Python: Datenverarbeitung mit NumPy, Matplotlib und Pandas

Softwareentwicklung in der Wissenschaft

Larissa Lach

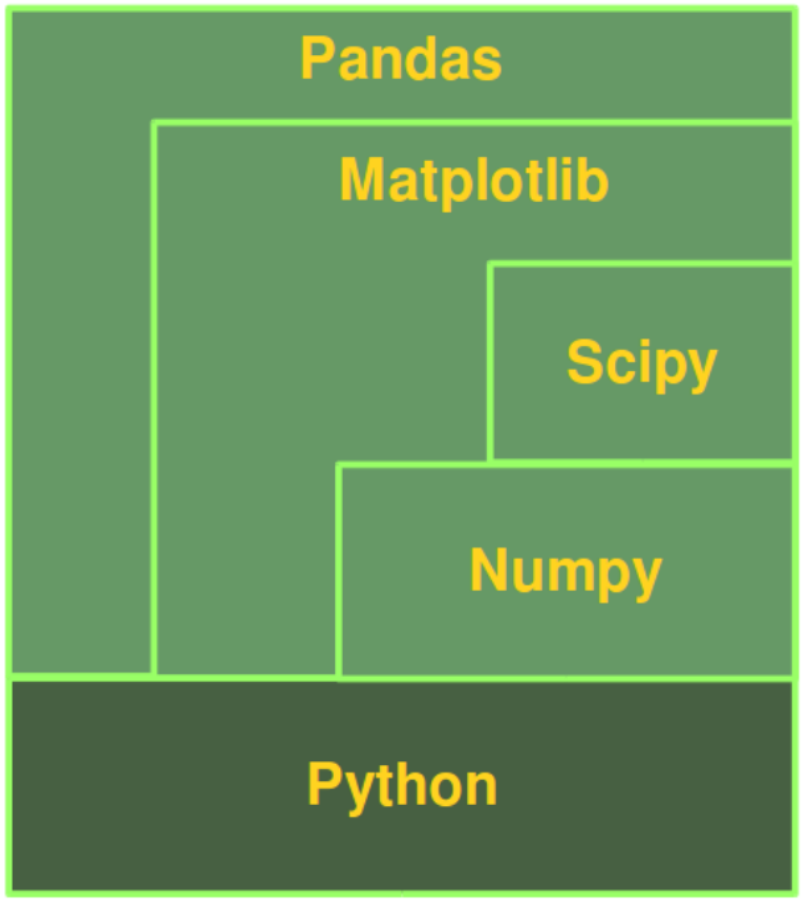
08.06.2020

Agenda

- Einleitung
- Numpy
  - Vorteile
  - Funktionen/ufunc
  - Beispiel
- Matplotlib
  - Allgemeines
  - Beispiel Fortsetzung
- Pandas
  - Daten-Strukturen
  - Fehlende Daten
  - Beispiel
- Zusammenfassung
- Literatur

Wozu Packages?

- Python ohne Erweiterungen: sehr langsam
- Python in Verbindung mit verschiedenen Packages
  → genauso effizient/ effizienter als Konkurrenz
- Pandas: Verarbeitung von Tabellen
- Matplotlib: grafische Darstellung
- SciPy: erweitert die Leistungsfähigkeit
- Numpy: mehrdimensionale Arrays und Matrizen
- Python: grundlegende Datenstruktur



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

Numpy (numeric Python)

- Backend für einige andere Pakete
- ndarray: multidimensionales Array
- Numpy vs. Listen → Numpy deutlich schneller
  Bsp.: 5

mit Numpy	32 Bit
mit Listen	
Größe:	32 Bit
Referenz:	64 Bit
Typ:	64 Bit
Wert:	64 Bit

- Listen nutzen Referenzen auf Speicherblöcke
  Numpy Arrays speichern zusammenhängend
- Größtenteils in C geschrieben

