

Kontroll- strukturen

Foliensatz 2

Jasmin Noll
wi20086@lehre.dhbw-stuttgart.de

25.04.2022

Was euch heute erwartet

- if-statements
 - if
 - elif
 - else
 - Struktur
 - Kombinierte Bedingungen
 - Good-to-know
- for-loop
 - range()
 - continue
 - break
- While-loop

if-statements

25.04.2022

if

- Eine Bedingung überprüfen
- Zeile einleiten mit `if` und beenden mit `:`
 - Alle Codezeile die nach Bestätigung der Bedingungen ausgeführt werden sollen werden entsprechend eingerückt
 - Beispiel:

```
if a == b:  
    print("gleich groß")
```
- Wenn man booleasche Werte überprüft benötigt man keine Operatoren für `True` und `not` für `False`

- Beispiele:

- `if bool_value:`
 `print("bool_value =TRUE")`

- `if not bool_value:`
 `print("bool_value = FALSE")`

Methode	Beschreibung
<code>==</code>	gleich
<code>!=</code>	nicht gleich
<code><</code>	kleiner als
<code><=</code>	kleiner gleich
<code>></code>	größer als
<code>>=</code>	größer gleich
<code>in</code>	Überprüfen, ob etwas in z.B. einer Liste ist
<code>not in</code>	Überprüfen, ob etwas nicht in z.B: einer Liste ist

elif

- else-if
- optional
- Wenn die vorherigen Bedingungen nicht erfüllt wurden werden die elif-Bedingungen als nächstes betrachtet
- Syntax gleich zur if-Syntax
- Beispiele:
 - `elif a != b:`
 `print("Nicht gleich")`
 - `elif bool_value:`
 `print("TRUE")`

else

- optional
- Fängt alles ab, was nicht den vorherigen Bedingungen erfüllt
- Einleitung des *e/se*-Blocks mit `else`:
 - Benötigt keine Bedingung
- Beispiel:
`else:`
`print("Keine Bedingung erfüllt")`

Struktur

```
if num_value_1 > num_value_2:  
    print("Größer")  
elif num_value_1 == num_value_2:  
    print("Gleich")  
elif num_value_1 < num_value_2:  
    print("Kleiner")  
else:  
    print("Keine Bedingung erfüllt")
```

Erste Bedingung

Code, wenn erste Bedingung erfüllt wird

Zweite Bedingung wenn erste Bedingung nicht erfüllt

Code, wenn zweite Bedingung erfüllt wird

Dritte Bedingung wenn erste & zweite Bedingung nicht erfüllt

Code, wenn dritte Bedingung erfüllt wird

Keine vorherige Bedingung erfüllt

Code, wenn keine Bedingung erfüllt wird

Kombinierte Bedingungen

- Man kann Bedingungen in einer Zeile kombinieren
 - Verhindert verschachtelte IF-Statements
- Verbindung der Bedingungen mit **and** und/oder **or**
 - Bedingungen in Klammern setzen
- Beispiele:
 - `if (a > b) and (b < c):`
 `print("Bedingung erfüllt")`
 - `elif (a < b) or (a > b):`
 `print("Bedingung erfüllt")`

Good-to-know

- if-Statement in einer Zeile
 - Beispiel: `if a > b: print("Größer")`
- If-else-Statement in einer Zeile
 - Beispiel: `print("Größer") if a > b else print("Bedingung nicht erfüllt")`
- Mit pass kann ein Statement-Block übersprungen werden
 - Beispiel:
`if a > b:`
 `pass`

for-loop

25.04.2022

10

- Iterationen über Sequenzen (z.B. Listen oder Dictionary)
- Entsprechenden Code für jedes Element der Sequenz ausführen
- Zeile wird eingeleitet mit **for**, zwischen Element und Sequenz **in** und beendet mit **:**
 - Beispiel:

```
for element in sequenz:  
    print(element)
```
- Man kann auch zwei Werte nach **for** angeben, die pro Iterationen übergeben bekommen
 - Hilfreich bei z.B. Dictionaries
 - Beispiel:

```
for key, value in dictionary.items():  
    print(key, value)
```
- Mit einem **else**-Block kann man Code spezifizieren, der nach Beendigung des **for**-loops ausgeführt werden soll
 - Beispiel:

```
• for element in sequenz:  
    print(element)  
else:  
    print("done")
```

range()

- Statt z.B. einer Liste kann man range() als Sequenz verwenden
- Mit range() wird eine entsprechende Anzahl an Iterationen durchgeführt
 - Standardmäßig startet die Funktion bei 0, erhöht um 1 pro Iteration und endet bei der vorgegebenen Iteration - 1
 - Beispiel:
 - ```
for i in range(5):
 print(i)
```
- Der range()-Funktion kann man auch Startwert und um wie viel pro Iteration erhöht wird mitgeben
  - range(Startwert, Endwert (ausgeschlossen), Erhöhung)
  - Beispiel:
    - ```
for i in range(2, 20, 4):  
    print(i)
```

continue

- Mit `continue` kann die aktuelle Iteration abgebrochen werden und die nächste Iteration beginnt

- Alles vor `continue` wird ausgeführt
- Alles nach `continue` wird nicht mehr ausgeführt

- Beispiel:

```
for element in elements:  
    print(element)  
    if element == condition:  
        continue  
    print("Bedingung erfüllt")
```

break

- Mit **break** kann die Iteration abgebrochen werden bevor alle Elemente durchiteriert werden

- Alles vor dem **break** wird ausgeführt
- Alles nach dem **break** wird nicht mehr ausgeführt

- Beispiel:

```
for element in elements:  
    print(element)  
    if element == condition:  
        break  
    print("Bedingung erfüllt")
```

Pass kann ebenfalls verwendet werden

while-loop

- Über bestimmten Code iterieren, solange eine Bedingung erfüllt ist
- Zeile wird mit `while` eingeleitet und beendet mit :
 - Beispiele:
 - ```
while a < b:
 print(a)
 a += 1
```
- Eine while-loop kann ebenfalls mit booleschen-Werten verwendet werden
  - **An das Abbruchkriterium denken, sonst besteht die Gefahr zu einer Endlosschleife!**
  - Beispiele:
    - ```
while bool_value_true:  
    print(a)  
    a += 1  
    if a == 10:  
        bool_value_true == False
```
 - ```
while not bool_value_false:
 print(a)
 a += 1
 if a == 10:
 bool_value_false == True
```

***pass, break und continue können hier ebenfalls verwendet werden***



# Übungen

