

Python ist eine sehr einfach zu erlernende Programmiersprache und für den Einstieg in die Welt der Programmierung besonders gut geeignet. Zudem zählt Python zu einer der bekanntesten textbasierten Programmiersprachen, genauer genommen zu den imperativen Programmiersprachen. Imperativ bedeutet, dass ein Befehl nach dem anderen - der Reihe nach, von oben nach unten - abgearbeitet wird.

Python ist eine frei verfügbare Programmiersprache, die unter verschiedenen Betriebssystemen verwendet werden kann.

Die aktuelle Python-Version kann einerseits unter dem folgenden Link für das entsprechende Betriebssystem heruntergeladen und installiert werden: <https://www.python.org/>.

Dabei wird auch die Entwicklungsumgebung IDLE (kurz für "Integrated Development and Learning Environment") mitinstalliert. Nähere Informationen zu IDLE gibt es hier: <https://docs.python.org/3/library/idle.html>

Zur Bearbeitung von Übung 4 ist es nicht zwingend notwendig, Python (und eine passende Entwicklungsumgebung) am Computer zu installieren. Es gibt diverse Online-Editoren, wie beispielsweise die Plattform <https://www.online-python.com/>. In dieser webbasierten Entwicklungsumgebung kannst du Python-Code schreiben, ausführen und herunterladen.

Das interaktive Buch zur Einführung in Python (Lektion 4 im MOOC) enthält eine Sammlung an Links, die dich gezielt zu Lernmaterialien weiterleiten. Um dir den Einstieg in die Programmierung mit Python weiter zu erleichtern und um sicherzustellen, dass du über die erforderlichen Konzepte zur Lösung von Übung 4 Bescheid weißt, stellen wir nachfolgend ein Schritt-für-Schritt-Tutorial mit einigen Code-Beispielen zur Verfügung. Es besteht allerdings kein Anspruch auf Vollständigkeit.

Programmieren ist Übungssache! Daher ist es wichtig ist, dass du aktiv mit**machst** und den Code direkt eintippst und ausführst. Die abgebildeten Screenshots zeigen die Plattform <https://repl.it/>.

Die Code-Beispiele (Abbildung 1 – 8) findest du hier: <https://replit.com/@MariaG1/tutorialcodeexamples#main.py>

1 Python als Taschenrechner

Python kann im ersten Schritt wie ein einfacher Taschenrechner eingesetzt werden. Die Operatoren +, -, * und / können wie gewohnt für Berechnungen eingesetzt werden.

Tippe **in der Konsole/im Terminal** die folgenden Ausdrücke nacheinander ein und drücke jeweils die ENTER-Taste.

```
2+2
```

```
(50 - 5 * 6) / 4
```

```
8 / 5
```

```
8 / -5
```

```
8 // -5          # Division von ganzen Zahlen
```

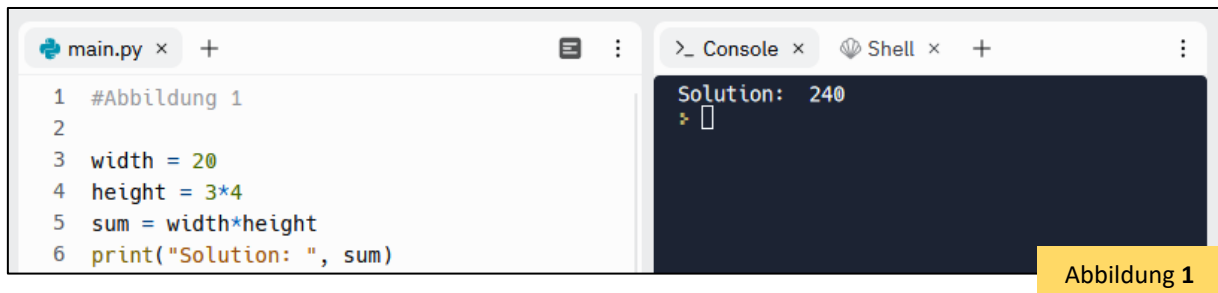
```
125.6 * 2.47     # Wichtig: Als Dezimalzeichen muss der Punkt . verwendet werden.
```

```
10 % 7           # Gibt den ganzzahligen Rest der Division 10:7 aus  
                  Mit dem Operator % (Modulo) wird  
                  der (ganzzahlige) Rest einer Division  
                  von ganzen Zahlen berechnet.
```

2 Variablen und die Bedeutung des Gleichheitszeichens

In Python können ganze Zahlen, Dezimalzahlen, Zeichen, Wörter und Texte (oder allgemein: Zeichenketten) in einer Variablen gespeichert werden. Das funktioniert ganz einfach mithilfe des Zuweisungsoperators (=). Der bezeichnende Name der Variablen kann auch aus Buchstaben, Ziffern und Zeichen (häufig: _) bestehen, darf aber nicht mit einer Ziffer beginnen. Die Groß- und Kleinschreibung ist zu beachten (case sensitive).

