

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis</b>	<b>2</b>
<b>1 Datentypen</b>	<b>3</b>
<b>2 Variablen</b>	<b>5</b>
<b>3 Arithmetische Operationen</b>	<b>12</b>
<b>4 Ausgabe</b>	<b>18</b>
<b>5 Vergleichsoperationen</b>	<b>21</b>
<b>6 Bedingte Anweisungen</b>	<b>28</b>
<b>7 for-Schleifen</b>	<b>34</b>
<b>8 while-Schleifen</b>	<b>44</b>

# 1 Datentypen

## 1.1 Übung: Lesen von Datentypen

Markieren Sie den Datentyp jedes Wertes:

Wert	Typ
123	integer / float / string / boolean
"123"	integer / float / string / boolean
123.0	integer / float / string / boolean
True	integer / float / string / boolean
'False'	integer / float / string / boolean
"123.0 False"	integer / float / string / boolean

## 1.2 Übung: Schreiben von Datentypen

1) Schreiben Sie den Wert 324 als integer, float und string:

Wert	Typ
	Zahl(integer)
	Zahl(float)
	String

2) Es gibt nur zwei Werte für Booleans. Schreiben Sie diese. Schreiben Sie diese danach als String:

Wert	Typ
	Boolean
	Boolean
	String
	String

## 2 Variablen

### 2.1 Lesen: Variablendeklaration

Bestimmen Sie, ob die angegebenen Variablennamen in Python zulässig sind. Wenn ein Variablenname nicht zulässig ist, erläutern Sie kurz, warum nicht.

Variablennamen	Erlaubt in Python?	Falls nicht erlaubt, wieso nicht?
<code>my_variable</code>	Ja / Nein	
<code>1st_place</code>	Ja / Nein	
<code>_important_</code>	Ja / Nein	
<code>first-name</code>	Ja / Nein	
<code>True</code>	Ja / Nein	
<code>false</code>	Ja / Nein	

### 2.2 Lesen: Variablendeklaration

Stellen Sie fest, ob in den folgenden Zeilen des Python-Codes eine Variable korrekt deklariert ist. Wenn der Code keine Variable deklariert, erklären Sie kurz, warum nicht.

Code zur Deklaration von Variablen	Erlaubt in Python?	Falls nicht erlaubt, wieso nicht?
<code>my_var = "two"</code>	Ja / Nein	
<code>"hello" = reply</code>	Ja / Nein	
<code>happy!response = "yay!"</code>	Ja / Nein	
<code>NAME = "abby"</code>	Ja / Nein	
<code>should_run == False</code>	Ja / Nein	

## 2.3 Übung: Lesen einer Variablenaktualisierung

Führen Sie den folgenden Code aus:

1. Setzen Sie ein Häkchen (✓) links von jeder Zeile, die eine Variable aktualisiert.
2. Gehen Sie den Code zeilenweise durch und aktualisieren Sie die Tabelle, um jede Zeile zu berücksichtigen.
  - 2.1 Bei einer Variablendeklaration schreiben Sie die neue Variable und den Wert in eine leere Zeile.
  - 2.2 Bei einer Variablenaktualisierung streichen Sie den vorherigen Wert durch und schreiben Sie den neuen hinein.

```

_____ num1 = 2
_____ num2 = 1
_____ temp = num1
_____ num1 = num2
_____ num2 = temp

```

Variablenname	Wert
num1	2

## 2.4 Übung: Schreiben von Variablendeklarationen und -aktualisierungen

1) Schreiben Sie in das Feld **Code**, der Folgendes bewirkt:

1. Deklarieren Sie eine Variable namens `cost` und setzen Sie sie auf 1.50.
2. Deklarieren Sie eine Variable namens `item` und setzen Sie sie auf "drink".
3. Deklarieren Sie eine Variable namens `should_buy` und setzen Sie sie auf `False`.
4. Aktualisieren Sie die Variable `cost` auf 1.00.
5. Aktualisieren Sie die Variable `should_buy` auf `True`.

2) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 2.5 Übung: Lesen von Variablenswaps

Lesen Sie für jeden Codeabschnitt den Code und aktualisieren Sie die Variablentabelle. Bestimmen Sie danach die korrekte Antwort aus den Multiple-Choice-Optionen.

