

---

# Ressources pour la carte Micro:bit

***Version 1***

**IREM Marseille**

**juin 19, 2018**



---

## Prise en main

---

<b>1</b>	<b>Prise en main de Micro :bit</b>	<b>3</b>
<b>2</b>	<b>Projets à réaliser</b>	<b>33</b>
<b>3</b>	<b>Index et page de recherche</b>	<b>47</b>
<b>Index</b>		<b>49</b>



par le groupe InEFLP de l'IREM de Marseille

### Contenu du site

Vous trouverez dans ce document des ressources permettant de se former à Micro :bit.

### Qui sommes-nous ?

Nous sommes des enseignants de maths/sciences regroupés au sein d'un groupe de recherche de l'IREM de Marseille.

Notre groupe, *Innovation, Expérimentation et Formation en Lycée Professionnel* (InEFLP) a une partie de son travail consacrée l'enseignement de l'algorithme en classes de lycée professionnel. Dans le cadre de cette recherche, nous explorons les objets connectés tels que Arduino ou Microbit.

Site du groupe InEFLP.



Table des matières du document



# CHAPITRE 1

---

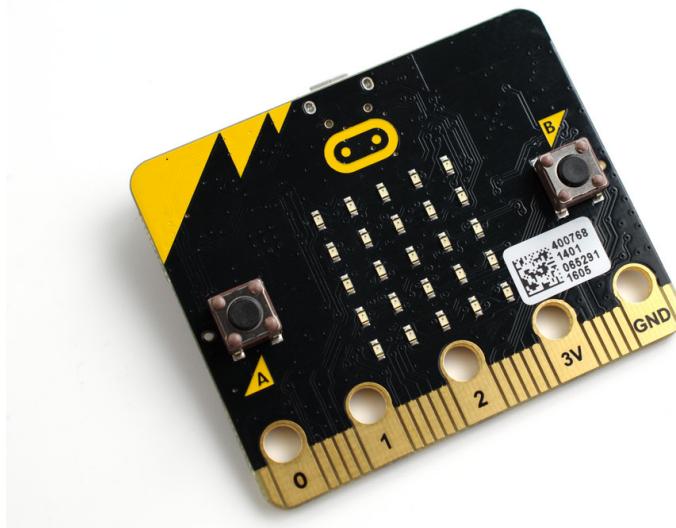
## Prise en main de Micro :bit

---

### 1.1 Micro :bit, c'est quoi ?

Micro :bit est un microcontrôleur développé au Royaume-Unis. Par ses caractéristiques techniques et ses interfaces pédagogiques, cet objet possède un fort potentiel pour l'enseignement de l'algorithme.

Après un bref rappel historique, nous expliquerons plus en détail les caractéristiques propres de cet objet. Nous mettrons ensuite en avant la facilité de mise en œuvre en formation puis nous poursuivrons en donnant un premier aperçu de l'intérêt pédagogique de Micro :bit.



#### 1.1.1 Bref historique

Le développement de Micro :bit s'inscrit dans le cadre d'une politique volontariste de développement de l'apprentissage de la programmation. L'objectif premier visait à équiper tous les élèves de 11/12 ans du Royaume-Unis ainsi que leur

enseignant. Maintenant que c'est chose faite, le reste du monde peut en profiter aussi.

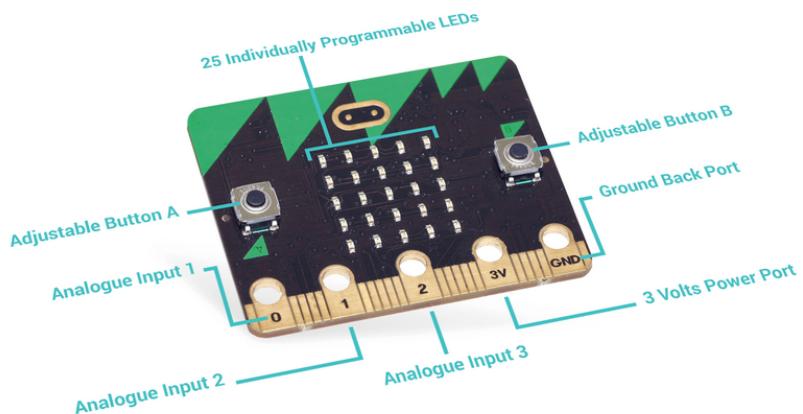


La BBC<sup>1</sup> est le moteur de ce projet. 30 ans après sa première distribution d'ordinateurs aux enfants britanniques<sup>2</sup>, “la Vieille Dame” remet ça aujourd’hui. La BBC utilise ses moyens de diffusions pour promouvoir et accompagner les utilisateurs, notamment en proposant des émissions de TV dédié à cet objet sur un mode ludique et divertissant. Sur les 29<sup>3</sup> partenaires de ce projet, se trouvent entre autres Microsoft<sup>4</sup> pour une partie logiciel et interface de programmation, ARM<sup>5</sup> pour la construction des processeur et la partie matériel, et Samsung<sup>6</sup> pour un support mobile. C'est donc un projet qui mobilise des acteurs majeurs du numériques et de la communication, prévu pour durer.

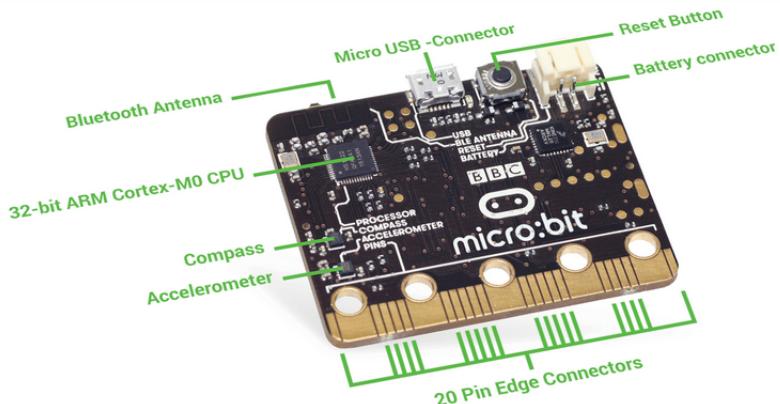
1. Make It Digital - The BBC micro :bit. (s. d.). Consulté 29 mars 2017, à l'adresse <http://www.bbc.co.uk/programmes/articles/4hVG2Br1W1LKcmw8nSm9WnQ/the-bbc-micro-bit>
2. BBC Micro. (2016, septembre 20). In Wikipédia. Consulté à l'adresse [https://fr.wikipedia.org/w/index.php?title=BBC\\_Micro&oldid=129763631](https://fr.wikipedia.org/w/index.php?title=BBC_Micro&oldid=129763631)
3. Partners. (s. d.). Consulté 29 mars 2017, à l'adresse <https://www.microbit.co.uk/partners>
4. The BBC micro :bit and Microsoft - Microsoft Research. (s. d.). Consulté 29 mars 2017, à l'adresse <https://www.microsoft.com/en-us/research/project/the-bbc-microbit-and-microsoft/>
5. Ltd, A. R. M. (s. d.). ARM | Innovation Hub - BBC micro :bit. Consulté 29 mars 2017, à l'adresse <http://www.arm.com/innovation/products/microbit.php>
6. Code on the go with Samsung & micro :bit. (s. d.). Consulté 29 mars 2017, à l'adresse <http://www.samsung.com/uk/citizenship/bbcmicrobit.html>



### 1.1.2 La carte Micro :bit



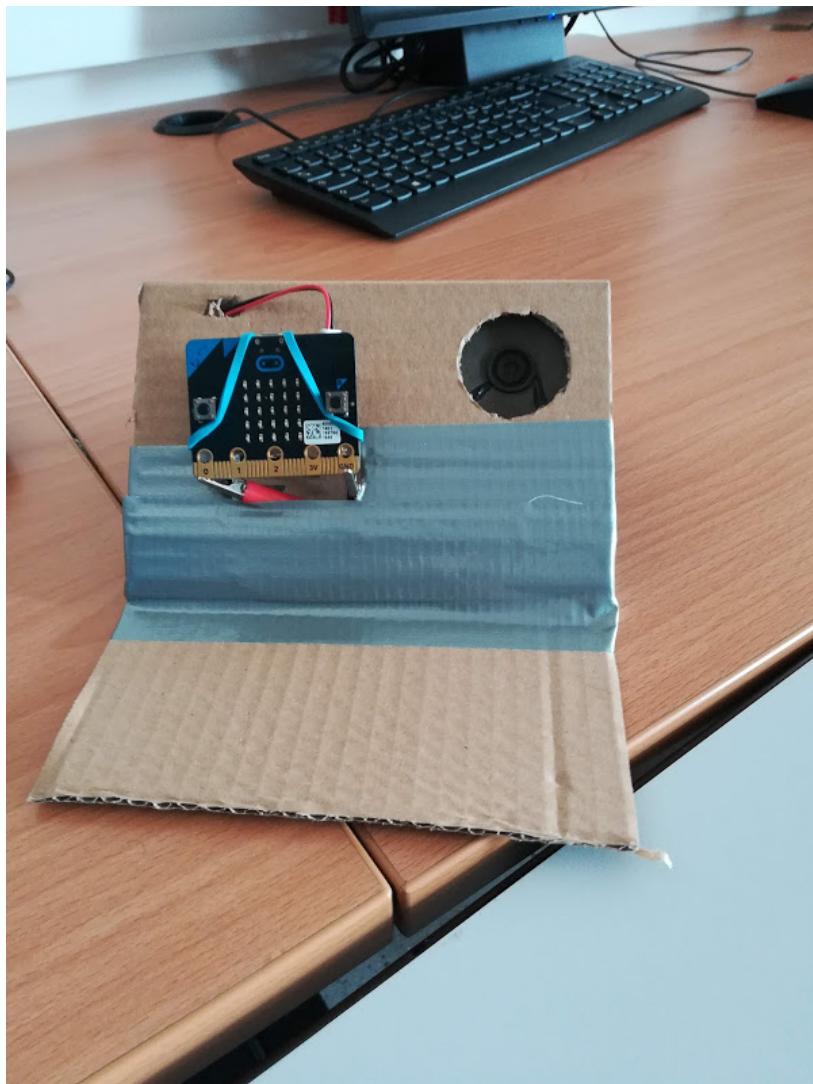
Concrètement de quoi s'agit-il ? On parle ici de microcontrôleur, à savoir une carte électronique programmable pour interagir avec le monde réel. C'est une version accessible de l'électronique que tout un chacun manipule au quotidien sans se poser de question, par exemple les dispositifs de domotique qui permettent de gérer à distance le chauffage, la sécurité, l'arrosage du géranium... Ou bien plus simplement la bouilloire programmable au degré °C près, la guirlande du sapin qui clignote au rythme de "Jingle Bells". Ce microcontrôleur permet d'élaborer par exemple un podomètre, un doudou sensoriel, un sismographe rudimentaire...



L'interface de programmation est conçue pour être utilisable par un enfant d'une dizaine d'année, c'est donc la simplicité qui prime. On dispose en première approche d'une application internet utilisant le principe de la programmation par bloc, à savoir sur le principe des Blockly que l'on retrouve dans Scratch ou StudioCode. En plus d'une programmation accessible, l'interface propose une simulation de la carte. Ceci permet de voir directement les effets du programme dans l'interface. Pour un usage plus avancé il est notamment possible de programmer avec le langage Python<sup>7</sup> ou Javascript.