Sieh dir das Python-Programm (Abbildung 1) an und tippe die folgenden Befehle im **Editor-Bereich** (deiner Entwicklungsumgebung) ein:



Erläuterung:

1. Wir deklarieren **2** Variablen mit den Namen `width` und `height` und weisen ihnen zwei Anfangswerte (rechts vom Gleichheitszeichen) zu.

```
width = 20
height = 5*4
```

2. Wir deklarieren eine weitere Variable mit dem Namen `sum` und weisen ihr einen Wert zu. Variablen müssen **deklariert** und **initialisiert** werden, bevor sie benutzt werden können. Ansonsten tritt ein Fehler auf. Probiere es einfach aus - TRIAL AND ERROR!

```
sum = width * height
```

3. Um den Wert der Variablen `sum` auszugeben, verwenden wir den Befehl `print`. Um Zeichenketten auszugeben, müssen diese beim Befehl `print` mit doppelten Anführungszeichen angegeben werden.

```
print("Solution: ", sum)
```

Übung 1

Schreibe ein Programm in Python, das die Umrechnung eines beliebigen Zahlenwerts von km/h (Kilometer pro Stunde) in m/s (Meter pro Sekunde) implementiert.

Speichere den Umrechnungsfaktor in einer Variablen und erzeuge die folgende Ausgabe:
(xx steht hier nur als Platzhalter für den gewählten Wert)

```
xx km/h = xx m/sec
```

3 Auf BenutzerInnen-Eingaben reagieren

Mit dem Befehl `input()` können Eingaben von BenutzerInnen in der **Konsole** gelesen werden.

Mit dem Symbol `#` können (einzeilige) Kommentare im Code eingeleitet werden.

Mehrzeilige Kommentare werden mit drei (einfachen oder doppelten) Anführungszeichen (`"""`) eingeleitet und beendet.

Kommentare werden vom Interpreter ignoriert und dienen dazu, das Programm zu strukturieren und einzelne Dinge im Programm zu erklären.

```
18 print("Das ist kein Kommentar!")
19
20 # Hier könnte dein Kommentar stehen.
21
22 """
23 Hier könnte dein
24 mehrzeiliges
25 Kommentar stehen.
26 """
```

Abbildung 2

Starte mit einem leeren Programm, gib den folgenden Code ein und teste dein Programm.



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 print("Please enter your name: ")
2
3 #Speichere die Eingabe in der Variablen username
4 username = input()
5
6 print("Hello ", username, "!")
7
```

The console on the right shows the output of the program:

