "", False or 0. None is
    None!')
```

Console Output:

```
None is not "", False or 0. None is
None!
```

1. Aliasing Recap

Aliasing

- Aliasing = Zwei Variabeln zeigen auf das gleiche Objekt im Speicher
- In Python **ist alles ein Pointer!**

```
first = "Hello"
second = first #making an alias

print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))

second = "Bye" #changing the value of second

print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))
```

Console Output:

```
first: Hello, ID: 4349862704
second: Hello, ID: 4349862704
```

```
first: Hello, ID: 4349862704
second: Bye,   ID: 4308446800
```

Aliasing – Immutable Objects

Unveränderbare Objekte (z.B. Integers, Floats, Strings, Tuples, Range…) -> Kein Problem.

```
first = 1
second = first #making an alias

print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))

second += 1 #+= also changes the pointer

print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))
```

Console Output:

```
first: 1, ID: 4349862704
second: 1, ID: 4349862704
```

```
first: 1, ID: 4349862704
second: 2, ID: 4308446800
```

Aliasing - Functions

Funktionen -> Kein Problem

```python
def fun(name):
    print(f"Hello {name}!")

#create object of Test class
cheer = fun #making an alias
print("ID of fun():", id(fun))
print("ID of cheer():", id(cheer))

fun('everyone')
cheer('students')
```

Console Output:

```
ID of fun(): 4408778960
ID of cheer(): 4408778960

Hello everyone!
Hello students!
```

Aliasing – Mutable Objects

Veränderbare Objekte (z.B. Lists, Dicts, Sets ...) -> **AUFPASSEN!**

```
first = [1, 2, 3]
second = first #making an alias
print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))

second[0] = 69  #changing value in the list
print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))

first = [2*x for x in second] #changing Pointer
print("first: ", first, ", ID: ", id(first))
print("second: ", second, ", ID: ", id(second))
```

Console Output:

```
first: [1,2,3], ID: 4349862704
second: [1,2,3], ID: 4349862704

first: [69,2,3], ID: 4349862704
second: [69,2,3], ID: 4349862704

first: [138,4,6], ID: 4308446800
second: [69,2,3], ID: 4349862704
```

Aliasing - Copy

copy.copy() und copy.deepcopy()

```
import copy #import copy library
first = [[1, 2, 3], [4, 5, 6]]
second = copy.copy(first) #make a shallow copy
third = copy.deepcopy(first) #make a deep copy

first.append([7, 8, 9])
first[0][0] = 0
print("first: ", first)
print("second: ", second)
print("third: ", third)
```

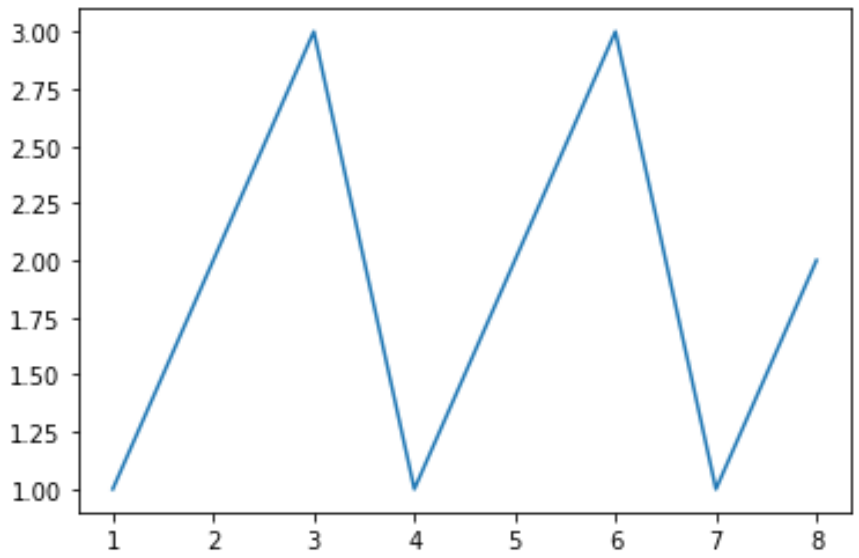
Console Output:

