Programmieren – 6. Funktionen 1

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Warm-up: Was gibt der Code aus?

Aufschreiben (von Hand verfolgen!), dann ausführen:

```
i = 0
while i < 4:
    i += 1
    print(i, end=" ")
```

Heute lernen Sie

- **Eigene Funktionen** definieren: `def`
- Werte **zurückgeben**: `return` – und warum das etwas ganz anderes ist als `print`
- Fertige Funktionen aus Modulen holen: `import math`

Sie benutzen längst Funktionen

```
print("Hallo")        # tut etwas: gibt aus
len("Hallo")          # liefert etwas: 5
input("Name: ")       # liefert die Eingabe
int("42")             # liefert die Zahl 42
range(5)              # liefert die Zahlen 0 bis 4
```

Seit Woche 1: Werte rein (in die Klammern), Ergebnis raus – oder Wirkung.

Heute bauen Sie zum ersten Mal **eigene**.

Die erste eigene Funktion

Was stört an diesem Skript?

```
c1 = 23.5
f1 = c1 * 9 / 5 + 32
print(f"{c1} °C = {f1} °F")

c2 = -10.0
f2 = c2 * 9 / 5 + 32
print(f"{c2} °C = {f2} °F")
```

```
c3 = 100.0
f3 = c3 * 9 / 5 + 32
print(f"{c3} °C = {f3} °F")
```

Dieselbe Rechnung – einmal aufgeschrieben

```
def fahrenheit(celsius):
    return celsius * 9 / 5 + 32

print(fahrenheit(23.5))
print(fahrenheit(-10.0))
print(fahrenheit(100.0))
```

- `def` **definiert** die Funktion: Name, ein **Parameter** in Klammern, Doppelpunkt, eingerückter Block
- `return` **gibt das Ergebnis zurück** – es landet an der Stelle des Aufrufs
- Die Rechnung steht **genau einmal** – eine Korrektur wirkt überall

Wichtig: Definieren führt nichts aus

```
def fahrenheit(celsius):
    return celsius * 9 / 5 + 32
```

Diese Zelle ausführen – und es passiert: **nichts Sichtbares.**

- `def` sagt Python nur: *merk dir dieses Rezept unter diesem Namen*
- Gekocht wird erst beim **Aufruf**: `fahrenheit(20)`
- Jeder Aufruf führt den Block neu aus – mit dem übergebenen Wert als `celsius`

Mini-Aufgabe (3 min)

1. Schreiben Sie eine Funktion `quadrat(x)`, die x^2 zurückgibt (`return`)
2. Testen Sie sie: `print(quadrat(5))` und `print(quadrat(1.5))`

Zusatz für Schnelle: `kreisflaeche(radius)` – Fläche zurückgeben (`return`), mit `3.14159` als Kreiszahl. Gleich lernen wir es genauer!

Achtung typischer Fehler

```
def begruessung():
    print("Willkommen zur Messung!")
```

„Mein Programm macht nichts!“ – Richtig, denn:

- Die Funktion wurde **definiert**, aber nie **aufgerufen**
- Es fehlt die Zeile: `begruessung()`
- Definition ohne Aufruf ist ein Rezept, das nie gekocht wird

`print` vs. `return`

Vorhersage-Aufgabe (3 min)

Zwei fast gleiche Funktionen – was gibt der Code aus? Aufschreiben!

```
def f1(x):
    print(x * 2)

def f2(x):
    return x * 2

a = f1(3)
b = f2(3)
print(a)
print(b)
```

Ausgabe ist nicht Rückgabe

- `print` = **Ausgabe**: zeigt einen Wert dem *Menschen* – gibt aber nichts zurück
- `return` = **Rückgabe**: übergibt einen Wert dem *Programm* – zum Weiterrechnen
- Eine Funktion ohne `return` gibt automatisch `None` zurück („nichts“)

```
print(f2(3) + 1)   # 7 – mit Rückgabe kann man rechnen
print(f1(3) + 1)   # TypeError – mit None kann man nicht rechnen
```

Die Verwechslung von `print` und `return` ist **der** Klassiker dieses Kurses – heute ausmerzen!

Mini-Aufgabe (2 min)

```
def doppelt(x):
    print(x * 2)

ergebnis = doppelt(7)
print(ergebnis)
```

1. Führen Sie den Code aus – was fällt auf?
2. Ersetzen Sie `print` in der Funktion durch `return` und führen Sie erneut aus

Debug-Aufgabe (3 min)

Dieses Programm soll den Nettopreis ausgeben (`print`) – stattdessen erscheint `None` . Finden Sie den Fehler:

```
def netto(brutto):
    ergebnis = brutto / 1.19

preis = netto(119)
print(f"Nettopreis: {preis}")
```

Funktionen aus Modulen

`import math`

Viele Funktionen sind fertig eingebaut – man muss sie nur holen:

```
import math

print(math.sqrt(2))
print(math.pi)
```

Oder gezielt einzelne Namen importieren:

```
from math import sin

print(sin(0.5))
```

- Beide Formen begegnen Ihnen ständig in echtem Code
- Achtung: `sin` rechnet im **Bogenmaß**, nicht in Grad!

Transfer

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

Die Schwingungsdauer eines Pendels: $T = 2\pi\sqrt{L/g}$ mit $g = 9{,}81\ \mathrm{m/s^2}$

Schreiben Sie ein Programm:

1. Eine Funktion `schwingungsdauer(laenge)` , die T zurückgibt (`return`) – mit `math.pi` und `math.sqrt`
2. Pendellänge in m **einlesen**
3. Funktion **aufrufen**

4. Ergebnis **ausgeben** (`print`) mit zwei Nachkommastellen

Notieren Sie zuerst: Was gehört in die Funktion – und was ins Hauptprogramm?

Zusatz für Schnelle: `sinus_grad(winkel)` – nimmt Grad entgegen und gibt den Sinus zurück (`return`). Denken Sie ans Bogenmaß!

Gemeinsam live

Wir entwickeln die Lösung jetzt zusammen.

Mini-Check

Ohne Computer:

1. Was gibt eine Funktion ohne `return` zurück?
2. Ausgabe vs. Rückgabe – der Unterschied in einem Satz?
3. `def verdreifache(x): return 3 * x` – was gibt `print(verdreifache(4))` aus?
4. Eine Zelle mit nur einer Funktionsdefinition wird ausgeführt – warum ist keine Ausgabe zu sehen?

Bis nächste Woche!

Nächste Woche: **Listen** – viele Werte, eine Variable

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