Larissa Lach

Funktionen

- Allgemeine Funktionen
  - Größe, Form, Dimension etc.
- Array Funktionen
  - Auf Daten zugreifen (slicing)
  - Array-Algorithmen
- Mathematische Funktionen
  - Komplexe Berechnungen auf ganzen Arrays
  - Logische Bedingungen
  - Funktionen der linearen Algebra
- Maskieren
- Fancy Indexing

```
import numpy as np

arr = np.array([[[1, 2, 3],
                 [4, 5, 6]],
                [[7, 8, 9],
                 [10, 11, 12]]], dtype="int16")

dimension = arr.ndim                3
form = arr.shape                    (2, 2, 3)
typ = arr.dtype                     int16
element = arr[1, 0, 2]              9
reihe = arr[0, 0, :]                [1 2 3]

reshape = arr.reshape((3, 4))       [[ 1  2  3  4]
                                     [ 5  6  7  8]
                                     [ 9 10 11 12]]
```


Larissa Lach

numpy.ufunc

- Geschwindigkeitsvorteil von C
- Keine Python-for-Schleifen benötigt
- Vektorisierung
  - Ohne ufunc: durchiterieren der Arrays
  - Mit ufunc: vektorbasierte Operation
    → schneller mit modernen CPUs
- Broadcasting
  - Wird benötigt um mit Eingaben verschiedener Formen arbeiten zu können

```
x = [[0],
     [1],
     [2],
     [3]]
y = [1, 1, 1, 1, 1]

z = np.add(x, y)
```

```
= [[0, 0, 0, 0, 0],
   [1, 1, 1, 1, 1],
   [2, 2, 2, 2, 2],
   [3, 3, 3, 3, 3]]
= [[1, 1, 1, 1, 1],
   [1, 1, 1, 1, 1],
   [1, 1, 1, 1, 1],
   [1, 1, 1, 1, 1]]
```

```
z = [[1, 1, 1, 1, 1],
     [2, 2, 2, 2, 2],
     [3, 3, 3, 3, 3],
     [4, 4, 4, 4, 4]]
```


Larissa Lach

Beispiel - Wetterdaten

```
1.5.   2.5.   3.5.   4.5.   5.5.   6.5.   7.5.   8.5.   9.5.   10.5.   11.5.   12.5.   13.5.   14.5.   15.5.   16.5.   17.5.   18.5.   19.5.   20.5.   21.5.   22.5.   23.5.   24.5.   25.5.   26.5.   27.5.   28.5.   29.5.   30.5.   31.5.
6.5   7.3   5.6   3.5   1.3   0.7   4.0   1.7   5.8   6.6   3.5   2.2   3.5   1.4   -0.4   5.8   6.2   11.1   7.4   4.4   8.0   12.2   8.6   8.5   8.5   8.2   7.0   5.3   5.1   7.5   8.2
14.5   13.6   14.8   14.2   14.0   14.3   15.1   19.3   23.0   24.2   11.6   10.6   11.5   12.2   12.7   15.9   17.2   17.0   19.0   20.0   22.0   22.2   17.0   16.0   15.2   20.3   19.7   20.0   21.9   21.2   20.2
10.50   10.45   10.20   8.85   7.65   7.50   9.55   10.50   14.40   15.40   7.55   6.40   7.50   6.80   6.15   10.85   11.70   14.05   13.20   12.20   15.00   17.20   12.80   12.25   11.85   14.25   13.35   12.65   13.50   14.35   14.20
4.1   2.6   2.2   11.1   9.9   14.1   6.6   12.4   8.7   8.7   5.2   5.2   4.2   10.1   5.1   6.1   6.7   0.1   7.6   12.2   9.3   1.7   5.4   4.6   2.7   6.7   9.7   13.6   13.1   11.9   10.6
```

