Unit 6 : micro:bit met Python — Oefenblad 2 : Knoppen en beweging

In deze les leer je de **knoppen** en de **bewegingsmogelijkheden** van de micro:bit te gebruiken. Je maakt een programma dat het werpen met een dobbelsteen simuleert en de uitkomsten in een lijst opslaat die je naar dataplot van TI-Nspire exporteert.
Deze les bestaat uit twee delen:

Deel 1: De werking van de knoppen en het effect van bewegen onderzoeken;
Deel 2: Beweging en knoppen gebruiken voor het genereren van data.

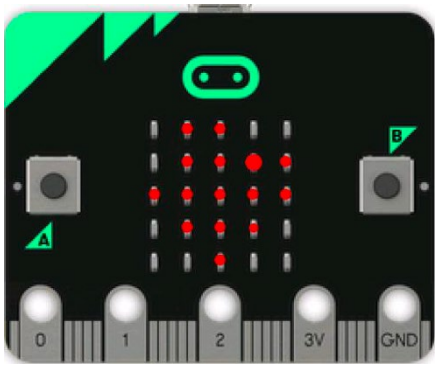
Doelen :

- Bepalen of knop A of B is ingedrukt;
- Het verschil tussen **.was** en **.is;**
- Data uitwisselen tussen Python en TI-Nspire;
- Data vanuit de micro:bit beheren;
- Bewegingen van de micro:bit gebruiken.

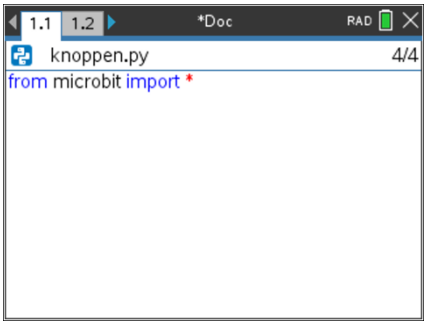
---

1. De micro:bit beschikt over twee knoppen, gelabeld A en B, aan weerszijden van het display.
   De Python-'micro:bit'-module kent twee gelijksoortige methodes om de knoppen af te lezen en om daarna taken uit te voeren, afhankelijk van de uitkomst.
   Eerst test je deze methodes en dan schrijf je een Python-programma waarmee je data verzamelt en analyseert met de TI-Nspire.

   Er zit ook een accelerometer/compas-chip op de achterkant van de micro:bit en er zijn methodes (opdrachten) waarmee beweging en oriëntatie van de micro:bit kan worden gemeten.



2. **Deel 1: Knoppen en bewegingen onderzoeken**
   Begin een nieuw Python-programma in een nieuw document.
   (Druk op de **[Home]** toets, kies **New** en dan **Add Python>New**).
   Geef het een naam bijvoorbeeld "knoppen".
   Start weer met **from microbit import ***, deze kun je vinden in **[menu]>More Modules>BBC micro:bit.**



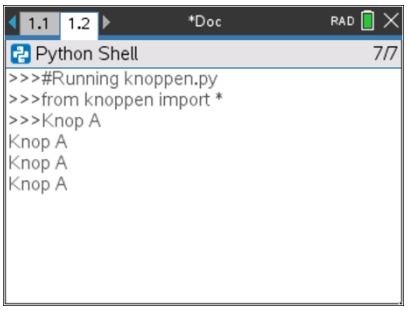
3. Voeg de **while-lus** (loop) toe
   (**[menu]>More Modules>BBC micro:bit>Commands**).

   Vrijwel alle voorbeelden met de micro:bit hebben deze structuur.

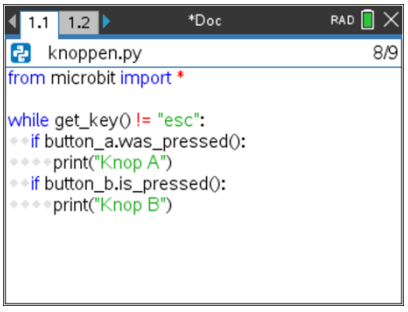
4. Om knop A te testen gebruiken we de **if**-structuur:
   if button_a.was_pressed():
   ◊ ◊ **print(“knop A”)**
   De if-opdracht is onderdeel van de while-opdracht en moet dus inspringen. De print opdracht springt extra in, omdat deze bij de if-opdracht hoort.
   If kun je vinden bij **[menu]>Built-ins>Control**.
   button_a_was_pressed() vind je bij:
   [menu]>More Modules>BBC micro:bit>Buttons and Logo Touch.

5. Je kunt nu het programma testen.
   Druk op **[ctrl] [R]** om het programma uit te voeren.
   Het lijkt alsof er niets gebeurt. Maar elke keer als je knop A indrukt verschijnt “Knop A” op het scherm.

   Druk op **[esc]** om het programma te stoppen.

