Unité 8 : micro:bit avec Python **Compétence 3 : Autour de la lumière**

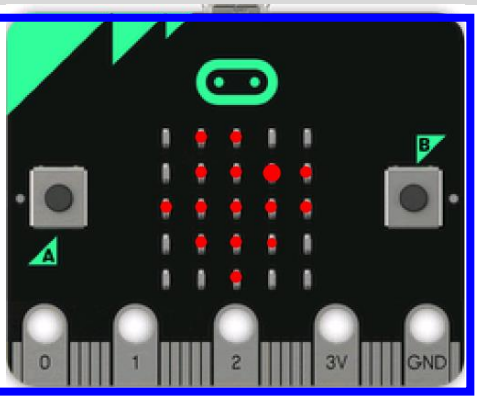
Dans cette leçon, vous allez observer le fonctionnement du capteur de lumière sur la carte micro:bit et stocker les données dans une liste de la calculatrice TI-Nspire pour une analyse plus approfondie.

Objectifs :

- Lire et afficher les mesures du capteur de luminosité sur le micro:bit.
- Transférer des données de Python vers TI-Nspire CX II.
- 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 la mesure de l'éclairement à partir d'une source de lumière ponctuelle. On peut établir une relation entre la distance de la source lumineuse et l'intensité lumineuse mesurée en un point. 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. En effet, 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 **[home]** et sélectionner **Nouveau >** puis **Ajouter Python >Nouveau...**
   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 **à partir de l'importation de la librairie microbit ***

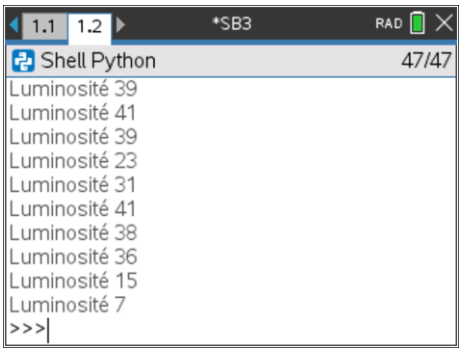
```
while get_key() != « esc »:
♦♦ lum = display.read_light_level( )
♦♦ print(« Luminosité » ,lum)
```

Vous trouverez **var = .read_light_level( )** sur **[menu]** > **BBC micro:bit > Sensors ; lum** sera le nom de la variable que vous tapez.

Rappel : **[menu] > Plus de modules > BBC micro:bit** > **Commands>** tant que **get_key() != « esc »:**

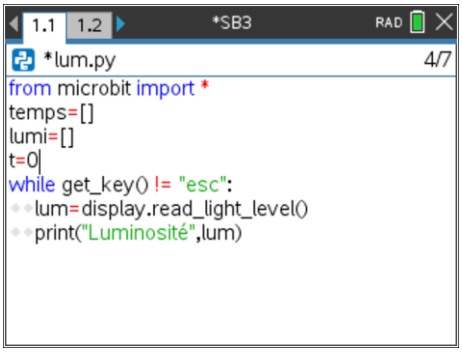
N'oubliez pas d'indenter les deux dernières instructions afin qu'elles soient incluses dans le corps de la boucle **while**.

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 le micro:bit vers puis loin de la source et observer les valeurs qui s'affichent sur l'écran de la calculatrice TI-Nspire CX II. Vous devez obtenir des valeurs entre 0 et 255.
Comme vous vous en doutez probablement, plus la source lumineuse est éloignée, plus la valeur du niveau de lumière reçu est faible. Maintenant, vous allez ajouter du code au programme pour collecter les données du niveau de luminosité pour créer un nuage de points de la luminosité par rapport au temps.



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

4. Créer deux listes vides avant votre boucle **while** :
temps = [ ]
lumi = [ ]
Trouver les crochets sur le pavé numérique **[ctrl] [parenthèse gauche]** ou sur **[menu] > Intégrés > Listes**



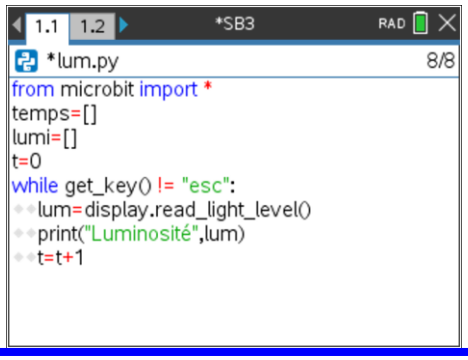
Toujours avant la boucle **while**, une instruction pour initialiser une variable du compteur temps. (**t**) à 0 : **t = 0**

*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 augmenter 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**

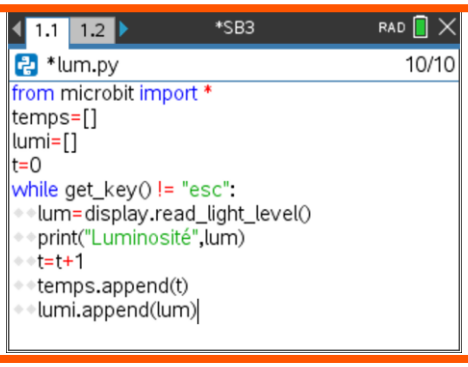


6. Ajoutez les valeurs de **lum** et **t** à leurs listes respectives à l'aide des instructions suivantes :

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

La méthode .**append( )** se trouve **dans [menu] > Intégrés > Listes**.