---

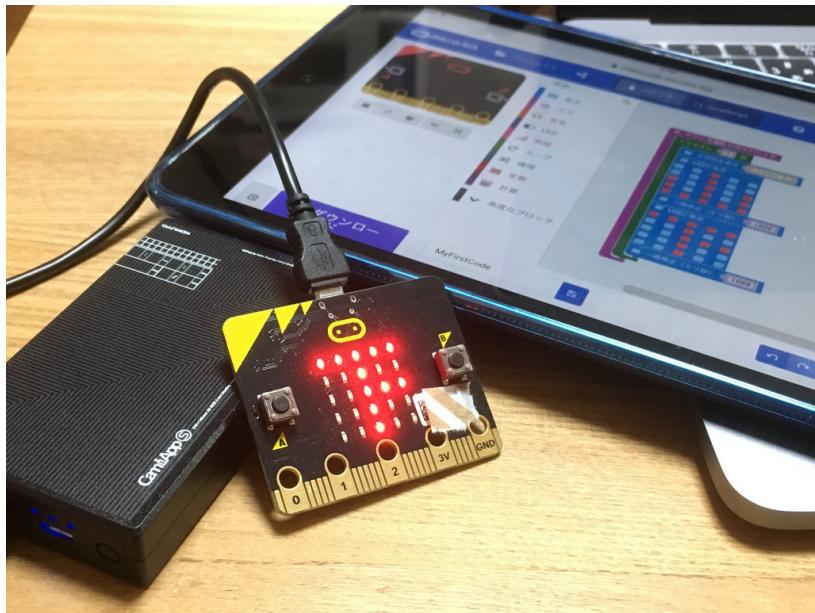
7. Python editor. (s. d.). Consulté 29 mars 2017, à l'adresse <http://python.microbit.org/editor.html>



Bien entendu de nombreux exemples de projets existent, qu'ils soient issus des émissions BBC ou de la communauté éducative. Sur le site officiel on trouve des idées, des tutoriels, des leçons<sup>8</sup> comme par exemple : une alarme de trousse, un compteur de frappe (au baseball) ou encore des leçons sur l'accélération.

---

8. Idées | micro :bit. (s. d.). Consulté 29 mars 2017, à l'adresse <http://microbit.org/fr/ideas/>



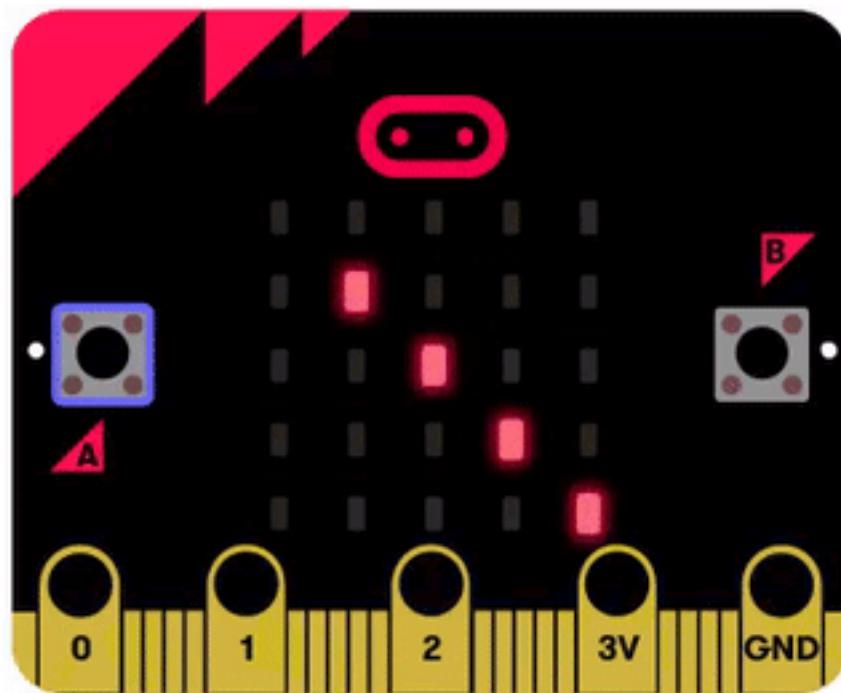
## Notes

### 1.2 Pile ou face (blocs)

#### 1.2.1 Description

Le but de ce projet de simuler une expérience aléatoire de lancer de pièce. A partir d'une situation simple idéale pour une prise en main de l'interface de programmation, il s'agit par la suite d'améliorer le programme pas à pas. L'objectif est d'obtenir un programme utilisable dans le cadre de d'un cours sur les statistiques et les probabilités.

On utilise l'interface de programmation par bloc : <https://makecode.microbit.org/>



#### Exemple(s) d'utilisation

- Algorithmique et programmation (thème E) du programme de cycle 4
- Domaine statistiques et probabilités du programme de mathématiques de Bac Pro
- Accompagnement personnalisé pour des élèves de seconde Bac Pro

### 1.2.2 Progression

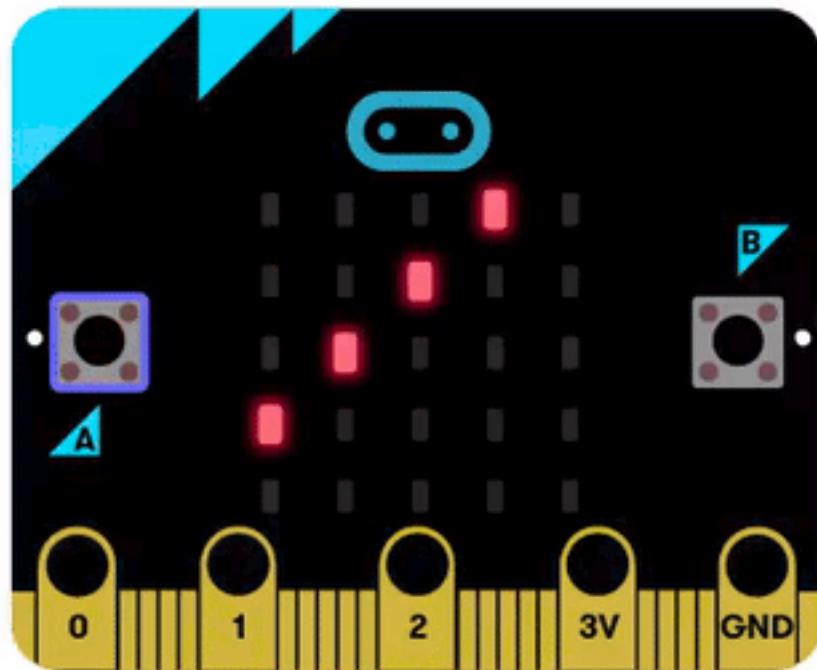
#### Niveau 1

Ce que l'on veut obtenir : afficher 0 ou 1 de façon aléatoire à l'issue d'une courte animation. Ce premier niveau permet de se familiariser avec l'interface tout en produisant un premier programme fonctionnel et utile.

#### Les notions abordées

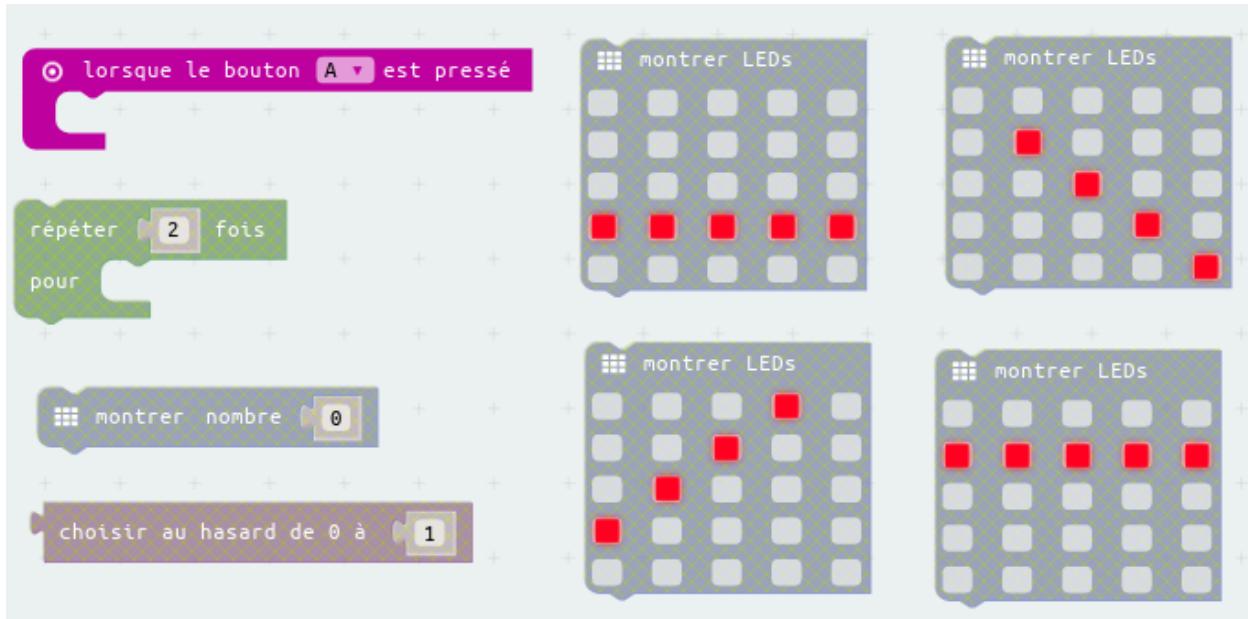
Dans ce niveau nous trouvons les notions suivantes :

- interactions avec l'utilisateur (bouton, affichage)
- boucle
- aléa



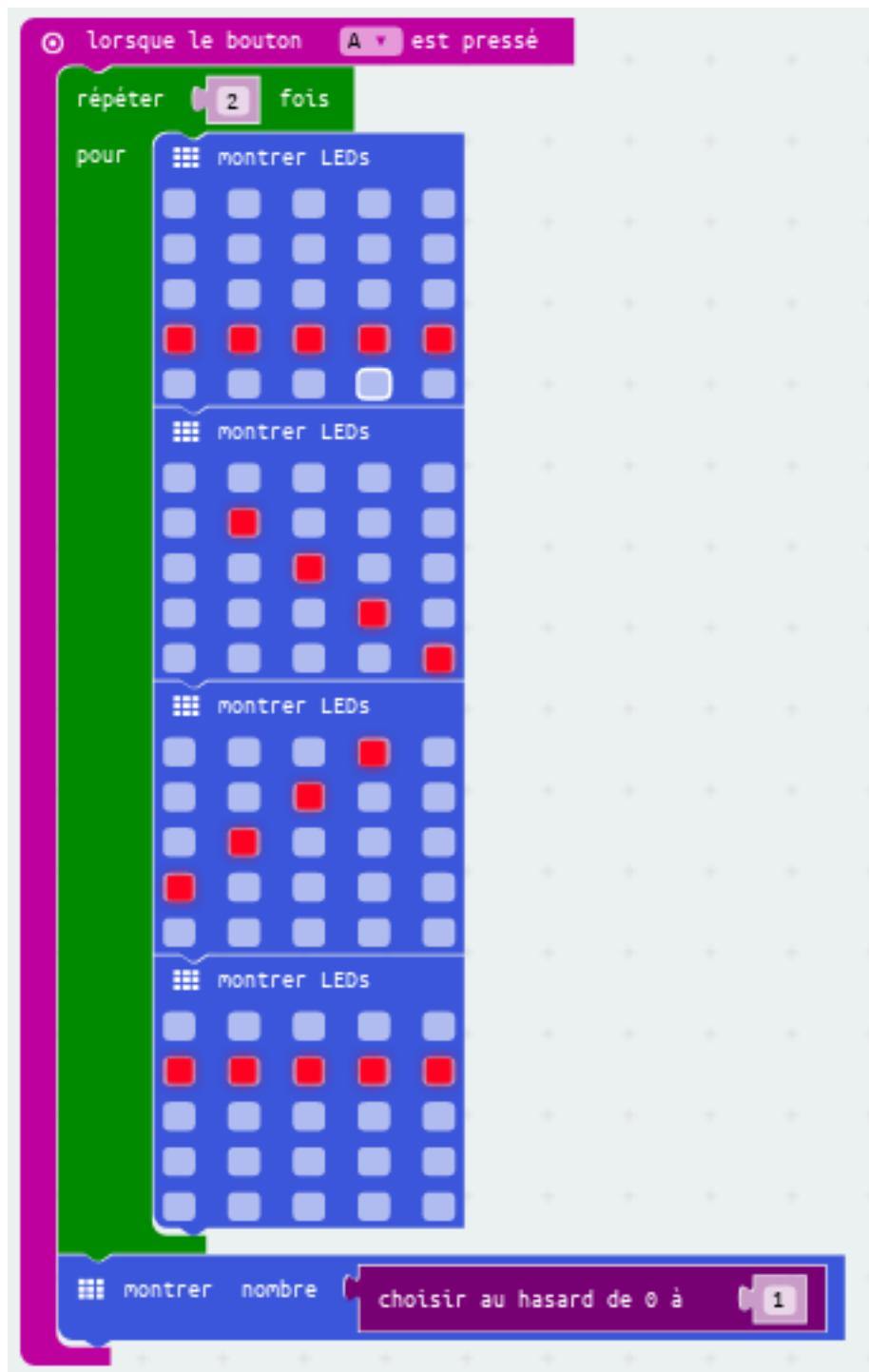
### Les blocs préconisés

On propose aux élèves d'utiliser les blocs suivant :



