Programmieren – 2. Erste Programme

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

Aufschreiben, bevor wir es ausführen:

```
print(3 * 7)
print(3 * 7.0)
print("3 * 7")
```

Heute lernen Sie

- Werte in **Variablen** speichern und wiederverwenden
- Die drei wichtigsten **Datentypen** unterscheiden: ganze Zahlen, Kommazahlen, Text
- Mit Python **rechnen** – und vorhersagen, was ein kleines Programm ausgibt

Arbeitsumgebung wie letzte Woche: https://davidstraub.de/teaching-materials/lite/lab/ – oder dieses Deck direkt als Notebook: „▸ Run“ auf der Kursseite

Variablen

Variablen speichern Werte

```
temperatur = 21.5
print(temperatur)
```

- Links: der **Name** (frei wählbar)
- Rechts: der **Wert**
- `=` bedeutet: *speichere den Wert unter diesem Namen*

```
temperatur = 23.0
print(temperatur)
```

Der Name zeigt immer auf den **zuletzt gespeicherten** Wert.

`=` ist eine Zuweisung – keine Gleichung

In der Mathematik unsinnig, in Python Alltag:

```
zaehler = 10
zaehler = zaehler + 1
print(zaehler)
```

- `=` ist eine **Zuweisung**: rechts wird **zuerst** ausgerechnet (mit dem alten Wert), **dann** links gespeichert
- Gedanklich als Pfeil lesen: `zaehler ← zaehler + 1`

Namen wählen

```
u_batterie = 12.6     # gut: sagt, was drinsteckt
x2 = 12.6             # schlecht: sagt nichts
```

- Buchstaben, Ziffern, `_` – kein Start mit Ziffer, keine Leerzeichen *(Syntax)*
- Groß-/Kleinschreibung zählt: `wert` ≠ `Wert` *(Syntax)*
- Kleinbuchstaben, Wörter mit `_` verbinden: `snake_case` *(Konvention)*
- Aussagekräftig: `spannung` statt `s` *(Konvention)*

Mini-Aufgabe (3 min)

```
seite = 5
flaeche = seite * seite
print(flaeche)
```

1. Führen Sie den Code aus
2. Ändern Sie `seite` auf `8` und führen Sie erneut aus
3. Ergänzen Sie eine Variable `umfang` (= `4 * seite`) und geben Sie sie aus (`print`)

Achtung typischer Fehler

```
flaeche = seite * seite
print(flache)
```

```
NameError: name 'flache' is not defined
```

- Python kennt nur **exakt** geschriebene Namen
- Fehlermeldungen von **unten** lesen: letzte Zeile sagt *was*, Pfeil davor zeigt *wo*

- `NameError` heißt fast immer: Tippfehler oder Variable noch nicht angelegt

Datentypen

Drei Typen für den Anfang

```
anzahl = 12          # int  – ganze Zahl
spannung = 12.6      # float – Kommazahl
name = "Zelle A"     # str  – Text, in Anführungszeichen
```

Welchen Typ ein Wert hat, verrät `type(...)`:

```
print(type(spannung))
```

Übrigens: `print(...)` und `type(...)` sind **Funktionen** – Sie rufen sie mit runden Klammern auf. Davon kommen noch viele.

Rechnen mit Zahlen

```
a = 10
b = 3
print(a + b)
print(a - b)
print(a * b)
print(a ** 2)
```

`**` ist Potenzieren: `a ** 2` ist a^2.

Division: Vorhersage-Aufgabe

Aufschreiben, dann ausführen:

```
a = 10
b = 4
print(a / b)
print(type(a / b))
print(10 / 2)
print(type(10 / 2))
```

Warum kommt da immer ein float heraus?

`/` liefert in Python **immer** eine Kommazahl – auch wenn es aufgeht:

```
print(10 / 2)
```

- Python kann vorher nicht wissen, ob die Division aufgeht
- Darum ist das Ergebnis von `/` **immer** `float`
- Genau die Frage aus dem Warm-up: `3 * 7` bleibt `int`, `3 * 7.0` wird `float` – sobald ein `float` mitrechnet, ist das Ergebnis `float`

Der Rest-Operator `%`

`a % b` liefert den **Rest** der Division:

```
print(17 % 5)
print(20 % 4)
print(7 % 2)
```

- `x % 2` verrät: gerade (`0`) oder ungerade (`1`) – und Reste braucht man überall: Takte, Zyklen, Prüfziffern

Vorhersage-Aufgabe (2 min)

Ohne Taschenrechner – aufschreiben, dann ausführen:

```
print(9999999 % 4)
print(9999999 % 2)
print(1000000 % 10)
```

Tipp: Wie viel vom Anfang der Zahl brauchen Sie wirklich?

Text: der Typ `str`

```
zelle = "Batteriezelle A"
print(zelle)
print(len(zelle))
```

- Text steht in Anführungszeichen: `"..."` oder `'...'`
- `len(...)` – noch eine Funktion: liefert die Länge
- `+` **verkettet** Texte:

```
vorname = "Ada"
nachname = "Lovelace"
print(vorname + " " + nachname)
```

Debug-Aufgabe (3 min)

Dieses Programm soll `77` ausgeben – tut es aber nicht. Finden Sie beide Fehler:

```
zahl = "7"
doppelt = zahl + zahl
print(Doppelt)
```

Zahl oder Text?

```
print(7 + 7)
print("7" + "7")
print(3 * "7")
```

- Gleiches `+`, zwei Bedeutungen: **rechnen** bei Zahlen, **verketten** bei Text
- `"7"` ist Text, keine Zahl – Anführungszeichen entscheiden
- Und `*` mit Text? **Wiederholen** – aufschreiben, dann ausführen!
- Mischen mit `+` geht schief: `7 + "7"` gibt einen `TypeError`

Transfer

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

Ein Akku hat eine Kapazität von **4800 mAh**. Ein Messgerät zieht dauerhaft **350 mA**.

Schreiben Sie ein Programm, das ausgibt (`print`):

1. die Kapazität,
2. den Strom,
3. die Laufzeit in Stunden.

Notieren Sie zuerst: Welche Variablen brauchen Sie? Welche Rechnung?

Zusatz für Schnelle: Geben Sie die Laufzeit auch in Minuten aus (`print`).

Gemeinsam live

Wir entwickeln die Lösung jetzt zusammen.

Mini-Check

Ohne Computer – was kommt heraus?

1. `print(type(10 / 2))`
2. `x = 5` und dann `x = x + 2` – was gibt `print(x)` aus?
3. `print(17 % 5)`
4. `print("3" + "4")`

Bis nächste Woche!

Nächste Woche: **Verzweigungen** – Programme, die Entscheidungen treffen

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