DS 1 – Einstieg: Variablen, Eingabe und Ausgabe

Informatik 10 – MINT-Zweig | Textuelle Programmierung und Datenstrukturen

Lernziele

Ich kann

- mit `print()` Texte und Zahlen auf dem Bildschirm ausgeben,
- Variablen anlegen, ihnen Werte zuweisen und mit ihnen rechnen,
- die Datentypen `int`, `float`, `str` und `bool` unterscheiden,
- mit `input()` Eingaben einlesen und mit `int()` und `float()` in Zahlen umwandeln,
- eine Fehlermeldung lesen und die Ursache des Fehlers finden.

🗣️ Sprachhilfe

- *„alter ist 15."* → „Der Variablen `alter` wird der Wert 15 zugewiesen."
- *„Das Programm fragt nach dem Namen."* → „Die Eingabeanweisung liest den Namen ein und speichert ihn als Zeichenkette in der Variablen `name`."
- *„Da kommt 14 raus."* → „Zur Eingabe 7 gibt das Programm den Wert 14 aus."

Worum es geht

Ein Programm ist eine Liste von Anweisungen, die der Computer der Reihe nach ausführt, von oben nach unten. In der Blockprogrammierung hast du Bausteine zusammengeschoben, in Python schreibst du Text. Heute lernst du die drei wichtigsten Dinge, die fast jedes Programm braucht: Es **gibt etwas aus**, es **merkt sich Werte** und es **liest Eingaben** ein.

1 Das erste Programm: Ausgaben mit `print`

`print` heißt „drucken", gemeint ist: auf dem Bildschirm anzeigen. Was angezeigt werden soll, steht in den **runden Klammern**, ein Text in **Anführungszeichen**. Python führt die Zeilen **nacheinander von oben nach unten** aus.

```
1 print("Hallo Welt!")
2 print("Guten Morgen!")
3 print("Heute lernen wir Python.")
```

```
Hallo Welt!
Guten Morgen!
Heute lernen wir Python.
```

🧪 **Probiere es aus:** Ändere die Texte. Was passiert, wenn du die letzte Zeile an den Anfang verschiebst?

Text oder Zahl?

🔮 **Vorhersage:** Was geben die beiden ersten Zeilen aus?

```python
print(3 + 4)
print("3 + 4")
print("Ich bin", 15, "Jahre alt.")
print("In fünf Jahren bin ich", 15 + 5, "Jahre alt.")
```

```
7
3 + 4
Ich bin 15 Jahre alt.
In fünf Jahren bin ich 20 Jahre alt.
```

- **Zeile 1:** Ohne Anführungszeichen ist `3 + 4` eine Rechnung, Python gibt `7` aus.
- **Zeile 2:** In Anführungszeichen ist es ein Text, der genau so ausgegeben wird.
- **Zeilen 3–4:** Mehrere Dinge trennt man durch **Kommas**. Python setzt je ein Leerzeichen dazwischen.

> 📌 **Merksatz:** Mit `print(...)` gibt man Werte auf dem Bildschirm aus. Texte stehen in Anführungszeichen, Zahlen und Rechnungen ohne. Mehrere Werte trennt man durch Kommas.

⚠️ Wenn etwas schiefgeht

Beim Abtippen passieren Fehler, auch Profis jeden Tag. Die Fehlermeldung ist ein Hinweis, wo das Problem liegt. Führe beide Zellen aus.

```python
print("Hallo)
```

```
  File "<Zelle 3>", line 1
    print("Hallo)
          ^
SyntaxError: unterminated string literal (detected at line 1)
```

```python
print(Hallo)
```

```
Traceback (most recent call last):
  File "<Zelle 4>", line 1, in <module>
NameError: name 'Hallo' is not defined
```

1. `SyntaxError: unterminated string literal (detected at line 1)`: Die Schreibweise stimmt nicht, der Text ist nicht abgeschlossen. Python nennt sogar die Zeile.
2. `NameError: name 'Hallo' is not defined`: Ohne Anführungszeichen hält Python `Hallo` für einen **Namen**, und den gibt es nicht.

