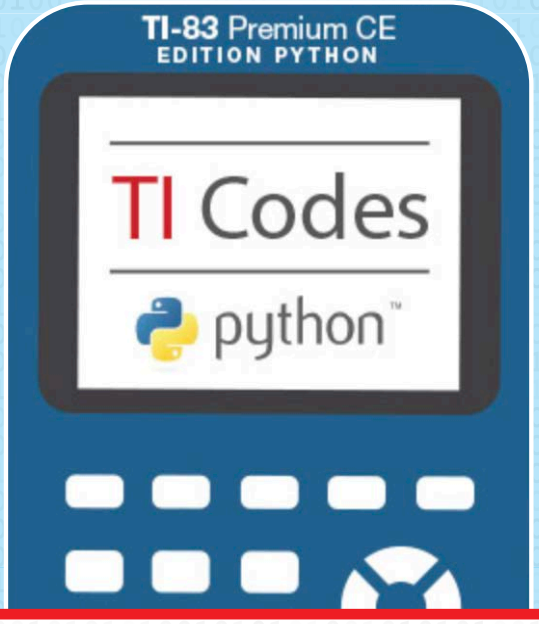
Modules complémentaires de la TI-83 Premium CE Edition Python



unité 7 → unité 8

Auteur : Jean-Louis Balas

Texas Instruments

© 2022 Texas Instruments

Sommaire

Unité 7 : micro:bit avec Python 1

Compétence 1 : L'affichage 1
Compétence 2 : Boutons et Accélérations 11
Compétence 3 : Autour de la lumière 20
Application : Lancer de dés 25

Unité 5 : Module Turtle et Python 34

Compétence 1 : Installation et découverte du module Turtle 34
Compétence 2 : Couleur et déplacement 41
Compétence 3 : Résoudre un problème 47
Application : Assemblons nos connaissances 50

Unité 7 : micro:bit avec Python

Compétence 1 : L'affichage

Dans cette leçon, vous allez écrire votre premier programme Python pour contrôler l'affichage de la carte micro:bit de différentes manières. Cette leçon est composée de deux parties :
Partie 1 : rencontre extraterrestre.
Partie 2 : affichage d'images.

Objectifs :

- Contrôler l'affichage sur la carte.micro:bit à l'aide .show( ), .scroll( ) et .show(image).
- Contrôler la vitesse de l'écran à l'aide de sleep(ms).

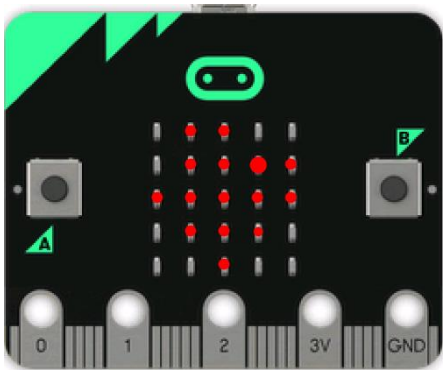
1. Avant de commencer, assurez-vous que :
   - votre TI-83 Premium CE dispose de l'OS **5.6.1** ou supérieur
   - vous êtes à l'aise avec la programmation en Python et/ou vous avez terminé les unités 1 à 5.
   - votre **micro:bit** est connectée à votre TI-83 Premium CE
   - vous avez suivi les instructions de configuration et les transferts de fichiers à l'aide du guide de mise en route micro:bit :
     https://education.ti.com/fr/enseignants/microbit

   Rappel : les Modules sont à télécharger à l'adresse :

   https://education.ti.com/fr/enseignants/microbit/microbit-83ce

   Ce processus de configuration ne devrait être effectué qu'une seule fois, mais tenez-vous informé périodiquement des mises à jour.

2. Si tout est correct et que la configuration a été réalisée avec succès, votre micro:bit devrait afficher l'image ci-contre lorsqu'elle est connectée à la TI-83 Premium CE.
   L'affichage sur la carte **micro:bit** montre le logo TI, une icône de l'État du Texas avec un point lumineux près de Dallas, le siège de **Texas Instruments, Inc**.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

3. Et enfin...

   Les modules micro:bit sont installés dans votre calculatrice. Dans un éditeur Python, appuyer sur **2nde catalog** puis sur la lettre **F**. Utiliser ensuite la touche de direction vers le bas afin de trouver l'instruction « from **SCRIPT** import * ».

   Compléter l'instruction en écrivant « **from microbit import *** ».

Appuyer sur la touche **F1** (Fns) puis ← afin de constater la présence du module **micro:bit** au côté des autres modules.

Les différentes librairies (modules : capteurs, boutons, affichage, radio…) du micro:bit seront accessibles par l'intermédiaire du menu **8 : micro:bit**. Une fois chargées, les instructions relatives à chaque module seront disponibles dans leur menu dédié.

Conseil de l'enseignant : Mise en route **avec micro:bit**

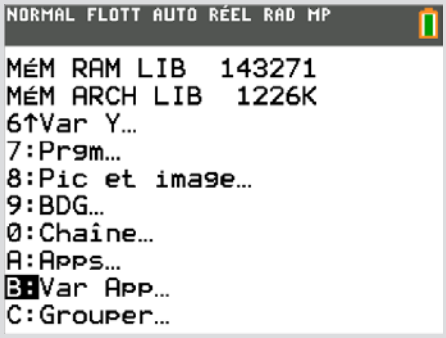
Cette unité suppose que le micro:bit et la calculatrice TI-83 Premium CE EDITION PYTHON soient « prêts à l'emploi ». Assurez-vous que vous (ou vos élèves) avez terminé les étapes de configuration sur leurs calculatrices individuelles et micro:bits. Assurez-vous de lire et d'exécuter les instructions de configuration à l'aide du guide « Mise en route » qui est inclus dans le dossier de téléchargement à partir du site Web TI Education. https://education.ti.com/fr/enseignants/microbit.

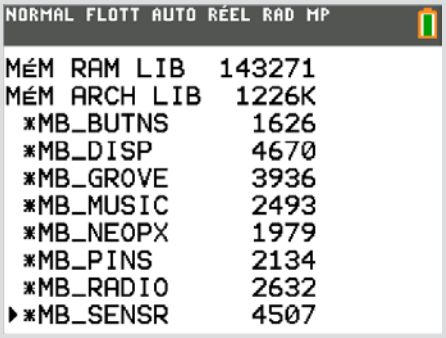
Comme pour tous nouveaux logiciels et matériels, veuillez rester informé des mises à jour.

Il y a aussi un document d'introduction TI-83 Premium CE, **PROG1.8xv**, qui s'assure que la carte micro:bit est connectée et fonctionne correctement.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Le module **BBC micro:bit**, et les modules associés, sont des modules spéciaux fournis par Texas Instruments qui sont installés dans chaque calculatrice comme des variables d'application. Pour constater leur présence dans la calculatrice de vos élèves, Appuyez sur **2nde mém** puis choisir **2 : Gest. Mémoire/Suppr…** et enfin **B : Var App** (ou utilisez les touches de direction vers le bas afin d'atteindre cette instruction).





Les cinq premières unités du programme Python : 10 Minutes de Code doivent être réalisées avant cette unité. Le codage présenté ici n'est pas compliqué, mais vous devez posséder une bonne maîtrise du codage Python sur la calculatrice TI-83 Premium CE EDITION PYTHON.

Cette unité est conçue pour fonctionner avec les versions micro:bit 1 et 2, mais ne traite pas des fonctionnalités uniques à la version 2. Le module BBC micro:bit prend en charge ces fonctionnalités.

Tous les projets micro:bit trouvés en ligne (y compris les projets micro:bit version 2) peuvent être développés sur la TI-83 Premium CE en utilisant Python et un micro:bit attaché. Mais il y a deux distinctions importantes :

- Vous ne pouvez pas déconnecter le micro:bit de la calculatrice pendant que le programme est en cours d'exécution, même si une batterie est connectée au micro:bit. Le runtime micro:bit (fichier .hex) est configuré pour communiquer avec la calculatrice pour ses instructions. La calculatrice alimente et contrôle le micro:bit.
- La plupart des programmes de démonstration trouvés en ligne utilisent l'instruction **While True**: pour créer une boucle infinie. Cette boucle s'exécute directement sur la carte micro:bit jusqu'à ce que le programme soit remplacé par un autre à l'aide d'un fichier .hex téléchargé. Lors de l'utilisation de la TI-83 Premium CE, la calculatrice contrôle le micro:bit de sorte que la boucle la plus couramment utilisée est **« While not escape( ) »:** cela permet à l'utilisateur d'appuyer sur la touche **[annul]** pour mettre fin au programme.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

4. **Partie 1 - rencontre extraterrestre :** Bien sûr, comme avec toute autre première expérience de programmation, vous commencerez par afficher un message sur l'écran de la carte micro:bit.
   Démarrer un nouveau script Python ; le nommer « **SB1** ». Dans l'éditeur Python appuyer 2nde catalog puis **F** pour sélectionner l'instruction « from **SCRIPT** import * ». Valider en appuyant sur la touche **[entrer]** et compléter manuellement l'instruction :

   from microbit import *

   Conseil : Si le message « micro:bit not connected » apparaît, débranchez simplement le micro:bit et rebranchez-le (reset).

Conseil de l'enseignant : Si un micro:bit n'est pas connecté, l'instruction **import** elle-même ne signalera pas une erreur sur la TI-83 lors de l'exécution du programme. Étant donné que Python est un langage « Modulaire », des fonctionnalités supplémentaires doivent être ajoutées si nécessaire. Cette instruction d'importation charge des fonctions/méthodes spécifiques requises pour faire fonctionner le micro:bit à l'aide de la TI-83 Premium CE. Mais attention, certaines méthodes utiles à partir d'autres modules standards doivent être chargés manuellement. Par exemple, pour faire fonctionner des capteurs ou avoir accès aux fonction de l'affichage des DEL de la matrice de la carte BBC micro:bit. Pour charger un module spécifique appuyer sur :
Fns puis ← **Modul** et enfin **8 : micro:bit..**

5. Pour afficher un texte message sur l'affichage micro:bit, utilisez l'instruction « **display.show(**image ou texte**)** », après avoir chargé le module **affichage**, que l'on trouve dans le module général **micro:bit.**
   Celle-ci se trouve sur :
   [Fns] < Modul 9 : affichage > 2 : .show(val)

   L'instruction est insérée en tant que **display.show(**image ou texte**)**, mais (image ou texte) est juste un espace réservé qui doit être remplacé par quelque chose. À l'intérieur des parenthèses, remplacez l'image ou le texte en tapant la chaîne de votre message entre guillemets :

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

« salutations terriens »

Lorsque vous exécutez ce programme en appuyant sur **F4 [Exec],** vous verrez les lettres de votre message apparaître, une lettre à la fois, sur l'écran. Si votre message est composée d'une répétition successive de la même lettre, vous ne pouvez pas en distinguer deux.

Si vous faites une erreur... Editez à nouveau le script pour le modifier après avoir interrompu son fonctionnement **[annul]**, *puis exécutez à nouveau le programme. Avez-vous oublié d'ajouter des guillemets autour du texte que vous vouliez afficher ?*

Remarque : l'instruction « display.show(**val**, del=400, wait=**True**) » comporte des paramètres optionnels (delay=400 en ms, et wait) dont l'utilisation n'est pas utile dans cette leçon. Ils peuvent donc être effacés en utilisant la touche **[suppr]**.

6. Une autre méthode pour afficher les messages est :

   display.scroll(« salutations terriens »)

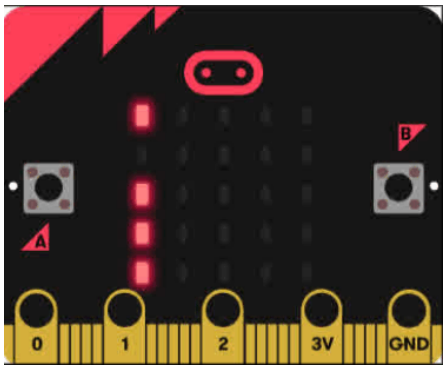
   qui se trouve également sur **[Fns] < Modul 9 : affichage > 2 : .show(val)**.
   Pour commenter la ligne précédente, placer le caractère # devant la ligne à commenter.
   Faites de l'instruction **.show( )** précédente un #commentaire (placez le curseur devant la ligne à commenter et appuyer sur **2nde 3**, puis exécutez à nouveau le programme).

   Vous pouvez également simplement remplacer **.show** *par* **.scroll** *en tapant l'instruction.*

7. La méthode **.scroll( )** fait défiler le message de droite à gauche comme une bannière à travers l'affichage, ce qui le rend plus facile à lire.
   Vous pouvez contrôler la vitesse du défilement en ajoutant le paramètre **delay =** :

   display.scroll(« salutations terriens », delay = 200)

   Cette instruction utilise un délai de 200 millisecondes (0,2 seconde) dans le défilement. Essayez également d'autres valeurs de délai.



