Programmieren – 4. Eingabe und Ausgabe

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

Aufschreiben, bevor wir es ausführen:

```
wert = 120

if wert < 1 and wert > 100:
    print("A")
elif wert > 100:
    print("B")
else:
    print("C")
```

Heute lernen Sie

- Programme, die **Eingaben** entgegennehmen: `input()`
- Eingaben in Zahlen **umwandeln**: `int()`, `float()`
- Ausgaben **formatieren**: f-Strings
- Das Muster, nach dem fast jedes Programm aufgebaut ist: **Eingabe → Verarbeitung → Ausgabe**

Eingabe

Eingaben lesen mit `input()`

```
name = input("Wie heißen Sie? ")
print("Hallo, " + name + "!")
```

- `input(...)` zeigt den Text an und **wartet**, bis Sie etwas eintippen und Enter drücken
- Die Eingabe landet in der Variable
- Wieder eine Funktion: Aufforderungstext rein, Eingabe raus

Vorhersage-Aufgabe (2 min)

Was passiert, wenn Sie hier `5` eintippen? Aufschreiben, dann ausführen:

```
zahl = input("Zahl: ")
print(zahl + zahl)
```

Warum kommt da 55 heraus?

`input()` liefert **immer** Text (`str`) – egal, was eingetippt wird:

```
zahl = input("Zahl: ")   # Eingabe: 5 → zahl ist "5"
print(zahl + zahl)       # "5" + "5" → "55"
```

- Kennen Sie aus Woche 2: `+` **verkettet** Text
- Wer rechnen will, muss erst **umwandeln**

Umwandeln: `int()` und `float()`

```
alter = int(input("Alter: "))
groesse = float(input("Größe in m: "))

print(alter + 1)
print(groesse * 100)
```

- `int(...)` wandelt Text in eine ganze Zahl, `float(...)` in eine Kommazahl
- Bei unsinniger Eingabe (`"abc"`) gibt es einen `ValueError` – Fehlermeldung lesen!

Mini-Aufgabe (3 min)

Schreiben Sie einen Reichweiten-Rechner für ein Elektrofahrzeug:

1. Nutzbaren Energieinhalt der Batterie in kWh einlesen (`float`)
2. Verbrauch in kWh/100 km einlesen (`float`)
3. Reichweite berechnen: Energieinhalt geteilt durch Verbrauch, mal 100
4. Ergebnis ausgeben (`print`)

Testen Sie mit 60 kWh und 15 kWh/100 km – das sollte 400 km ergeben.

Achtung typischer Fehler

```
energie = input("Energieinhalt in kWh: ")
haelfte = energie / 2
```

```
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

- Umwandlung vergessen: `energie` ist Text, und Text lässt sich nicht dividieren
- `TypeError` mit `str` in der Meldung nach einem `input()`: fast immer ein fehlendes `int(...)` oder `float(...)`

Ausgabe

Schöner ausgeben: f-Strings

```
name = "Ada"
reichweite = 397.43589743589746

print(f"Hallo {name}!")
print(f"Reichweite: {reichweite} km")
```

- `f` **vor** dem Anführungszeichen
- Alles in `{...}` wird durch den Wert ersetzt – auch Ausdrücke: `f"{energie / 2} kWh"`

Formatieren: Nachkommastellen und Feldbreite

```
reichweite = 397.43589743589746
messwert = 42

print(f"Reichweite: {reichweite:.1f} km")
print(f"|{messwert:>6}|")
print(f"|{reichweite:>8.1f}|")
```

- `:.1f` – eine Nachkommastelle (`.3f` – drei, …)
- `:>6` – rechtsbündig in 6 Zeichen Breite (praktisch für Tabellen)
- Beides kombinierbar – und beides begegnet Ihnen ab jetzt jede Woche: Sie kennen es bald auswendig, ohne es zu merken

Mini-Aufgabe (2 min)

Zurück zu Ihrem Reichweiten-Rechner:

Ändern Sie die Ausgabe (`print`) so, dass die Reichweite mit **genau einer Nachkommastelle** erscheint:

```
Reichweite: 400.0 km
```

Debug-Aufgabe (3 min)

Dieses Programm stürzt ab. Finden Sie den Fehler und beheben Sie ihn:

```python
alter = input("Alter: ")
naechstes_jahr = alter + 1
print(f"Nächstes Jahr sind Sie {naechstes_jahr}")
```

Das Muster: Eingabe → Verarbeitung → Ausgabe

```python
# 1. Eingabe
u = float(input("Spannung in V: "))
i = float(input("Strom in A: "))

# 2. Verarbeitung
r = u / i

# 3. Ausgabe
print(f"Widerstand: {r:.2f} Ohm")
```

Fast jedes Programm folgt diesem Muster – vom Mini-Skript bis zur Messdaten-Auswertung.

Transfer

Aufgabe: Akkulaufzeit 2.0 (Denkphase: 5 min, auf Papier)

Unser Programm aus Woche 2 – jetzt mit echter Eingabe:

1. Kapazität in mAh **einlesen** (`float`)
2. Stromaufnahme in mA **einlesen** (`float`)
3. Laufzeit in Stunden **berechnen**
4. **Ausgeben** (`print`) mit einer Nachkommastelle: `Laufzeit: 13.7 Stunden`

Notieren Sie zuerst die drei Phasen: Was wird eingelesen? Was berechnet? Was ausgegeben?

Zusatz für Schnelle: Geben Sie eine Warnung aus (`print`), wenn die Laufzeit unter 8 Stunden liegt – Woche 3 lässt grüßen.

Gemeinsam live

Wir entwickeln die Lösung jetzt zusammen.

Mini-Check

Ohne Computer – was kommt heraus?

1. Welchen Datentyp liefert `input()` – immer?
2. `print(f"{2.5:.2f}")`
3. `eingabe = input(...)` – der Benutzer tippt `12`. Was gibt `print(eingabe + 1)`?
4. Wie breit ist die Ausgabe von `f"{7:>4}"` – und wo steht die 7?

Bis nächste Woche!

Nächste Woche: **Schleifen** – Programme, die Dinge wiederholen

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