Simuler un dé avec Micro:bit (et python)

Description

Objectif

Le but de ce projet est de simuler une expérience aléatoire de lancer d'un dé à 6 faces mais en nous appuyant sur les possibilités offertes par le langage Python.
Ainsi, on ne se contentera pas seulement d'afficher l'issue, mais on montrera comment stocker et traiter facilement les effectifs obtenus.
Ici, contrairement à l'activité proposée en programmation par blocs, nous suggérons de commencer par simuler l'affichage tel qu'il apparaît sur un vrai dé. Cela permet de réexploiter les listes d'images introduites avec le projet pile ou face.
Cette activité permettra de couvrir de nombreuses capacités du programme de mathématiques de seconde Bac Pro, tant dans le domaine des statistiques que celui de l'algorithmique.

Intérêt

expérimenter En partant d'une modélisation simple, l'élève va pouvoir observer une expérience aléatoire, dont il va facilement pouvoir visualiser et traiter les résultats.

programmation événementielle Pour déclencher les tirages et l'affichage, on peut utilise les boutons, mais on peut imaginer utiliser la détection de mouvements.

gestion des listes Même les listes ne font pas partie explicitement des types de données que les élèves sont sensés maîtrisées, il est difficile de s'en passer en programmation. Cependant, il ne semble pas insurmontable de les initier aux principes de création, modification et parcours de listes dans la mesure où l'on retrouve les notions de variables et de boucle for. La majeure difficulté pourrait éventuellement provenir des index.

programmation fonctionnelle le calcul des fréquences peut se faire au travers d'une fonction, ce qui permet de scinder la production des données et leur traitement.

de multiples déclinaisons possibles Une fois la situation du tirage d'un dé bien établie, il est facilement envisageable d'étendre à d'autres situations : dés non conventionnels, dés multiples.

Matériel

— 1 ×  Micro:bit

— 1 × IDE programmation python (Mu) https://codewith.mu/ ou interface de programmation ligne https://python.microbit.org/v/1.1

Remarques

Méthode

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():
        display.show(str(randint(1, 6)))
        sleep(800)
```

Cette solution peut être envisagée afin de simplifier l'approche, mais la création d'image permet de travailler le repérage et la représentation des nombres, en plus de proposer un affichage plus attrayant.

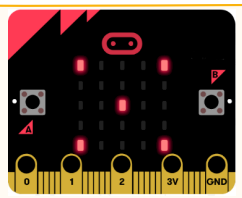
1) **Initiation - Mise en place du modèle.**
   Dans cette activité, il s'agit de proposer aux élèves de créer les images et d'associer l'affichage au résultat d'un tirage aléatoire.
2) **Intermédiaire - Enregistrement des résultats.**
   L'étape suivante consistera d'une part à stocker les effectifs des différentes issues dans une listes, et d'autre part à afficher les données recueillies.
3) **Intermédiaire - Calcul des fréquences**
   Le dernier niveau consistera à déterminer les fréquences des issues. Pour cela on utilisera une fonction pour calculer et arrondir.

Niveau initiation

Activité élève

Durée	Public	Maths	Sciences	Algo
0,5 h	2de	probabilités		affichage boucles listes

Activité



Ta Mission : Utiliser une carte Micro:bit pour simuler un dé, en utilisant le langage Python.

Nous utiliserons la bibliothèque `random` et la fonction `randint()` pour tirer un nombre entier aléatoirement entre deux bornes. Il faut donc importer un module en plus de microbit :

```
from microbit import *
from random import randint
```

Le programme précédent devra être complété avec les éléments suivants :

1) Dans un premier temps nous allons créer les images nécessaires pour afficher les nombres comme sur un vrai dé.
   Par exemple le code ci-dessous permet de créer l'image représentant l'issue "cinq".

```
cinq = Image("90009:00000:00900:00000:90009")
```

   Chaque nombre correspond à une diode, éclairée de 0 à 9.

2) Ces images sont ensuite enregistrée dans une liste :

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

3) Après le démarrage de la boucle `While True:` on déclenchera le tirage avec le bouton A : `button_a.get_presses()` .

4) La première image du tableau `issues` est numérotée à 0 et la dernière à 5. On effectue donc un tirage aléatoire d'un entier entre 0 et 5, puis on demande au `Micro:bit` d'afficher l'image correspondant à ce numéro.

