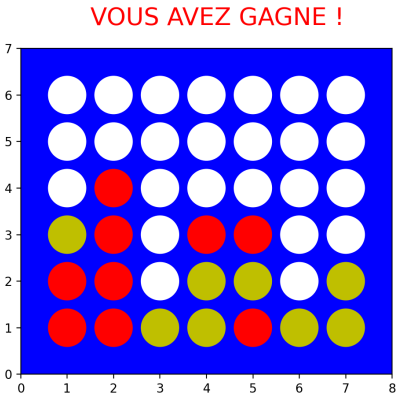
TP : Puissance 4
Informatique pour tous

Le *Puissance 4* est un jeu bien connu à deux joueurs, qui se joue sur une grille verticale de six lignes et sept colonnes. À tour de rôle, chaque joueur fait tomber un pion de sa couleur dans une colonne de son choix non encore pleine. Le premier joueur qui aligne quatre pions de sa couleur, horizontalement, verticalement ou en diagonal, gagne la partie. La partie est nulle si la grille est totalement remplie sans qu'aucun joueur ne gagne.



Pour représenter une grille de jeu, on utilise un tableau à deux dimensions de taille 6×7, la première dimension représentant les lignes et la seconde les colonnes. Une ligne est notée `l` et prend une valeur entre `0` et `5`, la ligne `0` étant située **en bas**. Une colonne est notée `c` et prend une valeur entre `0` et `6`, la colonne `0` étant située à gauche. Un joueur est noté `j` et prend la valeur `1` ou `2`. Dans une grille, notée `g`, la valeur `0` représente une case vide et la valeur `1` ou `2` représente un pion du joueur correspondant.

Remarque : On représentera la grille de jeu `g` par une liste de listes (i.e. une liste dont chaque élément est une liste représentant une ligne). On accède alors à la case `(l, c)` via l'instruction `g[l][c]`.

1. Écrire une fonction `grille_vide()` qui renvoie une grille vide, c'est-à-dire constituée uniquement de `0`. *Indication :* Pour définir une liste à deux dimensions dont les lignes sont *indépendantes*, il est conseillé de relire la section 5.1 du chapitre 7 sur les listes.

2. Écrire une fonction `affiche(g)` qui *affiche* une grille, avec le caractère `"*"` pour une case vide, le caractère `"X"` pour le joueur 1 et le caractère `"O"` pour le joueur 2. On prendra soin de bien afficher les lignes de haut en bas.

   Remarque : Pour éviter de passer à la ligne à chaque utilisation de `print`, on rappelle que l'on peut préciser le séparateur de fin comme suit : `print("*", end = " ")`. Par défaut, ce séparateur correspond à la chaîne de caractères `"\n"` (retour à la ligne). L'instruction `print()` réalise un retour à la ligne.

3. Écrire une fonction `coup_possible(g, c)` qui renvoie un booléen indiquant s'il est possible de jouer dans la colonne `c`.

4. Écrire une fonction `jouer(g, j, c)` qui joue un coup du joueur `j` dans la colonne `c`, en supposant que la colonne `c` n'est pas pleine.

   Remarque : La fonction doit directement modifier la grille `g` (l'instruction `return g` n'est donc pas nécessaire à la fin de la fonction).

5. Écrire quatre fonctions `horiz(g, j, l, c)`, `vert(g, j, l, c)`, `diag_haut(g, j, l, c)` et `diag_bas(g, j, l, c)` qui déterminent s'il y a un alignement horizontal, vertical, suivant une diagonale « montante vers la droite » ou une diagonale « descendante vers la droite » de quatre pions du joueur `j` à partir de la case `(l, c)`.

   Remarque : Certains alignements ne sont trivialement pas possibles pour certaines valeurs du couple `(l, c)`. On pourra tenir compte de ces impossibilités dans les fonctions précédentes ou bien dans la fonction ci-dessous, au choix.

6. Écrire une fonction `victoire(g, j)` qui renvoie un booléen indiquant si le joueur `j` a gagné.

7. Écrire une fonction `match_nul(g)` qui renvoie un booléen indiquant s'il y a match nul lorsque la grille est totalement remplie. *Indication :* On peut se contenter d'examiner la ligne du haut !

8. Écrire une fonction `coup_aleatoire(g, j)` qui joue un coup aléatoire autorisé pour le joueur `j`, en supposant que la grille n'est pas pleine. *Indication :* On rappelle qu'on peut générer un entier aléatoire compris entre un entier `a` et un entier `b` inclus à l'aide de l'instruction `randint(a, b)` de la bibliothèque `random` (attention à ne pas oublier d'importer la bibliothèque : `import random as rd`).

9. Écrire un programme qui fait jouer deux adversaires aléatoirement à tour de rôle, en affichant la grille après chaque coup, et qui s'arrête dès qu'un joueur gagne ou que la partie est nulle.

   Remarque : Pour lancer le jeu, il suffira de taper `partie_aleatoire()` dans la console. De même pour la fonction réalisée à la question suivante.

10. Écrire enfin un programme qui permet à l'utilisateur de jouer contre l'ordinateur jouant aléatoirement, en affichant la grille après chaque coup et le résultat en fin de partie.

Remarque : Dans le fichier `.py` à compléter se trouve une fonction `afficheJoli(g)` qui réalise un affichage de la grille plus agréable (voir image en début d'énoncé). Si vous souhaitez l'utiliser, il suffit de remplacer les appels à `affiche` par des appels à `afficheJoli`.
Attention : Cette fonction d'affichage fait appel à des fonctionnalités proposées par la bibliothèque `matplotlib`. Si cette bibliothèque n'est pas déjà installée sur votre machine, tapez `pip install matplotlib` dans la console de Pyzo/Spyder. Normalement, ce dernier devrait alors chercher les paquets d'installation sur Internet et les installer automatiquement au bon endroit (sans avoir à faire quoi que ce soit d'autre).