PROGRAMMATION DU BBC MICRO:BIT (PARTIE N°1)

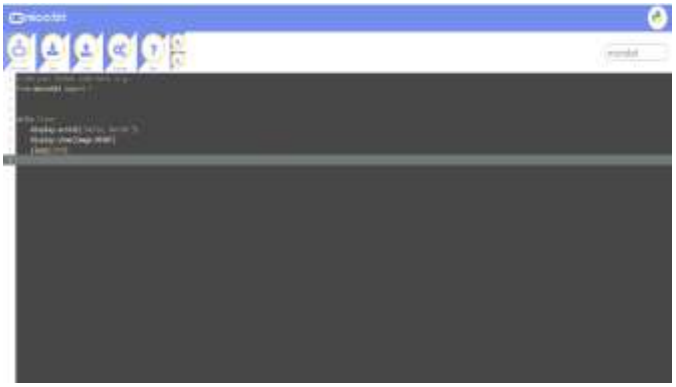
I – L'objectif.

Prendre en main la carte BBC Micro:Bit en utilisant deux types de logiciel de programmation :

- La programmation par bloc graphique
- La programmation ave le langage Python

II – Préparation du poste de travail.

Vous allez ouvrir deux fenêtres depuis votre navigateur WEB en utilisant les adresses suivantes :

Fenêtre n°1 : Editeur pour programmation blocs	Fenêtre n°2 : Editeur pour programmation Python
Adresse : https://makecode.microbit.org/#	Adresse : http://python.microbit.org/v/1
	

Vous basculerez d'une fenêtre à l'autre en utilisant la combinaison de touche « ALT »+ « ⇥ ».

III – Travail avec l'outil de programmation blocs.

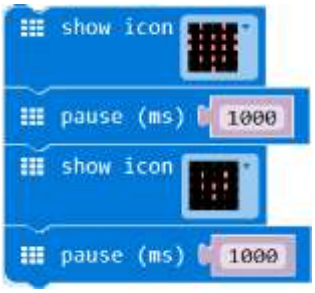
3.1 – Exercice n°1.

Nous allons essayer de faire « battre le cœur » de la carte BBC Micro:Bit, pour cela se positionner dans la fenêtre de programmation blocs puis choisir le bloc suivant :

Comme on peut le comprendre en regardant l'instruction, l'exécution de ce petit code se fera à la mise en marche de la carte BBC Micro:Bit, où après un appui sur le bouton « Reset ».

Mais attention les instructions qui seront incluses dans le crochet de ce bloc ne s'exécuteront qu'une seule fois !!!!!

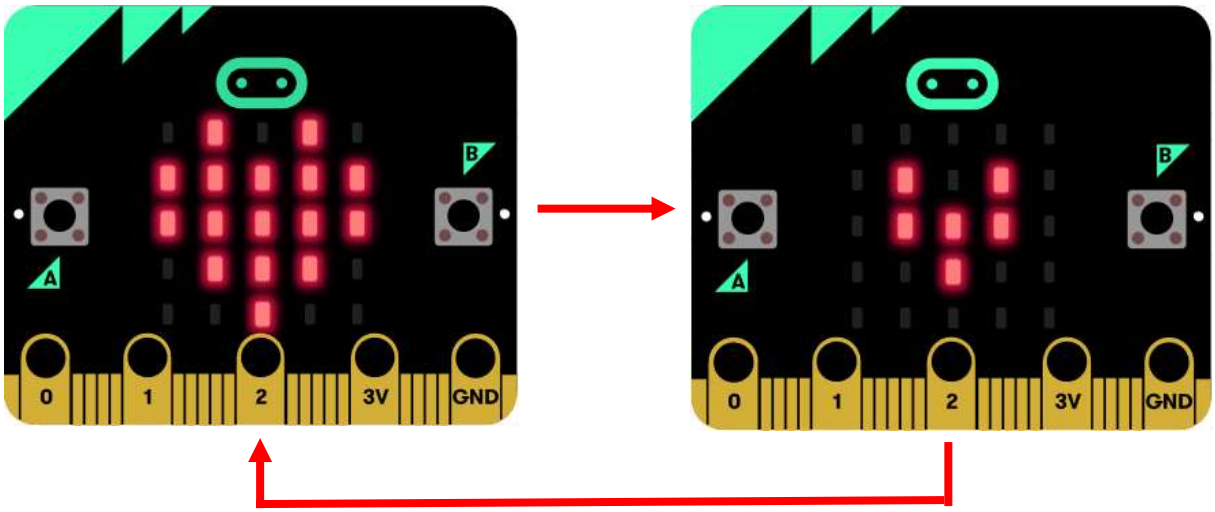
Ajouter dans le crochet du bloc « au démarrage » les blocs suivants :