1)

```
player1 = "Jose"  
player2 = "Kim"  
temp = player1  
player2 = player1  
player1 = temp
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Es gibt keine 2 zu tauschenden Werte.
  2. Es wurde keine temporäre Variable erstellt.
  3. Es wurde eine temporäre Variable erstellt, aber die Aktualisierung der Variablen wurde nicht korrekt durchgeführt.
  4. Dies ist ein korrekter Variablentausch.
-

2)

```
num1 = 23
temp_val = num1
num2 = num1
num1 = temp
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Es gibt keine 2 zu tauschenden Werte.
2. Es wurde keine temporäre Variable erstellt.
3. Es wurde eine temporäre Variable erstellt, aber die Aktualisierung der Variablen wurde nicht korrekt durchgeführt.
4. Dies ist ein korrekter Variablentausch.

3)

```
current_player = "Wanda"
next_player = "Jimmy"
current_player = next_player
next_player = current_player
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Es gibt keine 2 zu tauschenden Werte.
2. Es wurde keine temporäre Variable erstellt.
3. Es wurde eine temporäre Variable erstellt, aber die Aktualisierung der Variablen wurde nicht korrekt durchgeführt.
4. Dies ist ein korrekter Variablentausch.



## 2.6 Übung: Schreiben von Variablenswaps

Joyce schreibt einen Code, um zu verfolgen, wie viel Geld sie für einen Kredit gezahlt hat. Sie stellt fest, dass sie einen Fehler gemacht und die Werte für die Variablen `amount_paid` und `amount_owed` verwechselt hat. In Wirklichkeit hat sie den Kredit schon fast abbezahlt. Hilf Joyce, indem du die Werte vertauschst, sodass der Betrag, den sie für das Darlehen gezahlt hat (`amount_paid`), größer ist als der geschuldete Betrag (`amount_owed`)!

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

## 2 Variablen

2) Schreiben Sie Code, um die in `amount_paid` und `amount_owed` gespeicherten Werte zu swappen:

```
amount_paid = 231.89
amount_owed = 12152.23
```

Wenn Sie Ihre Arbeit überprüfen wollen, lesen Sie sich den Code durch und füllen Sie die Variablentabelle unten aus:

Variablenname	Wert

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie mit Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

# 3 Arithmetische Operationen

## 3.1 Übung: Lesen arithmetischer Operationen

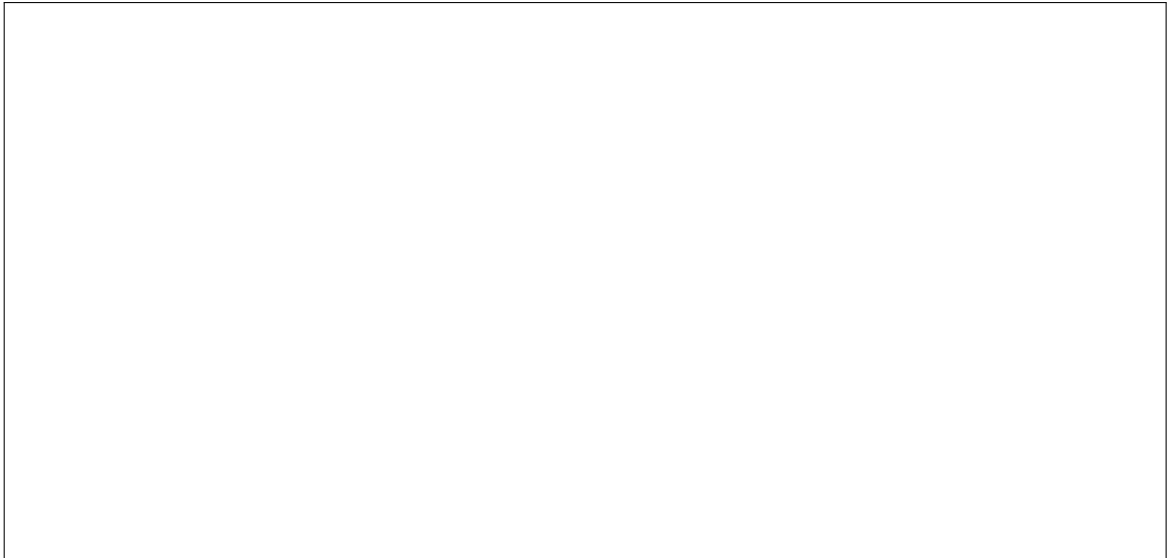
Bestimmen Sie für die folgenden Codezeilen die Datentypen und das Ergebnis.

Code	Datentyp (unterstreiche einen)	Ergebnis
<code>5.0 + 2</code>	integer / float	
<code>9 // 2</code>	integer / float	
<code>my_value = 4</code> <code>my_value * 2</code>	integer / float	
<code>(4 + 2) / 2.0</code>	integer / float	
<code>7 % 2 + 1.1</code>	integer / float	

## 3.2 Übung: Schreiben arithmetischer Operationen

1) Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Deklarieren Sie eine Variable `val` und setzen Sie sie auf  $7 \% 2$ .
- Aktualisieren Sie den Wert von `val`, indem Sie den aktuellen Wert mit der Summe aus `1.0` und `0.2` multiplizieren.



2) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

Vervollständigen Sie diese Aussagen über den Code, den Sie oben geschrieben haben:

- Bei der Deklaration der Variablen `val` handelt es sich um einen **Integer** / **Float** (Wählen Sie die korrekte Antwort aus).
- Wenn die Variable `val` aktualisiert wird, handelt es sich um einen **Integer** / **Float** (Wählen Sie die korrekte Antwort aus).

### 3.3 Übung: Lesen der Verarbeitung von Ziffern

Lesen Sie für jeden Abschnitt den Code und aktualisieren Sie die Variablentabelle. Stellen Sie dann fest, ob die Ziffern korrekt extrahiert werden. Wenn dies nicht der Fall ist, ermitteln Sie den Fehler anhand der Multiple-Choice-Optionen.

1)

