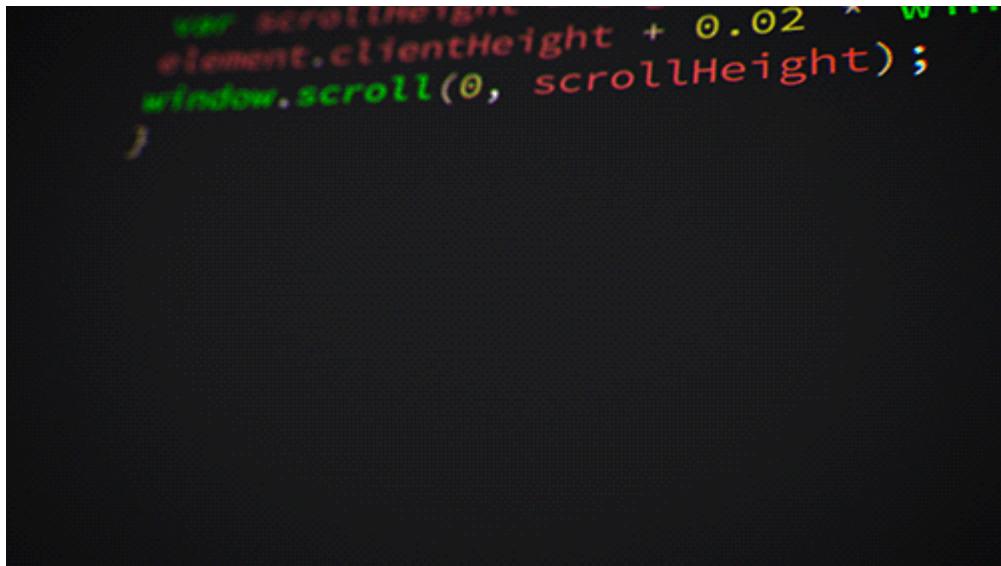


# Visualisierung mit Python

Parameter	Kursinformationen
Veranstaltung:	<u>Prozedurale Programmierung / Einführung in die Informatik / Erhebung, Analyse und Visualisierung digitaler Daten</u>
Semester	Wintersemester 2025/26
Hochschule:	Technische Universität Freiberg
Inhalte:	Visualisierung mit Python
Link auf Repository:	<a href="https://github.com/TUBAF-IfI-LiaScript/VL_EAVD/blob/master/09_Datenvisualisierung.md">https://github.com/TUBAF-IfI-LiaScript/VL_EAVD/blob/master/09_Datenvisualisierung.md</a>
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?

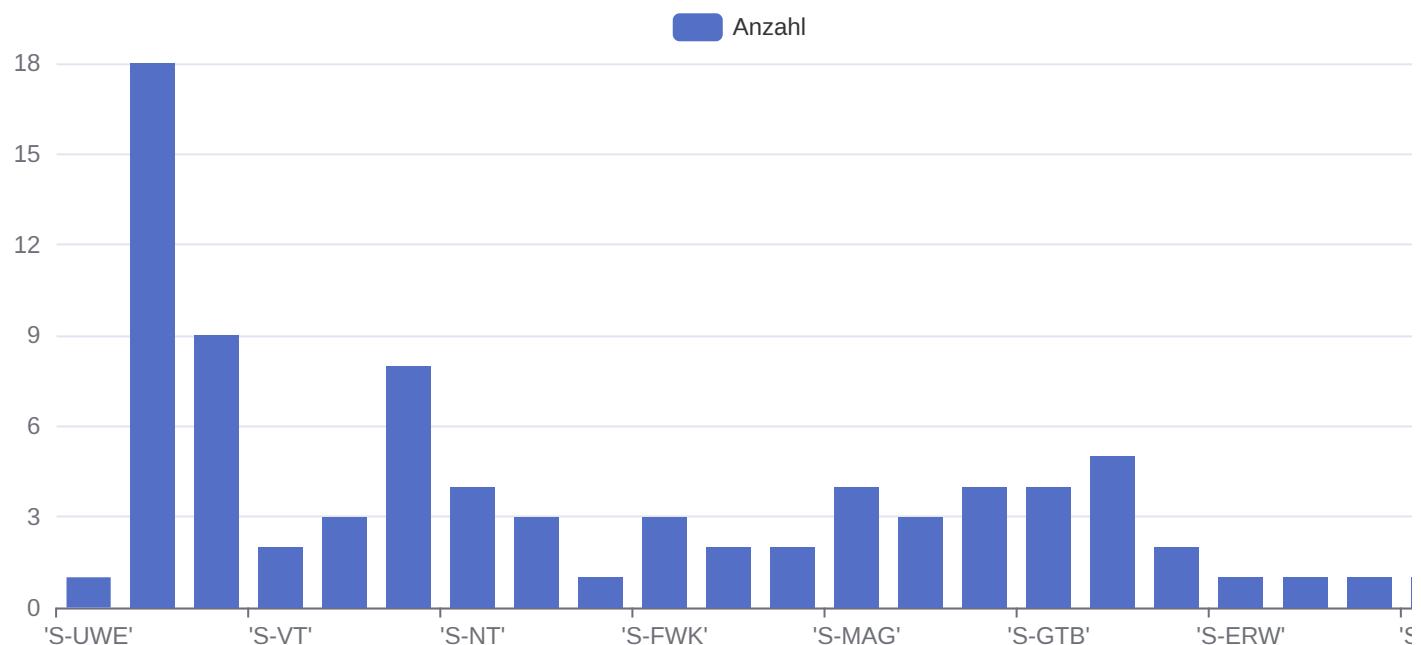
## Datenvisualisierung

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 Dictionary:

```
{'S-UWE': 1, 'S-WIW': 18, 'S-GÖ': 9, 'S-VT': 2, 'S-BAF': 3, 'S-WWT': 8, '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}
```

## Teilnehmende Studierende pro Studiengang



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 Adoptionsmöglichkeiten
- interaktive Diagramme
- Vielzahl von Exportschnittstellen

Package	Link	Besonderheiten
plotly	<a href="#">Link</a>	Fokus auf interaktive Diagramme eingebettetet in Webseiten
seaborn	<a href="#">Link</a>	Leistungsfähige Darstellung von statistischen Daten
matplotlib	<a href="#">Link</a>	
...		