```
first: [[0,2,3], [4,5,6], [7,8,9]]
second: [[0,2,3], [4,5,6]]
third: [[1,2,3], [4,5,6]]
```

2.Matplotlib

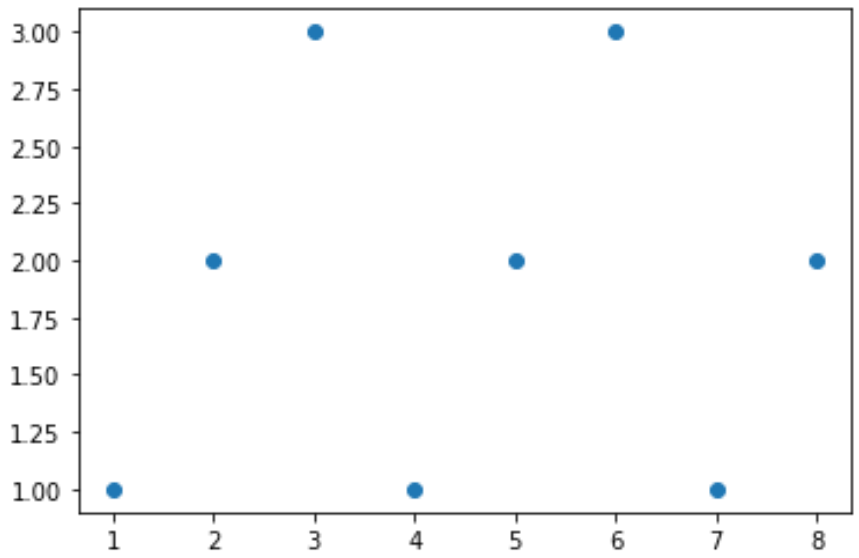
Matplotlib: Line Plot

```
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()
```



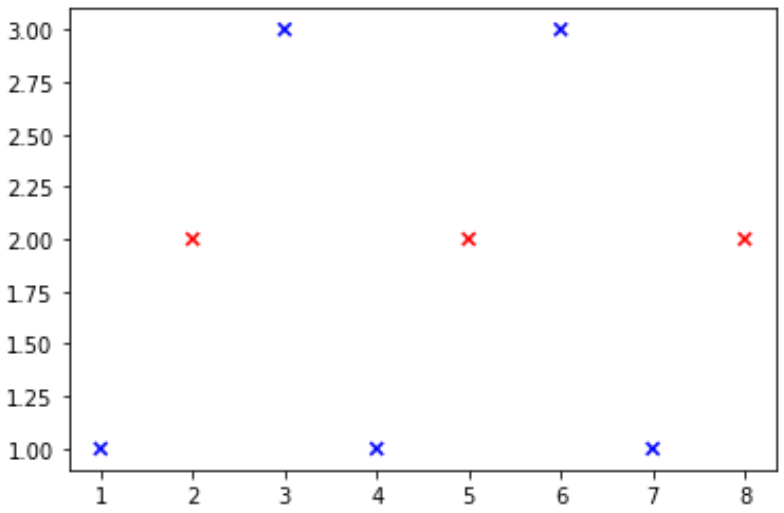
Matplotlib: Scatter Plot