### Une solution possible

Le résultat escompté est le suivant :



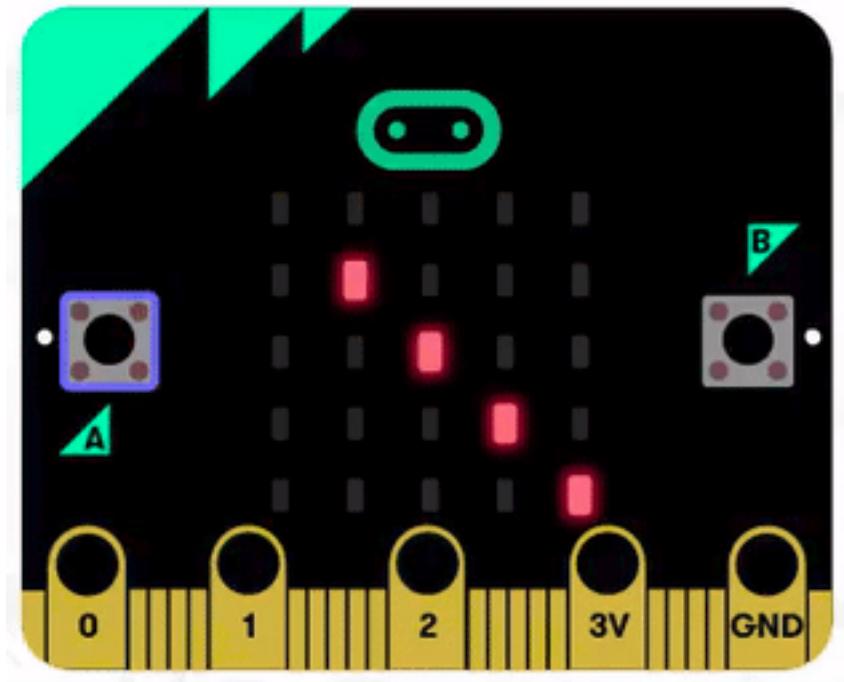
## Niveau 2

Ce que l'on veut obtenir : afficher P ou F de façon aléatoire à l'issue d'une courte animation. Pour ce deuxième niveau, on va utiliser deux nouveaux blocs seulement.

### Les notions abordées

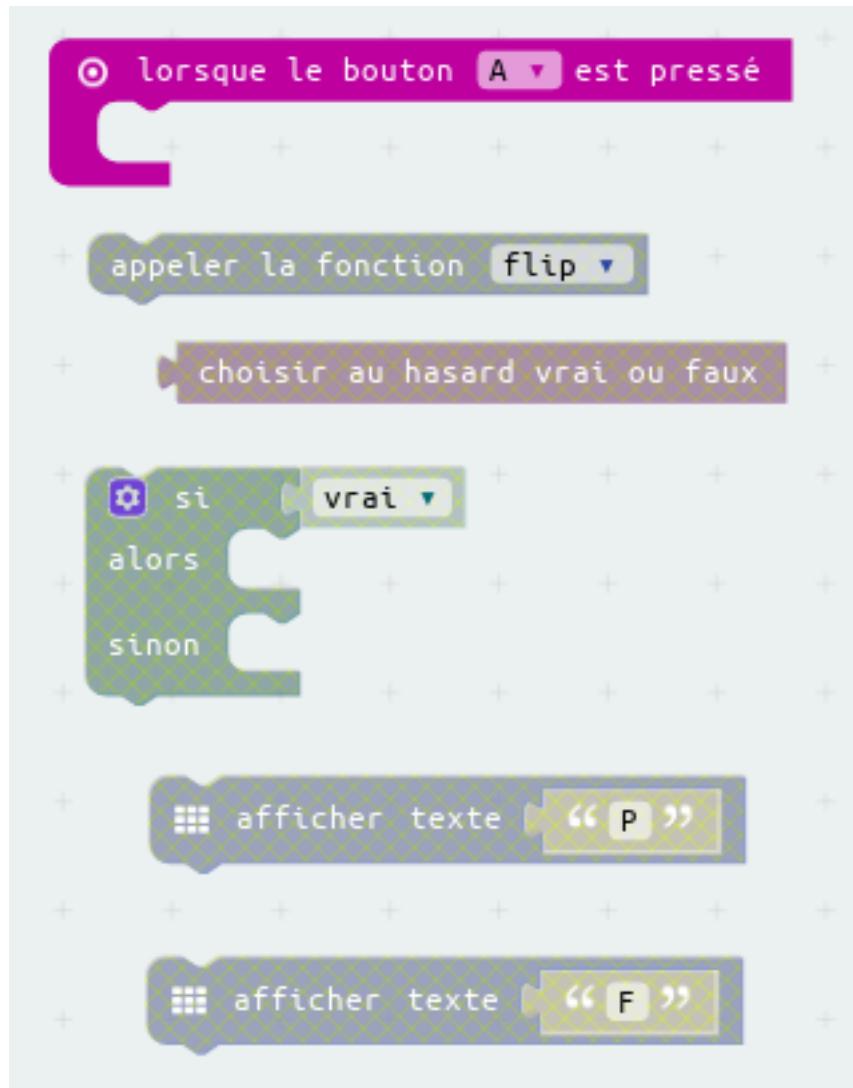
Dans ce niveau nous trouvons les notions suivantes :

- fonction (création et appel)
- instruction conditionnelle



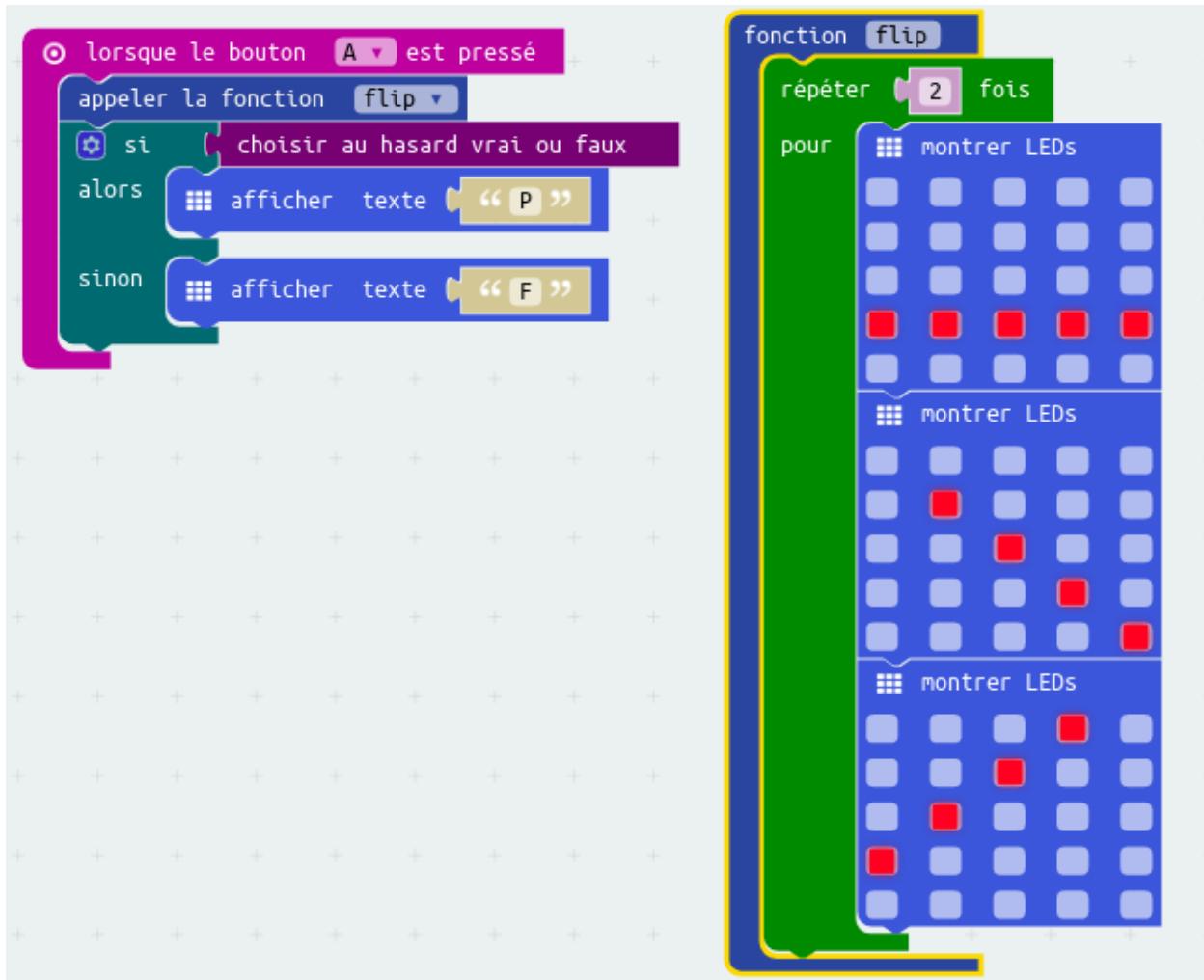
### Les blocs préconisés

On propose aux élèves d'utiliser les blocs suivant :



### Une solution possible

Le résultat escompté est le suivant :



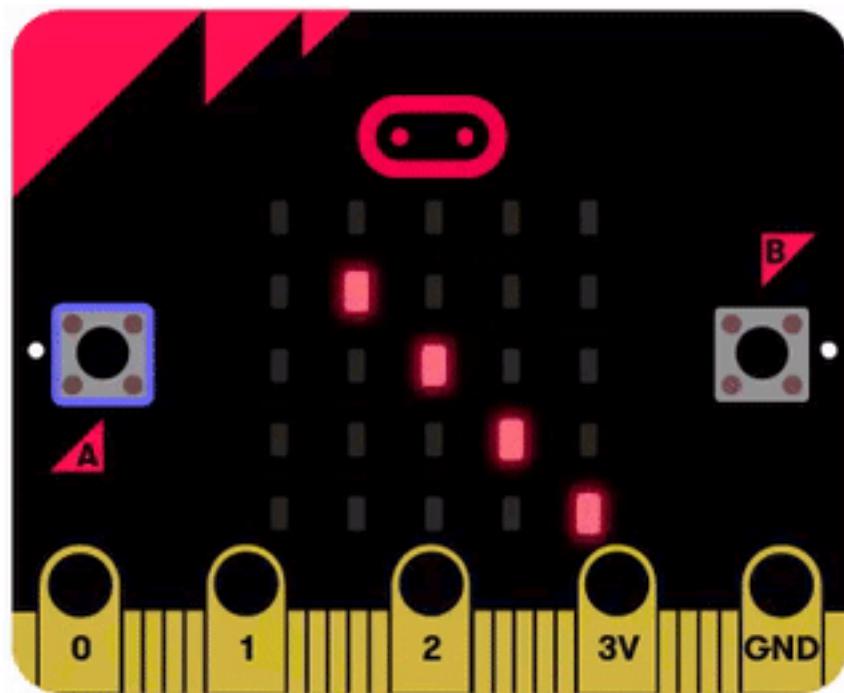
### Niveau 3

Ce que l'on veut obtenir : compter les issues obtenues et afficher le résultat. Pour parvenir à cela il va falloir utiliser une variable.

