Grundlagen zur Programmierung des micro:bit in Python V

Verzweigung II (aus Logik)

Wir haben bisher nun schon einiges gelernt und können auch schon einige Programme schreiben! Jetzt hast du die Grundlagen schon drauf! Als letztes haben wir uns die Variablen, also wie man Werte speichern und wieder auslesen kann, angesehen. Das ermöglicht uns bei den Fallunterscheidungen ganz neue Wege - wir können nämlich nun mehr als zwei Fälle unterscheiden! Wie, das liest du in diesem Kapitel.

Um Fallunterscheidungen mit mehr als zwei Fällen zu programmieren, benötigen wir nur einen neuen Begriff – man kann mit dem Befehl elif (kurz für „else if“) weitere Fälle hinzufügen:

```
if True:
    #Code
else:
    #Code
```

```
if True:
    #Code
elif *andere Bedingung*:
    #Code
else:
    #Code
```

Man sieht: Es gibt nun drei Fälle – für noch weitere, würde man erneut ein elif formulieren. Was wir auch sehen ist, dass wir bei drei Fällen zwei Bedingungen benötigen. Nur, wenn die erste Bedingung (hier jetzt True, also immer wahr) falsch ist, wird die zweite überprüft. Wurden beide Bedingungen überprüft und sind beide falsch, wird der Code im letzten Blockteil (‚else‘) ausgeführt.

Wir könnten das Orakelbeispiel von früher bei den einfachen Verzweigungen also nun erweitern – und zusätzlich zu Ja (Häkchen) und Nein (X) einen Fall mit ‚Weiß nicht‘ ergänzen. Wir hätten also dann drei Fälle, das heißt, die gewählte Zufallszahl müsste drei Werte annehmen (einen für jeden der drei Fälle). Wir wählen also zuerst eine Zufallszahl zwischen 0 und 2 aus und fragen danach zuerst nach dem ersten Fall (ob die Zahl = 1 ist), tritt dieser nicht ein, dann nach dem zweiten (ob die Zahl = 2 ist). Tritt weder Fall eins noch Fall zwei ein, so ist es Fall drei – dazu muss nichts mehr abgefragt werden (keine dritte Frage), da ja nur mehr ein Fall in Frage kommt.

Der Code des Orakels mit zwei Fällen (aus Kapitel III) sah so aus: (siehe nächste Seite)

Code des Orakels.

```
def on_forever():
    basic.show_leds("""
        . # # # .
        . . . # .
        . . # . .
        . . . . .
        . . # . .
        """)
basic.forever(on_forever)

def on_button_pressed_ab():
    if randint(0, 1) == 1:
        basic.show_icon(IconNames.YES)
    else:
        basic.show_icon(IconNames.NO)
input.on_button_pressed(Button.AB, on_button_pressed_ab)
```

Fügt man nun einen **weiteren Fall** hinzu und vergrößert den Zufallszahlbereich um eins, sieht der Code so aus:

```
def on_forever():
    basic.show_leds("""
        . # # # .
        . . . # .
        . . # . .
        . . . . .
        . . # . .
        """)
basic.forever(on_forever)

def on_button_pressed_ab():
    antwort=randint(0, 2)
    if  antwort == 0:
        basic.show_icon(IconNames.YES)
    elif antwort == 1:
        basic.show_icon(IconNames.NO)
    else:
        basic.show_icon(IconNames.SILLY)
input.on_button_pressed(Button.AB, on_button_pressed_ab)
```

Die Zufallszahl muss erst in eine Variable gespeichert werden.

Dadurch kann sie in den beiden Bedingungen verwendet werden.

Du kannst nun echt coole Programme 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!