# Programmieren mit Python

## Teil 2: Conditionals

Dr. Aaron Kunert
*aaron.kunert@salemkolleg.de*

22. April 2021

# Script Mode

**Script Mode**

Sobald man mehrere zusammenhängende Zeilen hat, wird die REPL sehr unübersichtlich. Daher gibt es auch die Möglichkeit, alle Programmzeilen zunächst aufzuschreiben und diese dann gebündelt von Python ausführen zu lassen. Im Gegensatz zum REPL bzw. interactive Mode von Python wird dies *Script Mode* genannt.

## Beispiel

```
1 name = "Max"
2 age = 20
3 print(f"Hello, I'm {name} and I'm {age} years old")
```

**Problem:**

Wie kann man Python erklären, diese 3 Zeilen auf einmal auszuführen?

**Old-School-Lösung**

- Erstelle eine neue Datei (z.B. `my_script.py`)
- Öffne die Datei mit einem Texteditor und speichere den Beispiel-Code darin ab.
- Öffne den Ordner mit der Datei `my_script.by` mit dem Terminal bzw. der Eingabeaufforderung
- Führe das Kommando `python my_script.py` aus.

**Optimallösung: Verwende eine IDE**

Eine IDE (integrierte Entwicklungsumgebung) hilft Dir beim Programmieren und unterstüzt Dich wo immer möglich. Dadurch lassen sich auch große Projekte schnell umsetzen.

**Nachteile**

Die anfängliche Einrichtung kann schnell kompliziert werden. Aufgrund der vielen Features fühlt man sich schnell mal überfordert.

→ Das machen wir etwas später.

**Kompromiss für den Anfang: Browserbasierte Editoren**

Um schnell einzusteigen, kann zu Beginn auch ein browsergestützter Editor/Interpreter verwendet werden. Zum Beispiel:

- Programiz (`https://www.programiz.com/python-programming/online-compiler`)
  - einfacher Einstieg
  - schnell und unkompliziert
  - geringer Funktionsumfang
- Repl.it (`https://replit.com/languages/python3`)
  - Auch für viele andere Sprachen geeignet
  - Manchmal etwas langsam
  - Man kann mehrere Dateien und Projekte verwalten (braucht Account)
  - Hat fast alle IDE-Features (braucht Account)

# Input/Output

**Kommunikation über die Konsole**

**Die Konsole**

Da wir zu Beginn noch über keine grafische Benutzeroberfläche verfügen, verwenden wir für die Kommunikation mit unserem Programm die *Konsole*. Dabei handelt es sich um ein einfaches Textfenster, auf dem Dein Programm Informationen ausgeben kann (*Output*) und Text einlesen kann (*Input*).

**Output**

Um einen String auf der *Konsole* auszugeben, verwende die Funktion `print()`.

Zum Beispiel: `print("Hello there")`.

Es können auch Variablen eingesetzt werden:

```
message = "Hello there"
print(message) # Hello there
```

### String Interpolation

Um Variablenwerte innerhalb eines Strings auszugeben, verwenden wir die String-Interpolation-Syntax:

```
my_value = 5
print(f"The variable my_value has the value {my_value}")
# The variable my_value has the value 5
```

Das geht auch als *inline expression*:

```
print(f"The sum of 1 and 2 is {1+2}")
# The sum of 1 and 2 is 3
```

### Input

Um einen String vom User einzulesen, verwende die Funktion `input()`:

```
1 age = input("How old are you?")
2 print(f"I am {age} years old")
```

### Achtung

Das Ergebnis von `input` hat stets den Datentyp `string` auch wenn Zahlen eingelesen werden. Gegebenenfalls muss das Ergebnis mittels `int()` oder `float()` in den gewünschten Typ umgewandelt werden.

## Beispiel: Input und Output kombiniert

```
1 name = input("What is your name?")
2 age = input("What is your age?")
3 print(f"Hello {name}, you are {age} years old")
```

# Übung

**Adressabfrage**

Schreibe ein kurzes Skript, dass Dich nach Deinem Namen, Alter und Adresse fragt. Wenn es alles eingelesen hat, soll es diese Infos in folgender Form auf der Konsole ausgeben:

```
Hallo Max, schön dass Du da bist. Du bist 21 Jahre alt und wohnst in der Bismarckstraße 12 in Glücksstadt.
```

# Übung

**Blick in die Zukunft**

Schreibe ein kurzes Skript, dass Dich nach Deinem Alter fragt. Daraufhin soll es auf der Konsole ausgeben, wie alt Du in 15 Jahren sein wirst.

# Kommentare

### Kommentare

Alle Zeichen einer Zeile, die hinter einem # (Hashtag) kommen, werden von Python ignoriert. So lassen sich Kommentare im Quellcode platzieren.

### Beispiel

```
1 print("This line will be printed")
2 # print("This line won't")
```

# Conditionals

**Ein Programm verzweigen**

**Problemstellung**

Lies eine Zahl `x` ein. In Abhängigkeit von `x` soll Folgendes ausgegeben werden:

```
Die Zahl x ist größer als 0
```

bzw.

```
Die Zahl x ist kleiner 0
```

Wie macht man das?