## Matplotlib Grundlagen

### Beispiel.py

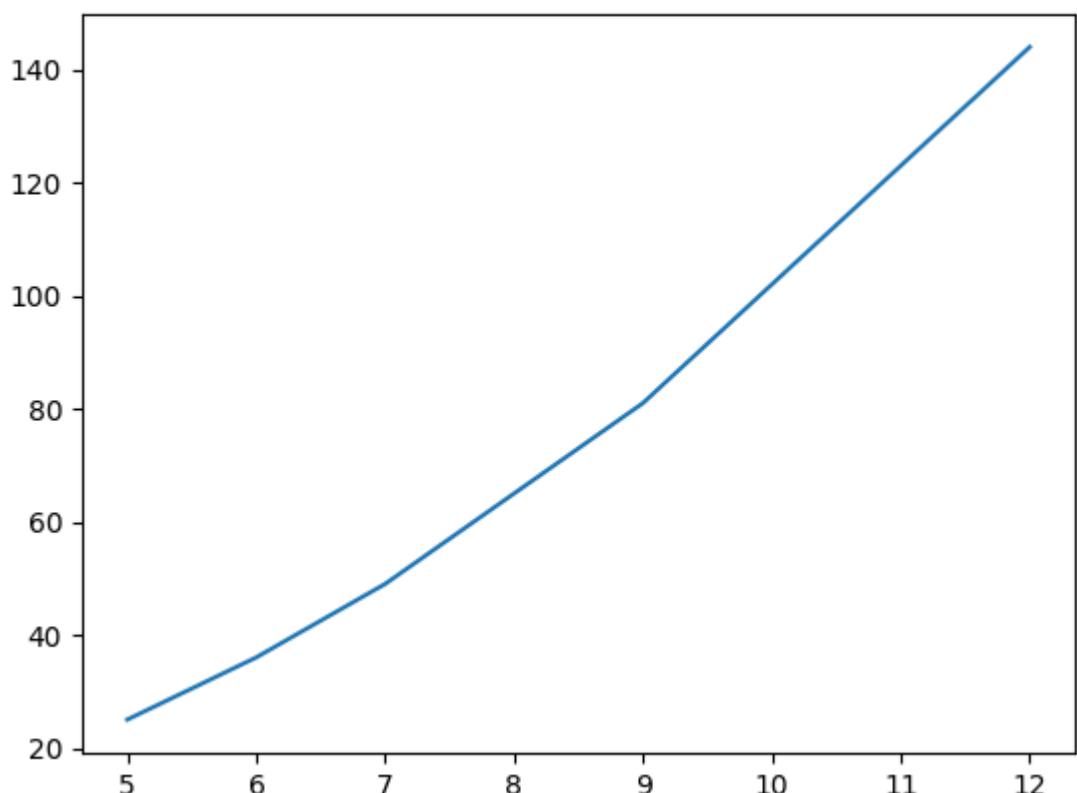


```

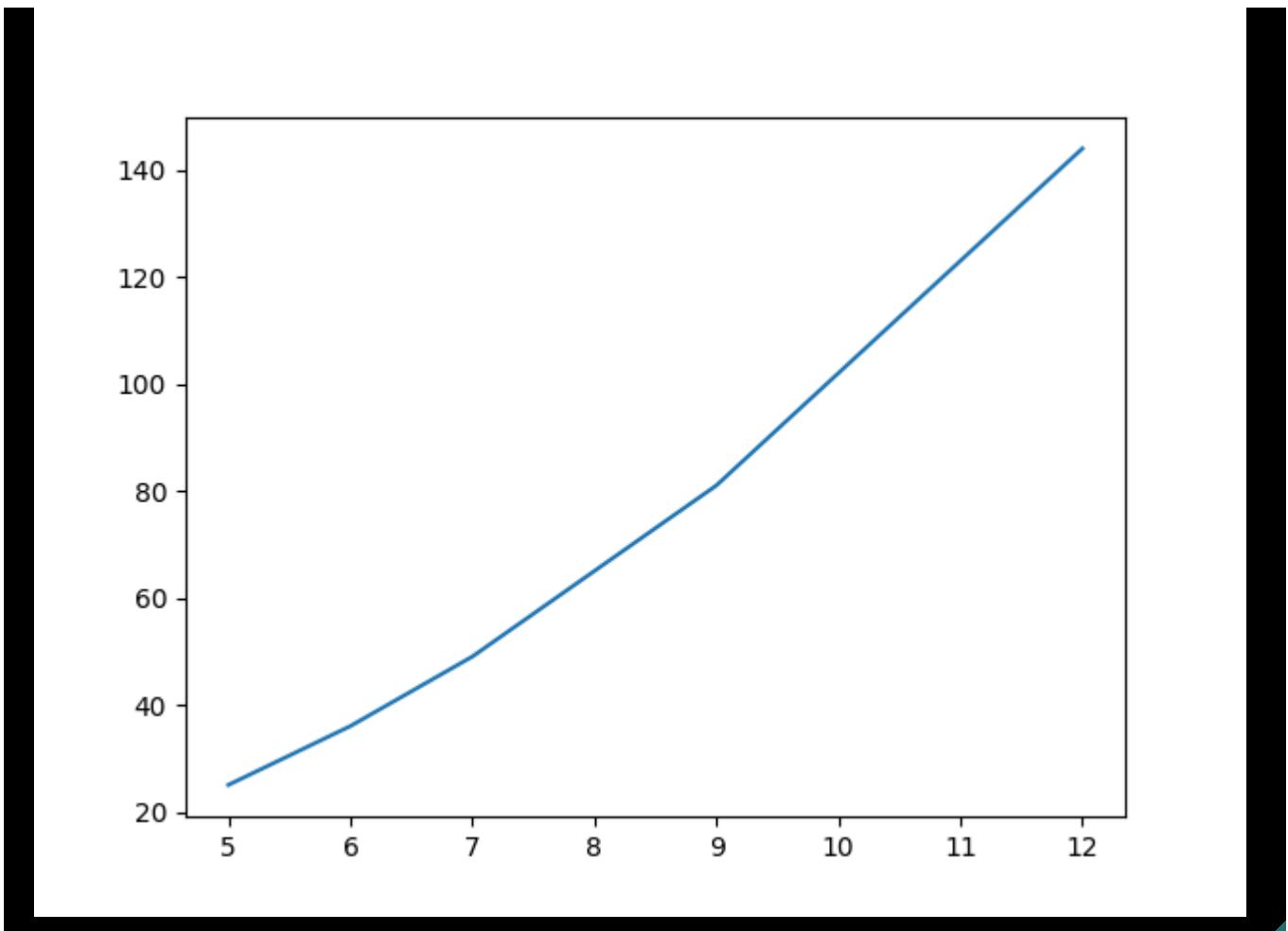
1 import matplotlib.pyplot as plt
2
3 a = [5,6,7,9,12]
4 b =[x**2 for x in a]    # List Comprehension
5 plt.plot(a, b)
6
7 #plt.show()
8 plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript

