

Werkzeuge für das wissenschaftliche Arbeiten

Python for Machine Learning and Data Science

Magnus Bender
bender@ifis.uni-luebeck.de
Wintersemester 2023/24

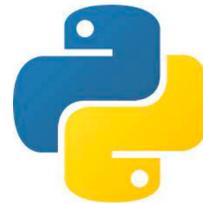
Inhaltsübersicht

1. Programmiersprache Python

a) *Einführung, Erste Schritte*

b) *Grundlagen*

c) *Fortgeschritten*



2. Auszeichnungssprachen

a) *LaTeX, Markdown*

L^AT_EX



3. Benutzeroberflächen und Entwicklungsumgebungen

a) *Jupyter Notebooks lokal und in der Cloud (Google Colab)*

4. Versionsverwaltung

a) *Git, GitHub*



5. Wissenschaftliches Rechnen

a) *NumPy, SciPy*



6. Datenverarbeitung und -visualisierung

a) **Pandas, matplotlib, NLTK**

7. Machine Learning (scikit-learn)

a) *Grundlegende Ansätze (Datensätze, Auswertung)*

b) *Einfache Verfahren (Clustering, ...)*



8. DeepLearning

a) *TensorFlow, PyTorch, HuggingFace Transformers*



Themen

- I. Projektaufgabe 2
 1. Lösungsvorschlag
- II. Datenverarbeitung
 1. Python-intern
 2. Pandas
 3. NLTK
- III. Datenvisualisierung
 1. Matplotlib



Heute

Projektaufgabe 2

„Objektorientierung in Python“

- Vorstellung möglicher Lösung
(wird nicht ins Moodle hochgeladen!)



II.

Datenverarbeitung

1. Python-intern

Ablauf

Datenverarbeitung



- JSON
- CSV
- Numpy (npz, npy)
- TXT
- ...



- JSON
- Pickle
- Shelve
- Numpy (npz, npy)

- Tabellen
- Grafiken

Datenvisualisierung

JavaScript Object Notation (JSON)

```
import json
```

```
d = {  
    "a" : 1,  
    "b" : "B",  
    "c" : [1.2, 1.3, 1.4],  
    "e" : None  
}
```

```
json_str = json.dumps(d, indent=2)  
print(json_str)
```

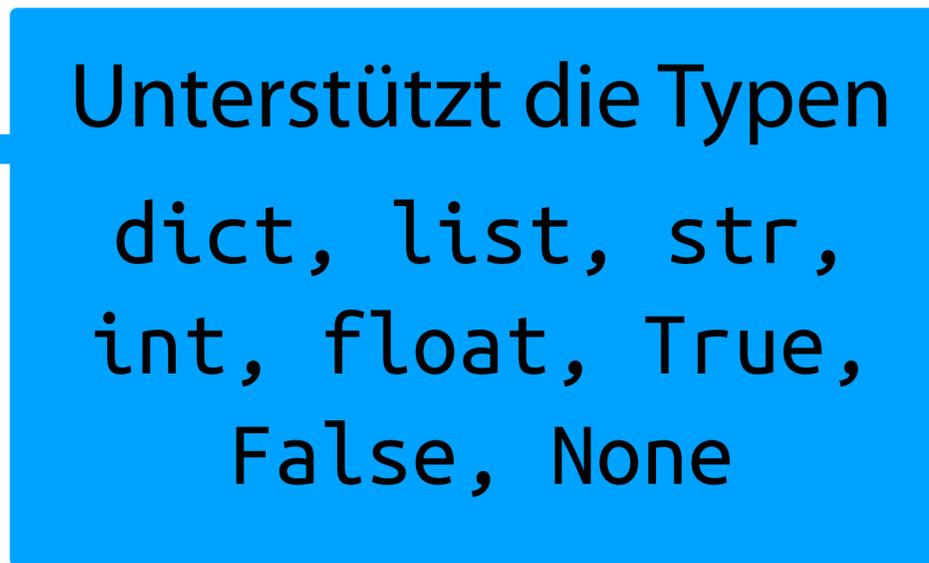
```
d_ = json.loads(json_str)  
print(d_)
```

```
{'a': 1, 'b': 'B', 'c': [1.2, 1.3, 1.4], 'e': None}
```

```
{  
  "a": 1,  
  "b": "B",  
  "c": [  
    1.2,  
    1.3,  
    1.4  
  ],  
  "e": null  
}
```