Häufige Tippfehler am Anfang: vergessenes Anführungszeichen oder Klammer, `Print` statt `print` (Groß- und Kleinschreibung zählt!) und deutsche Anführungszeichen „…“ statt `"…"`. Schau bei einer Fehlermeldung zuerst die genannte Zeile an und die Zeile davor.

2 Variablen: Werte speichern

Programme müssen sich Dinge merken: einen Namen, einen Punktestand, einen Preis. Stell dir eine **Variable** wie eine **beschriftete Schachtel** vor: Außen steht der Name, z. B. `alter`, innen liegt ein Wert, z. B. `15`.

> 📌 **Definition:** Eine **Variable** ist ein benannter Speicherplatz im Arbeitsspeicher, in dem ein Wert abgelegt wird. Über ihren Namen kann das Programm den Wert lesen und verändern.

```python
alter = 15
print(alter)
print("Ich bin", alter, "Jahre alt.")
```

```
15
Ich bin 15 Jahre alt.
```

- **Zeile 1:** `alter` bekommt den Wert `15`. Auf dem Bildschirm passiert nichts.
- **Zeile 2:** `alter` steht ohne Anführungszeichen, Python schaut in die Schachtel und gibt `15` aus.

Das Gleichheitszeichen ist ein Zuweisungszeichen

`=` heißt in Python **nicht** „ist gleich“, sondern **„bekommt den Wert“**. Zuerst wird die rechte Seite ausgerechnet, dann kommt das Ergebnis in die Variable links. Darum ist `alter = alter + 1` sinnvoll, obwohl es in Mathe falsch wäre.

```python
alter = 15
print(alter)
alter = alter + 1
print(alter)
```

```
15
16
```

Das **Ablaufprotokoll** zeigt nach jeder Zeile den Wert der Variablen:

Schritt	Zeile	ausgeführte Anweisung	`alter` danach
1	1	`alter = 15`	`15`
2	2	`print(alter)`	`15`
3	3	`alter = alter + 1`	`16`
4	4	`print(alter)`	`16`

Der alte Wert `15` ist nach Zeile 3 weg, er wurde **überschrieben**.

> 📌 **Merksatz:** Mit dem **Zuweisungsoperator** `=` wird einer Variablen ein Wert zugewiesen, z. B. `alter = 15`. Zuerst wird die rechte Seite ausgewertet, dann wird das Ergebnis in der Variablen links gespeichert.

Gute Namen für Variablen

- Erlaubt sind Buchstaben, Ziffern und `_`. Ein Name darf **nicht mit einer Ziffer beginnen** und **keine Leerzeichen** enthalten.
- `alter` und `Alter` sind zwei verschiedene Namen.
- Wähle **sprechende Namen**: `preis` statt `p`, `anzahl_artikel` statt `aa`. Verzichte auf Umlaute: `groesse` statt `größe`.

> ⚠️ **Stolperstein:** Vertippst du dich beim Namen, meldet Python `NameError: name 'Alter' is not defined. Did you mean: 'alter'?`

🔮 Übung 1 – Vorhersage

Lies das Programm, **ohne** es auszuführen. Schreibe auf, was es ausgibt. Überprüfe danach.

```python
punkte = 10
print(punkte)
punkte = punkte + 5
punkte = punkte * 2
print("Endstand:", punkte)
```

```
10
Endstand: 30
```

Meine Vorhersage:

...

3 Datentypen: Zahlen, Texte und Wahrheitswerte

Die Art eines Wertes heißt **Datentyp**. Davon hängt ab, was man mit dem Wert machen kann: Mit Zahlen kann man rechnen, mit Texten nicht.

Datentyp	Bedeutung	Beispiele
`int`	ganze Zahl (*integer*)	`15`, `-3`, `0`, `2026`
`float`	Kommazahl (*floating point number*)	`1.68`, `-0.5`, `3.0`
`str`	Zeichenkette, also Text (*string*)	`"Alex"`, `"Hallo Welt!"`, `"15"`
`bool`	Wahrheitswert (nach George Boole)	`True`, `False`

- Kommazahlen schreibt man mit **Punkt**: `1.68`, nicht `1,68`.
- Alles in Anführungszeichen ist Text, auch `"15"`.
- `True` und `False` schreibt man groß und ohne Anführungszeichen.

