SNT - Informatique embarquée et objets connectés

Partie I- Micro-controleur et programmation

1. Présentation de la carte BBC micro:bit

Un **micro-controleur** tel qu'Arduino (utiliser en SPC), Raspberry (au Fablab) ou Micro:bit (en SNT) sont des mini-ordinateurs constitués

- D'un microprocesseur (CPU : Central Processor Unit)
- de mémoire de type RAM (ici 128 octets seulement)
- de mémoire non volatile pour stocker le programme (c'est le disque dur pour un ordinateur).
- des ports d'entrées / sorties pour communiquer avec l'extérieur (USB, radio)
- un convertisseur analogique-numériques (CAN)

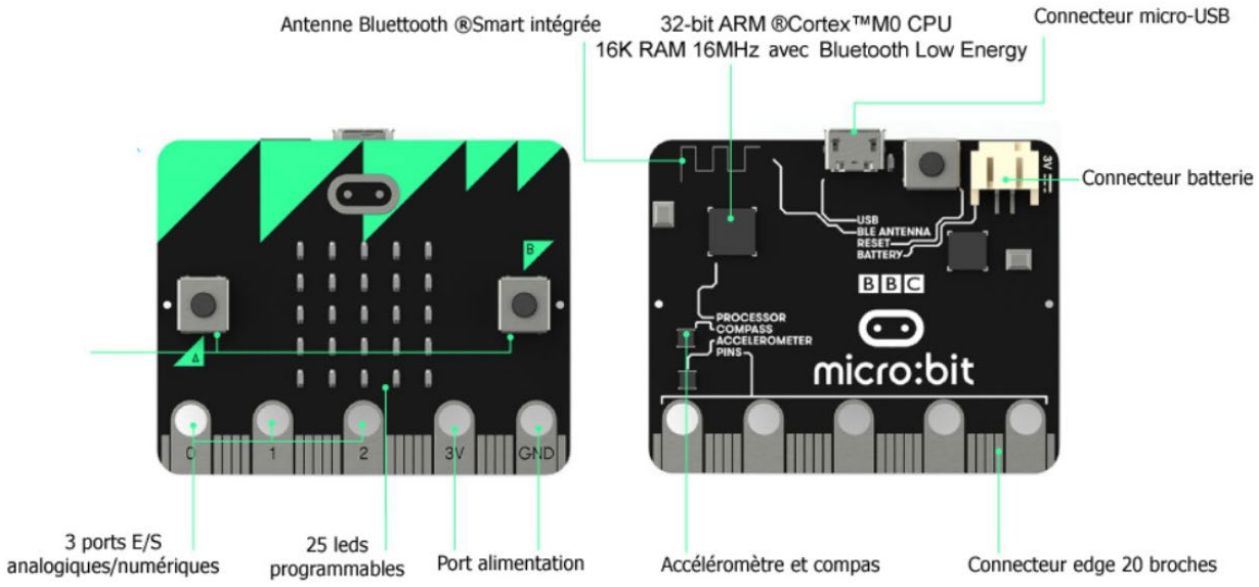
qui vont interpréter les commandes dictées par **un programme, script** que vous allez écrire.

BBC micro:bit est une carte à microcontrôleur conçue en 2015 au Royaume-Uni pour développer l'apprentissage de l'algorithmique et de la programmation. Pourvu de **capteurs et d'actionneurs**, ce petit ordinateur possède la dernière technologie qui équipe les appareils modernes : *téléphones mobiles, réfrigérateurs, montres intelligentes, alarmes antivol, robots, etc...*

Ainsi, il s'apparente à ce que l'on nomme l'**Internet des objets** : Internet of Things, abrégé **IoT**.

La carte micro:bit dispose des spécificités techniques (https://microbit.org/fr/guide/features/) suivantes :

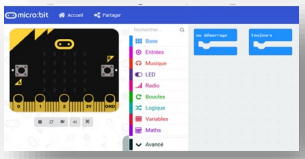
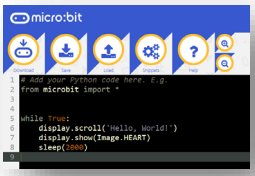
- 25 LEDs programmables individuellement
- 2 boutons programmables
- Broches de connexion
- Capteurs de lumière et de température
- Capteurs de mouvements(accéléromètre/boussole)
- Communication sans fil, via Radio et Bluetooth
- Interface USB



2. Présentation de Mu éditor

Pour programmer le micro-controleur, vous allez utiliser le logiciel **Mu editor** pour **programmer de petits scripts** qui seront déversés (Flashés) sur le micro-controleur . Il exécutera alors les lignes de votre script !