Unterstützung in (fast) allen Programmiersprachen



Unterstützt die Typen dict, list, str, int, float, True, False, None

Pickle

```
import pickle
```

```
class Triple():  
    def __init__(self, a, b, c):  
        self.a, self.b, self.c = a, b, c  
    def __repr__(self):  
        return str((self.a, self.b, self.c))
```

```
d = [  
    Triple(1, 2, 3),  
    Triple(4, 5, 6),  
    Triple(7, 8, 9),  
]
```

```
pickle_data = pickle.dumps(d)  
d_ = pickle.loads(pickle_data)  
print(d, d_)
```

```
[(1, 2, 3), (4, 5, 6), (7, 8, 9)]  
[(1, 2, 3), (4, 5, 6), (7, 8, 9)]
```

```
pickle.dump(d, open("d.pickle", "wb"))  
d__ = pickle.load(open("d.pickle", "rb"))  
print(d__)
```

```
[(1, 2, 3), (4, 5, 6), (7, 8, 9)]
```

Achtung, keine Pickles aus unbekannter Quelle laden!

Unterstützt alle Typen, Implementierung muss verfügbar und gleich sein!

Daten sind ein binärer Datenstrom

Direkt in eine Datei schreiben und daraus lesen (ohne s).

II.

Datenverarbeitung

2. Pandas

Installation



- Python Paket
- <https://pandas.pydata.org/>
- Installation z.B. mit pip3 `install pandas`
- Import

```
import pandas  
import pandas as pd
```

Pandas & DataFrame

- Verwaltung und Manipulation von Tabellen
- Verschiedene Datentypen kombiniert
- Geschwindigkeit durch C-Code im Hintergrund
- Import von CSV, JSON, SQL, Excel, ...

| ID | Name | Age | Species |
|----|-------|-----|---------|
| 1 | Otto | 60 | Human |
| 2 | Erna | 65 | Human |
| 3 | Bello | 10 | Dog |
| | 3 | 45 | 2 |

DataFrame

```
import pandas as pd
```

```
df = pd.DataFrame({  
    "ID" : pd.Index([1, 2, 3]),  
    "Name" : ["Otto", "Erna", "Bello"],  
    "Age" : [60, 65, 10],  
    "Species" : pd.Categorical(["Human", "Human", "Dog"])  
})
```

```
print(df)
```

| | ID | Name | Age | Species |
|---|----|-------|-----|---------|
| 0 | 1 | Otto | 60 | Human |
| 1 | 2 | Erna | 65 | Human |
| 2 | 3 | Bello | 10 | Dog |

```
print(df.head(1))
```

| | ID | Name | Age | Species |
|---|----|------|-----|---------|
| 0 | 1 | Otto | 60 | Human |

```
print(df.tail(2))
```

| | ID | Name | Age | Species |
|---|----|-------|-----|---------|
| 1 | 2 | Erna | 65 | Human |
| 2 | 3 | Bello | 10 | Dog |

```
print(df.dtypes)
```

```
ID          int64  
Name        object  
Age         int64  
Species     category  
dtype: object
```

```
print(df.index)
```

```
RangeIndex(start=0, stop=3, step=1)
```

```
print(df.columns)
```

```
Index(['ID', 'Name', 'Age', 'Species'], dtype='object')
```

| ID | Name | Age | Species |
|----|-------|-----|---------|
| 1 | Otto | 60 | Human |
| 2 | Erna | 65 | Human |
| 3 | Bello | 10 | Dog |

```
print(df.describe())
```

| | ID | Age |
|-------|-----|-----------|
| count | 3.0 | 3.000000 |
| mean | 2.0 | 45.000000 |
| min | 1.0 | 10.000000 |
| 50% | 2.0 | 60.000000 |
| max | 3.0 | 65.000000 |

```
print(df.to_numpy())
```

```
[[1 'Otto' 60 'Human']  
 [2 'Erna' 65 'Human']  
 [3 'Bello' 10 'Dog']]
```