```
inp = 31
last_digit = inp % 10
inp = inp // 10
first_digit = inp % 10
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Der Startwert ist kein Integer.
  2. Die Ziffern werden nicht richtig aus dem Startwert extrahiert.
  3. Der Startwert wird nicht richtig aktualisiert.
  4. Dieser Code verarbeitet die Ziffern richtig.
-

2)

```
start = 198
last_digit = inp % 10
inp = inp // 10
second_digit = inp % 10
inp = inp % 10
first_digit = inp % 10
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Der Startwert ist kein Integer.
2. Die Ziffern werden nicht richtig aus dem Startwert extrahiert.
3. Der Startwert wird nicht richtig aktualisiert.
4. Dieser Code verarbeitet die Ziffern richtig.

3)

```
current = 510
last = current % 10
current = current // 10
middle = current % 10
first = current % 10
current = current // 10
```

Variablenname	Wert

Wählen Sie die zutreffende Aussage aus:

1. Der Startwert ist kein Integer.
2. Die Ziffern werden nicht richtig aus dem Startwert extrahiert.
3. Der Startwert wird nicht richtig aktualisiert.
4. Dieser Code verarbeitet die Ziffern richtig.

### 3.4 Übung: Schreiben der Verarbeitung von Ziffern

Es sind 2 Integer als Eingabe (gespeichert in `input_a` und `input_b`) gegeben. Schreiben Sie einen Code, der alle Ziffern in den beiden Eingaben summiert und die Summe in einer Variable `total` speichert.

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

### 3 Arithmetische Operationen

2) Schreiben Sie Code, um alle Ziffern in `input_a` und `input_b` zu summieren.

