Unité 7 : micro:bit avec Python **Compétence 3 : Autour de la lumière**

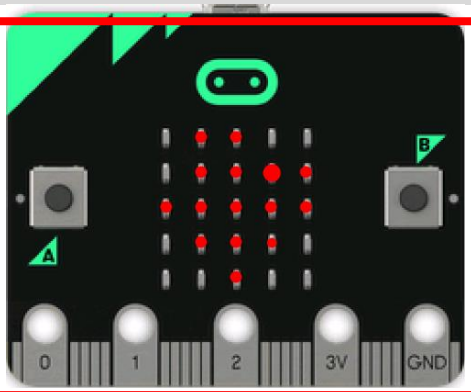
Dans cette leçon, vous allez observer le fonctionnement du capteur de lumière sur la carte micro:bit et stocker les données dans une liste de la calculatrice TI-83 Premium CE pour une analyse plus approfondie.

Objectifs :

- Lire et afficher les mesures du capteur de luminosité sur la carte micro:bit.
- Transférer des données de Python vers la deTI-83 Premium CE.
- Étudier les données collectées à partir de la carte micro:bit.

Conseil de l'enseignant : La leçon se rapproche de la loi inverse du carré de la distance pour l'éclairement mesuré à partir d'une source de lumière ponctuelle. On peut établir une relation entre la distance de la source lumineuse et l'intensité lumineuse. Mais un déplacement régulier et constant de la carte micro:bit s'éloignant en fonction du temps de la source lumineuse permettra d'établir une relation linéaire entre le temps et la distance.

1. La carte micro:bit peut lire le niveau de luminosité ambiante à l'aide des LED d'affichage. Oui, les LED d'affichage peuvent également être utilisées comme des périphériques d'entrée !



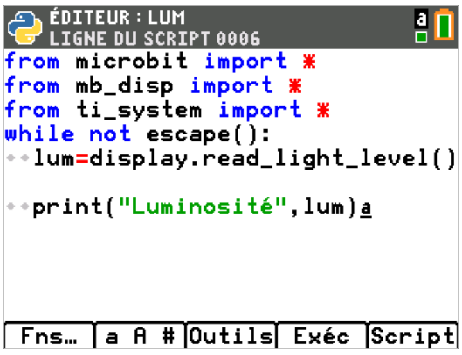
Conseil de l'enseignant : Pour plus d'informations sur le niveau de lumière mesuré avec la carte micro:bit voir :
Détecter les changements de luminosité sur le micro:bit : Aide &Support

2. Démarrer un nouveau programme Python dans un nouveau document.
   Appuyer sur la touche **[prgm]** et sélectionner **2 : Python App >** puis **[F3] > Nouv**.
   Nous appelons le programme **LUM.**

   Pour vérifier les valeurs que le capteur de lumière peut détecter, écrivez le programme court suivant à l'aide des menus BBC micro:bit et après l'importation des librairies **microbit, mb_disp** et **TI_system** :

```
while not escape( ) :
    lumi = display.read_light_level( )
    print(« Luminosité » ,lum)
```

   Vous trouverez **var = .read_light_level()** sur **[Fns] < Modul >9 : Affichage.**



Taper le nom de la variable « lum », puis rechercher ensuite l'instruction « **display.read_light_level( )** ».

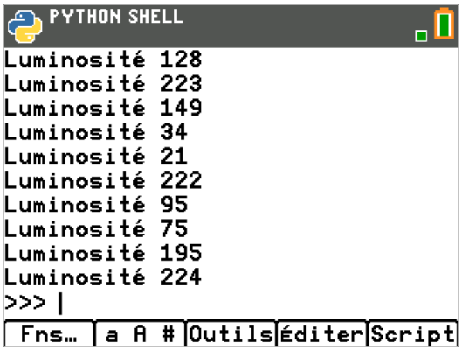
Rappel : **[Fns] < Modul > 4 : TI_System > 5 : while not escape :**

N'oubliez pas d'indenter les deux dernières instructions afin qu'elles soient incluses dans le corps de la boucle **while**.

3. Exécuter le programme et pointer l'affichage de la carte micro:bit vers une source lumineuse. Peu importe ce qui s'affiche sur l'écran. Déplacer la carte micro:bit vers puis loin de la source et observer les valeurs sur l'écran de la calculatrice TI-83 Premium CE. Vous devez obtenir des valeurs entre 0 et 255.

Vous vous en doutez probablement : plus la source lumineuse est éloignée, plus la valeur du niveau de lumière est faible. Maintenant, vous allez ajouter du code au programme pour collecter les données du niveau de luminosité afin de créer un nuage de points de la luminosité par rapport au temps.

Appuyer **sur [annul]** pour terminer le programme et revenir à l'éditeur.



