Lektion 6: micro:bit mit Python | **Übung 2: Tasten und Gesten**

In dieser Lektion erfahren Sie, wie Sie die micro:bit-Tasten **und** -Gesten verwenden, um dann ein Programm schreiben, das einen Würfelwurf simuliert , dessen Werte in einer Liste gesammelt werden, die in ein Datendiagramm übertragen werden soll.
Diese Lektion besteht aus zwei Teilen:

Teil 1: Untersuchen von Tasten und Gesten
Teil 2: Verwenden einer Taste oder einer Geste zum Erzeugen von Daten

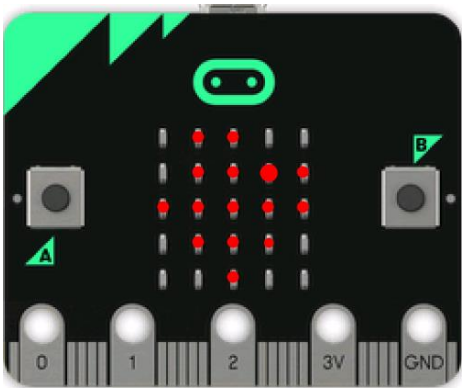
Lernziele:

- Lesen und verarbeiten Sie die Zustände der A- und B-Taste auf dem micro:bit
- Lernen Sie den Unterschied zwischen **.was** und **.is** kennen
- Übertragen von Daten mit Python zu TI-Nspire
- Untersuchen Sie die vom micro:bit gesammelten Daten
- Verwenden von Gesten zur Steuerung der Anzeige

Lehrerhinweis: Wie in Übung 1 ist diese Lektion von innen nach außen gestaltet. Code wird nicht sequenziell eingeführt, sondern entwickelt, um sich zuerst auf die Schaltflächen- und Gestenfunktionen des micro:bit zu konzentrieren und dann mit dem Herstellen einer Verbindung zwischen Python-Listen und TI-Nspire-Listen abzuschließen.

1. Das micro:bit verfügt über zwei Tasten mit den Bezeichnungen A und B auf jeder Seite des Displays. Das python micro:bit-Modul verfügt über zwei *ähnliche Methoden* zum Lesen der Tasten und zum Ausführen von Aufgaben basierend auf diesen Tasten. Zuerst testen Sie die Methoden und schreiben dann ein Programm, mit dem Sie Daten sammeln und an anderer Stelle im TI-Nspire CX II analysieren können.

   Es gibt auch einen 3-Achsen-Beschleunigungsmesser / Kompasschip auf der Rückseite des micro:bit und Methoden zur Interpretation von micro:bit-Bewegung und -Orientierung.



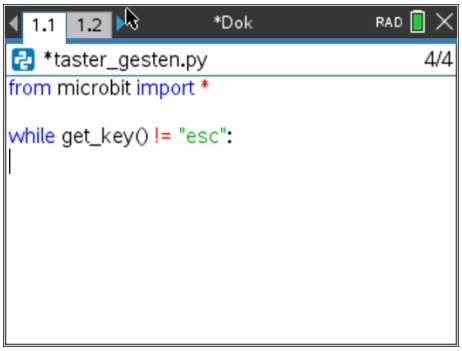
Lehrertipp: Bei der micro:bit Version 2 befindet sich auch ein 'Touch'-Button über dem Display (das grüne Oval). Im Modul wird dies als 'Logo Touch' bezeichnet und verwendet die Methode **'is_touched()'**. Diese Touch-Taste wird in diesen Lektionen nicht behandelt, kann aber basierend auf dem, was Sie in dieser Lektion lernen, leicht in Ihre Programme integriert werden. Beachten Sie den Unterschied zwischen dem Verhalten von **.is_** und **.was_** weiter unten...

2. Teil 1: Untersuchen von Schaltflächen und Gesten

Zunächst einmal sollte ein neues Dokument erzeugt werden; im Beispiel heißt es **taster_gesten**.

Über **[menu] > Weitere Module > BBC micro:bit** wird als erstes die **import** – Anweisung eingefügt:

from microbit import *



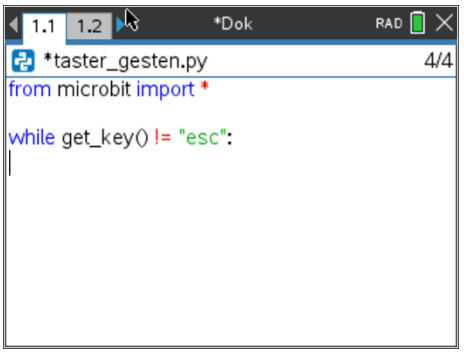
3. Hinzufügen einer while-Schleife:

   while get_key()! = 'esc':

   aus

   [menu] > Weitere Module > BBC micro:bit > Commands

   So sehen die meisten Programme dieser Lektion aus.



