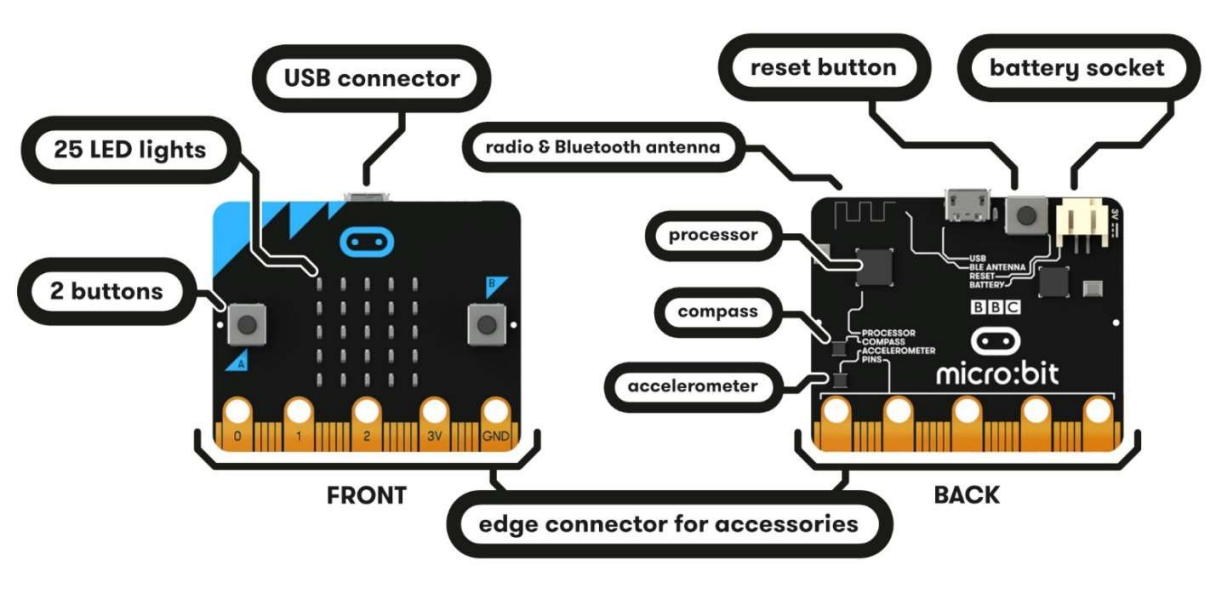
Nom: Classe:

Découverte de la carte Micro:Bit

1. Introduction

Cette activité propose une initiation à la programmation python de la carte micro :bit.

Les schémas ci-dessous résument de façon visuelle l'ensemble des ressources matérielles disponibles sur la carte BBC Micro:Bit :



Nous allons utiliser le logiciel MU Editor.

MU editor est un logiciel permettant de déposer directement le microprogramme sur la carte, sans avoir à passer par l'étape manuelle de dépôt du fichier .HEX et il permet également de recevoir et d'envoyer des données en temps réel à la carte (on appelle cela la **console REPL**). Aller sur le site https://codewith.mu/ et suivre les instructions pour l'installation. Ouvrir **MU editor.**

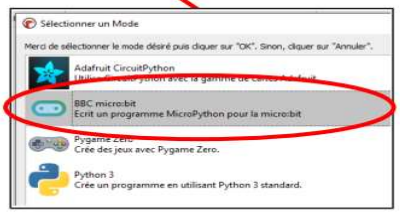
(Pour les versions inférieur à windows10, il faut également installer le Windows Serial driver :

https://os.mbed.com/docs/mbed-os/v5.7/tutorials/windows-serial-driver.html)

Dépose le microprogramme sur la carte.

Ouvre la console pour afficher les mesures (console)

Permet de vérifier les erreurs de codage (débuggage).



1) Choisir "**mode**" puis "**BBC micro:bit**".
2) Faire "**nouveau**", puis "**enregistrer**" pour sauvegarder.
3) Taper le code.
4) Faire "**vérifier**" et suivre les conseils données en cas d'erreurs ou de problèmes de mises en forme du code.
5) Déposer le microprogramme sur la carte : "**flasher**"

Le programme démarre, faire "**REPL**" pour afficher la console si besoin. Dans ce cas il faut appuyer sur le bouton RESET de la carte pour relancer le programme et l'affichage.