4. Créer deux listes vides avant votre boucle **while** :

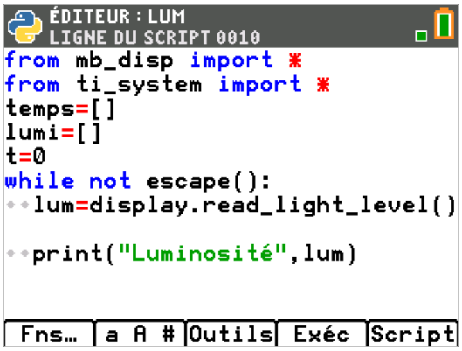
temps = [ ]

lumi = [ ]

Trouvez les crochets sur le pavé numérique **2nde** **[**×**]**[**−**] ou sur **[Fns] > List > 1 : [ ]**.

Toujours avant la boucle **while**, une instruction pour initialiser une variable du compteur temps (**t**) à 0 : **t = 0**

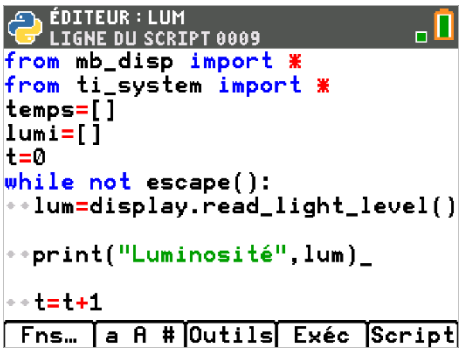
Évitez d'utiliser le mot « time » comme variable, car il existe un module temporel qui utilise ce nom. En outre, c'est une bonne habitude de pluraliser les noms de liste car ils contiennent de nombreuses valeurs.



5. Dans le corps de la boucle, après l'instruction **print,** ajouter une instruction pour incrémenter la variable de temps **t**. Nous utiliserons un intervalle de temps d'une seconde entre les lectures de lumière.

♦♦ **t = t + 1**

Remarque : On peut aussi écrire **t+=1**, *ce qui est plus élégant en Python.*

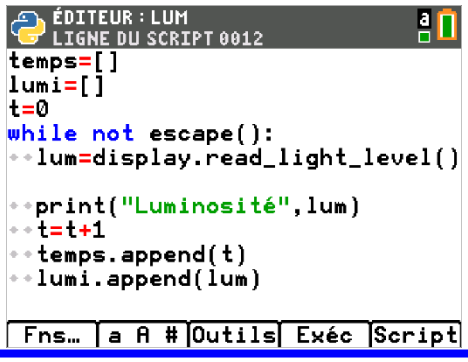


6. Ajoutez les valeurs de **lumi** et **t** à leurs listes respectives à l'aide des instructions suivantes :

   - **temps.append(t)**
   - **lumi.append(lum)**

   La méthode .**append( )** se trouve **dans [Fns] > List**.

   Ces instructions ajoutent la valeur de la luminosité actuelle et la valeur t (instants) aux listes.



7. Pour contrôler l'intervalle de synchronisation de l'échantillonnage, ajouter :

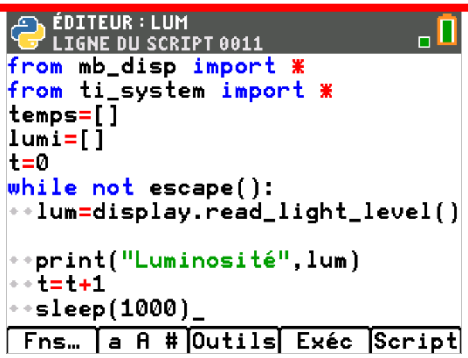
   - **sleep(1000)**

   Remarque : L'instruction **sleep( )** *se trouve dans le module* **time**, *mais aussi dans le menu* **ti_system.**

   Cela interrompt la collecte de données pendant une seconde entre les échantillons. **sleep( )** se trouve sur :

   [Fns] < Modul >4 : ti_system > A : sleep( )

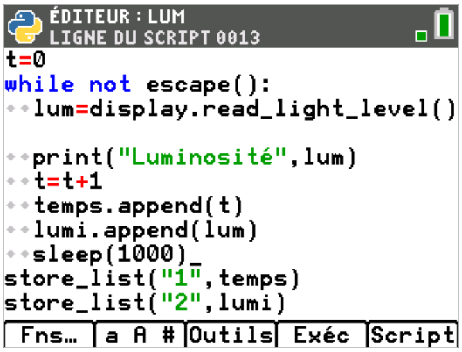
   Rappelez-vous que lors de l'utilisation de la carte micro:bit, **sleep( )** utilise des *millisecondes comme argument. L'échantillonnage a lieu une fois par seconde.*



8. Après le corps de la boucle (*plus d'indentation !*), **stocker** les deux listes Python dans deux listes TI-83 Premium CE en utilisant éventuellement les mêmes noms dans les deux environnements : commencer au début d'une nouvelle ligne (pas d'espace, ni d'indentation !).