3 façons de programmer :

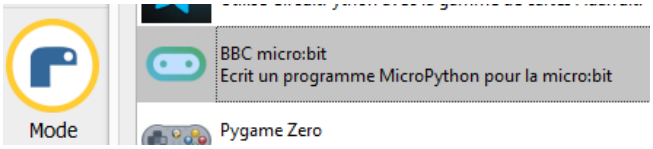
Avec PYTHON en LOCAL	Par BLOCK	Avec PYTHON en ligne
avec Mu éditor	https://makecode.microbit.org/#editor	**https://python.microbit.org/v/1**
		
« FLASHER » après avoir Vérifié	« télécharger» après avoir simulé	« download »

Dans tous les cas, la micro:bit devra être **reliée au PC via un cable USB** et le script, une fois **enregistré,** devra être **déversée (Flasher)** sur la carte.

Nous allons en classe privilégier **Mu editor** qui ne nécessite pas de liaison internet !

Il faudra .

1. **Lancer Mu editor,**  **sélectionner le mode BBC:microbit**



2. **Ecrire les lignes de votre script**
3. **Vérifier le code**  **et**

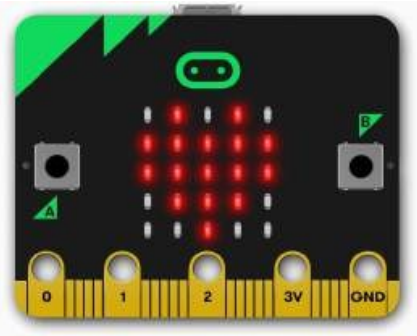
4. **Le Flasher sur votre micro:bit** 

5. **Enfin le Tester !**

3. Découverte des fonctionnalités !

3.1 Commandes de base de l'afficheur, matrice de 5x5 LEDs

LED signifie Light Emitting Diode, Diode électroluminescente. La carte micro:bit en dispose de 25, toutes programmables individuellement, ce qui permet d'afficher du texte, des nombres et des images.



3.1.1 Afficher un texte "défilant" `display.scroll(string, delay=400)`

Nous allons commencer par afficher quelques informations sur l'afficheur.

```
from microbit import *
display.scroll("SNT")
```

La première ligne de ce programme importe la bibliothèque de fonctions micro:bit.
La deuxième ligne fait défiler un message à l'écran. Cela n'arrive qu'une seule fois.

- ✓ **Exercice 1: Modifier le programme précédent pour qu'il affiche le texte de ton choix.**

La vitesse de défilement peut être ralentie ou accélérée à l'aide du paramètre *delay* . Plus le nombre est grand, plus le défilement est lent.

```
from microbit import *
display.scroll("Hello Guy and Girl ", delay=20)
```

- ✓ **Exercice 2: Le défilement de la phrase du programme ci-dessus est trop rapide pour pouvoir la lire correctement. Modifie la valeur du paramètre delay pour qu'on puisse la lire facilement.**

3.1.2 Afficher une "image" `display.show(image)`

Exécuter le programme suivant:

```
from microbit import *
display.show(Image.SAD)
```

- ✓ **Exercice 3: On constate que la carte est un peu triste. Modifier le programme précédent pour lui redonner de la joie.**

Prolongement: essayer plusieurs images intégrées à l'aide de la liste des images intégrées à la bibliothèque de micro :bit

Image.HEART Image.HEART_SMALL Image.HAPPY Image.SMILE Image.SAD Image.CONFUSED Image.ANGRY Image.ASLEEP Image.SURPRISED Image.SILLY Image.FABULOUS Image.MEH Image.YES Image.NO Image.CLOCK12, Image.CLOCK11, Image.CLOCK10, Image.CLOCK9, Image.CLOCK8, Image.CLOCK7, Image.CLOCK6, Image.CLOCK5, Image.CLOCK4, Image.CLOCK3, Image.CLOCK2, Image.CLOCK1 Image.ARROW_N, Image.ARROW_NE, Image.ARROW_E, Image.ARROW_SE, Image.ARROW_S, Image.ARROW_SW, Image.ARROW_W, Image.ARROW_NW Image.TRIANGLE Image.TRIANGLE_LEFT

Image.CHESSBOARD Image.DIAMOND Image.DIAMOND_SMALL Image.SQUARE Image.SQUARE_SMALL Image.RABBIT Image.COW Image.MUSIC_CROTCHET Image.MUSIC_QUAVER Image.MUSIC_QUAVERS Image.PITCHFORK Image.XMAS Image.PACMAN Image.TARGET Image.TSHIRT Image.ROLLERSKATE Image.DUCK Image.HOUSE Image.TORTOISE Image.BUTTERFLY Image.STICKFIGURE Image.GHOST Image.SWORD Image.GIRAFFE Image.SKULL Image.UMBRELLA Image.SNAKE

Créer sa propre image

Chaque pixel LED sur l'affichage physique peut prendre une parmi dix valeurs.