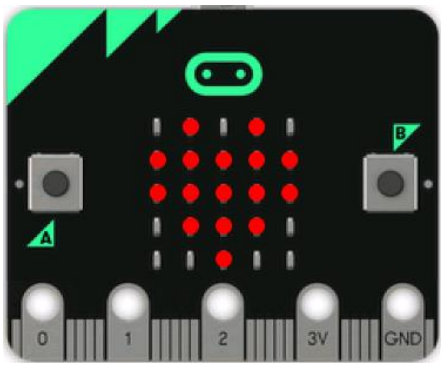
« salutations.gif »

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

8. **Partie 2 - Battements du coeur**
   Appuyer sur **[F5] [script]** puis **[F3] [Nouv]** pour créer un nouveau script et le nommer « **COEUR** ».
   Dans l'éditeur Python, utiliser **2nde catalog** puis **from SCRIPT import** et compléter l'instruction :
   from microbit import *

9. Charger le Module d'affichage **[Fns] < Modul 8 : micro:bit > 1 : affichage**.
10. Pour afficher le symbole HEART sur l'affichage micro:bit, utilisez l'instruction :
    display.show( )
    Cette instruction se trouve sur :
    [Fns] < Modul 9 : affichage > 2 : .show(val)
    Entre parenthèses, remplacez l'invite en sélectionnant :
    Image.HEART
    depuis
    [Fns] < Modul 9 : affichage > Image > 2 : HEART

11. Exécuter le programme (appuyez sur **[F4] [Exec]**) pour voir l'icône HEART affichée sur la grille LED 5x5 de la carte micro:bit. Cet affichage restera jusqu'à ce qu'un autre prenne sa place, même après la fin de l'exécution du programme.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

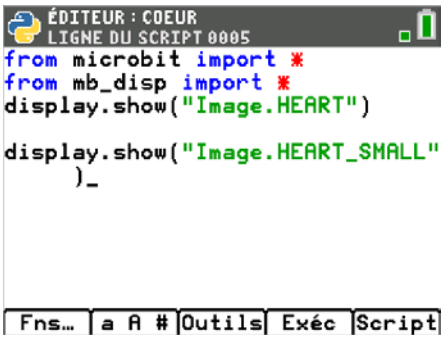
12. Revenir à l'éditeur de programmes sur la page précédente, et ajouter une autre instruction d'affichage pour montrer le petit cœur :

    display.show(Image.HEART_SMALL)

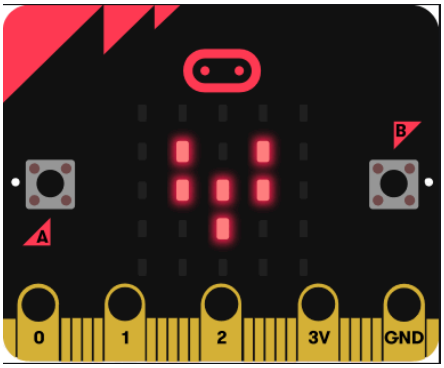
    Vous pouvez trouver cette image dans le même menu :

    [Fns] < Modul 9 : affichage > Image > 3 : HEART SMALL

    Astuce : vous pouvez également copier / coller la première instruction d'affichage et la modifier (insérez le **_SMALL**).



13. Exécuter à nouveau le programme. Il affiche rapidement le grand cœur, puis affiche le petit cœur qui ressemble à l'image ci-contre.



14. **Faire une boucle** : Pour que les deux cœurs clignotent à plusieurs reprises (« battements »), placez les deux instructions d'affichage dans une boucle. *Avant* les deux instructions d'affichage, insérez :

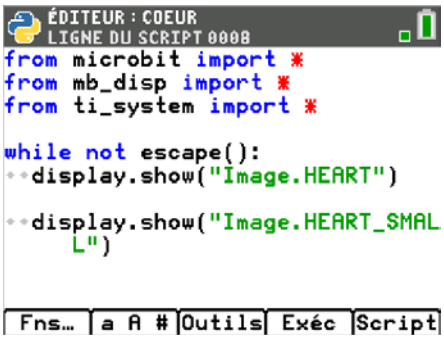
    while not escape( ):

    trouvé dans

    [Fns] < Modul > 4 : ti_system > 5 : while not escape( ) :

    N'oubliez pas de charger préalablement le module **ti_system** et mettez en retrait (indentation) les deux instructions d'affichage, afin qu'elles forment le corps de la boucle.

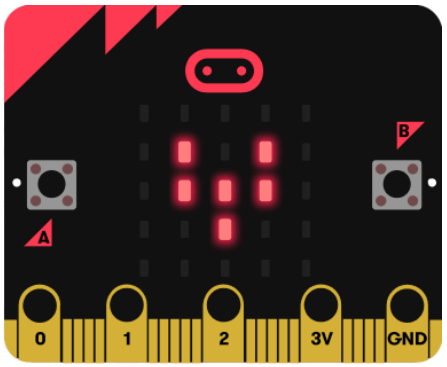
    *Conseil important : **L'indentation** est essentielle dans les programmes Python. C'est ainsi que Python interprète les blocs de boucle et les blocs* **if**. *Si les deux instructions d'affichage ne sont pas mises en retrait le même nombre d'espaces, vous verrez une erreur de syntaxe. Utilisez la touche* **[F3] [Outils]** *puis* **1 : Indent >** *pour mettre en retrait les deux lignes de la même quantité. Les espaces d'indentation sont indiqués dans l'éditeur par un symboles de diamant gris clair (◆◆) afin de faciliter une indentation correcte.*



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

15. Exécuter à nouveau le programme et regarder le cœur battre ! Appuyer sur la touche **[esc]** pour terminer le programme.

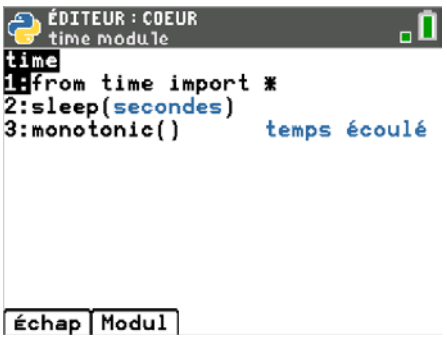
    Astuce : si jamais vous *pensez que votre programme est bloqué dans une boucle infinie, appuyez et maintenez la touche* **[on]** *sur votre* TI-83 Premium CE *pour « casser » le programme. Cela peut se produire si vous l'utilisez alors que l'instruction* **True:** *est incorrectement placée dans le script. Ces leçons évitent ce type de structure.*



<*cœur* .gif>

16. Le menu **time** micro:bit contient des instructions Python utiles que l'on trouve également dans d'autres menus. Le module micro:bit importe ces commandes Python pour vous.

    Vous pouvez trouver ces commandes et toutes les autres commandes Python à partir d'autres menus. Vous n'êtes pas limité à utiliser simplement le menu BBC micro:bit, mais vous devrez peut-être fournir les commandes d'importation appropriées.



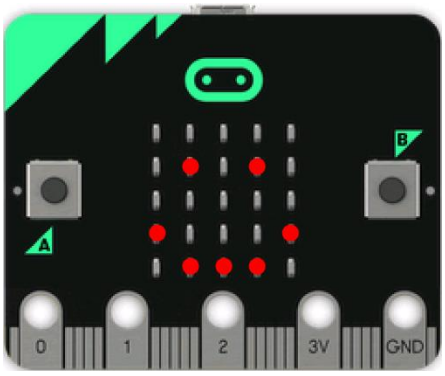
17. Pour contrôler la fréquence des battements cardiaques, ajoutez deux instructions **sleep( ),** une après chaque instruction **d'affichage** : **sleep(1000)** signifie 1000 milli secondes ou un délai de 1 seconde. Aussi trouvé sur

    [Fns] < Modul > 3 : time.. > 2 : sleep( )

    Astuce : observez l'indentation !

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

18. **Extension :** Essayer de faire une « série de visages » ou smiley. Utiliser une structure de programme similaire à celle du script « **Battements du coeur »**, mais sélectionner plutôt les images de visages trouvées dans l'ensemble des images.



Conseil de l'enseignant : Les leçons micro:bit en ligne utilisent généralement une boucle **While True: en Python et** une boucle **forever :** dans MakeCode**.** Ces « boucles infinies » s'exécutent directement sur le micro:bit jusqu'à ce qu'il soit désactivé ou remplacé par un autre programme (fichier .hex). Lorsque vous utilisez la TI-83 Premium CE pour contrôler le micro:bit, la boucle infinie n'est pas nécessaire puisque la calculatrice est en plein contrôle. **While True: se trouve dans le menu Fns puis Ctl, mais doit être utilisé conjointement avec if (Expression booléenne) : break** pour sortir de la boucle **infinie.**

Si un élève est bloqué dans une boucle infinie sur la calculatrice, pour « casser » le programme, appuyez et maintenez la touche [on]. Consultez le guide « Mise en route » (PDF) fourni avec le logiciel micro:bit de TI, pour davantage d'informations sur la rupture des programmes Python sur les ordinateurs et calculatrices.
Les menus contiennent un ensemble de commandes Python courantes et fréquemment utilisées qui sont utiles lors du codage pour micro:bit. Les modules micro:bit importent ces commandes et doivent être chargés au début du script après le module micro :bit. Mais toutes les commandes Python sont disponibles dans d'autres menus et peuvent être utilisées dans des projets micro:bit. L'utilisation des instructions d'importation appropriées est importante.

Facultatif : Pour récupérer l'affichage de son état « TI » (le logo TI du Texas) à la fin d'un programme, ajoutez l'instruction

display.show(ti)

après la fin de la boucle While. Cette instruction doit être désindentée.

Ce n'est pas nécessaire, mais cela permet d'indiquer que le programme est terminé sur l'écran micro:bit ainsi que sur l'écran.

Au début d'un programme micro:bit (avant le début de la boucle while), *il peut être utile d'imprimer quelque chose sur l'écran de la calculatrice, tel que* **print(« running... »)**.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

À propos de sleep( ): La seule fonction Python qui est considérablement modifiée pour le micro:bit est l'instruction **sleep( )** trouvée dans le module **time.** Normalement, la valeur de l'argument représente des « secondes », mais le module micro:bit modifie l'instruction **sleep** afin que l'argument représente un temps en millisecondes *(millièmes* de seconde). Après l'importation du module micro:bit, **sleep(1000)** représente un délai d'une seconde. Notez que **sleep( )** contrôle la vitesse du programme de la calculatrice, pas le micro:bit lui-même. Assurez-vous que l'importation de micro:bit se produit après le moment de l'importation (ou tout module qui pourrait importer du temps).

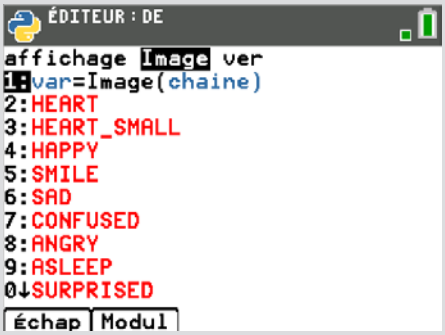
La méthode .show() a deux paramètres facultatifs : delay= et wait=

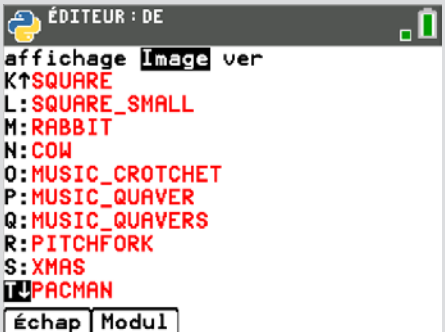
display.show(valeur, delay =*xxx* , wait = False/True)

- **la valeur du delay** est en millisecondes et ralentit l'affichage ;
- **wait = True** bloquera le traitement jusqu'à ce que l'animation soit terminée, sinon l'animation se produira en arrière-plan.

Voici la liste complète des images :

Un projet similaire est « Série de visages » en utilisant la même structure de programme que « Battements du cœur » mais utilisez les icônes de visage à la place des cœurs.





Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 7 : micro:bit avec Python

Compétence 2 : Boutons et Accélérations

Dans cette leçon, vous allez apprendre à utiliser les boutons et l'accéléromètre de la carte BBC micro:bit, puis écrire un programme pour créer un dé et collecter les valeurs dans une liste à transférer, afin de créer un diagramme de données statistiques.
Cette leçon comporte deux parties :
Partie 1 : Etude des instructions sur les boutons et utilisation de l'accéléromètre.
Partie 2 : Utilisation d'un bouton pour générer des données.

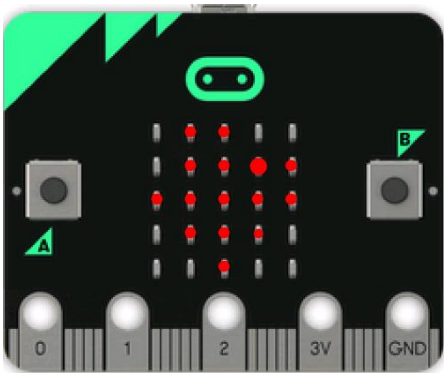
Objectifs :

- Lire et traiter les boutons A et B sur la carte micro:bit.
- Observer la différence entre .was et .is.
- Transférer des données du script Python vers la calculatrice.
- Étudier les données collectées à partir du micro:bit.
- Utiliser l'accéléromètre pour contrôler l'affichage.

Conseil de l'enseignant : Comme dans la Compétence 1, cette leçon est conçue de l'intérieur vers l'extérieur. Le code n'est pas introduit séquentiellement, mais développé pour se concentrer d'abord sur les fonctionnalités du bouton ou du déplacement spatial de la carte micro:bit, puis terminer par établir une connexion entre les listes Python et les listes TI-83 Premium CE.

