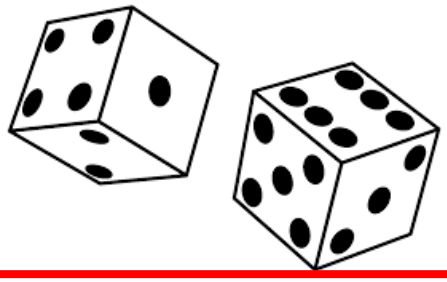
Unité 8 : micro:bit avec **Python** — **Application : Lancer les dés**

Dans cette application, vous allez écrire un programme pour collecter des données à l'aide du micro:bit et exécuter le programme tout en observant l'évolution d'une représentation graphique sur une page fractionnée de la TI-Nspire CX II.

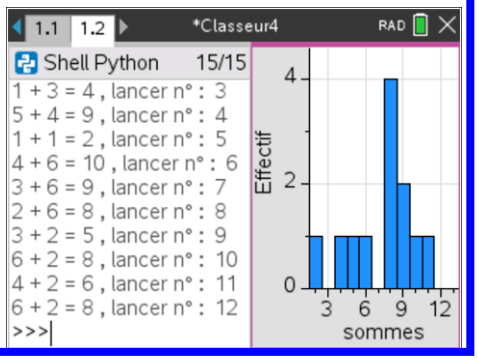
Objectifs :

- Écrire un programme de collecte de données micro:bit.
- Créer un diagramme dynamique (Données & Statistiques) des données collectées.

1. Ce projet est une compilation de toutes les compétences micro:bit que vous avez apprises dans les trois dernières compétences : écrire un programme qui utilise un geste, comme « shake » (ou une pression sur un bouton) pour collecter des données, stocker la liste comme une variable TI-Nspire ...etc.



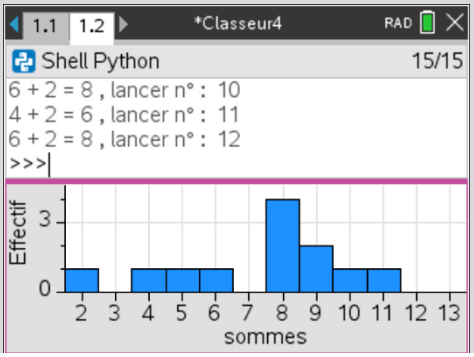
2. ... puis mettre cela en place sur une page TI-Nspire.
   - Exécuter le programme Python d'un côté de l'écran (le shell Python).
   - Afficher un nuage de points (ou un histogramme) des données collectées pendant l'exécution du programme (utilisation d'une application Données & Statistiques).



Conseil de l'enseignant : La disposition de l'écran partagé TI-Nspire ci-dessus peut être verticale ou horizontale. Lorsque vous appuyez sur **ctrl-4** pour regrouper deux applications sur une seule page, la valeur par défaut est verticale comme dans la leçon de l'élève. **Pour passer à la disposition horizontale sur la calculatrice, appuyez sur [doc] > Mise en page** > Sélectionner la mise en **page** > mise en **page 3.**

Vous pouvez également ajuster la barre de séparation pour rendre l'application Python Shell plus petite et l'application Données & Statistiques plus grande.

Cette leçon produit un diagramme en bâtons, pas un histogramme. Les instructions de conversion d'un diagramme en bâtons en histogramme dans l'application Données & Statistiques se trouvent plus loin dans les notes de l'enseignant de ce document.



3. Commencer le programme micro:bit avec les importations habituelles, y compris le module **aléatoire** et commencer par une liste vide appelée **sommes** :

   sommes = [ ]

   Stocker immédiatement cette liste dans une variable TI-Nspire (portant le même nom).

   store_list(« sommes », sommes)

   Ainsi la liste TI-Nspire est également effacée.

   print( ) quelques instructions à l'utilisateur avant le début de la boucle. Nous allons utiliser le geste « secouer (shake)» pour lancer les dés.

4. Dans le corps de la boucle **while,** utiliser le geste pour :
   - **lancer** deux dés (générer deux entiers aléatoires) ;
   - faire la somme des nombres obtenus ;
   - **ajouter** la somme à la liste **sommes** ;
   - **afficher** les deux valeurs de dés, leur somme et le numéro de lancer sur l'écran TI-Nspire. Astuce : **len(sommes)** est le numéro de lancer ;
   - **afficher** les deux valeurs sur la matrice de DEL de la carte micro:bit ;
   - **stocker** la **liste** dans une variable TI-Nspire.

