micro:bit Programmierung in Python – Lösungen zu Kapitel V

Grundlagen-Blöcke *(Schwierigkeit: ☺)*

Versuche selbst ein eigenes Programm zu schreiben und am micro:bit auszuführen! Hast du eigene Ideen, was du mit den vorgestellten Blöcken machen kannst? Super, versuche dich gleich daran!

Ideen wären weiters:

Arbeitsauftrag	Lösung
Probiere aus, was passiert, wenn man zwei oder sogar drei basic.show_icon() in die ‚on_forever()'-Methode schiebt!	def on_forever(): basic.show_icon(IconNames.HAPPY) basic.show_icon(IconNames.SAD) basic.show_icon(IconNames.CONFUSED) basic.forever(on_forever)
Schreibe eine eigene Begrüßung, die beim Starten (sobald Strom da ist) am micro:bit angezeigt wird.	basic.show_string("Ich bin ein micro:bit")
Erstelle ein eigenes Symbol (z.B. einen Pfeil) und zeige dieses mit den LEDs an.	def on_forever(): basic.show_leds(""" . . # . . . # # # . # # # # # . . # . . . . # . . """) basic.forever(on_forever)
„Flashing heart": Lasse ein Herz blinken und zwar so, dass abwechselnd immer ein großes und ein kleines Herz angezeigt wird.	def on_forever(): basic.show_icon(IconNames.HEART) basic.show_icon(IconNames.SMALL_HEART) basic.forever(on_forever)

Vergiss nicht dein Programm im Anschluss herunterzuladen und auf den micro:bit zu verschieben um es auszuprobieren!

Ereignissteuerung *(Schwierigkeit: ☺)*

Hast du auch für die neu vorgestellten Blöcke Ideen, die du ausprobieren willst? Dann los!

Was du sonst noch probieren kannst:

Arbeitsauftrag	Lösung
Namensausgabe: Beim Drücken der Taste A gib deinen Vornamen, beim Drücken der Taste B deinen Nachnamen aus.	def on_button_pressed_a(): basic.show_string("Informatik") input.on_button_pressed(Button.A, on_button_pressed_a) def on_button_pressed_b(): basic.show_string("Werkstatt") input.on_button_pressed(Button.B, on_button_pressed_b)
Symbolgeber: Beim Drücken der Taste A soll ein erfreuter Smiley, bei B ein trauriger Smiley und beim Drücken beider Tasten (A+B) soll ein Herz aufleuchten.	def on_button_pressed_a(): basic.show_icon(IconNames.HAPPY) input.on_button_pressed(Button.A, on_button_pressed_a) def on_button_pressed_b(): basic.show_icon(IconNames.SAD) input.on_button_pressed(Button.B, on_button_pressed_b) def on_button_pressed_ab(): basic.show_icon(IconNames.HEART) input.on_button_pressed(Button.AB, on_button_pressed_ab)
Kann man das Programm so erweitern, dass wenn keine Taste gedrückt wurde, nichts aufleuchtet (alle LEDs aus sind)? *Tipp: Verwende die basic.show_leds()-Methode, mit der man eigene Muster zeichnen kann.*	def on_forever(): basic.show_leds(""" . . . . """) basic.forever(on_forever) def on_button_pressed_a(): basic.show_icon(IconNames.HAPPY) input.on_button_pressed(Button.A, on_button_pressed_a)

	def on_button_pressed_b(): basic.show_icon(IconNames.SAD) input.on_button_pressed(Button.B, on_button_pressed_b) def on_button_pressed_ab(): basic.show_icon(IconNames.HEART) input.on_button_pressed(Button.AB, on_button_pressed_ab)
Gegenteil-Schüttler: Wird der micro:bit geschüttelt, so soll er z.B. das Symbol anzeigen, ansonsten (das Gegenteil – genau jene LEDs, die vorher nicht geleuchtet haben, leuchten dann).	def on_forever(): basic.show_leds(""" . . # . . . # # # . # # # # # . # # # . . . # . . """) basic.forever(on_forever) def on_gesture_shake(): basic.show_leds(""" # # . # # # . . . # . . . . . # . . . # # # . # # """) input.on_gesture(Gesture.Shake, on_gesture_shake)
Pfeilneigung: Wird der micro:bit nach rechts geneigt, soll ein rechtszeigender Pfeil (→) angezeigt werden und gleiches für links (linkszeigender Pfeil ← bei Linksneigung).	def on_gesture_tiltLeft(): basic.show_leds(""" . . # . . . # # . . # # # # # . # # . . . . # . . """) input.on_gesture(Gesture.TILT_LEFT, on_gesture_tiltLeft)