Remarque : Pour chaque nouvelle modification du programme, il faut fermer la console « REPL », et « flasher » de nouveau pour déposer le programme modifié sur la carte.

1. 1er programme

- Toujours commencer un programme par la ligne ***from microbit import ****
- Ecrire le programme suivant **(Attention à respecter les minuscules/majuscules et les espaces)**

```
from microbit import *

display.scroll("SNT")
```

- ***Flasher le programme dans la carte et choisir la microbit***

Que se passe-t-il ?

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.

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("IL ETAIT UNE FOIS...", delay=20)
```

Modifier le programme pour ralentir le défilement.

2. Boucle while

https://www.youtube.com/watch?v=aAdF7crqKnk

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 milliseconds choisi entre parenthèses.

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

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

Dans le programme suivant, on importe le module "`random`" de MicroPython et on l'utilise pour afficher unp ixel 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.

Ecrire un programme ci-dessous qui allume successivement et indéfiniment des pixels au hasard à l'écran.

3. Boucle for

https://www.youtube.com/watch?v=x4cVoamH6EE

On parle de boucle bornée lorsqu'on connaît le nombre de répétitions à effectuer. On utilise alors un compteur k qui va varier entre deux valeurs.

Le programme suivant utilise une boucle `for` pour faire défiler un pixel sur une ligne:

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

S'inspirer du programme précédent pour réaliser un programme qui fait défiler un pixel sur tout l'écran.

```
from microbit import *
while True:
  for i in range(5):
    for j in range(5):
```

dsgsdfg
fgs
sdfgdfs
ggfds
gfdfdhgs
gfds
fgd
fgdsgfd
dfsg
fdg

4. Les entrées boutons A, B et A+B – programmation événementielle

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 un code 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.
- `button_a.get_presses()` : renvoie le nombre de fois où le bouton a été appuyé depuis le démarrage ou la dernière fois que la méthode a été appelée et réinitialise ce total à zéro.

Le programme suivant fait défiler le texte "SNT" indéfiniment. On introduit l'**instruction conditionnelle** `if` qui va tester si le bouton A 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
```

5. Instructions conditionnelles `if` , `elif` , `else`

https://www.youtube.com/watch?v=5K2lwktygt4

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

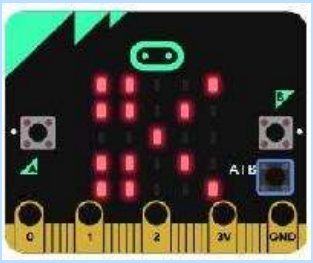
```
if quelque chose est vrai (``True``):
   # fais un truc
elif autre chose est vrai (``True``):
   # fais un autre truc
else:
   # fais encore autre chose.
```

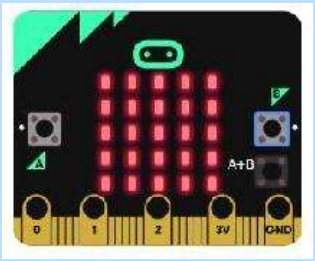
5.1 Pierre feuille ciseaux!

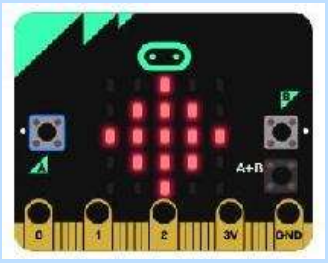
Compléter un programme suivant lequel une pression sur les boutons A et B affichera une image de ciseaux.

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.

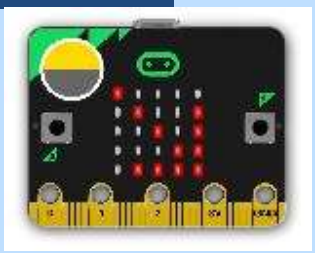







```
from microbit import *
#pierre = Image("00900:"
# "09990:"
# "99999:"
# "09990:"
# "00900:")
#feuille = Image("99999:"
# "99999:"
# "99999:"
# "99999:"
# "99999:")
ciseaux = Image("99009:"
"99090:"
"00900:"
"99090:"
"99009:")
while True:
```

5.2 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.

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

```
from microbit import *
soleil = Image("90909:"
"09990:"
"99999:"
"09990:"
"90909:")
lune = Image("00999:"
"09990:"
"09900:"
"09990:"
"00999:")
while True:
```