1. La carte micro:bit possède deux boutons, étiquetés A et B, de chaque côté de la carte. Le module Python micro:bit a deux méthodes similaires pour utiliser les boutons. Vous allez d'abord tester les instructions, puis écrire un programme qui vous permet de collecter des données et de les analyser ailleurs dans la calculatrice TI-83 Premium CE.

   Il y a aussi une puce accéléromètre/boussole à 3 axes à l'arrière du micro:bit. Une convention (orientations des axes) est nécessaire afin d'interpréter le mouvement et l'orientation de la carte micro:bit.



Conseil de l'enseignant : Sur la version micro:bit 2, il y a aussi un bouton « tactile » au-dessus de l'écran (l'ovale doré). Dans le module, ce bouton est appelé « Logo Touch » et utilise l'instruction **'is_touched( )'**. Ce bouton tactile n'est pas abordé dans ces leçons, mais il est facile à intégrer dans votre projet en fonction de ce que vous apprenez dans cette leçon. Notez la différence entre le comportement de **.is_** et **.was** ci-dessous...

2. **Partie 1 : Utilisation des boutons.**
   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 **BOUTONS.**
   Dans **2nde catalog**, sélectionner l'instruction d'importation **from SCRIPT import** et complétez manuellement l'instruction.
   from microbit import *

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

3. Importer également la bibliothèque **TI_System**.
   Ajouter la boucle **while** :

   while not escape( ):

   depuis

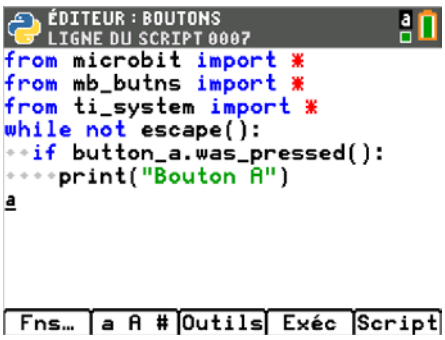
   [Fns] > modul > 4 : ti_system > 5 : while not escape( )

   La plupart de vos programmes micro:bit *seront conçus de cette façon.*

4. Pour tester le bouton A, charger la librairie relative aux boutons puis ajouter la condition :

   ♦♦**if button_a.was_pressed( ):**
   ♦♦♦♦**print("Bouton A")**

   if *est mis en retrait pour faire partie de la* boucle **while** *et* **print( )** est mis en retrait encore plus pour faire partie *du* bloc **if.** *N'oubliez pas qu'une indentation correcte est très importante en Python. Une mise en retrait incorrecte peut provoquer des erreurs de syntaxe ou une exécution incorrecte de votre code. Notez les symboles losange gris clair (♦♦)* qui *indiquent l'espacement de mise en retrait (indentation).*



   if se trouve dans **[Fns] > Ctl** .
   La *condition* **button_a.was_pressed( )** se trouve à

   [Fns] < modul> 9 : boutons > 2 : .was_pressed( )

   print( ) se trouve dans **[Fns] < E/S**.

   Taper le texte « Bouton A » sur le côté de la fonction **print( )**.

   Note : **is_pressed( )** *sera également utilisé plus tard.*

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

5. Vous êtes prêt à tester le programme. Appuyer sur **[F4] Exec** pour exécuter le programme. On dirait qu'il ne se passe rien. Appuyez et relâchez le bouton A sur la carte micro:bit. Vous verrez « Bouton A » apparaître sur l'écran de la calculatrice. Chaque fois que vous appuyez sur le bouton, le texte apparaîtra comme dans cette image.

   Appuyez sur **[annul]** et revenez à l'éditeur Python.

6. Ajoutez une autre instruction **if** pour vérifier le fonctionnement du bouton B à l'aide de la condition **button_b.is pressed( )**. Notez que « IS » est différent de « WAS ».
   Vous verrez bientôt en quoi ils diffèrent.

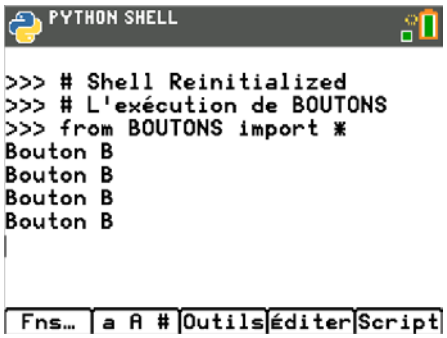
   ♦♦**if button_b.is_pressed() :**
   ♦♦♦♦**print("Bouton B")**

   Astuce : encore une fois, faites attention aux indentations !

Conseil de l'enseignant : Les deux fonctions se comportent différemment. Il existe des circonstances de programmation où le choix entre les deux comportements sera important.

7. Exécutez à nouveau le programme (appuyez sur **[F4] Exec**). Essayez les deux boutons A et B.
   Appuyez sur chaque bouton *et* **appuyez sans relâcher** chaque bouton. Vous verrez « Bouton B » affiché à plusieurs reprises tant que le bouton B est maintenu enfoncé, mais pas « Bouton A ». Il y a une différence entre **.was_pressed( )** (qui nécessite une libération du bouton pour être réinitialisé) et **.is_pressed( )** qui vérifie simplement si le bouton est enfoncé au moment même où l'instruction est traitée.

   Remarque : si vous appuyez sur le bouton B rapidement, le programme peut ne pas afficher le bouton B car le bouton n'est pas enfoncé au moment même où l'instruction **if** *est en cours de traitement.*



Conseil à l'enseignant :
.was_pressed( ) nécessite une action complète de « appuyer-relâcher » pour détecter les pressions individuelles (clics).

.was_pressed( ) détecte un événement appuyer-relâcher qui peut se produire même lorsque le bouton a été cliqué à un autre moment dans l'exécution de la boucle. Le bouton doit être relâché pour qu'un autre clic se produise. Le micro:bit « se souvient » que le bouton a été enfoncé.

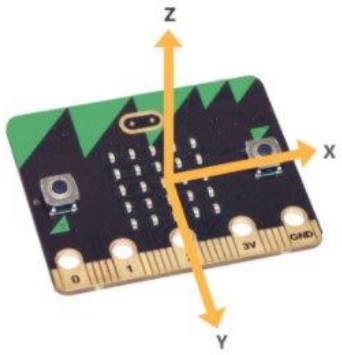
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

.is_pressed( ) est comme un événement « est enfoncé ? ». Certains langages de programmation pilotés par les événements ont une fonction similaire telle que « mouse_down ». **.is_pressed( )** ne produit **True** que si le bouton **est enfoncé** au moment exact **où l'instruction est traitée.**

Dans le reste de cette leçon, le code du bouton B est ignoré.

8. **Utilisation de l'accéléromètre.** Le micro:bit dispose d'un accéléromètre électronique qui peut mesurer l'intensité des accélérations dans trois directions différentes (un accéléromètre 3 axes ou 3D).
Cette leçon explore les possibilités de l'accéléromètre et montre comment l'utiliser dans la programmation de votre TI-83 Premium CE avec la carte micro:bit.

La carte



Charger la librairie relative aux capteurs. Cette librairie se trouve à :

[Fns] < modul > 8 : micro :bit > 3 : capteurs

9. Obtener les valeurs de l'accélération du mouvement à partir du micro:bit et les stocker dans les variables **ax, ay, az:**

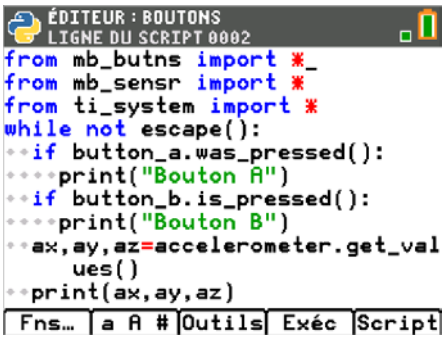
♦♦**ax,ay,az = accelerometer.get_values( )**

Astuce : encore une fois, faites attention aux indentations !

Taper ♦♦**ax,ay,az** puis

Trouver **accelerometer.get_values( )** à

[Fns] < modul > 0 : capteurs > 4 : var,var,var=.get values( )



Puis afficher les valeurs des accélérations selon les axes *Ox, Oy et Oz* :

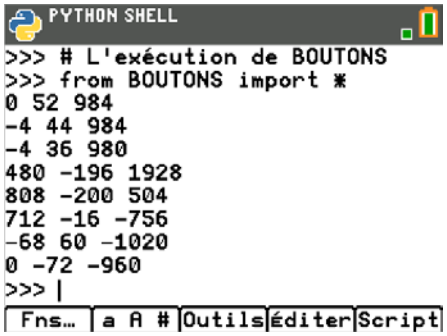
♦♦**print(ax,ay,az)**

Rappel : **print( )** est dans **[Fns] < E/S**

Notez que ces deux instructions sont indentées pour faire partie de la boucle **while**, *mais pas* de *l'instruction* **if** *au-dessus. N'oubliez pas que l'indentation de chaque ligne détermine la signification de ce que réalise le script. Alors SOYEZ PRUDENT !*

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

10. Exécuter le programme et regarder l'affichage de la calculatrice pour les différentes valeurs lorsque que vous déplacez la carte micro:bit dans les airs. Tournez la carte micro:bit, tenez-la sur chaque bord, secouez-la. Une partie de l'exécution du programme est affichée dans cette image.



L'accéléromètre du BBC micro :bit mesure le mouvement selon trois axes :

- X - l'inclinaison de gauche à droite.
- Y - l'inclinaison d'avant en arrière.
- Z - le mouvement haut et bas.

Pour chaque axe, la convention de repérage permet de comprendre le signe obtenu lors d'une mesure. Un nombre positif ou négatif qui indique une mesure en milli-g. Lorsque la lecture est de 0, on est « aligné » selon cet axe.

Nous allons réaliser un « niveau à bulle » très simple qui utilise l'instruction **accelerometer.get_x( )** pour mesurer l'alignement de la carte selon l'axe X. L'algorithme est le suivant

```
Lire l'accélération selon l'axe X
    Si lecture >20 :
        Afficher "Droite"
    Ou Si lecture < -20 :
        Afficher "Gauche"
    Sinon :
        Afficher "—"
```

11. Créer un nouveau script appelé « niveau » et importer les modules nécessaires (**micro:bit**, **ti_system, mb_sensor**, **mb_disp**).

Astuce : Dupliquer le script précèdent à l'aide de

[Script] ↓ selection du script [Gérer] 1 : Dupliquer

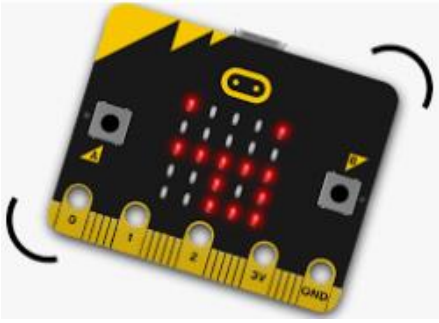
Effacer ensuite les lignes que vous ne souhaitez pas conserver, ou les commenter.

Mettre la carte BBC micro:bit sur une surface plane et exécuter le script. Vérifier ainsi la planéité d'une surface.

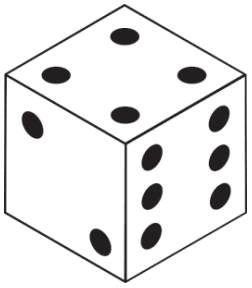
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Si l'appareil est tenu horizontalement il devrait afficher -- en revanche, si on l'incline vers la gauche ou vers la droite il devrait afficher G ou D respectivement.

12. Les boutons et les déplacements spatiaux sont deux façons d'obtenir une entrée sur la carte micro:bit et de produire des résultats sur l'écran de la calculatrice TI-83 Premium CE, l'écran micro:bit... ou les deux.
    La partie suivante de cette leçon crée une activité qui produit des données à l'aide de la carte micro:bit pour un apprentissage plus approfondi avec la TI-83 Premium CE.



13. **Partie 2 :** Jetons un dé (un cube numéroté de 1 à 6 sur ses faces). Lorsque le **bouton A est enfoncé**, un entier aléatoire de 1 à 6 est assigné à une variable. Vous pouvez utiliser le bouton A, le bouton B ou un ~~geste~~ déplacement de votre choix.
    Afficher la valeur sur la carte micro:bit uniquement. Essayez-vous même avant de regarder l'étape suivante. Nous utiliserons le programme actuel et ajouterons du code pour simuler le lancer de dé.



    Pouvez-vous déterminer quel numéro se trouve au bas du dé illustré ci-contre ?

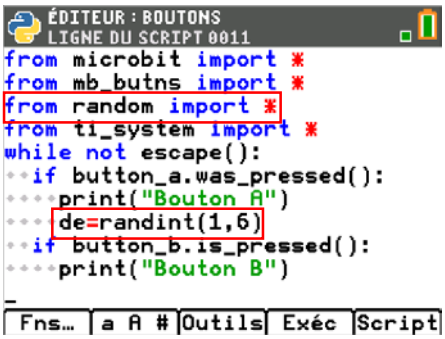
Conseil à l'enseignant :

Réponse : la valeur sur la face inférieure est 3. Les faces opposées d'un dé s'additionnent toujours à 7.

14. Reprendre le script « boutons ». Ajoutez les deux instruction mises en valeur selon les indications de la capture ci-contre :
    From random import * et **randint( )** se trouvent tous deux sur **[Fns] < Modul > 2 : Random**

    Le reste de l'instruction ♦♦♦♦**de = randint(1, 6)** est tapé manuellement et est soigneusement indenté, car il fait partie de la condition **If button_a.was_pressed( )**.