Fortsetzung

Ich kann hier nicht alle Funktionen vorstellen, daher Verweise ich auf weitere Quellen

D.h. natürlich auch, dass Pandas dann nicht in den Aufgaben drankommt.



- Import und Export
- Selektion (Slices, Indexe)
- Operation (Statistik, Funktionen anwenden)
- Zusammenfügen und Umformen
- Visualisierung

→ https://pandas.pydata.org/docs/user_guide/10min.html

→ https://pandas.pydata.org/docs/user_guide/index.html

II.

Datenverarbeitung

3. Natural Language Toolkit (NLTK)

Problem?

Sprachverarbeitung

„Geben Sie **den** Link zu Ihrem **Repository** im Moodle ein. **Dabei** werden Sie auch gefragt, ob **das Repository** öffentlich oder verborgen ist! Denken Sie bei verborgen **Repositories** daran, **dem** GitHub-User **Werkzeuge** **WissArbeiten** Zugriffsrechte für Ihr erstelltes **Git-Repository** zu gewähren.“

- Sätze aufsplitten
- Wortvektor(en) erstellen
 - Häufigkeit der Worte pro Satz
 - Wenig aussagekräftige Worte

Ich kann hier nur ein paar Ideen anreißen.

Vorverarbeitung (*Preprocessing*) ist eine sehr wichtige Aufgabe u.a. bei der Sprachverarbeitung (NLP).

Vorverarbeitung mit NLTK I

```
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer
```

```
text = "... Repository öffentlich oder verborgen ist! Denken Sie bei verborgen Repositories ..."
```

```
stop_words = set(stopwords.words("german"))
print(stop_words)
```

```
{'nun', 'weil', 'nichts', 'sie', 'solche',
'einem', 'warst', 'werden', 'das', ... }
```

```
stemmer_de = SnowballStemmer("german")
stemmer_en = SnowballStemmer("english")
```

```
print(stemmer_de.stem("öffentlich"))
print(stemmer_de.stem("Öffentlichkeit"))
```

```
öffent
öffent
```

```
print(stemmer_en.stem("repository"))
print(stemmer_en.stem("repositories"))
```

```
repositori
repositori
```

Stopwords: Worte die eher wenig zur Kernaussage eines Satzes beitragen.

Wort auf dem Wortstamm reduzieren, sodass es als das selbe Wort betrachtet werden kann.

Vorverarbeitung mit NLTK II

```
text = "Geben Sie den Link zu Ihrem Repository im ..."
```

```
stop_words = set(stopwords.words("german"))  
stemmer_de = SnowballStemmer("german")  
stemmer_en = SnowballStemmer("english")
```

```
text = text.replace("-", " ")
```

```
for sentence in sent_tokenize(text, language='german'):  
    print(sentence)
```

```
Geben Sie den Link zu Ihrem Repository im Moodle ein.
```

```
for word in word_tokenize(sentence, language='german'):  
    print(word)
```

Über Sätze und dann Worte iterieren.

```
Geben  
Sie  
den  
Link  
zu  
Ihrem  
Repository  
im  
Moodle  
ein  
.
```

Vorverarbeitung mit NLTK III

```
text = " ... Dabei werden Sie auch gefragt, ob das Repository öffentlich oder verborgen ist! ..."  
stop_words = set(stopwords.words("german"))  
stemmer_de = SnowballStemmer("german")  
stemmer_en = SnowballStemmer("english")
```

```
text = text.replace("-", " ")  
for sentence in sent_tokenize(text, language='german'):  
    for word in word_tokenize(sentence, language='german'):  
        print(word)
```

```
word = word.lower()  
word = re.sub("[^a-zA-Züöß]", "", word)  
print(word)
```

```
if word not in stop_words:  
    word = stemmer_de.stem(word)  
    word = stemmer_en.stem(word)
```

```
print(word)
```

