DS 2 – Operatoren: rechnen, vergleichen, verknüpfen

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- die Rechenoperatoren `//`, `%` und `**` benutzen und erklären, was sie berechnen,
- mit `//` und `%` Minuten in Stunden und Minuten umrechnen und prüfen, ob eine Zahl gerade ist,
- zwei Werte mit `==`, `!=`, `<`, `>`, `<=` und `>=` vergleichen,
- Wahrheitswerte mit `and`, `or` und `not` verknüpfen und das Ergebnis vorhersagen,
- die Zuweisung `=` und den Vergleich `==` sicher unterscheiden.

🗣️ Sprachhilfe

- *„Da kommt True raus."* → „Für `zahl = 7` liefert der Vergleich `zahl > 4` den Wahrheitswert `True`, weil 7 größer als 4 ist."
- *„Ich bekomme raus, ob die Zahl größer als 4 ist."* → „Die Anweisung prüft, ob eine Zahl größer als 4 ist, und gibt das Ergebnis als Wahrheitswert aus."
- *„Prozent rechnet den Rest."* → „Der Modulo-Operator `%` liefert den Rest der Ganzzahldivision."

Worum es geht

Ein Film dauert 135 Minuten. Wie viele Stunden und Minuten sind das? Ist eine Zahl gerade? Darf eine 15-Jährige ohne Ausweis ins Kino? Für solche Fragen muss ein Programm **rechnen**, **vergleichen** und Aussagen **verknüpfen**. Die Zeichen dafür heißen **Operatoren**.

📌 **Definition:** Ein **Operator** ist ein Zeichen oder Wort, das mit einem oder mehreren Werten (den **Operanden**) einen neuen Wert berechnet. In `7 + 3` ist `+` der Operator, `7` und `3` sind die Operanden. Eine solche Verbindung heißt **Ausdruck**.

1 Arithmetische Operatoren

Operator	Name	Beispiel	Ergebnis
`+`	Addition	`7 + 3`	`10`
`-`	Subtraktion	`7 - 3`	`4`
`*`	Multiplikation	`7 * 3`	`21`
`/`	Division	`7 / 3`	`2.3333333333333335`
`//`	Ganzzahldivision	`7 // 3`	`2`
`%`	Modulo (Rest)	`7 % 3`	`1`
`**`	Potenz	`7 ** 3`	`343`

Ganzzahldivision `//` und Rest `%`

17 Kekse werden gerecht an 5 Kinder verteilt. Jedes Kind bekommt 3 Kekse, 2 bleiben übrig: 17 : 5 = 3 Rest 2.

```python
print(17 // 5)
print(17 % 5)
print(17 / 5)
```

```
3
2
3.4
```

- **Zeile 1:** `//` gibt an, wie oft die 5 **ganz** in die 17 passt: 3-mal.
- **Zeile 2:** `%` (**Modulo**) liefert den **Rest**: 17 − 3 · 5 = 2.
- **Zeile 3:** Die normale Division `/` rechnet mit Nachkommastellen.

Probe: 3 · 5 + 2 = 17. Das Zeichen `%` hat in Python **nichts mit Prozent** zu tun.

Anwendung: Minuten in Stunden und Minuten

```python
minuten_gesamt = int(input("Wie viele Minuten dauert der Film? "))
stunden = minuten_gesamt // 60
minuten = minuten_gesamt % 60
print(minuten_gesamt, "Minuten sind", stunden, "Stunden und",
      minuten, "Minuten.")
```

```
Wie viele Minuten dauert der Film? 135
135 Minuten sind 2 Stunden und 15 Minuten.
```

- **Zeile 2:** `135 // 60` ergibt `2`, die 60 passt zweimal ganz in die 135.
- **Zeile 3:** `135 % 60` ergibt `15`, denn 135 − 2 · 60 = 15.
- **Zeilen 4–5:** Innerhalb der Klammern von `print` darf man in der nächsten Zeile weiterschreiben.

🧪 **Probiere es aus:** Starte mit `60`, `59` und `600`. Was passiert bei `0`?

Anwendung: Ist eine Zahl gerade?

