

---

# Programmieren mit Python

## Übungen (Teil I) (L+)

---

Version vom 5. Februar 2023

### Aufgabe 1.1

Ein Algorithmus ist eine Folge von Anweisungen zur Lösung einer Aufgabe mit folgenden Eigenschaften:

- eindeutig
- endlich
- korrekt

### Aufgabe 1.2

Algorithmisches Denken beschreibt die Art, Aufgaben so zu betrachten, dass sie von einem Computer gelöst werden können.

### Aufgabe 1.3

Ein Quellcode ist ein für Menschen lesbarer Text in einer Programmiersprache.

### Aufgabe 1.4

Ein Interpreter ist ein Programm, das einen Quellcode zeilenweise ausführt.

### Aufgabe 1.5

Ein (Python-)Modul ist ein Python-Quelltext.

### Aufgabe 1.6

- (a) Die Module `time` und `math`
- (b) Variablen: `a`, `b`, `c` und `text`
- (c)
  - Zeile 4 Operator: `=`  
Operanden: `a`, `4`
  - Zeile 5 Operator: `=`  
Operanden: `b`, `5`
  - Zeile 6 Operator: `=`  
Operanden: `c`, `a+b`
  - Zeile 10 Operator: `=`  
Operanden: `text`, `'{}+{}={}'`.`format(a,b,c)`
- (d)
  - `sleep( )` mit dem Argument `5`
  - `format( )` mit den Argumenten `a`, `b`, `c`
  - `print( )` mit dem Argument `text`
- (e) Ausgabe: `'4 + 5 = 9'`

## Aufgabe 1.7

$(252 * 27 + 319) / (95 * 3 - 729)$

Die Leerzeichen wurden nur der besseren Lesbarkeit wegen eingefügt. Python würde dieselbe Rechnung auch ohne die Leerzeichen durchführen.

## Aufgabe 2.1

Die User-Stories dienen dazu, eine bestimmte Funktion einer Software aus der Sicht einer Person (meist die der Nutzer) zu beschreiben. Sofern dies nicht aus der Beschreibung hervorgeht, ist auch der Nutzen dieser Funktion anzugeben.

Jede User-Story muss also mindestens ein „*Wer* möchte *Was*“ enthalten und zusätzlich noch ein *Wozu*, sofern dies aus dem *Was* nicht schon hervorgeht.

1. Als Benutzer möchte ich, dass meine Daten durch ein Passwort geschützt sind, damit nur ich Zugriff auf die Informationen habe.
2. Als Benutzer möchte ich Namen und Telefonnummern hinzufügen können.
3. Als Benutzer möchte ich, dass mir die Telefonnummer zu einem bestimmten Namen angezeigt wird.
4. Als Benutzer möchte ich die Telefonnummer zu einem Namen ändern können.
5. Als Benutzer möchte ich Namen und Telefonnummern löschen können.

## Aufgabe 2.2

- Eine *Syntax* beschreibt, wie Texte aufgebaut sein müssen.
- Eine *Semantik* beschreibt, was Texte bedeuten.

## Aufgabe 2.3

Die Ziffer 2 wird als Zeichenkette ('2') in der Variablen mit dem Namen x gespeichert. Die Multiplikation einer Zeichenkette mit einer Zahl bewirkt, dass die Zeichenkette entsprechend vervielfacht wird. Also enthält die Variable mit dem Namen y nach der Ausführung von Zeile 2 den Wert '22222'. Diese Zeichenkette wird anschliessend auf der Shell ausgegeben.

## Aufgabe 2.4

```
1 a = 3
2 a = a + 2
3 b = 4
4 c = a + b
5 a = a * c
6 print(a)
```

Ausgabe: 45

Um bei längeren Aufgaben den Überblick zu behalten, empfiehlt es sich, in jeder Zeile die aktuellen Werte der Variablen zu notieren. Noch unbekannte Werte kann man mit ? oder - kennzeichnen.

	a	b	c
a = 3	3	?	?
a = a + 2	5	?	?
b = 4	5	4	?
c = a + b	5	4	9
a = a * c	45	4	9

## Aufgabe 2.5

```
1 x = input('Eingabe: ')
2 y = 5*int(x)
3 print(y)
```

Die Ziffer 2 wird als Zeichenkette ('2') in der Variablen mit dem Namen x gespeichert. Vor der Multiplikation mit der ganzen Zahl 5 wird die Zeichenkette '2' in die entsprechende Zahl 2 umgewandelt. Somit hat das Produkt den Wert 10, der in der Variablen mit dem Namen y gespeichert wird. Dieser Wert 10 wird in Zeile 3 auf der Shell ausgegeben.

## Aufgabe 2.6

- (a) `anzahl_personen` ja
- (b) `4you` nein (Ziffer an erster Stelle verboten)
- (c) `_1234` ja
- (d) `lieferung-mai-2022` nein (enthält Sonderzeichen -)
- (e) `else` nein (Python-Schlüsselwort)
- (f) `Or` ja (`Or`  $\neq$  `or`)

## Aufgabe 2.7

- (a) `'Hello World'` korrekt
- (b) `"Hello World"` nicht korrekt
- (c) `""Hello World""` nicht korrekt
- (d) `'''Hello World'''` korrekt

## Aufgabe 2.8

- (a)  $2**3 \Rightarrow 8$  (int)
- (b)  $4.5 + 2.5 \Rightarrow 7.0$  (float)
- (c)  $24 // 7 \Rightarrow 3$  (int)
- (d)  $47 \% 5 \Rightarrow 2$  (int)
- (e)  $8 / 2 \Rightarrow 4.0$  (float)
- (f)  $'2' + '3' \Rightarrow '23'$  (str)