    Encore une fois, faites attention à l'indentation.

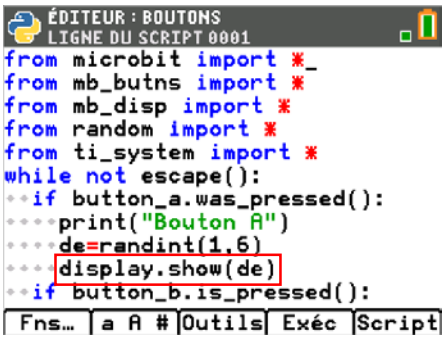


Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

15. Une fois que la valeur de la variable a été établie, ajouter l'instruction :

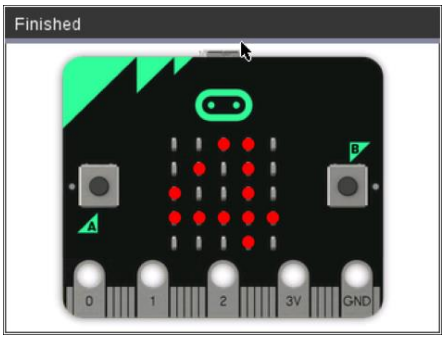
display.show (de)

pour afficher la valeur sur la matrice de l'écran de la carte micro:bit, sans oublier de charger le module d'affichage **mb_disp**.



Exécuter le programme à nouveau. Lorsque vous appuyez sur le bouton A, vous voyez « Bouton A » sur l'écran de la calculatrice et le numéro sur l'affichage de la carte micro:bit change... mais pas à chaque fois ! Parfois, le nombre aléatoire sélectionné est le même que le précédent. Les appuis successifs sur le bouton A sont des « événements indépendants ».

16. **Collecte de données :** Lancer le dé avec juste une pression sur un bouton est intéressant, mais pour une étude plus approfondie, il serait utile de stocker toutes ces valeurs dans des listes afin que vous puissiez interpréter les données : Un nombre a-t-il une occurrence plus fréquente que les autres ? Quel est le nombre moyen ? Et ainsi de suite…



Ajouter les instructions au programme afin de :

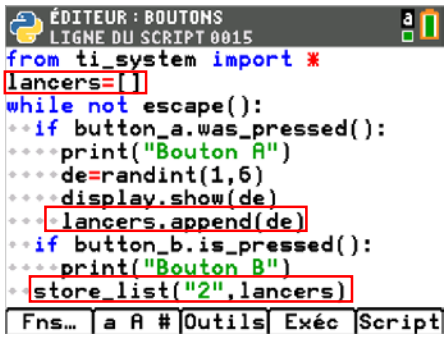
- Créer une liste vide.
- Ajouter (**.append**) la valeur d'un lancer à la liste.
- Stocker la liste de Python vers le système TI-83 Premium CE pour l'analyse.

Chacune de ces trois tâches se traduit par des instructions placées à des endroits particuliers dans le programme. Essayez-le vous-même avant de passer à l'étape suivante.

17. L'affectation de liste vide est à mettre au début du programme :

lancers = [ ]

Les crochets sont sur le clavier et dans la table de caractères **[a A #]**.



Lorsqu'un lancer est effectué, il est ajouté à la liste avec :

lancers.append(de)

La méthode **.append()** se trouve dans **[Fns] > List > 6.append(x)**

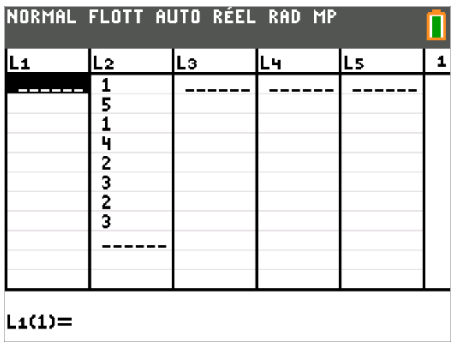
A la fin du programme, la liste finale est stockée dans une liste de la TI-83 Premium CE :

store_list("2", lancers)

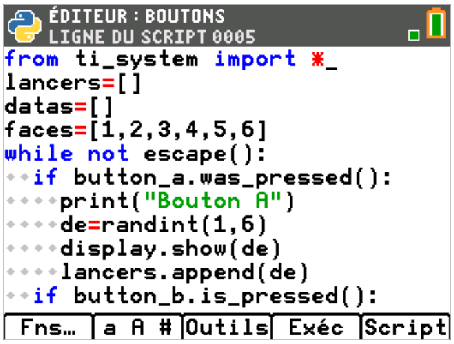
La fonction **store_list** est à **[Fns] < Modul > 4 : ti_system**.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

18. Lorsque vous exécutez le programme, appuyez plusieurs fois sur le bouton A, puis appuyez sur **[annul]** pour mettre fin au programme. Votre programme Python a une liste nommée « lancers » et maintenant les données de cette liste ont été exportées dans la liste L_2 de la calculatrice.



Mais ces données ne sont pas classées. Nous allons le faire en utilisant une boucle fermée (**For**) et l'instruction **.count(n° face)** effectuera le comptage des occurrences correspondant aux numéros des faces (1 à 6). Les données de ce comptage seront sauvegardées dans une liste **data( )** à initialiser aux côtés des autres (lancers( ) et faces=[1,2,3,4,5,6])



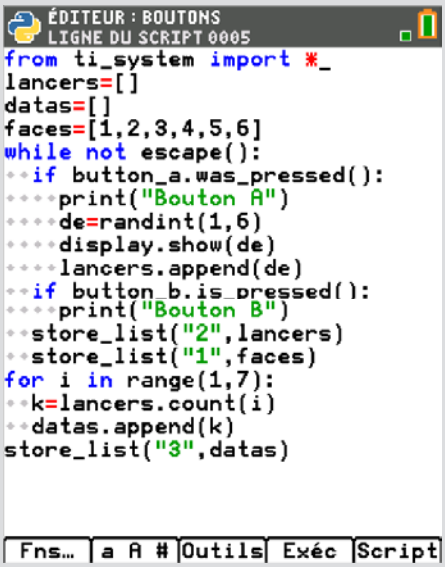
La liste faces sera exportée en L_1 et la liste data en L_3
Exécuter de nouveau votre script afin d'obtenir une cinquantaine de lancers, puis effectuer la représentation graphique des données sous la forme d'un diagramme en bâtons.

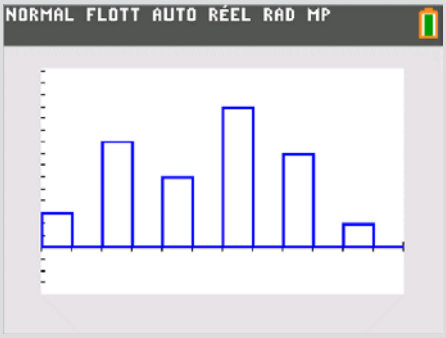
Remarque : le bouton B et votre geste n'ont aucun effet sur la matrice. Essayez de modifier votre code pour afficher un lancer du dé uniquement lorsque *vous « secouez » la carte micro:bit.*

N'oubliez pas d'enregistrer votre document !

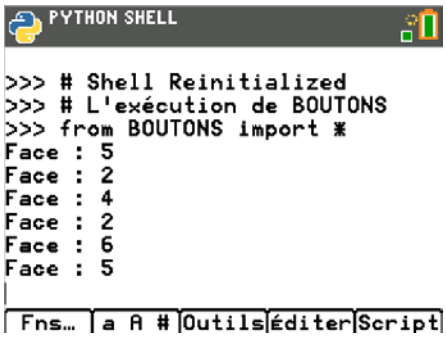
Conseil à l'enseignant :
Le bouton B pourrait être utilisé comme un bouton de « réinitialisation », effaçant les données et recommençant avec une liste vide.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée





19. **Extensions possibles :**
    - Modifier votre script afin que la face de chaque lancer soit affichée.
    - Utiliser l'accéléromètre afin de déclencher la réalisation d'un lancer.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 7 : micro:bit avec Python

Compétence 3 : Autour de la lumière

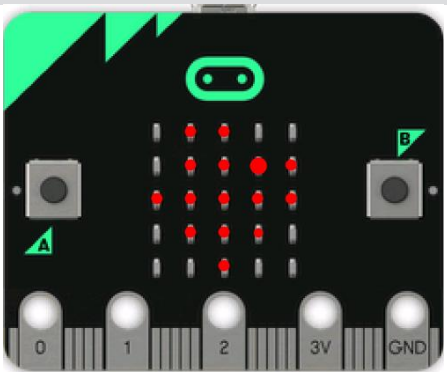
Dans cette leçon, vous allez observer le fonctionnement du capteur de lumière sur la carte micro:bit et stocker les données dans une liste de la calculatrice TI-83 Premium CE pour une analyse plus approfondie.

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.**

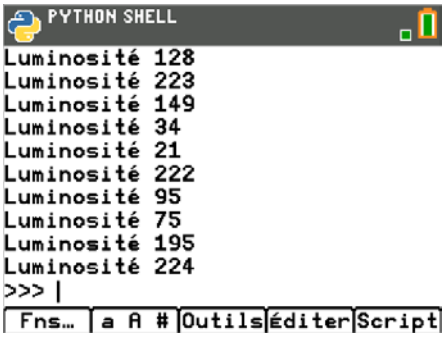
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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 = [ ]

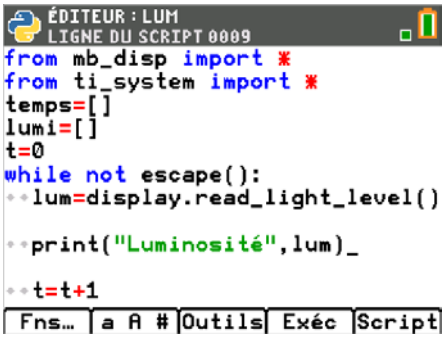
Trouvez les crochets sur le pavé numérique **2nde** **[×][–]** ou sur **[Fns] > List > 1 : [ ]**.
Toujours avant la boucle **while**, une instruction pour initialiser une variable du compteur temps (**t**) à 0 : **t = 0**

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.*



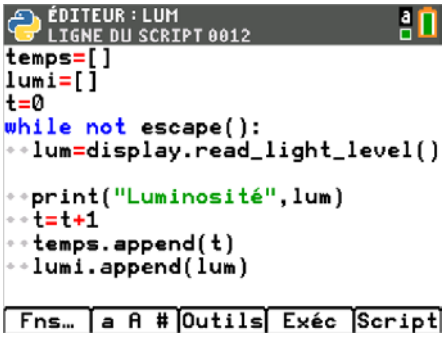
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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 :

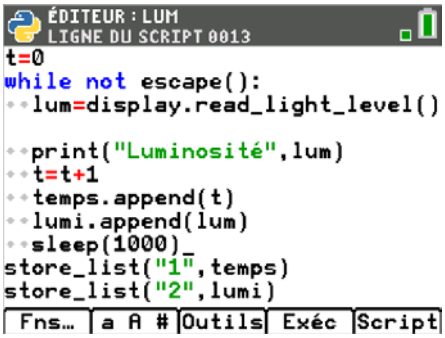
   [Fns] < Modul >4 : ti_system > A : sleep( )

   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.

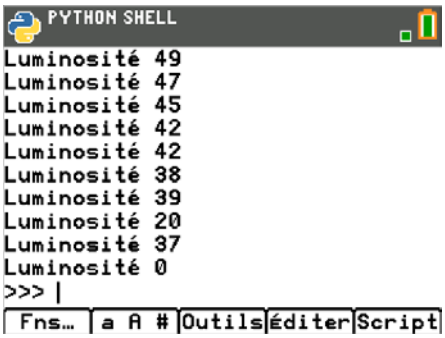
   Dans ce programme, les listes *Python* ne sont stockées dans les *listes TI-83 Premium CE juste qu'à la toute fin du programme,* lorsque vous *appuyez sur* **[annul]** pour quitter la *boucle while.*



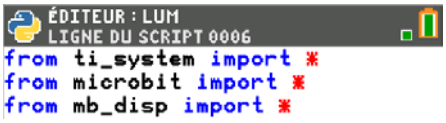
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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.

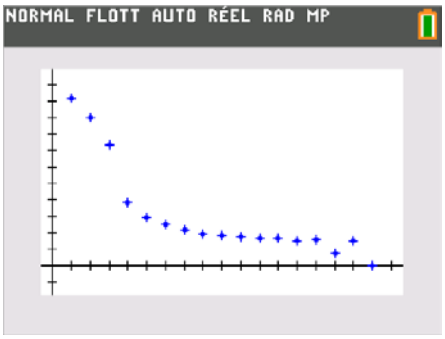
   Répéter le processus jusqu'à ce que vous pensiez avoir obtenu de « bonnes » mesures.



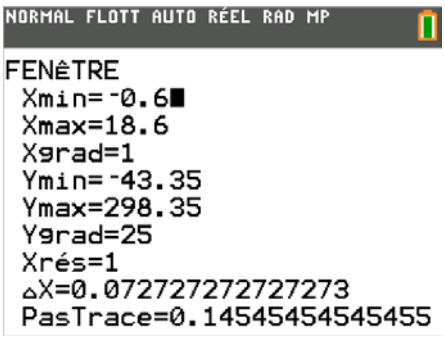
   Remarque : Si le programme ne fonctionne pas ou reste bloqué, modifiez l'ordre de chargement des librairies en plaçant le module ti_system en premier.