```

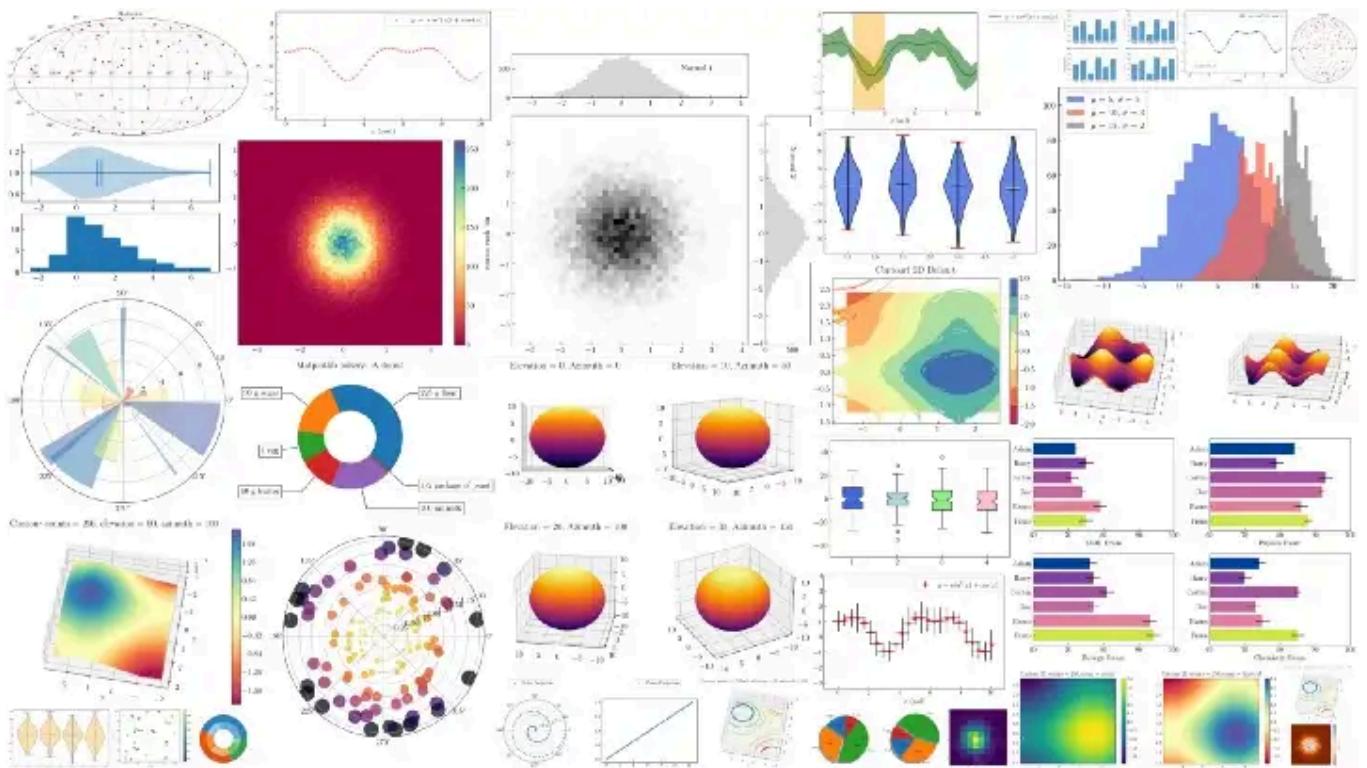
foo.png



foo.png

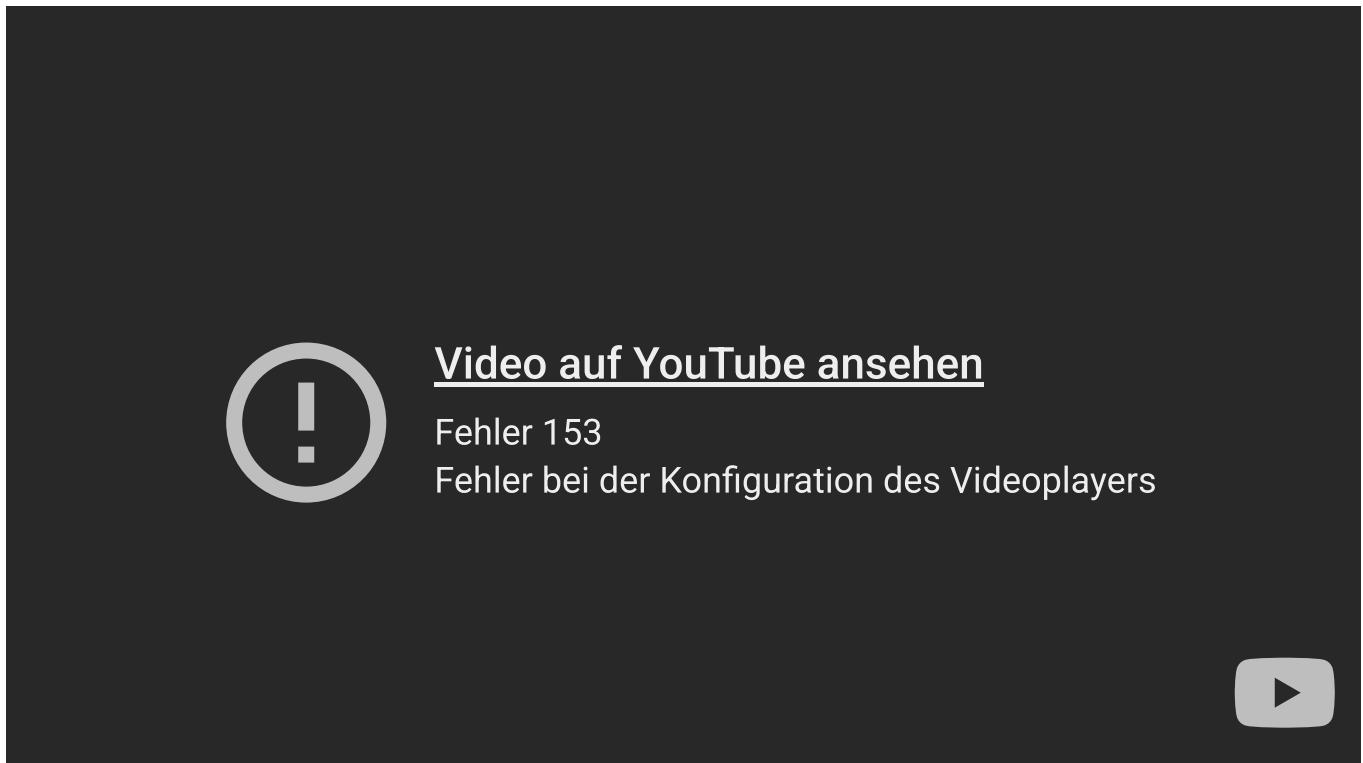


Anpassung	API	
Linientyp der Datendarstellung	<code>pyplot.plot</code>	<code>plt.plot(a, b, 'ro:')</code>
Achsenlabel hinzufügen	<code>pyplot.xlabel</code>	<code>plt.xlabel('my data', fontsize=14, color='red')</code>
Titel einfügen	<code>pyplot.title</code>	<code>plt.title(r'\$\sigma_i=15\$')</code>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