Alles klein und nur
Buchstaben

Wortstämme, sowohl
Englisch als auch
Deutsch

| | | |
|------------|------------|------------|
| Dabei | dabei | dabei |
| werden | werden | |
| Sie | sie | |
| auch | auch | |
| gefragt | gefragt | gefragt |
| , | | |
| ob | ob | |
| das | das | |
| Repository | repository | repositori |
| öffentlich | öffentlich | offent |
| oder | oder | |
| verborgen | verborgen | verborg |
| ist | ist | |
| ! | | |

Vorverarbeitung mit NLTK IV

```
text = "Geben Sie den Link zu Ihrem Repository im Moodle ein. ..."
```

```
stop_words = set(stopwords.words("german"))
```

```
stemmer_de, stemmer_en = SnowballStemmer("german"), SnowballStemmer("english")
```

```
sentences = []
```

```
for sentence in sent_tokenize(text.replace("-", " "), language='german'):
```

```
    words = []
```

```
    for word in word_tokenize(sentence, language='german'):
```

```
        word = re.sub("[^a-zäüöß]", "", word.lower())
```

```
        if word not in stop_words:
```

```
            word = stemmer_en.stem(stemmer_de.stem(word))
```

```
            if len(word) > 0:
```

```
                words.append(word)
```

```
    sentences.append(words)
```

```
print(sentences)
```

Alles zusammen :-)

```
[['geb', 'link', 'repositori', 'moodl'],  
 ['dabei', 'gefragt', 'repositori', 'offent', 'verborg'],  
 ['denk', 'verborg', 'repositori', 'daran', 'github', 'user',  
  'werkzeugewissarbeit', 'zugriffsrecht', 'erstellt', 'git',  
  'repositori', 'gewahr']]
```

Ergebnis

- *Bag-of-Words-Modell*
 - Dokument ist ein Bitvektor (oder auch Häufigkeitsvektor) seiner Wörter
 - Text sind jetzt Vektoren

Und die können wir vergleichen etc.

| Wort | Satz 0 | Satz 1 | Satz 2 |
|---------------------|--------|--------|--------|
| dabei | 0 | 1 | 0 |
| daran | 0 | 0 | 1 |
| denk | 0 | 0 | 1 |
| erstellt | 0 | 0 | 1 |
| geb | 1 | 0 | 0 |
| gefragt | 0 | 1 | 0 |
| gewahr | 0 | 0 | 1 |
| git | 0 | 0 | 1 |
| github | 0 | 0 | 1 |
| link | 1 | 0 | 0 |
| moodl | 1 | 0 | 0 |
| offent | 0 | 1 | 0 |
| repositori | 1 | 1 | 2 |
| user | 0 | 0 | 1 |
| verborg | 0 | 1 | 1 |
| werkzeugewissarbeit | 0 | 0 | 1 |
| zugriffsrecht | 0 | 0 | 1 |

Ergebnis als NumPy

Auch hierfür würde man i.A. eine Bibliothek nutzen

```
import numpy as np
```

```
sentences = [  
    ['geb', 'link', 'repositori', 'moodl'],  
    ['dabei', 'gefragt', 'repositori', 'offent', 'verborg'],  
    ['denk', 'verborg', 'repositori', 'daran', 'github',  
     'user', 'werkzeugewissarbeit', 'zugriffsrecht', 'erstellt',  
     'git', 'repositori', 'gewahr']  
]
```

```
id2word = list(set([ w for s in sentences for w in s ]))  
word2id = { word : i for i, word in enumerate(id2word) }
```

```
corpus = np.zeros((len(id2word), len(sentences)), dtype=int)  
for s_id, sentence in enumerate(sentences):  
    for word in sentence:  
        corpus[word2id[word], s_id] += 1
```

```
print(word2id) {'dabei': 0, 'daran': 1, ... 'repositori': 12, 'user': 13,  
print(corpus)  'verborg': 14, 'werkzeugewissarbeit': 15, ...}
```

```
[[0 1 0]  
 [0 0 1]  
 [0 0 1]  
 [0 0 1]  
 [1 0 0]  
 [0 1 0]  
 [0 0 1]  
 [0 0 1]  
 [0 0 1]  
 [1 0 0]  
 [1 0 0]  
 [0 1 0]  
 [1 1 2]  
 [0 0 1]  
 [0 1 1]  
 [0 0 1]  
 [0 0 1]]
```