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.*

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

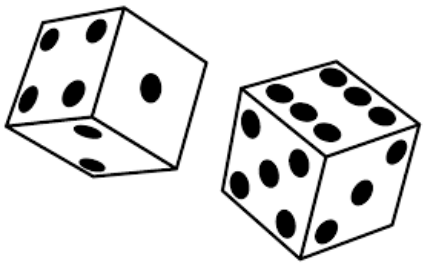
Unité 7 : micro:bit avec Python

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, 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
```

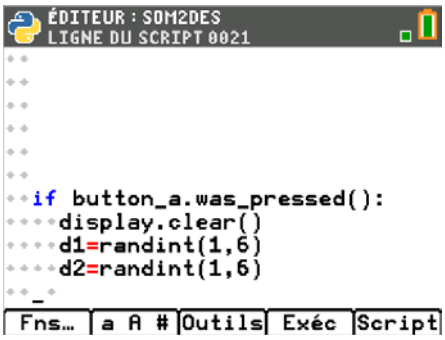
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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 :

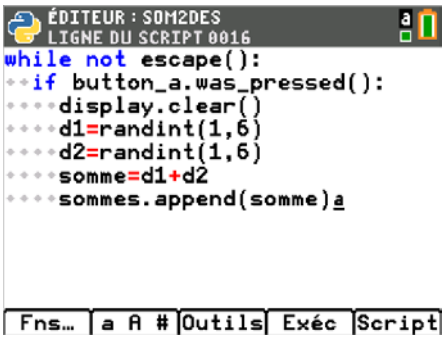
```
    If button_a.was_pressed( ):
        display.clear()
        d1 = randint(1,6)
        d2 = randint(1,6)
```

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)
        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.*

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 7 : micro:bit avec Python

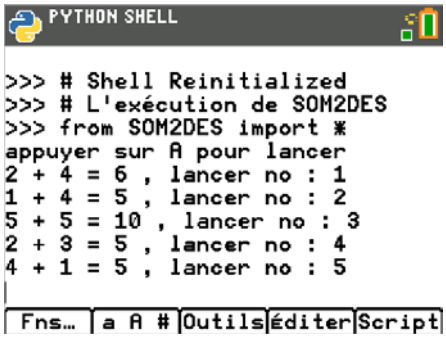
Conseil de l'enseignant : une autre instruction **sleep( )** peut être utile dans la boucle pour donner à la carte micro:bit le temps de surveiller l'appui sur le bouton.

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.

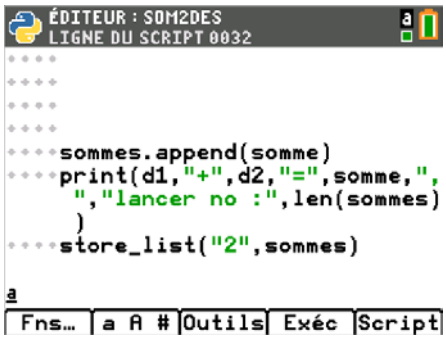
Attention à la ponctuation !



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

♦♦♦♦**store_list("2",sommes)**

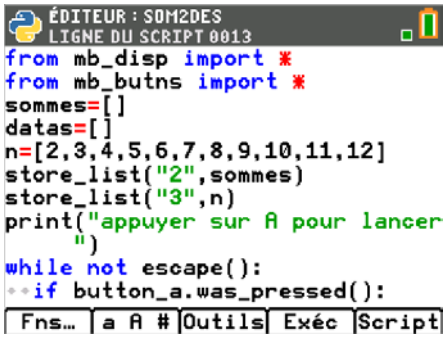
L'instruction **store_list( )** *est incluse à l'intérieur des blocs* **while** *et* **if** *afin que la liste TI-83 Premium CE soit mise à jour chaque fois qu'une nouvelle paire de dés est générée.*



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.

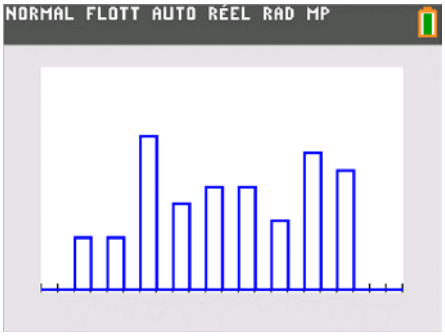
9. Lorsque vous êtes convaincu que votre programme fonctionne correctement, vous êtes prêt à l'utiliser avec les possibilités de représentation graphique de la calculatrice TI-83 Premium CE. Exécutez votre programme et générez environ 50 lancers. Appuyez sur **[annul]** pour terminer le programme.

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.



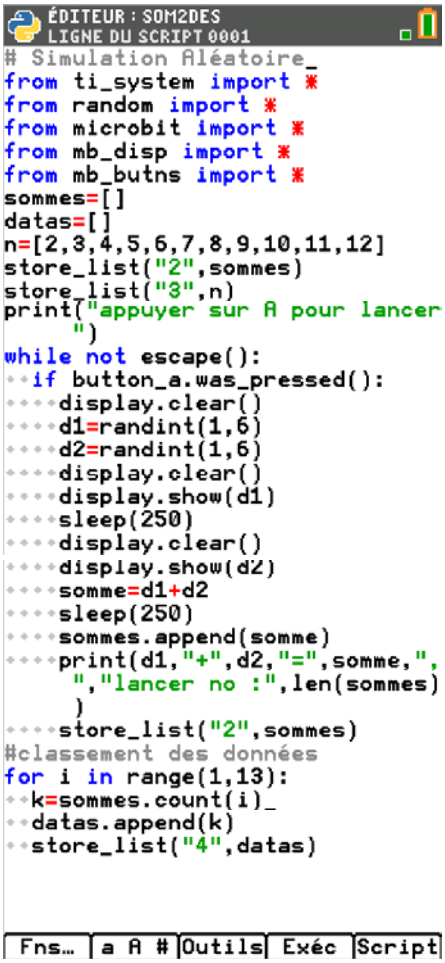
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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.

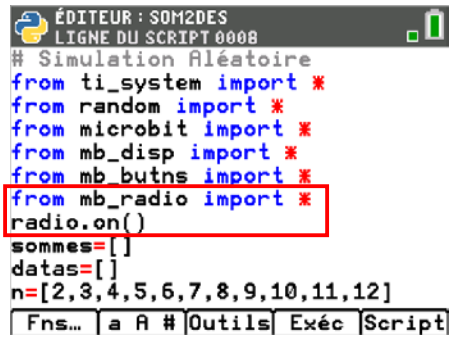
Pour utiliser la radio, vous devez avoir 2 cartes micro:bit et 2 calculatrices. La communication radio est très simple à mettre en œuvre comme vous pourrez le voir.

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.



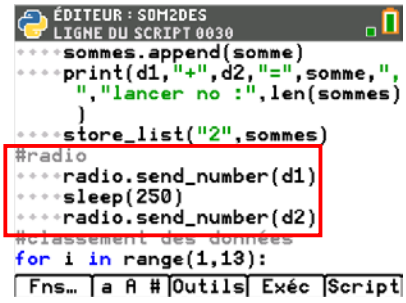
La communication peut alors s'établir entre plusieurs cartes.

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.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

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:"
```

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

```
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
```

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

```
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).
```

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 8 : Module Turtle et Python **Compétence 1 : Installation et découverte du module Turtle**

Dans cette leçon, vous allez installer le module Turtle dans la calculatrice TI-83 Premium CE Edition Python et découvrir les instructions permettant à la « Tortue » d'effectuer des déplacements.

Objectifs :

- Installer le module Turtle dans la calculatrice.
- Réaliser le tracé de quelques figures simples.
- Créer une figure de « Math-Art ».

Remarque : les sections suivantes supposent que vous avez mis à jour votre TI-83 Premium CE Edition Python vers la version 5.7 de l'OS et des Apps ou une version ultérieure. Si vous n'avez pas effectué la mise à jour vers la version 5.7 ou une version ultérieure, veuillez consulter le site : https://education.ti.com/fr/produits-ressources/mise-a-jour-ti-83-premium-ce. Veillez à sauvegarder ou archiver les fichiers et les données de votre TI-83 Premium CE Edition Python avant de procéder à la mise à jour.

Transférer le module Turtle à votre calculatrice.

Transférer le module Turtle et le fichier Grid à votre TI-83 Premium CE Edition Après avoir téléchargé le Zip et extrait les fichiers :

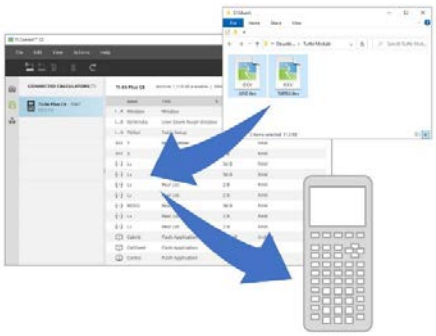
- Ouvrir le logiciel de bureau TI Connect CE.
- Connecter votre TI-83 Premium CE Edition Python à votre ordinateur en utilisant le câble USB ordinateur-calculatrice fourni avec la calculatrice.
- Vérifier que votre calculatrice connectée apparait dans le panneau Calculatrices connectées de l'espace de travail Explorateur de calculatrices.
- Transférer les fichiers TURTLE.8xv et GRID.8xv vers la ou les calculatrices connectées en faisant glisser les fichiers dans la fenêtre des calculatrices connectées.

  Remarque : le fichier GRID.8xv est une image utilisée comme grille de fond dans les programmes Turtle.

  Remarque : les fichiers Turtle et Grid peuvent être aussi transférés d'une calculatrice à une autre à l'aide d'une liaison par câble USB d'unité à unité. Consulter le guide de votre calculatrice pour connaitre le processus de transfert de fichiers de calculatrice à calculatrice.

Cette étape nécessite l'utilisation du logiciel (gratuit) TI Connect CE pour ordinateur PC ou Mac.

https://education.ti.com/fr/produits/logiciel-ordinateur/ti-connect-ce-sw

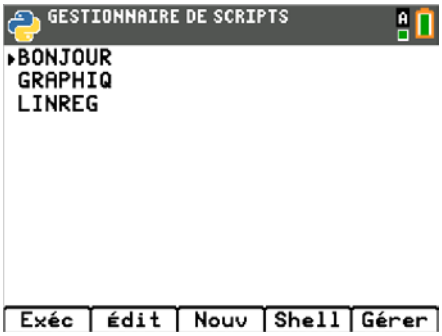


Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Importation du module Turtle.

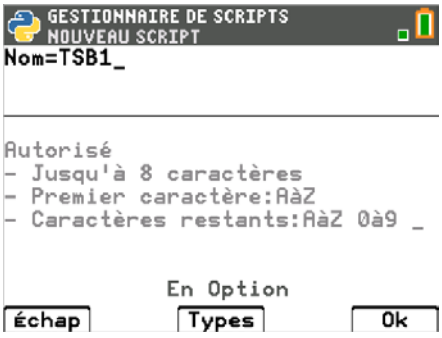
Une fois le module Turtle et le fichier Grid transférés, créer un script Python et importer le module Turtle afin d'accéder aux sélections de son menu.

- Sur votre TI-83 Premium CE Edition Python, ouvrir l'application Python. Vous pouvez la trouver à l'aide de la touche **[prgm]** ou de la touche **[apps]**. L'application Python s'ouvre sur un écran de gestion des scripts.



- Créer un nouveau script Python en sélectionnant l'onglet intitulé **Nouv** (appuyez sur la touche **[zoom]**).

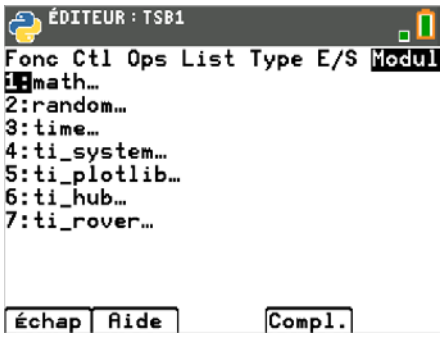
- Donner un nom au script Python (par exemple : **TSB1**), puis sélectionner **Ok**.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Vous êtes maintenant dans l'écran de l'éditeur de scripts Python. Saisir, ensuite, l'instruction d'importation Turtle et un objet Turtle.

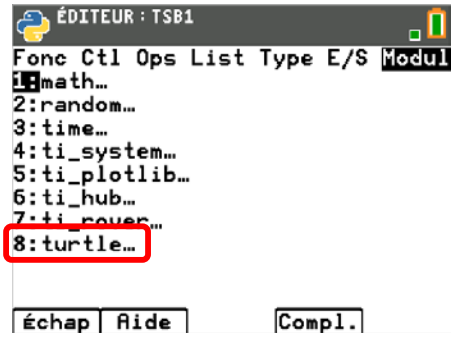
- Sélectionner l'onglet **[Fns...]** (en bas de l'écran), puis flèche gauche vers l'onglet **Modul** (en haut de l'écran).
- En bas de l'écran, sélectionner l'onglet **[Compl.].**
- Sélectionner l'option **3:from turtle import *** pour coller l'instruction d'importation du module Turtle.



Remarquer qu'il colle aussi un constructeur **t=Turtle()** pour assigner à t l'objet Turtle.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Retourner à l'onglet **Modul** et remarquer l'apparition de **8 : turtle...** dans le menu.



