Grundlagen zur Programmierung des micro:bit in Python VI

Wiederholungen (aus Schleifen)

Du kennst dich nun schon sehr gut mit Verzweigungen und Variablen aus und hast auch viele lustige Dinge damit gemacht. Es gibt aber noch mehr: Wenn wir z.B. einen Codeblock 5x wiederholen wollen ehe etwas anderes passiert (z.B. 5x blinken bevor das Programm startet) und diesen Block auch wirklich fünfmal hinschreiben, dann ist das ziemlich unpraktisch. Um gleichen Code nicht ständig untereinanderschreiben zu müssen gibt es **Wiederholungsbefehle**, die ihren Inhalt mehrmals wiederholen – entweder bis ein bestimmtes Ereignis eintritt (**while**) oder du gibst eine fixe Anzahl vor (**for**). Diese Befehle nennt man in der Informatik „**Schleifen**“. Auch die „**on_forever()“-Methode** ist eine **Schleife**, die durchgehend wiederholt wird.

Um **Schleifen** zu programmieren, benötigen wir folgende Befehle aus 

:

Block	Beschreibung

for
`for i in range(4):`
`  pass`
	wiederholt die Codezeilen, die darunter eingerückt stehen, so lange, bis die Variable ‚**i**‘ von 0 startend, den letzten Wert (**4**) erreicht hat. **Es wird jedoch beim letzten Mal nicht ausgeführt!** Somit wird die Schleife nur vier Mal (**i**=0,1,2,3) durchlaufen.
	nach jeder Ausführung des Codes (ersetzt „pass“), wird ‚**i**‘ automatisch um 1 erhöht. Man kann ‚**i**‘ auch im Code verwenden.
	Diese Schleife nennt man **FOR-Schleife.**

while
`while True:`
`  pass`
	wiederholt die Blöcke, die innerhalb stehen, solange das Bedingung ‚wahr‘ ist (wie bei den Verzweigungen, hier wieder True, also immer wahr);
	Wenn die Antwort ‚falsch‘ ist, werden die Wiederholungen beendet.
	Diese Schleife nennt man **WHILE-Schleife.**

For-Schleife

Bei dieser Schleifenart gibt es einen Startwert (0 bei range(4), 3 bei range(3,10)) und einen Endwert (bei den vorherigen Beispielen 4 bzw. 10) welche den Wert von ‚**i**' am Ende bestimmen. **Wichtig! Die Schleife wird jedoch nicht mehr durchlaufen, wenn ‚i' diesen Wert erreicht hat!** Zuerst hat die Variable ‚**i**' den Wert 0, nach jeder Ausführung des inneren Codes, wird diese automatisch um 1 geändert. ‚**i**' hat also die Werte 0, 1, 2 usw. – bis zur angegebenen Zahl. Nach der letzten Wiederholung ist die Variable ‚**i**' im unteren Beispiel 8. Es gibt hier insgesamt 8 Wiederholungen (0-7).

Mit der For-Schleife können wir ein Programm, wo Rechtecke 9-mal blinken so programmieren, dass jedes Mal auch die aktuelle Wiederholungszahl ausgegeben wird. Man weiß, also wie oft das Rechteck noch blinken wird.

```
for i in range(8):
    basic.show_string("i")
    basic.show_leds("""
    # # # # #
    # . . . #
    # . . . #
    # . . . #
    # # # # #
    """)
    basic.show_leds("""
    . . . . .
    . # # # .
    . # # # .
    . # # # .
    . . . . .
    """)
```

While-Schleife

Die While-Schleife könnte man auch als „so lange bis" Schleife verstehen. Es wird also der Code innerhalb so lange ausgeführt, bis die Bedingung ‚falsch' ist.

Mit der While-Schleife kann man den micro:bit z.B. am Beginn so lange blinken lassen, bis er geschüttelt wird. Wird er geschüttelt, wird das Blinken abgebrochen und das „normale" Programm (im dauerhaft) startet. Damit könnte man z.B. ein Spiel erst mit Schütteln starten – nicht sofort, wenn der micro:bit Strom bekommt.

```
start=0
```

Wir benötigen eine Variable (hier heißt sie „start“). Am Beginn setzen wir diese auf 0.

```
def on_gesture_shake():
    global start
    start += 1
input.on_gesture(Gesture.Shake, on_gesture_shake)
```

Wird der micro:bit geschüttelt, ändert sich „Start“ von 0 auf 1.

```
while start==0:
    basic.show_leds("""
    # # # # #
    # . . . #
    # . . . #
    # . . . #
    # # # # #
    """)
    basic.show_leds("""
    . . . . .
    . # # # .
    . # # # .
    . # # # .
    . . . . .
    """)
```

Überprüfe nun mit dem ‚Frage-Kästchen‘, ob „Start“ den Wert 0 hat. Und solange das gilt, wiederhole den Code innerhalb der Schleife.

Wenn sich „Start“ auf 1 ändert, wird die Schleife abgebrochen.

WICHTIG:
Wenn man eine Schleife im Code hat, dessen Abbruch jedoch nicht in der Schleife erfolgt, sondern durch Änderung von außerhalb, dann muss man den Abbruch vor der Schleife programmieren, da sonst das Programm nicht aus der Schleife „rauskommt“!!

Zuerst Abbruchbedingung (schütteln), dann Schleife (while)!

Du kannst nun neue Arten von Programmen schreiben, schaue dir dazu das zugehörige **Arbeitsblatt** an! Vergiss nicht dein Programm im Anschluss herunterzuladen und auf den micro:bit zu verschieben um es auszuprobieren!