Tutorial von Rizky Maulana Nurhidayat auf [medium](#)

Weiter Tutorials sind zum Beispiel unter



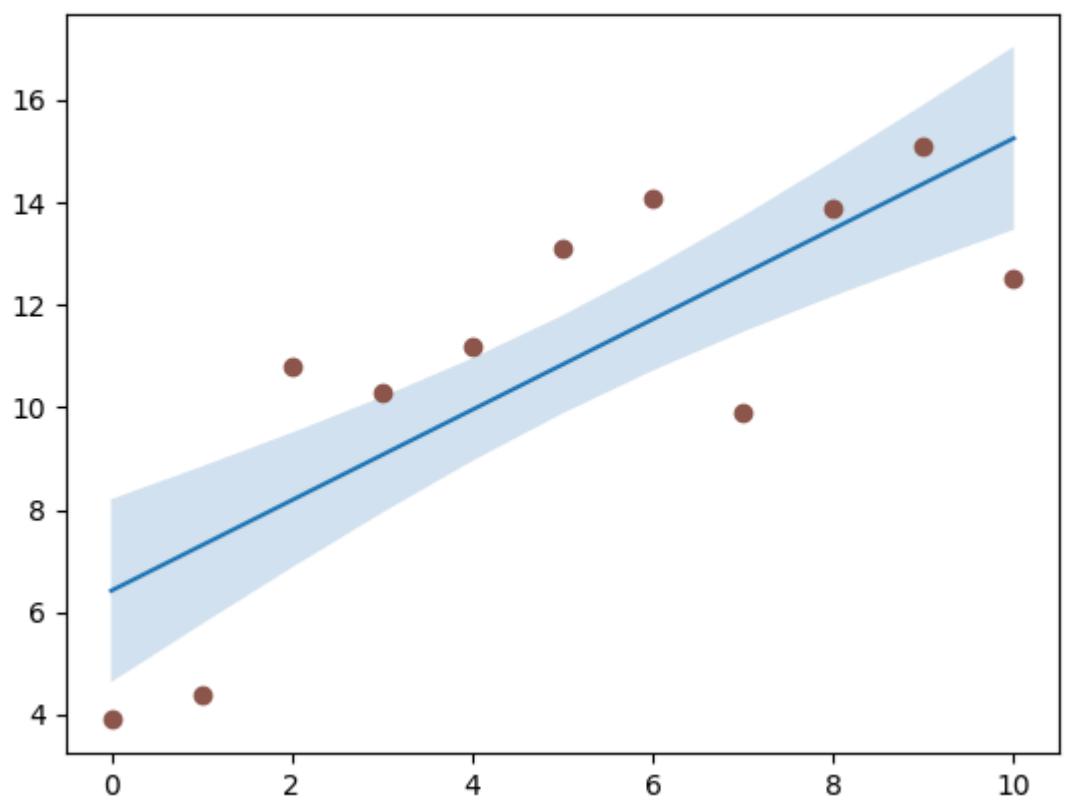
## Matplotlib Beispiele

## LinearRegression.py

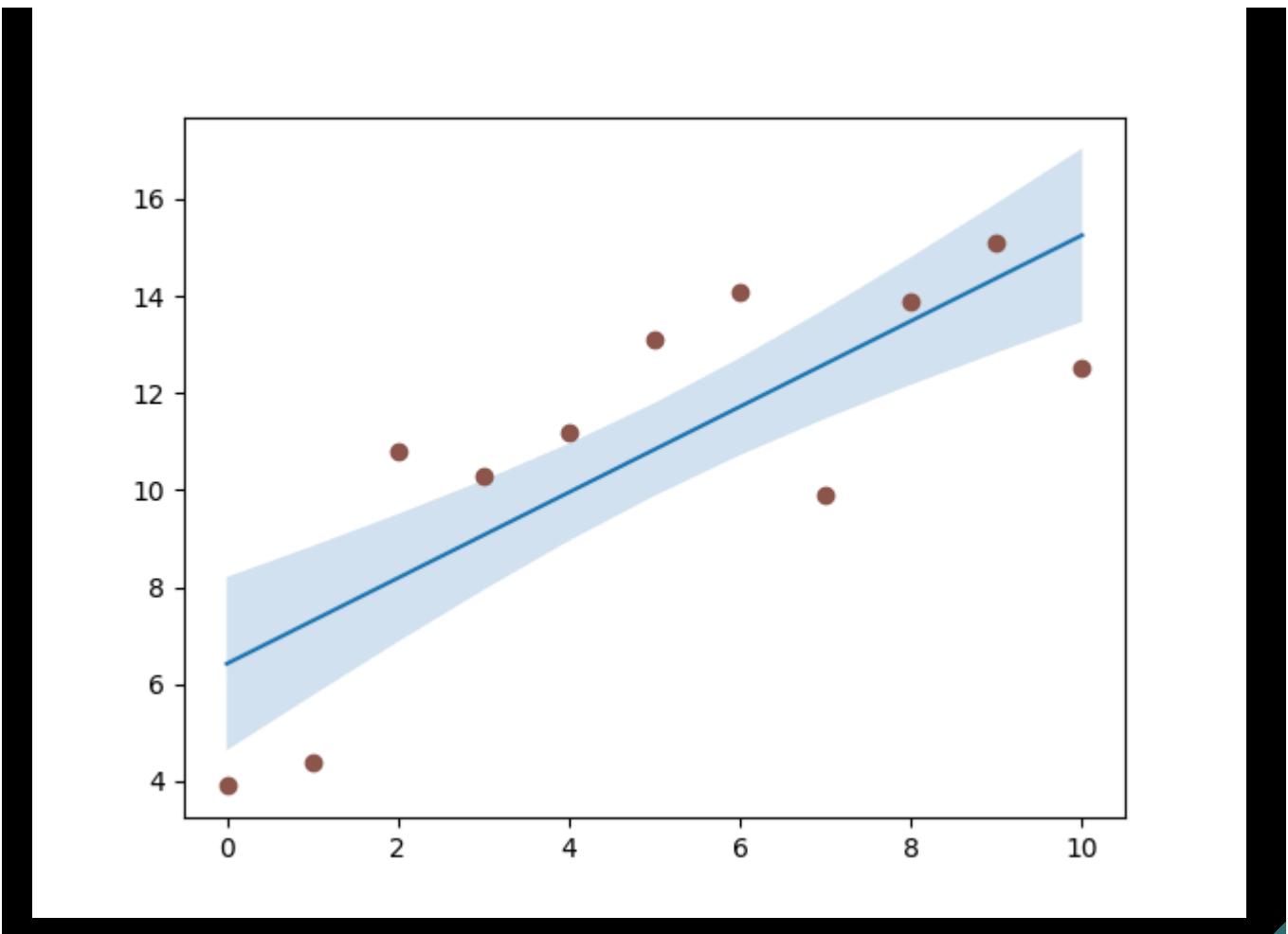


```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 N = 21
5 x = np.linspace(0, 10, 11) # [ 0.,  1.,  2.,  3., ..., 10.0]
6 y = [3.9, 4.4, 10.8, 10.3, 11.2, 13.1, 14.1, 9.9, 13.9, 15.1, 12.5]
7
8 # fit a linear curve an estimate its y-values and their error.
9 a, b = np.polyfit(x, y, deg=1)