NumPy-Array und ein Wörterbuch um IDs auf Worte zu übersetzen.

III.

Datenvisualisierung

Matplotlib

Installation



- Python Paket
 - <https://matplotlib.org/>
- Installation z.B. mit pip3 `install matplotlib`
- Import

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Der erste Plot

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

x = np.arange(-10, 10.5, step=0.5)

plt.plot(x, x**2)
plt.plot(x, x**3)

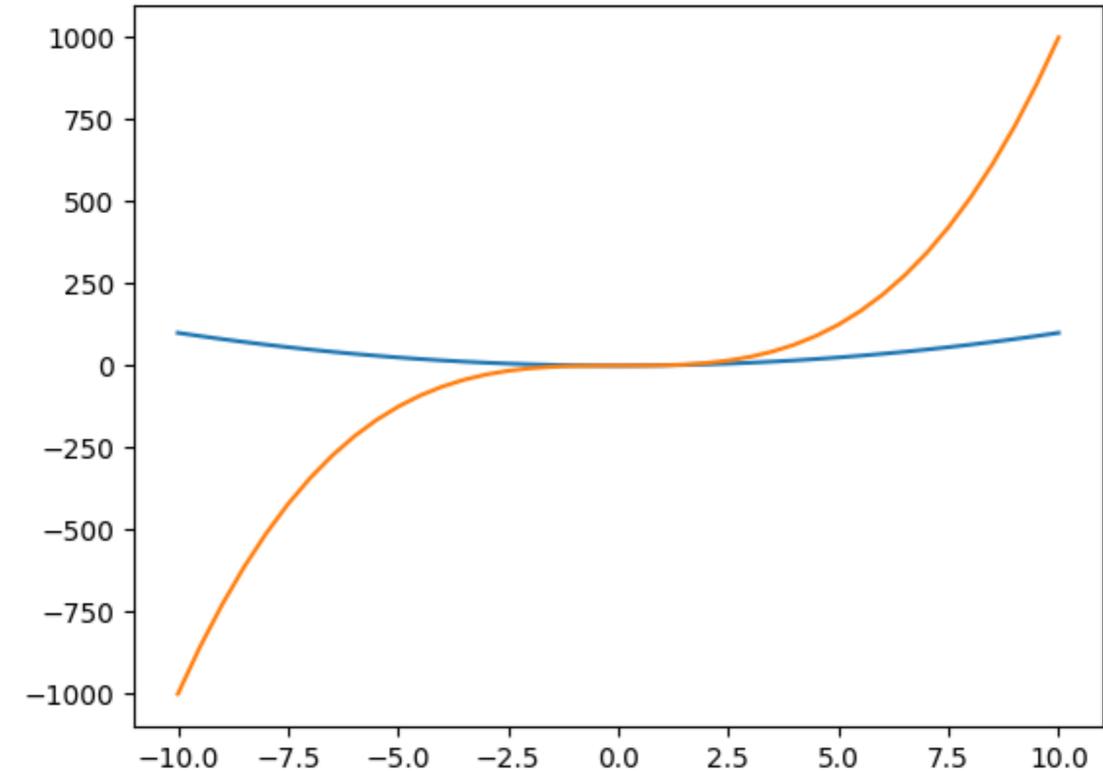
plt.show()

print(x)
```

„Linie“ mittels zwei Arrays, jeweils x- und y-Wert

Öffnen des Plot

