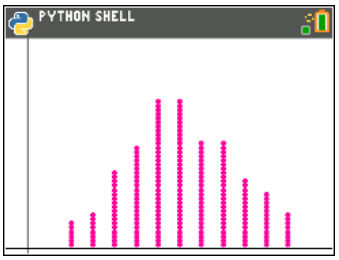
Unit 6: micro:bit — **Toepassing: Dobbelen**

In deze toepassing maak je een programma waarmee je met de micro:bit-data verzamelt en tegelijkertijd de grafiek daarvan ziet veranderen. Je gebruikt daarbij o.a. de module **ti_plotlib**.

Doelen:

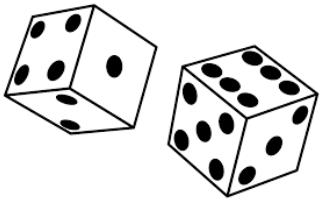
- Een programma schrijven voor het verzamelen van data met de BBC micro:bit;
- Een dynamische grafiek maken van de verzamelde data.

1. In deze toepassing gebruik je allerlei vaardigheden die in voorgaande units aan de orde kwamen:
   - Het gebruik van de knoppen op de micro:bit voor het verzamelen van data;
   - Het maken van een grafiek in Python;
   - Het exporteren van data naar een TI-84-lijst voor verdere analyse.



2. Met de micro:bit kun je het werpen met twee dobbelstenen simuleren waarbij de uitkomsten op het display verschijnen. De rekenmachine kan de grafiek van de som van de ogen dynamisch weergeven.

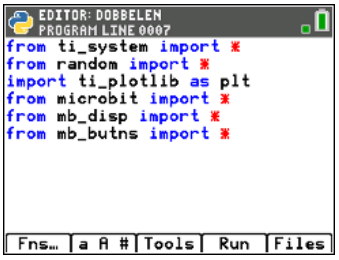
   We maken een programma dat het werpen met de dobbelstenen simuleert en dat de data uiteindelijk met **store_list()** naar de TI-84 exporteert.



3. Start een nieuw programma met de naam DOBBELEN.

   Je hebt een groot aantal Python-functies nodig, dus moet je hiervoor veel modules importeren:

   - **ti_system** voor **escape( )**
   - **random** voor **randint(,)**
   - **ti_plotlib** voor de Python-grafiek
   - **microbit** voor de micro:bit
   - **mb_disp** voor het micro:bit-display
   - **mb_butns** voor de micro:bit-knoppen

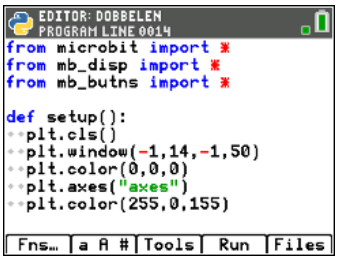


4. We gebruiken knop **A** om te dobbelen en knop **B** om de dataverzameling te 'resetten' voor een nieuwe sessie. Het is handig om een 'setup'-functie te maken voor het initialiseren van het grafiekenscherm.

   Begin het programma met **def setup( ):**

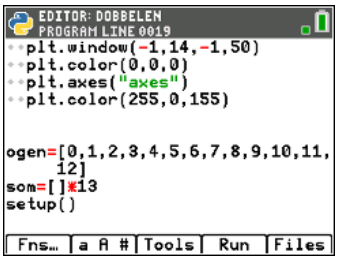
5. De functie 'setup' bevat de commando's voor het instellen van het assenstelsel en je kunt deze vinden met **[math] ti_plotlib**.

   Opmerking: De volgorde van de plt-functies is belangrijk, houd daarom de volgorde van het menu aan.
   *Het **color**-commando vind je in het **Draw**-submenu. We kiezen voor zwarte assen en paarse stippen.*



6. Het hoofdprogramma begint met het maken van twee lijsten:
   ogen = [0,1,2,3,4,5,6,7,8,10,11,12]
   som = [0]*13

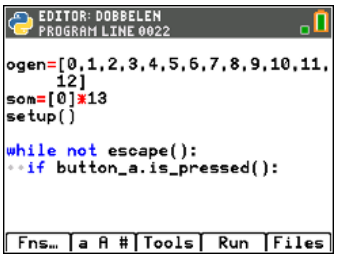
   ogen[0] en ogen[1] kunnen niet voorkomen maar dienen om de x-as bij 0 te laten beginnen.
   Om een scatterplot te maken heb je twee lijsten nodig.
   ogen *is de eerste lijst en* **som** *de tweede*



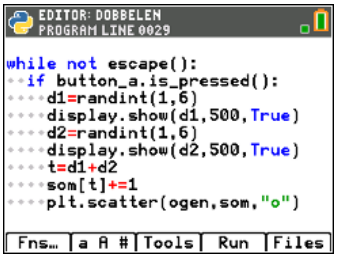
   Vervolgens wordt de al gedefinieerde functie **setup( )** aangeroepen om het grafiekenscherm te initialiseren.

7. Ga nu verder met een **while**-lus:
   while not escape():
   ♦♦ **if button_a.is_pressed():**

   Als knop A is ingedrukt moet het volgende gebeuren:
   - Twee keer een random getal kiezen;
   - Beide uitkomsten op het display tonen;
   - De som uitrekenen;
   - De lijst **som** updaten;
   - De scatterplot aanpassen.



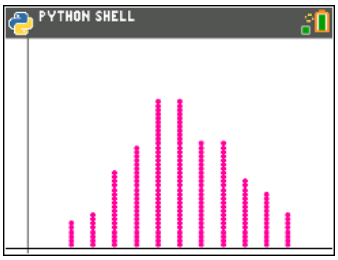
8. De code staat hiernaast.



9. Als je het programma uitvoert kun je knop A ingedrukt houden, want er wordt alleen getest of de knop **is** ingedrukt.

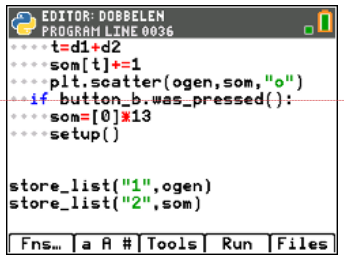
   Je kan het programma versnellen als je de twee display-commando's weglaat.

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



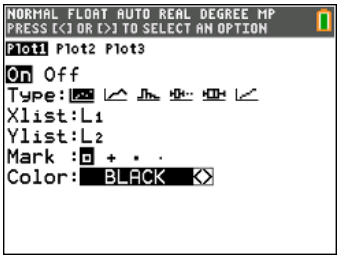
10. Je moet nog twee dingen doen:
    - Knop B moet de functie van 'reset' krijgen:
      Je moet de **setup( )** aanroepen waardoor je het scherm wist en de lijst **som** zich opnieuw vult met nullen.
      De lijsten moet je aan het eind exporteren naar de TI-84 voor verdere analyse. Hiervoor hebben we het **store_list(,)**-commando nodig.

    *In het programma hiernaast gebruiken we **store_list("1", ogen)** en **store_list("2", som)**. In de TI-84 zijn de lijsten dan opgeslagen in L1 en L2.*



Met opmerkingen [DN1]: Dit klinkt iets beter, maar zeggen we nu nog hetzelfde? Wordt deze dan automatisch gevuld met nullen?

11. **<Run>** het programma en druk een groot aantal keren op knop A. Stop dan het programma met **[clear]** en sluit de Python-app weer af ([2nd] **[quit]**). Maak vervolgens een scatterplot van de **L1**, **L2** -lijsten zoals hiernaast aangegeven.



12. Welk **mathematisch model** past bij deze plot?