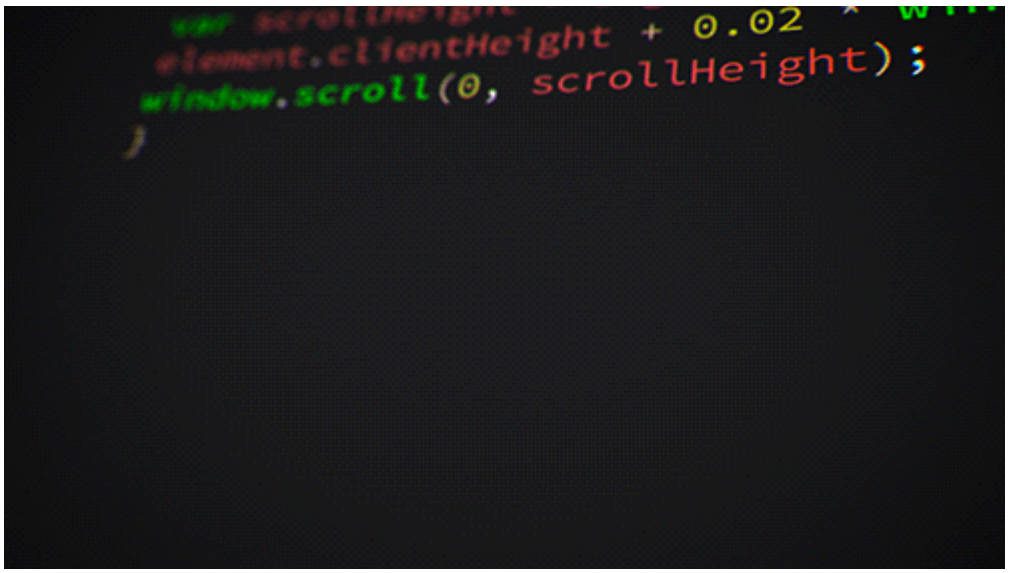
Visualisierung mit Python

Parameter	Kursinformationen
Veranstaltung:	`Prozedurale Programmierung / Einführung in die Informatik / Erhebung, Analyse und Visualisierung digitaler Daten`
Semester	`Wintersemester 2025/26`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Visualisierung mit Python`
Link auf Repository:	https://github.com/TUBAF-IfI-LiaScript/VL_EAVD/blob/master/09_Datenvisualisierung.md
Autoren	Sebastian Zug & André Dietrich & Galina Rudolf & Bernhard Jung



Fragen an die heutige Veranstaltung ...

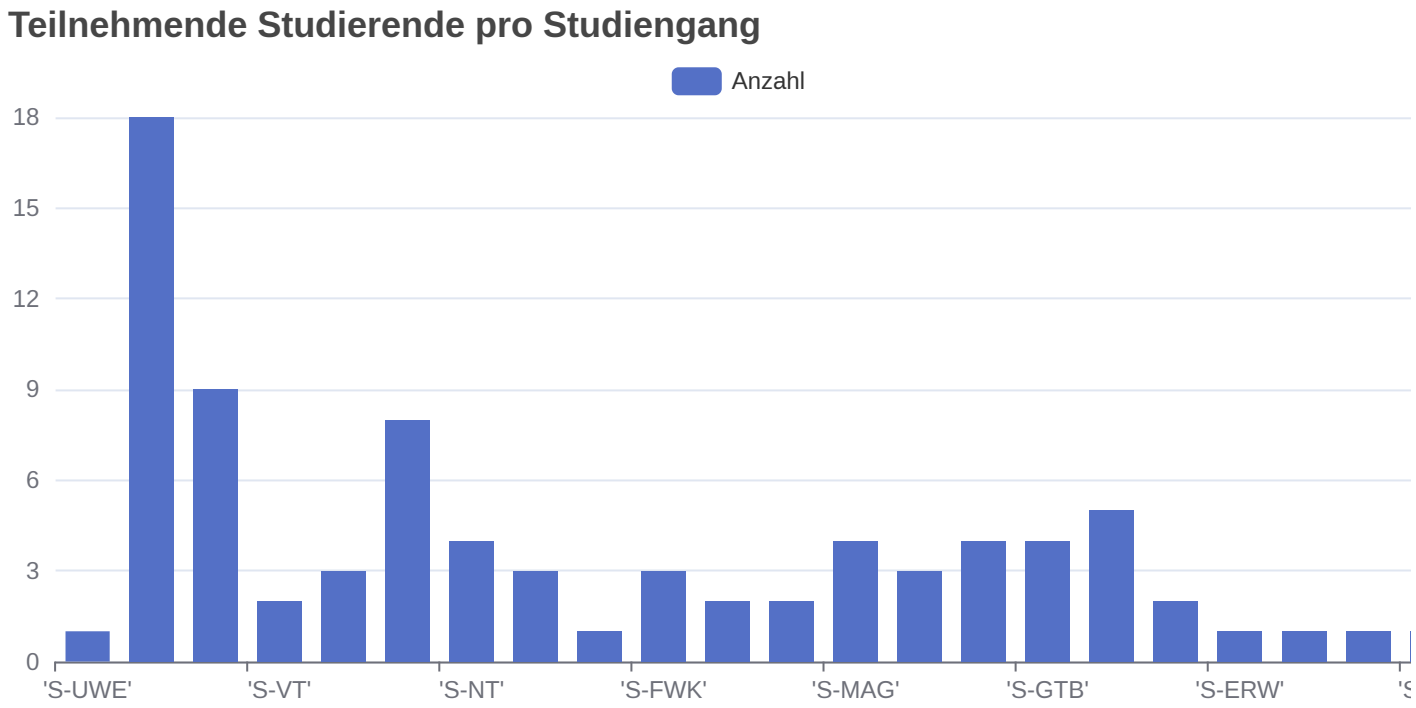
- Welche Grundkonzepte stehen hinter der Programmierung von Grafiken?
- Wie geht man bei der Erschließung von unbekannten Methoden sinnvoll vor?

Datenvisualisierung

In einem vorigen Termin haben wir Ihre Zugehörigkeit zu verschiedenen Studiengängen eingelesen und analysiert Link L08.

Auf die Frage hin, welche Häufigkeiten dabei auftraten, beantwortete unser Skript mit einem Dictonary:

```
{'S-UWE': 1, 'S-WIW': 18, 'S-GÖ': 9, 'S-VT': 2, 'S-BAF': 3, 'S-WWT': 8, 'S-
  ,
 'S-ET': 3, 'S-MB': 1, 'S-FWK': 3, 'F1-INF': 2, 'S-BWL': 2, 'S-MAG': 4, 'F2
   : 3,
 'S-ACW': 4, 'S-GTB': 4, 'S-GBG': 5, 'S-GM': 2, 'S-ERW': 1, 'S-INA': 1, 'S-
   1,
  'S-CH': 1}
```



Die textbasierte Ausgabe ist nur gering geeignet, um einen raschen Überblick zu erlangen. Entsprechend suchen wir nach einer grafischen Ausgabemöglichkeit für unsere Python Skripte.