10 y_est = a * x + b
11 y_err = x.std() * np.sqrt(1/len(x) +
12 ..... (x - x.mean())**2 / np.sum((x - x.mean())**
13
14 fig, ax = plt.subplots()
15 ax.plot(x, y_est, '-') # plot the fitted (regression) line
16 ax.fill_between(x, y_est - y_err, y_est + y_err, alpha=0.2)
17 ax.plot(x, y, 'o', color='tab:brown') # plot data points as brown c
18
19 #plt.show()
20 plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

foo.png



foo.png

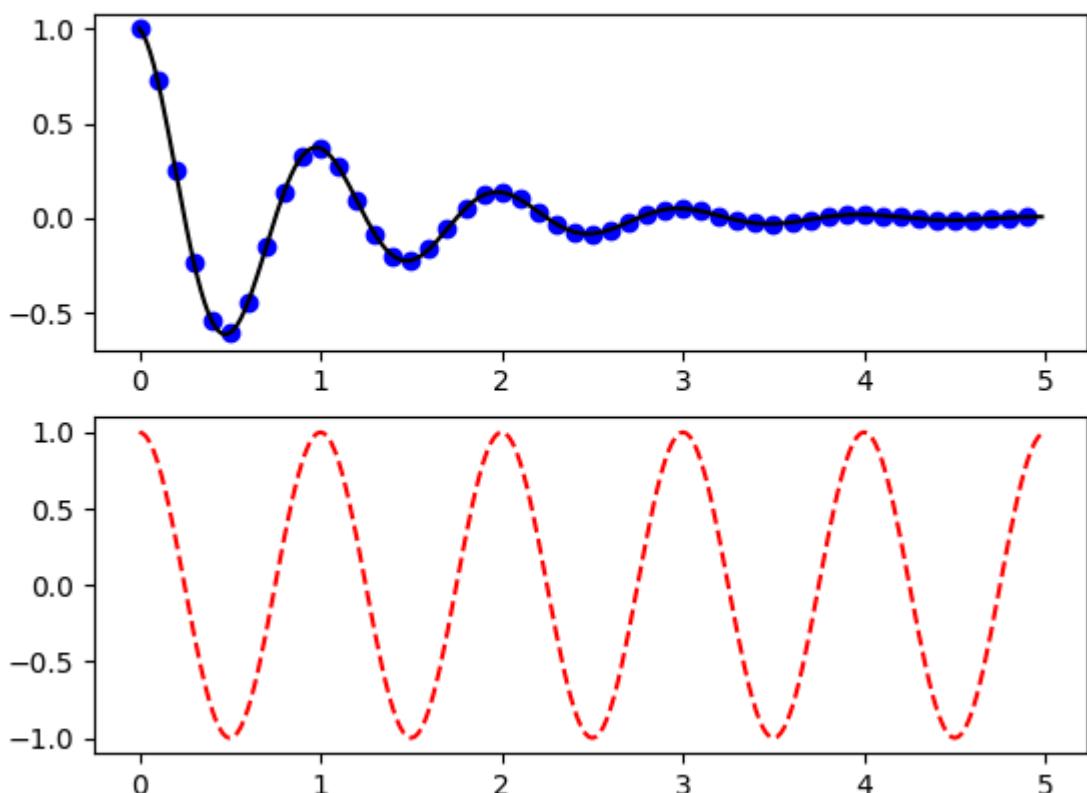


## MultipleDiagrams.py

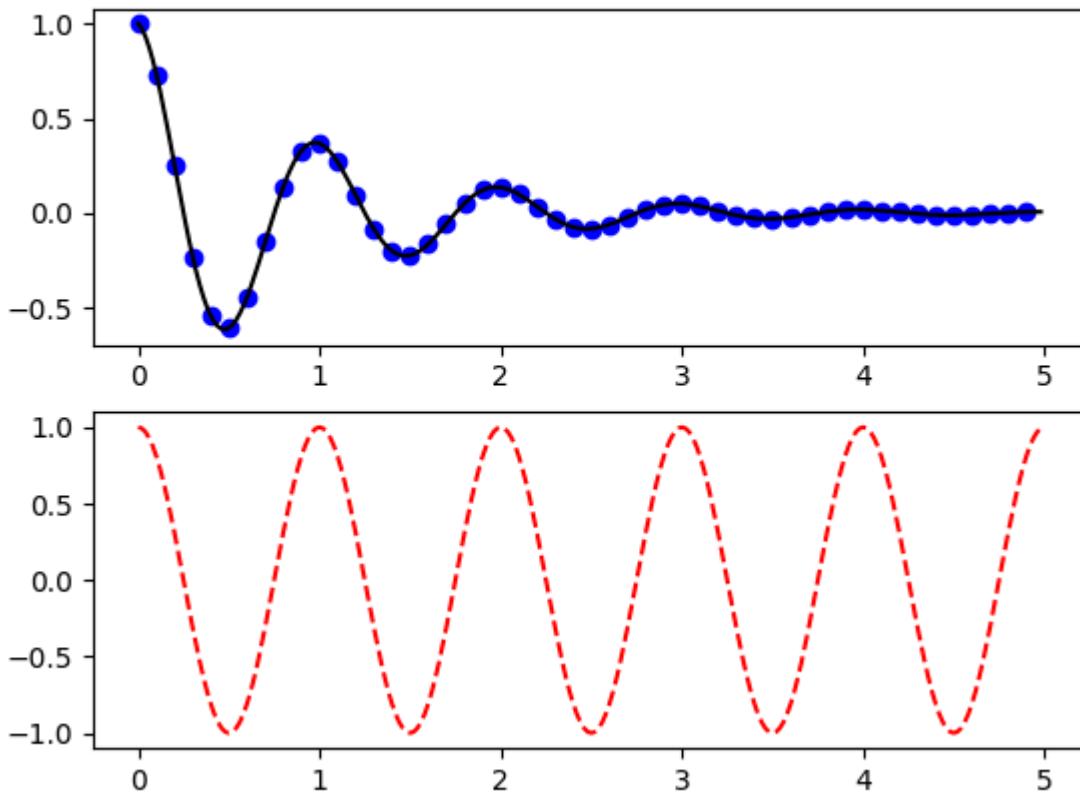