4. Um die Taste A zu testen, wird eine Verzweigung **if** eingeführt:

   if button_a.was_pressed():
   print("Taste A")

   ***if** ist eingerückt, um Teil der while-Schleife* zu sein und ***print( )** wird noch mehr eingerückt, um Teil des **if-Blocks** zu sein. Denken Sie daran, dass die richtige Einrückung in Python sehr wichtig ist. Der falsche Einzug kann zu Syntaxfehlern oder einer fehlerhaften Ausführung des Codes führen. Beachten Sie die hellgrauen Diamantsymbole (♦♦), die diesen Einzug markieren.*

   if findet man in **[menu] > Eingebaute > Control**.

   Die *Bedingung* **button_a.was_pressed()** findet man in **[menu] > Weitere Module> BBC micro:bit > Buttons and Logo Touch**

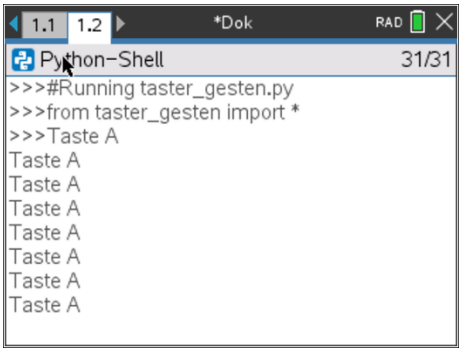
   print( ) ist im Menü **[menu] > Eingebaute > I/O**

   Der Text „Taster A“ muss in die Klammern bei **print( )** geschrieben werden.

   Hinweis: **is_pressed()** *wird später behandelt.*

5. Nun kann das Programm getestet werden. Drücken Sie **[ctrl] [R],** um das Programm auszuführen. Es sieht so aus, als würde nichts passieren. Drücken und lösen Sie die Taste A am micro:bit. Auf dem Rechnerbildschirm wird "Taste A" angezeigt. Jedes Mal, wenn Sie die Schaltfläche drücken und lösen, wird der Text wie in diesem Bild angezeigt.

   Drücken Sie **[esc]** und kehren Sie zum Python-Editor zurück.



6. Fügen Sie eine weitere **if**-Anweisung hinzu, um Taste B mithilfe der Bedingung **button_b.is pressed()** zu überprüfen. Beachten Sie, dass "IS" sich von "WAS" unterscheidet.
Sie werden bald sehen, wie sie sich unterscheiden .

```
    if button_b.is_pressed():
        print("Taste B")
```

Tipp: Achten Sie auch hier auf die Einrückungen!

Lehrertipp: Die beiden Funktionen verhalten sich unterschiedlich. Es gibt Programmierumstände, bei denen die Wahl zwischen den beiden verschiedenen Verhaltensweisen wichtig ist.

7. Führen Sie das Programm erneut aus (drücken Sie **[ctrl] [R]**). Probieren Sie die beiden Tasten A und B aus.
Tippen Sie auf jede Taste *und* **halten** Sie jede Taste gedrückt.
Sie sehen Taste B' wiederholt angezeigt, solange Taste B gedrückt gehalten wird, aber nicht Taste A'. Es gibt einen Unterschied zwischen **.was_pressed()** (ein Loslassen der Taste zum Zurücksetzen ist erforderlich) und **.is_pressed ()** (es wird nur überprüft, ob die Taste in dem *Moment,* in dem die Anweisung verarbeitet wird, gedrückt ist).

Hinweis: Wenn Sie schnell auf Taste B tippen, zeigt das Programm möglicherweise *nicht* Taste B *an, da die Taste in dem Moment, in dem die if-Anweisung verarbeitet wird,* noch nicht wieder losgelassen wurde.

Hinweis:

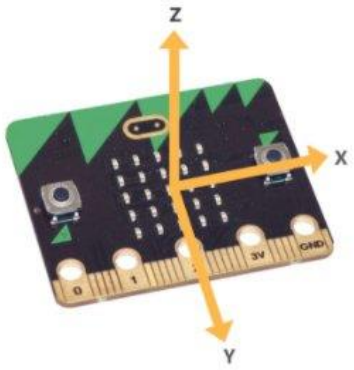
.was_pressed() erfordert eine vollständige Drücken-und-Loslassen – Aktion, um einzelne Betätigungen des Tasters erkennen zu können.

