Programmieren – 10. Diagramme mit matplotlib

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

```
def spanne(werte):
    return werte[0], werte[-1]

a, b = spanne([3, 8, 5])
print(a, b)
```

Heute lernen Sie

- Daten als **Diagramm** darstellen: `matplotlib`
- Plots **beschriften**: Titel, Achsen, Gitter
- **Mehrere Datenreihen** und **markierte Punkte**
- Den Titel **vom Ergebnis abhängig** machen – Programme, die ihr Ergebnis bewerten

Der erste Plot

Kennen Sie schon: die Demo aus Woche 1

Heute bauen Sie sie selbst.

```
import matplotlib.pyplot as plt

stunden = [8, 9, 10, 11, 12, 13, 14, 15]
temperaturen = [18.2, 19.1, 21.4, 24.0, 26.3, 27.1, 25.8, 22.5]

plt.plot(stunden, temperaturen)
plt.show()
```

- `import matplotlib.pyplot as plt` – der Kurzname `plt` ist überall Standard
- `plt.plot(x, y)`: erste Liste ⟶ x-Achse, zweite Liste ⟶ y-Achse
- `plt.show()` zeigt das Bild an

Schritt für Schritt: Beschriftung

```
plt.plot(stunden, temperaturen)
plt.xlabel("Uhrzeit")
plt.ylabel("Temperatur in °C")
plt.title("Tagesverlauf")
plt.grid(True)
plt.show()
```

Ein Diagramm ohne Achsenbeschriftung ist keins.

Mini-Aufgabe (3 min)

Nehmen Sie den Minimalplot von eben und ergänzen Sie:

1. Achsenbeschriftungen (`xlabel` , `ylabel`)
2. Einen Titel
3. Das Gitter

Achtung typischer Fehler

```
plt.plot(stunden, temperaturen)
plt.show
```

Kein Fehler, keine Meldung – aber auch **kein Bild.**

- `plt.show` ohne Klammern **ruft die Funktion nicht auf** (Woche 6: erst die Klammern führen aus!)

Stil und mehrere Datenreihen

Stil-Strings: die Referenz

```
plt.plot(x, y, "b-")    # blau, durchgezogene Linie
plt.plot(x, y, "r--")   # rot, gestrichelt
plt.plot(x, y, "g:")    # grün, gepunktet
plt.plot(x, y, "ko")    # schwarz, nur Punkte
plt.plot(x, y, "b-o")   # blau, Linie mit Punkten
```

Farbe		Linie		Marker	
`b`	blau	`-`	durchgezogen	`o`	Punkt
`r`	rot	`--`	gestrichelt	`+`	Plus
`g`	grün	`:`	gepunktet	`s`	Quadrat
`k`	schwarz				

Zweite Datenreihe und markierte Punkte

```
plt.plot(stunden, temperaturen, "b-")
plt.plot(stunden, prognose, "r--")
plt.plot([13], [27.1], "ro")
plt.show()
```

- Jeder weitere `plot`-Aufruf **vor** `show()` zeichnet ins selbe Diagramm
- Ein **einzelner Punkt**: Listen mit nur einem Element plus Marker-Stil – so hebt man ein Maximum oder eine Nullstelle hervor

Mini-Aufgabe (3 min)

Ergänzen Sie in Ihrem Temperatur-Plot:

1. Den Höchstwert (Stunde 13, 27.1 °C) als **roten Punkt**
2. Zusatz: Finden Sie den Höchstwert nicht von Hand, sondern mit der Maximum-Schleife aus Woche 7

Der dynamische Titel: das Diagramm bewertet sein Ergebnis

```
maximum = temperaturen[0]
for t in temperaturen:
    if t > maximum:
        maximum = t

if maximum > 25:
    titel = f"Warnung: Höchstwert {maximum:.1f} °C"
else:
    titel = "Alles im Normalbereich"

plt.title(titel)
```

Je nachdem, was herauskommt, soll das Diagramm etwas anderes sagen – Titel in eine Variable, per `if` entscheiden, fertig.

Debug-Aufgabe (3 min)

Zwei Fehler – finden und beheben:

```
import matplotlib.pyplot as plt

zeiten = [0, 1, 2, 3]
werte = [0, 2, 8, 18]

plt.plot(zeiten werte)
plt.xlabel("Zeit in s")
plt.grid(True)
plt.show
```

Datenreihen berechnen statt eintippen

Messdaten kommen aus Dateien – berechnete Kurven aus **Schleifen und Funktionen**:

```
def restkapazitaet(zyklen):
    return 100 - 0.015 * zyklen

zyklen = []
kapazitaeten = []
for z in range(0, 1001, 100):
    zyklen.append(z)
    kapazitaeten.append(restkapazitaet(z))

plt.plot(zyklen, kapazitaeten, "b-o")
plt.show()
```

Das Muster *Listen füllen per Schleife, Werte aus einer Funktion* ist der Kern jeder berechneten Kurve.

Transfer

Aufgabe: Batterie-Alterung (Denkphase: 6 min, auf Papier)

Alle Bausteine des Semesters in einem Programm – **Eingabe → Funktion → Schleife → Plot**:

1. Zyklenzahl `n` **einlesen** (`int`)
2. Funktion `restkapazitaet(zyklen)` , die `100 - 0.015 * zyklen` zurückgibt (`return`)
3. **Schleife**: für 0 bis `n` in 100er-Schritten beide Listen füllen – Werte aus der Funktion
4. **Plot** mit Achsenbeschriftung und Gitter
5. **Titel per** `if` : unter 80 % Restkapazität → `Warnung: Batterie altert!` , sonst → `Batterie ok`

Notieren Sie zuerst die Bausteine und ihre Reihenfolge.

Zusatz für Schnelle: den letzten Punkt der Kurve rot markieren; die Eingabe validieren (Woche 5!)

Gemeinsam live

Wir entwickeln die Lösung jetzt zusammen.

Mini-Check

1. Was bedeutet der Stil-String `"r--"` ?
2. Wie kommt eine zweite Linie ins selbe Diagramm?
3. `plt.show` – warum erscheint kein Bild?
4. Wie markiert man einen einzelnen Punkt im Plot?

Bis nächste Woche!

Nächste Woche: **Zahlensysteme** – wie Computer Zahlen speichern, und warum `0.1 + 0.2` nicht `0.3` ist

- Üben: KI-Quizze auf OneTutor: https://hm.onetutor.ai/
- Fragen: jederzeit im Matrix-Chat