Unit 6 : micro:bit — Oefenblad 2: Knoppen

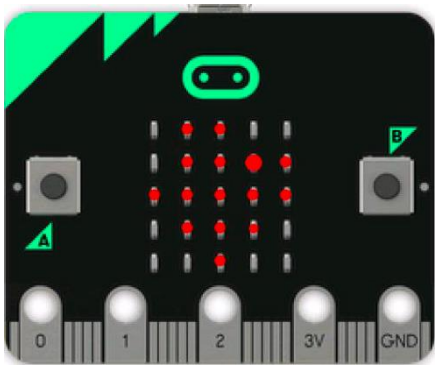
In deze les leer je de **knoppen** van de micro:bit gebruiken. Je maakt een programma dat het werpen met een dobbelsteen simuleert en de uitkomsten in een lijst opslaat. Daarna exporteer je de lijst naar de TI-84-omgeving waarmee je vervolgens een plot maakt.
Deze les bestaat uit drie delen:

- Deel 1: De werking van de knoppen onderzoeken;
- Deel 2: De knoppen gebruiken voor het genereren van data;
- Deel 3: Een lijst naar de TI-84 Plus CE-T Python Edition exporteren.

Doelen :

- Bepalen of knop A of B is ingedrukt;
- Het verschil tussen **.was** en **.is;**
- Data vanuit de micro:bit beheren;
- Exporteren van de data naar de TI-84 Plus CE-T Python Edition.

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 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-84-functies.



1. **Deel 1: Knoppen onderzoeken**
   Start een nieuw Python-programma en noem het KNOPPEN.

   Begin met **from ti_system import ***
   (Dit kun je vinden in **[math] ti_sysrem**)
   Importeer ook weer de micro:bit-module:
   from microbit import *

   Start daarna een while-loop: **while not escape()**
   (zie hiernaast).

   Belangrijk: 'micro:bit import' moet na de 'ti_system import'-opdracht komen!

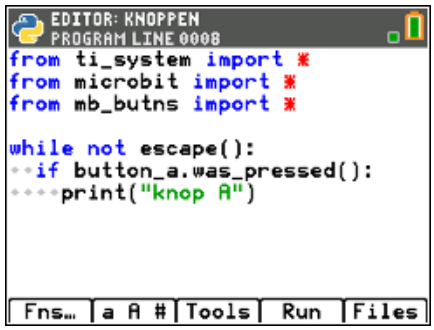
   Opmerking: Vrijwel alle voorbeelden met de micro:bit hebben deze structuur.

2. Om de micro:bit-knoppen te kunnen gebruiken moet je eerst een module importeren.
   Plaats de cursor op de lege regel na de twee importopdrachten.
   Druk op **[math]** en selecteer **microbit…** en kies nu voor **Buttons en Touch Logo**. Dit voegt de regel **from mb_butns import *** toe.

3. Om knop A te testen gebruiken we de **if**-structuur:
   if button_a.was_pressed():
   ◆◆ **print(“knop A”)**

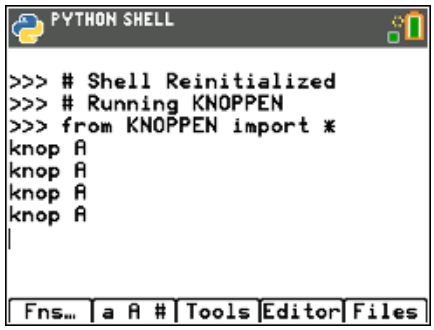
   De if-opdracht is onderdeel van de while-opdracht en moet dus inspringen. De print-opdracht springt extra in omdat deze bij de if-opdracht hoort.

   If kun je vinden met **<Fns…> Ctl**.
   button_a_was_pressed() vind je met: **[menu] buttons…**
   *Merk op dat er een **button A**- en een **button B**-submenu is.*
   Kies **.was_pressed()** onder het **button A**-menu.
   print() vind je bij **<Fns> I/O**.

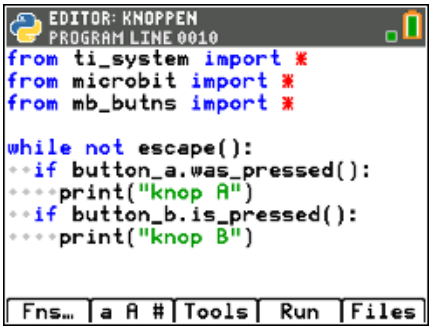


4. Je kunt nu het programma testen.
   <Run> het programma.
   Het lijkt alsof er niets gebeurt. Maar elke keer als je knop A indrukt verschijnt “knop A” op het scherm.