```
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: Color and Marker

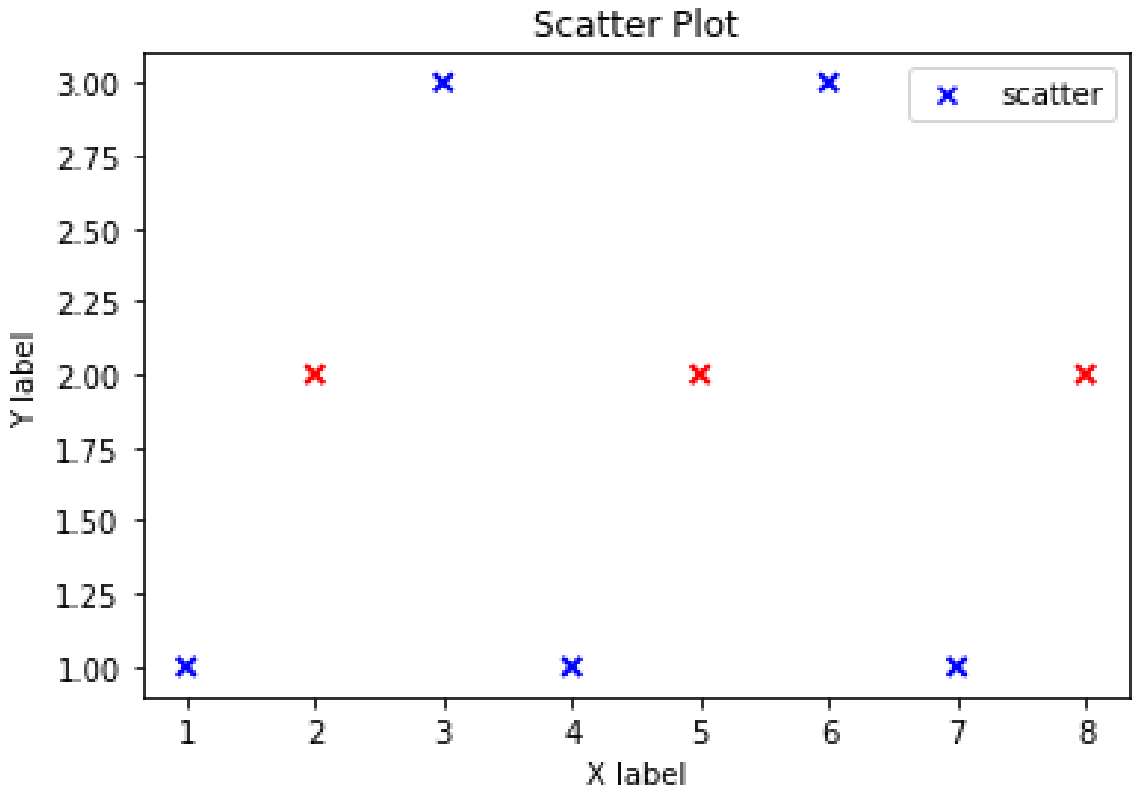
```
import matplotlib.pyplot as plt

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: Label, Title, Legend