Datum	minimale Temp.	maximale Temp.	mittlere Temp.	Sonnenschein
01.5.	6,50	14,50	10,50	4,10
02.5.	7,30	13,60	10,45	2,60
03.5.	5,60	14,80	10,20	2,20
04.5.	3,50	14,20	8,85	11,10
05.5.	1,30	14,00	7,65	9,90
06.5.	0,70	14,30	7,50	14,10
07.5.	4,00	15,10	9,55	6,60
08.5.	1,70	19,30	10,50	12,40
09.5.	5,80	23,00	14,40	8,70
10.5.	6,60	24,20	15,40	8,70
11.5.	3,50	11,60	7,55	5,20
12.5.	2,20	10,60	6,40	5,20
13.5.	3,50	11,50	7,50	4,20
14.5.	1,40	12,20	6,80	10,10
15.5.	-0,40	12,70	6,15	5,10
16.5.	5,80	15,90	10,85	6,10
17.5.	6,20	17,20	11,70	6,70
18.5.	11,10	17,00	14,05	0,10
19.5.	7,40	19,00	13,20	7,60
20.5.	4,40	20,00	12,20	12,20
21.5.	8,00	22,00	15,00	9,30
22.5.	12,20	22,20	17,20	1,70
23.5.	8,60	17,00	12,80	5,40
24.5.	8,50	16,00	12,25	4,60
25.5.	8,50	15,20	11,85	2,70
26.5.	8,20	20,30	14,25	6,70
27.5.	7,00	19,70	13,35	9,70
28.5.	5,30	20,00	12,65	13,60
29.5.	5,10	21,90	13,50	13,10
30.5.	7,50	21,20	14,35	11,90
31.5.	8,20	20,20	14,20	10,60

```python
import numpy as np

celsiusmin = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=1)
celsiusmax = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=2)
celsiusmid = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=3)
sonne = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=4)

datum = np.genfromtxt("Daten.txt", dtype="str", delimiter="   ", max_rows=1)
```

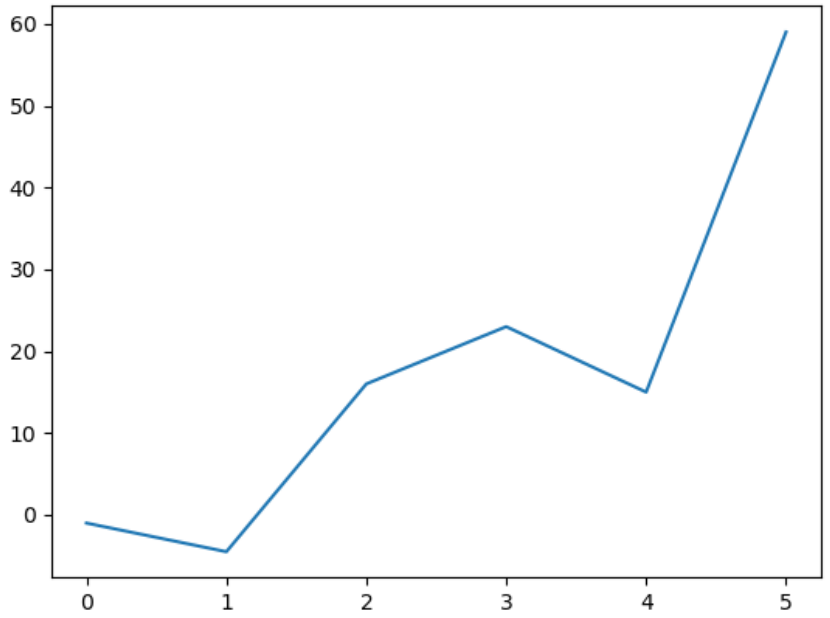
Matplotlib

- Erweitert Numpy um graphische Darstellungsmöglichkeiten
- In Kombination mit Numpy Alternative für MATLAB
- objektorientierte API
- nur wenige Codezeilen für Plots, Histogramme, Leistungsspektren, Balkendiagramme, Fehlerdiagramme, Streudiagramme etc.
- Viele Möglichkeiten zur optischen Formatierung der Darstellungen

Matplotlib

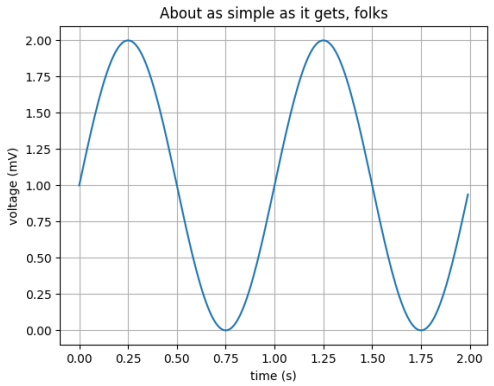
- Daten werden als Array/Liste benötigt
- Ein weiteres Array für x-Achsen Daten
- Optische Einstellungen
  - Legende
  - Linenart/-farbe
  - Marker statt Linie
  - Achseneinstellungen
- pyplot.subplot: mehrere Graphen die nicht übereinander dargestellt werden