Eine Zahl ist **gerade**, wenn beim Teilen durch 2 der Rest `0` bleibt.

```python
print(14 % 2, 15 % 2, 2024 % 4, 2026 % 4)
```

```
0 1 0 2
```

> 📌 **Merksatz:** Eine Zahl `a` ist genau dann durch `b` teilbar, wenn `a % b` den Wert `0` ergibt. Eine Zahl ist gerade, wenn `zahl % 2` gleich `0` ist.

Potenzieren mit `**`

`2 ** 3` bedeutet $2^3 = 2 \cdot 2 \cdot 2 = 8$. Die erste Zahl ist die Basis, die zweite der Exponent.

```python
print(2 ** 3, 10 ** 6, 5 ** 2)
```

```
8 1000000 25
```

⚠️ Stolperstein: `^` ist nicht „hoch“

In Tabellenkalkulationen schreibt man „hoch“ mit `^`. In Python gibt das **keine** Fehlermeldung, aber ein falsches Ergebnis. 🔮 **Vorhersage:** Was kommt heraus?

```python
print(2 ^ 3)
```

```
1
```

Für Potenzen brauchst du in Python immer `**`.

> 📌 **Merksatz:** Arithmetische Operatoren: `+`, `-`, `*`, `/` (Ergebnis immer `float`), `//` (Ganzzahldivision), `%` (Rest) und `**` (Potenz).

Was wird zuerst gerechnet?

Klammern, dann `**`, dann Punktrechnung (`*`, `/`, `//`, `%`), zuletzt `+` und `-`.

```
print(2 + 3 ** 2)
print((2 + 3) ** 2)
print(10 - 7 % 4)
```

```
11
25
7
```

- **Zeile 1:** Erst $3^2 = 9$, dann 2 + 9 = 11.
- **Zeile 2:** Erst die Klammer 2 + 3 = 5, dann $5^2 = 25$.
- **Zeile 3:** Erst `7 % 4` = 3, dann 10 − 3 = 7.

Wenn du unsicher bist, setze Klammern. Sie schaden nie.

🔮 Übung 1 – Rechenausdrücke

Bestimme jedes Ergebnis **zuerst im Kopf**, dann überprüfe in der Codezelle.

Ausdruck	`23 // 4`	`23 % 4`	`23 / 4`	`2 ** 5`	`10 % 2`	`7 % 10`	`3 + 2 ** 3`	`(3 + 2) ** 2`
Ergebnis								

```
print(23 // 4)
# TODO: die übrigen sieben Ausdrücke prüfen
```

```
5
```

2 Vergleichsoperatoren: wahr oder falsch?

Viele Fragen beantwortet man mit Ja oder Nein. Python antwortet mit einem **Wahrheitswert**: `True` oder `False` (Datentyp `bool`).

Operator	Bedeutung	Beispiel	Ergebnis
`==`	ist gleich	`7 == 3`	`False`
`!=`	ist ungleich	`7 != 3`	`True`
`<`	ist kleiner als	`7 < 3`	`False`
`>`	ist größer als	`7 > 3`	`True`
`<=`	ist kleiner oder gleich	`7 <= 7`	`True`
`>=`	ist größer oder gleich	`3 >= 7`	`False`

`!=` liest man „ungleich“. Bei `<=` und `>=` steht das Gleichheitszeichen immer **rechts**.

```python
a = 7
b = 3
print(a > b)
print(a == b)
print(a != b)
print(a <= 7)
```

```
True
False
True
True
```

Ergebnisse von Vergleichen speichern

Ein Wahrheitswert ist ein ganz normaler Wert und kann in einer Variablen gespeichert werden.

```python
alter = int(input("Wie alt bist du? "))
ist_volljaehrig = alter >= 18
print("Volljährig:", ist_volljaehrig)
print(type(ist_volljaehrig))
```

```
Wie alt bist du? 15
Volljährig: False
<class 'bool'>
```

- **Zeile 2:** Zuerst wird die rechte Seite ausgewertet: Ist 15 ≥ 18? Nein, also `False`. Dann wird der Wert gespeichert.

Für solche Variablen wählt man Namen wie eine Ja-Nein-Frage: `ist_volljaehrig`, `hat_ausweis`, `ist_gerade`.