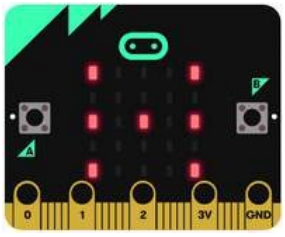
- Si un pixel prend la valeur 0 c'est qu'il est éteint. Littéralement, il a une luminosité de zéro.
- En revanche, s'il prend la valeur 9 il est à la luminosité maximale.
- Les valeurs de 1 à 8 représentent des niveaux de luminosité entre éteint (0) et « au maximum » (9).

```
from microbit import *

bateau = Image("05050:"
               "05050:"
               "05050:"
               "99999:"
               "09990")

display.show(bateau)
```

Comment dessiner une image? Chaque ligne de l'affichage physique est représentée par une ligne de nombres se terminant par : et entourée de guillemets doubles " . Chaque nombre indique une luminosité. Il y a cinq lignes de cinq nombres donc il est possible de spécifier la luminosité individuelle de chacune des cinq LED sur chacune des cinq lignes sur l'affichage physique. C'est ainsi que l'on crée une image.

- ✓ **Exercice 4: Le programme précédent crée et affiche l'image d'un bateau à deux mâts. Le modifier (en le recopiant) un bateau à un seul mât.**

- ✓ **Exercice 5: Rédiger ci-dessous le programme qui affiche l'image suivante:**




```
# Ecrire votre programme
```

Remarque: Nous pouvons aussi écrire les images en une seule ligne:

```
from microbit import *

bateau = Image("05050:05050:05050:99999:09990")

display.show(bateau)
```

3.1.3 Les pixels (`display.set_pixel(x, y, val)`)

Vous pouvez régler la luminosité des pixels de l'affichage individuellement de 0 (désactivé) à 9 (luminosité maximale). Pour des informations sur les coordonnées de l'affichage, voir le guide pour matrice à LED (https://microbit.org/guide/hardware/leds/).

Exécuter le programme suivant:

```
from microbit import *
display.set_pixel(1, 4, 9)
```

- ✓ **Exercice 6: Recopier le programme précédent ci-dessous et le modifier pour allumer la LED du centre de la matrice.**

```
# Ecrire votre programme
```

3.2 Boucle `while`

Le programme suivant utilise une boucle `while` pour faire clignoter le pixel central de manière répétée sur l'écran. La boucle `while` se répète tant que la condition spécifiée est vraie (`True`). Dans ce cas, nous avons dit que la condition est vraie. Cela crée une *boucle infinie*. Le code qui doit être répété est en retrait (c'est une "indentation" du texte).

L'instruction de veille `sleep()` provoque la pause du micro:bit pendant un nombre défini de millisecondes choisi entre parenthèses.

L'instruction `display.clear()` éteint l'affichage.

Exécuter le programme ci-dessous:

```
from microbit import *
while True:
    display.set_pixel(2, 2, 9)
    sleep(500)
    display.clear()
    sleep(500)
```

Dans le programme suivant que vous exécuterez, on importe le module `random` de MicroPython et on l'utilise pour afficher un pixel au hasard sur la matrice.

```
from microbit import *
import random
n=random.randint(0,4)
p=random.randint(0,4)
display.set_pixel(n, p, 9)
```

Tester le programme précédent plusieurs fois de suite.

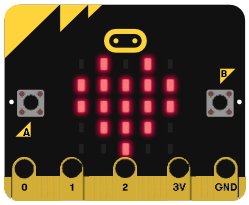
Pour cela, redémarrer la micro:bit en appuyant sur le bouton `RESET` situé à l'arrière de la carte.