```
input_a = 23  
input_b = 314
```

Hier ist eine Tabelle für den Fall, dass Sie den Überblick über Ihre Variablen behalten wollen.

Variablenname	Wert

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie mit Ihren eigenen Worten beschreiben, was die Zeile bewirkt.



# 4 Ausgabe

## 4.1 Übung: Lesen von Ausgabeanweisungen

Geben Sie die Ausgabe des Codestücks an.

Code	Ausgabe
<code>print("hello world")</code>	
<code>print(5 - 2.0)</code>	
<code>val = 7</code> <code>print(val)</code> <code>print(val + 1)</code> <code>val = 3</code> <code>print(val)</code>	
<code>print("3+5")</code>	
<code>should_pay = False</code> <code>print(should_pay)</code>	

## 4.2 Übung: Schreiben von Ausgabeanweisungen

1) Schreiben Sie Code, der folgendes macht:

- Deklarieren Sie eine Variable `input` und setzen Sie diese auf 5.
- geben Sie den String "begin" aus.
- geben Sie den in `input` gespeicherten Wert aus.
- geben Sie den in `input` gespeicherten Wert multipliziert mit 2 aus.
- geben Sie den in `input` gespeicherten Wert  $\% 3$  aus.



2) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie mit Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

3) Schreiben Sie die Ausgabe dieses Codes:

# 5 Vergleichsoperationen

## 5.1 Übung: Lesen von Vergleichsoperationen

Bestimmen Sie, ob das Ergebnis der folgenden Codezeilen True oder False ist:

Code	Ergebnis
<code>1.2 &lt; 5</code>	True / False
<code>x = 5</code> <code>y = 4</code> <code>(x + x) &lt;= y and x &lt;= (y + y)</code>	True / False
<code>"HELLO" == "hello"</code>	True / False
<code>a = 1</code> <code>b = 1</code> <code>a != b</code>	True / False
<code>3 &gt;= (5 % 2)</code>	True / False
<code>hello = "Hello"</code> <code>hello == "Hello"</code>	True / False

## 5.2 Übung: Schreiben von Vergleichsoperationen

1) Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Deklarieren Sie eine Variable `greet1` und setzen Sie sie auf `"hello"`.
- Deklarieren Sie eine Variable `greet2` und setzen Sie sie auf `"Hello"`.
- Deklarieren Sie eine Variable `diff_greet` und setzen Sie sie auf das Ergebnis, ob `greet1` und `greet2` nicht gleich sind.

2) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

---

3) Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Deklarieren Sie eine Variable `val1` und setzen Sie diese auf `93`.
- Deklarieren Sie eine Variable `val2` und setzen Sie diese auf die Variable `val1`.
- Deklarieren Sie eine Variable `one_is_less` und setzen Sie diese auf das Ergebnis, ob `val1` kleiner oder gleich `val2` ist.
- Deklarieren Sie eine Variable `check_both` und setzen Sie diese auf das Ergebnis von `val1 % 2 == 0 and one_is_less`.

4) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 5.3 Übung: Lesen des Float-Gleichheits-Templates

Lesen Sie für jeden Codeabschnitt den Code und aktualisieren Sie die Variablentabelle. Schreiben Sie die endgültige Ausgabe in das Feld "Ausgabe", oder schreiben Sie "Fehler", wenn die Ausgabe nicht bestimmt werden kann. Stellen Sie dann fest, ob die Gleichheit der Fließkommazahlen korrekt überprüft wurde. Wenn dies nicht der Fall ist, bestimmen Sie den Fehler anhand der Mehrfachauswahlmöglichkeiten.

1)

```
a = 1.21
b = "1.21"
e = 0.01
d = abs(a - b)
print(d < e)
```

Variablenname	Wert
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die zu vergleichenden Werte sind keine Zahlen (Integer oder Float).
2. Der Schwellenwert ist definiert, aber keine kleine positive Zahl.
3. Die absolute Differenz wird nicht mit einem Wert kleiner als der Schwellenwert verglichen.
4. Dieser Code prüft korrekt die Gleichheit von Floats.

2)

```
x = 2
y = 2.000000004
z = x-y
print(z < 0.00001)
```

Variablenname	Variablenwert
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die zu vergleichenden Werte sind keine Zahlen (Integer oder Float).
2. Der Schwellenwert ist definiert, aber keine kleine positive Zahl.
3. Die absolute Differenz wird nicht mit einem Wert kleiner als der Schwellenwert verglichen.
4. Dieser Code prüft korrekt die Gleichheit von Floats.

3)