Texte vergleichen

Es kommt auf jedes Zeichen an, auch auf Groß- und Kleinschreibung und Leerzeichen.

```python
print("ja" == "ja", "Ja" == "ja", "ja " == "ja")
```

```
True False False
```

⚠️ Stolperstein: Eingabe nicht umgewandelt

🔮 **Vorhersage:** Gib `7` ein. Was gibt das Programm aus?

```python
zahl = input("Gib eine Zahl ein: ")
print(zahl == 7)
```

```
Gib eine Zahl ein: 15
False
```

Keine Fehlermeldung, aber immer `False`: Der Text `"7"` ist nicht die Zahl `7`. Abhilfe: `zahl = int(input(...))`.

⚠️ Zuweisung `=` oder Vergleich `==`?

- `x = 5` ist eine **Zuweisung**: „`x` bekommt den Wert 5." Es wird nichts geprüft.
- `x == 5` ist ein **Vergleich**: „Ist `x` gleich 5?" Es wird nichts verändert.

Führe die Zelle aus und lies die Fehlermeldung.

```
print(7 = 7)
```

```
  File "<Zelle 12>", line 1
    print(7 = 7)
          ^^^
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
```

`SyntaxError: expression cannot contain assignment, perhaps you meant "=="?` heißt: „Ein Ausdruck darf keine Zuweisung enthalten. Meintest du `==`?“

> 📌 **Merksatz:** Vergleichsoperatoren liefern immer einen Wahrheitswert (`True` oder `False`): `==`, `!=`, `<`, `>`, `<=`, `>=`. Verwechsle nicht `=` (Zuweisung) und `==` (Vergleich)!

3 Logische Operatoren: `and`, `or` und `not`

Ein Film ist ab 12, und an der Kasse wird ein Ausweis verlangt. Du darfst nur hinein, wenn **beide** Bedingungen erfüllt sind.

- `and` („und“): nur `True`, wenn **beide** Seiten `True` sind.
- `or` („oder“): `True`, wenn **mindestens eine** Seite `True` ist, also **auch, wenn beide** wahr sind. Anders als „entweder … oder“ im Alltag!
- `not` („nicht“): steht **vor** einem Wahrheitswert und kehrt ihn um.

```
1 print(True and False)
2 print(True or False)
3 print(not True)
4 print(not 7 > 3)
```

```
False
True
False
False
```

- **Zeile 4:** Zuerst der Vergleich: `7 > 3` ist `True`. Dann kehrt `not` das um.

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

> 📌 **Merksatz:** `and`: beide Bedingungen müssen wahr sein. `or`: mindestens eine muss wahr sein. `not`: kehrt einen Wahrheitswert um.

Vergleiche verknüpfen

Ist eine Person jugendlich, also mindestens 13 und jünger als 18?

```
1 alter = int(input("Wie alt bist du? "))
2 ist_jugendlich = alter >= 13 and alter < 18
3 print("Jugendlich:", ist_jugendlich)
```

```
Wie alt bist du? 15
Jugendlich: True
```

Python wertet Zeile 2 so aus:

1. `15 >= 13` ergibt `True`.
2. `15 < 18` ergibt `True`.
3. `True and True` ergibt `True`.

Rechnungen und Vergleiche kommen **vor** den logischen Operatoren. Unter ihnen gilt: erst `not`, dann `and`, zuletzt `or`.

⚠️ Stolperstein: halber Vergleich

Im Deutschen sagt man „älter als 12 und jünger als 18“. Python braucht auf **beiden** Seiten von `and` einen vollständigen Vergleich.

```
alter = 15
print(alter > 12 and < 18)
```

```
  File "<Zelle 15>", line 2
    print(alter > 12 and < 18)
                         ^
SyntaxError: invalid syntax
```

Richtig ist `alter > 12 and alter < 18`.

🔮 Übung 2 – Wahr oder falsch?

Es gilt `x = 5` und `y = 10`. Bestimme **zuerst im Kopf**, ob jeder Ausdruck `True` oder `False` ergibt.