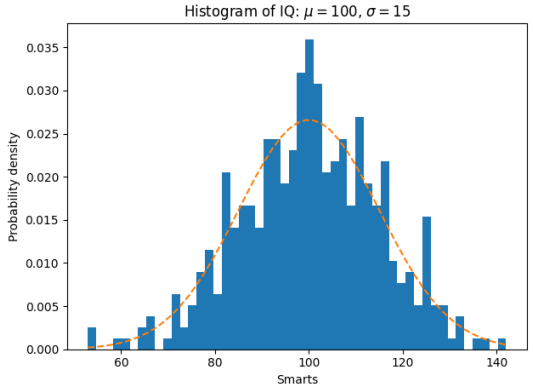
```
import matplotlib.pyplot as plt
plt.plot([-1, -4.5, 16, 23, 15, 59])
plt.show()
```

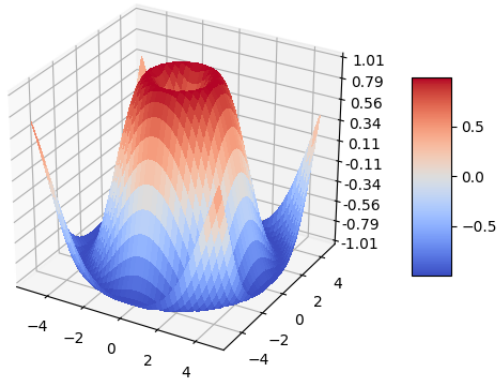


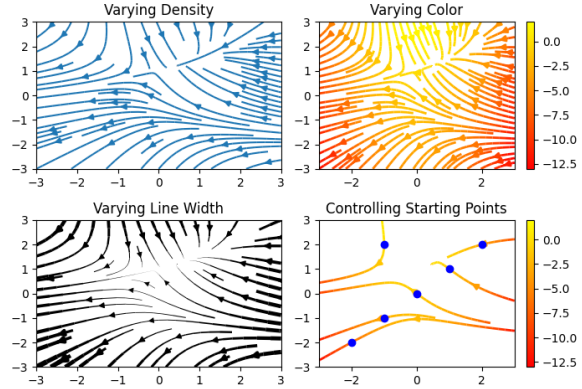
Larissa Lach

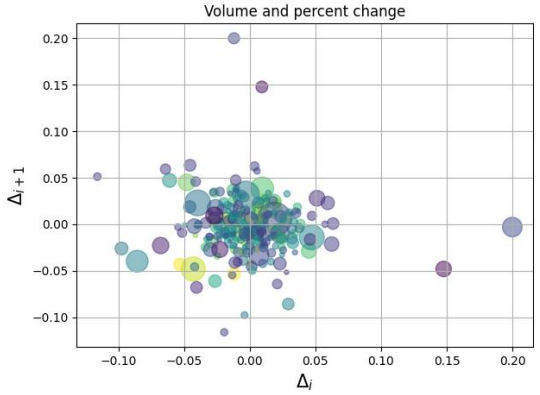
Diagrammtypen

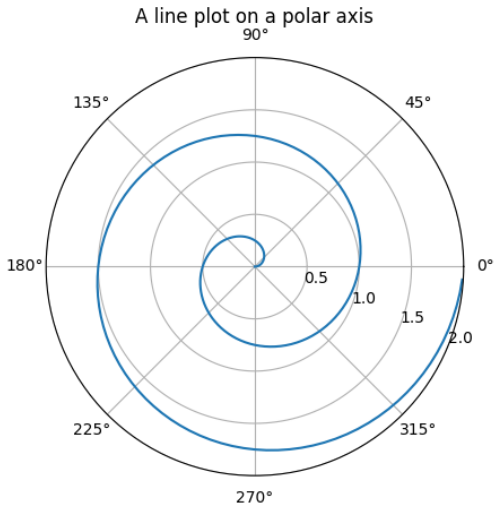


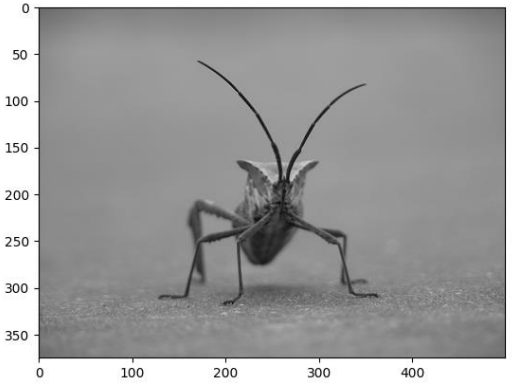


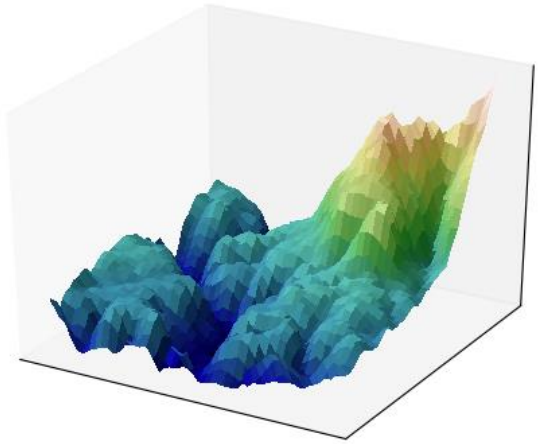












https://matplotlib.org/

Larissa Lach

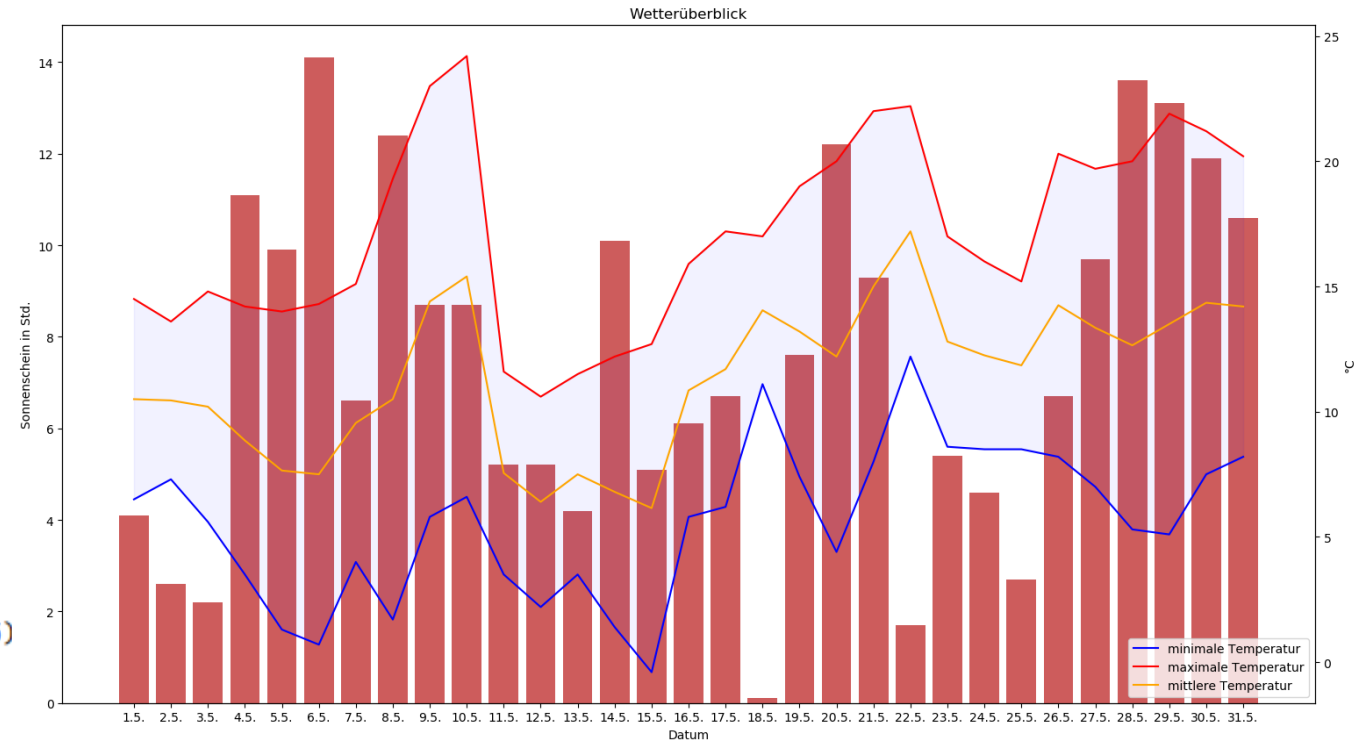
Format-Parameter

Zeichen	Beschreibung
'-'	(Bindestrich) durchgezogene Linie
'--'	(zwei Bindestriche) gestrichelte Linie
'-.'	Strichpunkt-Linie
':'	punktierte Linie
'.'	Punkt-Marker
','	Pixel-Marker
'o'	Kreis-Marker
'v'	Dreiecks-Marker, Spitze nach unten
'^'	Dreiecks-Marker, Spitze nach oben