	def on_gesture_tiltRight(): basic.show_leds(""" . . # . . . . # # . # # # # # . . # # . . . # . . """) input.on_gesture(Gesture.TILT_RIGHT, on_gesture_tiltRight)
Kopfstand: Dreht man den micro:bit auf den Kopf, so soll er einen umgedrehten Smiley anzeigen, steht er normal (Augen oben), so soll ein normaler Smiley angezeigt werden.	def on_gesture_up(): basic.show_icon(IconNames.HAPPY) input.on_gesture(Gesture.LOGO_UP, on_gesture_up) def on_gesture_down(): basic.show_leds(""" . # # # . # . . . # . . . . . . # . # . . . . . . """) input.on_gesture(Gesture.LOGO_DOWN, on_gesture_down)
„Der freundliche micro:bit“: Wird der micro:bit nach rechts geneigt, soll er deinen rechten Sitznachbarn begrüßen (z.B.. „Hallo Alex!“), nach links geneigt den linken Sitznachbarn. Drückst du A+B, soll der micro:bit dich begrüßen.	def on_gesture_tiltLeft(): basic.show_string("Hallo Max!") input.on_gesture(Gesture.TILT_LEFT, on_gesture_tiltLeft) def on_gesture_tiltRight(): basic.show_string("Hallo Nina!") input.on_gesture(Gesture.TILT_RIGHT, on_gesture_tiltRight) def on_button_pressed_ab(): basic.show_string("Hallo Alex!") input.on_button_pressed(Button.AB, on_button_pressed_ab)

Zufall *(Schwierigkeit: ☺☾)*

Im Beispiel wurde ein Würfel mit Zufallszahlen programmiert. Was könnte man sonst noch damit machen? Man könnte den micro:bit folgende Dinge tun lassen:

Arbeitsauftrag	Lösung
Erstelle ein Orakel, das bei Schütteln mittels Zufalls anzeigt, wie viele Geschenke man beim nächsten Geburtstag bekommen wird. Überlege zuerst zwischen welchen Werten der Zufall auswählen soll und was Sinn machen würde.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) basic.forever(on_forever) def on_gesture_shake(): basic.show_number(randint(1, 10)) input.on_gesture(Gesture.Shake, on_gesture_shake)
Spiel: Wer ist besser im Raten? Programmiere den micro:bit so, dass er bei Drücken der Taste A eine zufällige Zahl anzeigt. Du kannst dann vorm Drücken versuchen die Zahl zu erraten. Vielleicht spielt dein Freund oder deine Freundin mit und ihr schaut, wer näher an der Zahl ist? *Fragen zum Nachdenken: Bei welchen Zufallszahl-Bereichen ist das Spiel leicht, bei welchen schwer? Woran liegt das?*	def on_button_pressed_a(): basic.show_number(randint(1, 10)) input.on_button_pressed(Button.A, on_button_pressed_a)
Kann man mit den bisher bekannten Blöcken eine Losziehung machen? Das heißt, man verteilt Lose in der Klasse mit verschiedenen Losnummern (zum Beispiel Durchzählen oder Katalognummern) und möchte dann einen Gewinner oder eine Gewinnerin ziehen. Zwischen welchen Zahlen muss der Zufall hier auswählen?	def on_button_pressed_a(): basic.show_number(randint(1, 25)) input.on_button_pressed(Button.A, on_button_pressed_a)
Spiel - Orakel: Programmiere den micro:bit als Ja-Nein-Sager: Wenn A+B gedrückt wird, so soll zufällig 0 oder 1	def on_forever(): basic.show_leds("""