```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(t):
5     return np.exp(-t) * np.cos(2*np.pi*t)
6
7 t1 = np.arange(0.0, 5.0, 0.1)    # [0, 0.1, 0.2, ..., 4.9]
8 t2 = np.arange(0.0, 5.0, 0.02)   # [0., 0.02, 0.04, 0.06, ..., 4.98]
9
10 plt.figure()
11 plt.subplot(2, 1, 1) # 2 Zeilen, 1 Spalte: erstes Diagramm
12 # plt.subplot(211) # geht auch, falls Anzahl der Zeilen/Spalten <= 9
13 plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
14 # 'bo': blaue Kreise, 'k': schwarze Linie
15
16 plt.subplot(212) # 2 Zeilen, 1 Spalte: zweites Diagramm
17 plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
18 #plt.show()
19 plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript
```

foo.png



foo.png



## barchart.py

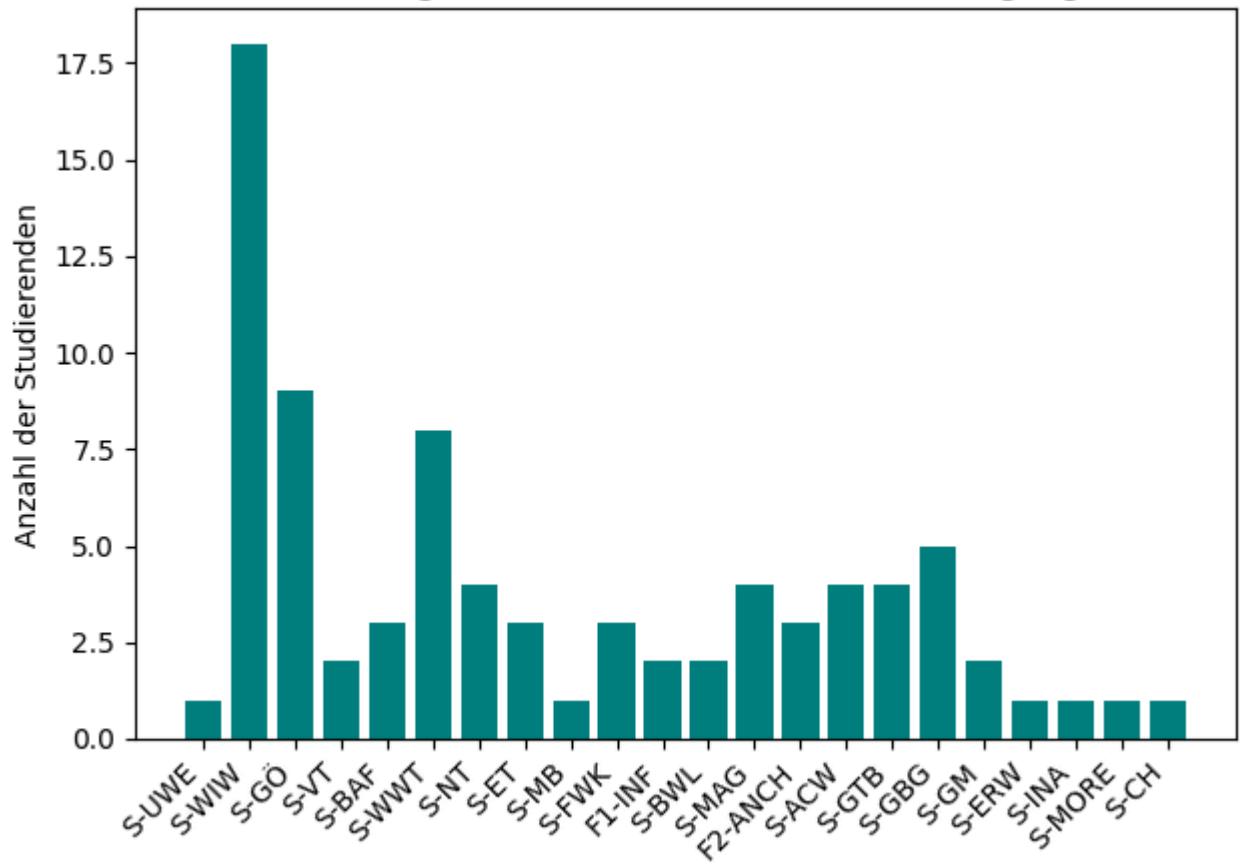
```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 data ={'S-UWE': 1, 'S-WIW': 18, 'S-GÖ': 9, 'S-VT': 2, 'S-BAF': 3, 'S-