'<'	Dreiecks-Marker, Spitze nach links
'>'	Dreiecks-Marker, Spitze nach rechts
'1'	tri-runter-Marker
'2'	tri-hoch-Marker
'3'	tri-links Marker
'4'	tri-rechts Marker
's'	quadratischer Marker
'p'	fünfeckiger Marker
'*'	Stern-Marker
'h'	Sechseck-Marker1
'H'	Sechseck-Marker2
'+'	Plus-Marker
'x'	x-Marker
'D'	rautenförmiger Marker
'd'	dünner rautenförmiger Marker
'|'	Marker in Form einer vertikalen Linie
'_'	Marker in Form einer horizontalen Linie

Zeichen	Farbe
'b'	blau
'g'	grün
'r'	rot
'c'	cyan
'm'	magenta
'y'	gelb
'k'	schwarz
'w'	weiß

Beispiel - Wetterdaten

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

celsiusmin = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=1)
celsiusmax = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=2)
celsiusmid = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=3)
sonne = np.loadtxt("Daten.txt", delimiter="   ", max_rows=1, skiprows=4)
datum = np.genfromtxt("Daten.txt", dtype="str", delimiter="   ", max_rows=1)

x = np.arange(0, 31, 1)

fig, ax1 = plt.subplots()

ax1.set_xlabel('Datum')
ax1.set_ylabel('Sonnenschein in Std.')
ax1.bar(x, sonne, color='indianred')

ax2 = ax1.twinx()  # zweite Achse mit der gleichen X-Achse

ax2.set_ylabel('°C')
ax2.plot(x, celsiusmin, 'blue', label='minimale Temperatur')
ax2.plot(x, celsiusmax, 'red', label='maximale Temperatur')
ax2.plot(x, celsiusmid, 'orange', label='mittlere Temperatur')

plt.title('Wetterüberblick')
plt.fill_between(x, celsiusmin, celsiusmax, color='blue', alpha=0.05)
plt.xticks(x, datum)
plt.legend(loc='lower right')