Python Visualisierungstools

Python stellt eine Vielzahl von Paketen für die Visualisierung von Dateninhalten bereit. Diese zielen auf unterschiedliche Visionen oder Features:

- einfache Verwendbarkeit
- große Bandbreite von Diagrammarten und Adaptionsmöglichkeiten
- interaktive Diagramme
- Vielzahl von Exportschnittstellen

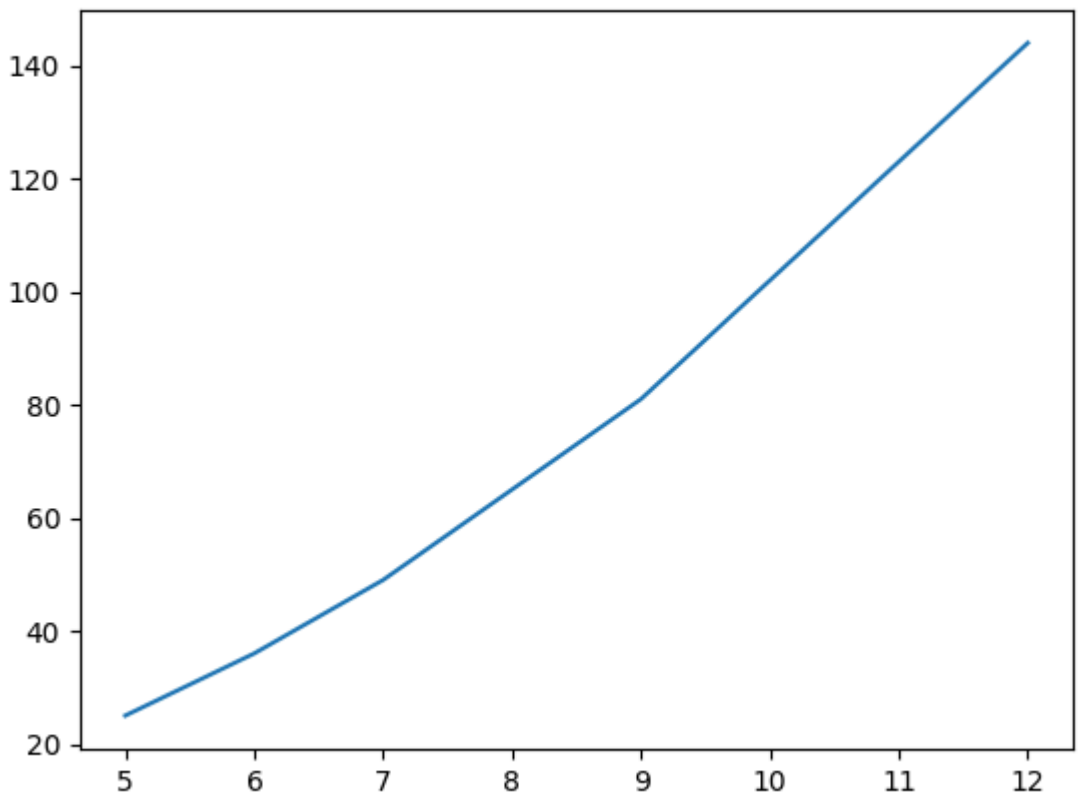
Package	Link	Besonderheiten
plotly	Link	Fokus auf interaktive Diagramme eingebettetet in Webseiten
seaborn	Link	Leistungsfähige Darstellung von statistischen Daten
matplotlib	Link	
...		

