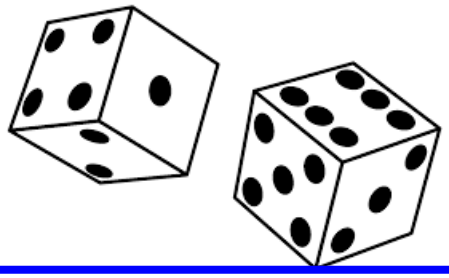
Unité 7 : micro:bit avec Python — Application : Lancer de dés

Dans cette application, vous allez écrire un programme pour collecter des données à l'aide du micro:bit et exécuter le programme puis utiliser le mode radio de la carte BBC micro:bit afin de transférer ces données à une autre calculatrice.

Objectifs :

- Écrire un programme de collecte de données à partir de la carte micro:bit.
- Créer un diagramme de représentation des données après les avoir classées.
- Transférer les données à une autre calculatrice.

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, par l'intermédiaire de l'accéléromètre (ou une pression sur un bouton) pour collecter des données, stocker la liste comme une variable de la TI-83 Premium CE ...etc.



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

   sommes = [ ]

   Stockez immédiatement cette liste dans une liste de la calculatrice, $\mathbf{L_2}$ par exemple.

   store_list(« 2 », sommes)

   Ainsi la liste **sommes** est également effacée.

   print( ) quelques instructions à l'utilisateur avant le début de la boucle. Nous allons utiliser le bouton **A** pour simuler un lancer de dés.

ÉDITEUR : SOM2DES
LIGNE DU SCRIPT 0011

```
from ti_system import *
from random import *
from microbit import *
from mb_disp import *
from mb_butns import *
sommes=[]
store_list("2",sommes)
print("appuyer sur A pour lancer
    )
while not escape():
```

Fns… | a A # | Outils | Exéc | Script

3. 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-83 Premium CE. 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-83 Premium CE.

   Essayez-le maintenant.

ÉDITEUR : SOM2DES
LIGNE DU SCRIPT 0016

```
sommes=[]
store_list("2",sommes)
print("appuyer sur A pour lancer
    ")
while not escape():
  if button_a.was_pressed():
    display.clear()
    d1=randint(1,6)
    d2=randint(1,6)
```

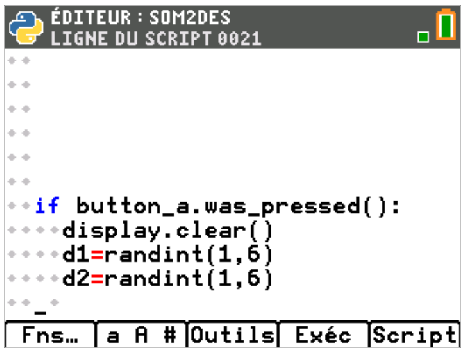
Fns… | a A # | Outils | Exéc | Script

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.

4. Pour effectuer le lancer des dés, utilisez appuyez sur un bouton :

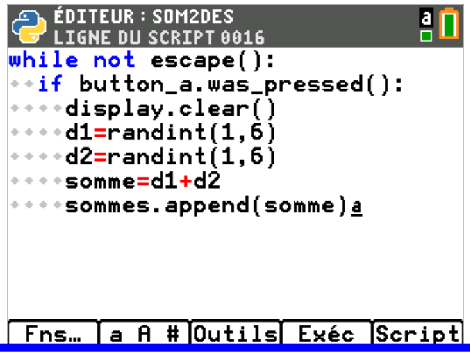
```
    If button_a.was_pressed( ):
        display.clear()
        d1 = randint(1,6)
        d2 = randint(1,6)
```

Encore une fois, notez les indentations.



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

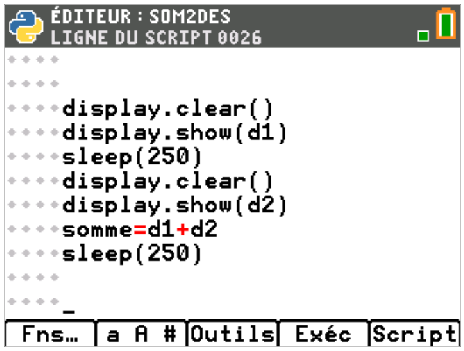
```
        somme = d1 + d2
        sommes.append(somme)
```