```
[ -10.  -9.5  -9.   -8.5  -8.   -7.5  -7.   -6.5  -6.   -5.5  -5.   -4.5
  -4.   -3.5  -3.   -2.5  -2.   -1.5  -1.   -0.5  0.    0.5   1.    1.5
   2.    2.5   3.    3.5   4.    4.5   5.    5.5   6.    6.5   7.    7.5
   8.    8.5   9.    9.5  10. ]
```



PyPlot und Objektorientiert

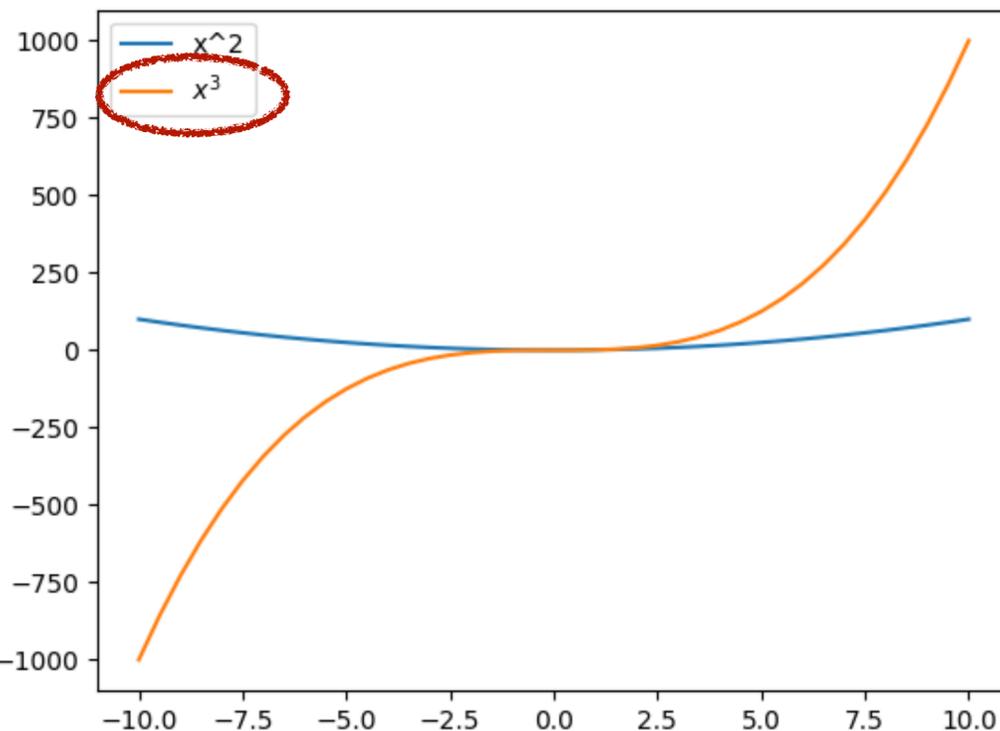
```
import matplotlib.pyplot as plt
import numpy as np
```

```
x = np.arange(-10, 10.5, step=0.5)
```

```
plt.plot(x, x**2, label='x^2')
plt.plot(x, x**3, label='$x^3$')
plt.legend()
```

PyPlot: Direkt plt genutzt

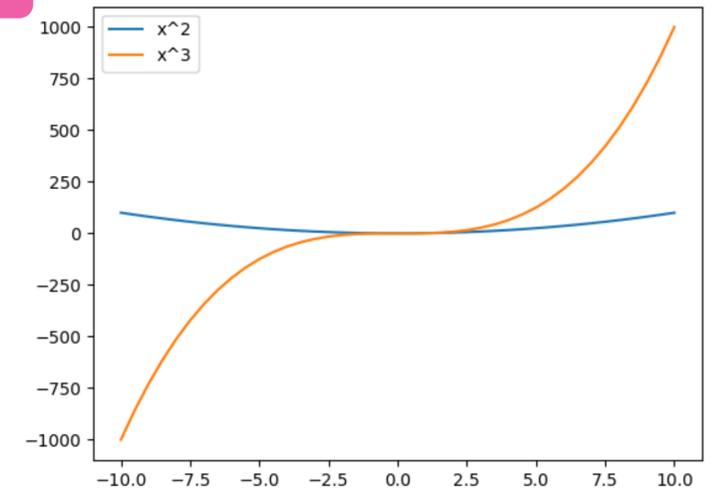
```
plt.show()
```



Objektor.: Objekt erstellt und dann dieses genutzt.