8,
5 'S-NT': 4, 'S-ET': 3, 'S-MB': 1, 'S-FWK': 3, 'F1-INF': 2, 'S-BWL': 2
6 'S-MAG': 4, 'F2-ANCH': 3, 'S-ACW': 4, 'S-GTB': 4, 'S-GBG': 5, 'S-GM'
7 'S-ERW': 1, 'S-INA': 1, 'S-MORE': 1, 'S-CH': 1}
8
9 labels = list(data.keys())
10 values = list(data.values())
11
12 fig, ax = plt.subplots()
13 ax.bar(labels, values, color='teal')
14 ax.set_ylabel('Anzahl der Studierenden')
15 ax.set_title('Verteilung der Studierenden auf die Studiengänge')
16 plt.xticks(rotation=45, ha='right')
17 plt.tight_layout()
18 #plt.show()
19 plt.savefig('foo.png') # notwendig für die Ausgabe in LiaScript

```

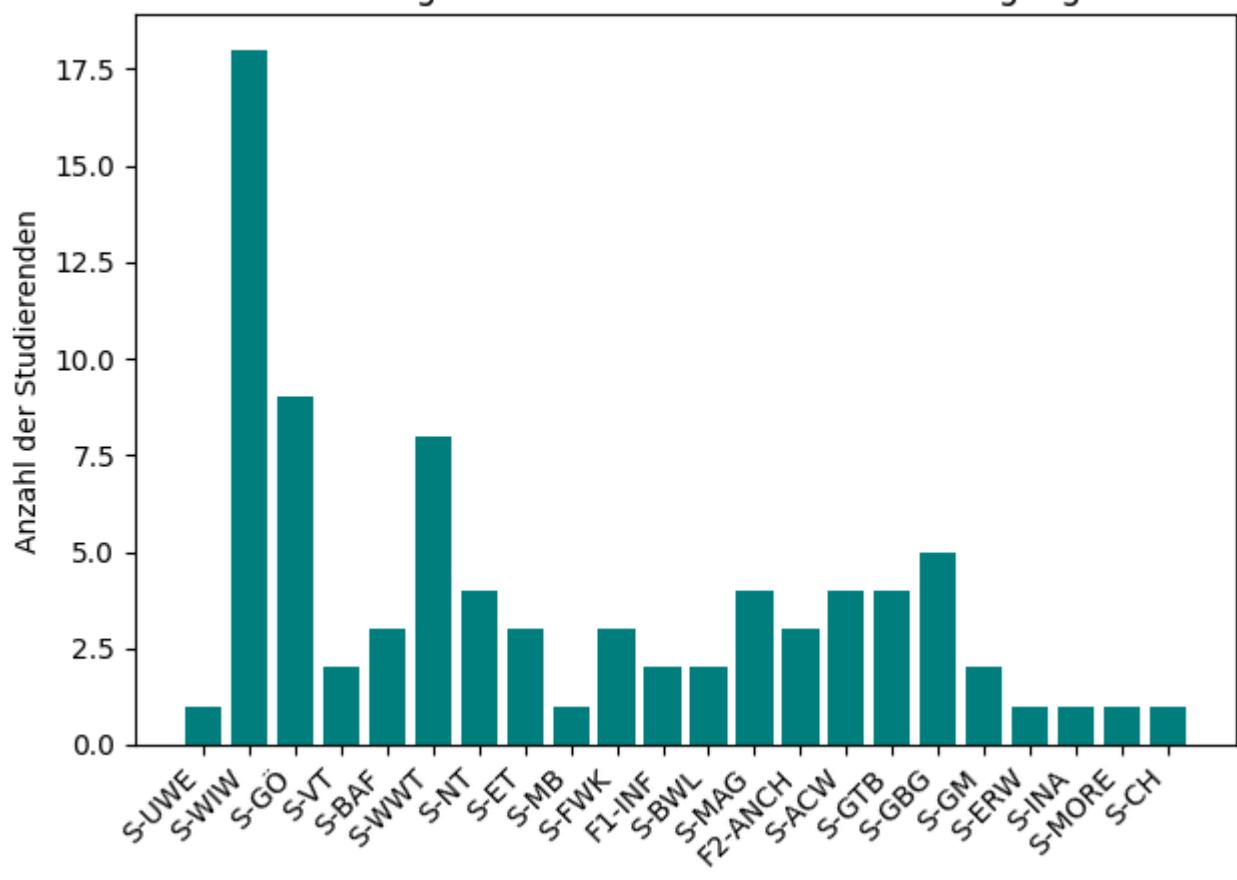
foo.png

Verteilung der Studierenden auf die Studiengänge



foo.png

Verteilung der Studierenden auf die Studiengänge



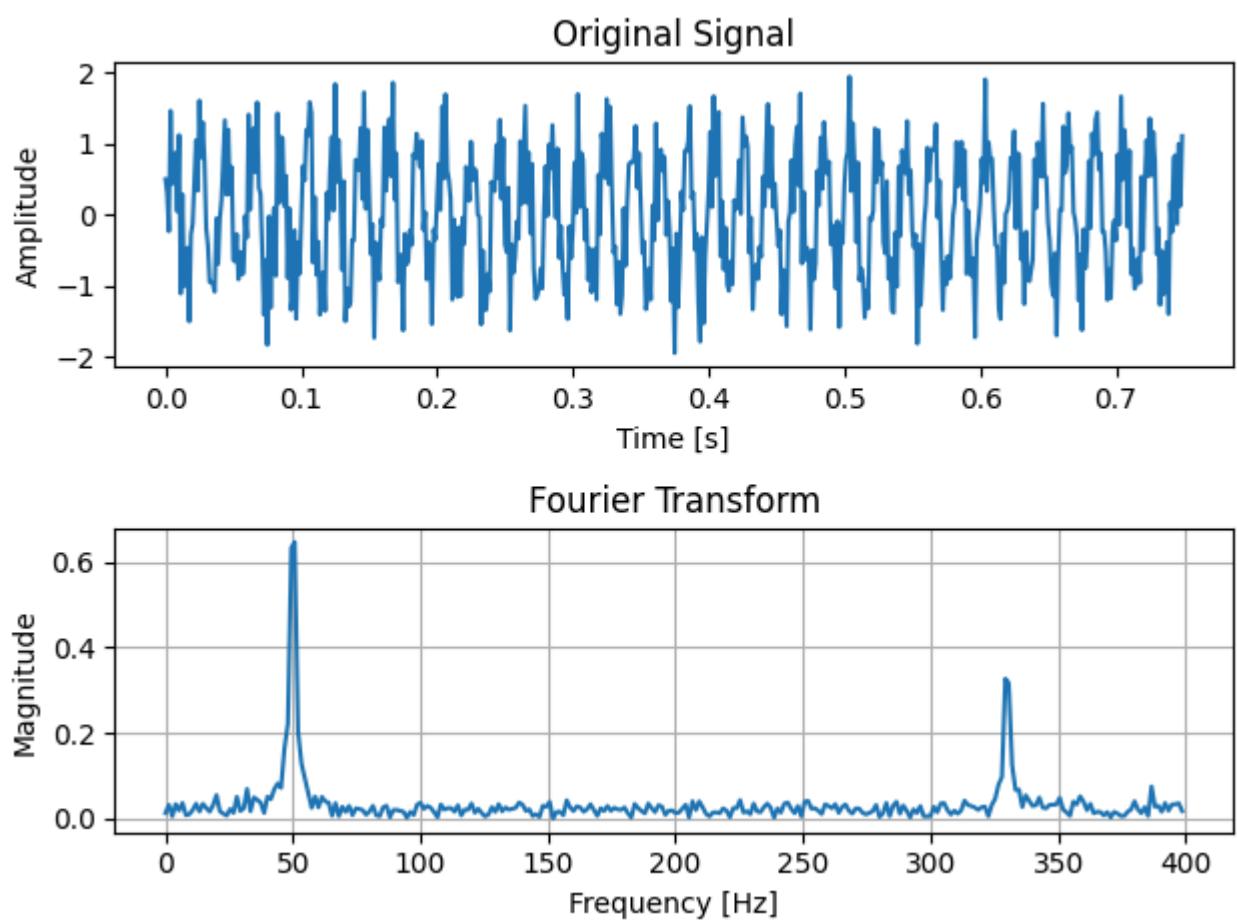
## Beispiel der Woche

## BeispielFourierTransformation.py

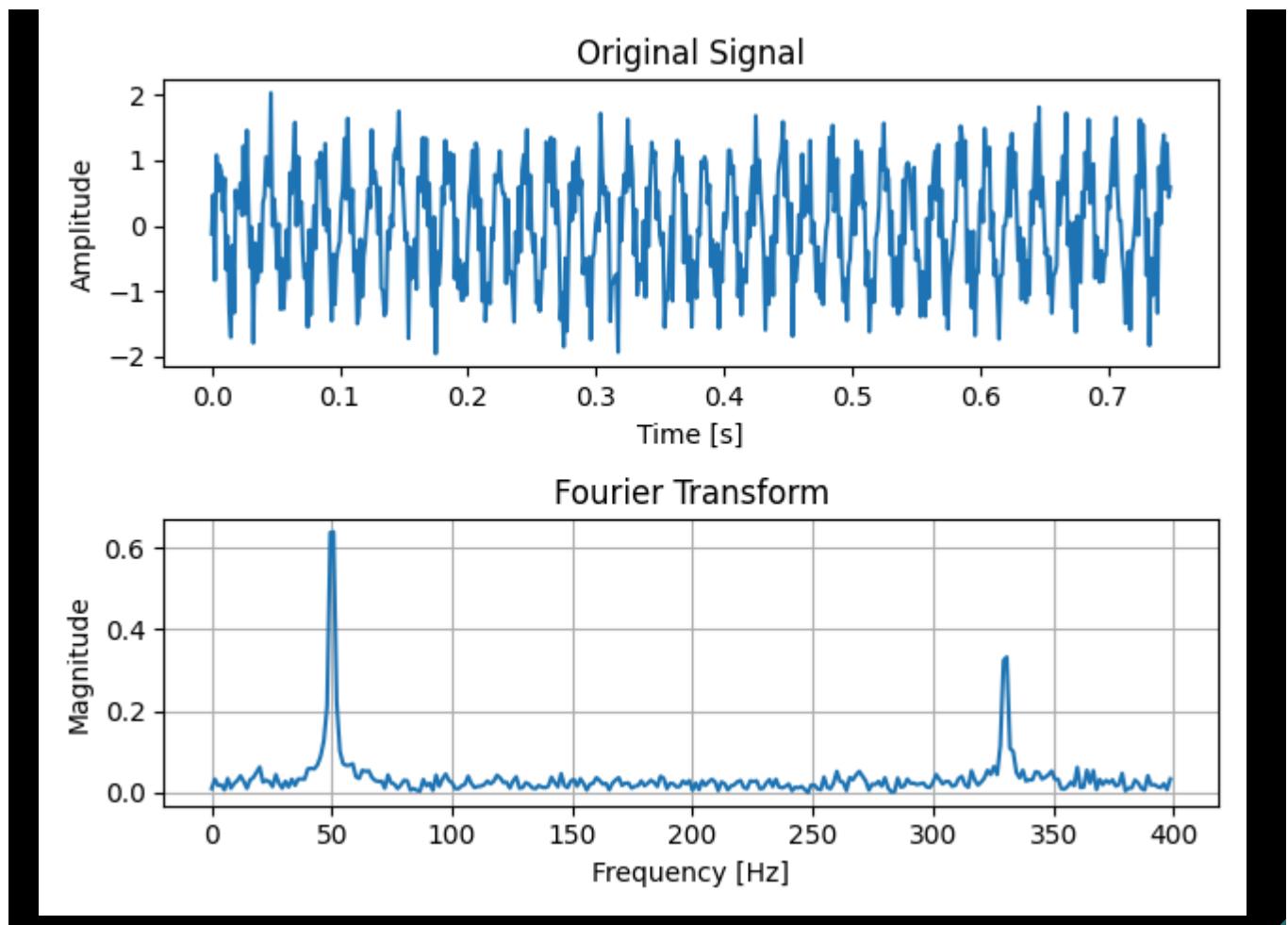