```
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
ax.scatter(X, Y, marker = "x", c = cols)
plt.show()
```

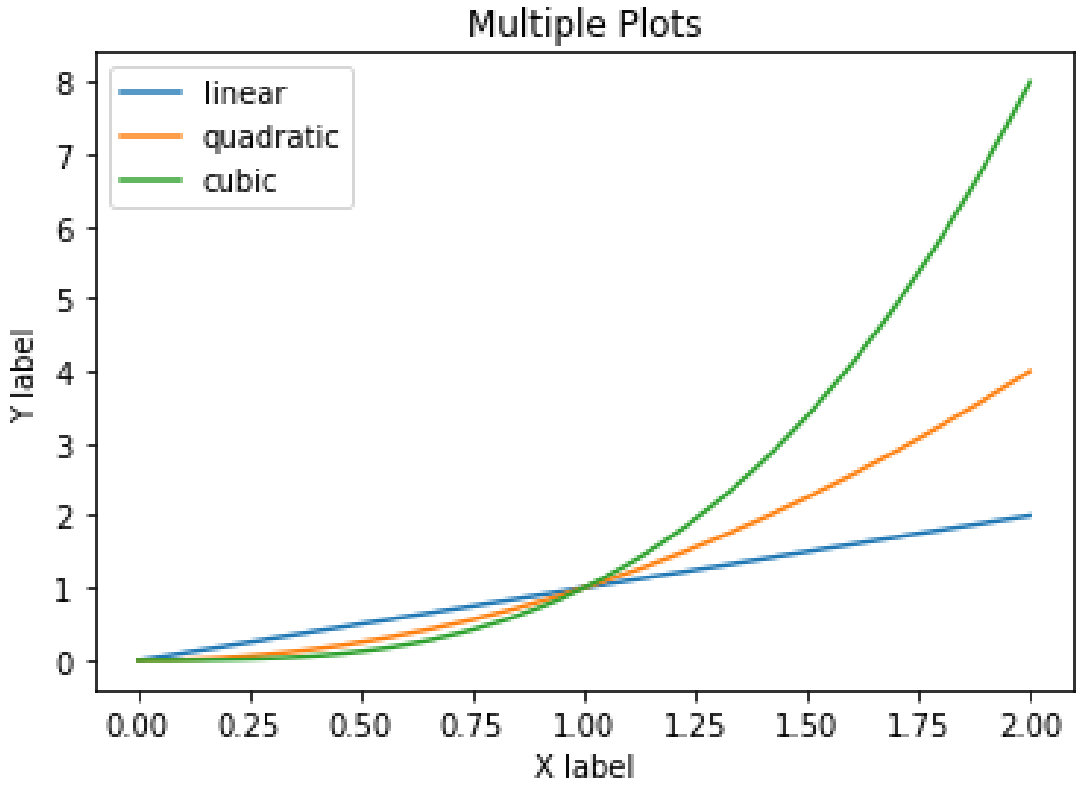
Matplotlib: Label, Title, Legend



Matplotlib: Multiple Plots

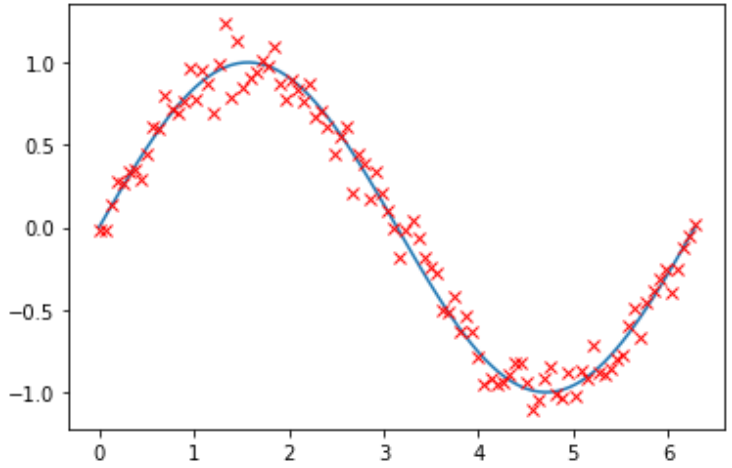
```
import matplotlib.pyplot as plt
import numpy as np

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: Multiple Plots



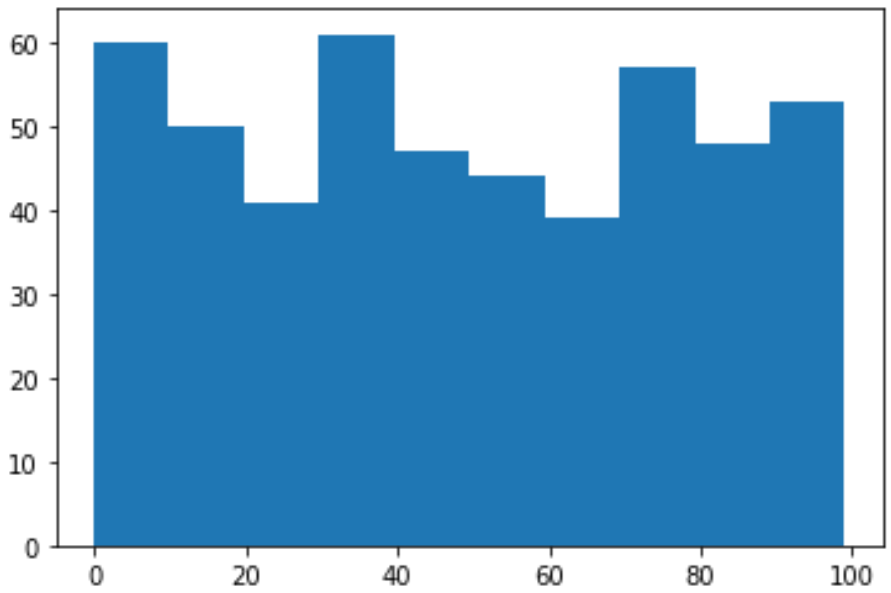
Matplotlib: Plot a Function

```
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: Histogram Plot