- ✓ **Exercice 7: Ecrire un programme ci-dessous qui allume successivement et indéfiniment des**

pixels au hasard à l'écran (aide: utiliser une boucle infinie).

```
# Ecrire votre programme
```

- ✓ **Exercice 8: Ecrire ci-dessous un programme qui fait clignoter un cœur indéfiniment (voir illustration ci- dessous).**

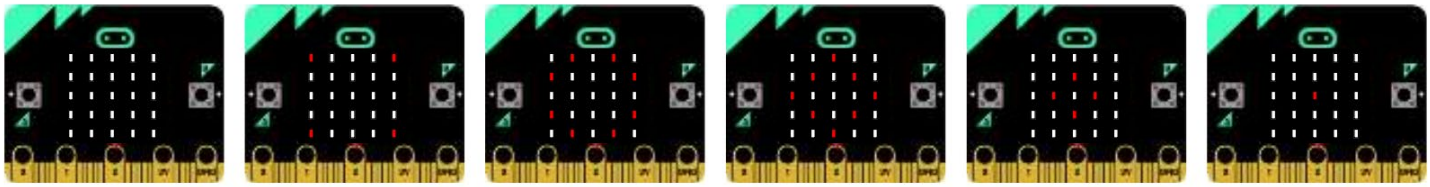



```
# Ecrire votre programme
```

Créer une animation

En affichant plusieurs images successives, on peut réaliser une animation.

- ✓ **Exercice 9: Ecrire un programme qui réalise l'animation suivante:**



```
# Ecrire votre programme
```

3.3 Boucle `for`

Le programme suivant utilise une boucle `for` pour faire défiler un pixel sur une ligne. Exécutez-le.

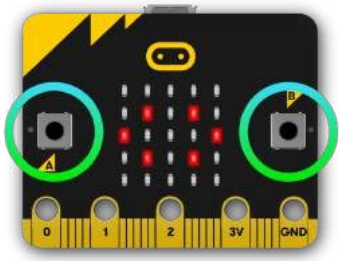
```
from microbit import *
while True:
    for i in range(5):
        display.set_pixel(i,0,9)
        sleep(200)
        display.clear()
```

- ✓ **Exercice 10 (la double boucle): S'inspirer du programme précédent pour réaliser un programme ci-dessous qui fait défiler un pixel sur tout l'écran.**

```
# Ecrire votre programme
```

3.4 Les entrées boutons A, B et A+B

Programmation évènementielle



Il y a deux boutons sur la face avant du micro:bit (étiquetés A et B). On peut détecter quand ces boutons sont pressés, ce qui permet de déclencher des instructions sur l'appareil.

Exemples avec le boutton A:

- `button_a.is_pressed()` : renvoie *True* si le bouton spécifié est actuellement enfoncé et *False* sinon.
- `button_a.was_pressed()` : renvoie *True* ou *False* pour indiquer si le bouton a été appuyé depuis le démarrage de l'appareil ou la dernière fois que cette méthode a été appelée.

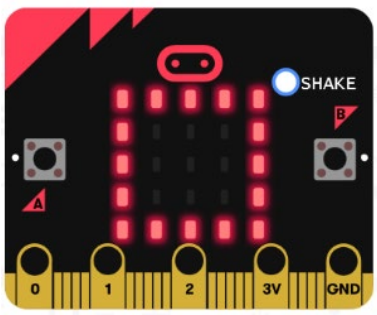
Exemple : Essayer le programme suivant qui fait défiler le texte **"SNT"** indéfiniment.
On introduit l'**instruction conditionnelle** `if` qui va tester si le bouton A été pressé (pendant le défilement du texte ou pendant la pause), auquel cas le programme s'arrête en exécutant la commande `break` .

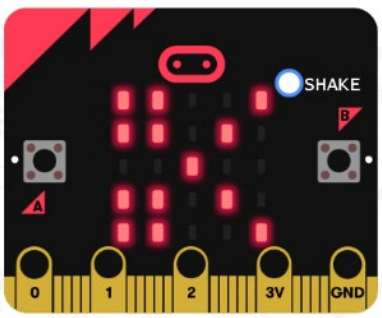
```
from microbit import *
while True:
    display.scroll("SNT")
    sleep(200)
    if button_a.was_pressed():
        break
```

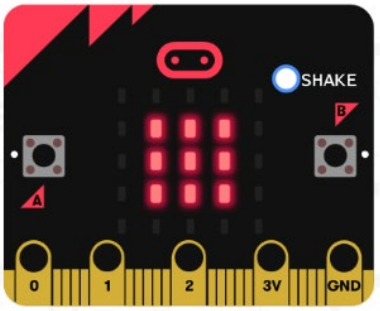
Instructions conditionnelles `if` , `elif` , `else`
Voici comment se structure une instruction conditionnelle. Selon la situation, il n'est pas forcément nécessaire d'utiliser `elif` ou `else` .

```
if quelque chose est vrai (``True``):
    # fais un truc
elif autre chose est vrai (``True``):
    # fais un autre truc
else:
    # fais encore autre chose.
```