Matplotlib Grundlagen

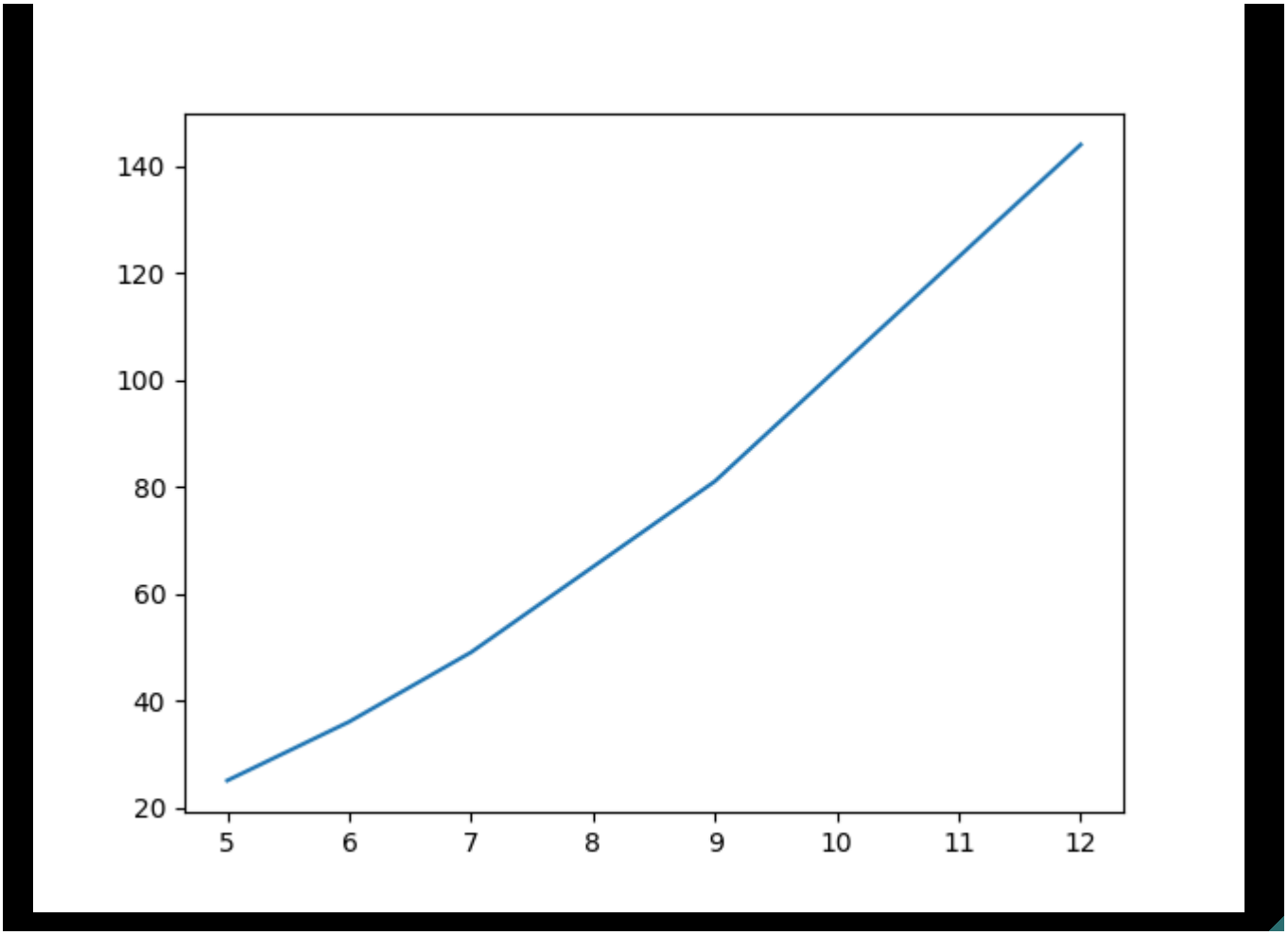
Beispiel.py

```
1  import matplotlib.pyplot as plt
2
3  a = [5,6,7,9,12]
4  b =[x**2 for x in a]   # List Comprehension
5  plt.plot(a, b)
6
7  #plt.show()
8  plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

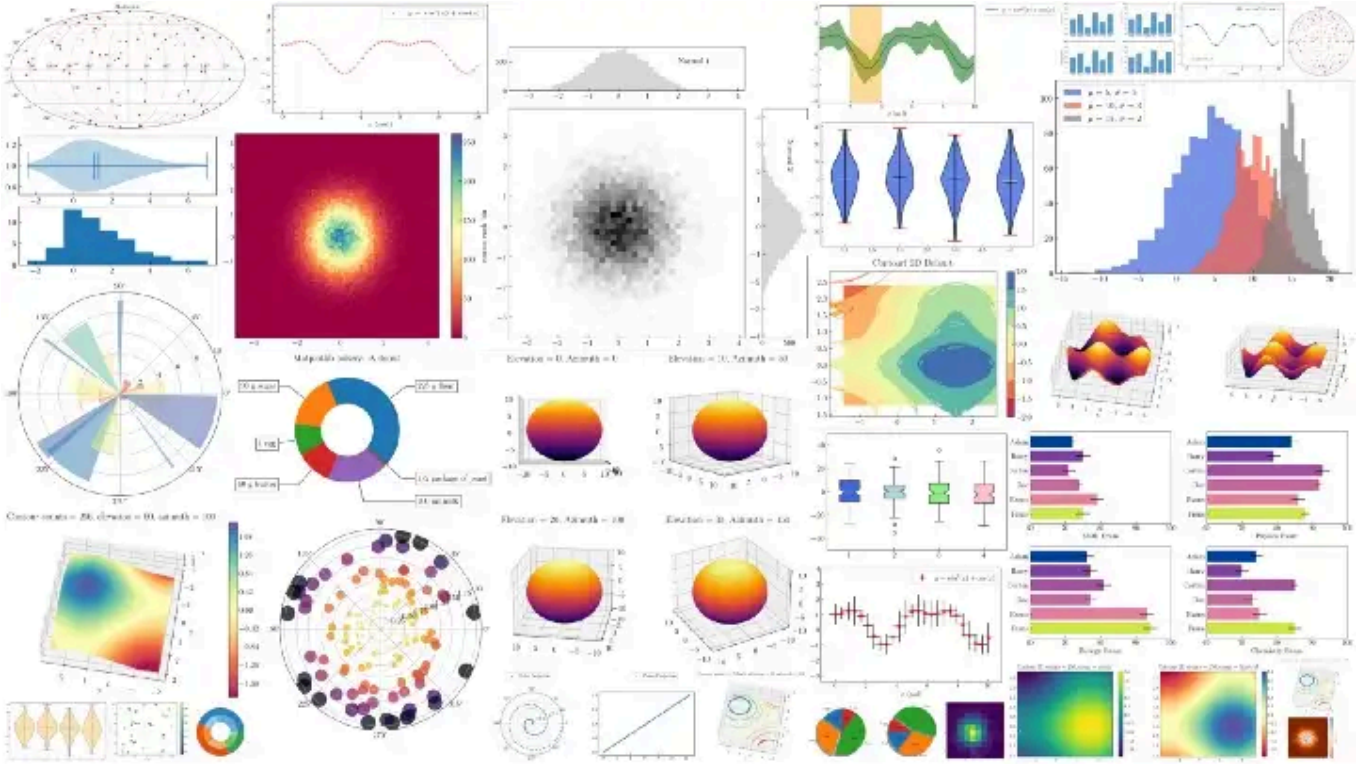
foo.png



foo.png

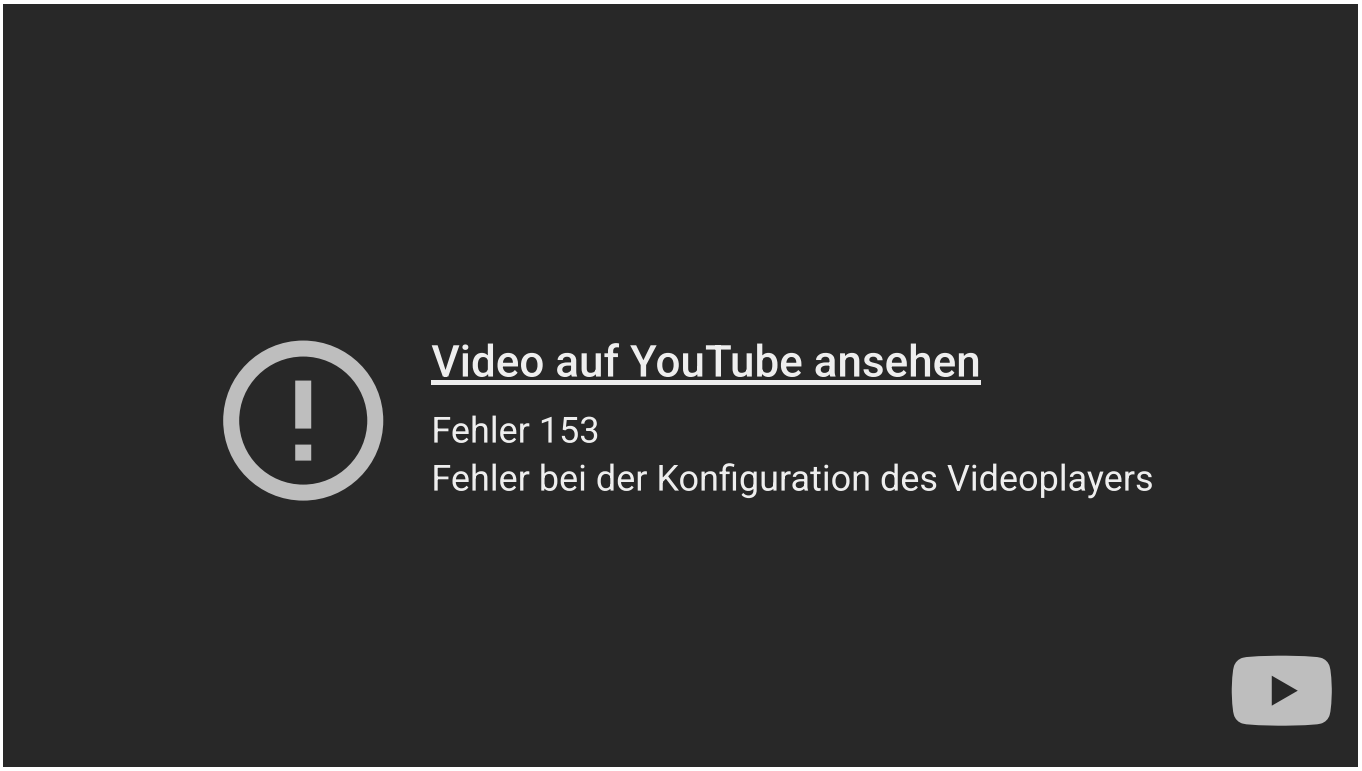


Anpassung	API	
Linientyp der Datendarstellung	pyplot.plot	`plt.plot(a, b, 'ro:')`
Achsenlabel hinzufügen	pyplot.xlabel	`plt.xlabel('my data', fontsize=14, color='red')`
Titel einfügen	pyplot.title	`plt.title(r'$\sigma_i=15$')`
Gitter einfügen	pyplot.grid	`plt.grid()`
Legende	pyplot.legend	`plt.plot(a, b, 'ro:', label="Data")`
		`plt.legend()`
Speichern	pyplot.savefig	`plt.savefig('foo.png')`



