Algorithmische Lösung technischer Probleme

Einführung in Python

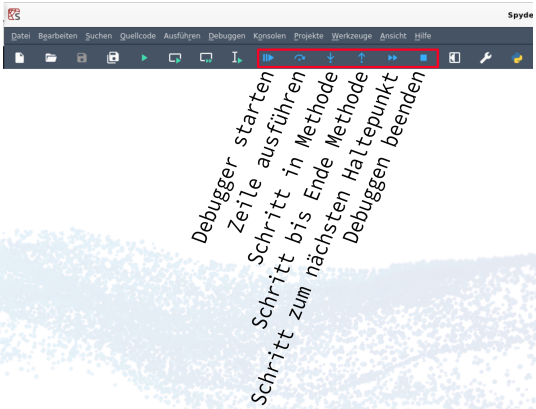
Debuggen

- Entfernen von Fehlern aus dem Programmcode
- in Python sehr interaktiv da interpretierte Sprache
- manuelles Setzen von Haltepunkten (Break Points)

Debuggen in Spyder

- Dokumentation
- Haltepunkt setzen

```
9   ### initial sine params ###
10  AMPL0 = 1
11  FREQ0 = 1  # Hz
12  PHASEANGLE0 = 0
```

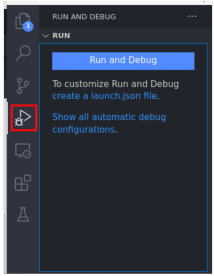


- interaktives Arbeiten am Haltepunkt
  - Variablen einsehen und manipulieren
  - schrittweise Programm ablaufen und nachvollziehen

Debuggen in VS-Code

- Dokumentation
- Haltepunkt setzen

```
 9   ### initial sine params ###
10   AMPL0 = 1
11   FREQ0 = 1   # Hz
12   PHASEANGLE0 = 0
```



- Debuggen im Grunde identisch zu Spyder, jedoch großer Unterschied bei der Darstellung der aktiven Variablen/Objekte und deren Inhalt

Umgang mit Dateien

- Dateien werden in Python mit „Filehandlern“-`fh` gehandhabt
- `open` erzeugt ein Filehandler-Objekt
  - `file` → Pfad zur Datei (string)
  - `mode` → `r` - read, `w` - write, ...
  - `encoding` → Dateikodierung (bei Problemen hilft meistens: `encoding='latin1'`)
- `with` schließt `fh` automatisch
- `as` weist einen Alias zu (`fh`)
- Methoden des Filehandlers:
  - `readline` → liest eine Zeile
  - `readlines` → liest alle/mehrere Zeilen
  - `write` → schreibt eine Zeile
  - `writelines` → schreibt mehrere Zeilen

```
with open(file, mode='r', buffering=-1, encoding=None, errors=None,
newline=None, closefd=True, opener=None) as fh:
    content = fh.readlines()
    for line in lines:
        ...
```

Umgang mit Dateien

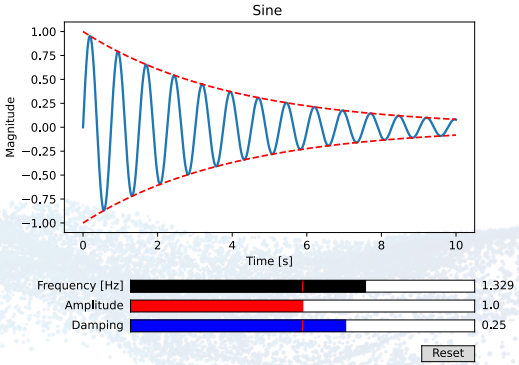
- Dateien werden als Strings eingelesen → Bearbeiten mit String-Methoden
  - `string.split(sep)`
  - `string.replace(old, new)`
  - ...
- für Fortgeschrittene: parsen mit „Regulären Ausdrücken“ (Regex)
  - `import re` (sehr mächtig)
  - regex101 zum üben, probieren und testen
- strukturierte Dateien können auch mit Bibliotheken eingelesen werden
  - `numpy`, `pandas`, `json`, ...

```
import numpy as np
data = np.genfromtxt(filename)
```

```
import pandas as pd
dataframe = pd.read_csv(filename)
```

Visualisieren mit matplotlib

- `import matplotlib.pyplot as plt`
- wichtige Befehle:
  - `plt.subplots()` → erstellt eine Figur mit Achsen
  - `plt.plot(x, y)` oder `axes.plot(x, y)` plottet Linie
  - `plt.scatter(x, y)` oder `axes.scatter(x, y)` plottet Punkte
  - `plt.show()` oder `figure.show()` zeigt Plot an
  - Beispiele und Anregungen



Visualisieren mit matplotlib

STRÖMUNGS MECHANIK | MASCHINEN

Individualisierung von Plots

- fertige Stylesheets
  - `plt.style.use("name-des-stylesheets")`
- eigene Anpassung mit `rcParams` oder `matplotlibRc`-Datei

```
import matplotlib as mpl
mpl.rcParams['lines.linewidth'] = 2 # Standardliniendicke = 2
```

mystyle.mplrc

```
lines.linewidth:  1.5    # line width in points
lines.linestyle:  -    # solid line
```