6. 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)
        somme=d1+d2
        sleep(250)
```

Vous préférerez peut-être un délai plus long pour les commandes **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.*



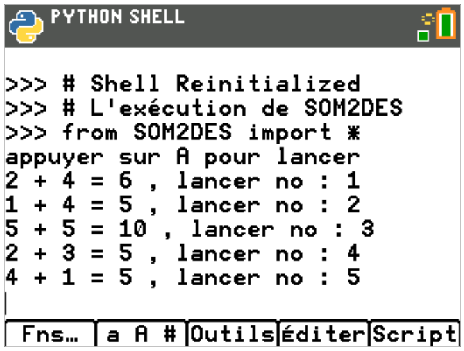
Conseil de l'enseignant : une autre instruction **sleep( )** peut être utile dans la boucle pour donner à la carte micro:bit le temps de surveiller l'appui sur le bouton.

7. Ajouter un code pour afficher la sortie des dés, la somme et les numéros des lancers sur l'écran de la calculatrice TI-83 Premium CE. 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.

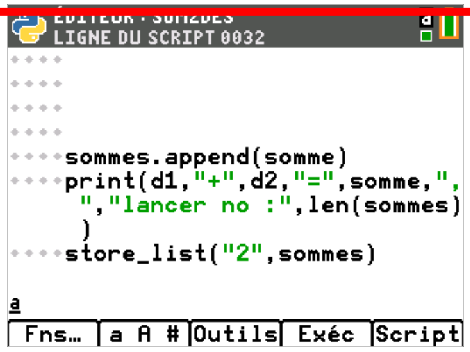
Attention à la ponctuation !



8. Stocker les **sommes** de la liste Python en une liste TI-83 Premium CE L_2 par exemple :

♦♦♦♦**store_list("2",sommes)**

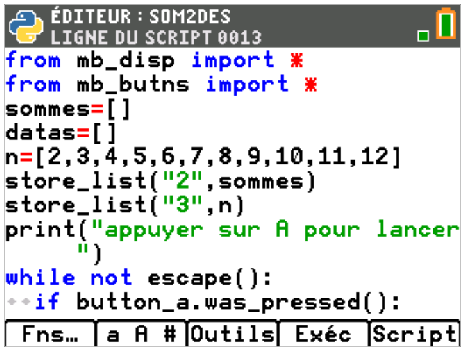
L'instruction **store_list( )** *est incluse à l'intérieur des blocs* **while** *et* **if** *afin que la liste TI-83 Premium CE soit mise à jour chaque fois qu'une nouvelle paire de dés est générée.*



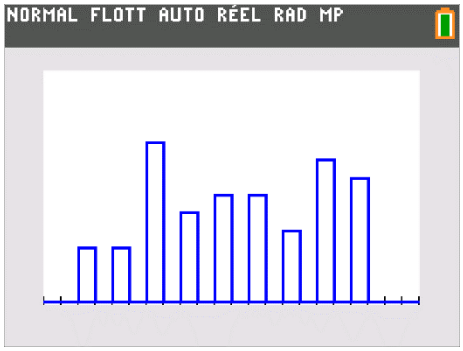
Conseil à l'enseignant : notez l'absence d'une variable « compteur » dans l'énoncé affiché ci-dessus. Il n'est pas nécessaire puisque le programme utilise simplement **len(sommes)** comme nombre de lancers.

9. 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-83 Premium CE. Exécutez votre programme et générez environ 50 lancers. Appuyez sur **[annul]** pour terminer le programme.

Les données ne sont pas classées. Complétez votre script par une partie de code permettant de lire la liste **sommes**, puis de compter les données à l'aide de la méthode **.count( ),** correspondant à une somme (de 2 à 12). Les résultats seront stockés dans une liste **datas( )**, qui est ensuite exportée vers la calculatrice en L_4.
Remarque : pour plus de facilités, les valeurs possibles des sommes sont stockées dans une liste **n** également exportée vers la calculatrice en L_3.
Ne pas oublier d'initialiser les listes au début du programme.



10. Effectuer ensuite la représentation graphique des données sous la forme d'un diagramme en bâtons.



11. Exemple de solution :

Le script ci-après est un exemple de solution. Vérifier que le vôtre est similaire, puis l'exécuter avant de passer à la partie suivante.

ÉDITEUR : SOM2DES
LIGNE DU SCRIPT 0001