Testez le fonctionnement à l'aide du simulateur, puis en réel en transférant le programme dans la cible en suivant la procédure décrite sur le document « Prise en main rapide de BBC Micro:Bit ».

3.2 – Exercice n°2.

Proposer une solution permettant de faire battre le cœur en permanence du BBC Micro:bit de façon permanente :



Pour réaliser cette modification du programme il vous faudra utiliser une structure répétitive, en effet le microprocesseur du BBC:MicroBit doit fonctionner en permanence pour exécuter les ordres que vous lui transmettez.

Testez le fonctionnement à l'aide du simulateur, puis en réel en transférant le programme dans la cible en suivant la procédure décrite sur le document « Prise en main rapide de BBC Micro:Bit ».

IV – Travail avec l'éditeur python.

4.1 - Exercice n°1.

Nous allons essayer de réaliser deux programmes similaires aux précédents mais cette fois à l'aide de ligne de codes en langage « Python ».

Nous en profiterons pour changer les dessins à afficher, nous utiliserons l'image « DIAMOND » et « DIAMOND_SMALL ».

Lancer pour cela l'éditeur de programmation en langage Python, puis saisir puis tester les lignes de codes suivantes :

```
# Add your Python code here. E.g.
from microbit import *                    // importer la totalité de la librairie « microbit »

display.show(Image.DIAMOND)
sleep(1000)
display.show(Image.DIAMOND_SMALL)
sleep(1000)
```

Transférer ce programme dans la « BBC Micro:Bit » afin de vérifier son fonctionnement.

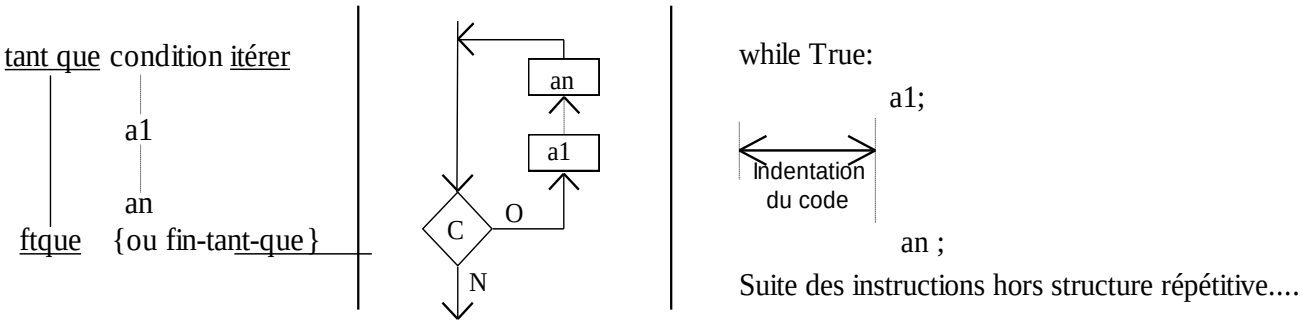
4.2 – Exercice n°2.

On souhaite à présent, modifier le code afin d'obtenir un cycle de fonctionnement ou successivement un grand diamant, puis un petit diamant, puis à nouveau un grand diamant… apparaît toutes les secondes.

