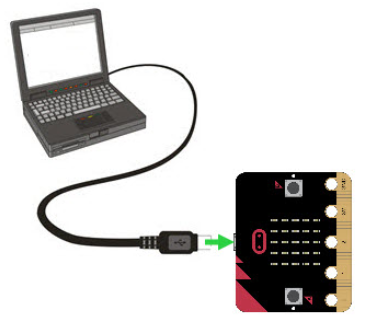
Hello World !

Le**premier programme**que réalise tout apprenti informaticien est le fameux "[**Hello World !**](https://fr.wikipedia.org/wiki/Hello_world)"... cela permet avec un programme minimal de prendre en main un langage et de s'assurer du bon fonctionnement de l'environnement de développement choisi (IDE).

**1.** Au moyen du câble fourni, **raccorder**la carte BBC micro:bit sur un port USB de l'ordinateur. Le câble USB assure la communication et l’alimentation.

Remarque : Si vous utilisez un câble magnétique, attention au sens du connecteur sur l'embout de la carte Raspberry, la **LED verte** du câble doit être dirigée **vers le haut**

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement**2. Lancer** le logiciel Thonny

**3.Choisir l’interpréteur avec** le menu Exécuter  Une image contenant texte

Description générée automatiquement    puis choisir **MicroPython BBC micro:bit**

**3.** La console Python (interpréteur interactif avec coloration du code entré, des sorties et des messages d'erreurs) indique que vous communiquez bien avec la carte.

Une image contenant texte

Description générée automatiquement

**4.** **Tapez** le code suivant dans l'éditeur du programme. Bien respecter les indentations (touche tabulation).  
Une image contenant texte

Description générée automatiquement

**5.** **Exécuter** le programme sur la micro:bit.  

Analysez ce code. Que fait-i l ?

**6. Modifiez** le programme pour réaliser un **badge lumineux** avec l'algorithme suivant :

* + Affichage de votre prénom
  + Attente 2s
  + Affichage de votre Nom
  + Attente 2s
  + Ensuite on recommence.

**Recopier** ce programme ici.

Pour la suite, **recopiez** uniquement les programmes que vous avez modifiés.

Matrice de LED : afficher des messages

**Exercice 1 : rôle des fonctions show et scroll avec la console (= sans utiliser un programme)**

Il est possible **d'envoyer les instructions une par une** dans la **console** au lieu d'exécuter un **programme**. **Arrêter** l’exécution du programme  avec 

Dans la console :

* **taper** from microbit import \*, **valider** avec entrée ;
* **taper** display.show("Hello World!"), **valider** avec entrée ;
* **observer** la matrice de LED;
* **taper** display.scroll('Hello World!'), valider avec entrée.
* **observer** la matrice de LED;

Quelle est la différence entre les deux fonctions ?

**Exercice 2 :**

**Copiez** puis **collez** le code suivant : (Ctrl+C puis Ctrl + V)

|  |
| --- |
| **from** microbit **import** **\***  **while** **True:**     display**.**show**(**"1"**)**     sleep**(**500**)**     display**.**show**(**" "**)**     sleep**(**500**)** |

**Analysez** ce code. Que fait-i l ?

Faîtes-le fonctionner sur la carte ou le simulateur en ligne.

**Aide  Structure répétitive : Répéter**

Une image contenant texte

Description générée automatiquementPour répéter à l'infini :

|  |  |
| --- | --- |
| **Tant que** vrai        Action 1        Action 2  **Fin tant que** |  |

**Exercice 3 :**

|  |
| --- |
| **from** microbit **import** **\*** display**.**clear**()**  **for** x **in** range**(**0**,** 5**):**     display**.**show**(str(**x)**)**     sleep**(**500**)** |

**La fonction range(0, 5**) donne une séquence d'entiers avec la variable x = 0 puis 1, 2, 3, 4.

**str(x)** convertit l'entier x en chaîne de caractères.

**Modifiez** le programme ci-dessus afin qu'il compte en boucle de 0 jusqu'à 9.

**Modifiez** le programme pour qu'il se répète après une pause de 4 s.

**Aide  Structure répétitive : Pour**

La boucle Pour (for en anglais) est une [structure de contrôle](https://fr.wikipedia.org/wiki/Structure_de_contr%C3%B4le) de programmation qui permet de répéter l'exécution d'une séquence d'instructions.