   Druk op **[clear]** om het programma te stoppen en ga weer naar de Python-editor.



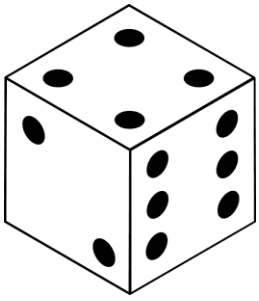
5. Voeg nu een tweede if-opdracht toe voor knop B maar gebruik nu de opdracht: **if button_b.is_pressed()**
   Merk op dat er nu **"**.IS" staat in plaats van ".WAS".
   Let ook nu weer op de juiste inspringingen.



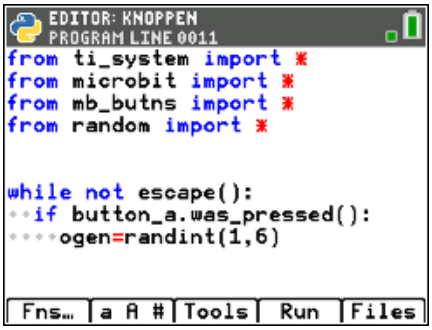
6. **<Run>** het programma weer en druk op de knoppen A en B.
   Het verschil is dat "knop A" pas wordt afgedrukt als je de knop weer loslaat, terwijl "knop B" steeds opnieuw wordt afgedrukt zolang deze ingedrukt blijft.

   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.

7. **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.
   Laat het getal afdrukken op het display van de micro:bit.
   Voordat je verder gaat, kun je eerst zelf proberen om het programma te schrijven.



8. Om een willekeurig (random) getal te kunnen kiezen hebben we een functie nodig uit de random-module.
   Deze module kun je vinden met **[math] random…**
   Voeg in het vorige programma de regel **from random import *** toe.
   De regel **ogen = randint(1,6)** geeft elke keer als knop A is ingedrukt de variabele 'ogen' een willekeurige waarde (1 tot 6).
   (De functie **randint()** kun je vinden met **[math] random…**)



9. Het aantal ogen moet elke keer op het micro:bit-display worden getoond. Voordat we dat kunnen doen moet je eerst de display-module importeren. (Net als in de vorige unit.)
   Voeg dus eerst toe: **from mb_display import ***

   De regel **display.show(ogen)** toont het gegooide aantal ogen op het display van de micro:bit.

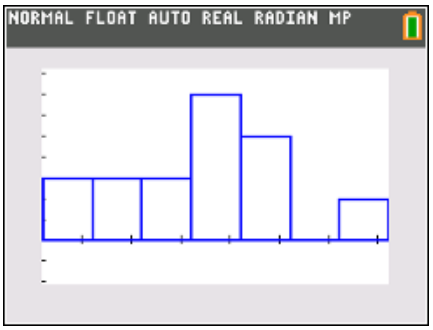
   Voer het programma uit en elke keer als je knop A indrukt, zie je de gekozen waarde op het display van de micro:bit verschijnen.

10. **Deel 3: 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.
    We breiden het programma uit met drie regels.

    Begin met een lege lijst waarin de uitkomsten worden opgeslagen, in dit programma heet die lijst: uitkomsten.
    (uitkomsten = [ ])
    Elke keer als je knop A indrukt moet de uitslag worden toegevoegd aan de lijst. Dit kan met uitkomsten.append().
    (**.append()** kun je vinden met **<Fns…> list**)

11. Om de elementen van de lijst met uitkomsten op te slaan in een TI-84-lijst voegen we aan het eind van het programma nog een regel toe: **store_list("DOBBL"),** Te vinden met **[math] ti_system**).
    Deze functie slaat de lijst uitkomsten op in een TI-84-lijst met de naam DOBBL. Deze naam moet in hoofdletters en mag niet uit meer dan 5 letters bestaan.

12. **<Run>** het programma en druk een aantal malen op knop A en stop het programma weer door op de **[clear]**-toets te drukken. Sluit de Python-app af en controleer dat de TI-84-rekenmachine nu een lijst heeft met de naam DOBBL.
(Druk hiervoor op **[2nd] [stat]**).
Met de 'stat plot-optie van de rekenmachine kun je nu bijvoorbeeld een grafiek maken zoals hiernaast.



*Opmerking: Vanwege de geheugenbeperking kan een rij met **store_list()** maximaal 100 elementen bevatten.*