Pour vous aider dans votre recherche, voici un exemple de structure répétitive et sa traduction en langage Python :

Type Boucle n°1 : Condition d'arrêt en tête d'itération ou boucle **while**

Syntaxe ***Sémantique*** ***Langage Python***



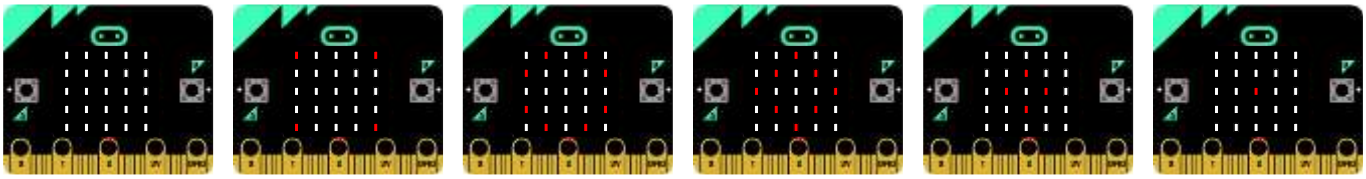
Remarque : Attention à bien penser à indenter (décaler vers la droite) le code qui sera compris dans la structure répétitive !!!!

Transférer ce programme dans la « BBC Micro:Bit » afin de vérifier son fonctionnement.

V – Travail de création personnelle.

5.1 – Exercice 1.

On souhaiterait maintenant réaliser une animation plus complète en 6 étapes :



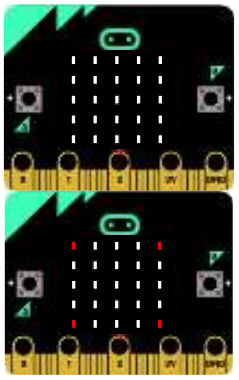
A l'aide de l'éditeur de programmation blocs, proposer une solution permettant d'obtenir une animation des diamants en 6 étapes.

Transférer ce programme dans la « BBC Micro:Bit » afin de vérifier son fonctionnement.

5.2 – Exercice 2.

On souhaite pouvoir réaliser la même animation à l'aide de l'éditeur Python, pour cela vous allez devoir réaliser vous-même les six dessins correspondant à l'aide de l'instruction suivante :

```
diamond1=Image("00000:"
               "00000:"
               "00000:"
               "00000:"
               "00000")
diamond2=Image("90009:"
               "00000:"
               "00000:"
               "00000:"
               "90009")
               ...
```



…

Puis appeler l'affichage et le maintenir 500ms à l'aide de l'instruction :

```
display.show(diamond1)
sleep(500)
…
```

Compléter cette séquence afin d'obtenir l'animation en boucle.

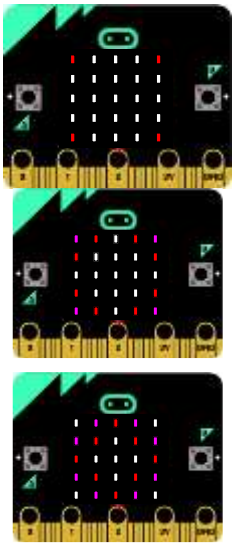
Puis transférer ce programme dans la « BBC Micro:Bit » afin de vérifier son fonctionnement.

5.2 – Exercice 3.

Un des avantages de la programmation en Python est de pouvoir aller bien plus loin dans les effets visuels, il est possible de moduler l'intensité lumineuse des leds en jouant sur la valeur numérique, celle-ci pouvant varier de 0 à 9.

```
                …
diamond2=Image("90009:"
               "00000:"
               "00000:"
               "00000:"
               "90009")
diamond3=Image("59095:"
               "90009:"
               "00000:"
               "90009:"
               "59095")
diamond4=Image("05950:"
               "59095:"
               "90009:"
               "59095:"
               "05950")
               …
```

…



…

Modifier votre programme, en particulier les dessins des images de façon à donner l'impression d'une trace dans l'animation en jouant sur le niveau d'intensité.

Transférer ce programme dans la « BBC Micro:Bit » afin de vérifier son fonctionnement.

5.4 – Exercice 4.

Il est également possible avec la méthode display.show() d'avoir accès à suites d'animations déjà préprogrammées, en particulier ALL_CLOCKS et ALL_ARROWS.

En utilisant le document sur le « Jeu d'instructions du BBC Micro:Bit » Proposer une programmation permettant de mettre en œuvre l'animation d'une horloge, puis l'animation des flèches.

Transférer les programmes dans la « BBC Micro:Bit » afin de vérifier leurs fonctionnements.