```
Please enter your name:
Maria
Hello Maria !
```

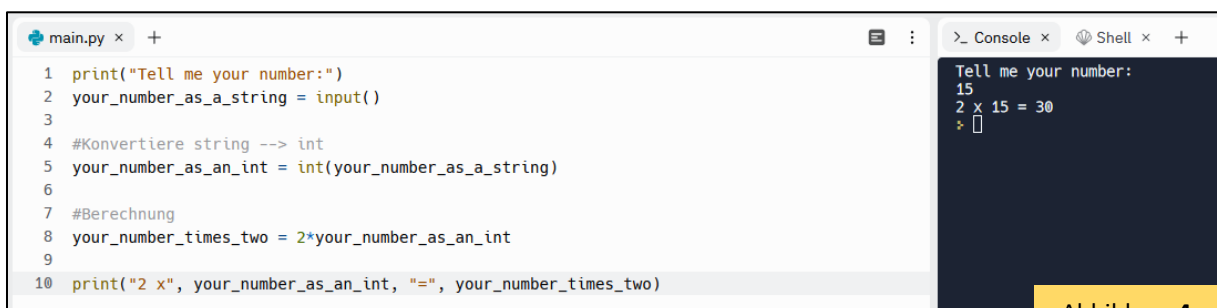
Abbildung 3

4 Mit BenutzerInnen-Eingaben rechnen

Mit dem Befehl `input()` wird eine Variable vom Datentyp "String" erstellt.

Ein "String" ist ein Text oder - genauer gesagt - eine Zeichenkette (aus mehreren Zeichen oder Wörtern). Mit einem "Text" kann das Python-Programm jedoch nicht rechnen. Deshalb muss die eingegebene Zeichenkette in eine Zahl umgewandelt werden.

Um die BenutzerInnen-Eingabe in eine ganze Zahl umzuwandeln, muss der Befehl `int()` verwendet werden. (siehe Abbildung 4)



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

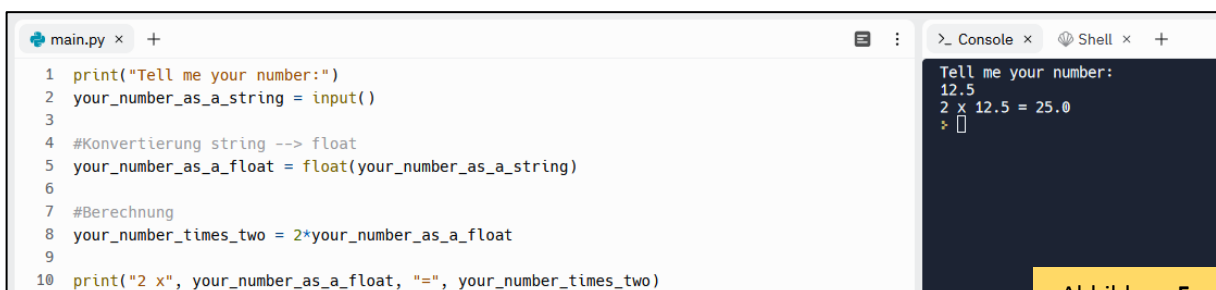
```
1 print("Tell me your number:")
2 your_number_as_a_string = input()
3
4 #Konvertiere string --> int
5 your_number_as_an_int = int(your_number_as_a_string)
6
7 #Berechnung
8 your_number_times_two = 2*your_number_as_an_int
9
10 print("2 x", your_number_as_an_int, "=", your_number_times_two)
```

The console on the right shows the output of the program:

```
Tell me your number:
15
2 x 15 = 30
```

Abbildung 4

Analog dazu gibt es auch den Befehl `float()`, um eine Variable vom Datentyp "string" in eine **Gleitkommazahl** umzuwandeln.



The screenshot shows a Python IDE with a file named `main.py`. The code in the editor is as follows:

```
1 print("Tell me your number:")
2 your_number_as_a_string = input()
3
4 #Konvertierung string --> float
5 your_number_as_a_float = float(your_number_as_a_string)
6
7 #Berechnung
8 your_number_times_two = 2*your_number_as_a_float
9
10 print("2 x", your_number_as_a_float, "=", your_number_times_two)
```

The console on the right shows the output of the program:

```
Tell me your number:
12.5
2 x 12.5 = 25.0
```

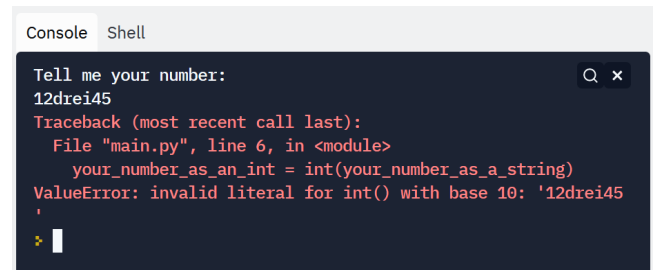
Abbildung 5

Der in Abbildung 4 und Abbildung 5 gezeigte Code ist bei beliebigen BenutzerInnen-Eingaben sehr fehleranfällig.

Warum?

Die Befehle `int(user_input)` bzw. `float(user_input)` akzeptieren nur Strings, welche auch tatsächlich in eine Ganzzahl bzw. Gleitkommazahl umgewandelt werden können.

Lautet die BenutzerInnen-Eingabe "12drei45", so kann dieser String nicht in eine Zahl konvertiert werden und das Programm wird mit einer Fehlermeldung abgebrochen.