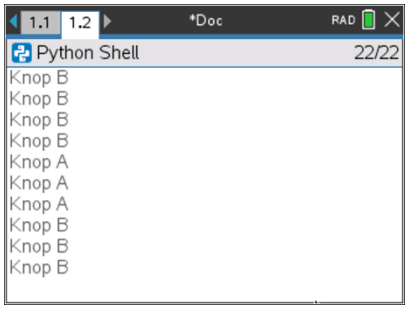


6. Voeg nu een tweede if-opdracht toe voor knop B, maar gebruik nu de opdracht: **if button_b.is_pressed()**
   Merk op dat er nu **“**.IS” staat in plaats van “.WAS”.
   Let ook nu weer op de juiste inspringingen.



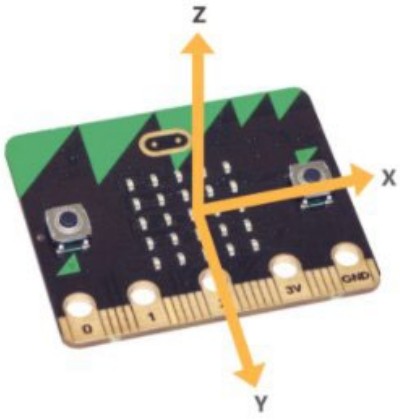
7. Voer het programma weer uit en druk op de verschillende knoppen A en B.
   Het verschil is dat “Knop A” pas wordt getoond als de knop weer wordt losgelaten, terwijl “Knop B” steeds opnieuw wordt getoond, zolang deze ingedrukt blijft.

   Merk op dat als je knop B indrukt en snel weer loslaat het mogelijk is dat er geen actie volgt, omdat op het moment dat de if-opdracht wordt uitgevoerd de knop niet is ingedrukt.



8. **Bewegingsdetectie (gestures)**
   De micro:bit heeft een ingebouwde *accelerometer* (versnellingssensor) die in drie verschillende richtingen versnellingen kan meten (3D-accelerometer).
   Naast numerieke waarden voor de versnelling in elke richting (zie afbeelding hiernaast) kent de micro:bit ook duidelijke 'gestures' zoals "face up" en "face down" gebaseerd op deze numerieke waarden.

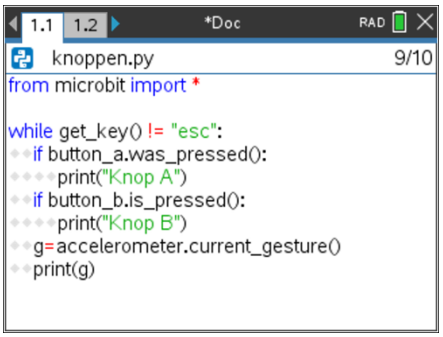
   We gaan dit onderzoeken met ons programma.



9. De opdracht **g = accelerometer.current_gesture()** slaat de 'waarde' op in de variabele g.
   (Je kunt deze opdracht vinden met:
   [menu]>More Modules>BBC micro:bit>Sensors>Gestures)

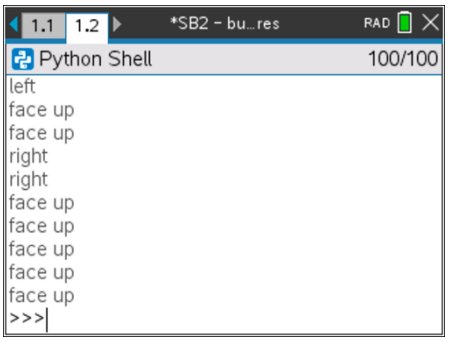
   Print(g) toont deze op het scherm.

   Let op dat je het juiste aantal spaties gebruikt. De laatste twee regels horen bij de while-opdracht en niet bij de if-opdracht.



10. Voer het programma uit en kijk naar de verschillende meldingen op het rekenmachinescherm als je de micro:bit schudt, omdraait, rechtop zet, enz.

    Sommige beschikbare 'gestures' zijn: **face up**, **face down**, **up**, **down**, **left**, **right**, **shake**.



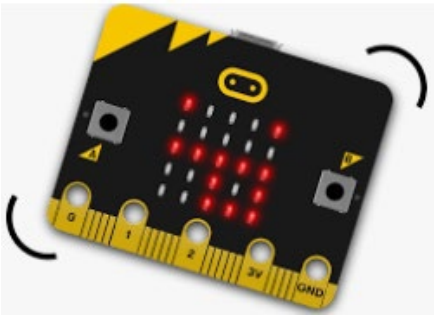
11. Voorzie de laatste twee regels in je programma van het commentaarsymbool (#) (*dit wil zeggen dat het programma deze regels tijdens het uitvoeren negeert*) en voeg de volgende twee regels toe:
if accelerometer.was_gesture(“face down”):
◆ ◆ **print(“face down”)**

Je vindt alle ‘gesture’-opties en functies in:
menu]>More Modules>BBC micro:bit>Sensors>Gestures.