Ausdruck	Ergebnis
`x < y`	
`x == 5 and y == 5`	
`x == 5 or y == 5`	
`not x > y`	
`x % 2 == 0`	
`x + y >= 15 and not y == 10`	

```
x = 5
y = 10
print(x < y)
# TODO: die übrigen fünf Ausdrücke prüfen
```

```
True
```

4 Klammern entscheiden: Darf ich zocken?

- `A` = Die Hausaufgaben sind fertig.
- `B` = Das Zimmer ist aufgeräumt.
- `C` = Es ist Wochenende.

Zwei Eltern formulieren ihre Regel unterschiedlich:

- **Regel 1:** `(A and B) or C`: Hausaufgaben **und** Zimmer, **oder** Wochenende. Am Wochenende reicht `C` allein.
- **Regel 2:** `A and (B or C)`: Hausaufgaben **immer**, dazu Zimmer **oder** Wochenende.

🔮 **Vorhersage:** Hausaufgaben nicht fertig, Zimmer nicht aufgeräumt, aber Wochenende. Wer darf zocken?

```python
A = False   # Hausaufgaben sind fertig
B = False   # Zimmer ist aufgeräumt
C = True    # Es ist Wochenende

ausdruck_1 = (A and B) or C
ausdruck_2 = A and (B or C)

print("(A and B) or C =", ausdruck_1)
print("A and (B or C) =", ausdruck_2)
```

```
(A and B) or C = True
A and (B or C) = False
```

- **Zeile 5:** Erst die Klammer: `False and False` ergibt `False`. Dann `False or True`, also `True`.
- **Zeile 6:** Erst die Klammer: `False or True` ergibt `True`. Dann `False and True`, also `False`.

Die Tabelle zeigt alle acht Möglichkeiten:

A	B	C	A and B	B or C	(A and B) or C	A and (B or C)
False	False	False	False	False	False	False
False	False	True	False	True	True	False
False	True	False	False	True	False	False
False	True	True	False	True	True	False
True	False	False	False	False	False	False
True	False	True	False	True	True	True
True	True	False	True	True	True	True
True	True	True	True	True	True	True

In vier Zeilen unterscheiden sich die beiden letzten Spalten: Die Ausdrücke sind **nicht äquivalent**.

> 📌 **Merksatz:** Klammern bestimmen, welcher Teilausdruck zuerst ausgewertet wird. Setze immer Klammern, wenn du `and` und `or` mischst.

Bedingungen verneinen: die Regeln von De Morgan

Ist eine Person **nicht** jugendlich?

```python
alter = 20
print(not (alter >= 13 and alter < 18))
print(alter < 13 or alter >= 18)
```

```
True
True
```

Beide Zeilen liefern für **jedes** Alter dasselbe. Probiere es mit `alter = 15` und `alter = 9`. Beim Umformen wurde jeder Vergleich verneint (`>=` wird `<`, `<` wird `>=`), und aus `and` wurde `or`.

> 📌 **Regeln von De Morgan:** `not (A and B)` ist dasselbe wie `not A or not B`. `not (A or B)` ist dasselbe wie `not A and not B`. Verneint man eine verknüpfte Bedingung, wird **jede** Teilbedingung verneint, und aus `and` wird `or` (und umgekehrt).

Vergleich	verneint
`a == b`	`a != b`
`a < b`	`a >= b`
`a > b`	`a <= b`

> ⚠️ **Stolperstein:** `alter < 13 and alter >= 18` ist **immer** `False`, denn niemand ist gleichzeitig jünger als 13 und mindestens 18. Beim Verneinen wird aus `and` ein `or`!

✏️ Übung 3 – Bedingungen verneinen

Schreibe zu jeder Bedingung eine Bedingung **ohne** `not`, die das Gegenteil prüft. Teste in der Codezelle: Beide Werte einer Zeile müssen gleich sein.