```
a = 1.0000002
b = 1.0000001
e = -0.000001
d = a - b
print(abs(d) < e)
```

Variablenname	Variablenwert
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die zu vergleichenden Werte sind keine Zahlen (Integer oder Float).
2. Der Schwellenwert ist definiert, aber keine kleine positive Zahl.
3. Die absolute Differenz wird nicht mit einem Wert kleiner als der Schwellenwert verglichen.
4. Dieser Code prüft korrekt die Gleichheit von Floats.



## 5.4 Übung: Schreiben des Float-Gleichheits-Templates

Angenommen, die Variablen `a`, `b` und `c` sind als Integer oder Floats deklariert. Schreiben Sie einen Code, der prüft, ob `a`, `b` und `c` annähernd gleich sind (innerhalb von 0.001 voneinander). Wenn alle Variablen gleich sind, wird `True` ausgegeben. Andernfalls wird `False` ausgegeben.

Tipp: Ermitteln Sie die Differenzen zwischen `a` und `b` und die Differenzen zwischen `b` und `c`.

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

## 5 Vergleichsoperationen

2) Schreiben Sie Code, um zu prüfen, ob die Variablen `a`, `b` und `c` alle ungefähr gleich sind (innerhalb von 0.001 voneinander).

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

# 6 Bedingte Anweisungen

## 6.1 Übung: Lesen von Bedingten Anweisungen

Lesen Sie die folgenden Codeblöcke. Streichen Sie die Codezeilen durch, die nicht ausgeführt werden. Bestimmen Sie dann, was der Code ausgeben würde.

```
friend = "juan"
temp = 30
if (temp <= 40 and friend == "sue"):
    print("Hole 2 Jacken.")
elif temp <= 35:
    print("Hole 1 Jacke.")
else:
    print("Du brauchst keine Jacke!")
```

Ausgabe:

---

```
num_people = 11
seats_per_table = 4
extra_chairs = 0
max_chairs = 2

if num_people % seats_per_table > 0:
    extra_chairs = num_people % seats_per_table
else:
    print("Es werden keine zusätzlichen Stühle benötigt.")

if extra_chairs > 0 and extra_chairs <= max_chairs:
    print("Wir benötigen keine zusätzlichen Stühle.")
else:
    print("Wir haben nicht genug Stühle.")
```

Ausgabe:

## 6.2 Übung: Schreiben von Bedingten Anweisungen

1) Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Weisen Sie der Variable `profit` den Wert 87 zu.
- Weisen Sie der Variable `cost` den Wert 75 zu.
- Nutzen Sie eine if-Anweisung, um zu prüfen, ob der Gewinn größer als die Kosten ist. Wenn diese Bedingung wahr ist, wird der folgende Code ausgeführt:
  - Weisen Sie der Variable `money_made` das Ergebnis der Subtraktion von `profit` minus `cost` zu.
  - Geben Sie "Gewinn" aus.
  - Geben Sie den Wert aus, der in `money_made` gespeichert ist.
- Nutzen Sie eine else-if-Anweisung, die prüft, ob `profit` und `cost` gleich hoch sind. Wenn diese Bedingung wahr ist, wird der folgende Code ausgeführt:
  - Geben Sie "Ausgeglichen" aus.
- Nutzen Sie eine else-Anweisung, die folgenden Code ausführt:
  - Geben Sie "Verlust" aus.

2) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 6.3 Übung: Lesen des Min- oder Maxtemplates

Lesen Sie für jeden Codeabschnitt den Code und füllen Sie die Variablentabelle aus, um die Werte für die Variablen a, b und c zu ermitteln, die dazu führen würden, dass der Code den Mindestwert **nicht korrekt** ermittelt. Identifizieren Sie dann den Fehler aus den Multiple-Choice-Optionen.

1)

```
if a < b and a < c:
    print(a)
elif c < b:
    print(b)
else:
    print(c)
```

Wählen Sie die zutreffende Aussage aus:

1. Ein Wert wird nicht korrekt eliminiert.
  2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der in der Bedingung geprüft wird.
  3. Alle Werte werden nicht korrekt eliminiert.
- 

2)

```
if a < b:
    print(a)
elif a < c:
    print(a)
elif b < c:
    print(b)
else:
    print(c)
```

Wählen Sie die zutreffende Aussage aus:

1. Ein Wert wird nicht korrekt eliminiert.
2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der in der Bedingung geprüft wird.
3. Alle Werte werden nicht korrekt eliminiert.

3)

