Programmieren mit Python

Teil 2: IO und Verzweigungen

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

21. Oktober 2021

Operatoren

Die wichtigsten Operatoren

- `+` (Addition oder Zusammenkleben von Strings)
- `-` (Subtraktion)
- `*` (Multiplikation)
- `/` (Division, ergibt immer ein Wert vom Typ `float`)
- `**` (Potenzierung)
- `%` (*modulo-Operator*: Rest bei ganzzahliger Division)
- `//` (Division und Abrunden, ergibt immer ein Wert vom Typ `int`)
- `==` (Vergleichsoperator, ergibt immer ein Wert vom Typ `bool`)
- `!=` (Ungleichheitsoperator, ergibt das Gegenteil von `==`)

Operator-Präzedenz

1. Klammern
2. **
3. *, /, //, %
4. +,-

Operatoren gleichen Rangs werden innerhalb eines Ausdrucks von links nach rechts abgearbeitet.

Ausnahmen:
Potenzierung (**) und Zuweisung (=) werden von rechts nach links verarbeitet.

Kombinierte Zuweisung

Oft möchte man eine gegebene Variable neu zuweisen:

```
counter = 1
counter = counter + 1          # counter = 2
```

Dies lässt sich auch kurz schreiben als

```
counter = 1
counter += 1          # counter = 2
```

Analog sind die Operatoren `-=`, `*=`, `/=`, etc. definiert.

Von der REPL zum Quellcode

Script Mode

Sobald man mehrere zusammenhängende Zeilen hat, wird die Eingabekonsole (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

```
name = "Max"
age = 20
age = age + 1
```

Ausführung

Um diesen Code auszuführen, muss man bei Replit auf den Run-Button klicken oder alternativ den Shortcut Strg+Enter (Windows) bzw. Cmd+Enter (Mac) verwenden.

Achtung

Im Gegensatz zum REPL werden Ergebnisse von Rechnungen nicht mehr automatisch auf der Konsole ausgegeben.

Input/Output

Kommunikation über die Konsole

Die Konsole

Grafische Benutzeroberflächen sind zu Beginn relativ kompliziert, daher verwenden wir zunächst die *Python-Konsole* für die Kommunikation mit unserem Programm.

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()`:

```
age = input("How old are you?")
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

```
name = input("What is your name?")
age = input("What is your age?")
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.
```

Lösung

Adressabfrage

```
name = input("Dein Name: ")
age = input("Dein Alter: ")
street = input("Deine Addresse: ")
city = input("Deine Stadt: ")

print(f"Hallo {Name}, schön, dass Du da bist. Du bist {age} Jahre alt")
print(f"und wohnst in der {street} in {city}.")
```

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.

Lösung

```
age = input("Wie alt bist Du? ")
age = int(age) + 15
print(f"In 15 Jahren wirst Du {age} sein.")
```

Kommentare

Kommentare

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

Beispiel

```
print("This line will be printed")
# 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.

Übung

Antwort überprüfen

Schreib ein Programm, dass folgende Frage auf der Konsole ausgibt und die Antwort einliest.

```
Was ist die Hauptstadt von Frankreich?
```

Darauf hin soll entsprechend der Antwort folgendes Feedback auf der Konsole erscheinen:

```
Das war richtig!
```

bzw.

```
Das war falsch! Die richtige Antwort ist Paris.
```

Lösung

Antwort überprüfen

```
answer = input("Was ist die Hauptstadt von Frankreich?")
if answer == "Paris"
  print("Das ist richtig!")
else
  print("Das war falsch! Die richtige Antwort ist Paris.")
```

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 folgendes Feedback erscheinen:

```
Willkommen
```

bzw.

```
Du darfst hier nicht rein
```

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.
```

Lösungen

Zutrittskontrolle

```
age = input("Wie alt bist Du? ")
age = int(age)

if age >= 18:
  print("Willkommen!")
else:
  print("Du darfst hier nicht rein!")
```

Teilbarkeit bestimmen

```
x = input("Gib eine Zahl ein: ")
x = int(x)

if x % 7 == 0:
print(f"Die Zahl {x} ist durch 7 teilbar")
else:
print(f"Die Zahl {x} 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

```
if x < 0:
    print("x is < 0")
elif x == 0:
    print("x is 0")
elif x == 1:
    print("x is 1")
else:
    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
```

Komplexere Übung

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

Lösung

Urlaubsort

```
number = input("Gib eine Zahl zwischen 1 und 9 ein: ")
number = int(number)

number = number * 3
number = number + 3
number = number * 3

cross_sum = number // 10 + number % 10
print("Dort verbringst Du Deinen Urlaub: ")
if cross_sum == 1:
  print("Italien")
elif cross_sum == 2:
  print("Spanien")
# ... more elif statements ...
elif cross_sum == 9:
  print("Zu Hause")
else:
  print("USA")
```

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

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

Stattdessen mit Ternary Operator

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

Übung

Ternary Operator

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?

Lösung

```
x = input("Gib eine Zahl ein: ")
x = float(x)
abs_value = x if x >= 0 else -x
print(f"Der Betrag von {x} ist {abs_value}")
```