plt.show()
```



Pandas ("Python and data analysis" und "panal data“)

- Daten-Manipulation und -Analyse
- Datenstrukturen: Series und DataFrame
- Series
  - Index und Daten
  - Standard Index
- DataFrame
  - Zeilen- und Spaltenindex
  - x, y und z sind series
    pandas.concat([x, y, z], axis=1)
    x, y und z sind in ein DataFrame umgewandelt
- pandas.read_*() → Datei importieren/lesen
- pandas.to_*() → Datei exportieren

Larissa Lach

Fehlende Daten: NaN

- z.B. durch addieren zweier Tabellen
- Überprüfung mit isnull() oder notnull()
- Entfernen mit dropna() → entfernt die gesamte Zeile
  - dropna(axis=1) → entfernt Spalte
  - fillna() bestimmten Wert einfügen
- dropna(thresh=5, axis=0)

	A	B	C	D	E	F
0	**NaN**	**NaN**	**14.2**	**14.3**	**NaN**	**NaN**
1	14.5	14.5	14.0	15.0	14.5	NaN
2	14.6	NaN	14.8	15.3	14.0	14.2
3	**NaN**	**14.9**	**15.7**	**NaN**	**14.0**	**15.3**
4	15.2	15.2	14.6	15.3	15.5	14.9
5	NaN	15.8	15.9	16.9	16.0	16.2

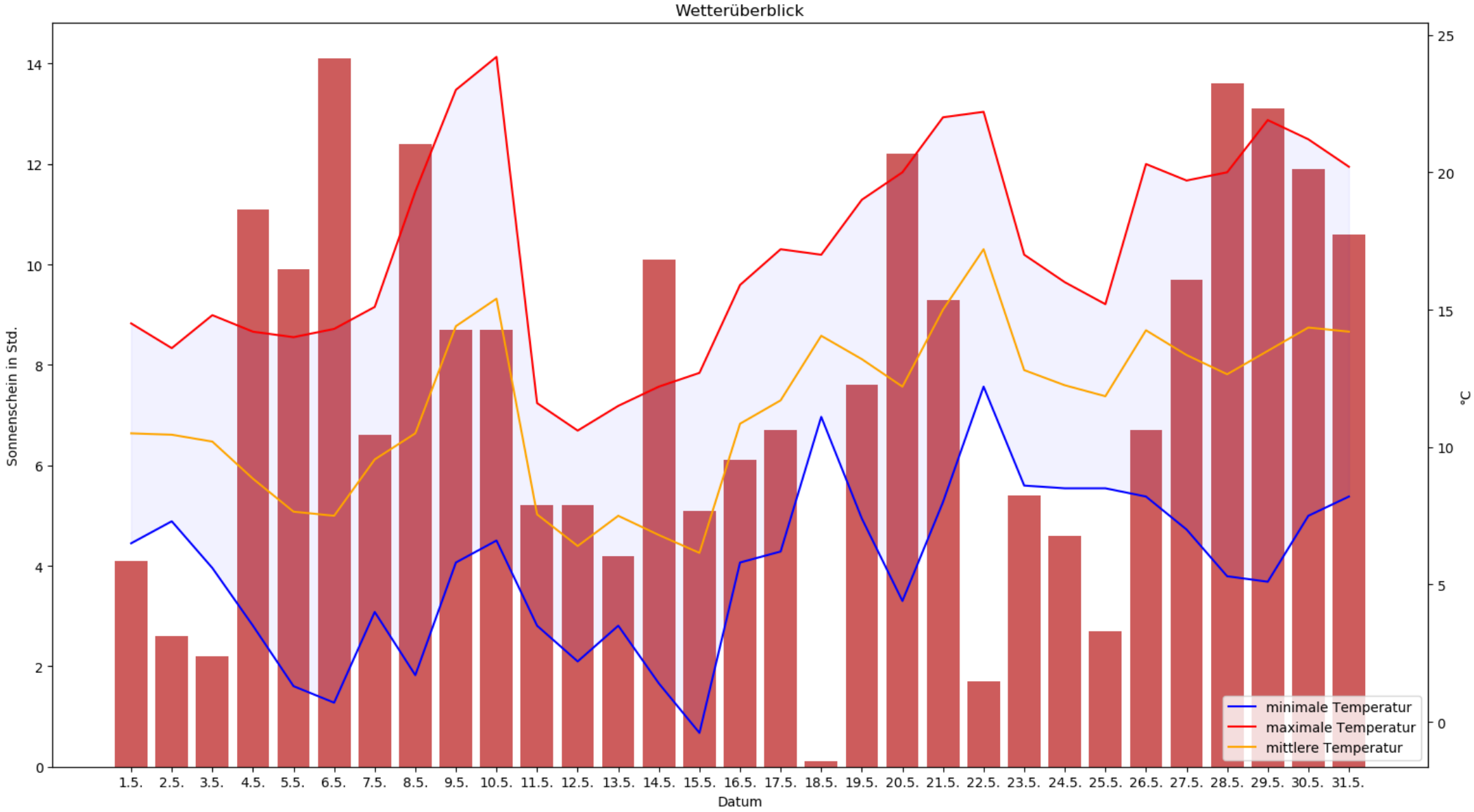
Larissa Lach

Beispiel - Wetterdaten

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

daten = pd.read_excel('Daten.xlsx')
print(daten)

x = np.arange(0, 31, 1)

fig, ax1 = plt.subplots()

ax1.set_xlabel('Datum')
ax1.set_ylabel('Sonnenschein in Std.')
ax1.bar(x, daten['Sonnenschein'], color='indianred')

ax2 = ax1.twinx()  # zweite Achse mit der gleichen X-Achse

ax2.set_ylabel('°C')
ax2.plot(x, daten['minimale Temp.'], 'blue', label='minimale Temperatur')
ax2.plot(x, daten['maximale Temp.'], 'red', label='maximale Temperatur')
ax2.plot(x, daten['mittlere Temp.'], 'orange', label='mittlere Temperatur')


plt.title('Wetterüberblick')
plt.fill_between(x, daten['minimale Temp.'], daten['maximale Temp.'], color='blue', alpha=0.05)
plt.xticks(x, daten['Datum'])
plt.legend(loc='lower right')

plt.show()
```