```
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: Get Information using "help"

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

Jupyter Notebook Aufgaben

Lade das Jupyter Notebook 'U4_matplotlib', öffne es und löse die Aufgaben.

Die detaillierte Aufgabenstellung befindet sich im Notebook.

Matplotlib: In-class Exercise

Übt das Plotten mit Matplotlib und lernen Sie, wie man Animationen erstellt.

Schreibe ein Python Programm welches:

- Streupunkte mit vorgegebenen Positionen, Grössen und Farben.
- Aktualisieren Sie die Streudaten bei jedem Frame.

Eine ausführliche Aufgabenbeschreibung ist im Jupyter Notebook 'Plot In-Class Exercise'.

3. Pandas

All my homies love Pandas 🐼

Pandas: DataFrame

- Zwei Dimensionale Tabelle mit variabler Grösse und Achsen mit Labels (Reihen und Spalten)
- Daten in den Spalten (Columns) haben den gleichen Typ

Column Labels (Index)

Row Labels (Index)		0	1	2	3
	Item	noah	liam	emma	mia
0	Food	270	140	188	200
1	Insurance	72	310	88	72
2	Fun	410	0.0	60	130
3	Salary	0.0	1510	0.0	0.0
4	Tuition	720	820	720	720
5	Rent	0.0	430	390	0.0

Data

Pandas: CSV Data

- CSV steht für **Comma Separated Values**.
- Ein CSV file enthält eine Tabelle von Daten
- CSV is ein Textformat. Zeilen werden durch ein Zeilenumbruchzeichen getrennt
- Spalten werden durch ein Komma oder ein anderes Zeichen getrennt (Oftmals durch ein Tab `\t`).

```
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: Read CSV Data

```
import pandas as pd
```

Daten aus csv-Datei lessen:

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

	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: Read CSV Data

Hard-coded Daten einlesen:

```
import io
csv_file = io.StringIO("""
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;-
""")
data = pd.read_csv(csv_file, sep=";", index_col="Item")
```

Pandas: Rename Columns

Die "rename" Funktion verwenden:

```
data = data.rename(columns={
"noah": "Noah", "liam": "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: Rename Columns

Column (Spalten) Namen direkt setzen

```
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: Deal with Invalid Data

- Normalerweise erkennt Pandas Spalten mit Zahlen und setzt den Datentyp der Spalte automatisch auf **numeric**.
- Leider sind unsere Spalten nicht numerisch, da sie "-" Zeichen enthalten.

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

Console Output:

```
int64
```

```
print(data["Noah"].dtype) #check Data type for Noah column
print(data["Liam"].dtype) #check Data type for Liam column
print(data["Emma"].dtype) #check Data type for Emma column
print(data["Mia"].dtype) #check Data type for Mia column
```

```
object
object
object
object
```

Pandas: Deal with Invalid Data

Spalten numeric machen:

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

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	-

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

Pandas: Deal with Invalid Data

Zeilen mit NaN-Werten löschen

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

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
Tuition	720.0	820.0	720.0	720.0

Pandas: Deal with Invalid Data

Die NaN Werte mit einem anderen Wert ersetzen, 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: Data Filtering

Daten mit einer condition filtern

	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	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

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

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

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

	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	0.0	60.0	130.0
Tuition	720.0	820.0	720.0	720.0

Pandas: Data Filtering

Daten mit mehreren conditions filtern

	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	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

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

	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: Data Filtering

Bestimmte Zeilen ab-/auswählen

	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	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