```
        i = randint(0, 5)
        display.show(issues[i])
```

Vérifie ton code avec  et flash-le sur la carte avec 

Notes pour l'enseignant

MÉTHODE

Proposition de résolution : Il faut bien entendu importer les modules nécessaires

```
from microbit import *
from random import randint
```

puis créer les images

```
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")
```

et enfin créer la boucle permettant d'afficher le résultat d'un tirage :

```
    if button_a.get_presses():
        display.show(str(randint(1, 6)))
        sleep(800)
        display.clear()
```

REMARQUE

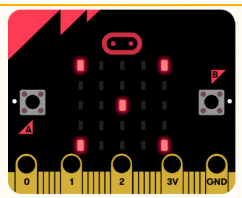
Les deux dernières lignes ne sont pas indispensables, mais elles permettent de mieux différencier les tirages, surtout lorsque l'on tire le même nombre.

Niveau intermédiaire

Activité élève

Durée	Public	Maths	Sciences	Algo
0,5 h	2de	probabilités		affichage boucles listes

Activité



Ta Mission : Utiliser une carte Micro:bit pour simuler un dé, et dénombrer les issues en utilisant le langage Python.

Comme pour l'activité précédente, nous utiliserons la bibliothèque `random` et la fonction `randint()` , le programme commencera donc aussi par :

```
from microbit import *
from random import randint
```

Le programme de l'activité précédente devra être complété avec les éléments suivants, à vous de déterminer quelle doit être leur position dans le programme :

1) Nous allons devoir stocker les effectifs des différentes issues, pour cela il faudra créer un tableau de données que nous initialiserons avec des "0". Par exemple :
   `data = [0, 0, 0, 0, 0, 0]`
2) Lorsqu'un nombre est tiré, il faudra augmenter l'effectif correspondant de 1. Étant donné que l'on se sert déjà du nombre tiré aléatoirement pour afficher l'image, il n'y a qu'une ligne à rajouter. Par exemple si le tableau s'appelle "data" :
   `data[i] = [data[i]+1`
3) Pour déclencher l'affichage des effectifs, on utilisera le bouton B :
   `button_b.get_presses()`
4) Pour parcourir le tableau, il faudra faire une boucle `for ... in ...` sur le tableau et afficher chaque élément du tableau avec `display.scroll(...)` .

```
        for effectif in data :
            display.scroll(str(effectif)+" /")
```

Notes pour l'enseignant

MÉTHODE

Proposition de résolution :

Le début du programme est inchangé.

```
from microbit import *
from random import randint
```

```
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]
```

Il faut ajouter l'initialisation du tableau d'effectifs :

```
data = [0, 0, 0, 0, 0, 0]
```

Dans la boucle permettant d'afficher le résultat d'un tirage, il faut inclure l'incrémentation :

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

L'affichage des effectifs se fait par parcours du tableau et défilement :

```
    if button_b.get_presses():
        for effectif in data :
            display.scroll(str(effectif)+" /")
```

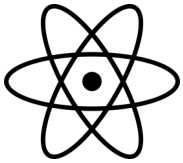
REMARQUE

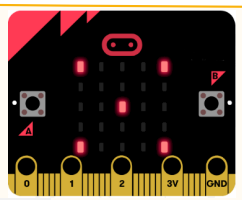
Cette activité étant très guidée, la complexité réside dans le positionnement des instructions proposées dans le programme et dans la compréhension du fonctionnement des listes.

Les listes n'étant pas spécifiquement demandées dans le référentiel de seconde, elles sont à utiliser avec précaution, il est aussi envisageable de créer des variables pour enregistrer l'effectif de chaque issue, mais cela alourdi considérablement le programme.

Niveau avancé

Activité élève

 Durée	 Public	 Maths	 Sciences	 Algo
0,5 h	2de	probabilités		affichage boucles listes fonctions



Ta Mission : Utiliser une carte Micro:bit pour simuler un dé, et observer la fluctuaion des fréquences en utilisant le langage Python.

Comme pour l'activité précédente, nous utiliserons la bibliothèque `random` et la fonction `randint()` , le programme commencera donc aussi par :

```
from microbit import *
from random import randint
```

Le programme de l'activité précédente devra être complété avec les éléments suivants, à vous de déterminer quelle doit être leur position dans le programme :

1) Nous allons ajouter une fonction qui nous permettra de calculer les fréquences de chaque issue.
2) Pour calculer les fréquences, il nous faut connaître l'effectif total des tirages, et donc pour cela nous devons créer une variable N.
3) Afin d'obtenir rapidement un grand nombre de tirage, nous allons inclure les instructions liés au tirage dans une boucle de 10 répétitions.
4) Pour déclencher l'affichage des fréquences, on utilisera le bouton B : `button_b.get_presses()`
5)

Notes pour l'enseignant

Méthode

Proposition de résolution :

Le début du programme est inchangé.

```
from microbit import *
from random import randint
```

```
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]
```

Il faut ajouter l'initialisation du tableau d'effectifs :

```
data = [0, 0, 0, 0, 0, 0]
```

Dans la boucle permettant d'afficher le résultat d'un tirage, il faut inclure l'incrémentation :

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

L'affichage des effectifs se fait par parcours du tableau et défilement :