Mit `type(...)` fragst du Python nach dem Datentyp:

```python
name = "Alex"
alter = 15
groesse = 1.68
volljaehrig = False
print(type(name))
print(type(alter))
print(type(groesse))
print(type(volljaehrig))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
```

> 📌 **Merksatz:** Python unterscheidet u. a. die Datentypen `int` (Ganzzahl), `float` (Kommazahl), `str` (Zeichenkette) und `bool` (Wahrheitswert).

Rechnen mit Zahlen

`+`, `-`, `*` (mal) und `/` (geteilt). Es gilt **Punkt vor Strich**, Klammern zuerst. Das Ergebnis von `/` ist **immer** ein `float`, auch wenn die Division aufgeht.

```python
a = 7
b = 2
print(a + b)
print(a - b)
print(a * b)
print(a / b)
print(2 + 3 * 4)
print((2 + 3) * 4)
```

```
9
5
14
3.5
14
20
```

⚠️ Stolperstein: Komma statt Punkt

🔮 **Vorhersage:** Was gibt die Zelle aus?

```python
preis = 2,50
print(preis)
```

```
(2, 50)
```

Keine Fehlermeldung, aber Python hat zwei Zahlen gespeichert, `2` und `50`. Richtig ist `preis = 2.50`.

✏️ Übung 2 – Datentypen bestimmen

Gib zu jedem Wert den Datentyp an. Überprüfe danach mit `type(...)` in der Codezelle.

Wert	42	"42"	4.2	True	"True"	-7	7 / 7	"Hallo"
Datentyp								

```python
print(type(42))
# TODO: die übrigen sieben Werte prüfen
```

```
<class 'int'>
```

4 Eingaben mit `input`

Mit `input(...)` kann die Nutzerin oder der Nutzer etwas eingeben. `input` zeigt zuerst den Text in den Klammern an (die **Eingabeaufforderung**) und **wartet**, bis jemand etwas eintippt und die Eingabetaste drückt.

```python
name = input("Wie heißt du? ")
print("Hallo", name)
```

```
Wie heißt du? Alex
Hallo Alex
```

> 💡 **Tipp:** Setze ans Ende der Eingabeaufforderung ein Leerzeichen, sonst klebt die Eingabe am Fragezeichen.

Die Eingabe ist immer ein Text

🔮 **Vorhersage:** Gib `7` ein. Was erwartest du?

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

```
Gib eine Zahl ein: 15
1515
<class 'str'>
```

Python rechnet nicht `7 + 7 = 14`, sondern gibt `77` aus. Was jemand eintippt, ist für Python **immer eine Zeichenkette**, und `+` hängt zwei Zeichenketten aneinander: `"7" + "7"` ergibt `"77"`.

> 📌 **Merksatz:** Mit `input(...)` liest man eine Eingabe ein. **Die Eingabe ist immer vom Typ `str`!**

5 Typumwandlung: aus Text wird eine Zahl

Befehl	wandelt um in	Beispiel	Ergebnis
`int(...)`	ganze Zahl	`int("7")`	`7`
`float(...)`	Kommazahl	`float("2.5")`	`2.5`
`str(...)`	Zeichenkette	`str(15)`	`"15"`

```python
lieblingszahl = input("Gib deine Lieblingszahl ein: ")
print("Deine Eingabe war:", lieblingszahl, "- Typ:", type(lieblingszahl))

lieblingszahl_int = int(lieblingszahl)
print("Deine Zahl mal 2 ist:", lieblingszahl_int * 2)
```

```
Gib deine Lieblingszahl ein: 15
Deine Eingabe war: 15 - Typ: <class 'str'>
Deine Zahl mal 2 ist: 30
```

Eingabe und Umwandlung in einer Zeile

Meistens schreibt man beides in **eine** Zeile. Lies sie **von innen nach außen**: Zuerst liefert `input(...)` den Text `"15"`, dann macht `int(...)` daraus die Zahl `15`. Am Ende stehen **zwei** schließende Klammern. Für Kommazahlen nimmst du `float`.

```python
alter = int(input("Wie alt bist du? "))
print("Nächstes Jahr bist du", alter + 1)

groesse = float(input("Wie groß bist du in Metern? "))
print("In Zentimetern sind das", groesse * 100)
```

```
Wie alt bist du? 15
Nächstes Jahr bist du 16
Wie groß bist du in Metern? 1.72
In Zentimetern sind das 172.0
```

> 📌 **Merksatz:** Soll eine Eingabe als Zahl weiterverarbeitet werden, muss sie umgewandelt werden: `int(...)` bzw. `float(...)` (**Typumwandlung**).