```
Console Shell
Tell me your number:
12drei45
Traceback (most recent call last):
  File "main.py", line 6, in <module>
    your_number_as_an_int = int(your_number_as_a_string)
ValueError: invalid literal for int() with base 10: '12drei45'
```

Um solche Fehlermeldungen zu vermeiden, gibt es jedoch eine Lösung in Python.

5 Fehlerbehandlung mit `try` und `except`

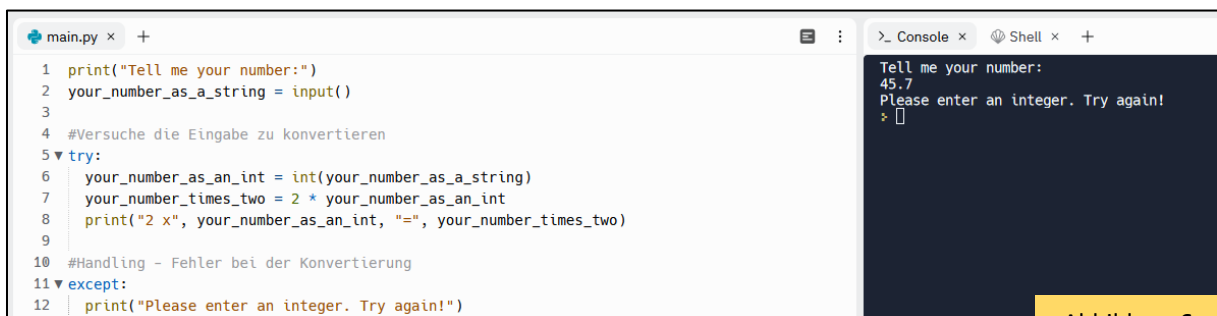
Um das oben beschriebene Problem zu lösen, können an bestimmten (kritischen) Stellen des Programms Fehler zugelassen bzw. abgefangen werden. Häufig sind das jene Stellen im Programm, wo mit BenutzerInnen-Eingaben weitergearbeitet wird.

Siehe dir das folgende Code-Beispiel (Abbildung 6) an.

Tritt bei der Ausführung des `try`-Blocks ein Fehler auf, dann wird dieser im `except`-Block abgefangen.

Tritt bei der Ausführung des `try`-Blocks **kein** Fehler auf, dann wird die "Fehlerbehandlung" im `except`-Block erst gar nicht ausgeführt.

Merke dir: Nur die **kritischen Zeilen** gehören in den `try`-Block.



```
main.py x +
1 print("Tell me your number:")
2 your_number_as_a_string = input()
3
4 #Versuche die Eingabe zu konvertieren
5 try:
6     your_number_as_an_int = int(your_number_as_a_string)
7     your_number_times_two = 2 * your_number_as_an_int
8     print("2 x", your_number_as_an_int, "=", your_number_times_two)
9
10 #Handling - Fehler bei der Konvertierung
11 except:
12     print("Please enter an integer. Try again!")
```

```
>_ Console x Shell x +
Tell me your number:
45.7
Please enter an integer. Try again!
```

Abbildung 6

Im obigen Code werden **alle Arten von Fehlern** abgefangen, die Python automatisch erkennen kann. Du kannst auch nur bestimmte Fehler (z.B. `except ValueError:`) abfangen.

Sieh dir nun den folgenden Code (Abbildung 7) genau an.

Versuche zu verstehen, wie das Programm auf eine richtige und eine falsche BenutzerInnen-Eingaben reagiert.

```
main.py x +
1 print("Tell me your number:")
2 your_number_as_a_string = input()
3 error = 0
4
5 #Versuche die Eingabe zu konvertieren
6 try:
7     your_number_as_an_int = int(your_number_as_a_string)
8
9 #Handling - Fehler bei der Konvertierung
10 except:
11     print("Please enter an integer. Try again!")
12     error = 1
13
14 if error == 0:
15     your_number_times_two = 2 * your_number_as_an_int
16     print("2 x", your_number_as_an_int, "=", your_number_times_two)
17 else:
18     print("Bye!")
```

Abbildung 7

```
>_ Console x Shell x +
Tell me your number:
7
2 x 7 = 14
> []
```

```
>_ Console x Shell x +
Tell me your number:
12drei45
Please enter an integer. Try again!
Bye!
> []
```

```
>_ Console x Shell x +
Tell me your number:
14.3
Please enter an integer. Try again!
Bye!
> []
```

6 Bedingte Anweisungen im Python

Nicht nur im täglichen Leben sondern auch beim Programmieren gibt es Situationen, in denen bestimmte Entscheidungen getroffen werden müssen. Abhängig davon, ob bestimmte Ereignisse eintreten bzw. ob eine Variable zu einem bestimmten Zeitpunkt einen bestimmten Wert hat, werden bestimmte Befehle (Codezeilen) ausgeführt oder eben nicht ausgeführt.

Die Rede ist von bedingten Anweisungen oder Verzeigungen.
Solche Anweisungen sind nach dem **WENN-DANN-(SONST)-Schema** aufgebaut.

Dazu werden in Python die Schlüsselwörter `if... elif... else` benötigt.
(wenn... sonst wenn... sonst...)

Wir schreiben zur Übung wieder ein einfaches Programm (Abbildung 8).
Da im Programm nun Zufallszahlen vorkommen, müssen wir das Modul *random* importieren.
Dieses Modul enthält alle Befehle, die wir zum Arbeiten mit Zufallszahlen benötigen.

Gib das Programm Zeile für Zeile ein und führe es aus. Beachte, dass die Befehle, die bei erfüllter Bedingung ausgeführt werden sollen, (mit einem Tab) eingerückt werden müssen.