Afficher 0, 1, 2, 3 ,4  s'écrit :

|  |  |
| --- | --- |
| **Pour** i allant de 0 à 4 par pas de 1    Afficher i        Attendre 500 ms  **Fin pour** |  |

Matrice de LED : afficher des pixels

|  |  |
| --- | --- |
| **Exercice 1 :**  La fonction **set\_pixel** allume un point sur l'écran. Elle prend 3 arguments :   * + l'abscisse**x (colonne**) ;   + l'ordonnée **y (ligne) ;**   + le dernier argument est la **luminosité** du point entre **0 et 9**: 0 point éteint,  9 luminosité maximale. |  |

|  |
| --- |
| **from** microbit **import \*** display**.**clear**()**   **for** x **in** range**(**0**,**5**):**     display**.**set\_pixel**(**x**,**0**,**9**)**     sleep**(**500**)** |

Une image contenant texte

Description générée automatiquement**Analysez** ce code. Que fait-i l ?

**Faîtes**-le fonctionner sur la carte ou le simulateur en ligne.

**Modifiez** ce programme afin qu'il allume la colonne centrale. Remarque : vous pouvez relancer le programme avec le bouton poussoir Reset situé au dos de la carte.

**Exercice 2 : pixel/pixel**

|  |  |
| --- | --- |
|  | **Modifiez** ce programme afin qu'il allume successivement tous les pixels de l'écran, **ligne par ligne**.    Indication : On pourra utiliser 2 boucles for imbriquées l'une dans l'autre.  Pensez à changer le nom de la variable de la seconde boucle ! |

**Exercice 3 : colonne / colonne**

|  |  |
| --- | --- |
|  | **Modifiez** le programme précédent afin qu'il allume tous les pixels **colonne par colonne** en 5 étapes et en boucle.  Indication : On pourra utiliser display.show("  ") ou display.clear()  pour effacer la matrice. |

**Exercice 4 : le ciel étoilé**

|  |  |
| --- | --- |
|  | **Modifiez** ce programme :   * Afin d'obtenir un affichage avec des pixels dont la luminosité est aléatoire. * Qu'il se répète indéfiniment. |

**Indice :** pour obtenir un nombre aléatoire entre 0 et 9 compris :

Importez la fonction randint depuis la librairie random  avec :