> ⚠️ **Stolperstein:** Gibt jemand `sieben` ein, meldet `int(...)` einen `ValueError: invalid literal for int() with base 10: 'sieben'`. Ebenso bei `int("2.5")` und `float("2,5")`. Schau dann nach, **was eingegeben wurde**.

6 Texte zusammensetzen

Mit Kommas setzt `print` immer ein Leerzeichen dazwischen. Soll direkt nach einem Namen ein Komma oder Punkt folgen, hängst du Texte mit `+` **aneinander** (verketten). Leerzeichen musst du dann selbst in die Anführungszeichen schreiben.

```
name = "Alex"
print("Hallo, " + name + "!")
print("Hallo,", name, "!")
```

```
Hallo, Alex!
Hallo, Alex !
```

⚠️ Stolperstein: Text plus Zahl

Mit `+` lassen sich nur **Texte mit Texten** verbinden. Führe die Zelle aus.

```
alter = 15
print("Du bist " + alter + " Jahre alt.")
```

```
Traceback (most recent call last):
  File "<Zelle 17>", line 2, in <module>
TypeError: can only concatenate str (not "int") to str
```

Lies die Fehlermeldung **von unten nach oben**:

1. **Letzte Zeile:** `TypeError: can only concatenate str (not "int") to str`, also „kann nur `str` (nicht `int`) an `str` anhängen“.
2. **Darüber:** die Zeile, in der der Fehler passiert ist, und Zeichen `^`, die auf die Stelle zeigen.

Richtig ist `print("Du bist " + str(alter) + " Jahre alt.")`.

Fehler	Bedeutung	typische Ursache
`SyntaxError`	Schreibweise falsch	Anführungszeichen oder Klammer fehlt
`NameError`	Name unbekannt	Tippfehler, Variable noch nicht angelegt, Anführungszeichen vergessen
`TypeError`	Datentyp passt nicht	Zahl und Text mit `+` verbunden
`ValueError`	Wert passt nicht	`int("sieben")`, `float("2,5")`

📝 Übungen: Schritt für Schritt selbstständig

Die Übungen werden schrittweise schwieriger. Das **Musterbeispiel** ist ganz vorgemacht. Übung 3 ist eine Fehlersuche, bei Übung 4 füllst du Lücken `___` aus, bei Übung 5 steht nur noch ein Gerüst aus Kommentaren, die Zusatzaufgabe schreibst du allein. Gehe jedes Mal in drei Schritten vor:

Schritt	Frage
1	**Eingabe:** Was muss das Programm abfragen? Text oder Zahl (`int` / `float`)?
2	**Verarbeitung:** Was muss gerechnet werden?
3	**Ausgabe:** Was soll auf dem Bildschirm stehen?

🧩 Musterbeispiel (vorgemacht) – Kinobesuch

Aufgabe: Das Programm fragt nach dem Namen und nach der Anzahl der Kinokarten. Eine Karte kostet 9.50 €. Es gibt den Gesamtpreis in einem Satz aus.

1. **Eingabe:** Der Name ist ein Text, also nur `input(...)`. Die Anzahl ist eine ganze Zahl, also `int(input(...))`.

2. **Verarbeitung:** `gesamt = anzahl * 9.50`.
3. **Ausgabe:** Der Name soll direkt vor einem Komma stehen, deshalb `+`. Die Zahl `gesamt` muss dafür mit `str(...)` umgewandelt werden.

