Programmieren – 11. Zahlensysteme

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

```
x = [0, 1, 2]
y = []
for wert in x:
    y.append(wert ** 2)
print(y)
```

Heute lernen Sie

- Wie Computer Zahlen speichern: **Bits, Bytes, Binärsystem**
- Umrechnen: **binär ↔ dezimal ↔ hexadezimal** – mit nachvollziehbaren Zwischenschritten
- Warum `0.1 + 0.2 == 0.3` den Wert `False` hat – und wie man mit Kommazahlen richtig vergleicht

Bits und Bytes

Alles ist 0 oder 1

- Ein **Bit**: die kleinste Speichereinheit – `0` oder `1`
- Ein **Byte**: 8 Bit
- Mit n Bit lassen sich 2^n verschiedene Werte darstellen

Beispiel: Eine Ganzzahl wird mit 8 Byte gespeichert – wie viele unterschiedliche Werte lassen sich darstellen?

8 Byte = 64 Bit $\longrightarrow 2^{64}$ Werte. Die Zweierpotenz **ist** die Antwort – niemand rechnet die 20-stellige Zahl aus.

Das Stellenwertsystem

Dezimal (Basis 10) – so lesen Sie Zahlen seit der Grundschule:

$$257 = 2 \cdot 100 + 5 \cdot 10 + 7 \cdot 1$$

Binär (Basis 2) – dasselbe Prinzip, andere Stellenwerte:

$$1101_2 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 13$$

Stellenwerte von rechts: 1, 2, 4, 8, 16, 32, 64, 128, …

Binär → Dezimal

$$110011_2 = ?$$

Stellenwert	32	16	8	4	2	1
Ziffer	1	1	0	0	1	1

$$32 + 16 + 2 + 1 = 51$$

Nur die Stellen mit `1` addieren – mehr ist es nicht.

Dezimal → Binär

$45 = ?_2$ – fortgesetzte Division durch 2, **Reste notieren**:

		Rest
45 : 2	= 22	**1**
22 : 2	= 11	**0**
11 : 2	= 5	**1**
5 : 2	= 2	**1**
2 : 2	= 1	**0**
1 : 2	= 0	**1**

Reste **von unten nach oben** lesen: 101101_2. Probe: $32 + 8 + 4 + 1 = 45$ ✓

Mini-Aufgabe (4 min, auf Papier)

Rechnen Sie um – **mit allen Zwischenschritten**:

$$123_{10} = ?_2$$

Und dann die Probe: rechnen Sie Ihr Ergebnis zurück ins Dezimalsystem.

Hexadezimal: Basis 16

- Ziffern: `0` – `9`, dann `A` (10) bis `F` (15)
- Warum? **4 Bit = genau eine Hex-Ziffer** – kompakte Schreibweise für Bitmuster
- Stellenwerte: 1, 16, 256, …

$$2A_{16} = 2 \cdot 16 + 10 = 42$$

Und rückwärts: $123_{10} = 7 \cdot 16 + 11 \rightarrow 7B_{16}$

Mini-Aufgabe (3 min, auf Papier)

$$AAA_{16} =?_{10}$$

Zur Übersetzung der Ziffern (`0` – `9` zählen wie im Dezimalsystem):

Hex	0…9	A	B	C	D	E	F
Dezimal	0…9	10	11	12	13	14	15

Kommazahlen im Binärsystem

Stellen hinter dem Komma

Vor dem Komma: 1, 2, 4, 8, … – **hinter** dem Komma geht es genauso weiter:

$$\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots$$

$$0,25_{10} = 0,01_2 \qquad 0,75_{10} = 0,11_2 \qquad 0,625_{10} = 0,101_2$$

Eine Kommazahl umrechnen – langsam

$$257,25_{10} =?_2$$

Vorkomma 257: Division mit Rest (wie eben) ⟶ 100000001_2

Nachkomma 0,25: fortgesetztes **Verdoppeln**, Vorkommastelle notieren:

		Vorkomma
0,25 · 2	= 0,5	**0**
0,5 · 2	= 1,0	**1** – Rest 0, fertig

Von oben lesen: $0,01_2$. Zusammen: $257,25_{10} = 100000001,01_2$

Warum ist 0,1 im Computer nicht exakt 0,1?

Dasselbe Verfahren für $0,1_{10}$:

		Vorkomma
$0{,}1 \cdot 2$	$= 0{,}2$	0
$0{,}2 \cdot 2$	$= 0{,}4$	0
$0{,}4 \cdot 2$	$= 0{,}8$	0
$0{,}8 \cdot 2$	$= 1{,}6$	1
$0{,}6 \cdot 2$	$= 1{,}2$	1
$0{,}2 \cdot 2$	$= \ldots$	**wieder 0,2 – Zyklus!**

$$0,1_{10} = 0,\overline{0011}_2 \quad \text{– unendlich periodisch}$$

Der Computer speichert endlich viele Stellen – **0,1 ist im Rechner nie exakt 0,1.**

Erinnern Sie sich? Die Batterie-Schleife aus Woche 1

```
ladezustand = 0.0
while ladezustand != 1.0:
    ladezustand = ladezustand + 0.1
```

Zehnmal ein *fast*–0,1 ergibt *fast*–1,0 – aber nie **exakt** 1,0. Die Bedingung wurde nie falsch. Deshalb hing die Schleife.

Vorhersage-Aufgabe (1 min)

Aufschreiben, dann ausführen:

```
print(0.1 + 0.2 == 0.3)
print(0.1 + 0.2)
```

Erklär-Aufgabe (3 min, auf Papier)

Dieses Skript gibt `10029.999999999996362` aus. Erklären Sie in zwei Sätzen, warum eine „krumme“ Zahl erscheint:

```
x = 100.3
summe = 0
for i in range(100):
    summe += x
print(f"{summe:.15f}")
```

Und wie macht man es richtig?

- Floats nie mit `==` vergleichen – mit **Toleranz**:

```python
a = 0.1 + 0.2
if abs(a - 0.3) < 1e-9:
    print("praktisch gleich")
```

- Und die Batterie-Schleife von Woche 1, korrekt:

```python
while ladezustand < 1.0:
    ladezustand = ladezustand + 0.1
```

`<` statt `!=` – die Schleife endet, sobald 1,0 erreicht oder überschritten ist.

So üben Sie Umrechnungen

- KI-Quizze auf **OneTutor**: https://hm.onetutor.ai/
- Oder direkt mit einem Chatbot: Lassen Sie sich Umrechnungsaufgaben stellen – und Ihre Lösungen prüfen

Mini-Check

1. Wie viele Werte lassen sich mit 10 Bit darstellen?
2. 1010_2 – dezimal?
3. Warum ist 0,1 binär nicht exakt darstellbar – in einem Satz?
4. Wie prüft man zwei floats auf „gleich“?

Bis nächste Woche!

Nächste Woche: **Eigene Datentypen mit Klassen** – und der Semesterrückblick

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