micro:bit Programmierung in Python – Lösungen zu Kapitel II

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*	def on_forever(): basic.show_leds(""" . . . . """) basic.forever(on_forever)

eigene Muster zeichnen kann.	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	def on_gesture_tiltLeft(): basic.show_string("Hallo Max!") input.on_gesture(Gesture.TILT_LEFT, on_gesture_tiltLeft) def on_gesture_tiltRight():

geneigt den linken Sitznachbarn. Drückst du A+B, soll der micro:bit dich begrüßen.	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 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.	def on_forever(): basic.show_leds(""" . # # # . . . . # . . . # # . . . . # . . """) 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)