Lektion 1b: Grundlagen der Programmierung in Python

1.1 Operatoren und Operanden

Damit Variablen miteinander agieren können, benötigen sie sog. Operatoren. Diese führen bestimmte Aktionen auf den sog. Operanden aus. Dies können Variablen oder sonstige Ausdrücke sein. Ein Ausdruck ist beinahe alles, was einen Wert hat. Variablen, Ziffern, Text sind Beispiele für Ausdrücke. In den nachfolgenden Abschnitten schauen wir uns die wichtigsten Typen von Operatoren an.

Überblick über einige wichtige Operatoren:

Rechen-,String- und Vergleichsoperatoren

Operator	Funktion
+ -	Vorzeichen
+ - * /	Grundrechenarten, liefern float Ergebnisse!
//	Ganzzahlige Division (20 // 6 ergibt 3)
%	Modulo = Rest der ganzzahligen Division (20%6 = 2)
**	Exponentialfunktion bzw. Hochstellen (2**3=8)
+=	Zuweisung und Addition x+=3 entspricht x=x+3 (geht auch mit -=,*=,/=)
< > <= >=	Kleiner, größer, kleiner gleich, größer gleich

Erste Rechenaufgaben

- 1+2
- 4*(3+2)
- 3-6
- 9/4 (Vorsicht: Geteilt ist kein : , sondern ein /)
- Rest berechnen von 11 geteilt durch 2 (mit Modulo-Operator %)
- 4 hoch 3 (Exponentialfunktionen mit **)

Zusatzaufgabe: Experimentieren Sie mit eigenen Rechenaufgaben!

1.2 Einschub: Wie Programmieren funktioniert

a) ***Algorithmus:*** Zunächst muss das Problem oder die Aufgabe in ein allgemeines Rezept übersetzt werden. Dieses soll die Schritte beschreiben, die der Computer ausführen muss, um das gewünschte Ergebnis zu erreichen. Das Rezept wird in der Fachsprache als Algorithmus bezeichnet. Die Erstellung des Rezeptes können Sie zunächst auch als Pseudocode vornehmen. Ein Pseudocode kann nicht von Computer gelesen werden, da er der höheren Programmiersprachen nur ähnelt und gemischt mit natürlicher Sprache und mathematischer Notation ist.

b) ***Programm schreiben:*** *A*ls nächstes übersetzen Sie Ihr Rezept in Anweisungen einer Programmiersprache. In dieser Stufe sprechen wir dann vom Programmieren. Das Ergebnis nennt man dann ein *Programm* oder *Quellcode*.

c) ***Programm ausführen:*** Nun übergeben Sie Ihren Quellcode an den Computer. Dieser beginnt Ihre Anweisungen auszuführen. Man spricht von *interpretieren*, *ausführen* oder *auswerten* des Codes.

1.3 Datentypen in Python

Datentyp	Bedeutung	Beispiel
`int`	Ganze Zahl	`x = 5`
`float`	Gleitkommazahl	`pi = 3.14`
`str`	Zeichenkette (Text)	`name = "Anna"`
`bool`	Wahrheitswert	`wahr = True, falsch = False`

Umwandlung von Datentypen

Manchmal müssen Werte von einem Typ in einen anderen umgewandelt werden.

```python
# int -> float
x = 5
print(float(x))  # 5.0

# float -> int (Nachkommastellen gehen verloren)
y = 3.9
print(int(y))    # 3

# str -> int / float
s1 = "42"
s2 = "3.14"
print(int(s1))        # 42
print(float(s2))      # 3.14

# int -> str
zahl = 7
text = str(zahl)
print(text + " Tage") # '7 Tage'
```

2 Übungen zu Datentypen und Umwandlungen

Schreiben Sie jeweils den passenden Code unter die Aufgabenstellung.

2.1 Aufgabe 1: Eingabe und Umwandlung

Fragen Sie den Benutzer nach seinem Alter (z. B. `"18"`) und wandeln Sie die Eingabe in eine ganze Zahl um.
Geben Sie anschließend eine passende Ausgabe aus, z. B. `"Sie sind 18 Jahre alt."`.

```python
# Ihr Code hier
```

2.2 Aufgabe 2: Zahl in Text umwandeln

Wandeln Sie die Zahl `5` in einen String um und fügen Sie `" Euro"` an.

```python
# Ihr Code hier
```

2.2.1 Lösung zu Aufgabe 1

```python
# Eingabe simulieren (in Jupyter kann input() deaktiviert sein)
eingabe = "18"  # normalerweise: input("Wie alt sind Sie? ")
alter = int(eingabe)
print("Sie sind", alter, "Jahre alt.")
```

2.2.2 Lösung zu Aufgabe 2

```python
betrag = 5
text = str(betrag) + " Euro"
print(text)
```