Tutorial von Rizky Maulana Nurhidayat auf medium

Weiter Tutorials sind zum Beispiel unter

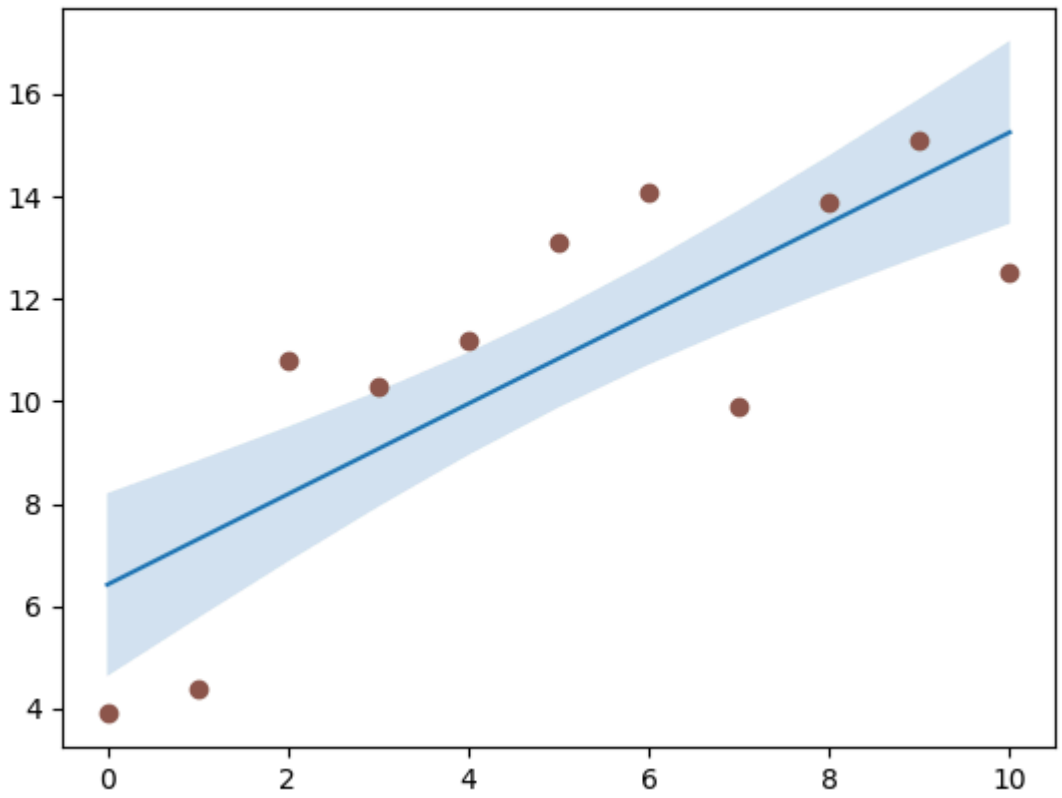


Matplotlib Beispiele

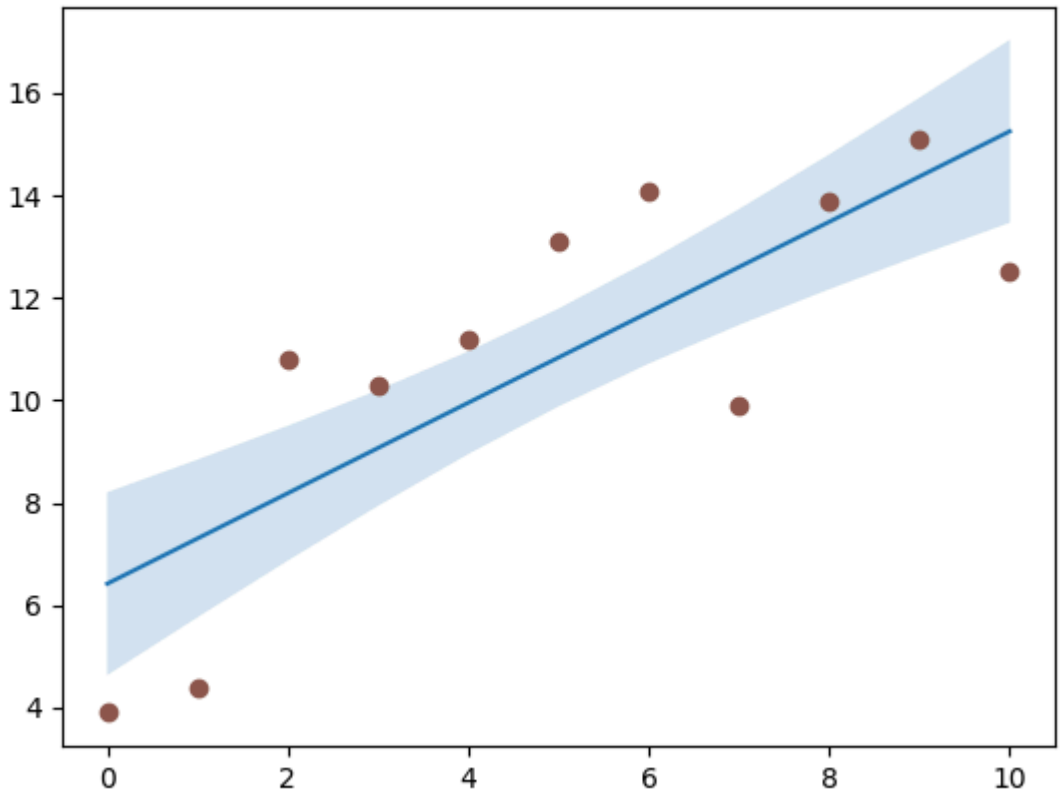
LinearRegression.py

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

N = 21
x = np.linspace(0, 10, 11)   # [ 0.,  1.,  2.,  3., ..., 10.0]
y = [3.9, 4.4, 10.8, 10.3, 11.2, 13.1, 14.1,  9.9, 13.9, 15.1, 12.5]

# fit a linear curve an estimate its y-values and their error.
a, b = np.polyfit(x, y, deg=1)
y_est = a * x + b
y_err = x.std() * np.sqrt(1/len(x) +
                          (x - x.mean())**2 / np.sum((x - x.mean())**

fig, ax = plt.subplots()
ax.plot(x, y_est, '-')    # plot the fitted (regression) line
ax.fill_between(x, y_est - y_err, y_est + y_err, alpha=0.2)