#### Les notions abordées

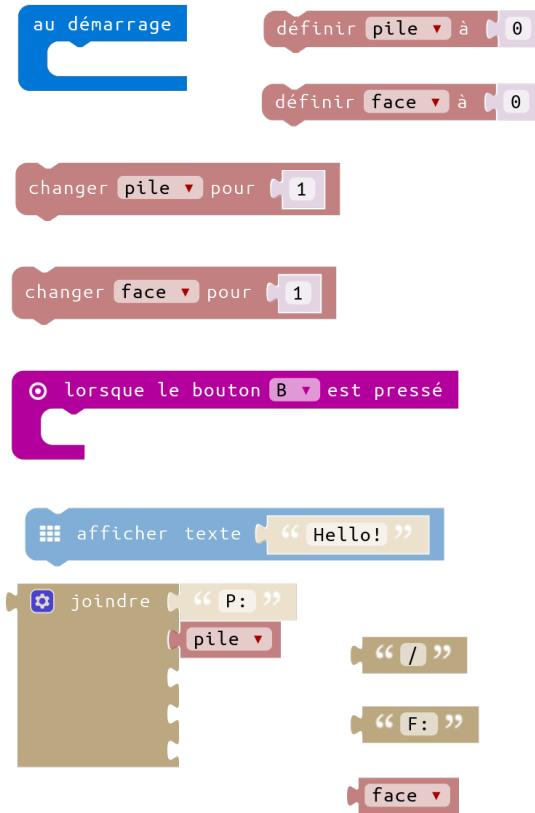
Dans ce niveau nous trouvons les notions suivantes :

- définition d'une variable
- incrémentation d'une variable
- concaténation de texte et de valeur



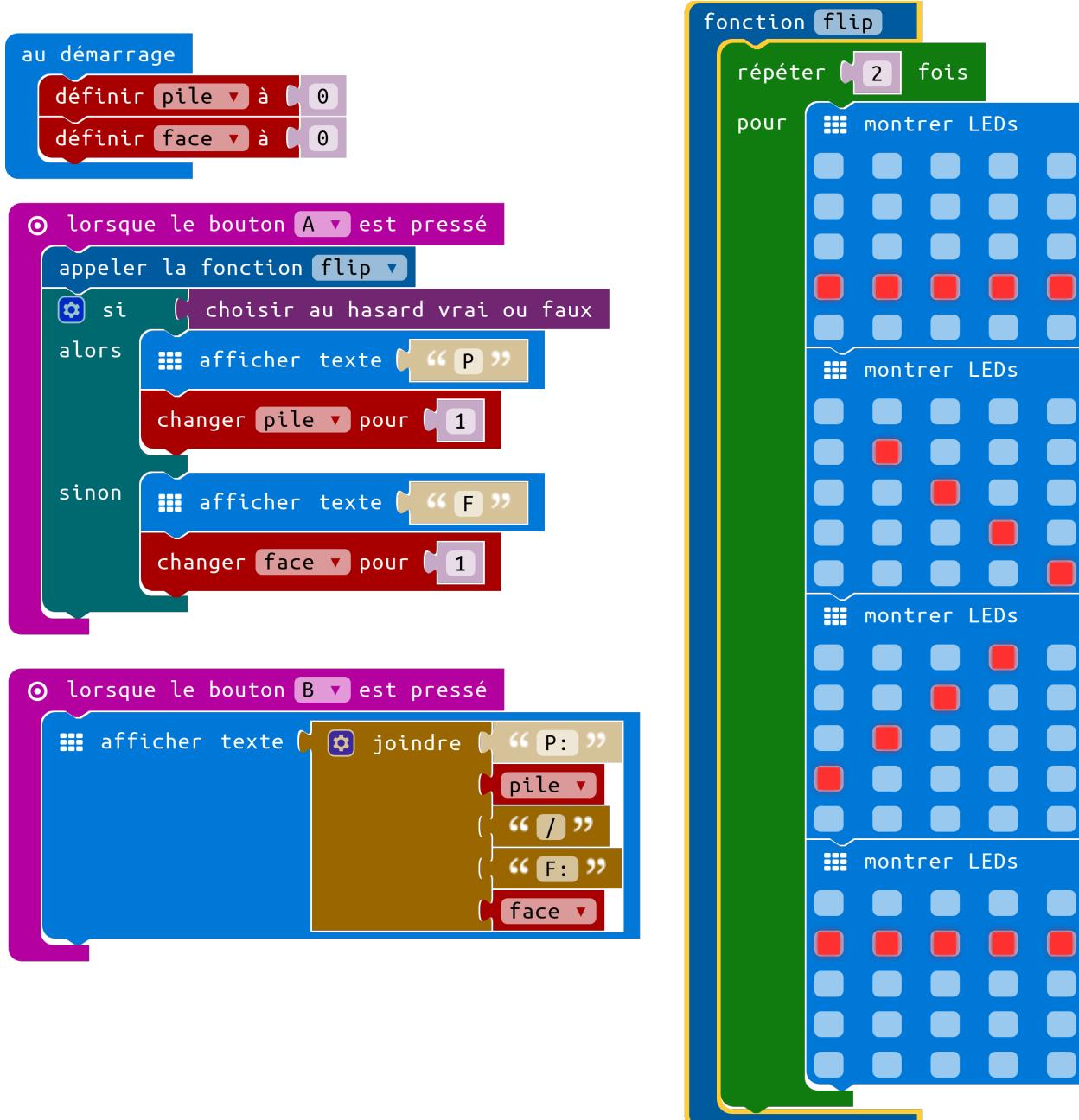
### Les blocs préconisés

On propose aux élèves d'utiliser les blocs suivant :



### Une solution possible

Le résultat escompté est le suivant :

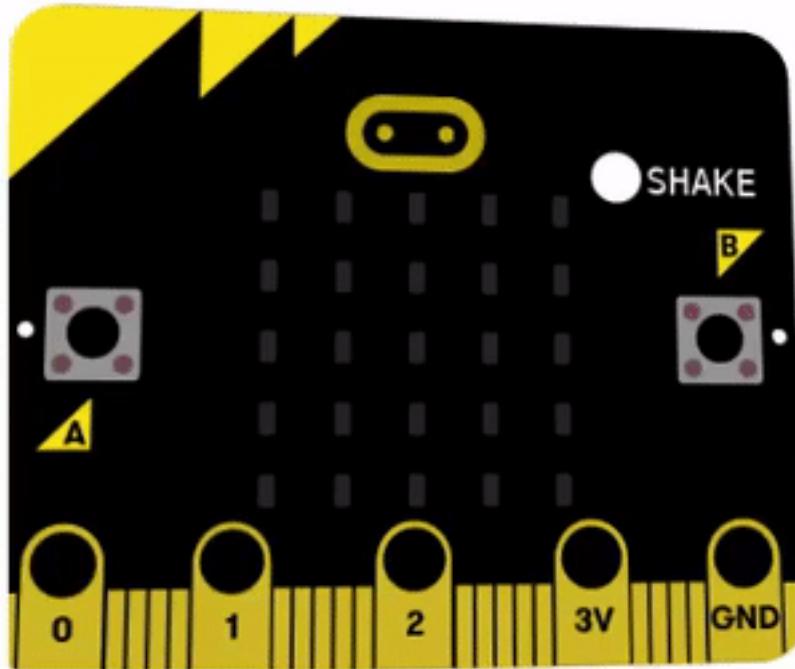


## 1.3 Dé 6 faces (blocs)

### 1.3.1 Description

Le but de ce projet de simuler une expérience aléatoire de lancer d'un dé à 6 faces.. Toujours à partir d'une situation simple idéale , le programme peut être étoffé au gré des besoins.

On utilise l'interface de programmation par bloc : <https://makecode.microbit.org/>



#### Exemple(s) d'utilisation

- Algorithmique et programmation (thème E) du programme de cycle 4
- Domaine statistique et probabilités du programme de mathématiques en seconde et première Bac Pro.
- Accompagnement personnalisé pour des élèves de seconde Bac Pro

#### 1.3.2 Progression

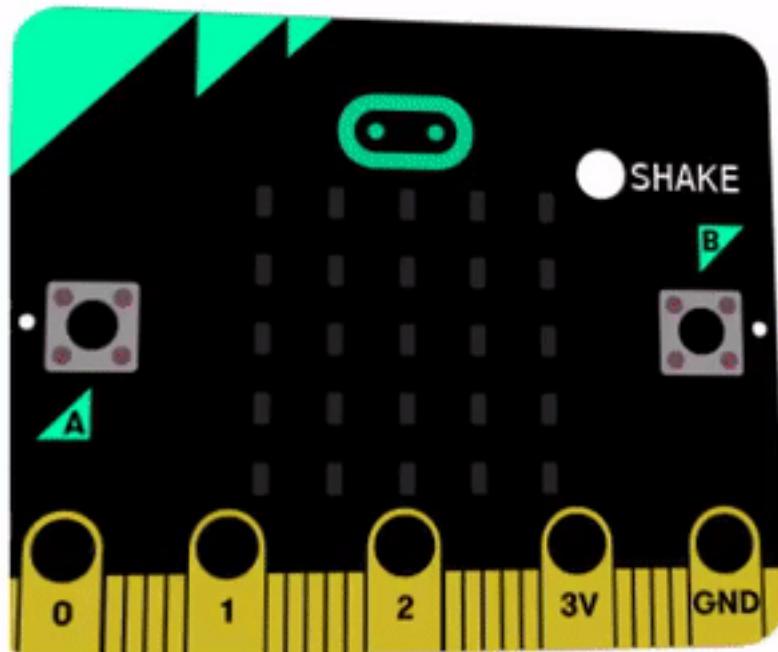
##### Niveau 1

Ce que l'on veut obtenir : afficher un nombre entier entre 1 et 6 de façon aléatoire en secouant l'appareil. Ce premier niveau permet d'introduire la problématique.

##### Les notions abordées

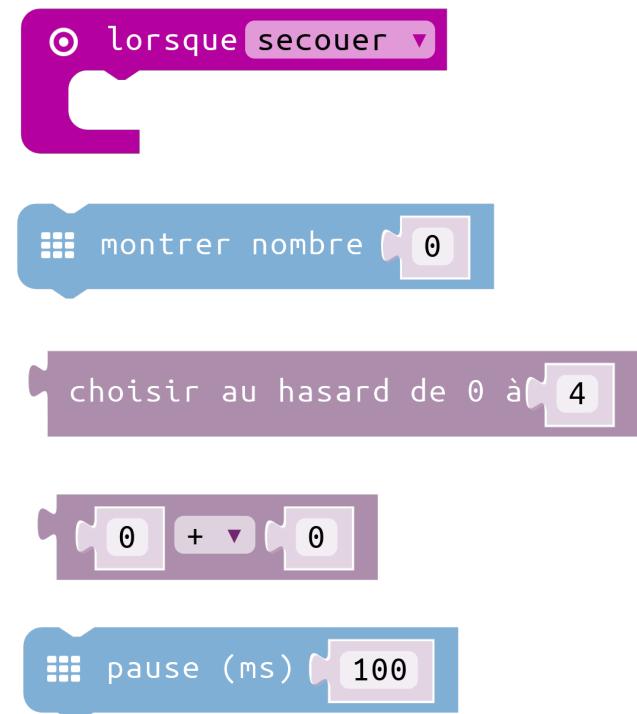
###### Dans ce niveau nous trouvons les notions suivantes :

- interactions avec l'utilisateur (bouton, affichage)
- boucle
- aléa



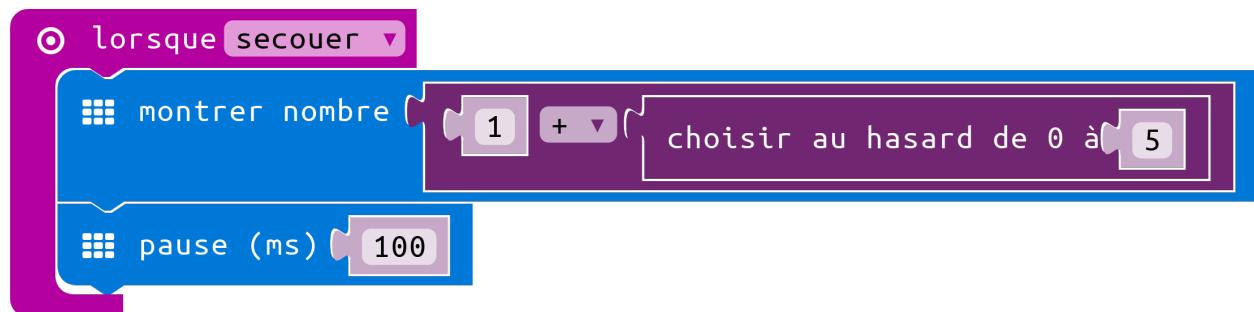
### Les blocs préconisés

On propose aux élèves d'utiliser les blocs suivant :



