Conseils pour la mise en place des scripts pour puissance 4

Tout ce qui est énoncé dans ce documents ne sont que des conseils et n'ont pas de caractère obligatoire. Vous pouvez faire d'autres choix, mais je serai moins en mesure de vous aider.
Toutes les fonctions intermédiaires devront être testée avec une batterie de tests étendue avant d'être validée.

1 – Choix d'implémentation du jeu, premières fonctions

● J'ai choisi de représenter le jeu (la grille) sous forme d'une liste de 7 listes de longueurs 6 représentant les 7 colonnes et les 8 lignes. Je code :

- 0 pour une case pas encore occupée
- 1 pour une case occupée par un jeton du joueur 1, 2 pour une case occupée par l'autre joueur.

On prendra garde que `jeu[j][i]` *désigne la case de la j-ième colonne et de la i-ième ligne en partant du bas. Ces conventions ne sont pas les conventions usuelles des matrices.*

● J'ai écrit une fonction `affiche(jeu)` qui permet l'affichage (par un `print`) d'une grille (notée ici *jeu*).

Ainsi si *jeu* désigne la liste de liste ci-dessous :

```
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0], [2, 2, 2, 0, 0, 0], [1, 0, 0, 0, 0, 0], [1, 2, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

Alors la commande `affiche(jeu)` déclenche l'affichage ci-dessous :

```
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |
|   |   |   | 2 |   | 1 |   |
|   |   |   | 2 |   | 2 |   |
|   |   | 1 | 2 | 1 | 1 |   |
```

● Une fonction `joueCoup(jeu,colonne,joueur)` qui permet la mise à jour du plateau de jeu après que le joueur identifié par son numéro (1 ou 2) ait joué dans la colonne mentionnée.

Ainsi si à partir de la grille ci-dessus, nommée *jeu*, on tape la commande `joueCoup(jeu,3,1)`, alors *jeu* est modifié et `affiche(jeu)` montre la grille :

```
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |
|   |   |   | 1 |   |   |   |
|   |   |   | 2 |   | 1 |   |
|   |   |   | 2 |   | 2 |   |
|   |   | 1 | 2 | 1 | 1 |   |
```

● Une fonction `init()`, sans argument, qui crée une grille vide.

En écrivant prioritairement ces fonctions, vous pouvez vous faire facilement une liste de grilles tests pour tester les fonctions ultérieures.

2 – Fonctions de recherche de fin de partie

Pour les fonctions ci-dessous, ne pas hésiter à utiliser la commande `break` *qui permet d'interrompre une boucle sans pour autant quitter la fonction*.

● Une fonction `ligneGagnante(jeu)` qui renvoie 1 s'il y a une ligne gagnante pour le joueur 1, 2 s'il y en a une pour le joueur 2, et 0 sinon.

● Sur le même principe, on écrit une fonction `colonneGagnante`, `diagonaleMontanteGagnante`, `diagonaleDescendanteGagnante`.
● Une fonction `vainqueur(jeu)` qui renvoie 1 si la grille (appelée *jeu* en entrée) est gagnante pour le joueur 1, 2 si elle est gagnante pour le joueur 2, et 0 si elle n'est pas gagnante pour aucun des deux joueurs.

3 – Fonctions pour une simulation et une heuristique

● Une fonction `coupsPossibles(jeu)` qui détermine la liste des colonnes non pleines, dans laquelle il est possible de jouer.
● Une fonction `simulation(jeu,joueur)` : déclenche une simulation à partir de la position donnée par *jeu*, tandis que *joueur* vaut 1 ou 2 et désigne le premier joueur à jouer.
Pour ce faire, l'ordinateur choisit alternativement l'un des coups possibles pour chacun des joueurs et s'arrête soit en cas de victoire de l'un des deux joueurs, soit quand la grille est pleine sans victoire d'aucun des deux joueurs. La fonction renvoie 0 en cas de match nul, 1 ou 2 sinon selon le joueur vainqueur.
Attention : il ne faut pas que la grille *jeu* passée en entrée soit modifiée. On utilisera `deepcopy` du module `copy` pour en faire une copie de travail.
On rappelle la commande `randint(a,b)`, du module `random` qui permet de choisir aléatoirement un entier entre *a* et *b* inclus.
● Une fonction `evalue(jeu,joueur,nbSimulations)` : fonction qui déclenche le nombre de simulations indiqué, depuis la grille indiquée, le joueur indiqué étant le premier à jouer. On renvoie un triplet ou une liste (*nbNulles, nbVictoiresJoueur1, nbVictoiresJoueur2)*