ax.plot(x, y, 'o', color='tab:brown')   # plot data points as brown c

#plt.show()
plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

foo.png



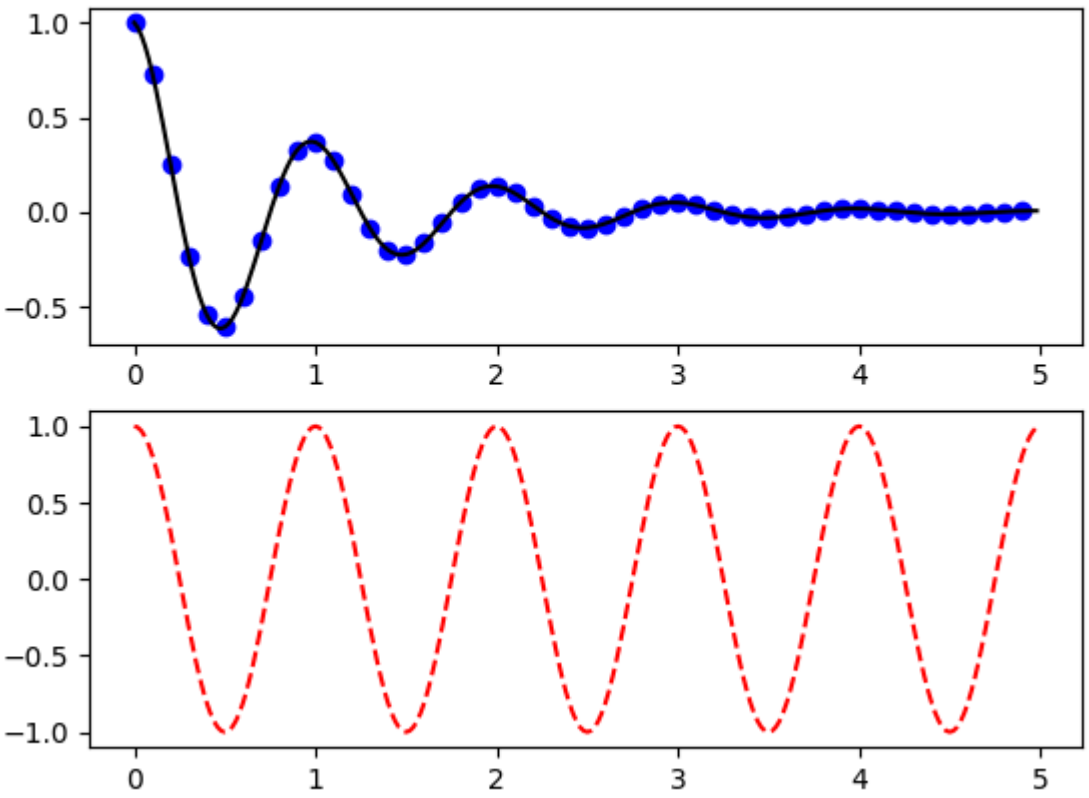
foo.png



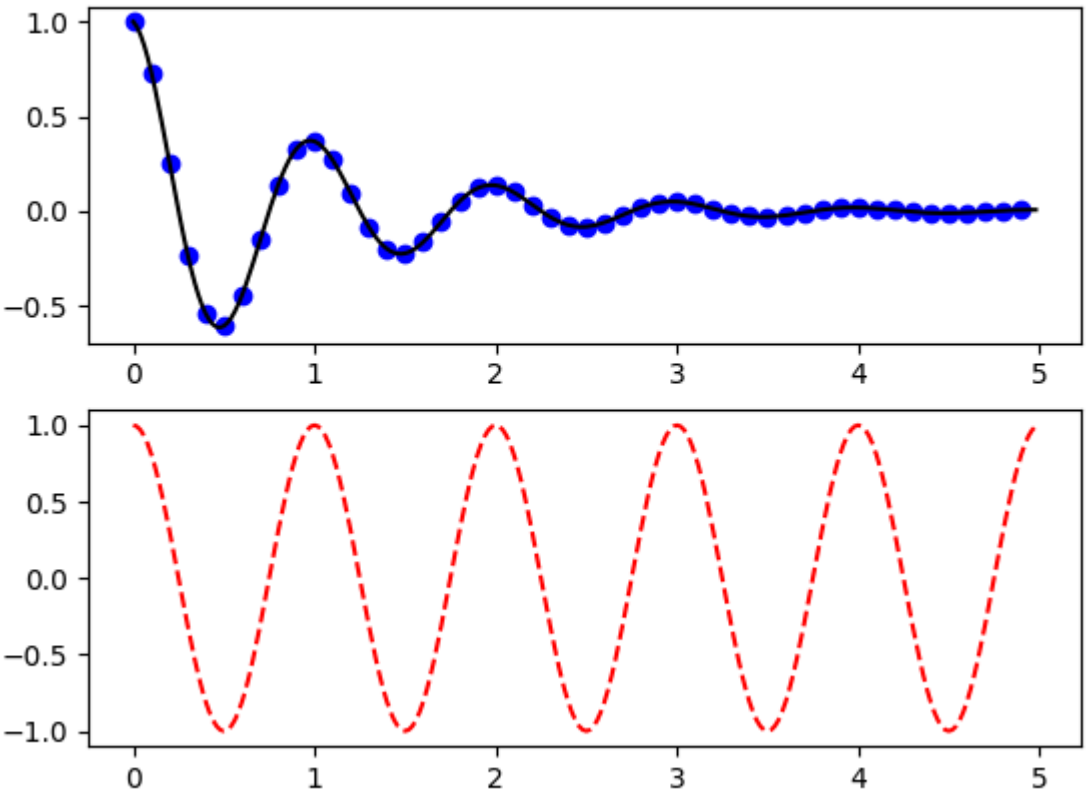
MultipleDiagrams.py

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

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)   # [0, 0.1, 0.2, ..., 4.9 ]
t2 = np.arange(0.0, 5.0, 0.02)  # [0.  , 0.02, 0.04, 0.06, ..., 4.98]

plt.figure()
plt.subplot(2, 1, 1) # 2 Zeilen, 1 Spalte: erstes Diagramm
# plt.subplot(211) # geht auch, falls Anzahl der Zeilen/Spalten <= 9
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
# 'bo': blaue Kreise, 'k': schwarze Linie

plt.subplot(212) # 2 Zeilen, 1 Spalte: zweites Diagramm
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