## Aufgabe 2.9

```
1 print('{}-{}-{}-{}'.format(3, 7, 'a', 'x'))
```

37ax

## Aufgabe 2.10

```
1 a = int(input('Zahl a: '))
2 print(a + b)
3 b = int(input('Zahl b: '))
4 print(a * b)
5 print(a + b
```

- Zeile 2: *Laufzeitfehler*, da die Variable b zu diesem Zeitpunkt noch nicht definiert ist.
- Zeile 4: *semantischer Fehler*, denn die Zahlen sollen addiert und nicht multipliziert werden
- Zeile 5: *Syntaxfehler*, da die schliessende Klammer fehlt

## Aufgabe 3.1

3  
5  
7  
9  
11  
13

### Aufgabe 3.2

6  
9  
12  
15

### Aufgabe 3.3

14  
10  
6  
2

### Aufgabe 3.4

2  
3  
4  
5  
6  
7

### Aufgabe 3.5

18

### Aufgabe 3.6

bryqgm

### Aufgabe 3.7

HGPQEN

### Aufgabe 3.8

Atnybh

### Aufgabe 3.9

True

**Aufgabe 3.10**

False

**Aufgabe 3.11**

False

**Aufgabe 3.12**

True

**Aufgabe 3.13**

True

**Aufgabe 3.14**

True

**Aufgabe 3.15**

True

**Aufgabe 3.16**

False

**Aufgabe 3.17**

True

**Aufgabe 3.18**

False

**Aufgabe 3.19**

True

### Aufgabe 3.20

True

### Aufgabe 3.21

4

### Aufgabe 3.22

8

### Aufgabe 3.23

16

### Aufgabe 3.24

20

### Aufgabe 3.25

Nein

### Aufgabe 3.26

Nein

### Aufgabe 3.27

Ja

### Aufgabe 3.28

Wähle eine ganze und eine gebrochene Zahl, deren Produkt wieder eine ganze Zahl ist, wobei die Darstellung des Bruchs mit vier Stellen zu einem Rundungsfehler führt.

Exakt:  $3 \cdot \frac{1}{3} == 1 \Rightarrow \textit{wahr}$

Für einen Computer, der Dezimalzahlen mit vier Stellen darstellt:

$3.000 \cdot 0.333 == 1.000 \Rightarrow 0.999 == 1.000 \Rightarrow \textit{falsch}$



### Aufgabe 3.29

x	y
9	?
9	10
90	10
90	80

Ausgabe: 80

### Aufgabe 3.30

x	Test
9	-
9	Nein
14	
21	

Ausgabe: 21

### Aufgabe 3.31

x	Test
10	-
10	Ja
15	
23	

Ausgabe: 23

### Aufgabe 3.32

x	Test
3	
7	Nein
11	Nein
15	Nein
19	Ja
19	

Ausgabe: 19

### Aufgabe 3.33

x	Test
13	Nein
9	Nein
5	Nein
1	Nein
-3	Ja

Ausgabe: 13, 9, 5, 1

### Aufgabe 3.34

n	Test	Ausgabe
36	Nein	0
18	Nein	0
9	Nein	1
4	Nein	0
2	Nein	0
1	Nein	1
0	Ja	

### Aufgabe 3.35

```
1 x = 5
2 if x <= 5:
3     x = x + 1
4 else:
5     x = x + 2
6 print(x)
```

Zeile 4: falsche Einrückung

### Aufgabe 3.36

```
1 liste = [1, 2, 3, 4, 5, 6, 7, 8 9, 10, 11, 12]
2 x = 13
3 if x in liste:
4     print('Element kommt nicht in der Liste vor.')
```

Zeile 1: Zwischen 8 und 9 fehlt ein Komma.

Zeile 3: es sollte not in statt in heissen (logischer Fehler)

### Aufgabe 3.37

```
1 x = 52
2 if x =< 50
3     print('ok')
4 else:
5     print('not okay')
```

Zeile 2: <= statt =<

Zeile 2: Doppelpunkt fehlt

### Aufgabe 3.38

```
1 note = 4.5
2 if note = 6:
3     print('sehr gut')
4 else if note == 5:
5     print('gut')
6 elif note == 4:
7     print('genügend')
8 elif note < 4
9     print('ungenügend')
10 else:
11     print('keine gültige Note')
```

Zeile 2: Zuweisung statt Vergleich

Zeile 4: elif statt else if

Zeile 8: Doppelpunkt fehlt

Zeile 11: Hochkomma fehlt am Schluss

### Aufgabe 3.39

```
1 1 = x
2     while x < 4:
3     x = x + 1
4     print(x)
```

Zeile 1: falsche Zuweisung

Zeile 2: falsche Einrückung

### Aufgabe 3.40

```
1 n = 4
2 s = 0
3 i = 1
4 while i <= n
5     s = s + i
6     i = i + 1
7 print s
```

Zeile 4: Doppelpunkt fehlt

Zeile 7: Klammern fehlen (mindestens in Python 3)

### Aufgabe 3.41

```
1 import random
2
3 buchstaben = [a, b, c]
4
5 wahl_computer = random.choice(buchstaben)
6 wahl_user = none
7
8 while wahl_user not in buchstaben:
9     wahl_user = input('a, b, c ?')
10    wahl_user = wahl_user.lower()
11
12 print(wahl_computer, wahl_user)
```

Zeile 3: Hochkommas (Anführungszeichen) fehlen

Zeile 5: random statt random

Zeile 6: None statt none

Zeile 9: schliessendes Hochkomma am falschen Ort

Zeile 10: wahl\_user statt wahl\_user