LU06c - For-Schleife

Lernziele:

- Sie wissen, wie man die for-Schleife verwendet.
- Sie erkennen Situationen, in denen eine while-Schleife verwendet werden sollte und solche, in denen eine for-Schleife besser geeignet ist.

Einführung

Im letzen Kapitel haben wir gelernt, wie eine while-Schleife mit einer Bedingung verwendet werden kann, um Zahlen in einem bestimmten Intervall zu durchlaufen.

Die Struktur dieser Art von Schleife ist die folgende.

```
i = 0
while (i < 10):
    print(i)
    i += 1
```

Die obige Schleife kann in drei Teile aufgeteilt werden. Zunächst führen wir die Variable `i` ein, die dazu dient, zu zählen, wie oft die Schleife bisher ausgeführt wurde, und setzen ihren Wert auf 0: `i = 0`. Danach folgt die Definition der Schleife - die Bedingung der Schleife ist `i < 10`, so dass die Schleife ausgeführt wird, solange der Wert der Variablen `i` kleiner als 10 ist. Der Schleifenkörper enthält die auszuführende Funktionalität `print(i)`, woraufhin der Wert der Variablen `i += 1` erhöht wird. Der Befehl `i += 1` ist die Kurzform für `i = i + 1`.

Dasselbe lässt sich mit einer for-Schleife wie der folgenden erreichen.

```
for i in range(1,10):
    print(i)
```

Eine for-Schleife besteht aus drei Teilen:

- die Einführung der Indexvariablen i zum Zählen der Anzahl der Durchläufe
- die Bedingung der Schleife
- die auszuführende Funktionalität.

```
for target_list in expression_list:
    # Functionality to be executed
```

Typische Anwendungen

Range()

Um einen Codeblock bestimmte Anzahl von Malen zu durchlaufen, können wir die Funktion `range()` verwenden, Die `range()`-Funktion gibt eine Folge von Zahlen zurück, die standardmäßig bei 0 beginnt, um 1 erhöht wird und bei einer bestimmten Zahl endet.

```
for x in range(6):
  print(x)

0
1
2
3
4
5
```

Beachten Sie, dass der `range(6)` nicht die Werte von `0` bis `6`, sondern die Werte `0` bis `5` umfasst.

Die Funktion `range()` ist standardmäßig auf `0` als Startwert eingestellt. Es ist jedoch möglich, den Startwert durch Hinzufügen eines Parameters zu spezifizieren: `range(2, 6)`, d. h. Werte von 2 bis 6 (aber ohne 6):

```
for x in range(2, 6):
  print(x)

2
3
4
5
```

Die Funktion `range()` erhöht die Sequenz standardmäßig um 1, es ist jedoch möglich, den Inkrementwert durch Hinzufügen eines dritten Parameters anzugeben: `range(2, 30, 3)`:

```
for x in range(2, 30, 3):
  print(x)


2
5
8
11
14
17
20
23
26
29
```

Iterieren durch Strings

Auch Strings sind iterierbare Objekte, sie enthalten eine Folge von Zeichen:

```
for x in 'banana':
  print(x)

b
a
n
a
n
a
```

M319-F3G, M319-F3E