### Une solution possible

Le résultat escompté est le suivant :



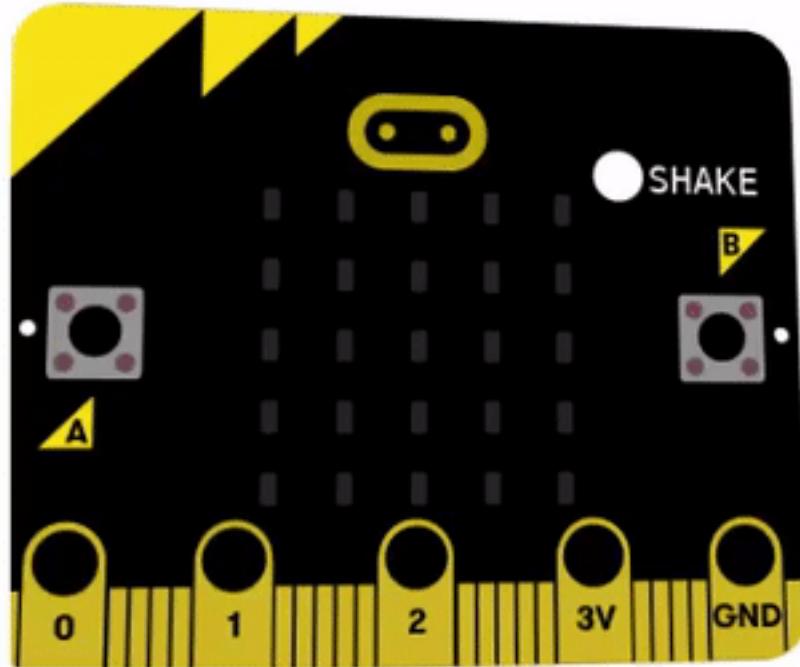
### Niveau 2

Ce que l'on veut obtenir : simuler l'affichage d'un dé à l'issue d'un tirage aléatoire. Ce deuxième niveau, va permettre d'introduire de nouvelle notions.

### Les notions abordées

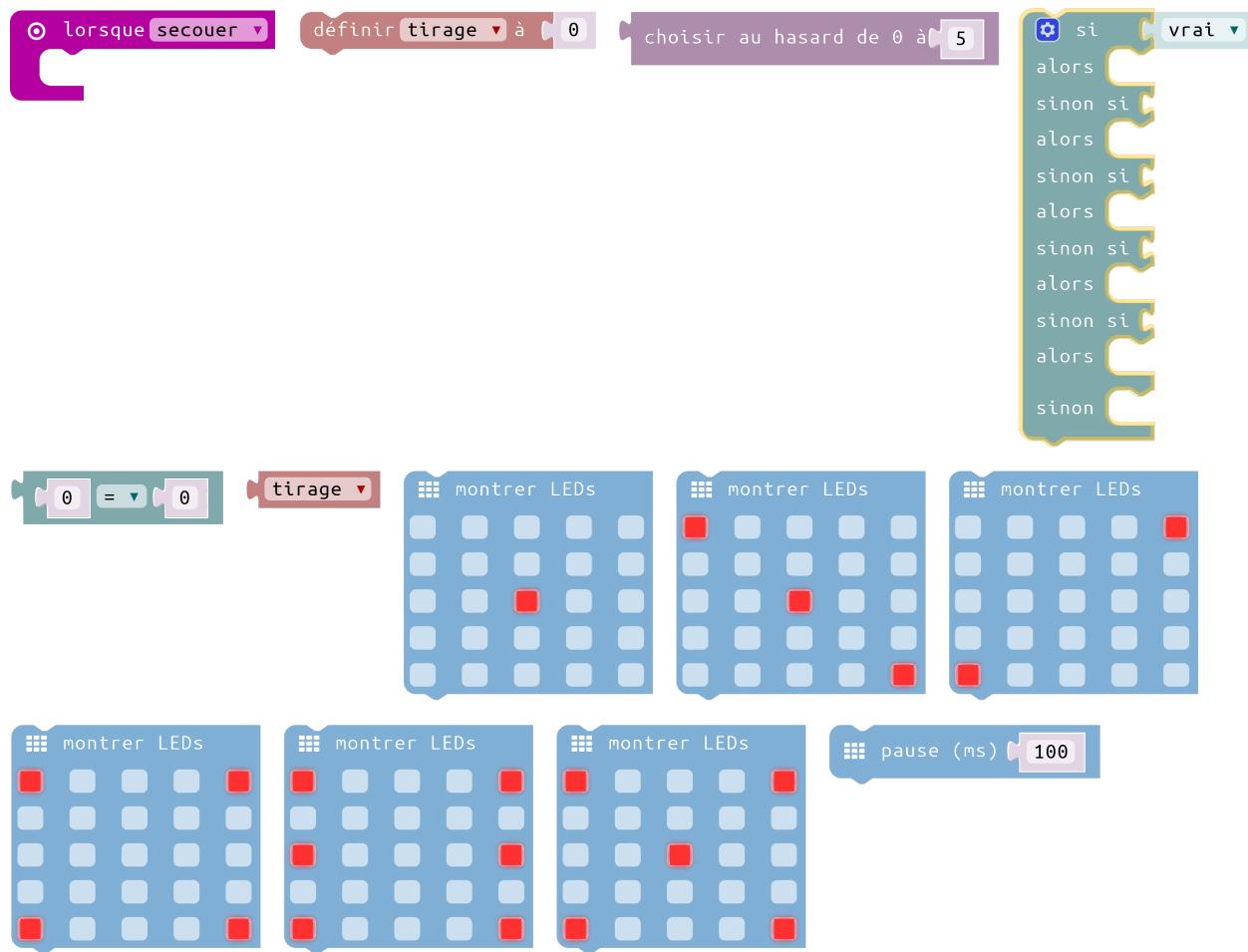
Dans ce niveau nous trouvons les notions suivantes :

- variable
- instruction conditionnelle si/sinon



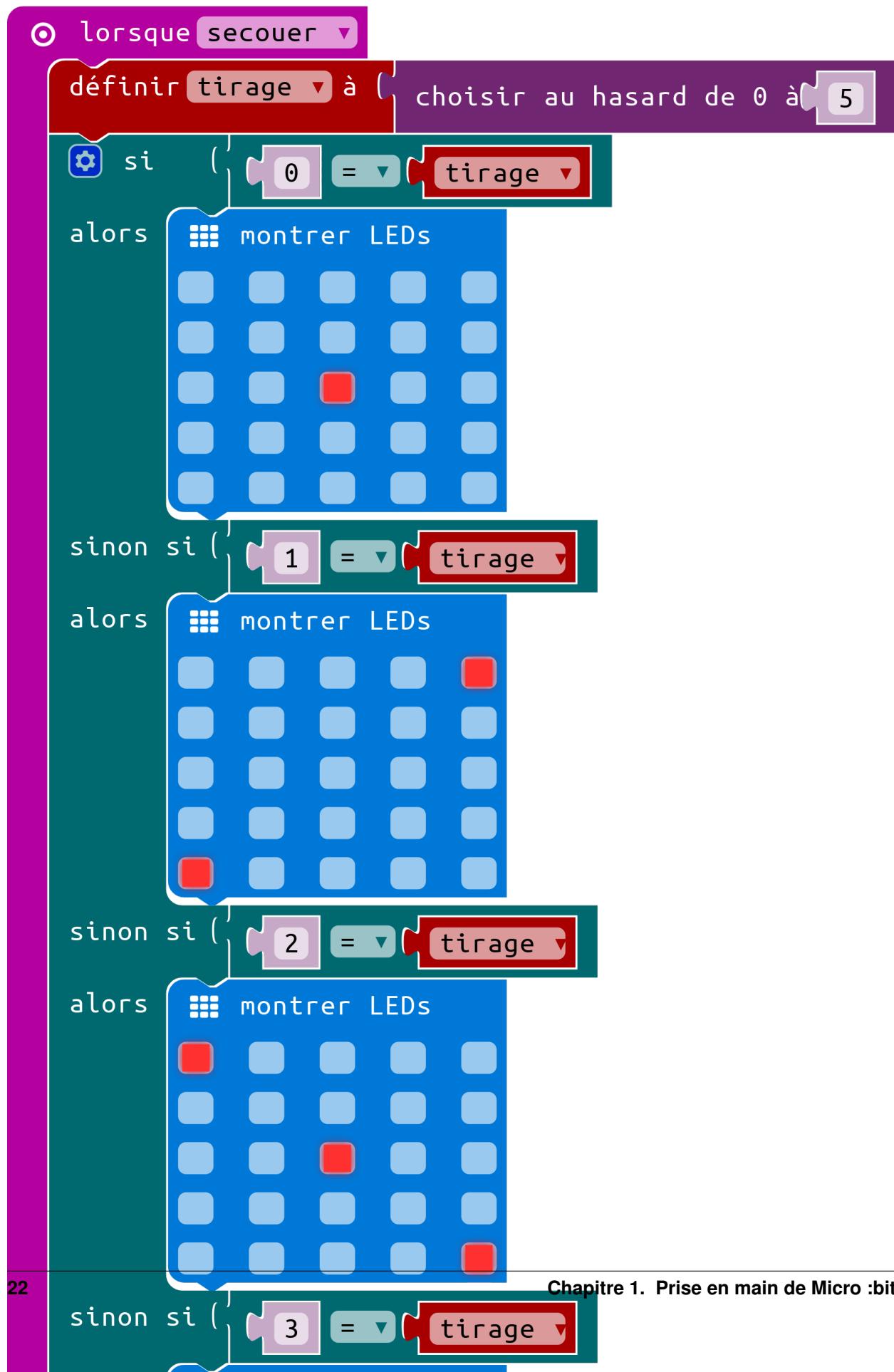
### Les blocs préconisés

On propose aux élèves d'utiliser les blocs suivant :



### Une solution possible

Le résultat escompté est le suivant :



## 1.4 Pile ou face (Python)

### 1.4.1 Description

Le but de ce projet de simuler une expérience aléatoire de lancer de pièce. A partir d'une situation simple idéale pour une prise en main avec Python il s'agit par la suite d'améliorer le programme pas à pas. *Pile ou face (blocs)* On verra qu'il ne s'agit pas forcément de la transposition en Python des activités proposées en bloc.

Il est intéressant de relever pour chaque étape l'apport que représente la programmation avec ce langage.

### Exemple(s) d'utilisation

- Domaine statistiques et probabilités du programme de mathématiques de Bac Pro
- Accompagnement personnalisé pour des élèves de seconde Bac Pro
- Algorithmique et programmation au lycée général et technologique

### 1.4.2 Progression

#### Niveau 1

Ce que l'on veut obtenir : afficher « P » ou « F » de façon aléatoire à l'issue d'une courte animation. Ce premier niveau permet de se familiariser avec les fonctions utilisées pour interagir avec le microbit. Contrairement à la programmation par bloc, il est plus efficace ici de choisir « P » ou « F » aléatoirement dans la liste composée de ces 2 singletons. De plus cela permettra facilement de truquer l'expérience aléatoire.