✓ **Exercice 11 :** ***Pierre ciseaux feuille !***







✓ Compléter le programme suivant dans lequel une pression simultanée sur les boutons A et B affichera

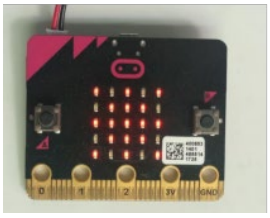
une image de ciseaux. Sinon si, une pression sur le bouton A affichera une image de pierre. Sinon si, une pression sur le bouton B affichera une image de feuille. Il faudra créer vous-même l'image de la *pierre* et de la *feuille* avec un temps d'affichage de 0,5 seconde.

```
# Ecrire votre programme ici

from microbit import *

pierre =

feuille =

ciseaux = Image("99009:"
                "99090:"
                "00900:"
                "99090:"
                "99009:")

while True:
    if button_a.is_pressed() and button_b.is_pressed():
        display.show(ciseaux)
        sleep(500)
    elif button_a.is_pressed():
        display.show(pierre)
        sleep(500)
    elif button_b.is_pressed():
        display.show(feuille)
        sleep(500)
    display.clear()
    sleep(100)
```

3.5 Capteur de lumière

En inversant les LEDs d'un écran pour devenir un point d'entrée, l'écran LED devient un capteur de lumière basique, permettant de détecter la luminosité ambiante.

La commande `display.read_light_level()` retourne un entier compris entre 0 et 255 représentant le niveau de lumière.



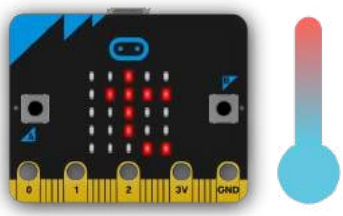
Teste le programme ci-dessous qui affiche une image **de lune** si on baisse la luminosité (en recouvrant la carte avec sa main par exemple) et **un soleil** sinon.

```
# Ecrire votre programme ici

from microbit import *

soleil = Image("90909:"
               "09990:"
               "99999:"
               "09990:"
               "90909:")

lune = Image("00999:"
             "09990:"
             "09900:"
             "09990:"
             "00999:")

while True:
    if display.read_light_level()>50:
        display.show(soleil)
    else:
        display.show(lune)
    sleep(10)
```

Prolongement: créer un programme qui affiche le niveau de luminosité et le tester avec la LED d'un téléphone portable ou une lampe-torche par exemple. Plus la luminosité sera élevée, plus il y aura de LEDs affichées sur la matrice.

3.6 Capteur de température

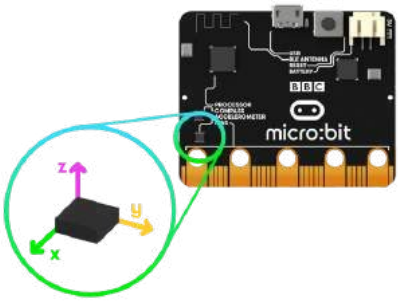
Le micro:bit n'a pas un capteur de température dédié. Au lieu de cela, la température fournie est en fait la température de la puce de silicium du processeur principal. Comme le processeur chauffe peu en fonctionnement (c'est un processeur ARM à grande efficacité), sa température est une bonne approximation de la température ambiante. L'instruction **`temperature()`** renvoie la température de la carte micro:bit en degrés Celsius.



- ✓ **Exercice 12: Ecrire un programme qui affiche la température (aide: on pourra utiliser l'instruction `display.scroll(.....)` ; revoir le point 3.4**

```
# Ecrire votre programme
```

3.7 Accéléromètre



Un accéléromètre mesure l'accélération de la carte micro:bit, ce composant détecte quand la micro:bit est en mouvement. Il peut aussi détecter d'autres actions (gestes), par exemple quand elle est secouée, inclinée ou qu'elle tombe.

Si tu t'es déjà demandé comment un téléphone portable sait dans quel sens afficher les images sur son écran, c'est parce qu'il utilise un accéléromètre. Les manettes de jeux contiennent aussi des accéléromètres pour t'aider à tourner et à te déplacer dans les jeux.

La carte micro:bit est munie d'un accéléromètre. Il 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.