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



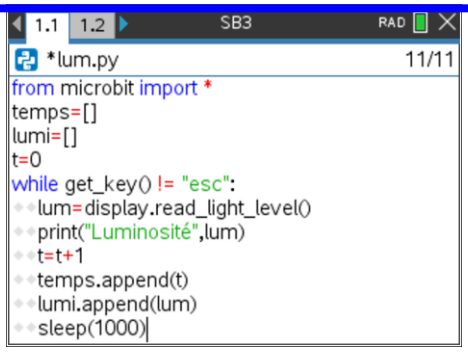
7. Pour contrôler l'intervalle de synchronisation de l'échantillonnage, ajouter :

♦♦**sleep(1000)**

après les deux ♦♦ ajouter des instructions. Cela interrompt la collecte de données pendant une seconde entre les échantillons. **sleep( )** se trouve sur :

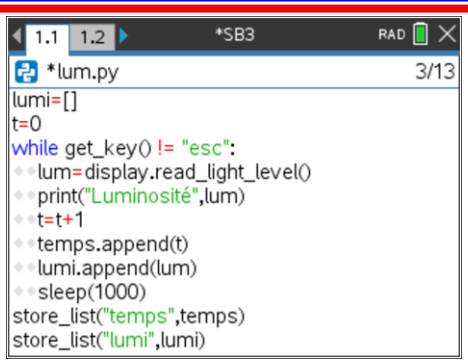
[menu] > commandes BBC micro:bit >

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-Nspire en utilisant éventuellement les mêmes noms dans les deux environnements : commencer au début d'une nouvelle ligne (pas d'espaces ni d'indentation !).
Trouver **store_list()** dans **[menu] > BBCmicro:bit > Commandes**. Il faut deux arguments : le « nom de la liste TI-Nspire » entre guillemets et celui de la liste Python sans guillemets. Suivez les invites des infos-bulles.

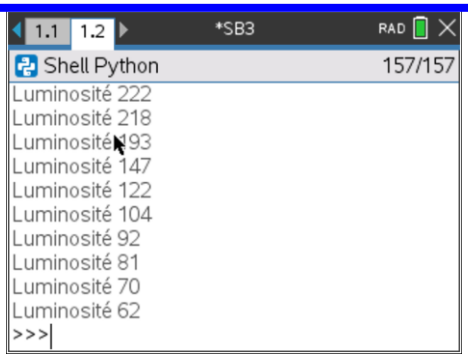
Dans ce programme, les listes Python ne sont stockées que dans les listes TI-Nspire juste à la toute fin du programme, lorsque vous appuyez sur **[esc]** pour quitter la boucle.



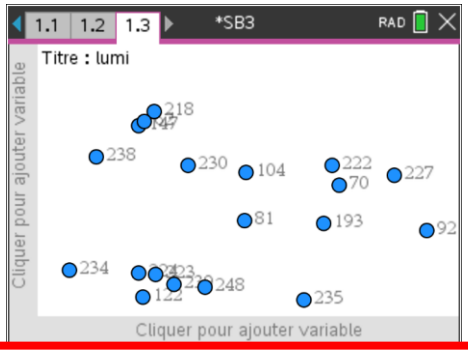
9. Exécuter le programme. Commencer par placer le micro:bit proche de votre source de lumière. Une ampoule exposée ou une lampe de poche pour smartphone fonctionne bien. Déplacer lentement mais régulièrement le micro:bit loin de la lumière avec une vitesse uniforme jusqu'à ce que la lecture de la luminosité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

   Répéter le processus jusqu'à ce que vous ayez l'impression d'avoir de « bonnes » données.

   Astuce : Une lampe de poche de téléphone fonctionne très bien.



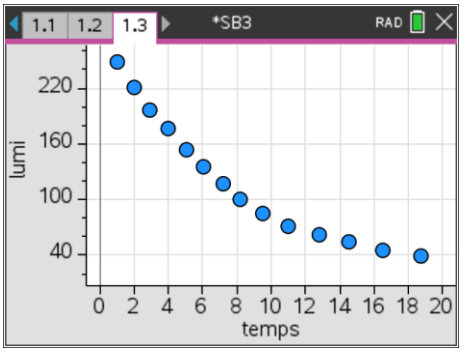
10. Lorsque le programme se termine, ajoutez une page à votre document en appuyant sur **[ctrl] [doc]**. Choisissez **« Données & Statistiques ».** Vous verrez une page similaire à cette image. L'une de vos listes apparaît avec les points dispersés autour de l'écran. Il faut maintenant organiser la représentation graphique de ces données.



11. Lorsque l'invite s'affiche au bas de la page « *Cliquez pour ajouter une variable* » et sélectionnez la liste des **instants**. Il s'agit de la variable indépendante.
    Cliquer sur l'invite « *Cliquez pour ajouter* une variable » sur le côté gauche de l'écran et sélectionner la liste **lumi.** Il s'agit de la variable dépendante. Cette action organise votre nuage de points comme l'image ci-contre.
    Vous devrez peut-être exécuter le programme *plusieurs fois* pour obtenir une belle courbe lisse comme celle illustrée ici. 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-Nspire 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 et non le temps.

Intensité = k / (distance2)**

Un déplacement lent et constant du micro:bit 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 une touche 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 **get_key(1)** pour suspendre le programme jusqu'à ce que le micro:bit soit positionné correctement pour l'échantillon suivant.