```
if a < b and a < c:  
    print(a)  
elif b < c:  
    print(b)
```

Wählen Sie die zutreffende Aussage aus:

1. Ein Wert wird nicht korrekt eliminiert.
2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der in der Bedingung geprüft wird.
3. Alle Werte werden nicht korrekt eliminiert.

## 6.4 Übung: Schreiben des Min- oder Maxtemplates

Geben Sie bei den Variablen `x`, `y` und `z`, die alle Zahlen speichern, die Summe aus Höchst- und Mindestwert aus.

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

2) Schreiben Sie den Code, der die Summe des Maximal- und Minimalwerts der Variablen `x`, `y` und `z` ausgibt. Die Variablen speichern alle Zahlen.

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.



# 7 for-Schleifen

## 7.1 Übung: Lesen von for-Schleifen (1)

Lesen Sie die folgenden Codeblöcke durch. Bestimmen Sie, was der Code ausgibt und schreiben Sie den Wert des Iterators (**number**) und den Wert der Variable **result** nach jedem Durchgang in die Tabelle.

1)

```
result = 0

for number in range(7):
    if number % 2 == 0:
        result = result + number

print(result)
```

Variablenname	Variablenwert
number	
result	
Ausgabe	

---

2)

```
result = 0

for number in range(5):
    if number % 3 == 0:
        result = result + number

print(result)
```

Variablenname	Variablenwert
number	
result	
Ausgabe	

---

3)

```
result = 5

for number in range(1, 6):
    if number % 2 == 0:
        result = result - number

print(result)
```

Variablenname	Variablenwert
number	
result	
Ausgabe	

## 7.2 Übung: Lesen von for-Schleifen(2)

Lesen Sie die folgenden Codeblöcke durch. Bestimmen Sie, was der Code ausgibt und schreiben Sie den Wert des Iterators (`number`) und den Wert der Variable `result` nach jedem Durchgang in die Tabelle.

1)

```
result = 0
```

```
for number in range(7):
    if number % 2 == 0:
        result += number
    else:
        result -= number
```

```
print(result)
```

Variablenname	Variablenwert
number	
result	
Ausgabe	

2)

```
result = 0
```

```
for number in range(5):
    if number % 3 == 0:
        result += number * 2
    else:
        result += number
```

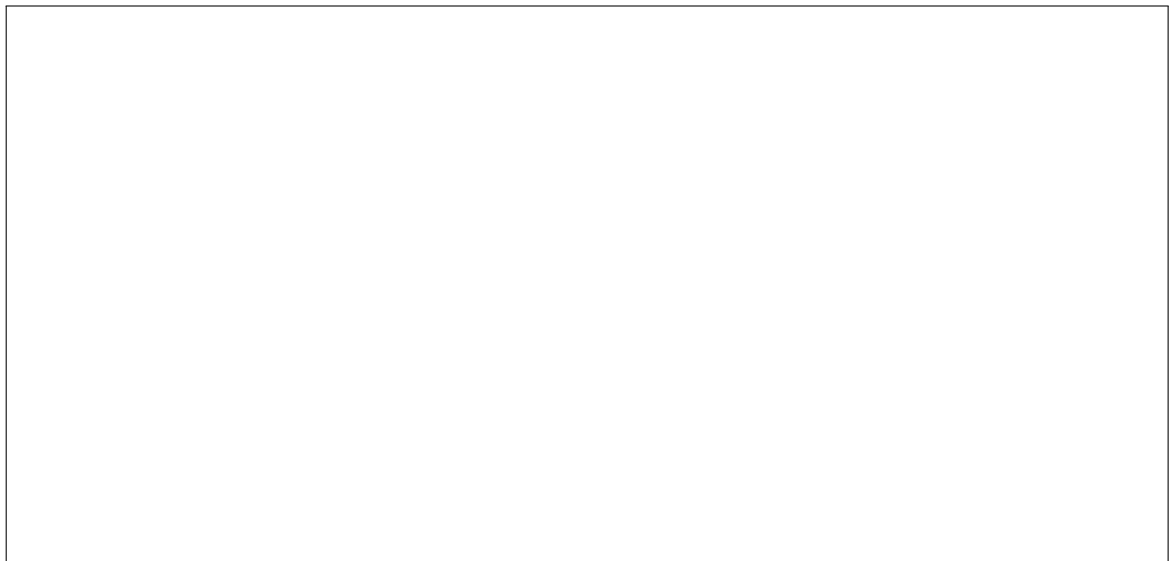
```
print(result)
```

Variablenname	Variablenwert
number	
result	
Ausgabe	

## 7.3 Übung: Schreiben von for-Schleifen (1)

Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Nutzen Sie eine for-Schleife, um alle geraden natürlichen Zahlen zwischen 1 und einschließlich 10 auszugeben.
  - Nutzen Sie eine if-Anweisung, um mithilfe des Modulo-Operators zu überprüfen, ob die Zahl (ihr Iterator) gerade ist und demnach durch 2 teilbar ist.
  - Wenn dies der Fall ist, dann geben Sie die Zahl aus.



Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 7.4 Übung: Schreiben von for-Schleifen (2)

Schreiben Sie Code, der das Ergebnis aller ungeraden ganzen Zahlen zwischen 1 bis einschließlich 10 berechnet.

- Weisen Sie der Variable `total` den Wert 0 zu.
- Nutzen Sie eine for-Schleife, um alle ungeraden natürlichen Zahlen zwischen 1 und einschließlich 10 zu addieren und in der Variable `total` zu speichern.
- Geben Sie die Variable `total` nach der for-Schleife aus.

Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 7.5 Übung: Schreiben von *for*-Schleifen (3)

Schreiben Sie Code, der das Ergebnis aller ganzen natürlichen Zahlen zwischen 1 bis einschließlich 30 berechnet.

Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 7.6 Übung: Lesen des Primzahltemplates

Lesen Sie für jeden Codeabschnitt den Code, der überprüfen soll, ob es sich bei `number` um eine Primzahl handelt. Füllen Sie danach die Variablentabelle mit den Werten für die Variablen `i` und `is_prime` aus. Wählen Sie zuletzt die korrekte Antwort aus den Multiple-Choice-Optionen.

1)

