ALGO2 – Algorithmique et Programmation 2

Fiche de TP supplémentaire - 1

Mastermind

Le but de ce tp est de réaliser un jeu de mastermind. Le mastermind est un jeu de société qui se joue à deux. Ils disposent de pions colorés (nous considérerons les couleurs violet, jaune, rouge, orange, rose, bleu foncé, bleu clair, vert et le vide). Un joueur choisit un code secret de quatre pions parmi ces couleurs. Le second joueur va essayer de trouver le code. Il a droit à dix essais. À chaque proposition qu'il fait, le premier joueur lui donne deux indications : le nombre de pions de la bonne couleur qui sont bien placés (BP), et le nombre de pions de la bonne couleur qui sont mal placés (MP).

Vous allez réaliser une première version en mode texte du jeu.

Exemple d'interaction (ici, on joue avec 6 couleurs de 0 à 5) :

```
projet-mastermind$ python3 main_mastermind.py
Tour 1
Votre proposition ? [1,2,3,4]
La réponse est  0 bien placés et  1 mal placés.
Tour 2
Votre proposition ? [0,1,5,0]
La réponse est  1 bien placés et  2 mal placés.
Tour 3
Votre proposition ? [0,5,0,2]
La réponse est  1 bien placés et  1 mal placés.
Tour 4
Votre proposition ? [0,0,5,1]
La réponse est  0 bien placés et  3 mal placés.
Tour 5
Votre proposition ? [5,1,0,5]
La réponse est  4 bien placés et  0 mal placés.
Bravo, la partie est finie, vous avez gagné.
```

Vous devez programmer la classe `Proposition`. Elle contient un code (une liste de quatre valeurs), et propose un certain nombre de méthodes pour le manipuler :

- la méthode constructeur prend en paramètre une liste d'entiers qui représentera le code de la proposition.
- `calcule_bp_mp(self,code)` : cette méthode prend en paramètre un `code` numérique sous la forme d'une liste d'entiers et retourne le nombre de bien placés et le nombre de mal placés dans `code` par rapport à `self`. La spécification de cette méthode est `Proposition, list(int) -> int, int`. C'est la première fonction très importante pour cette première semaine (et même pour tout le projet). Nous reviendrons ensuite sur cette méthode.
- `__str__(self)` : une méthode pour afficher une `Proposition`.

Cette classe doit être compatible avec le code suivant :

```
>>> from mastermind import Proposition
>>> p = Proposition([1,2,3,4])
>>> print(p)
```

```
[1, 2, 3, 4]
>>> p.calcule_bp_mp([3,2,3,1])
(2, 1)
>>> print(p)
[1, 2, 3, 4]
>>> p = Proposition([1,1,3,4])
>>> p.calcule_bp_mp([3,2,3,1])
(1, 1)
```

Vous aurez besoin de calculer dans un premier temps les bien placés, puis dans un deuxième temps les mal placés. Attention : une valeur ne doit pas être utilisée deux fois. Dans le premier exemple ci-dessus, le `3` qui est dans `p` est utilisée pour compter un bien placé, mais ne doit pas être utilisé ensuite lorsqu'on va compter les mal placés. Dans le deuxième exemple ci-dessus, le dernier `1` du code passé en paramètre de la méthode `calcule_bp_mp` ne compte qu'une seule fois comme un mal placé. Il faut donc retirer au fur et à mesure les valeurs qui sont comptabilisées (en bien placé comme en mal placé). Pour cela, vous prendrez soin au début de votre méthode de faire en sorte de travailler sur des listes d'entier qui soient des copies du contenu de la proposition comme du code passé en paramètre.

Vous devrez également programmer la classe `Mastermind` qui contient la modélisation du jeu général. Elle permettra ultérieurement de gérer une partie où c'est l'ordinateur qui 'joue'. Pour le moment, la classe `Mastermind` contient :

- le constructeur qui prend en paramètre le nombre de couleurs (i.e. si le nombre de couleurs est `6`, alors les valeurs d'un code seront comprises entre `0` et `5`) et la taille d'un code (nombre de valeurs dans un code). Une valeur par défaut sera donnée pour ces deux paramètres. Le constructeur initialise également un attribut `__mystere` à `None`
- une méthode `dim(self)` qui retourne la taille du code
- une méthode `lancer(self)` qui génére aléatoirement un code secret. Ce code secret sera une `Proposition` qui sera mémorisé par l'attribut `__mystere`.
- une méthode `bp_mp(self,code)` dont la spécification est `Mastermind, list(int) -> int,int` et qui retourne le nombre de bien placés et de mal placés dans le code passé en paramètre par rapport au mystère du `Mastermind`.

Les deux classes `Proposition` et `Mastermind` que vous allez implémenter doivent être compatibles avec le code suivant pour le jeu en mode texte :

```
import mastermind

mm = mastermind.Mastermind()
## pour générer un code secret :
mm.lancer()
cpt = 1
bp, mp = 0, 0
while (cpt < 10) and (bp != mm.dim()) :
    print("Tour "+str(cpt))
    rep = eval(input("Votre proposition ? "))
    bp,mp = mm.bp_mp(rep)
    print("La réponse est ",bp,"bien placés et ",mp,"mal placés.")
    cpt += 1
## end of while
if (bp,mp) == (mm.dim(),0) :
    print("Bravo, la partie est finie, vous avez gagné.")
```