```
    if button_b.get_presses():
        for effectif in data :
            display.scroll(str(effectif)+" /")
```

Remarque

Cette activité étant très guidée, la complexité réside dans le positionnement des instructions proposées dans le programme et dans la compréhension du fonctionnement des listes.

Les listes n'étant pas spécifiquement demandées dans le référentiel de seconde, elles sont à utiliser avec précaution, il est aussi envisageable de créer des variables pour enregistrer l'effectif de chaque issue, mais cela alourdi considérablement le programme.

À propos de cette publication

Pourquoi les objets connectés ?

Alors que dans certaines disciplines le temps commence à manquer pour traiter l'ensemble du programme, certains évoquent déjà l'idée d'en faire plus !

En effet, les enseignants utilisent déjà les outils numériques. Par exemple, dans les classes de mathématiques, l'utilité du tableur et de GeoGebra n'est plus à démontrer. Jusqu'à l'introduction de l'algorithmique, ces deux logiciels efficaces et maîtrisés par les enseignants étaient amplement suffisants. Est-ce donc juste un effet de mode de faire cours avec les robots (Thymio, Mbot), les objets programmables et connectés (Arduino, `Micro:bit`, STM education, Raspberry Pi) ou est-ce une nouvelle façon d'aborder notre enseignement ? Ces nouvelles possibilités technologiques, forcément chronophages, nous permettront-elles de traiter un contenu disciplinaire exigeant dans un cadre institutionnel contraignant ?

Nous n'avons bien sûr pas toutes les réponses à ces questions mais nous pensons que lorsqu'il est accompagné de certains de ces outils, notre enseignement a beaucoup à y gagner.

L'introduction de l'algorithmique en lycée professionnel nous interroge. Longtemps il nous a semblé impensable et inenvisageable d'avoir à enseigner un langage de programmation comme Python auprès d'un public d'élèves globalement en difficulté avec les mathématiques. Fort de ce constat, nous avons cherché les moyens de lier les mathématiques à la logique et au raisonnement algorithmique. C'est pourquoi nous avons exploré les potentialités des objets connectés.

Notre postulat est double. Nous pensons que :

— grâce à des situations réelles et concrètes, les objets connectés facilitent la mise en activité de tous les élèves ;

— grâce à des activités simples mais évolutives centrées autour de réalisations matérielles, la dimension affective du travail est valorisée. Soyons fous et espérons que l'élève tisse une histoire personnelle avec l'activité, qu'il soit fier du travail accompli et qu'il prenne également du plaisir à expliquer et à montrer ses réalisations.

En devenant de plus en plus simples, accessibles et facilement utilisables, les objets connectés permettent d'aborder des contenus disciplinaires et de développer des compétences transversales essentielles pour l'élève.

En travaillant à partir des objets connectés, la situation de départ est plus concrète et l'objectif à atteindre suffisamment clair pour l'élève. Plus ou moins guidé selon son niveau d'expertise technique, il est alors libre dans sa démarche. Avec des interfaces de programmation accompagnées parfois de simulateurs, la démarche par essais et erreurs a ici toute sa place. Par ailleurs, l'élève devra clarifier sa pensée avant de verbaliser ses idées en langage naturel. Il pourra ainsi proposer et élaborer un modèle acceptable par la machine pour enfin traduire son algorithme en se pliant à la rigueur du langage de programmation.

Effectuant régulièrement des va-et-vient entre abstraction et réalité, cherchant à valider son algorithme à partir d'un visuel ou d'une exploitation des résultats, l'élève entre progressivement dans la modélisation.

Les scénarios proposés dans cette brochure permettent tout cela : une approche des mathématiques et des sciences qui laisse la place à l'expérimentation : manipulation, programmation et auto-validation.

Qui sommes-nous ?

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

groupe Lycée Professionnel innovation, expérimentation et formation de l'IREM d'Aix-Marseille

Notre groupe, Innovation, Expérimentation et Formation en Lycée Professionnel (InEFLP) consacre une partie de son travail à l'enseignement de l'algorithmique en classes de lycée professionnel. Dans le cadre de cette recherche, nous explorons les objets connectés tels que Arduino, `Micro:bit`, `STM32 Éducation` ou mbot.

Liens utiles

Page du groupe InEFLP
`http://url.univ-irem.fr/ineflp`

IREM de Marseille Site académique de l'IREM de Marseille
`http://url.univ-irem.fr/mars`

Portail des IREM Site national des IREM
`http://www.univ-irem.fr/`

Formation à l'algorithmique LP et SEGPA Padlet de utilisé lors de nos formations académiques
`http://url.univ-irem.fr/stage-algo`

Collecte de ressources pour `Micro:bit` Padlet sur `Micro:bit` utilisé en formation
`http://url.univ-irem.fr/algo2017-microbit`

Brochure sur `Micro:bit` Publication de la C2i TICE pour une prise en main de `Micro:bit`
`http://url.univ-irem.fr/c2it-mb-t1-pdf`

Description `Micro:bit` Fiche sommaire de description de `Micro:bit`
`http://url.univ-irem.fr/ineflp-microbit`

Site IREM dédié à `Micro:bit` Site de ressources sur `Micro:bit` du groupe
`http://url.univ-irem.fr/o`

Un extrait de la brochure

Les objets connectés pour enseigner l'algorithmique en lycée professionnel

< version du 30 janvier 2020 >