TP d'informatique bonus : jeu de la bataille navale

PCSI 2021 – 2022

Ce TP est un classique en informatique. Cet énoncé est fortement inspiré d'un énoncé de l'université Paris-Diderot.
Le but du TP est de programmer un jeu de la bataille navale (jeu à deux joueurs). On pourra ensuite faire jouer un humain contre l'ordinateur, ou un humain contre un autre humain, ou un ordinateur contre un autre ordinateur.
Les règles sont les suivantes :

- chaque joueur dispose d'une grille 10 x 10 (cachée à la vue de l'adversaire) ;
- les colonnes sont repérées par des lettres (A à J) et les lignes par des numéros (1 à 10) ;
- chaque joueur place 5 navires, horizontalement ou verticalement
  1. un porte-avions (5 cases)
  2. un croiseur (4 cases)
  3. un contre-torpilleurs (3 cases)
  4. un sous-marin (3 cases)
  5. un torpilleur (2 cases).

Le but de chaque joueur est de couler tous les bateaux de l'autre joueur. Chaque joueur joue tour à tour en proposant une position où lancer une torpille pour toucher un navire adverse en indiquant une position sur la grille (J2 par exemple) et l'adversaire répond *Touché* si la torpille touche un bateau et *Coulé* si l'adversaire touche un bateau et le coule (c'est-à-dire qu'il a touché toutes les cases du bateau correspondant) ou *À l'eau* si la torpille n'a rien touché ou a touché un emplacement d'un bateau déjà coulé.
Une grille sera encodée informatiquement par une liste de 10 listes de 10 entiers chacune avec le codage suivant :

- 0 indique qu'il n'y a pas de bateau
- un entier entre 1 et 5 indique qu'il y a un bateau (1 pour porte-avions, 2 pour croiseurs .... comme ci-dessus)
- 6 indique la présence d'un endroit où un bateau a déjà été touché

	A	B	C	D	E	F	G	H	I	J
1	0	6	3	3	0	0	4	0	0	0
2	0	0	1	0	0	0	4	0	0	0
3	0	0	1	0	0	0	4	2	0	0
4	0	0	1	5	5	0	0	2	0	0
5	0	0	1	0	0	0	0	2	0	0
6	0	0	1	0	0	0	0	2	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

Par exemple la grille ci-contre sera représenté par la liste de listes :

```
[[0, 6, 3, 3, 0, 0, 4, 0, 0, 0],
 [0, 0, 1, 0, 0, 0, 4, 0, 0, 0],
 [0, 0, 1, 0, 0, 0, 4, 2, 0, 0],
 ...                    ...]]
```

Le 6 en A6 indiquant que le contre-torpilleurs de ce joueur a déjà été touché.

Toutes les fonctions devront être dans le même fichier car il faudra les réutiliser. Pensez à vérifier qu'elles fonctionnent avant de passer à la suite.

Pour les tests, on pourra initialiser 2 grilles de la façon suivante :

```
grille_1 = [[0]*10 for i in range(10)]
grille_2 = [[0]*10 for i in range(10)]
```

Pour avoir les bateaux pour la suite, on pourra initialiser une variable globale contenant pour chaque navire son nom, son numéro ainsi que sa taille

```
liste_bateau = [["porte-avion",1,5],
                ["croiseur",2,4],
                ["contre-torpilleurs",3,3],
                ["sous-marin",4,3],
                ["torpilleur",5,2]]
```

Exercice n°1 Commencer par écrire une fonction `affiche_grille(grille)` qui réalise un bel affichage (attention à l'aligment)

```
   A B C D E F G H I J
1  0 3 3 3 0 0 4 0 0 0
2  0 0 1 0 0 0 4 0 0 0
3  0 0 1 0 0 0 4 2 0 0
4  0 0 1 5 5 0 0 2 0 0
5  0 0 1 0 0 0 0 2 0 0
6  0 0 1 0 0 0 0 2 0 0
7  0 0 0 0 0 0 0 0 0 0
8  0 0 0 0 0 0 0 0 0 0
9  0 0 0 0 0 0 0 0 0 0
10 0 0 0 0 0 0 0 0 0 0
```

Exercice n°2 Écrire une fonction `positionne(grille, ligne, col, sens, taille, num_bateau)` qui prend en argument une grille, un numéro de ligne (entier entre 0 et 9) un numéro de colonne (idem), un sens (une lettre "h" ou "v" pour horizontal ou vertical), une taille (nombre de case occupé, dépend du navire) et un numéro (1 pour porte-avion etc...).
Cette fonction doit renvoyer `True` s'il est possible de positionner le navire à la position donnée (les numéros de ligne et de colonne correspondant au point le plus "en haut à gauche") dans le sens souhaité et `False` sinon.
Dans le cas où il est possible de positionner le navire, la fonction doit modifier la grille pour ajouter les numéros correspondant au bateau
Remarque : il faut vérifier que le bateau ne sort pas de la grille et qu'il ne chevauche pas un autre bateau.