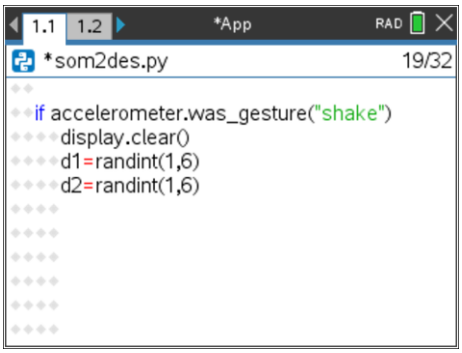
   Essayez-le maintenant.

Conseil à l'enseignant : Les élèves doivent avoir suffisamment de connaissances des compétences 1, 2 et 3 pour développer ce programme. S'ils ont du mal avec les menus, encouragez-les à revenir vers les compétences précédentes.

5. Pour effectuer le lancer des dés, utilisez un geste ou appuyez sur un bouton :

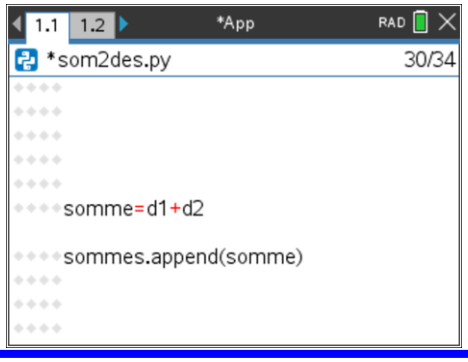
```
    If accelerometer.was_gesture(« shake »):
        display.clear()
        d1 = randint(1,6)
        d2 = randint(1,6)
```

Encore une fois, notez les indentations.



6. Additionner les numéros sortis et **ajoutez** la **somme** à la liste des sommes :

```
        somme = d1 + d2
        sommes.append(somme)
```



7. Afficher les deux dés sur l'écran de la carte micro:bit. N'oubliez pas que les deux dés peuvent avoir la même valeur, nous voulons donc nous assurer que les deux apparaîtront réellement :

```
        display.clear()
        display.show(d1)
        sleep(250)
        display.clear()
        display.show(d2)
        sleep(250)
```

Vous préférerez peut-être un délai plus long comme argument des instructions **sleep( ).**
Si vous avez entré le code correctement et dans le *bon ordre, essayez d'exécuter le programme maintenant et secouez la carte micro:bit. Vous devriez voir deux nombres affichés sur la micro:bit.*

Conseil de l'enseignant : une autre instruction **sleep( )** peut être utile dans la boucle pour donner à la micro:bit le temps de surveiller le geste.

8. Ajouter un code pour afficher la sortie des dés, la somme et les numéros des lancers sur l'écran de la calculatrice TI-Nspire. Vous pouvez utiliser une seule instruction **print()** comme ceci:

♦♦♦♦ **print (d1, "+", d2,"=",somme, ",","lancer n° =", len(sommes))**

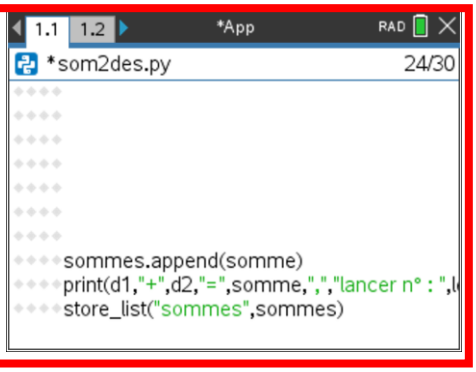
qui a comme conséquence l'affichage des lignes dans cette image.

Attention à la ponctuation !

9. Stocker les **sommes** de la liste Python en une liste TI-Nspire du même nom :

♦♦♦♦**store_list("sommes",sommes)**

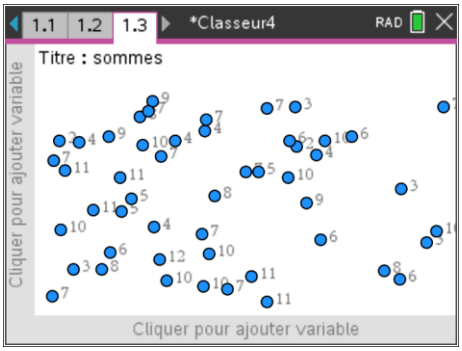
L'instruction **store_list( )** *est incluse à l'intérieur des blocs* **while** *et* **if** *afin que la liste TI-Nspire soit mise à jour chaque fois qu'une nouvelle paire de dés est générée.*