	Datum	minimale Temp.	maximale Temp.	mittlere Temp.	Sonnenschein
0	01.5.	6.5	14.5	10.50	4.1
1	02.5.	7.3	13.6	10.45	2.6
2	03.5.	5.6	14.8	10.20	2.2
3	04.5.	3.5	14.2	8.85	11.1
4	05.5.	1.3	14.0	7.65	9.9
5	06.5.	0.7	14.3	7.50	14.1
6	07.5.	4.0	15.1	9.55	6.6
7	08.5.	1.7	19.3	10.50	12.4
8	09.5.	5.8	23.0	14.40	8.7
9	10.5.	6.6	24.2	15.40	8.7
10	11.5.	3.5	11.6	7.55	5.2
11	12.5.	2.2	10.6	6.40	5.2
12	13.5.	3.5	11.5	7.50	4.2
13	14.5.	1.4	12.2	6.80	10.1
14	15.5.	-0.4	12.7	6.15	5.1
15	16.5.	5.8	15.9	10.85	6.1
16	17.5.	6.2	17.2	11.70	6.7
17	18.5.	11.1	17.0	14.05	0.1
18	19.5.	7.4	19.0	13.20	7.6
19	20.5.	4.4	20.0	12.20	12.2
20	21.5.	8.0	22.0	15.00	9.3
21	22.5.	12.2	22.2	17.20	1.7
22	23.5.	8.6	17.0	12.80	5.4
23	24.5.	8.5	16.0	12.25	4.6
24	25.5.	8.5	15.2	11.85	2.7
25	26.5.	8.2	20.3	14.25	6.7
26	27.5.	7.0	19.7	13.35	9.7
27	28.5.	5.3	20.0	12.65	13.6
28	29.5.	5.1	21.9	13.50	13.1
29	30.5.	7.5	21.2	14.35	11.9
30	31.5.	8.2	20.2	14.20	10.6

Beispiel - Wetterdaten



Zusammenfassung

- Python in Verbindung mit den Paketen deutlich effizienter
- Numpy
  - Schnelle und einfache Ausführung/Programmierung
  - Viele Möglichkeiten durch vorimplementierte Funktionen
- Matplotlib
  - Auswahl an Diagrammarten riesig
- Pandas
  - Übersichtlichkeit für große Tabellen mit Index

Literatur

- NumPy
  https://www.python-kurs.eu/numpy.php
  https://www.w3schools.com/python/numpy_intro.asp
  https://numpy.org/doc/stable/
- Matplotlib
  https://www.python-kurs.eu/matplotlib.php
  https://matplotlib.org/
  https://matplotlib.org/contents.html
- Pandas
  https://www.python-kurs.eu/pandas.php
  https://pandas.pydata.org/docs/