Conseil à l'enseignant : Remarque : l'onglet [Compl.] n'est visible qu'avec la version 5.7 ou ultérieure du système d'exploitation. Si vous utilisez une version antérieure, mettez à jour vers la version 5.7 ou ultérieure ; sinon, vous devez saisir manuellement l'instruction d'importation et le constructeur **t=Turtle().**

Maintenant, vous êtes prêt à accéder aux sélections de menu du module Turtle et à écrire un script Turtle !

Conseil à l'enseignant : « Turtle Graphics » a été introduit dans le langage de programmation Logo créé par Wally Feurzeig, Cynthia Solomon et Seymour Papert au milieu des années 1960. La 'tortue' est un objet graphique qui répond à des commandes de programmation comme **forward(**distance**)** et **right(**angle**)**. Le module tortue est orienté objet et les fonctions de base tortue sont incluses dans les menus tortues.

A) Déplacer la tortue.

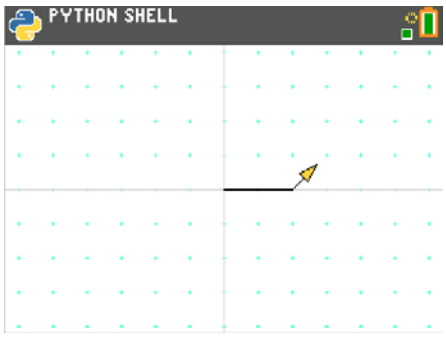
- Appuyer sur **f1 (Fns…)** afin d'accéder aux menus, puis choisir **Modul**.
- Choisir **8 : turtle…** afin d'accéder à l'ensemble des menus de la bibliothèque. Les instructions du menu **Move** (mouvement) sont assez explicites. Les distances à renseigner entre parenthèses sont en pixels.
- Compléter le script afin de faire réaliser à la tortue l'algorithme suivant :

Début de l'algorithme
t ← Turtle( )
Avance de 50 pixels
Tourne vers la gauche de 45°
Avance de 25 pixels
Fin de l'algorithme

- L'instruction **t.done( )** est nécessaire afin de pouvoir afficher la figure lorsque l'exécution du script est terminée.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Appuyer sur la touche **f4** (**Exéc**) et observer la réalisation.
La tortue est par défaut placée à l'origine du repère.



Par défaut également, la tortue est visible et vous la voyez dessiner. Noter également que la grille est visible, (l'échelle de la grille est de 25 pixels par carré).

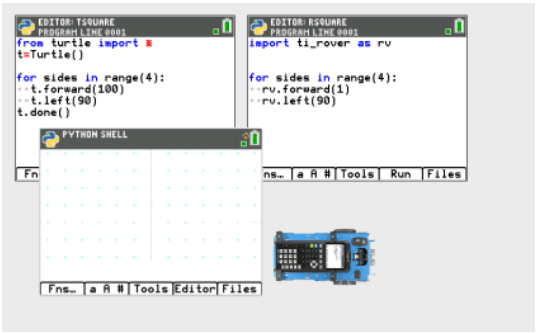
Les instructions du menu Settings permettent de modifier ces options.

- t.hideturtle() – rend la tortue invisible.
- t.showturtle() – rend la tortue visible.
- t.hidegrid() – masque la grille.

Lorsque vous avez fini d'admirer votre travail, appuyer sur la touche **[annul]** pour effacer l'écran. Remarquer que vous vous retrouvez sur un écran du Shell de Python. C'est là que votre script Python a été traité.

B) Tracer un polygone régulier.

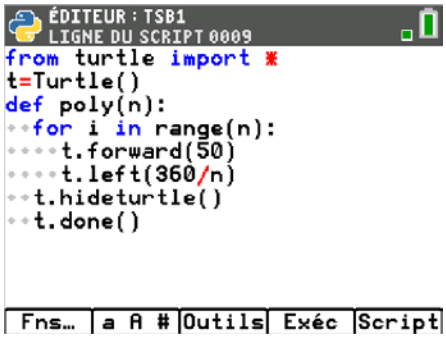
On se propose de réaliser un script permettant de tracer un polygone régulier. En effet, le fonctionnement de la tortue est analogue dans ses déplacements à celui du TI-Rover.



On propose de créer une fonction **poly(n)** permettant de tracer un polygone régulier de n côtés. Il semble raisonnable de conserver 50 pixels pour longueur de chaque côté.

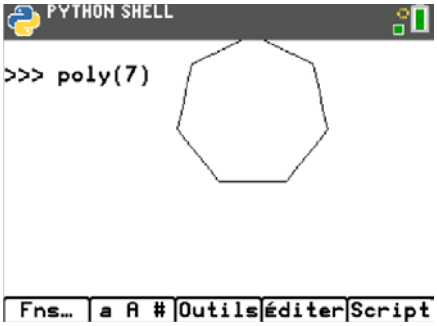
Chaque angle intérieur d'un polygone régulier de n côtés mesure $\frac{360°}{n}$.

Il est préférable de cacher la tortue à la fin de l'exécution du script.



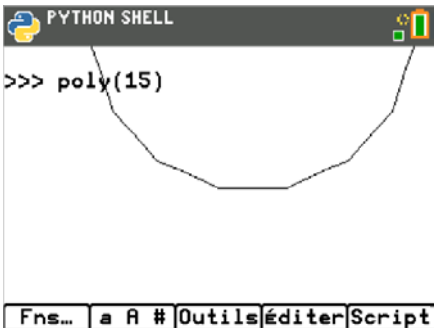
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Exécuter le script pour quelques polygones simples.
A partir d'un heptagone (7 côtés), la figure commence à ne pas être totalement visible dans l'écran.

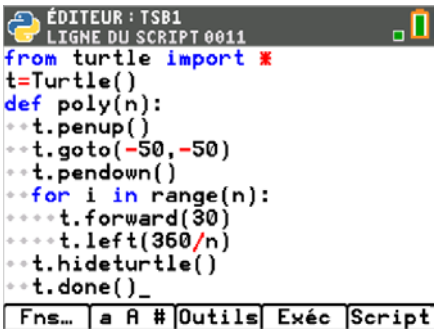


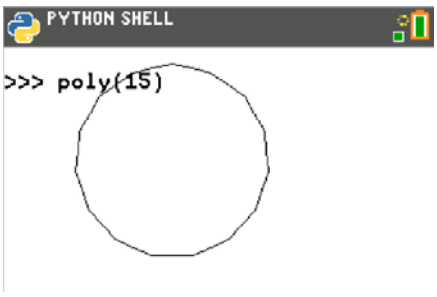
Réaliser la représentation d'un pentadécagone (15 côtés).
Comment inscrire la figure complète dans l'écran ?

- Modifier l'origine du tracé.
- Réaliser un polygone de côté 30 pixels au lieu de 50.



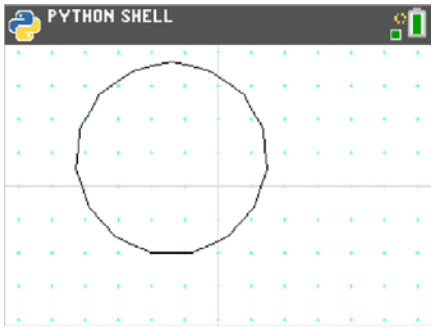
Le point de départ du tracé doit être fixé plus bas dans l'écran. Au point de cordonnées (-50 ; -50) par exemple.
Afin que ne soit tracé un segment du point de coordonnées (0,0) au point de coordonnées (-50,-50), il convient de soulever le crayon **t.penup( )**, puis de le replacer en position d'écriture une fois le changement effectué **t.pendown( )**.





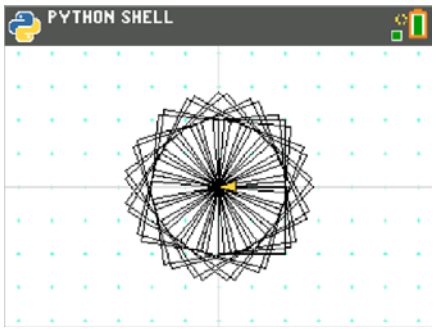
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Remarque :
Les fonctions sont très utiles en Python, mais dans le cadre de l'exécution d'un script utilisant le module Turtle, l'appel de celles-ci au sein d'un programme évitera d'avoir un affichage peu esthétique. On conserve également la représentation de la grille, ce qui a l'avantage de faciliter les réglages des paramètres des instructions de déplacement.



Un peu de « maths arts » si vous en avez le temps

- Dupliquer le script précédent (Menu **Gérer** à partir du Gestionnaire de scripts) et le nommer TSB11.
- Modifier le script afin que soit représenté un carré.
- La tortue effectue ensuite une rotation de 5° à droite.
- Le tracé est réitéré 36 fois par exemple (tracé un peu long : un essai à 18 donne aussi une belle image).



Il ne nous manque plus qu'à mettre un peu de couleur, ce que nous ferons lors de la leçon suivante.

Conseil à l'enseignant : Lorsque vous voulez travailler sur un script existant, il peut être pratique d'en faire une copie, puis de modifier le code de la copie. Ceci peut être réalisé à partir du gestionnaire de scripts 1) en sélectionnant le script à copier, puis 2) en sélectionnant l'onglet **[Gérer]**, puis 3) en sélectionnant l'option **1:Dupliquer le script...** et enfin 4) en entrant le nom du nouveau script.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 8 : Module Turtle et Python

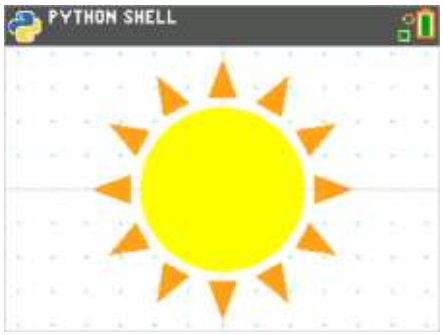
Compétence 2 : Couleur et déplacement

Dans cette leçon, vous allez poursuivre la découverte du module Turtle en utilisant les options de dessin et de déplacement de la tortue du module Turtle.

Objectifs :

- Créer un triangle isocèle et le remplir de couleur.
- Colorier un disque.
- Assembler ces deux objectifs pour créer un dessin.

Comment réaliser la figure ci-contre à l'aide d'un script Python en utilisant le module Turtle ?



Cela semble relativement simple, mais qu'en est-il vraiment ?

On observe que la figure est constituée de :

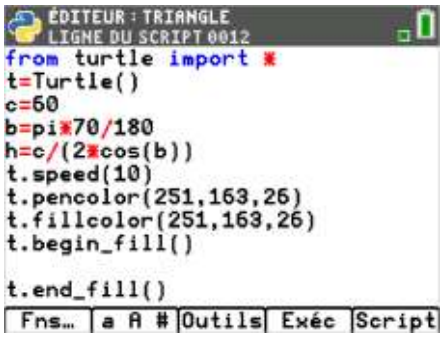
- un disque de couleur jaune centré sur l'origine du repère.
- une série de triangles isocèles équirépartis autour du disque.

Avant de fournir en complément de cette leçon quelques éléments permettant de réaliser le soleil, réalisons séparément, chaque élément.

A : Tracé d'un triangle isocèle

- Commencer un nouveau script en le nommant TRIANGLE.
- Importer la librairie de fonctions mathématiques (utile pour les calculs de trigonométrie) et la librairie Turtle.
- Créer la variable c correspondant à la longueur de la base du triangle. ($c = 60$) correspond à 60 pixels.
- Créer la variable b exprimant en radian l'angle de 70°.
- Créer la variable h, hauteur du triangle isocèle.

- Appuyer sur la touche **f1 [Fns…]** puis ← afin d'accéder au menu **Modul**.
- Choisir **8:turtle...** et dans le sous menu **Settings**, choisir le menu **5:t.speed( )** et donner le paramètre de votre choix entre parenthèses (de 0 à 10).
- A partir du sous menu **Pen** et **Draw** du module Turtle, copier les instructions **t.pencolor(r,v,b)**, **t.fillcolor(r,v,b)** permettant de choisir en code R, V, B (rouge, vert, bleu) la couleur du crayon ainsi que celle qui est utilisée pour remplir le triangle.

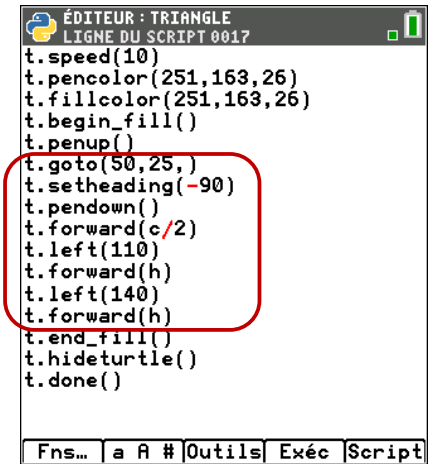


Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

- Copier également les instructions **t.begin_fill( )** et **t.end_fill( )** permettant d'établir les instants entre lesquels la méthode de remplissage débute puis se termine.

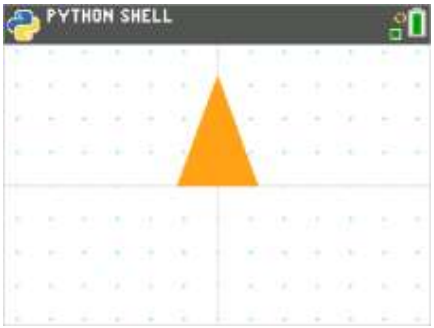
Conseil à l'enseignant : **t.speed(0-10)** – spécifie onze niveaux de vitesse de la tortue allant de 1 (lent) à 10 (rapide). La vitesse zéro (0) est le réglage le plus rapide. La valeur par défaut est 5, lorsque **t.speed( )** n'est pas spécifié.