Dans l'exemple suivant à essayer, l'instruction `accelerometer.get_x()` permet de détecter un mouvement de gauche à droite en renvoyant un nombre compris entre -1023 et 1023; 0 étant la position "d'équilibre"

```
#Exemple
from microbit import *

while True:
    abscisse = accelerometer.get_x()
    if abscisse > 500:
        display.show(Image.ARROW_E)
    elif abscisse < -500:
        display.show(Image.ARROW_W)
    else:
        display.show("-")
```

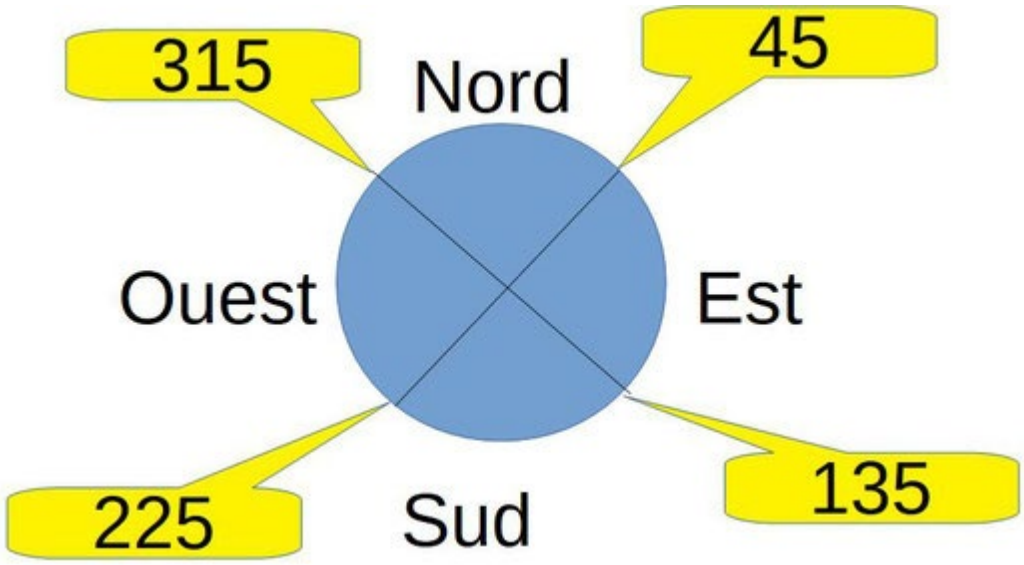
✓ **Exercice 13: Compléter le programme suivant et comprendre son fonctionnement**

```
# Ecrire votre programme ici
from microbit import *

while True:
    abscisse = accelerometer.get_x()
    ordonnee = accelerometer.get_y()
    if abscisse > 500:
        display.show(Image.ARROW_E)
    elif abscisse < -500:
        display.show(Image.ARROW_W)
    elif ordonnee > 500:
        display.show(Image.ARROW_S)
    elif "remplir ici":
        "remplir ici"
    else:
        display.show("-")
```

3.8 Boussole

La boussole détecte le champ magnétique de la Terre, nous permettant de savoir quelle direction la micro:bit indique. La boussole doit être étalonnée avant de pouvoir être utilisée. Pour cela, on utilise `compass.calibrate()` qui exécute un petit jeu: au départ, micro:bit fait défiler "Tilt to fill screen". Ensuite, incliner micro:bit pour déplacer le point au centre de l'écran autour jusqu'à ce que vous ayez rempli la totalité de l'écran.

La fonction `compass.heading()` donne le cap de la boussole sous la forme d'un entier compris entre 0 et 360, représentant l'angle en degrés, dans le sens des aiguilles d'une montre, avec le nord égal à 0.



✓ **Exercice 14: Compléter le programme suivant qui indique le Nord.**

```
# Ecrire votre programme ici

from microbit import *

compass.calibrate()

while True:
    if compass.heading() < "remplir ici" or compass.heading() > "remplir ici":
        display.show(Image.ARROW_N)
    else:
        display.show(Image.DIAMOND_SMALL)
```

✓ **Exercice 15: Compléter le programme pour qu'il affiche aussi les directions Sud et Ouest**

```
# Ecrire votre programme ici

from microbit import *

compass.calibrate()

while True:
     angle = compass.heading()
    if 315 <= "angle or angle <= 45:
        display.show('N')
    elif 45 < "angle or angle <= 135:
        display.show('E')
        "remplir ici"


    #attente d'une seconde
    sleep(1000)
```