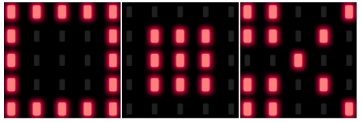
**from random import randint**

Utilisez **randint(0, 9)** pour choisir un nombre aléatoire entre 0 et 9 compris

ChiFouMi

**Règle du ChiFouMi**: Chaque joueur doit secouer sa carte 3 fois avant que la carte fasse un choix et l'affiche.

 Feuille            Pierre            Ciseaux

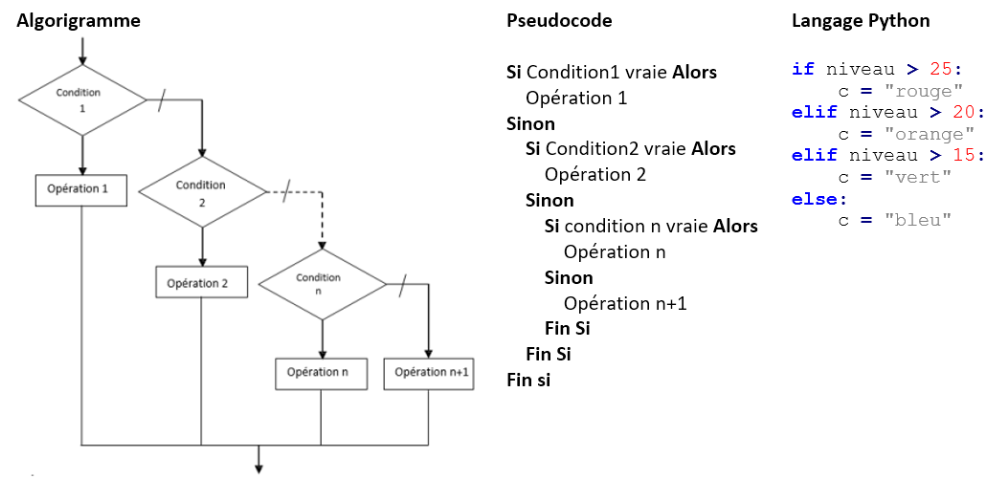


**Modifier** le programme pour respecter les règles.

|  |
| --- |
| **from** microbit **import** **\*** **from** random **import** randint  feuille **=** Image**(**"99999:90009:90009:90009:99999"**)** pierre **=**  ciseaux **=**  nbSecousses **=** 0  **while** **True:**     **if** accelerometer**.**was\_gesture**(**"shake"**):**         nbSecousses **=** nbSecousses **+** 1      **if** nbSecousses **>=**3**:**         nbSecousses **=** 0                 choix **=** randint**(**1**,**3**)**         **if** choix **==** 1**:**             display**.**show**(**pierre) |

**Aide**

**Structures alternatives à choix multiples**



**Structure alternative**

Une image contenant texte

Description générée automatiquement

**La structure alternative réduite**

Une image contenant texte

Description générée automatiquement

Dé 6 faces

Un algorithme de contrôle fréquent sur un système informatique embarqué consiste en une boucle infinie où s’enchaînent l'acquisition d’événements, le traitement puis une action sur le système.

|  |  |
| --- | --- |
|  | Initialiser les actionneurs à leur position de départ  **Tant que** vrai        Lire les informations des capteurs        Traiter ces informations        Transmettre ces informations aux actionneurs  **Fin tant que** |

**Utilisation des boutons :**

|  |  |
| --- | --- |
|  | 2 boutons sont présents sur la carte. Les fonctions associées sont : |

* + button\_a.is\_pressed renvoie True si le bouton A est effectivement appuyé ;
  + button\_a.was\_pressed() renvoie True si le bouton A a été appuyé ;
  + button\_a.get\_pressed() renvoie le nombre de fois que le bouton A a été appuyé ;
  + Remplacer a par b pour l'autre BP.

Exemple de code utilisant un bouton poussoir :

|  |
| --- |
| **from** microbit **import** **\***  **while** **True:**     **if** button\_a**.**is\_pressed**():**         display**.**show**(**Image**.**HAPPY**)**     **else:**         display**.**show**(**Image**.**SAD**)** |

**Réaliser**un programme qui affiche pendant 2 s un chiffre choisi au hasard dans l'intervalle [0, 6] après l'appui sur le bouton A.

Bonus : simuler le roulage du dé, en faisant une petite animation avant d'afficher le nombre.

Podomètre

|  |  |
| --- | --- |
| Un podomètre est un dispositif portable sensible au mouvement permettant de mesurer en temps réel le nombre de pas d'une personne.    **Principe du programme de la carte Micro:bit** |  |

On initialise à zéro le nombre de pas.

On répète à l'infini :

Si un mouvement vers le haut est détecté :

Le nombre de pas est incrémenté ;

Si le nombre de pas est égal à 5, on le remet à zéro et on affiche l'image "DIAMOND" pendant 200ms ;

Sinon on efface l'écran. On allume tous les pixels de la ligne du milieu un par un toutes les 100 ms. Ensuite on affiche le nombre de pas

Malheureusement, à force de secouer la carte, le programme a été mis sens dessus-dessous !

**Remettez** le programme en ordre de marche ! Vous avez droit à 2 essais de Flashage.

|  |
| --- |
| **from** microbit **import** **\***                          nbPas **=** 0  display**.**set\_pixel**(**x**,** 2 **,**9**)**   display**.**show**(**nbPas**)**              sleep**(**200**)**        display**.**show**(**Image**.**DIAMOND**)**   **if** nbPas **==** 5**:**        **else:**                         display**.**show**(**" "**)**  **for** x **in** range**(**0**,** 5**):**                         sleep**(**100**)**               **if** accelerometer**.**was\_gesture**(**"up"**):**    nbPas **=** 0    nbPas **=** nbPas **+** 1 **while** **True:** |

**Aide**

|  |  |
| --- | --- |
| La carte Microbit possède un accéléromètre.  La fonction accelerometer.was\_gesture("up") détecte si un mouvement vers le haut a eu lieu.  D'autres gestes peuvent être reconnus : down, left, right, face up, face down, freefall, 3g, 6g, 8g, shake  Comme pour les boutons :   * + accelerometer.is\_gesture("up") pour détecter un geste en cours ;   + accelerometer.get\_gestures()  pour compter les gestes. |  |