- Dans le menu **Move** du module Turtle, recopier les instructions **t.forward( )** et **t.left( )** afin de compléter la construction du triangle.
- Par souci d'esthétisme, masquer la tortue à l'aide de l'instruction **t.hideturtle( )** située dans le menu **5 Settings**.
- Ne pas oublier l'instruction **t.done( )** pour afficher le dessin à la fin du script.



Remarque : Le tracé du triangle peut être simplifié. Il a été conçu ici de cette façon, afin d'assurer la compréhension du calcul de l'angle en se ramenant à 2 triangles rectangles et également sur le fait que le point de départ du tracé est l'origine du repère.

- Exécuter votre script et observer le tracé du triangle isocèle.
- Effectuer plusieurs tracés en modifiant les paramètres du script (valeur de c, couleurs du fond et du contour …).
- Pour construire la couronne solaire, nous devrons être capables de modifier l'orientation du triangle ainsi que sa position dans le repère.

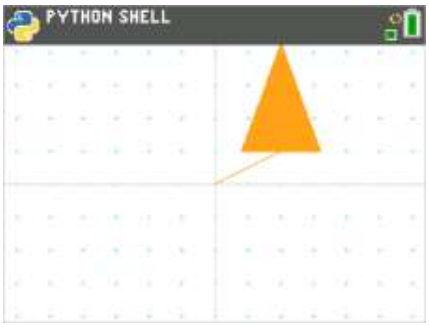


Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

a) Modification de la position

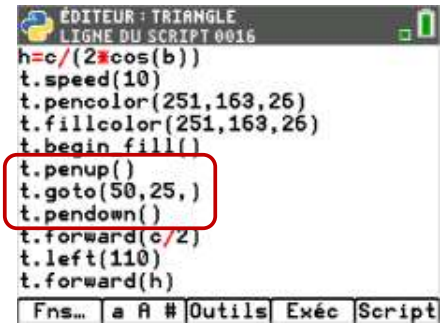
- Dans le menu **Move**, choisir l'instruction **t.goto(x,y)**. Cette instruction indique les coordonnées (x,y) du point vers lequel la tortue doit se déplacer.

- Bien entendu, la place de cette instruction s'impose. Dans notre exemple celle-ci sera avant le tracé du triangle.
- Mettons par exemple **t.goto(50,25)**.
- Exécuter à nouveau le script. Un changement de repère au point de coordonnées (50,25) a bien été effectué, avec l'inconvénient d'observer également le tracé d'un segment entre les points de coordonnées (0,0) et (50, 25).



Conseil à l'enseignant : Le code couleur (r,v,b) entré dans les instructions **t.pencolor( )** et **t.fillcolor( )** peut être fourni sous la forme d'un tuple (coul=(r,v,b)).

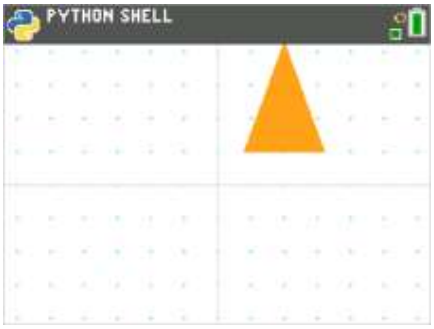
- Pour éliminer le segment entre les points de coordonnées (0,0) et (50,25) aller dans le sous menu **Pen** du module Turtle et choisir les instructions **2 : t.penup( )** et **3 : t.penddown( )** .
- Naturellement, l'instruction de déplacement de la tortue doit être située entre ces deux instructions.



Conseil à l'enseignant : t.penup() permet à la tortue de se déplacer sans tracer de ligne. **t.pendown()** utilisé après t.penup(), permet à la tortue de reprendre le tracé.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

- Exécuter le script et constater l'effet produit par ce trio d'instructions.

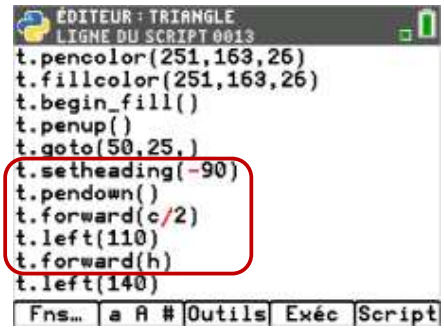


b) Modification de l'orientation

- Dans le sous menu **State** du module Turtle, choisir l'instruction **2 : t.setheadind( )**.
- Placer cette instruction avant le tracé du triangle.

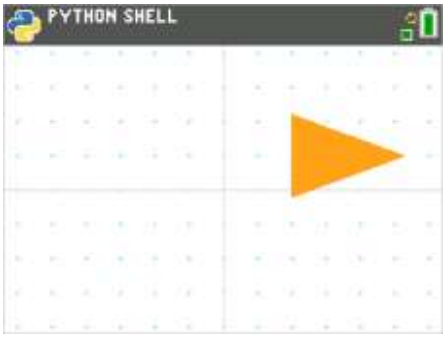
Conseil à l'enseignant : **t.setheading(degrés)** – définit le cap de la tortue en degrés. Les valeurs positives sont dans le sens inverse des aiguilles d'une montre à partir du cap zéro. Les valeurs négatives sont dans le sens des aiguilles d'une montre.

- Par rapport à la position précédente, le triangle devrait tourner de 90° vers la droite.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

- Exécuter de nouveau le script et observer le résultat obtenu.
- Lors de la réalisation du soleil, il sera ainsi possible à l'aide de cette instruction de créer la couronne de rayon. Bien entendu, nous aurons besoin de le faire au sein d'une boucle fermée.



B : Tracé du disque solaire

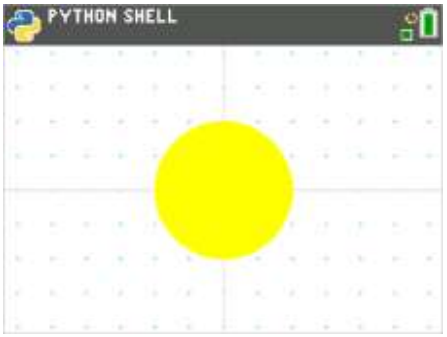
Les instructions **t.fillcolor(r,g,b)** associées à **t.begin_fill( )** et **t.end_fill( )** ne fonctionnent que pour des polygones et non pour remplir des cercles d'une couleur donnée.

t.circle( ) trace un cerle et non un disque.

En remplacement, utiliser l'instruction **t.dot(diamètre)** que l'on trouve dans le sous menu **Draw** dessinera un point de diamètre fixé en pixels comme argument de la fonction.

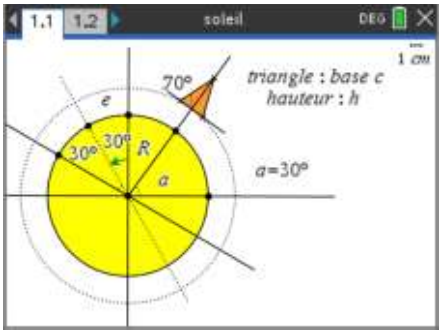
L'instruction **t.pencolor(r,g,b)** permettra de choisir la couleur du crayon utilisé pour dessiner ce point.

- Créer un nouveau script et le nommer DISQUE.
- Recopier les instructions ci-contre afin de réaliser le disque de couleur jaune.
- Tester votre script.



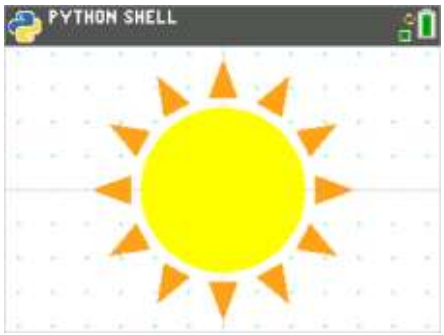
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

C : Réalisation du soleil (Pour aller plus loin)



- Créer un nouveau script et le nommer TSB2.
- La représentation ci-contre donne le nom des variables utilisées dans le script.
- Autour du cœur du soleil de rayon R, on dispose tous les 30°, un triangle isocèle de base c et de hauteur h. Ces triangles sont représentés sur un cercle de rayon $R + e$.
- Afin d'orienter correctement chaque triangle représentant un rayon du soleil, on modifie le cap de la tortue à l'aide de l'instruction **t.setheading( )**.
- La position initiale de la tortue correspond à un angle $\alpha = 0°$ à partir d'une orientation **t.setheading(-90°)**.
- Le script complet reprend en tout point ce qui a été vu dans cette leçon.

Exécuter le script, admirer le résultat obtenu.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 8 : Module Turtle et Python

Compétence 3 : Résoudre un problème

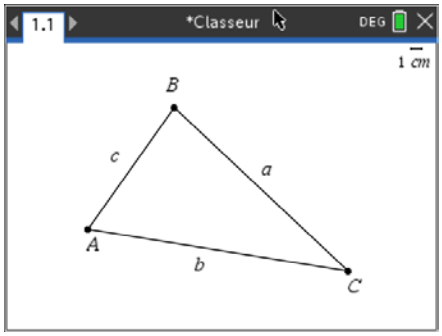
Dans cette troisième leçon, vous allez écrire un programme vous permettant d'utiliser la bibliothèque Turtle afin d'illustrer la résolution d'un problème élémentaire de mathématiques.

Objectifs :

- Résoudre un triangle quelconque.
- Dessiner une représentation géométrique.

On se propose de résoudre un triangle scalène en utilisant les relations du mathématicien Al Kashi.

Ce résultat porte le nom du mathématicien persan du début du XVe siècle al-Kâshî. Appelé aussi « *loi des cosinus* », il généralise le **théorème de Pythagore** aux triangles non rectangles. Le **théorème d'al-Kâshî** s'énonce ainsi :



Soit *ABC* un triangle et soit α une mesure (en radians) de l'angle au sommet *A*. Alors : $AB^2 + AC^2 = BC^2 + 2 \times AB \times AC \cos(\alpha)$

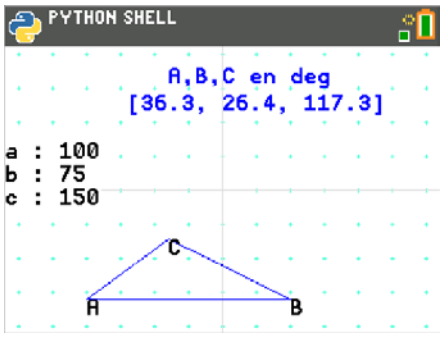
Afin de simplifier, on adopte la notation de la figure ci-contre, ce qui permet d'établir pour chaque côté du triangle, les relations :

$$a^2 = b^2 + c^2 - 2bc \times \cos(\hat{A})$$
$$b^2 - a^2 + c^2 - 2ac \times \cos(\hat{B})$$
$$c^2 = a^2 + b^2 - 2ab \times \cos(\hat{C})$$

Connaissant la longueur des 3 côtés du triangle, écrire un script qui :

- Affiche la représentation du triangle.
- Affiche la mesure des angles en degrés.

La longueur des côtés sera donnée en pixels.

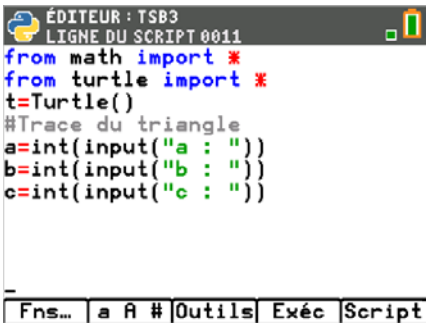


Conseil à l'enseignant : Les mesures proposées devront respecter l'inégalité triangulaire. C'est-à-dire que la longueur d'un côté doit être inférieure ou égale à la somme des deux autres.

Il sera toujours possible d'enrichir le script par une éventuelle gestion de l'erreur renvoyée en cas de non-respect de cette consigne.

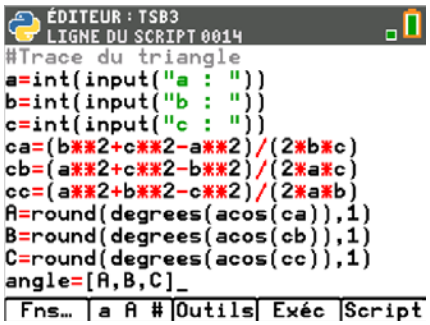
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

- Créer un nouveau script et le nommer **TSB3**.
- Importer la bibliothèque Turtle.
- Créer les variables a , b, c correspondant aux longueurs.
- (pixels) des côtés du triangle.

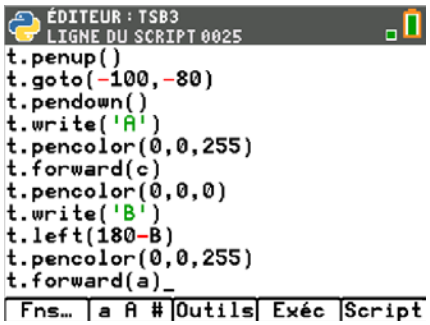


Conseil à l'enseignant : L'entrée des longueurs des côtés du triangle comme paramètre d'une fonction python est toujours intéressante, mais dans cette leçon, on souhaite avoir la représentation du triangle tout en conservant l'affichage de la grille de la librairie Turtle. Attention, l'appel d'une fonction à l'intérieur d'un script utilisant des instructions de cette librairie supprime l'affichage de la grille et superpose le prompt **>>>**.