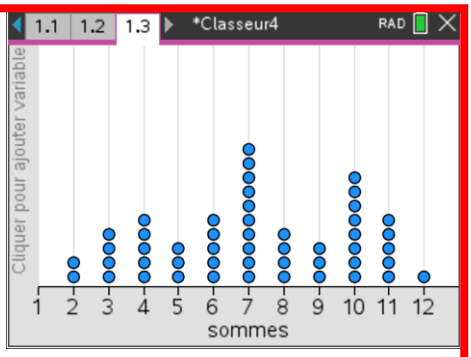
Conseil à l'enseignant : notez l'absence d'une variable « compteur » dans l'énoncé affiché ci-dessus. Il n'est pas nécessaire puisque le programme utilise simplement **len(sommes)** comme nombre de lancers.

10. Lorsque vous êtes convaincu que votre programme fonctionne correctement, vous êtes prêt à l'utiliser avec les possibilités de représentation graphique de la calculatrice TI-Nspire. Exécutez votre programme et générez environ 50 lancers. Appuyez sur **[esc]** pour terminer le programme.

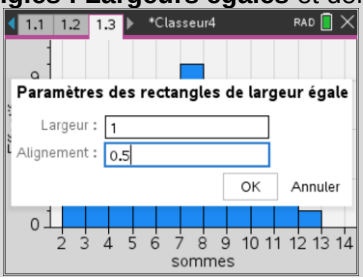
Dans le shell Python (à l'invite de commande **>>>**) appuyez sur **[ctrl] [doc]** ou **[ctrl] [I]** pour insérer une page. Sélectionnez l'application **Données & statistiques**. Vous devriez voir un écran similaire à celui de droite. Vos données de sommes sont dispersées autour de l'écran.



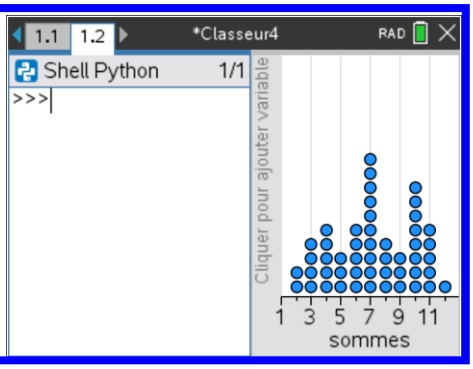
11. Cliquer sur le message « Cliquez pour ajouter une variable » en bas de l'écran et sélectionnez la liste des sommes des **variables**. Vos points dispersés sont maintenant organisés le long de l'axe des *x* en fonction de leur valeur et la fenêtre est adaptée aux données. Il s'agit d'un **nuage de points**.



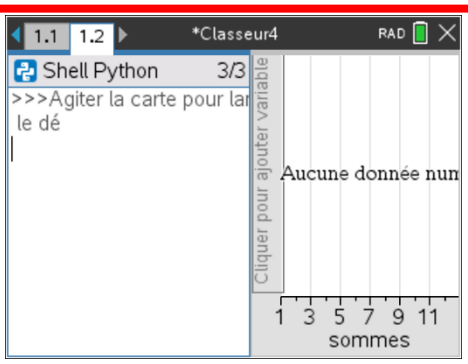
Conseil de l'enseignant : il est possible de changer le tracé en un histogramme : appuyez sur **[menu] > Type de tracé > Histogramme**. Mais vous devez également ajuster l'alignement de l'axe des abscisses à 0,5 afin que les barres soient centrées sur leurs valeurs d'axe *x*. *Appuyez sur* **[menu] > Propriétés du tracé > Propriétés de l'histogramme > Réglages des rectangles : Largeurs égales** et définissez **l'alignement sur 0,5**.



12. Reculer d'une page à l'application Python Shell (**[ctrl] [flèche gauche]**) et appuyer sur **[ctrl] [4]** pour « regrouper » cette application avec l'application Données & statistiques, en créant une page en écran partagé avec votre Python Shell à gauche et votre application Données &Statistiques à droite, comme indiqué ici.



13. Le Shell a été « réinitialisé » en appuyant sur **[ctrl] [R]** et le programme ne se réexécutera pas seul. Retourner à l'éditeur Python et appuyer sur **[ctrl] [R]** pour exécuter le programme. Il fonctionne dans la moitié-écran Shell comme indiqué ici. Vous ne voyez aucune donnée numérique sur la droite parce que le programme stocke une liste vide tout de suite.



Lorsque vous collectez les données (secouez la carte micro:bit pour lancer les dés), les valeurs des sommes apparaissent sous forme de nuage de points dans l'application Données & Statistiques sur la droite.