#### Les notions abordées

##### Dans ce niveau nous trouvons les notions suivantes :

- interactions avec le microbit (bouton, affichage)
- aléa (random)
- notion de liste

#### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

```
import random # bibliothèque pour générer de l'aléa
Image("xxxxx:xxxxx:xxxxx:xxxxx:xxxxx") # où x représente l'intensité d'une diode
# comprise entre 0 et 9
random.choice(liste) # pour choisir un élément au hasard dans une liste
["P", "F"] # liste des issues (texte) que l'on veut afficher
```

#### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *

import random

piece1 = Image(          # définition de l'image "piece1"
    "00000:"           # aucune diode n'est éclairée
    "00000:"
    "00000:"
    "99999:"          # toutes les diodes de la 4ème ligne sont éclairées
    ↪ au maximum
    "00000:")

piece2 = Image(
    "00000:"
    "90000:"
    "09000:"
    "00900:"
    "00090:")

piece3 = Image(
    "00000:"
    "00900:"
    "00900:"
    "00900:"
    "00900:")

piece4 = Image(
    "00000:"
    "00009:"
    "00090:"
    "00900:"
    "09000:")

piece5 = Image(
    "00000:"
    "00000:"
    "99999:"
    "00000:"
    "00000:")

while True:
    if button_a.get_presses():
        display.show(piece1)      # la matrice de LED montre l'image "piece1"
        sleep(200)
        display.show(piece2)
        sleep(200)
        display.show(piece3)
        sleep(200)
        display.show(piece4)
        sleep(200)
        display.show(piece5)
        sleep(200)
        display.show(piece1)
        sleep(200)
        display.show(random.choice(["P", "F"])) # affichage au hasard de P ou F
```

## Niveau 2

Ce que l'on veut obtenir : afficher « P » ou « F » de façon aléatoire à l'issue d'une courte animation. L'intérêt ici est de comprendre l'appel à une liste pour l'animation et ainsi de gagner en efficacité et en lisibilité.

### Les notions abordées

Ce niveau permet d'appréhender une utilité supplémentaire du type d'objet « liste ».

### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

```
[a, b ,c ...] # une liste ou a,b,c ... sont le nom d'images déclarées précédemment
```

### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *
import random

piece1 = Image(
    "00000:"
    "00000:"
    "00000:"
    "99999:"
    "00000:")

piece2 = Image(
    "00000:"
    "90000:"
    "09000:"
    "00900:"
    "00090:")

piece3 = Image(
    "00000:"
    "00900:"
    "00900:"
    "00900:"
    "00900:")

piece4 = Image(
    "00000:"
    "00009:"
    "00090:"
    "00900:"
    "09000:")

piece5 = Image(
    "00000:"
```

(suite sur la page suivante)

(suite de la page précédente)

```
"00000:"  
"99999:"  
"00000:"  
"00000:")  
  
pieces = [piece1, piece2, piece3, piece4, piece5, piece1] # la séquence d'images  
  
while True:  
    if button_a.get_presses():  
        display.show(pieces, delay=200) # la matrice affiche chacune des images de la liste "pieces" avec une pause de 200ms entre chaque image  
        display.show(random.choice(["P", "F"]))
```

### Niveau 3

Ce que l'on veut obtenir : afficher « P » ou « F » de façon aléatoire à l'issue d'une courte animation et compter le nombre d'issues obtenues.

#### Les notions abordées

Pour ce niveau, on va avoir besoin :

- d'une variable pour stocker le résultat du tirage
- de variables pour dénombrer les issues « P » et les issues « F »
- d'une instruction conditionnelle pour tester et agir selon le résultat du tirage

#### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

```
if: et else: # instructions conditionnelle  
== # qui permet de vérifier l'égalité entre deux objets  
+= 1 # qui permet d'incrémenter une variable de 1
```

#### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *  
  
import random  
  
p = 0 # variable stockant le nombre d'issues pile  
f = 0 # variable stockant le nombre d'issues face  
  
piece1 = Image("00000:"  
               "00000:"  
               "00000:"  
               "99999:"  
               "00000:")
```

(suite sur la page suivante)

(suite de la page précédente)

```

piece2 = Image("00000:"
               "90000:"
               "09000:"
               "00900:"
               "00090:")

piece3 = Image("00000:"
               "00900:"
               "00900:"
               "00900:"
               "00900:")

piece4 = Image("00000:"
               "00009:"
               "00090:"
               "00900:"
               "09000:")

piece5 = Image("00000:"
               "00000:"
               "99999:"
               "00000:"
               "00000:")

pieces = [piece1, piece2, piece3, piece4, piece5, piece1]

while True:
    if button_a.get_presses():
        display.show(pieces, delay=200)
        issue = random.choice(["P", "F"])
        if issue == "P":
            display.show("P")
            p += 1
            # incrémentation de la variable p (pile)
        else:
            display.show("F")
            f += 1
            # incrémentation de la variable f (face)

    if button_b.get_presses():
        display.scroll("P:"+str(p))      # affichage du nombre d'issues associées à P
        delay = 200
        display.scroll("F:"+str(f))      # affichage du nombre d'issues associées à F

```

## 1.5 Dé 6 faces (Python)

### 1.5.1 Description

Le but de ce projet de simuler une expérience aléatoire de lancer d'un dé à 6 faces. Tout comme le projet simulant un pile ou face, cette situation permet de comprendre l'intérêt des listes et va un peu plus loin dans leur utilisation.

#### Exemple(s) d'utilisation

- Algorithmique et programmation au lycée général et technologique
- Domaine statistique et probabilités du programme de mathématiques en seconde et première Bac Pro.

— Accompagnement personnalisé pour des élèves de seconde Bac Pro

### 1.5.2 Progression

Pour afficher aléatoirement un nombre entier entre 1 et 6, on peut se contenter des 3 lignes ci-dessous.

```
while True:  
    if button_a.get_presses():  
        # "str" car "display.show" n'affiche que du texte  
        display.show(str(random.randint(1, 6)))
```

Ce qui n'a pas d'autre intérêt que d'introduire la fonction `randint`. Ici, contrairement à la progression utilisée lors du projet

, nous suggérons de commencer par simuler l'affichage tel qu'il apparaît sur un vrai dé. Cela qui permet de réexploiter les listes d'images introduites avec le projet pile ou face.

### 1.5.3 Progression

#### Niveau 1

Ce que l'on veut obtenir : afficher un nombre entier entre 1 et 6 de façon aléatoire mais à la façon d'un vrai dé, c-a-d avec des points. Ce premier niveau permet d'introduire la problématique et de réinvestir les notions utilisées lors du projet pile ou face.

#### Les notions abordées

Dans ce niveau nous trouvons les notions suivantes :

—

#### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

#### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *  
  
import random  
  
un = Image("00000:00000:00900:00000:00000")  
deux = Image("00009:00000:00000:00000:90000")  
trois = Image("90000:00000:00900:00000:00009")  
quatre = Image("90009:00000:00000:00000:90009")  
cinq = Image("90009:00000:00900:00000:90009")  
six = Image("90009:00000:90009:00000:90009")  
  
issues = [un, deux, trois, quatre, cinq, six]
```

(suite sur la page suivante)

(suite de la page précédente)

```
while True:
    if button_a.get_presses():
        display.show(random.choice(issues))
        sleep(800)
        display.clear()
```

## Niveau 2

Ce que l'on veut obtenir :

### Les notions abordées

Dans ce niveau nous trouvons les notions suivantes :

### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *
import random

un = Image("00000:00000:00900:00000:00000")
deux = Image("00009:00000:00000:00000:90000")
trois = Image("90000:00000:00900:00000:00009")
quatre = Image("90009:00000:00000:00000:90009")
cinq = Image("90009:00000:00900:00000:90009")
six = Image("90009:00000:90009:00000:90009")

n1 = 0
n2 = 0
n3 = 0
n4 = 0
n5 = 0
n6 = 0

issues = [un, deux, trois, quatre, cinq, six]

while True:
    if button_a.get_presses():
        issue = random.choice(issues)
        display.show(issue)
        if issue == un:
            n1 += 1
        elif issue == deux:
            n2 += 2
```

(suite sur la page suivante)

(suite de la page précédente)

```
elif issue == trois:  
    n3 += 1  
elif issue == quatre:  
    n4 += 1  
elif issue == cinq:  
    n5 += 1  
elif issue == six:  
    n6 += 1  
sleep(1000)  
display.clear()  
  
if button_b.get_presses():  
    display.scroll(  
        str(n1)+" /"+str(n2)+" /"  
        + str(n3)+" /"+str(n4)+" /"  
        + str(n5)+" /"+str(n6)  
    )
```

### Niveau 3

Ce que l'on veut obtenir : afficher un nombre entier entre 1 et 6 de façon aléatoire en secouant l'appareil.

#### Les notions abordées

Dans ce niveau nous trouvons les notions suivantes :

#### Les éléments utiles

On propose aux élèves d'appeler les éléments suivants

#### Une solution possible

Le résultat escompté est le suivant :

```
from microbit import *  
  
import random  
  
un = Image("00000:00000:00900:00000:00000")  
deux = Image("00009:00000:00000:00000:90000")  
trois = Image("90000:00000:00900:00000:00009")  
quatre = Image("90009:00000:00000:00000:90009")  
cinq = Image("90009:00000:00900:00000:90009")  
six = Image("90009:00000:90009:00000:90009")  
  
n = [0, 0, 0, 0, 0]  
  
issues = [un, deux, trois, quatre, cinq, six]  
  
while True:
```

(suite sur la page suivante)

(suite de la page précédente)

```
if button_a.get_presses():
    i = random.randint(0, 5)
    display.show(issues[i])
    n[i] += 1
    sleep(1000)
    display.clear()