.was_pressed() erkennt ein solches Ereignis, auch wenn die Taste zu einem anderen Zeitpunkt in der Ausführung der Schleife angeklickt wurde. Die Taste muss freigegeben werden, damit ein weiterer Klick erfolgen kann. Das micro:bit "erinnert" sich, dass die Taste gedrückt wurde.

.is_pressed() wirkt wie ein "ist gedrückt?"-Ereignis. Einige ereignisgesteuerte Programmiersprachen haben eine ähnliche Funktion wie z.B. "mouse_down". **.is_pressed()** erzeugt **True** nur, **wenn die Taste genau zu dem Zeitpunkt schaltet, zu dem die Anweisung verarbeitet wird.**

Im Rest dieser Lektion wird Code nur für eine Taste berücksichtigt.

8. **Einführung von Gesten**. Der micro:bit verfügt über einen elektronischen *Beschleunigungsmesser,* der Beschleunigungskräfte in drei verschiedene Richtungen messen kann (ein 3-Achsen- oder 3D-Beschleunigungsmesser). Neben numerischen Werten für die Beschleunigung in jede Richtung (siehe Bild) bietet der micro:bit auch einfache, leicht verständliche "Gesten" wie "Face up" und "Face down", die auf diesen Richtungswerten basieren.



   In dieser Lektion werden die Gesten untersucht und gezeigt, wie Sie sie bei der Programmierung Ihres TI-Nspire CX II mit micro:bit verwenden können.

9. Abrufen des Gestenwerts und Speichern in der Variablen **g:**

   ♦ ♦g = accelerometer.current_gesture()

   Zunächst **♦ ♦g =** schreiben (Einrückung beachten!)

   Dann **accelerometer.current_gesture()** entnehmen aus

   [menu] > Weitere Module > BBC micro:bit > Sensors > Gestures >

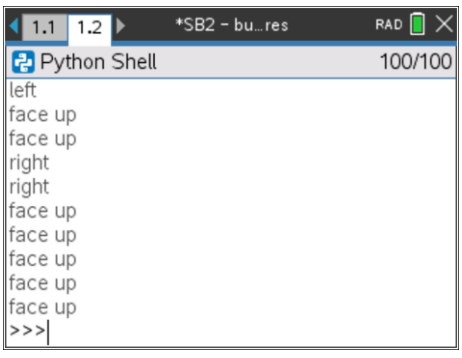
   Dann soll noch der Wert von **g** dargestellt werden:

   ♦ ♦print(g)

   Erinnerung: **print( )** findet man in **[menu] > Eingebaute > I/O**

   Beachten Sie, dass diese beiden Anweisungen so eingerückt werden, um Teil der while-Schleife zu sein, aber *nicht Teil der if-Anweisung darüber. Denken Sie daran, dass die Einrückung jeder Zeile die Bedeutung bestimmt, also SEIEN SIE VORSICHTIG!*

10. Führen Sie das Programm aus und beobachten Sie, wie der Taschenrechner die verschiedenen Werte anzeigt, während Sie das micro:bit in der Luft bewegen. Drehen Sie micro:bit um, stellen Sie es auf jede Kante, schütteln, rasseln und rollen Sie es. Ein Teil eines Beispiellaufs ist in dieser Abbildung dargestellt.



   Einige der verfügbaren Gesten sind: **face up, face down, up, down, left, right, shake.** Diese werden als Zeichenfolgenwerte zurückgegeben. Können Sie all diese Gesten auf Ihrem Bildschirm sehen?

11. Deaktivieren Sie die letzten beiden Zeilen, indem Sie sie als Kommentar kennzeichnen **([ctrl][T])** und fügen Sie die beiden neuen Zeilen hinzu:

    if accelerometer.was_gesture("face down"):

    print("face down")

    Man findet die Anweisung zu den Gesten wieder in:

    [menu] > Weitere Module > BBC micro:bit > Sensors >-Gestures

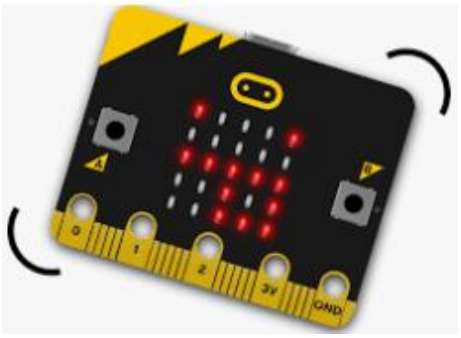
    Man kann die Anweisungen einfach schreiben, aber in der **.was_gesture()-Methode** muss der Text *genau mit dem Menüpunkt übereinstimmen.*

    Wenn Sie dieses Programm ausführen, werden Sie die Änderung in der Ausgabe bemerken:

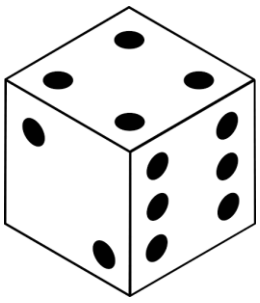
    .current_gesture() druckt die Geste ständig.