```
fig1, ax1 = plt.subplots()
```

```
ax1.plot(x, x**2, label='x^2')
ax1.plot(x, x**3, label='x^3')
ax1.legend()
```

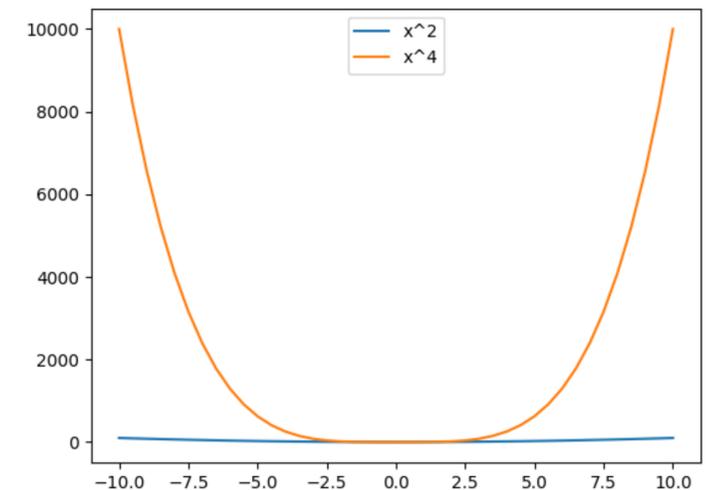


```
fig2, ax2 = plt.subplots()
```

```
ax2.plot(x, x**2, label='x^2')
ax2.plot(x, x**4, label='x^4')
ax2.legend()
```

```
plt.show()
```

Zwei Objekte, also es öffnet sich ein Fenster mit zwei Plots.



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

```
y = np.array([1, 2, 3, 4, 5, 4, 3, 2, 1])
x = np.arange(9)
```

```
fig, ax = plt.subplots(figsize=(8, 4))
```

```
ax.plot(x, y, label='y')
ax.plot(x, y+0.5, label='y+0.5', linewidth=3, linestyle='--')
ax.plot(x, y+1.5, 'o', label='y+1.5')
ax.bar(x, y-0.5)
```

```
ax.legend();
```

```
ax.set_xlabel("X-Achse")
ax.set_ylabel("Y-Achse")
ax.set_title('Eine Grafik mit Punkten,
Linien und Balken')
```

