Programmierung mit Python für Programmierer

Kapitel 2 - Kontrollstrukturen

Autor: Dr. Christian Heckler

Vorbemerkungen

- Verwendete Literatur: siehe Referenzen
- Verwendete Symbole:
  - ❗: Beispielprogramm
  - ❗: Weitere Erläuterungen im Kurs
  - ❓: Übung

Ein- und Ausgabe

- Eingabe:
  - Input-Funktion: `eingabe = input("Ihre Eingabe")`
  - Die Input-Funktion liefert immer eine Zeichenkette zurück.
  - Möchte man also eine Zahl eingeben, muss die Zeichenkette konvertiert werden:

```
zahl = float(input("Eingabe Zahl: " ))
```

- Ausgabe:
  - Print-Funktion
  - Ausgabe mehrere Objekte durch Komma getrennt
  - Automatische Umwandlung in Zeichenkette
  - Bsp.: `print("Das Ergebnis ist:", 42)`

Anweisungsblöcke

- Anweisungen werden in „Anweisungsblöcke“ strukturiert.
- Dies geschieht in Python durch Einrückung.
- Anweisung mit dergleichen Einrücktiefe gehören dabei zusammen und bilden einen Block
- Die Einrückungen müssen einheitlich mit Leerzeichen oder Tabulatoren erfolgen.
- Empfehlung:
  - Leerzeichen (Tabulatortaste mit Leerzeichen „belegen“)
  - Pro Stufe 4 Leerzeichen

Bedingte Anweisungen

Drei Formen:

- Einfache `if`-Anweisung:
  - Wenn eine Bedingung erfüllt ist, werden die folgenden Anweisungen (Anweisungsblock) ausgeführt.
  - Andernfalls nicht, und die Programmausführung geht hinter der `if`-Anweisung weiter
- `if … else`:
  - Je nach Bedingung wir der eine oder der andere Anweisungsblock ausgeführt
- `elif`:
  - Vereinfachte Schreibweise bei verschachtelten Bedingungen

Einfache if-Anweisungen

- Allgemeine Form

```
if bedingung:
    anweisungen
anweisungen
```

- Beispiel [Kle]:

```
alter = int(input("Dein Alter"))
if alter < 12:
    print("Zu jung")
    print("Waehle einen anderen Film")
print("Programmende")
```

If-Anweisung mit Else-Teil

- Allgemeine Form

```
if bedingung:
    anweisungen
else:
    anweisungen
anweisungen
```

- Beispiel [Kle]:

```
alter = int(input("Dein Alter"))
if alter < 12:
    print("Zu jung")
    print("Waehle einen anderen Film")
else:
    print("Viel Spass")
print("Programmende")
```

„Verschachtelte“ Blöcke

- In einem Anwendungsblock, der in einer If-Anweisung vorkommt, kann auch eine weitere If-Anweisung vorkommen.
- Das nennt man dann verschachtelte If-Anweisung.
- Bsp [Kle]:

```
alter = int(input("Dein Alter"))
if alter < 4:
    print("Der Film ist zu kompliziert")
else:
    if alter < 12:
        print("Viel Spass")
    else:
        if alter < 16:
            print("Bist Du Dir sicher?")
        else:
            print("Wollen Sie sich das antun?")
print("Programmende")
```

Elif

- Das ist im Zweifelsfall schwer zu lesen. Daher gibt es mit „elif“ eine Abkürzung [Kle]

```
alter = int(input("Dein Alter"))
if alter < 4:
    print("Der Film ist zu kompliziert")
elif alter < 12:
    print("Viel Spass")
elif alter < 16:
    print("Bist Du Dir sicher?")
else:
    print("Wollen Sie sich das antun?")
print("Programmende")
```

Kurzschreibweisen

- Den „ternärem Operator“ (`x = bedingung ? wert1 : wert2`) gibt es nicht. Stattdessen:

```
x = wert1 if bedingung else wert2
```

- Einzeilige Bedingungen:

```
if bedingung: anweisung
```

- Kein Switch. Statt dessen:
  - `elif` (insbesondere einzeilig, s. oben)
  - evtl. Verwendung von Dictionaries

Bedingungen

- Was genau steht hinter dem „if“?
- Ein „Ausdruck“, dessen Wert „wahr“ oder „falsch“ ist.
- Vergleichsoperatoren ℹ️:

Operator	**Bedeutung**	**Beispiel**
`==`	gleich	`42 == 42`
`!=`	ungleich	`42 != 43`
`<`	kleiner	`42 < 43`
`<=`	kleiner gleich	`42 <= 42`
`>`	größer	`43 > 42`
`>=`	größer gleich	`42 >= 42`

Bedingungen II

- Der Ausdruck hinter `if` wird automatisch nach bool konvertiert, z.B.

```
liste = []
if liste: (1)
```

(1) Statt: `if len(liste) == 0:`

- Zu `False` ausgewertet wird:
  - numerische Null-Werte
  - leere Zeichenketten
  - leere Listen, leere Tupel, leere Dictionaries
  - Der `None`-Wert

Logische Operatoren

- Wahrheitswerte lassen sich mit logischen Operatoren verknüpfen
- Diese sind: `and, or, not`
- Beispiel: Prüfung, ob eine Zahl a zwischen 20 und 100 liegt:

```
if 20 <= a and a <= 100:
```

- Prüfung des Gegenteils: ℹ
  - a liegt nicht zwischen 20 und 100:

```
if not (20 <= a and a <= 100):
```

  - a ist kleiner als 20 oder größer als 100

```
if a < 20 or a > 100:
```

- Wahrheitstabellen ℹ

Referenzen

- [Ste] Ralph Steyer: Programmierung Grundlagen, Herdt-Verlag
- [Kle] Bernd Klein: Einführung in Python 3, Hanser-Verlag
- [Kof] Kofler: Python – Der Grundkurs, Rheinwerk Computing
- [EK] Johannes Ernesti, Peter Kaiser: Python 3 – Das umfassende Handbuch, Rheinwerk Computing
- [Mat] Eric Matthes: Python Crashkurs – Eine praktische, projektbasierte Programmiereinführung, dpunkt.verlag
- [Swe] Sweigart: Eigene Spiele programmieren: Python lernen