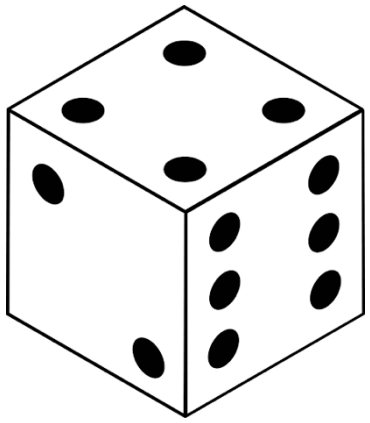
*Tip: Om een programmaregel als commentaarregel te markeren kun je **[ctrl] [T]** gebruiken.*

Als je dit laatste programma uitvoert merk je een verschil met de vorige versie;
.current_gesture() toont continu de melding.
.was_gesture() toont deze alleen als er iets wijzigt.

12. Knoppen en ‘gestures’ bieden beide de mogelijkheid om input van de micro:bit te krijgen.
In het volgende deel verzamelen we data met behulp van de micro:bit en verwerken we deze met de TI-Nspire.



13. **Deel 2: Werpen met een dobbelsteen**
Als knop A is ingedrukt geef dan een variabele, een willekeurig heel getal van 1 t/m 6. Dit simuleert de dobbelsteen (in plaats van knop A kun je ook knop B gebruiken of één van de ‘gestures’).
Laat het getal op het display van de micro:bit zien.
Voordat je verdergaat, kun je eerst zelf proberen om het programma te schrijven.



14. Begin me een nieuw Python-programma en voeg weer de ‘micro:bit’-module toe.
    Om een willekeurig (random) getal te kunnen kiezen hebben we een functie nodig uit de random-module. Deze voegen we ook toe (zie de tweede regel in het programma).
    Het programma lijkt op het vorige met ook weer de while-structuur.
    De regel **uitkomst = randint(1,6)** geeft elke keer als knop A is ingedrukt de variabele ‘uitkomst’ een willekeurige waarde (1 tot 6).

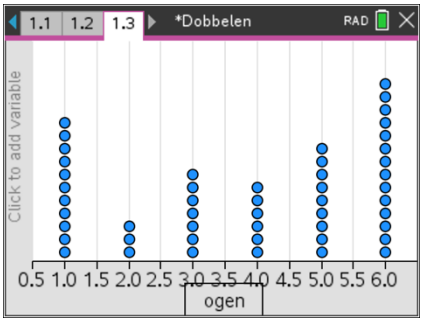
15. Voeg nu ook de opdracht toe om deze waarde op het display van de micro:bit weer te geven.
    Dat kan met **display.show(uitkomst)**

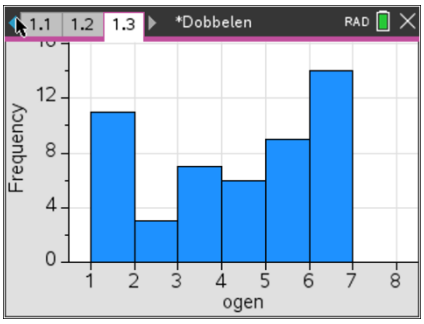
    Voer het programma uit en elke keer als je knop A indrukt zie je de gekozen waarde op het display van de micro:bit verschijnen.

16. **Data verzamelen**
    Het werpen met een dobbelsteen door op een knop te drukken is leuk, maar we willen de uitkomsten opslaan om er conclusies uit te kunnen trekken.

17. Begin met een lege lijst waarin de uitkomsten worden opgeslagen, in dit programma heet die lijst: uitkomsten.
    Elke keer als knop A wordt ingedrukt moet de uitslag worden toegevoegd aan de lijst. Dat kan met **uitkomsten.append().**
    Dit kun je vinden met: **[menu]>Built-ins>List.**
    Aan het eind van het programma worden de elementen van de lijst opgeslagen in een ‘TI-Nspire’-lijst met de opdracht:
    store_list(“ogen”,uitkomsten)
    Deze functie kun je vinden met:
    [menu]>BBC micro:bit>Commands.

18. Voer het programma uit en druk een groot aantal keren op knop A. Druk dan op **[esc]** om het programma weer te stoppen. Het Python-programma bevat nu een lijst “uitkomsten” met alle uitkomsten en TI-Nspire heeft nu ook een lijst met dezelfde uitkomsten, deze heet “ogen”.
Druk op **[ctrl]-[doc]** (of **[ctrl]-[i]**) om een nieuwe pagina aan het document toe te voegen en kies dan voor **Data & Statistics**.
Je ziet een scherm met een groot aantal punten. Door onderaan het scherm te klikken en de variabele “ogen” te kiezen, verschijnt een figuur zoals hiernaast.





In het menu van deze toepassing kun je het plottype veranderen en bijvoorbeeld voor een histogram kiezen.

19. **Uitbreiding:**
Verander het programma zo dat elke keer als knop A wordt ingedrukt de uitkomstenlijst wordt opgeslagen en de uitkomst zelf wordt getoond.

Combineer de toepassingsschermen door in de Python-shell op **[ctrl]-[4]** te drukken.
Als je nu het programma weer uitvoert zie je dat de uitkomsten in het linkerdeel, en het histogram in het rechterdeel van het scherm steeds worden aangepast.