```
ax.text(2, 6, 'Ein Text!')
```

```
plt.savefig("plot.png")
```

Beispiel

Größe (Seitenverhältnis des Plot)

Konfiguration der Linie und Dicke

Kreise statt Linie

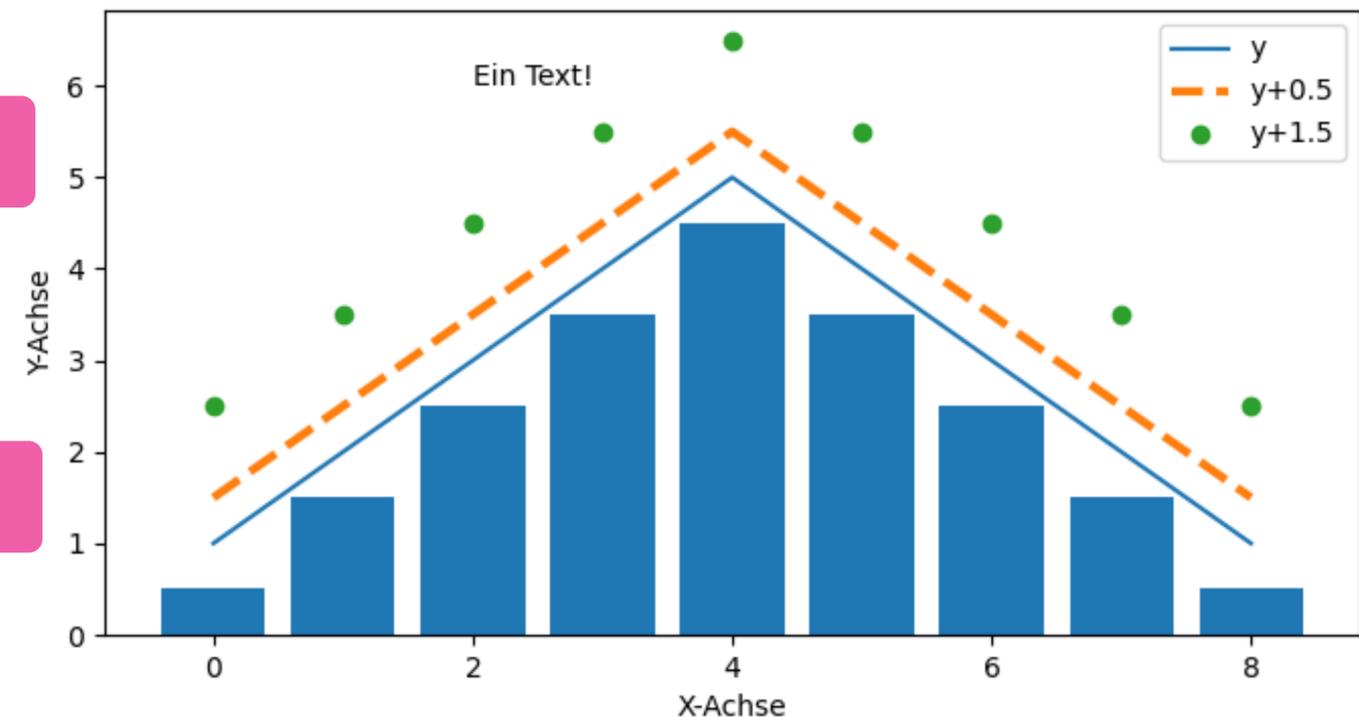
Balkendiagramm

Beschriftungen

Text im Plot

Als Datei speichern statt zu öffnen

Eine Grafik mit Punkten, Linien und Balken



Zusammenfassung

I. Projektaufgabe 2

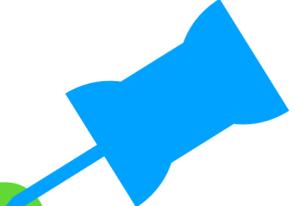
1. Lösungsvorschlag

II. Datenverarbeitung

1. Python-intern
2. Pandas
3. NLTK

III. Datenvisualisierung

1. Matplotlib



Aufgabe 4 wird am
Donnerstag freigeschaltet
und am Freitag ist die
Abgabe von Aufgabe 3!



~~Heute~~

Inhaltsübersicht

1. Programmiersprache Python
 - a) *Einführung, Erste Schritte*
 - b) *Grundlagen*
 - c) *Fortgeschritten*
2. Auszeichnungssprachen
 - a) *LaTeX, Markdown*
3. Benutzeroberflächen und Entwicklungsumgebungen
 - a) *Jupyter Notebooks lokal und in der Cloud (Google Colab)*
4. Versionsverwaltung
 - a) *Git, GitHub*
5. Wissenschaftliches Rechnen
 - a) *NumPy, SciPy*
6. Datenverarbeitung und -visualisierung
 - a) *Pandas, matplotlib, NLTK*
7. Machine Learning (scikit-learn)
 - a) **Grundlegende Ansätze (Datensätze, Auswertung)**
 - b) *Einfache Verfahren (Clustering, ...)*
8. DeepLearning
 - a) *TensorFlow, PyTorch, HuggingFace Transformers*