   Trouver **store_list()** dans **[Fns] < ti_system > 3 : store_list(« nom »,var)**. Il faut deux arguments : le « nom de la liste TI-83 Premium CE » entre guillemets et celui de la liste Python sans guillemets.

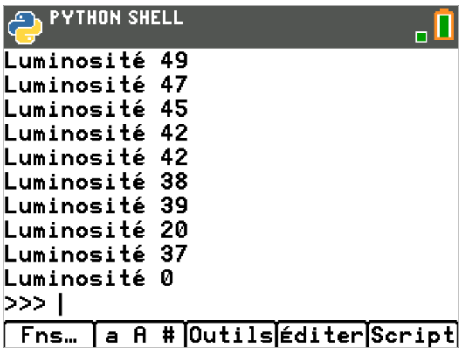
   Dans ce programme, les listes *Python* ne sont stockées dans les *listes TI-83 Premium CE juste qu'à la toute fin du programme,* lorsque vous *appuyez sur* **[annul]** pour quitter la *boucle while.*

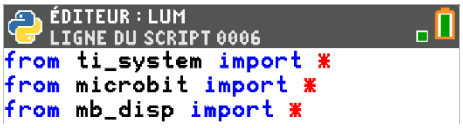


9. Exécuter le programme. Commencer par placer la carte micro:bit proche de votre source de lumière. Une ampoule exposée ou une lampe de smartphone fonctionne bien. Déplacer lentement mais régulièrement le micro:bit loin de la lumière à une vitesse uniforme jusqu'à ce que la lecture de la luminosité soit inférieure à 10.
   Appuyer **sur [annul]** pour terminer le programme.

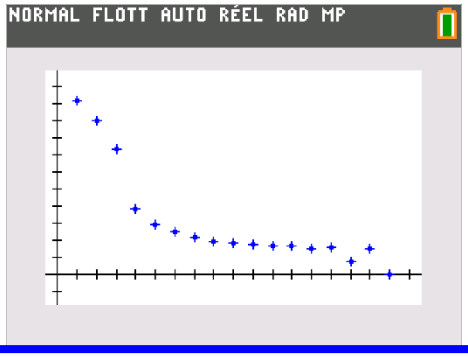
   Répéter le processus jusqu'à ce que vous pensiez avoir obtenu de « bonnes » mesures.

   Remarque : Si le programme ne fonctionne pas ou reste bloqué, modifiez l'ordre de chargement des librairies en plaçant le module ti_system en premier.



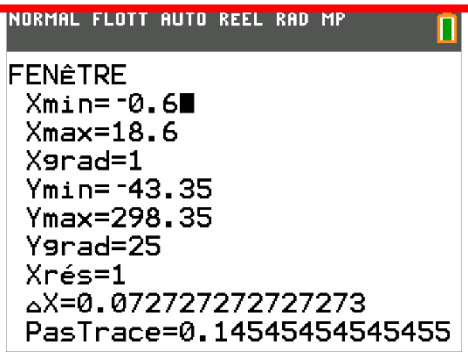


10. Lorsque le programme se termine, quitter l'application Python et effectuer la représentation graphique des données, sous la forme d'un nuage de points.



Vous devrez peut-être exécuter le programme *plusieurs fois* pour obtenir une belle courbe lisse. Vous devrez peut-être également ajuster la fenêtre d'affichage lorsque vous utilisez des données différentes.

Vous pouvez également ajuster *l'intervalle de temps entre les échantillons, mais veillez à modifier à la fois l'argument* **sleep(1000)** et la valeur du compteur *(t = t + 1).*



Que remarquez-vous ?

- *Quelle courbe observez-vous ? Quel modèle mathématique correspond à ce phénomène physique ?*
- *Utilisez les outils d'analyse de données de la TI-83 Premium CE pour déterminer un modèle mathématique adapté à vos données.*

Conseil de l'enseignant : La leçon étudie la loi inverse du carré en fonction de la distance pour l'éclairage à partir d'une source de lumière, mais la loi est une relation entre la distance de la source lumineuse et l'intensité lumineuse, pas le temps.

Intensité = k / (distance2)**

Un déplacement lent et constant de la carte micro:bit ou de la source lumineuse établit une relation linéaire entre le temps et la distance, de sorte que la proportion carrée inverse est toujours applicable.

Vous pouvez modifier l'activité pour utiliser une mesure de distance entre la source lumineuse et le micro:bit et utiliser une pression sur un bouton pour collecter chaque point de données. Par exemple, éloignez le micro:bit de 10 cm de la lumière à chaque moment de la collecte. Utilisez **disp_wait( )** pour suspendre le programme jusqu'à ce que le micro:bit soit positionné correctement pour l'échantillon suivant.