```
main.py x +
1 #Importiere das random-Modul (random = zufällig)
2 import random
3
4 #Initialisiere den Zufallszahlen-Generator
5 random.seed()
6
7 #Erzeuge eine ganzzahlige Zufallszahl (integer) zwischen 0 und 20
8 number = random.randint(0,20)
9 print("Your number is:", number)
10
11 #Gib eine Nachricht in Abhängigkeit vom Wert der Zufallszahl aus
12 if number > 7:
13     print(number, "is greater than 7")
14 elif number < 7:
15     print(number, "is smaller than 7")
16 else:
17     #number == 7
18     print("Congratulations! Lucky Seven!")
```

Abbildung 8

Weitere kurze Code-Beispiele zu diesem Thema findest du hier:
<https://www.sivakids.de/python-if-bedingung/>

7 Schleifen in Python

Schleifen sind eine besondere Stärke der Programmierung:
Die schnelle wiederholte Bearbeitung ähnlicher Vorgänge.

In Python wird zwischen 2 Typen von Schleifen unterschieden: **for**-Schleife und **while**-Schleife

Klicke auf den Link, um das Beispielprogramm zu öffnen:

<https://replit.com/@MariaG1/schleifen#main.py>

Das Programm beinhaltet 3 Beispiele zum Einsatz von Schleifen.

Zur Erinnerung:

Mit `"""... """` (3 Anführungszeichen) kannst du größere Teile des Codes auskommentieren.
Der auskommentierte Code wird dann nicht interpretiert/ausgeführt.

Übung 2

Versuche zu verstehen, was bei der Ausführung des Programms

<https://replit.com/@MariaG1/schleifen#main.py>

passiert und wie die Ausgaben zustande kommen.

Denke darüber nach, in welcher Situation es Sinn macht, eine **for**-Schleife bzw. eine **while**-Schleife zu verwenden.

Weitere kurze Code-Beispiele zu diesem Thema findest du hier:

<https://www.sivakids.de/python-for-schleife/>

<https://www.sivakids.de/python-while-schleife/>

Übung 3

Schreiben Sie ein Programm in **Python**, das die folgende Funktionalität implementiert:

- Signalisieren Sie dem Benutzer/der Benutzerin, dass er/sie eine **ganze Zahl (Zahl 1)** eingeben soll.
- Signalisieren Sie dem Benutzer/der Benutzerin, dass er/sie eine **ganze Zahl (Zahl 2)** eingeben soll.
- Handelt es sich bei der ersten Eingabe oder bei der zweiten Eingabe (oder bei beiden) **nicht** um eine Ganzzahl, dann soll der Benutzer/die Benutzerin erneut aufgefordert werden, zwei ganze Zahlen nacheinander einzugeben.
- Signalisieren Sie dem Benutzer/der Benutzerin, dass er/sie eine **falsche Eingabe** getätigt hat.
- Die Eingabe-Aufforderung soll **solange wiederholt** werden, **bis** der Benutzer/die Benutzerin zwei ganze Zahlen eingegeben hat.
- Berechnen Sie das Produkt der eingegebenen Zahlen und geben Sie das **Ergebnis** aus.
- **Prüfen** Sie, **ob** das Produkt **größer als Null**, **kleiner als Null** bzw. **gleich der Zahl 42** ist.
 - **Produkt > 0:**
 - Geben Sie den Text *Countdown starts!* aus.
 - Programmieren Sie einen **Countdown**, der den Wert des Produkts **jede Sekunde* um 1** erniedrigt. Der Countdown endet bei **0**.
 - Geben Sie immer den aktuellen Wert des Countdowns in der Konsole aus.

- Geben Sie nach Ablauf des Countdowns den Text *It's time for a coffee break!* aus.

***Hinweis:**

Sie müssen das Modul `time` importieren, um die Funktion `time.sleep(seconds)` verwenden zu können.

```
main.py ×  
1  import time  
2  
3  print("Hi")  
4  time.sleep(0.5)  
5  print("Maria!")
```

- **Produkt = 42:**
 - Geben Sie den Text *It is always coffee time!* aus.
- **Produkt <= 0:**
 - Geben Sie den Text *Try again, if you love coffee!* aus.

Eine mögliche Lösung zu diesem Beispiel finden Sie hier:

<https://replit.com/@MariaG1/musterloesunguebung3#main.py>