```
# Simulation Aléatoire_
from ti_system import *
from random import *
from microbit import *
from mb_disp import *
from mb_butns import *
sommes=[]
datas=[]
n=[2,3,4,5,6,7,8,9,10,11,12]
store_list("2",sommes)
store_list("3",n)
print("appuyer sur A pour lancer
    ")
while not escape():
  if 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 no :",len(sommes)
      )
    store_list("2",sommes)
#classement des données
for i in range(1,13):
  k=sommes.count(i)_
  datas.append(k)
  store_list("4",datas)
```

Fns… | a A # | Outils | Exéc | Script

12. Communiquer avec une autre calculatrice.

On se propose maintenant de communiquer les données (résultats des lancers) à une autre carte BBC micro:bit.

Le module radio permet d'envoyer ou de recevoir des messages vers ou depuis une ou plusieurs cartes microbit.

La communication se fait sans fil sur une fréquence entre 2,4GHz et 2,5Ghz selon le **canal** choisi (numéroté entre 0 et 83).

Les messages ont une taille maxi de 251 caractères. Par défaut, la taille est fixée à 32 caractères.

Pour utiliser la radio, vous devez avoir 2 cartes micro:bit et 2 calculatrices. La communication radio est très simple à mettre en œuvre comme vous pourrez le voir.

a) Initialiser la communication radio sur les deux calculatrices

from mb_radio import*

radio.on( )

Par défaut, la radio est toujours désactivée pour des raisons d'économie d'énergie.

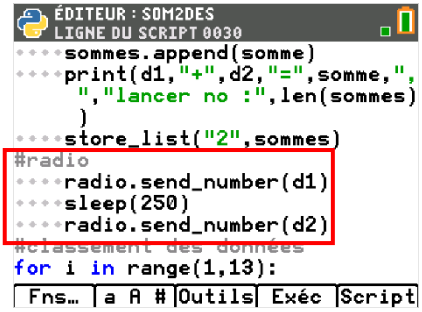
La communication peut alors s'établir entre plusieurs cartes.

b) Envoyer et recevoir

Pour envoyer un message sous forme d'une chaîne de caractères, utiliser l'instruction : **radio.send(« message »)** pour une chaîne de caractères et **radio.send(number)** pour un nombre.

Calculatrice émettrice :

Lorsque le bouton **A** a été pressé, la carte BBC micro:bit envoie les résultats des lancers à la calculatrice réceptrice. Il suffit de rajouter la petite portion de code ci-contre.



Calculatrice réceptrice :

Créer un nouveau script portant le même nom et importer les modules **ti_system, microbit, mb_disp** et **mb_radio.**

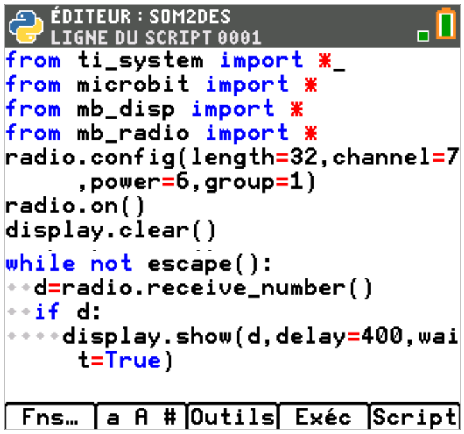
Configurer la liaison radio de manière identique à la calculatrice émettrice.

Effacer l'écran de la matrice **display.clear( )**. Cette instruction se trouve dans le module d'affichage **mb_disp( ).**

Créer une boucle ouverte (while) mettant la calculatrice en attente de la réception d'un nombre **d**. Cela se réalise à l'aide de l'instruction :

var = radio.receive_number( )

Si un nombre est reçu, celui-ci est affiché avec un délai modifiable de 400 ms.



c) Exécution des scripts.

Exécuter les scripts sur chaque calculatrice. La calculatrice réceptrice est en attente des informations de la calculatrice émettrice.

Lorsque l'on appuie sur le bouton **A** du micro:bit de la carte émétrice, l'affichage est transféré sur la carte réceptrice.

Conseil de l'enseignant : afin d'éviter les conflicts de communication entre plusieurs cartes et au sein d'une même classe, il est conseillé de configurer le groupe de radio et ou le canal en utilisant l'instruction :

radio.config(lengh=32, channel=7, power=6, groupe=0)

Avec :

- **lengh :** longueur maximale d'une chaîne de caractères de 0 à 251 caractères.
- **channel :** n° du canal de transmision de 0 à 83.
- **power :** force du signal de 0 à 7
- **group :** de 0 à 255

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 not escape( ):
 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).
```