## Lösung (fast)

```
x = input("Gib eine Zahl x an")
x = int(x)

if x < 0:
  print("x ist größer 0")
else:
  print("x ist kleiner 0")
```

**Struktur `if-else` Statement**

`if` *Bedingung*`:`

␣␣ *Codezeile A1*

␣␣ *Codezeile A2*

␣␣ ⋮

`else:`

␣␣ *Codezeile B1*

␣␣ *Codezeile B2*

␣␣ ⋮

*Codezeile C1*

⋮

### Wie funktioniert's?

Ist die `if`-Bedingung `True`, so wird der `if`-*Block* ausgeführt. Ist sie `False` wird der `else`-*Block* ausgeführt.

**Definition: Block**

Aufeinanderfolgende Codezeilen, die alle die gleiche Einrückung besitzen, nennt man *Block*. D.h. Leerzeichen am Zeilenanfang haben in Python eine syntaktische Bedeutung.

### Good to know

- Der `else`-Block ist optional.
- Falls die Bedingung nicht vom Typ `bool` ist, so wird sie implizit umgewandelt.

# Übungen

**Volljährigkeit prüfen/Zutrittskontrolle**

Schreibe ein Skript, dass nach dem Alter eines Users fragt und überprüft, ob der User schon volljährig ist. Dementsprechend soll auf der Konsole entweder

```
Willkommen
```

oder

```
Du darfst hier nicht rein
```

erscheinen.

**Teilbarkeit bestimmen**

Schreibe ein Skript, dass eine ganze Zahl einliest. Daraufhin soll auf der Konsole ausgegeben werden, ob die Zahl durch 7 teilbar ist. Beispiel: Ist die Eingabe 12, so ist die Ausgabe:

```
Die Zahl 12 ist nicht durch 7 teilbar.
```

**Logische Operatoren**

Booleans können mittels folgender Operatoren miteinander verknüpft werden:

| | |
|---|---|
| `and` | Ist genau dann `True`, wenn beide Operanden `True` sind. |
| `or` | Ist genau dann `True`, wenn mindestens ein Operand `True` ist. |
| `not` | Kehrt den nachfolgenden Wahrheitswert um. |

**Beispiel**

- `2 > 0 and 3 > 4` ist `False`
- `1 > 0 or 6 > 1` ist `True`
- `not 2 < 1` ist `True`

# Übung

**Was ergeben die folgenden Ausdrücke?**

- `not 2 < 3 and 4 < 7`
- `4 not == 8`
- `3 != 4 and not 4 == 8`
- `7 <= 7.0 and not 7 != 7.0`
- `7 > 5 or 4 < 5 and not 9 > 6`
- `not 3 < 6 > 8`
- `not 3`

**Präzedenz beachten!**

1. ==, !=, <=, <, >, >=
2. not
3. and
4. or

### Das `elif`-Statement

Mit der reinen `if-else`-Syntax können nur *binäre* Verzweigungen dargestellt werden. Um mehrer, gleichrangige Verzweigungsäste zu realisieren kann man das `elif`-Conditional verwenden.

**Beispiel**

```
1 if x < 0:
2     print("x is < 0")
3 elif x == 0:
4     print("x is 0")
5 elif x == 1:
6     print("x is 1")
7 else:
8     print("x is not negative but neither 0 nor 1")
```

Die Anzahl der `elif`-Blöcke ist beliebig. Der `else`-Block ist wie immer optional.

**Worin unterscheiden sich die beiden Abschnitte?**

Abschnitt 1:

```
if x % 2 == 0:
    # some Code here
if x % 3 == 0:
    # some Code here
else:
    # some Code here
```

Abschnitt 2:

```
if x % 2 == 0:
# some Code here
elif x % 3 == 0:
# some Code here
else:
# some Code here
```

# Übung

**Baue einen Bestätigungsdialog**

Schreibe ein Skript was einen typischen Bestätigungsdialog simuliert. Zum Beispiel:

`Are you sure to continue? (y)es/(n)o.`

Mögliche Antworten sind `yes`, `no` bzw. `y`, `n`. Daraufhin soll auf der Konsole `confirmed` oder `aborted` erscheinen.

# Komplexere Übung


Lies eine Zahl zwischen 1 und 9 ein und gib auf der Konsole *deinen nächsten Urlaubsort* aus.

### Der *Ternary Operator*

Oftmals möchte man eine Variable in Abhängigkeit eines Wahrheitswertes definieren. Für diesen einfachen Fall, ist das `if-else`-Konstrukt sehr umständlich. Stattdessen kann man für die Kürze den *ternary operator* verwenden.

**Beispiel**

```
1 if x < 0:
2   sign = "negative"
3 else:
4   sign = "positive"
```

**Stattdessen mit Ternary Operator**

```
sign = "negative" if x < 0 else "positive"
```

# Übung

Lies eine ganze Zahl ein und gib ihren Betrag auf der Konsole aus. Schaffst Du es, das Ganze mit weniger als 5 Zeilen Code zu programmieren?