Exercice n°3 Écrire une fonction `initGrilleHasard(grille)` qui prend en argument une grille et qui la remplie aléatoirement.
Remarque : on pourra utiliser la fonction randrange du module random. Pensez à regarder la documentation et à tester
Remarque 2 : on pourra faire une boucle sur tous les navires de `liste_bateau`.
Remarque 3 : n'oubliez pas de tester grâce à votre affichage !

Exercice n°4 Écrire une fonction `initGrilleSurDemande(grille)` qui prend en argument une grille et qui la fait remplir par l'utilisateur (grâce à input).
Pour chaque bateau, on demande à l'utilisateur de saisir d'abord la lettre, puis le nombre, puis la direction (1 pour horizontal, 2 pour vertical), en répétant la question si l'utilisateur saisit une réponse incompréhensible (par exemple une lettre qui n'est pas comprise entre A et J, ou un mauvais nombre pour la ligne ou la direction).

Tant que l'utilisateur répond de façon incompréhensible ou que le bateau ne peut pas être placé à cet endroit on lui repose la question.
N'oubliez pas avant d'utiliser la fonction positionne de convertir l'entrée de l'utilisateur en "indice python", c'est-à-dire un nombre compris entre 0 et 9.
Une exécution de cette fonction peut avoir la forme suivante :

```
positionnement de votre croiseur de longueur 4
(rappel : les positions entrées correspondent au point en haut à gauche
    de votre bateau)

entrez la ligne souhaitée (nombre entre 1 et 10)
9
entrez la lettre correspondant à la colonne souhaitée (entre A et J)
J
entrez le sens souhaité (h pour horizontal et v pour vertical)
h
non, cela ne fonctionne pas, veuillez recommencer
entrez la ligne souhaitée (nombre entre 1 et 10)
```

Exercice n°5 Écrire une fonction `est_coule(num_bateau,grille)` qui prend en argument un numéro de bateau et une grille et qui renvoie `True` si le bateau est coulé (c'est-à-dire que son numéro n'apparait plus dans la grille) et `False` sinon.

Exercice n°6 Écrire une fonction `attaque(grille, ligne, col)` qui prend en argument une grille, un numéro de colonne et qui affiche les messages "Touché", "Coulé" ou "À l'eau" correspondant au résultat d'une attaque à la torpille sur la position repérée par le numéro de ligne et de colonne.
La fonction devra aussi le cas échéant mettre la grille à jour et indiquer quel bateau a été touché/coulé.

Exercice n°7 Écrire une fonction `a_perdu(grille)` qui prend en argument une grille et qui renvoie `True` si le joueur possédant la grille a perdu et `False` sinon.

Exercice n°8 Écrire une fonction `joue_ordi_idiot()` qui renvoie au hasard un numéro de ligne entre 0 et 9 et un numéro de colonne entre 0 et 9
Remarque : cette fonction pourra être améliorée ultérieurement pour prendre en compte les coups déjà joué

Exercice n°9 Programmer un jeu d'un ordinateur contre un autre

Exercice n°10 Écrire une fonction `joue_humain()` qui interragit avec un utilisateur pour lui demander en quel case il veut jouer et qui renvoie le numéro de ligne et de colonne en "convention python".

Exercice n°11 Programmer un jeu d'un ordinateur contre un humain

Exercice n°12 Programmer un jeu d'un humain contre un deuxième humain. Il faudra prendre soin de "masquer" les grilles précédentes.

Exercice n°13 Améliorer « l'intelligence artificielle » de votre ordinateur en tenant compte des coups précédemments joués (ainsi que des résultats de chaque coup). On modifiera la fonction `joue_ordi_idiot()` en `joue_ordi_malin(memoire)` ou memoire est une liste de deux liste :

1. la liste des positions déjà jouées ;
2. la liste des résultats correspondant aux positions déjà joué.

Par exemple si mémoire vaut `[[ [1,1], [1,2]],[0, 1]` cela voudrait dire qu'on a déjà joué en position `[1,1]` mais que l'on n'a rien touhé puis en `[1,2]` et que l'on a touché le porte-avions.
On pourra déjà commencé par ne pas rejouer des coups déjà joués !

Exercice n°14 Comparer vos différentes intelligences artificielles en les faisant se battre les unes contre les autres et en faisant des statistiques de résultats.

Exercice n°15 Envoyez moi l'intelligence artificielles la plus forte parmi celles que vous avez codées pour que j'organiser un tournoi contre les autres étudiants.