    .was_gesture() druckt nur, wenn sich die Geste ändert.

12. Tasten und Gesten sind zwei Möglichkeiten, Eingaben vom micro:bit zu erhalten und Ergebnisse entweder auf dem TI-Nspire CX II-Bildschirm oder dem micro:bit-Display zu erzielen... oder beides.
    Im nächsten Teil dieser Lektion wird ein Programm erstellt, das mit dem micro:bit einige Daten für weitere Untersuchungen auf dem TI-Nspire CX II erzeugt.



13. Teil 2: Würfelwurf



Wenn **Taste A** gedrückt wird, soll eine zufällige ganze Zahl von 1 bis 6 erzeugt und einer Variablen zugewiesen werden. Sie können Taste A, aber auch Taste B oder eine Geste Ihrer Wahl verwenden.
Der Wert soll nur auf dem micro:bit angezeigt werden. Probieren Sie es selbst aus, bevor Sie sich den nächsten Schritt ansehen. Wir werden das aktuelle Programm verwenden und Code hinzufügen, um den Würfelwurf zu simulieren.

Wissen Sie, welche Zahl sich auf der Unterseite des abgebildeten Würfels befindet?

Lehrertipp:

Es ist die 3, denn die gegenüberliegenden Seiten eines Würfels ergänzen sich immer zu 7.

14. Das Modul **Zufallszahl** aus **[menu]>Zufallszahl** muss hinter dem micro:bit-Modul eingefügt werden und daraus die Anweisung **randint(1,6**) an der hervorgehobenen Stelle**:**

    from random import * und **randint()**

    Die Anweisung ♦♦♦♦**w = randint(1, 6)** muss manuell ergänzt werden, wobei man wieder auf die Einrückung achten muss, da sie zum if-Block gehört.

15. Jetzt muss noch durch

    display.show(w)

    der „gewürfelte“ Wert w auf dem micro:bit angezeigt werden.

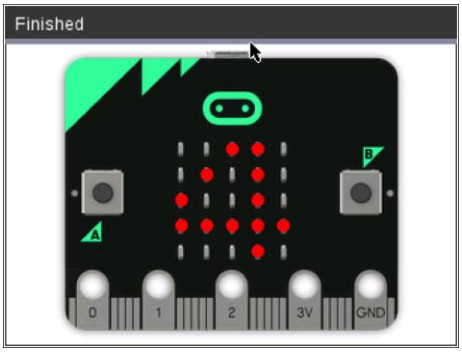
    Führen Sie das Programm erneut aus. Jedesmal wenn Sie die Taste A drücken, sehen Sie 'Taste A' auf dem Handheld-Bildschirm und die Zahl auf dem micro:bit-Display ändert sich... aber nicht *jedes* Mal, denn manchmal ist die Zufallszahl die gleiche wie beim letzten „Wurf“. Das ist in Ordnung.

16. **Sammeln von Daten:** Auf Knopfdruck zu würfeln, ist zwar nett, aber für weitere Studien wäre es hilfreich, all diese Werte zu speichern, damit man die Daten interpretieren kann: Welche Zahl tritt am häufigsten auf? Was ist die durchschnittliche Zahl? Und so weiter...

    Fügen Sie dem Programm Anweisungen hinzu, um:

    - eine leere Liste zu erstellen
    - einen Wert zu dieser Liste hinzuzufügen (.append)
    - die Liste von Python im TI-Nspire CX II-System zur Analyse zu speichern

    Jede dieser drei Aufgaben wird in Anweisungen übersetzt, die an speziellen Stellen im Programm platziert werden. Probieren Sie es selbst aus, bevor Sie mit dem nächsten Schritt fortfahren.