```
data[(data.index != "Fun")
```

	Noah	Liam	Emma	Mia
Item				
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"], :]
```

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

Pandas: Data Modifying

Neue Spalte hinzufügen:

```
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: Data Modifying

Spalte entfernen:

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

	Noah	Liam	Emma	Mia	Total
Item					
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

	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	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: Data Summarising

- Alle Spalten aufsummieren

```
data.sum()
```

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

- Das Maximum von jeder Spalte finden

```
data.max()
```

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

- Noah Spalte aufsummieren

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

```
1472.0
```

- Das Maximum von der Noah Spalte finden

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

```
1472.0
```

Pandas: Data Summarising

Mehrere aggregate Funktionen auf das Datenset anwenden:

```
data.agg("max", "sum")
```

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

Pandas: Data Summarising

Statistiken zum Datenset erzeugen:

```
data.T.describe() #transpose data first
```

Item	Food	Insurance	Fun	Salary	Tuition	Rent
count	4.0	4.0	4.0	4.0	4.0	4.0
mean	199.5	135.5	150.0	377.5	745.0	205.0
std	53.674948	116.577585	181.291662	755.0000	50.000000	237.276210
min	140.0	72.0	0.0	0.0	720.0	0.0
25%	176.0	72.0	45.0	0.0	720.0	0.0
50%	194.0	80.0	95.0	0.0	720.0	195.0
75%	217.5	143.5	200.0	377.5	745.0	400.0
max	270.0	310.0	410.0	1510.0	820.0	430.0

Pandas: Data Grouping

Neue Spalte "method" hinzufügen, die zum gruppieren verwendet wird:

```
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: Data Grouping

Das ganze Datenset nach "method" gruppieren

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

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

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

	Noah	Liam	Emma	Mia
method				
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()
```

	Noah	Liam	Emma	Mia
method				
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: Data Summarising

Statistiken zur "method" Gruppe erhalten

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

	Noah								Liam	...
	count	mean	std	min	25%	50%	75%	max	count	...
method										...
cash	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...
credit card	2.0	495.0	318.2	270.0	382.5	495.0	607.5	720.0	2.0	...
transfer	2.0	241.0	239.0	72.0	156.6	241.0	325.5	410.0	2.0	...

In-class exercise 2: CSVs & Pandas

Gehe zu Code Expert -> Code Examples -> CSV & Pandas: Verwende Pandas um ein CSV file (auseinsteiger.csv) zu bearbeiten und Informationen zu bekommen

Schreibe ein Python Programm, welches:

- Die Spalten umbenennt (Task 1).
- Die ungültigen Zeilen entfernt (Task 2).
- Zusammenfassende Informationen ausgibt(Task 3, 4).
- Die Anzahl der Daten ausgibt, welche gewisse Bedingungen erfüllt (Task 5, 6)
- Zusammenfassende Informationen der gruppierten Daten ausgibt (Task 7)

Detaillierte Aufgabenbeschreibung -> siehe Code Expert

4. Hausaufgaben

Exercise 3: Python III

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

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

Abgabe bis spätestens: Monday 17.03.2023, 20:00 CET

NO HARDCODING

Feedback?

Zu schnell? Zu langsam? Weniger Theorie, mehr Aufgaben?
Dankbar für Feedback am besten mir direkt sagen oder Mail schreiben

Credits

Die Slide(-templates) stammen ursprünglich von Julian Lotzer und Daniel Steinhauser, vielen Dank!

→Checkt ihre Websites ab für zusätzliches Material in Informatik I, Informatik II und Stochastik & Machine Learning.

- https://n.ethz.ch/~jlotzer/
- https://n.ethz.ch/~dsteinhauser/