#plt.show()
plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

foo.png



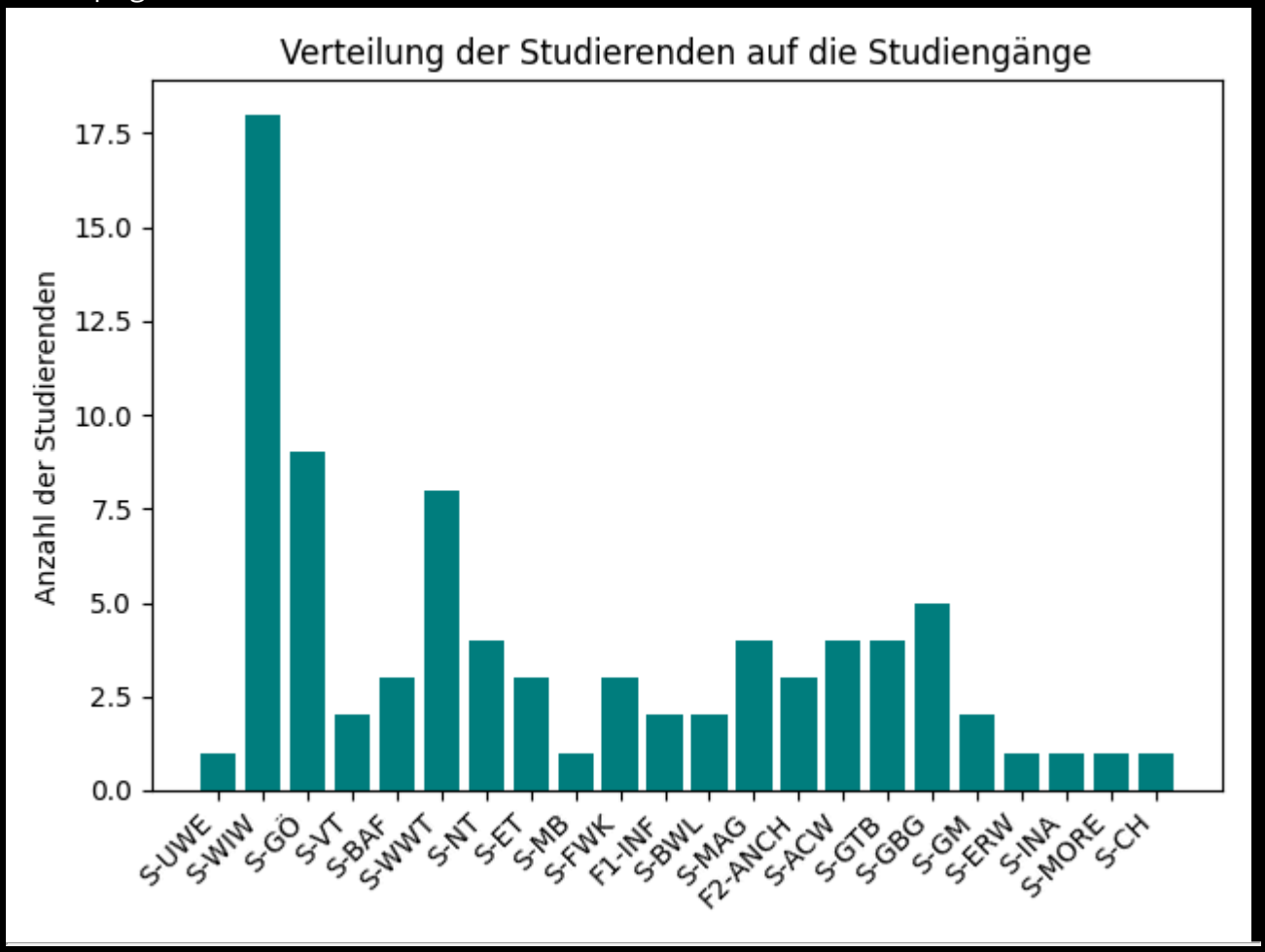
foo.png



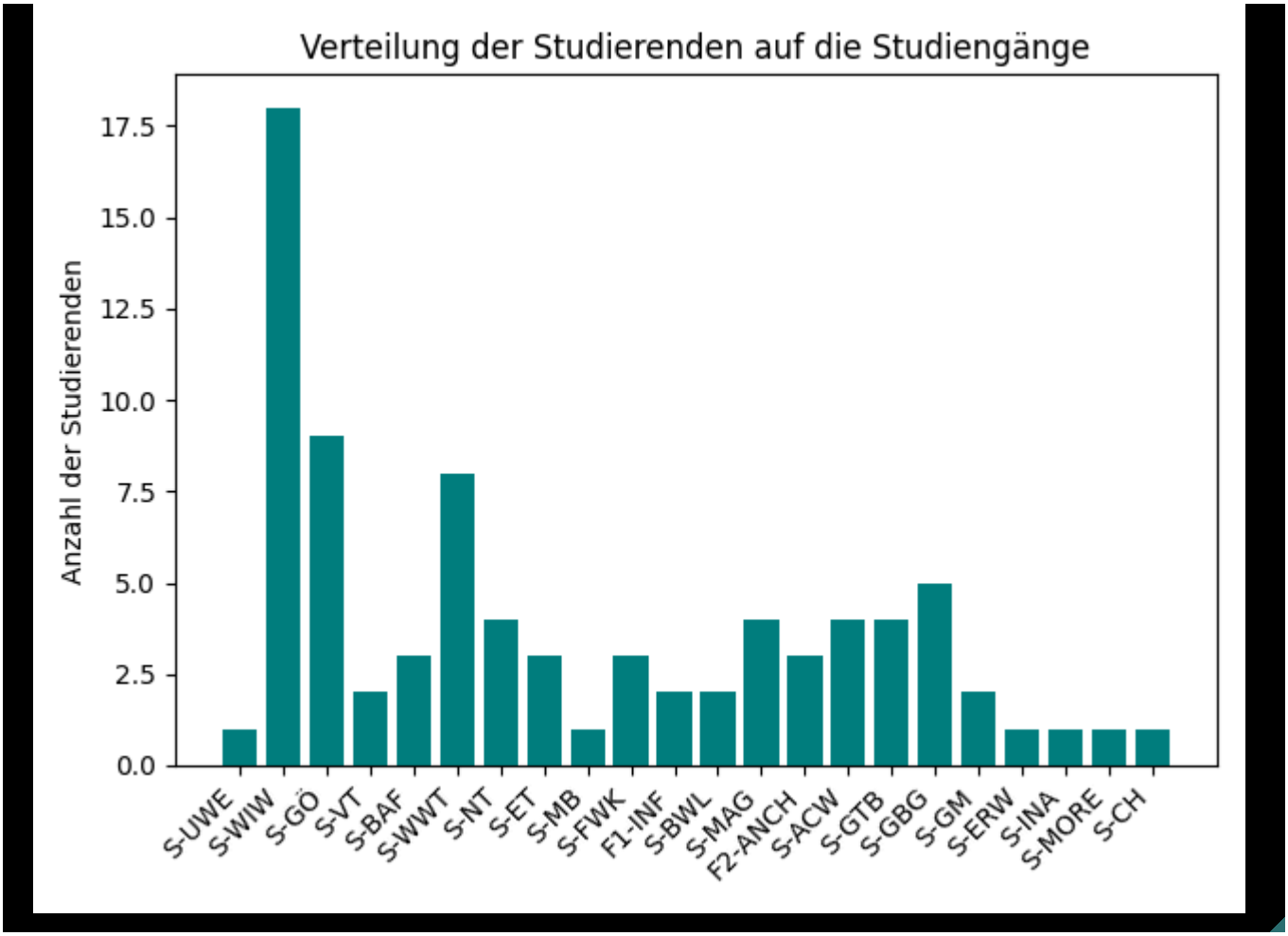
barchart.py

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

data ={'S-UWE': 1, 'S-WIW': 18, 'S-GÖ': 9, 'S-VT': 2, 'S-BAF': 3, 'S-
  8,
 'S-NT': 4, 'S-ET': 3, 'S-MB': 1, 'S-FWK': 3, 'F1-INF': 2, 'S-BWL': 2
 'S-MAG': 4, 'F2-ANCH': 3, 'S-ACW': 4, 'S-GTB': 4, 'S-GBG': 5, 'S-GM'
 'S-ERW': 1, 'S-INA': 1, 'S-MORE': 1, 'S-CH': 1}

labels = list(data.keys())
values = list(data.values())

fig, ax = plt.subplots()
ax.bar(labels, values, color='teal')
ax.set_ylabel('Anzahl der Studierenden')
ax.set_title('Verteilung der Studierenden auf die Studiengänge')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