if button_b.get_presses():
    for k in range(5):
        display.scroll(str(n[k]) + " /")
```



# CHAPITRE 2

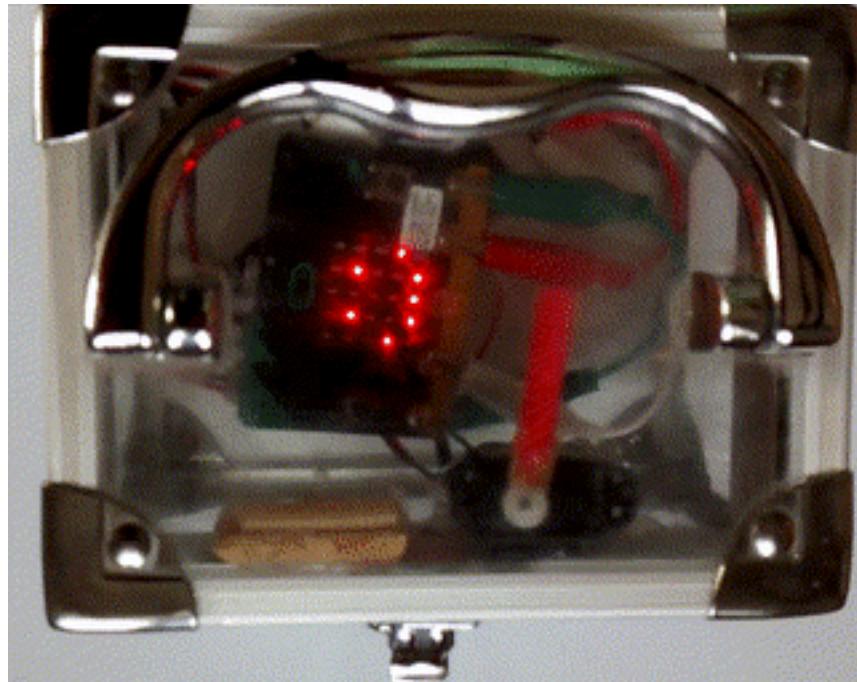
---

Projets à réaliser

---

## 2.1 Boîte fermée

### 2.1.1 Description

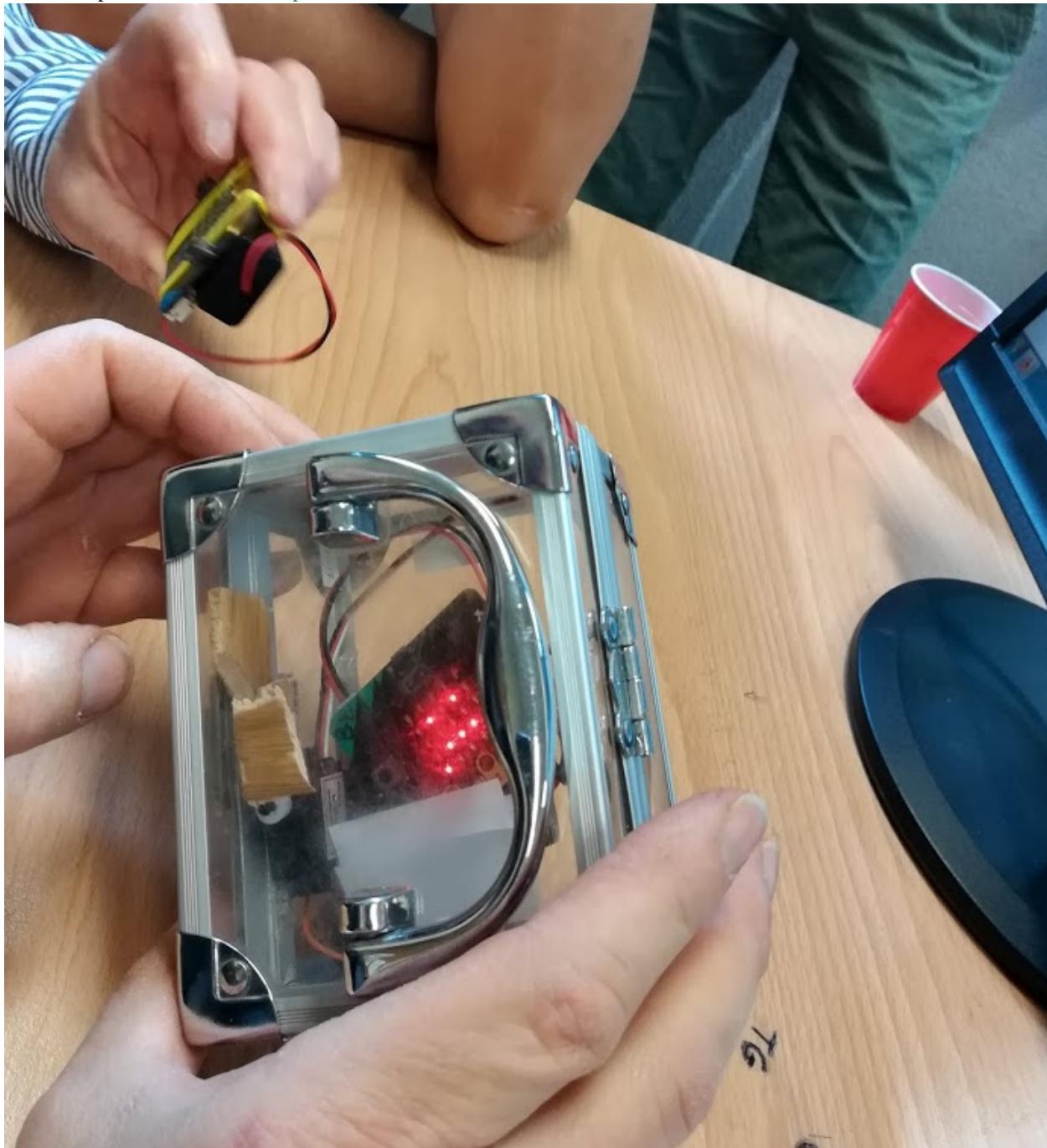


Exemple(s) d'utilisation

### Escape game

Nous avons utilisé le projet *Boîte fermée* pour un escape game proposé en stage.

— diaporama d'accueil : <http://url.univ-irem.fr/boite>



#### 2.1.2 Réalisation

## Fabriquer

Nous détaillons ici comment fabriquer et assembler le matériel nécessaire à la réalisation du projet *Boîte fermée*.

### Matériel :

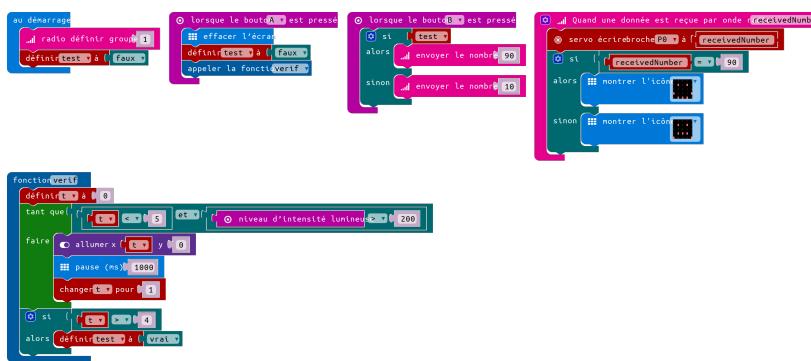
- 2 cartes microbit avec leur alimentation
- 1 servomoteur
- quelques fils de contact et pinces crocodiles
- une boite (transparente)
- pistolet à colle
- chutes de bois

### Assemblage :

- coller un morceau de bois d'environ 2 cm sur le servomoteur
- coller le servomoteur au niveau de l'ouverture de la boîte de façon à ce que la partie mobile se trouve au niveau du couvercle
- coller une cale de bois sur la partie inférieure de la boîte de façon à ce que la partie mobile du servo puisse vérrouiller lorsqu'elle vient en butée
- brancher l'alimentation sur servo sur les bornes 3v et GND du microbit et le fil de commande sur la borne P0

## Coder

Nous détaillons ici le code nécessaire à la réalisation du projet *Boîte fermée*.



## 2.2 Pierrot et Simon

### 2.2.1 Description

---

**À faire :** capture d'écran / gif animée

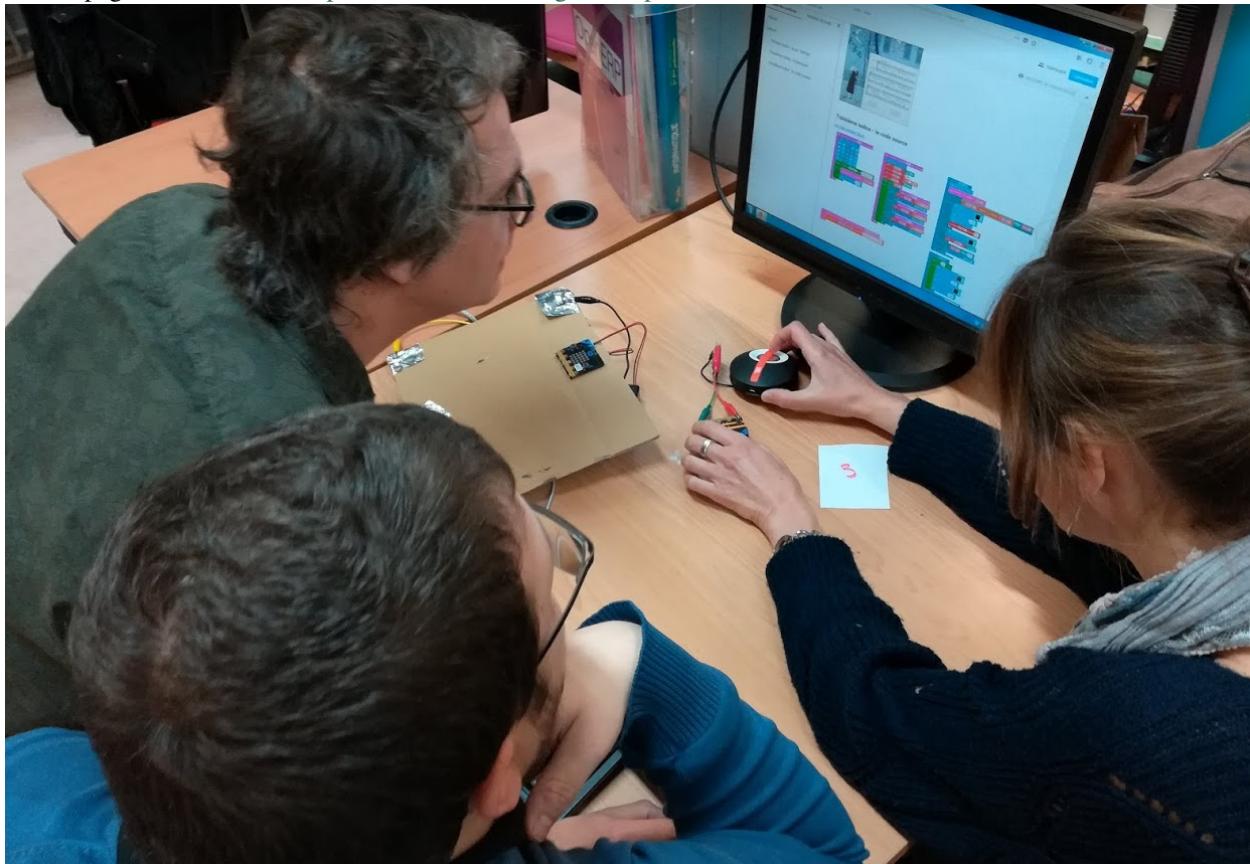
---

### Exemple(s) d'utilisation

#### Escape game

Nous avons utilisé le projet *Pierrot et Simon* pour un escape game proposé en stage.

- diaporama d'accueil : <http://url.univ-irem.fr/pierrot>
- page de formation : <http://url.univ-irem.fr/algo1718-pierrot>



## 2.2.2 Réalisation

### Fabriquer

Nous détaillons ici comment fabriquer et assembler le matériel nécessaire à la réalisation du projet *Pierrot et Simon*.

---

**À faire :** tout faire.

---

### Coder

Nous détaillons ici le code nécessaire à la réalisation du projet *Pierrot et Simon*.

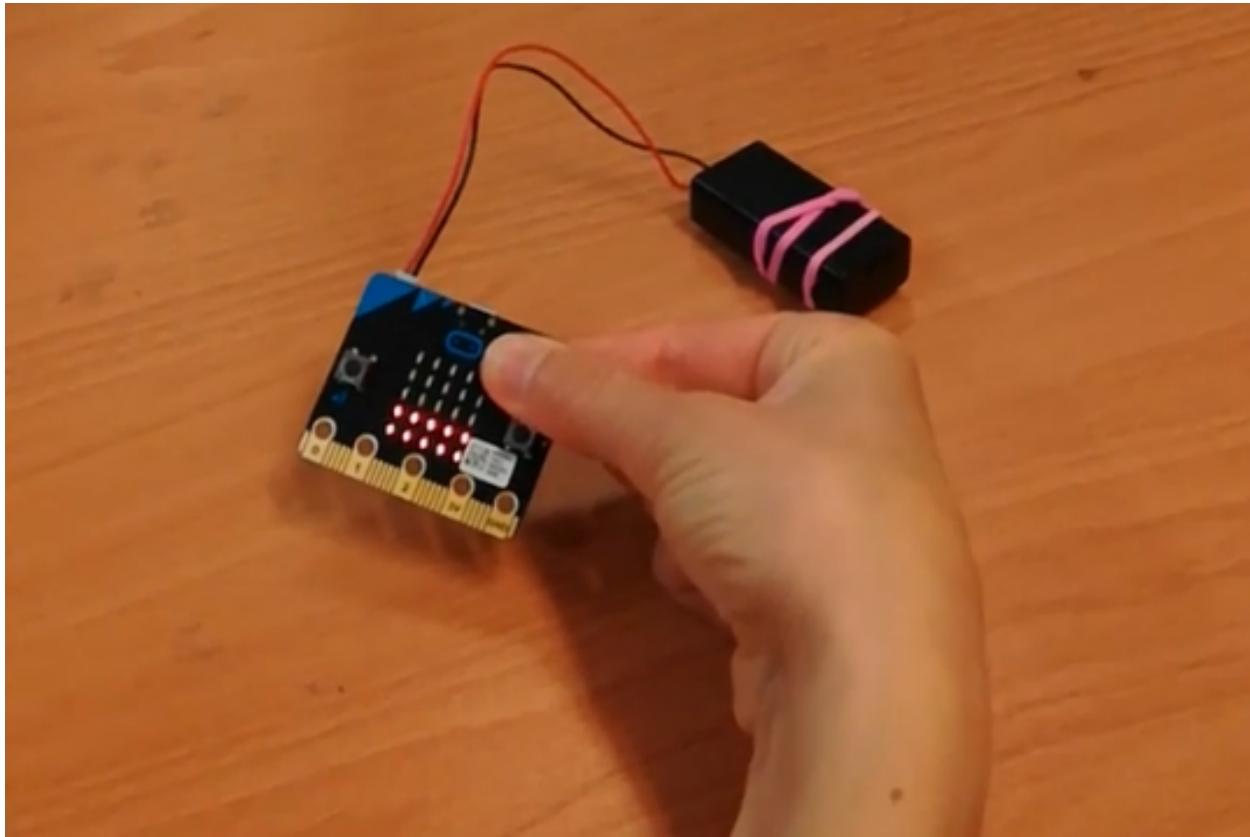
---

**À faire :** tout à faire !

---

## 2.3 Températures

### 2.3.1 Description



Le code téléchargé dans le Micro :bit permet d'afficher un **code secret**. Pour cela, il faut que la température augmente et dépasse les 34°C. L'écran affiche une jauge qui se remplit.

Une fois la température atteinte, le code secret s'affiche après une petite animation.

#### Exemple(s) d'utilisation

##### Escape game

Nous avons utilisé le projet *Températures* pour un escape game proposé en stage.

- diaporama d'accueil : <http://url.univ-irem.fr/temp>
- page de formation : <http://url.univ-irem.fr/algo1718-temp>

Le but était d'afficher un code secret permettant d'accéder à l'énigme suivante.



Les stagiaires devaient donc prendre connaissance du diaporama d'accueil qui les renvoyaient vers des indices et le code source téléchargé dans le micro:bit.

À l'aide de l'étude du code source, des indices et la bidouille, les stagiaires devaient effectuer les manipulations nécessaires à l'affichage du code source.

De très bons moments pour tous !

### 2.3.2 Réalisation

#### Fabriquer

Pour le projet *Températures*, il n'y a besoin de presque rien :

- carte micro:bit;
- alimentation électrique (pile par exemple).

#### Coder

Le code nécessaire à la réalisation du projet *Températures* a été écrit en micropython. Vous trouverez ci-dessous :

- [Le code, étape par étape](#)
- [Le code final](#)

#### Le code, étape par étape

1. Incluons la bibliothèque Micro:bit