angezeigt werden (0 für Nein, 1 für Ja). Wird nichts gedrückt, so soll ein Fragezeichen angezeigt werden. Man stellt dem micro:bit also dann eine Frage (z.B. Soll ich heute noch ein Eis essen?), drückt A+B – und hat die Antwort.	.###. ...#. ..##. ....#.. """) basic.forever(on_forever) def on_button_pressed_ab(): basic.show_number(randint(0, 1)) input.on_button_pressed(Button.AB, on_button_pressed_ab)
Anweisungsgenerator: Schreibe eine Liste mit Aktivitäten von 1 bis 9, zum Beispiel: 1 --> Aufstehen 2 --> Hände in die Höhe 3 --> rechte Hand hoch usw. und programmiere den micro:bit nun so, dass er beim Drücken der Taste A dir und deinen Freunden und Freundinnen eine Anweisung gibt (z.B. „2“, also Hände hoch). Wer am langsamsten ist, hat verloren und scheidet aus – oder jedes Kind hat 2 Leben und scheidet aus, sobald es keine mehr hat.	def on_button_pressed_a(): basic.show_number(randint(1,9)) input.on_button_pressed(Button.A, on_button_pressed_a)

Verzweigung *(Schwierigkeit: ☺☺)*

Denke dir unbedingt das Beispiel mit dem Mini-Orakel gut durch und versuche zu verstehen, was dort passiert. Lies dir auch die Beschreibung (unten) nochmals gut durch. Danach kannst du dich gerne an eigenen Ideen probieren – oder du versuchst dich an untenstehenden Aufgaben!

```
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)
```

Zeige ein Fragezeichen, solange nichts Anderes (A+B-Knopf gedrückt) passiert.

Wird A+B am micro:bit gedrückt, mache den Code darunter.

Wähle eine zufällige Zahl zwischen 0 und 1 aus.

Überprüfe, ob die gewählte Zufallszahl gleich 1 ist.

Wenn die **Zufallszahl = 1** (die Frage ist wahr), dann zeige das Häkchen an.

Ist die Zufallszahl **nicht** = 1 (also hier 0), dann zeige das X an.

Arbeitsauftrag	Lösung
Ändere das Orakel aus dem Beispiel darüber so, dass das Wort ‚Ja' angezeigt wird, wenn die Zufallszahl gleich 1 ist, und im anderen Fall soll das Wort ‚Nein' angezeigt werden.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # . . . . # . . """) basic.forever(on_forever) def on_button_pressed_ab(): if randint(0, 1) == 1: basic.show_string("JA") else: basic.show_string("NEIN") input.on_button_pressed(Button.AB, on_button_pressed_ab)

Was passiert bei folgendem Code? Welche Zufallswerte sind möglich und bei welchen wird ein Häkchen/ein X angezeigt? Überlege und notiere deine Vermutung.

Code	Mögliche Zufallszahlen	Zahlen, wo Häkchen gezeigt	Zahlen, wo X gezeigt
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)	0, 1	1	0
def on_button_pressed_ab(): if randint(0, 2) == 1: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	0, 1, 2	1	0,2
def on_button_pressed_ab(): if randint(1, 2) == 1: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1, 2	1	2
def on_button_pressed_ab(): if randint(0, 1) == 0: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	0, 1	0	1

def on_button_pressed_ab(): if randint(1, 1) == 0: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1	1	nie
def on_button_pressed_ab(): if randint(0, 2) == 0: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	0, 1, 2	0	1,2
def on_button_pressed_ab(): if randint(0, 5) == 3: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	0 - 5	3	1,2, 4,5
def on_button_pressed_ab(): if randint(0, 5) == 7: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	0 - 5	nie	0-5
def on_button_pressed_ab(): if randint(1, 3) < 2: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1, 2, 3	1	2,3

def on_button_pressed_ab(): if randint(1, 3) < 3: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1, 2, 3	1,2	3
def on_button_pressed_ab(): if randint(1, 10) < 5: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1 - 10	1-4	5-10
def on_button_pressed_ab(): if randint(1, 10) > 7: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1 - 10	8-10	1-7
def on_button_pressed_ab(): if randint(1, 10) >= 7: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)	1 - 10	7-10	1-6

Gewinnspiel: Erstelle mit dem micro:bit einen Würfel, bei dem man eine zufällige Zahl zwischen 1 und 6 würfelt. Bei der Zahl 6 soll der Text ‚Gewonnen!‘ (oder ein glücklicher Smiley) erscheinen, bei den restlichen Zahlen nur ein X erscheinen. Den micro:bit kann man nun in einer Runde von Freunden durchgeben und jeder „zieht“ ein Los (schüttelt den micro:bit) – diese sind entweder gewinnende Lose oder Nieten.	def on_gesture_shake(): if randint(1, 6)==6: basic.show_string("GEWONNEN!") else: basic.show_icon(IconNames.NO) input.on_gesture(Gesture.Shake, on_gesture_shake)

