# L1\_3 Die while-Schleife

Die „while“–Schleife löst Problemstellungen, bei denen ein Vorgang solange wiederholt wird, bis eine Bedingung nicht mehr erfüllt ist.

Ein Beispiel wäre die Problemstellung: „Erhöhe den Wert der Variablen *anzahl* um 1 und gebe sie aus, solange sie den Wert 10 nicht erreicht hat“.

## **1 Vergleichsoperatoren**

Vergleichsoperatoren werden immer zum Vergleich zweier Werte benutzt und finden ihre Anwendung zumeist beim Einsatz von Wiederholungen oder Alternativen. Als Ergebnis des Vergleichs wird ein boolescher Wert (true / false) zurückgegeben. Meist wird eine Variable mit einem festen Wert oder eine Variable mit einer anderen Variablen verglichen. Je nach Ergebnis des Vergleichs wird das Pro­gramm eine andere Reaktion zeigen.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Beispiel** | **Beschreibung** |
| < | a < b | a ist kleiner als b |
| <= | a <= b | a ist kleiner oder gleich b |
| > | a > b | a ist größer als b |
| >= | a >= b | a ist größer oder gleich b |
| == | a == b | a ist gleich b |
| != | a != b | a ist ungleich b |

## **2 Logische Operatoren in Python**

Der Käfer *kara* kann durch die Nutzung seiner Sensoren auf Ereignisse reagieren. Dem Käfer *kara* ist es auch möglich, auf mehrere Sensoren gleichzeitig zu reagieren. Hier eine kurze Übersicht über die Verwendung von Operatoren.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Beispiel** | **Beschreibung** | **Erläuterung** |
| and | kara.treeFront() **and**  kara.treeRight() | und | Die Bedingung ist erfüllt (true), wenn beide Aussagen zutreffen, d.h. wenn der Käfer *kara* einen Baum vor und rechts von sich hat. |
| or | kara.treeFront() **or**  kara.treeRight() | oder | Die Bedingung ist erfüllt (true), wenn die eine **oder** die andere **oder** beide Aussage/n erfüllt sind. |
| not | **not** kara.treeFront() | nicht | Das Ausrufezeichen ändert einen Ausdruck von true in false und umgekehrt. Hier ist die Bedingung erfüllt, wenn der Käfer *kara* **nicht** vor einem Baum steht. |

## **3 Aufbau einer while-Schleife**

Die while-Schleife benötigt zuerst eine Bedingung, die dafür sorgt, dass die Schleife durchlaufen wird oder nicht. Diese wird im Schleifenkopf hinter dem Schlüsselwort *while* in Klammern angegeben.

Beispiel: while anzahl < 10

Danach folgt der Schleifenkörper. Hier stehen die Anweisungen, die abgearbeitet werden sollen, falls es zu einem Schleifendurchlauf kommt.

Im Schleifenkörper stehen die Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden.

## **4 Das Struktogramm einer while-Schleife**

|  |  |
| --- | --- |
|  | Wiederholungsstruktur mit vorausgehender Bedingungsprüfung. Der Schleifenkörper (Anweisungsblock) wird nur durchlaufen, solange die Bedingung erfüllt (wahr) ist. |

## **5 Syntax einer while-Schleife**



Beachte: Die Anweisungen, die wiederholt ausgeführt werden sollen, müssen eingerückt eingegeben werden.

Im **Schleifenkopf** wird die Bedingung festgelegt.

Der **Schleifenkörper** enthält die Anweisungen, die bei jeder Wiederholung ausgeführt werden.

Im dargestellten Beispiel wird im Schleifenkopf geprüft, ob der Wert der Variablen *anzahl* < 10 ist.

Ist dies der Fall, wird der Schleifenkörper abgearbeitet:

- Der Käfer *kara* geht einen Schritt vorwärts.

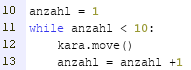
- Der Wert der Variablen *anzahl* wird um 1 erhöht.

Danach wird wieder geprüft, ob der Wert der Variablen *anzahl* immer noch kleiner als 10 ist …

## **6 Realisierung der Problemstellung**

6.1 Mit der Schleifenbedingung *while* *anzahl* < 10 kann der Käfer *kara* vom linken Rand einer 10 x 10 Felder großen Welt an den rechten Rand bewegt werden.

Damit die Bedingung *anzahl* < 10 einen Wert (true oder false) zurückliefern kann, benötigt die Variable *anzahl* einen Anfangswert. Dieser muss vor dem Schleifenbeginn festgelegt werden:



6.2 Mit der Schleifenbedingung *while not kara.treeFront()*   
kann der Käfer *kara* so lange Anweisungen ausführen,   
bis er vor einem Baum steht.