- Calculer les cosinus des angles $\hat{A}, \hat{B}, \hat{C}$ respectivement nommés **ca**, **cb**, **cc**.
- En déduire les valeurs des angles $\hat{A}, \hat{B}, \hat{C}$ arrondies au dixième.



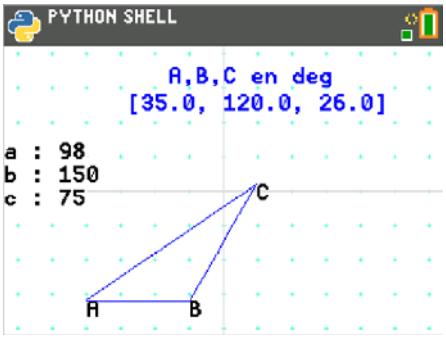
- L'origine du tracé est choisie au point de coordonnées (-100 , -80).
- Les noms des sommets sont écrits à l'aide de l'instruction **t.write( )** située dans le menu DRAW de la librairie Turtle.
- L'instruction **t.pencolor(0, 0, 0)** permet d'attribuer la couleur noire au stylo.



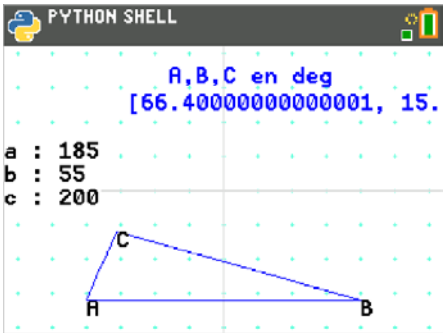
Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

- Les valeurs des angles stockées dans une liste sont affichées à partir du point de coordonnées (-70 , 70).

- Selon certaines valeurs de *a, b, c*, le calcul des angles arrondis au dixième sera à modifier à une valeur entière afin d'éviter des problèmes d'affichage des résultats.



- Ce type d'affichage n'est pas une erreur, mais provient de la représentation des nombres en virgule flottante.



Conseil à l'enseignant : La représentation des nombres en virgule flottante a été développée par JB Civet et Boris Hanüs dans l'ouvrage « Enseignement de spécialité : Mathématiques » Editions Eyrolles.
https://www.eyrolles.com/Entreprise/Livre/enseignement-de-specialite-mathematiques-9782416003202/

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Unité 8 : Module Turtle et Python

Application : Assemblons nos connaissances

Dans cette leçon, vous allez assembler les connaissances vues dans les leçons précédentes afin de réaliser un script permettant de concevoir vous-même la réalisation d'un projet.

Objectifs :

- Appliquer les notions vues dans les leçons précédentes.
- Réaliser une figure complexe nécessitant de la programmation fonctionnelle.

Dans les leçons précédentes, nous avons vu que l'utilisation de la librairie Turtle dans un script Python afin de construire une figure géométrique nécessitait fréquemment la mise en place de calculs.

La clarté du script impose que celui-ci soit commenté, mais également que chaque tâche complexe soit décomposée en un certain nombre de tâches élémentaires.

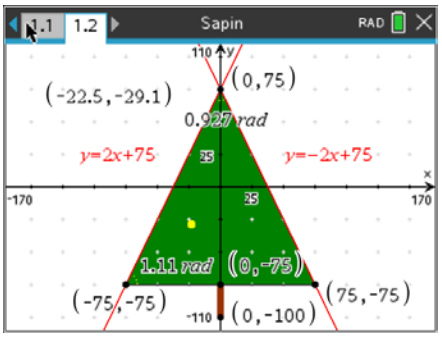
Dans cette leçon, nous allons réaliser une décoration de Noël, le programme principal appelant des fonctions pour :

- Dessiner et colorier un triangle isocèle.
- Dessiner le pied du sapin.
- Tracer et colorier une étoile.
- Décorer et animer le sapin.

Conseil à l'enseignant : La programmation fonctionnelle favorise auprès des étudiants réunis en groupe, la réalisation de projets complexes. Chaque groupe travaillant à la réalisation d'un élément particulier de celui-ci.

Mise en place du projet

Tout projet comporte une sorte de cahier des charges. Pour celui-ci utilisant le module Turtle et affichant un objet graphique, les contraintes sont aussi celles de la taille de l'écran (grille d'unité 25 pixels).



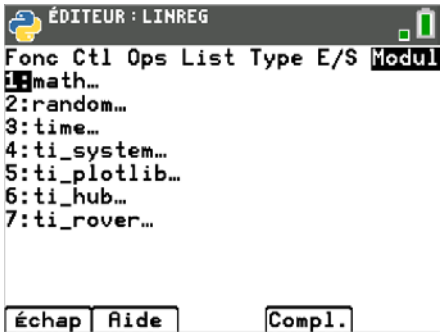
- La décoration est constituée de points de couleurs **t.dot(diamètre)**, la taille et la position étant aléatoires.
- Les points de couleurs doivent être contenus dans le triangle isocèle.
- La représentation préalable à l'aide d'un logiciel de géométrie dynamique sur lequel le repère orthonormal est identique à la grille de la librairie Turtle facilite le travail de recherche.
- Ainsi les points de coordonnées (-75 , 75) ; (75 , -75) et (0 , 75) seront les sommets du triangle.
- Les droites d'équation réduites $y = 2x + 75$, $y = -2x + 75$ et $y = -75$ délimitent le plan de représentation des points de couleur.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Conseil à l'enseignant : Selon les souhaits, il est possible de réaliser un sapin dont la largeur de la base est variable $(-a\,,b)$ étant les coordonnées du premier point, les équations des droites, les mesures des angles peuvent être calculés dans chaque fonction. On peut aussi envisager de faire varier la position de l'origine du tracé et représenter plusieurs sapins sur le même graphique.

Création du script

- Commencer un nouveau script et le nommer SAPIN.
- Appuyer sur **Fns…** puis dans les modules complémentaires (**F4**), sélectionner le module Turtle.

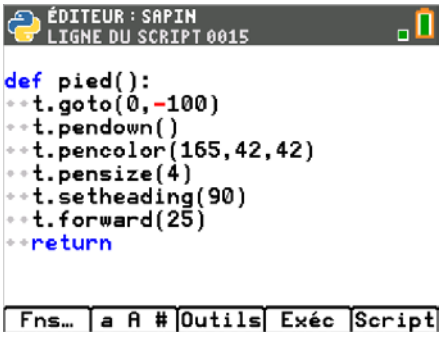


- Importer également le module **time** dont nous aurons besoin pour réaliser une petite animation.
- Importer le module **random** pour le calcul aléatoire des positions et des nombres des objets colorés.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Dessin du pied du sapin

- Le dessin du pied commence au point de coordonnées (0,-100) **t.goto(0,-100)**.
- Il a pour longueur 25 pixels **t.forward(25)**.
- Afin de le tracer le plus simplement possible, choisir la taille la plus épaisse du crayon **t.pensize(4).**
- Orienter la tortue vers les $y > 0$ **t.setheading(90)**.
- La couleur du crayon est un marron correspondant au code RVB suivant (165, 42, 42) et l'instruction pour l'utiliser est **t.pencolor( )**.



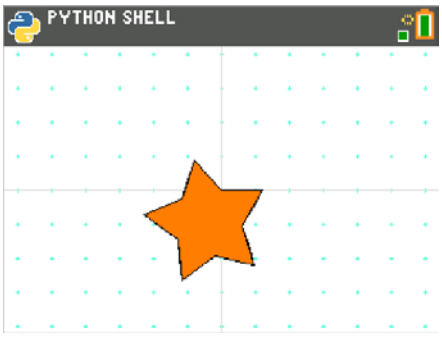
Toutes les instructions sont disponibles en appuyant sur **Fns…**, puis **Modul** et en choisissant **8 : Turtle** (Voir Compétence 1).

Dessin de l'étoile

Pour la fonction **etoile( )**, nous aurions pu rappeler le script de la **compétence 1** au cours de laquelle nous avons dessiné des polygones réguliers. Le rappel d'un autre script s'effectue à l'aide de l'instruction **from SCRIPT import*** disponible dans le catalogue.

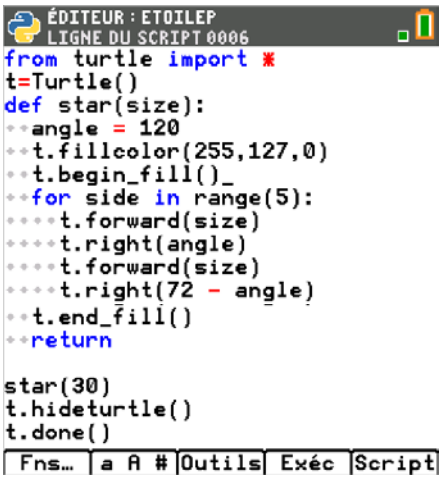
- L'étoile est fixée au sommet du sapin au point de coordonnées (0, 75).
- Les branches sont remplies **t.fillcolor( )** d'une teinte orangée de code RVB (255,127,0).
- L'instruction **t.fillcolor( )** termine la coloration des branches de l'étoile.
- À chaque itération, déplacer la tortue de 30 pixels vers l'avant et la déplacer à droite de 144 degrés.
- Cela constituera un angle de **36** degrés à l'intérieur d'une étoile.
- **5** itérations constitueront parfaitement une étoile.

Conseil à l'enseignant : Comment réaliser une étoile pleine ?
Il s'agit en fait d'un polygone pour lequel on doit tracer 2 lignes pour chaque côté. Les angles sont alors incrémentés de 72 (360/5).



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

Le script proposé ci-contre réalise le tracé d'une étoile pleine à l'aide d'une fonction.
Selon le temps accordé à votre projet, vous pourrez choisir le type d'étoile à placer au sommet de votre sapin.



Dessin du sapin

L'enveloppe de celui-ci est constituée d'un simple triangle isocèle comme nous l'avons vu dans la mise en place du projet.

L'ensemble des mesures (valeurs des angles, longueur des côtés..) sont issues de ce travail préparatoire.

Les valeurs des angles mesurées en radian lors de l'utilisation du logiciel de géométrie dynamique sont converties en degrés.

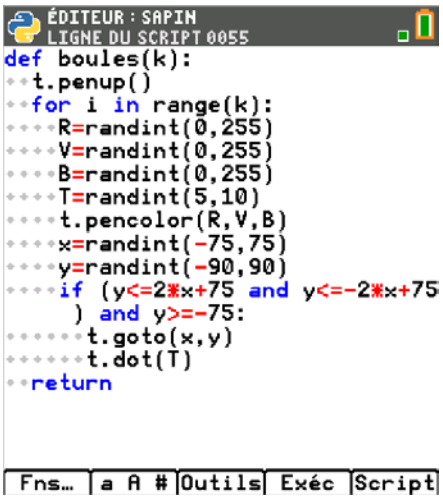
Le triangle isocèle est rempli de couleur « vert sapin », dont le code RVB est : (1, 121, 111).

Conseil à l'enseignant : la non utilisation de l'instruction **degrees( )** de la librairie **math…** est volontaire. Les jeunes élèves doivent être capables de convertir une mesure d'angle d'une unité vers une autre.

Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

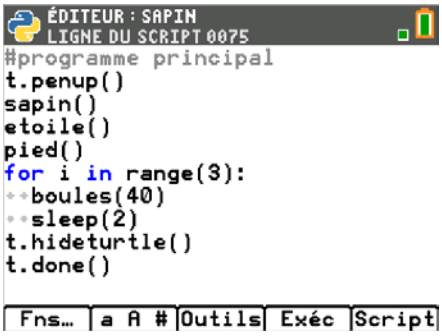
Dessin des boules de Noël

- On dispose de k boules de couleur R, V, B à placer à l'intérieur du triangle isocèle. Chaque composante de couleur est un nombre entier aléatoire compris entre 0 et 255 (Le codage d'une composante étant sur 8 bits, soit 2^8 possibilités).
- La taille d'une boule de couleur est aléatoirement choisie entre 5 et 10 pixels
- L'instruction conditionnelle permet de limiter l'emplacement de chaque décoration à l'intérieur du triangle isocèle. (voir **mise en place du projet**).
- Si on souhaite mettre des guirlandes à la place de boules de Noël, supprimer ou commenter l'instruction **t.penup( )** de la fonctions **boules( )**.



Le programme principal

- Chaque fonction élémentaire est exécutée, il n'y a pas d'ordre particulier.
- La boucle permet de réaliser une petite animation de 6 secondes
- L'instruction **t.hideturtle( )** cache la tortue et permet d'obtenir une belle image.
- L'instruction **t.done( )** laisse l'image affichée. L'appui sur la touche **annul** réaffichera le **prompt** >>>.



Ce document est mis à disposition sous licence Creative Commons
http://creativecommons.org/licenses/by-nc-sa/2.0/fr/
© 2022 Texas Instruments / Photocopie autorisée

education.ti.france

@TIEducationFR

TIedtechFR

Contactez notre service client TI-Cares
education.ti.com/fr/csc
01 41 04 60 40

education.ti.com/fr

© 2022 Texas Instruments