Großes oder kleines Herz: Wird der micro:bit geschüttelt, soll er ein Symbol deiner Wahl anzeigen, ansonsten ein Fragezeichen. Werden beide Tasten (A + B) gleichzeitig gedrückt, soll zufällig entschieden werden, ob ein großes oder ein kleines Herz angezeigt wird.	def on_forever(): basic.show_icon(IconNames.HAPPY) basic.forever(on_forever) def on_gesture_shake(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) input.on_gesture(Gesture.Shake, on_gesture_shake) def on_button_pressed_ab(): if randint(0, 1) ==1: basic.show_icon(IconNames.HEART) else: basic.show_icon(IconNames.SMALL_HEART) input.on_button_pressed(Button.AB, on_button_pressed_ab)
Kniffligere Aufgabe: Es soll ein Regen-Sonnenschein-Simulator programmiert werden. Dabei soll der micro:bit ein beliebiges (selbstgewähltes) Symbol anzeigen – bis er geschüttelt wird. Passiert das, so soll zufällig entschieden und angezeigt werden, ob es morgen regnet oder die Sonne scheint (kreiere ein passendes Symbol). Achte dabei darauf, dass es doppelt so viele Sonnentage geben soll wie Regentage (2 Sonnentage und 1 Regentag im Durchschnitt). Überlege, wie viele Zufallszahlen benötigst du?	def on_forever(): basic.show_icon(IconNames.HAPPY) basic.forever(on_forever) def on_button_pressed_ab(): if randint(0, 2) ==1: basic.show_icon(IconNames.UMBRELLA) else: basic.show_leds(""" # . # . # . # # # . # # # # # . # # # . # . # . # """) input.on_button_pressed(Button.AB, on_button_pressed_ab)

Verzweigungen II *(Schwierigkeit: ☺☺☺)*

Du kannst nun Fallunterscheidungen mit mehr als zwei Fällen in deinen Programmen verwenden! Was könnte man damit umsetzen? Probiere dich selbst an deinen Ideen – oder den Aufgabenideen hier.

Arbeitsauftrag	Lösung
Erweitere das Orakelbeispiel mit drei Fällen (aus dem Vorzeige-Beispiel) mit noch einem weiteren Fall. Es soll also vier Fälle geben: ‚Ja', ‚Nein', ‚Weiß nicht' sowie z.B. ‚Frag wen anderen'. Wähle für den vierten Fall ein geeignetes Symbol aus.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) basic.forever(on_forever) def on_button_pressed_ab(): meine_Zufallszahl = randint(0, 3) if meine_Zufallszahl == 0: basic.show_icon(IconNames.YES) elif meine_Zufallszahl == 1: basic.show_icon(IconNames.SILLY) elif meine_Zufallszahl == 2: basic.show_icon(IconNames.MEH) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.AB, on_button_pressed_ab)
Ändere das große Orakel so, dass anstatt der Symbole der entsprechende Text ausgegeben wird.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) basic.forever(on_forever) def on_button_pressed_ab(): meine_Zufallszahl = randint(0, 3) if meine_Zufallszahl == 0: basic.show_string("Ja!") elif meine_Zufallszahl == 1: basic.show_string("Frag wen Anderen!") elif meine_Zufallszahl == 2: basic.show_string("Ist mir egal!") else: basic.show_string("Nein!") input.on_button_pressed(Button.AB, on_button_pressed_ab)

Kann man mit dem jetzigen Wissen das Spiel Schere-Stein-Papier mit dem micro:bit programmieren? Also so, dass wenn man ihn schüttelt, dieser entweder eine Schere, einen Stein oder ein Symbol für Papier anzeigt? Wenn das gelingt, könnte man gegen den micro:bit spielen!