1. `zahl >= 1 and zahl <= 6` (eine Würfelzahl)
2. `note == 1 or note == 2`
3. `not (temperatur > 30 or regen)` (`regen` ist ein Wahrheitswert, „nicht `regen`“ darf als `not regen` stehen bleiben)

Lückencode

```
zahl = 7
print(not (zahl >= 1 and zahl <= 6), ___)
note = 2
print(not (note == 1 or note == 2), ___)
temperatur = 25
regen = False
print(not (temperatur > 30 or regen), ___)
```

📝 Übungen: Schritt für Schritt selbstständig

Das **Musterbeispiel** ist ganz vorgemacht. Bei Übung 4 füllst du Lücken `___`, bei Übung 5 steht nur noch ein Gerüst aus Kommentaren, die Zusatzaufgabe schreibst du allein.

Schritt	Frage
1	**Eingabe:** Was wird eingelesen? Als welcher Datentyp?
2	**Verarbeitung:** Welcher Operator beantwortet die Frage?
3	**Ausgabe:** Was soll auf dem Bildschirm stehen?

🧩 Musterbeispiel (vorgemacht) – Durch 3 teilbar?

Aufgabe: Das Programm liest eine Ganzzahl ein und gibt als Wahrheitswert aus, ob sie durch 3 teilbar ist.

1. **Eingabe:** eine ganze Zahl, also `int(input(...))`.
2. **Verarbeitung:** „durch 3 teilbar“ heißt „Rest beim Teilen durch 3 ist 0“: `zahl % 3 == 0`. Python rechnet erst `%`, dann vergleicht es mit `==`.
3. **Ausgabe:** Text und Wahrheitswert, mit Komma getrennt.

```python
# Schritt 1: Eingabe
zahl = int(input("Ganzzahl eingeben: "))

# Schritt 2: Verarbeitung
ist_teilbar = zahl % 3 == 0

# Schritt 3: Ausgabe
print("Durch 3 teilbar:", ist_teilbar)
```

```
Ganzzahl eingeben: 15
Durch 3 teilbar: True
```

Teste mit 12 (`True`), 13 (`False`) und 0 (`True`).

📝 Übung 4 – Gerade oder ungerade? (Lücken füllen)

Das Programm liest eine Ganzzahl ein und gibt als Wahrheitswert aus, ob sie gerade ist. Ersetze jedes `___`.
Kontrolle: 14 → `Ist gerade: True`, 7 → `Ist gerade: False`.

Lückencode

```python
zahl = ___(input("Ganzzahl eingeben: "))
ist_gerade = zahl ___ 2 ___ 0
print("Ist gerade:", ist_gerade)
```

📝 Übung 5 – Schaltjahr-Check (mit Gerüst)

Ein Jahr ist ein Schaltjahr, wenn es durch 4 teilbar ist, aber nicht durch 100, **außer** es ist auch durch 400 teilbar. Du brauchst nur Operatoren. Ersetze die Kommentare durch Programmzeilen.

Kontrolle: 2024 → `True`, 2023 → `False`, 1900 → `False`, 2000 → `True`.

```python
# Jahreszahl als Ganzzahl einlesen
# A = durch 4 teilbar?
# B = durch 100 teilbar?
# C = durch 400 teilbar?
# schaltjahr = durch 4 und nicht durch 100, oder durch 400 (Klammern setzen!)
# alle Zwischenergebnisse und das Ergebnis ausgeben
```

🚀 Zusatzaufgabe – Sekunden umrechnen (selbstständig)

Lies eine Anzahl von Sekunden ein und gib aus, wie viele volle Stunden, Minuten und Sekunden das sind.
Kontrolle: 3725 Sekunden sind 1 Stunden, 2 Minuten und 5 Sekunden. Tipp: Eine Stunde hat 3600 Sekunden.

📌 Das Wichtigste in Kürze

- `//` Ganzzahldivision, `%` Rest, `**` Potenz. `^` ist **nicht** „hoch“.
- `a % b == 0` heißt: `a` ist durch `b` teilbar.
- Vergleiche `==`, `!=`, `<`, `>`, `<=`, `>=` liefern `True` oder `False`. `=` weist zu, `==` vergleicht.
- `and`: beide wahr. `or`: mindestens eins wahr. `not`: umkehren. Beim Mischen Klammern setzen.
- De Morgan: Beim Verneinen jede Teilbedingung verneinen und `and` ↔ `or` tauschen.

Zum Weiterüben: `DS02_Weitere_Uebungen.ipynb`.