```
from microbit import *
```

2. Créons les images qui nous servirons à animer l'écran. La luminosité d'une diode varie de 0 (éteinte) à 9 (maximale).

Lorsque la température augmente, l'affichage passe progressivement de image1 à image 5.

```
image1 = Image(
    '00000:'
    '00000:'
    '00000:'
    '00000:'
    '99999')
image2 = Image(
    '00000:'
    '00000:'
    '00000:'
    '99999:'
    '77777')
image3 = Image(
    '00000:'
    '00000:'
    '99999:'
    '77777:'
    '77777')
image4 = Image(
    '00000:'
    '99999:'
    '77777:'
    '77777:'
    '77777')
image5 = Image(
    '99999:'
    '77777:'
    '77777:'
    '77777:'
    '77777')
```

3. Il y aura deux états dans le jeu :

- La variable victoire est vrai et l'écran affiche le code secret.
- la variable victoire est fausse et l'écran affiche l'énigme.

Au début, la variable est donc fausse.

```
victoire = False
```

4. La phase de configuration est terminée. Passons maintenant à la boucle qui... tourne en boucle.

Tout ce qui suivra cette codee sera donc indenté (tabulation).

```
while True:
```

5. Nous envisageons trois actions possibles :

- (a) le jeu se réinitialise grâce au bouton A ;

```
if button_a.is_pressed():
    victoire = False
```

- b) le jeu est gagné et l'écran affiche le code final (après une petite animation) ;

```
if victoire:  
    # petite image joyeuse  
    display.show(Image.HAPPY)  
    sleep(500)  
    # code secret à afficher...  
    display.scroll("XXXXXX")
```

(c) le jeu est en cours et l'écran affiche les images.

```
if not victoire:
```

Lire la température

```
temp = temperature()
```

Plus la température augmente, plus les images affichées remplissent l'écran

```
if temp < 29:  
    display.clear()  
elif 29 <= temp < 30:  
    display.show(image1)  
elif 30 <= temp < 31:  
    display.show(image2)  
elif 31 <= temp < 32:  
    display.show(image3)  
elif 32 <= temp < 33:  
    display.show(image4)  
elif 33 <= temp < 34:  
    display.show(image5)
```

et enfin, si la température dépasse 34°C, alors là on passe en mode victoire vrai. On ajoute une petite animation pour montrer que la victoire approche.

```
elif 34 <= temp:  
    victoire = True  
    # petite animation  
    for i in range(2):  
        display.show(Image.SQUARE_SMALL)  
        sleep(100)  
        display.show(Image.SQUARE)  
        sleep(100)
```

Pour finir, une pause syndicale de 500ms.

```
sleep(500)
```

### Le code final

```
1  # -*- coding: utf-8-*# Encoding cookie added by Mu Editor  
2  from microbit import *  
3  
4  # définir mes images perso  
5  # pour les lignes qui se colorent  
6  image1 = Image(  
7      '00000:'  
8      '00000:'
```

(suite sur la page suivante)

(suite de la page précédente)

```

9    '00000:'
10   '00000:'
11   '99999')
12 image2 = Image(
13     '00000:'
14     '00000:'
15     '00000:'
16     '99999:'
17     '77777')
18 image3 = Image(
19     '00000:'
20     '00000:'
21     '99999:'
22     '77777:'
23     '77777')
24 image4 = Image(
25     '00000:'
26     '99999:'
27     '77777:'
28     '77777:'
29     '77777')
30 image5 = Image(
31     '99999:'
32     '77777:'
33     '77777:'
34     '77777:'
35     '77777')

36
37 # booléen pour savoir si l'énigme est réussie
38 victoire = False

39
40 # à faire toujours et toujours...
41 while True:
42     # utiliser le bouton A pour réinitialiser
43     if button_a.is_pressed():
44         victoire = False

45
46 # si l'énigme est résolue
47 if victoire:
48     # petite image joyeuse
49     display.show(Image.HAPPY)
50     sleep(500)
51     # code secret à afficher...
52     display.scroll("XXXXXX")

53
54 # si l'énigme n'a pas été résolue
55 if not victoire:
56     # lire la température (en °C)
57     temp = temperature()
58     # affichage des images en fonction
      # de temp
59     if temp < 29:
60         display.clear()
61     elif 29 <= temp < 30:
62         display.show(image1)
63     elif 30 <= temp < 31:
64         display.show(image2)

```

(suite sur la page suivante)

(suite de la page précédente)

```
66     elif 31 <= temp < 32:
67         display.show(image3)
68     elif 32 <= temp < 33:
69         display.show(image4)
70     elif 33 <= temp < 34:
71         display.show(image5)
72     # victoire !
73     elif 34 <= temp:
74         victoire = True
75         # petite animation
76         for i in range(2):
77             display.show(Image.SQUARE_SMALL)
78             sleep(100)
79             display.show(Image.SQUARE)
80             sleep(100)
81         sleep(500)
```

## 2.4 Coffre fort

### 2.4.1 Description

---

À faire : capture d'écran / gif animée

---

#### Exemple(s) d'utilisation

##### Escape game

Nous avons utilisé le projet *Coffre fort* pour un escape game proposé en stage.

- diaporama d'accueil : <http://url.univ-irem.fr/coffre>
- page de formation : <http://url.univ-irem.fr/algo1718-coffre>



## 2.4.2 Réalisation

### Fabriquer

Nous détaillons ici comment fabriquer et assembler le matériel nécessaire à la réalisation du projet *Coffre fort*.

---

**À faire :** tout faire.

---

### Coder

Nous détaillons ici le code nécessaire à la réalisation du projet *Coffre fort*.

---

**À faire :** tout à faire !

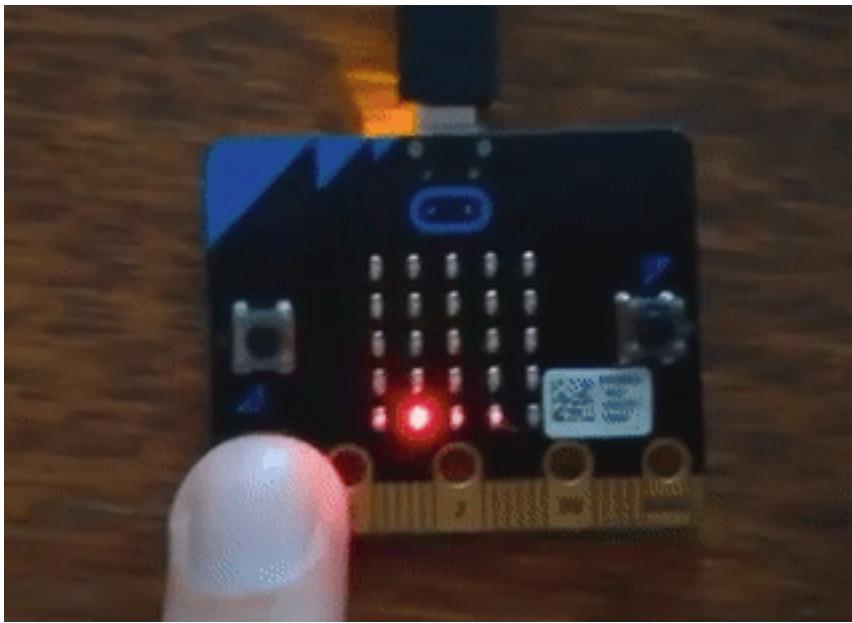
---

## 2.5 Planche de Galton

### 2.5.1 Description

Le but de ce programme est de simuler le parcours de billes sur une planche de Galton. Lorsque la bille tombe, elle a deux possibilités : elle descend verticalement ou elle descend verticalement en se décalant horizontalement vers la droite.

La bille position de la bille est représentée par l'allumage d'une diode.



#### Type de programmation :

Le projet est ici programmé en python.

#### Exemple(s) d'utilisation

- Domaine statistiques et probabilités du programme de mathématiques de Bac Pro
- Accompagnement personnalisé pour des élèves de seconde Bac Pro

### 2.5.2 Réalisation

#### Coder

Nous détaillons ici le code nécessaire à la réalisation du projet *Planche de Galton*.

```
from microbit import *
from random import random, seed

n = [0, 0, 0, 0, 0]      # le tableau contenant les compteurs
```

(suite sur la page suivante)

(suite de la page précédente)

```

def aff(n, m):          # la fonction affichant le graph
    q = n // 9           # nombre de led eclairé totalement
    r = n % 9            # portion de la dernière led éclairée
    for i in range(0, q):
        display.set_pixel(m, 4-i, 9)
    display.set_pixel(m, 4-q, r)

def chute(t):           # fonction affichant la chute
    display.clear()
    y, x = 0, 0
    display.set_pixel(x, y, 9)
    sleep(t)
    while y < 4:
        display.clear()
        if random.randint(0, 1):      # si aléa entre 0 ou 1 est vrai
            y = y + 1               # on augmente y de 1
        else:
            x = x + 1
            y = y + 1
        display.set_pixel(x, y, 9)
        sleep(t)
    n[x] = n[x]+1           # incrementation du compteur de la position x
    display.set_pixel(x, y, 1)

while True:
    if button_a.is_pressed():
        chute(500)

    elif button_b.get_presses():
        n = [0, 0, 0, 0, 0]
        for k in range(80):
            chute(round(500 / (1.05**k))) # accélération de la chute
            for j in range(5):
                aff(n[j], j)
            sleep(200)
        print(n)

```

## Fabriquer

Nous détaillons ici comment fabriquer et assembler le matériel nécessaire à la réalisation du projet *Planche de Galton*.

---

**À faire :** tout faire.

---



# CHAPITRE 3

---

## Index et page de recherche

---

- genindex
  - search
-



---

## Index

---

### A

accéléromètre, 43  
audio, 36

### G

galton, 44

### L

luminosité, 35

### M

micropython, 36, 38, 43, 44

### P

programmation par blocs, 35

### S

servo, 35

### T

température, 38