```
number = 9
is_prime = True

for i in range(2, number):
    if number % i != 0:
        is_prime = False

if is_prime:
    print(number)
```

Variablenname	Variablenwert
i	
is_prime	
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die Primzahlprüfung ist **nicht** korrekt.
  2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der ausgegeben werden soll.
  3. Die Primzahlprüfung ist korrekt.
-

2)

```

number = 7
is_prime = True

for i in range(2, number):
    if number % i == 0:
        is_prime = False

if is_prime:
    print(i)

```

Variablenname	Variablenwert
i	
is_prime	
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die Primzahlprüfung ist **nicht** korrekt.
2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der ausgegeben werden soll.
3. Die Primzahlprüfung ist korrekt.

3)

```

number = 7
is_prime = False

for i in range(2, number):
    if number % i == 0:
        is_prime = False

if is_prime:
    print(number)

```

Variablenname	Variablenwert
i	
is_prime	
Ausgabe	

Wählen Sie die zutreffende Aussage aus:

1. Die Primzahlprüfung ist **nicht** korrekt.
2. Der ausgegebene Wert stimmt nicht mit dem Wert überein, der ausgegeben werden soll.
3. Die Primzahlprüfung ist korrekt.



## 7.7 Übung: Schreiben des Primzahltemplates

Nachdem wir in der Übung zum Lesen von Templates bereits betrachtet haben, wie Sie eine Primzahlprüfung durchführen, sollen Sie nun Code schreiben, der bestimmt, ob die Zahl **number** eine Primzahl ist und sofern dies wahr ist "Primzahl ausgeben", ansonsten geben Sie "Keine Primzahl" aus.

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

```
number = 25
```

2) Schreiben Sie Code, der bestimmt, ob es sich bei der Zahl 25 um eine Primzahl handelt.

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

# 8 while-Schleifen

## 8.1 Übung: Lesen von while-Schleifen

Lesen Sie für jeden Codeabschnitt den Code durch, welcher die Werte zwischen 0 und 9 addieren soll. Wählen Sie danach den Fehler aus, der dazu führen würde, dass der Code die Summe **nicht korrekt** ermittelt.

1)

```
result = 0

while count < 10:
    result += count
    count += 1
```

Wählen Sie die korrekte Antwort aus:

1. Die Bedingung der while-Schleife führt zu einer Endlosschleife.
  2. Der Iterator der while-Schleife wurde nicht definiert.
  3. Der Iterator wird nicht erhöht.
- 

2)

```
count = 0
result = 0

while count < 10:
    result += count
```

Wählen Sie die korrekte Antwort aus:

1. Die Bedingung der while-Schleife führt zu einer Endlosschleife.
2. Der Iterator der while-Schleife wurde nicht definiert.
3. Der Iterator wird nicht erhöht.

3)