#plt.show()
plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

foo.png



foo.png

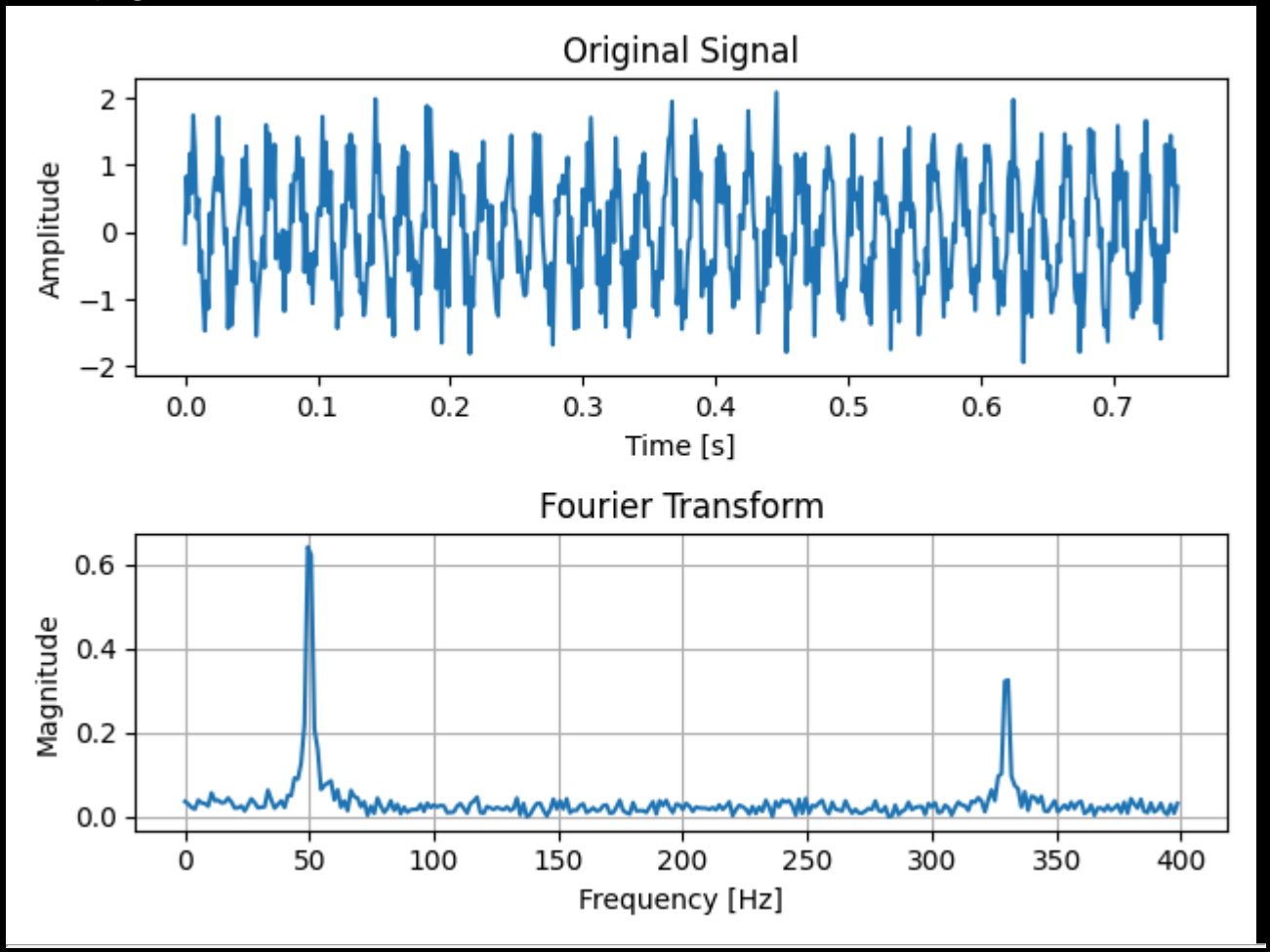


Beispiel der Woche

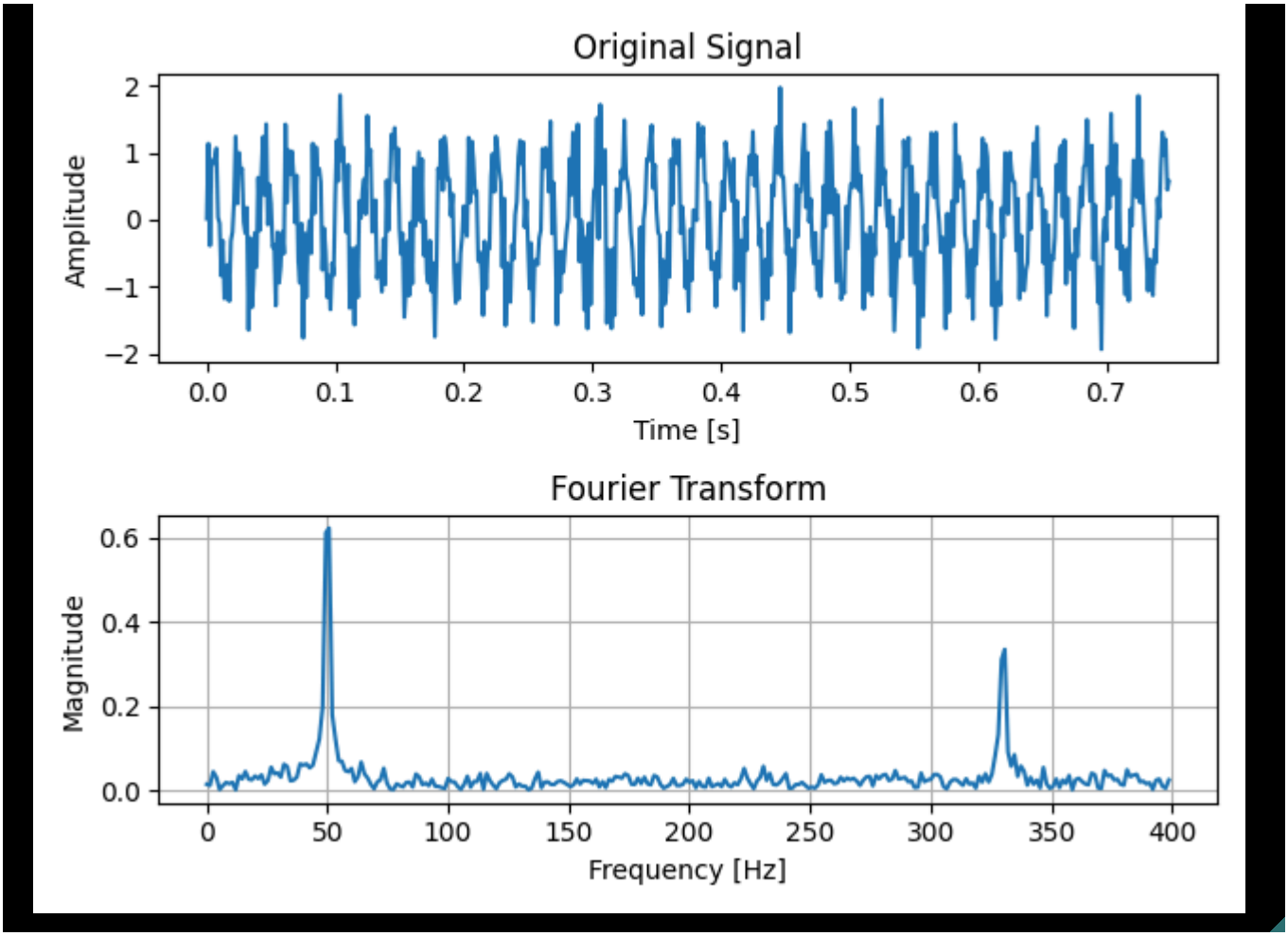
BeispielFourierTransformation.py

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq

# Number of sample points
N = 600

# sample spacing
T = 1.0 / 800.0
x = np.linspace(0.0, N*T, N, endpoint=False)

# create a signal with two frequencies (50 Hz and 330 Hz) and some no
y = np.sin(50.0 * 2.0*np.pi*x)  # 50 Hz component
y += 0.5*np.sin(330.0 * 2.0*np.pi*x) # 330 Hz component, smaller ampl
y += 0.3*np.random.normal(size=x.shape) # add some noise (optional)

# compute the Fourier Transform and corresponding frequencies
yf = fft(y)
xf = fftfreq(N, T)[:N//2]  # positive frequencies only

# plot the original signal and its Fourier Transform
fig, axs = plt.subplots(2, 1)
axs[0].plot(x, y)
axs[0].set_title('Original Signal')
axs[0].set_xlabel('Time [s]')
axs[0].set_ylabel('Amplitude')
axs[1].plot(xf, 2.0/N * np.abs(yf[0:N//2]))
axs[1].set_title('Fourier Transform')
axs[1].set_xlabel('Frequency [Hz]')
axs[1].set_ylabel('Magnitude')
plt.grid()
plt.tight_layout()

#plt.show()
plt.savefig('foo.png')
```

foo.png



foo.png



Quiz

Matplotlib Grundlagen

Wodurch muss `[_____]` ersetzt werden, um einen Plot mit dem Jahr auf der X-Achse und der Anzahl der Tassen Tee auf der Y-Achse zu erstellen?

```
import matplotlib.pyplot as plt

year = [2000, 2001, 2002, 2003, 2004]
ttg =[232, 533, 433, 410, 450] # Tassen Tee getrunken
plt.[_____]

plt.show()
```