```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from scipy.fft import fft, fftfreq
4
5 # Number of sample points
6 N = 600
7
8 # sample spacing
9 T = 1.0 / 800.0
10 x = np.linspace(0.0, N*T, N, endpoint=False)
11
12 # create a signal with two frequencies (50 Hz and 330 Hz) and some no
13 y = np.sin(50.0 * 2.0*np.pi*x) # 50 Hz component
14 y += 0.5*np.sin(330.0 * 2.0*np.pi*x) # 330 Hz component, smaller ampl
15 y += 0.3*np.random.normal(size=x.shape) # add some noise (optional)
16
17 # compute the Fourier Transform and corresponding frequencies
18 yf = fft(y)
19 xf = fftfreq(N, T)[:N//2] # positive frequencies only
20
21 # plot the original signal and its Fourier Transform
22 fig, axs = plt.subplots(2, 1)
23 axs[0].plot(x, y)
24 axs[0].set_title('Original Signal')
25 axs[0].set_xlabel('Time [s]')
26 axs[0].set_ylabel('Amplitude')
27 axs[1].plot(xf, 2.0/N * np.abs(yf[0:N//2]))
28 axs[1].set_title('Fourier Transform')
29 axs[1].set_xlabel('Frequency [Hz]')
30 axs[1].set_ylabel('Magnitude')
31 plt.grid()
32 plt.tight_layout()
33
34 #plt.show()
35 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()
```