TP DICTIONNAIRE : MELODIES

durée = 35 min

Objectif :

Voici quelques mélodies qu'on souhaiterait faire jouer :

```
melodie1 = "do - do - do - re - mi - re - do - mi - re - re - do"
melodie2 = "mi mi mi mi mi mi mi sol do re mi fa fa fa mi mi mi re re re mi re sol mi mi
mi mi mi mi mi sol do re mi"
```

I. Initiation

Dans le module `music` :

- `music.pitch(fréquence_en_Hz, durée_en_ms, broche)` génère un son de fréquence maximale 10 000 Hz (un entier).
  La limitation de durée est facultative, par défaut elle est illimitée `-1` .
  La broche est facultative, par défaut c'est `pin0` .
- `music.play(music.BIRTHDAY)` joue la mélodie.
- `music.stop(broche)` stoppe le son sur la broche.
  La broche est facultative, par défaut c'est `pin0` .

La liste des mélodies :
`dir(music)` tapé dans REPL :
'DADADADUM', 'ENTERTAINER', 'PRELUDE', 'ODE', 'NYAN', 'RINGTONE', 'FUNK', 'BLUES', 'BIRTHDAY', 'WEDDING', 'FUNERAL', 'PUNCHLINE', 'PYTHON', 'BADDY', 'CHASE', 'BA_DING', 'WAWAWAWAA', 'JUMP_UP', 'JUMP_DOWN', 'POWER_UP', 'POWER_DOWN'

Exemple 1 :

```
1 from microbit import *
2 import music
3
4 music.play(music.BLUES)
5 for i in range (1,11):
6     music.pitch(100*i, 2000, pin0) # i*100 Hz pendant 2 secondes
7     print('Un son de fréquence',i*100,'Hz pendant 2 secondes')
```

Exemple 2 :

```
 1 from microbit import *
 2 import music
 3
 4 while True:
 5     music.pitch(440,1000)
 6     sleep(1000)
 7     music.pitch(660,500)
 8     sleep(1000)
 9     music.pitch(880,750)
10     sleep(1000)
11     music.stop()
12     sleep(1000)
```

Exercice :

1) Tester ces deux exemples.
2) Jouer la musique "Python"
3) Jouer 3 fois de suite : un son de 400Hz pendant 1s, un son de 800Hz pendant 0,5s puis un son de 1200Hz pendant 0,25s

II. Travaux pratiques

A. 1ère étape : transformer une chaine de caractères en liste

Ma méthode `split` permet de séparer une chaine de caractères selon un séparateur donné. Les élements séparés sont mis dans une liste.
Voici un exemple d'utilisation de `split` :

```
### Transforme la chaine en liste selon le séparateur "; " (avec l'espace)
liste = "thé; café; jus d'orange; eau gazeuse".split("; ")
# le résultat est la liste de str :
# liste = ['thé', 'café', "jus d'orange", 'eau gazeuse']

### Idem selon le séparateur espace
donnée = "13 -5 8 12.1"
liste = donnée.split(" ")
# attention, le résultat est une liste de str :
# liste = ['13', '-5', '8', '12.1']
```

Exercice :
Choisir `melodie1` ou `melodie2` donnée au début du TP et la transformer en une liste de notes en utilisant la méthode `split`.
Par exemple `melodie1` commencera par ; `["do", "do", "do", "ré", …]`

B. 2ème étape : Créer des dictionnaires : note→ fréquence

Exercice
Choisir une octave et créer un dictionnaire où les clés sont les 7 notes de cette octave et les valeurs sont les fréquences :

NOTE	1ère octave	2e octave
do	131	262
ré	147	294
mi	165	330
fa	175	349
sol	196	392
la	220	440
si	247	494
(do)	(262)	(523)

C. 3ème étape : jouer la mélodie

Exercice
Au moyen d'une boucle, parcourir la liste des notes de `melodie1` ou `melodie2` et grâce au dictionnaire jouer la fréquence de la note avec l'instruction `music.pitch(fréquence_en_Hz, durée_en_ms)` .

D. 4ème étape : jouer en respectant la durée

Voici une 3ème mélodie avec un peu de rythme cette fois : en multipliant l'entier par 125 ms, vous obtiendrez la durée de la note.

```
melodie3 = "sol 4, sol 4, sol 4, mi 3, si 1, sol 4, mi 3, si 1, sol 8"
```

Par exemple : `sol 4` signifie qu'il faut jouer la note « sol » pendant 500 ms (car 4x125 = 500).

Exercice
Adapter le code précédent pour jouer cette nouvelle mélodie.

Indice :
Après le premier split, vous devrez utiliser un 2e split sur chaque élément pour séparer la note et la durée. Par exemple, en appliquant un split sur `sol 4` , on obtient la note `sol` et la durée `4`.

NB : Petite correction pour les oreilles absolues : le « si » est plutôt un « si bémol ».