```
count = 0  
result = 0
```

```
while count > 0:  
    result += count  
    count += 1
```

Wählen Sie die korrekte Antwort aus:

1. Die Bedingung der while-Schleife führt zu einer Endlosschleife.
2. Der Iterator der while-Schleife wurde nicht definiert.
3. Der Iterator wird nicht erhöht.

## 8.2 Übung: Schreiben von while-Schleifen

Schreiben Sie Code, der die folgenden Aufgaben erfüllt:

- Deklarieren Sie eine Variable `count` und setzen Sie diese auf 10.
- Deklarieren Sie eine Variable `cost` und setzen Sie diese auf 10.
- Deklarieren Sie eine Variable `profit` und setzen Sie diese auf 15.
- Nutzen Sie eine while-Schleife mit der Abbruchbedingung, dass `profit > cost` ist.
  - Nutzen Sie innerhalb der while-Schleife eine if-Anweisung, um zu prüfen, ob die Variable `count`  $\geq 10$  ist. Wenn diese Bedingung wahr ist, führen Sie folgenden Code aus:
    - \* erhöhen Sie `profit` um 3.
    - \* erhöhen Sie `cost` um 1.
    - \* verringern Sie `count` um 1.
  - Nutzen Sie eine else-if-Anweisung, die prüft, ob `count > 5` ist. Wenn diese Bedingung wahr ist, führen Sie folgenden Code aus:
    - \* erhöhen Sie `profit` um 2.
    - \* erhöhen Sie `cost` um 2.
    - \* verringern Sie `count` um 1.
  - Nutzen Sie eine else-if-Anweisung, die prüft, ob `count > 0` ist. Wenn diese Bedingung wahr ist, führen Sie folgenden Code aus:
    - \* erhöhen Sie `profit` um 1.
    - \* erhöhen Sie `cost` um 3.
    - \* verringern Sie `count` um 1.
- Geben Sie am Ende die Variablen `count`, `cost` und `profit` aus.

Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.

## 8.3 Übung: Lesen des Quersummentemplates

Lesen Sie für jeden Codeabschnitt den Code, der die Quersumme einer Zahl berechnen soll. Wählen Sie die korrekte Antwort aus den Multiple-Choice-Optionen.

1)

```
number = 123
checksum = 0

while number > 0:
    checksum += number % 10
    number = number / 10
```

Wählen Sie die korrekte Antwort aus:

1. Die Ziffern der ursprünglichen Zahl werden nicht korrekt eliminiert.
  2. Der Ziffern der ursprünglichen Zahl werden nicht korrekt der Quersumme hinzugefügt.
  3. Die Entfernung der Ziffer erfolgt vor dem Hinzufügen zur Quersumme.
- 

2)

```
number = 123
checksum = 0

while number > 0:
    number = number // 10
    checksum += number % 10
```

Wählen Sie die korrekte Antwort aus:

1. Die Ziffern der ursprünglichen Zahl werden nicht korrekt eliminiert.
  2. Der Ziffern der ursprünglichen Zahl werden nicht korrekt der Quersumme hinzugefügt.
  3. Die Entfernung der Ziffer erfolgt vor dem Hinzufügen zur Quersumme.
-

3)

```
number = 123
checksum = 0

while number > 0:
    checksum = number % 10
    number = number // 10
```

Wählen Sie die korrekte Antwort aus:

1. Die Ziffern der ursprünglichen Zahl werden nicht korrekt eliminiert.
2. Der Ziffern der ursprünglichen Zahl werden nicht korrekt der Quersumme hinzugefügt.
3. Die Entfernung der Ziffer erfolgt vor dem Hinzufügen zur Quersumme.



## 8.4 Übung: Schreiben des Quersummentemplates

Sie bekommen die Zahl **12345** übergeben. Ihre Aufgabe ist es, die Quersumme zu bilden. Geben Sie die Quersumme aus.

1) Beschreiben Sie einen schrittweisen Plan zur Lösung des Problems:

2) Schreiben Sie Code, der die Quersumme der Zahl **12345** berechnet.

3) Wenn Sie mit dem Schreiben des Codes fertig sind, schreiben Sie neben jede Zeile einen Kommentar, in dem Sie in Ihren eigenen Worten beschreiben, was die Zeile bewirkt.