17. Zunächst muss eine leere Liste erzeugt werden:

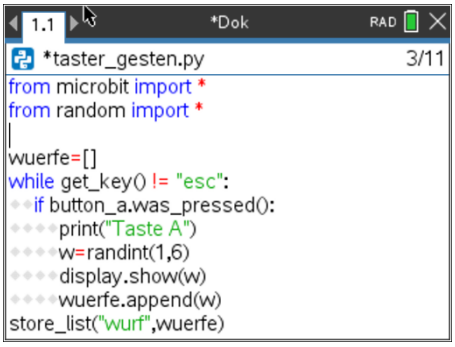
    wuerfe = [ ]

    Die Klammern finden sich auf der Tastatur und in **[menu] > Eingebaute > Listen**

    Ein Wurf wird der Liste hinzugefügt durch:

    wuerfe.append(w)

    .append() findet man in **[menu] > Eingebaute> Listen**



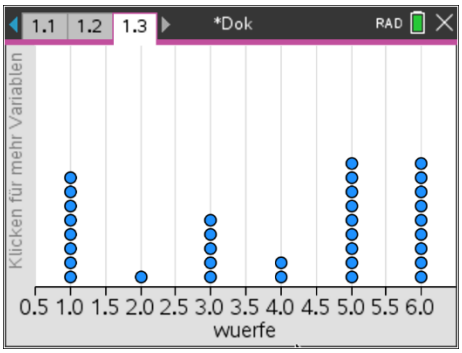
Zuletzt wird die Liste in einer TI-nspire-Liste gepeichert:

store_list("wurf", wuerfe)

Die **store_list** – Funktion befindet sich in **[menu] >BBC micro:bit >Commands**

Hinweis: Die kommentierten und alle überflüssigen Zeilen wurden entfernt oder ausgeblendet, damit der gesamte Code auf einen Bildschirm passt.

18. Wenn Sie das Programm jetzt ausführen, drücken Sie mehrmals die Taste A und zum Schluss **[esc],** um das Programm zu beenden. Ihr Python-Programm hat eine Liste namens "wuerfe" und jetzt hat Ihr TI-Nspire auch eine Liste namens "wuerfe". Dies sind zwei separate Listen.



Drücken Sie **[ctrl]-[doc]** oder **[ctrl]-[I],** um eine **Data & Statistics-Seite** hinzuzufügen. Man sieht eine Wolke aus verstreuten Punkten. Klicken Sie unten in der App auf die Meldung '*Klicken Sie hier, um eine Variable hinzuzufügen*' und wählen Sie die Liste **aus.** Sehen Sie den Bildschirm rechts? Jeder Punkt repräsentiert einen Ihrer Würfelwürfe. Welche Informationen können Sie aus diesem Diagramm entnehmen? Können Sie das Diagramm in ein Histogramm ändern? Tipp: Überprüfen Sie das **[Menü]** der App.

Sie können dieses Programm auf verschiedene Arten ändern. Zum Beispiel: Zeigen Sie mit **print()** an, wie oft Sie gewürfelt haben.

Hinweis: Wie man leicht sieht, haben die Taste B und eine Geste keinen Einfluss auf den Würfel. Versuchen Sie, Ihren Code so zu ändern, dass ein Wurf nur dann erfolgt, wenn Sie das micro:bit "schütteln".

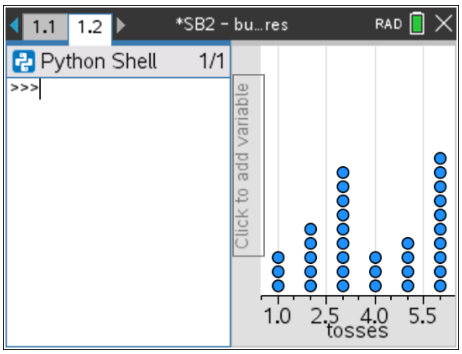
Und zuletzt: Denken Sie daran, Ihr Dokument zu speichern!

19. **Erweiterung:**
    - *Rücken Sie die Anweisung **store_list()** ein, sodass die Liste jedesmal im TI-nspire gespeichert wird, wenn man die Taste A drückt.*

20. Wechseln Sie zur Shell-App und drücken Sie **[ctrl] [4],** um die Shell mit der Data & Statistics-App auf derselben Seite wie im Bild rechts zu kombinieren.

    Führen Sie das Programm aus, indem Sie **[ctrl] [R]** auf der Shell drücken. Drücken Sie die Taste A auf dem micro:bit, um mit dem Würfeln zu beginnen und das Punktdiagramm wachsen zu sehen!

    Das Diagramm zeigt zunächst "Keine numerischen Daten" *an, da die* **Zählliste** *ja vom Programm geleert wurde, aber keine Angst: Drücken von Taste A beginnt mit dem Plotten der Daten.*



21. Anstatt bei jedem Tastendruck nur "Taste A" zu drucken**,** können Sie die gesamte Liste der Würfe drucken. Im Beispiel oben ist die entsprechende Anweisung eingesetzt.