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


def on_button_pressed_ab():
    meine_Zufallszahl = randint(0, 2)
    if meine_Zufallszahl == 0:
        basic.show_icon(IconNames.SCISSORS)
    elif meine_Zufallszahl == 1:
        basic.show_icon(IconNames.SMALL_SQUARE)
    else:
        basic.show_icon(IconNames.SQUARE)
input.on_button_pressed(Button.AB, on_button_pressed_ab)
```

Kennst du die Erweiterung des Spiels? Stein-Papier-Schere-Echse-Spock? Dabei kommen zwei weitere Möglichkeiten – Echse und Spock – hinzu und die Regeln werden dafür erweitert.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) basic.forever(on_forever) def on_gesture_shake(): meine_Zufallszahl = randint(0, 4) if meine_Zufallszahl == 0: basic.show_icon(IconNames.SCISSORS) elif meine_Zufallszahl == 1: basic.show_icon(IconNames.SMALL_SQUARE) elif meine_Zufallszahl == 2: basic.show_icon(IconNames.SQUARE) elif meine_Zufallszahl == 3: basic.show_leds(""" # # . . . # # . # . . # # . # # . # # . . # . # # """) else: basic.show_string("v") input.on_gesture(Gesture.Shake, on_gesture_shake)
„Würfel mit Augen“: Programmiere einen Würfel, der beim Schütteln zufällig eine Würfelzahl zeigt (mit den Augen) und ansonsten ein Symbol blinkt. Schaffst du es, dass die zuletzt gewürfelte Zahl als Ziffer durch Druck der	meine_Zufallszahl = 0 def on_forever(): if meine_Zufallszahl == 6: basic.show_icon(IconNames.SMALL_DIAMOND) basic.show_icon(IconNames.DIAMOND) else: basic.clear_screen()

Taste A nochmal angesehen werden kann? Und wie muss man den Code erweitern, dass der micro:bit nach einem gewürfelten Sechser anders blinkt (z.B. Sterne)?

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

def on_button_pressed_a():
    basic.show_number(meine_Zufallszahl)
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_gesture_shake():
    global meine_Zufallszahl
    meine_Zufallszahl = randint(1, 6)
    if meine_Zufallszahl == 1:
        basic.show_leds("""
        . . . . .
        . . . . .
        . . # . .
        . . . . .
        . . . . .
        """)
    elif meine_Zufallszahl == 2:
        basic.show_leds("""
        . . . . .
        . . . # .
        . . . . .
        . # . . .
        . . . . .
        """)
    elif meine_Zufallszahl == 3:
        basic.show_leds("""
        . . . . #
        . . . . .
        . . # . .
        . . . . .
        # . . . .
        """)
    elif meine_Zufallszahl == 4:
        basic.show_leds("""
        . . . . .
        . # . # .
        . . . . .
        . # . # .
        . . . . .
        """)
    elif meine_Zufallszahl == 5:
        basic.show_leds("""
        # . . . #
        . . . . .
        . . # . .
```

	`` . . . . . # . . . # """) else: basic.show_leds(""" # . . . # . . . . . # . . . # . . . . . # . . . # """) input.on_gesture(Gesture.Shake, on_gesture_shake) ```
Expertenaufgabe: Erstelle einen Anweisungsgenerator! Der micro:bit soll dir Anweisungen geben, z.B. A (als ‚Drücke Taste A'), B (für Taste B), II (für A+B), S (für Schütteln) usw. Warte zwischen den einzelnen Anweisungen 3,5 Sekunden (verwende basic.pause() in der Kategorie Grundlagen – Angabe in Millisekunden!). Erweiterung: Lass den micro:bit anzeigen, ob das, was getan wurde (z.B. Schütteln) richtig war – z.B. durch Häkchen und X.	``` meine_Zufallszahl = 0 def on_forever(): global meine_Zufallszahl meine_Zufallszahl = randint(0, 3) if meine_Zufallszahl == 0: basic.show_string("A") elif meine_Zufallszahl == 1: basic.show_string("B") elif meine_Zufallszahl == 2: basic.show_leds(""" . # . # . . # . # . . # . # . . # . # . . # . # . """) else: basic.show_string("S") basic.pause(3500) basic.forever(on_forever) def on_button_pressed_a(): if meine_Zufallszahl == 0: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.A, on_button_pressed_a) def on_button_pressed_b(): if meine_Zufallszahl == 1: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_button_pressed(Button.B, on_button_pressed_b) def on_button_pressed_ab(): if meine_Zufallszahl == 2: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) ```

	input.on_button_pressed(Button.AB, on_button_pressed_ab) def on_gesture_shake(): if meine_Zufallszahl == 3: basic.show_icon(IconNames.YES) else: basic.show_icon(IconNames.NO) input.on_gesture(Gesture.Shake, on_gesture_shake)