Lies den Code Zeile für Zeile und führe ihn aus, z. B. mit `Alex` und `3`.

```python
# Schritt 1: Eingabe
name = input("Wie heißt du? ")
anzahl = int(input("Wie viele Karten? "))

# Schritt 2: Verarbeitung
gesamt = anzahl * 9.50

# Schritt 3: Ausgabe
print("Danke, " + name + "! Das macht " + str(gesamt) + " Euro.")
```

```
Wie heißt du? Alex
Wie viele Karten? 4
Danke, Alex! Das macht 38.0 Euro.
```

🐞 Übung 3 – Fehler finden

Das Programm soll nach zwei Zahlen fragen und ihre Summe ausgeben. Es enthält **drei Fehler**. Finde sie, erkläre jeweils, welche Fehlermeldung oder welches falsche Verhalten sie auslösen, und schreibe das Programm in der zweiten Zelle richtig auf. Tipp: Ein Fehler erzeugt keine Fehlermeldung, sondern ein falsches Ergebnis.

```python
# Fehlerhafter Code – erst lesen, dann korrigieren
# zahl1 = input("Erste Zahl: )
# zahl2 = input("Zweite Zahl: ")
# print("Die Summe ist", Zahl1 + zahl2)
```

Die drei Fehler:

…

📝 Übung 4 – Steckbrief (Lücken füllen)

Das Programm fragt Name, Alter (als Zahl!) und Lieblingsfach ab und gibt einen vollständigen Satz aus. In der letzten Zeile steht das **doppelte Alter**: Daran siehst du, ob das Alter wirklich als Zahl gespeichert ist. Ersetze jedes `___`.

```
Wie heißt du? Alex
Wie alt bist du? 15
Was ist dein Lieblingsfach? Informatik
Hallo, ich bin Alex, 15 Jahre alt und mein Lieblingsfach ist Informatik.
Doppeltes Alter: 30
```

Lückencode

```python
# Schritt 1: Eingabe
name = input("Wie heißt du? ")
alter = ___(input("Wie alt bist du? "))
fach = input("Was ist dein Lieblingsfach? ")

# Schritt 3: Ausgabe
print("Hallo, ich bin " + name + ", " + ___(alter)
      + " Jahre alt und mein Lieblingsfach ist " + ___ + ".")
print("Doppeltes Alter:", ___)
```

📝 Übung 5 – Einfacher Rechner (mit Gerüst)

Das Programm liest zwei Zahlen ein und gibt **Summe, Differenz und Produkt** aus. Ersetze die Kommentare durch Programmzeilen. Kontrolle: 7.5 und 2 ergeben 9.5, 5.5 und 15.0.

```
# Schritt 1: erste Zahl einlesen und in eine Kommazahl umwandeln
# Schritt 1: zweite Zahl einlesen und umwandeln
# Schritt 2 und 3: Summe ausgeben
#                  Differenz ausgeben
#                  Produkt ausgeben
```

🚀 Zusatzaufgabe – Gesamtpreis (selbstständig)

Frage einen Preis in Euro (als Kommazahl) und eine Anzahl an Artikeln (als ganze Zahl) ab und berechne den Gesamtpreis. Runde die Ausgabe auf zwei Stellen. Kontrolle: 1.99 € und 3 Artikel ergeben 5.97 Euro.

> 🧰 **Werkzeug:** `round(wert, stellen)` rundet auf eine bestimmte Anzahl Nachkommastellen: `round(3.14159, 2)` ergibt `3.14`.

📌 Das Wichtigste in Kürze

- `print(...)` gibt Werte aus. Texte stehen in Anführungszeichen, mehrere Werte trennt man durch Kommas.
- Eine **Variable** ist ein benannter Speicherplatz. `=` heißt „bekommt den Wert“: erst rechts rechnen, dann links speichern.
- Datentypen: `int`, `float` (mit Punkt!), `str`, `bool`. `/` liefert immer einen `float`.
- `input(...)` liefert **immer** einen Text. Zum Rechnen umwandeln mit `int(...)` oder `float(...)`.
- Texte verkettet man mit `+`, Zahlen vorher mit `str(...)` umwandeln.
- Fehlermeldungen von unten nach oben lesen: `SyntaxError`, `NameError`, `TypeError`, `ValueError`.

Zum Weiterüben: `DS01_Weitere_Uebungen.ipynb`.