L'appui sur **[esc]** mettra fin au programme et vous pouvez faire beaucoup d'autres analyses de données à une variable dans l'environnement TI-Nspire.

Appuyer à nouveau sur **[ctrl] [R]** maintenant (dans le shell Python) exécutera de nouveau le programme.

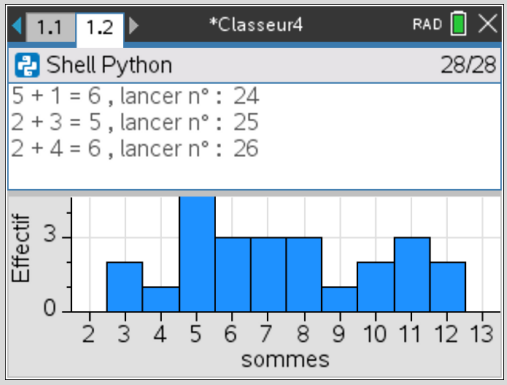
Astuce : pour effacer le Shell au début de chaque exécution, ajoutez l'instruction :

clear_history()

que l'on trouve *dans* **[menu] > Plus de modules > BBC micro:bit > Commandes** *au début de votre programme.*

Profitez-en et n'oubliez pas d'enregistrer votre document !

Conseil à l'enseignant : La disposition de l'écran partagé TI-Nspire peut être verticale ou horizontale. Lorsque vous appuyez sur ctrl-4 pour fusionner deux applications sur une seule page, la valeur par défaut est verticale comme on le voit dans la leçon de l'élève. Pour passer à la disposition horizontale sur la calculatrice, appuyez sur [doc] > Mise en page > Sélectionner la mise en page > la mise en page 3. Vous pouvez également ajuster la barre de séparation pour réduire la taille de l'application Python Shell :



Pour remplacer le tracé du nuage de points par un histogramme, appuyez sur [menu] > Type de tracé et sélectionnez « Histogramme ».

Exemple de solution :

som2des.py 1/23

```
from random import *
from microbit import *
clear_history()
sommes=[]
store_list("sommes",sommes)
print("Agiter la carte pour lancer le dé")

while get_key() != "esc":
    if accelerometer.was_gesture("shake") or button_a.was_pressed():
        display.clear()
        d1=randint(1,6)
        d2=randint(1,6)
        display.clear()
        display.show(d1)
        sleep(250)
        display.clear()
        display.show(d2)
        somme=d1+d2
        sleep(250)
        sommes.append(somme)
        print(d1,"+",d2,"=",somme,",","lancer n° : ",len(sommes))
        store_list("sommes",sommes)
```

Extension optionnelle : afficher les images des faces d'un vrai dé.

Il est facile de concevoir des images personnalisées à afficher sur la carte micro:bit. Le code suivant crée les six faces du dé (modèles ou pips). Notez la majuscule **I** dans Image**.**

Taper le premier bloc, **un=Image(...**, puis copier/coller/éditer les cinq autres faces. Ensuite, créer la liste des faces_images, en utilisant None pour l'élément #0 afin que les valeurs de la matrice correspondent aux index de la liste. Python interprète deux chaînes écrites sur des lignes séparées sans délimiteur (comme une virgule) comme une seule chaîne, donc

« aaa »
« bbb »

est identique à

« aaabbb »

Dans le code suivant, les 5 lignes de la carte micro:bit sont dupliquées dans la fonction **Image( )** pour faciliter la conception d'une image. La valeur de chaque nombre de l'image peut être comprise entre 0 et 9 pour contrôler la luminosité de chaque LED.

```
#####################################################################
from microbit import *
from random import *

un=Image(
"00000:"
"00000:"
"00900:"
"00000:"
"00000")

deux=Image(
"00000:"
"09000:"
"00000:"
"00090:"
"00000")

trois=Image(
"00000:"
"09000:"
"00900:"
"00090:"
"00000")

quatre=Image(
"00000:"
"09090:"
"00000:"
"09090:"
"00000")

cinq=Image(
"00000:"
"09090:"
"00900:"
"09090:"
"00000")

six=Image(
"00000:"
```

```
"09090:"
"09090:"
"09090:"
"00000")

faces_images=[Aucun, un, deux, trois, quatre, cinq, six]
# indice: 0 1 2 3 4 5 6
print(« en cours d'exécution... »)
while get_key() != « esc »:
  If button_a.is_pressed():
   de = randint (1,6)
    display.show(faces_images[de]) # afficher l'un des images des dés en utilisant des modèles (pips).
```