Merkblatt - Schleifen

May 7, 2020

1 Merkblatt: Schleifen

Schleifen erlauben es dir, dafür zu sorgen, dass bestimmte Teile vom Programm mehrfach ausgeführt werden. Das ist praktisch, wenn du z.B. eine Liste von Studierenden hast und dafür sorgen möchtest, dass z. B. ein print() - Befehl einmal für jeden Studierenden ausgeführt wird.

Eine Schleife erlaubt dir, sowas in recht wenig Code auszudrücken!

1.1 Die for-Schleife

Neben der while-Schleife, die wir schon kennen gelernt haben, gibt es noch die **for-Schleife**. Hier durchläuft eine Schleifenvariable nacheinander die Werte in einer ebenfalls anzugebenden Sequenz.

Bei dieser Sequenz kann es sich z. B. um eine Liste handeln:

[3]:
```
liste = [5, 8, 10]
for i in liste:
    print(i)
```

```
5
8
10
```

[5]:
```
liste = ["Max", "Moritz", "Monika"]
for i in liste:
    print(i)
```

```
Max
Moritz
Monika
```

Wir sehen, dass unsere Schleifenvariable i nacheinander und automatisch die Werte aus der Liste annimmt.

1.1.1 Das range-Objekt

Als Sequenz für eine for-Schleife braucht man nicht zwangsläufig eine Liste. Häufig greift man stattdessen auf ein **range**-Objekt zurück:

[1]:
```
print(range(0,10))
```

```
range(0, 10)
```

[2]:
```
for i in range(0, 10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

[8]:
```
# Hier summieren wir alle Zahlen von 1 bis 10 mithilfe einer for-Schleife und␣
↪eines range-Objektes auf
sum = 0
for i in range(1, 11):
    sum += i
print(sum)
```

```
55
```

1.2 Die while-Schleife

Ein Code-Block innerhalb einer if-elif-else-Struktur wird jeweils nur einmal ausgeführt. Bei Schleifen wie der **while-Schleife** wird ein Code-Block so lange mehrmals hintereinander ausgeführt, bis eine Abbruchbedingung erfüllt ist:

[1]:
```
counter = 0

while counter < 10:
    print(counter)
    counter = counter + 1

print("Hallo Welt")
```

```
0
1
2
3
4
5
6
7
8
```

```
9
Hallo Welt
```

Innerhalb einer Schleife **muss** sich **unbedingt** ein Zustand in jedem Schritt verändern, damit die Schleifenbedingung nicht dauerhaft erfüllt ist, und das Programm die Schleife auch wieder verlassen kann.

[3]:
```
students = ["Moritz", "Klara", "Monika", "Max"]
i = 0
while i < len(students):
    print(students[i])
    i = i + 1
```

```
Moritz
Klara
Monika
Max
```

2 Continue & Break

Wir können während eines Schleifendurchlaufs den aktuellen Durchlauf vorzeitig abbrechen und unmittelbar mit dem nächsten Schleifendurchlauf fortfahren (**continue**) oder auch die gesamte Schleife abbrechen (**break**).

2.0.1 Continue

Wir brauchen einfach das Wort **continue** in eine Schleife zu schreiben, wenn an einer bestimmten Stelle zum neuen Schleifendurchlauf gesprungen werden soll:

[5]:
```
for i in range(0, 10):
    if i == 3:
        continue
    print(i)
```

```
0
1
2
4
5
6
7
8
9
```

In dem obigen Beispiel wird für den Wert 3 der print()-Befehl übersprungen.

[3]:
```
for i in range(1, 10):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
```

2.0.2 Break

Auch **break** schreiben wir einfach in eine Zeile, und schon wird die ganze Schleife abgebrochen, wenn das Programm diese Stelle erreicht:

[4]:
```
for i in range(0, 10):
    if i == 3:
        break
    print(i)
```

```
0
1
2
```

[6]:
```
liste = [4, 6, 7, 2, 4, 6, 7]

s = 0

for element in liste:
    s = s + element
    if s > 10:
        break

print(s)
```

```
17
```