Projet Python – Jeu du Morpion

Le jeu du Morpion

Les deux joueurs placent à tour de rôle un symbole (une croix pour le joueur 1 et un cercle pour le joueur 2) dans une grille de 3 cases par 3 cases.
Le premier qui arrive à aligner 3 des ses symboles, sur une ligne, une colonne ou une diagonale, a gagné la partie.
Si aucun des deux joueurs n'a réussi à aligner 3 symboles identiques, il y a un match nul.

Consignes

Vous devez appeler votre fichier `projet_morpion_NOMS.py` et éventuellement rajouter un numéro de version si nécessaire.
Vous pouvez rajouter en commentaires les tests que vous avez réalisés.
À la fin de la séance, copier votre fichier dans le dossier `devoir` de votre groupe.
Une partie des exercices sont à réaliser sur la feuille à rendre. Ils ont pour but de vous aider à faire les exercices suivants.

Création et affichage de la grille de jeu

La grille du jeu sera représentée par un tableau à 2 dimensions de 3 tableaux de 3 cases chacunes. Les cases vides seront représentées par `'.'` et les cases jouées par `'x'` et `'o'`. Pour cela, il faut rajouter les lignes suivantes au début de votre fichier :

```
import random

SYMBOLES = {1: 'x', 2: 'o'}

def creer_grille():
    return [['.', '.', '.'], ['.', '.', '.'], ['.', '.', '.']]
```

Le dictionnaire `SYMBOLES` permet de modifier les symboles de chaque joueur.
La fonction `creer_grille()` renvoie une grille "vide".
Lors de l'affichage, il faut indiquer les cases disponibles. Pour cela, on numérotera les cases de 0 à 8.

Exercice 1 : Faire l'exercice sur la feuille à rendre.

Exercice 2 : Compléter la fonction `afficher(grille)` permet d'afficher la grille.

```
def afficher(grille):
    ligne_vide = "+-+-+-+"
    print(ligne_vide)
    for i in range(...):
        ligne = "|"
        for j in range(...):
            if grille[i][j] == ".":
                case = str(...) # la formule qui donne le numéro en fonction de i et j
            else:
                case = ... # On met le symbole qui est dans la grille
            ligne += case + "|"
        print(...)
        print(...)
```

```
>>> afficher([['.', '.', '.'], ['.', '.', '.'], ['.', '.', '.']])
+-+-+-+
|0|1|2|
+-+-+-+
|3|4|5|
+-+-+-+
|6|7|8|
+-+-+-+
```

```
>>> afficher([['x', 'o', '.'], ['.', 'x', '.'], ['.', '.', 'o']])
+-+-+-+
|x|o|2|
+-+-+-+
|3|x|5|
+-+-+-+
|6|7|o|
+-+-+-+
```

Lorsqu'un des adversaires joue, il donnera le numéro de la case et il faudra convertir ce nombre en numéro de ligne et numéro colonne.

Tour de jeu

Afin de permettre au joueur de choisir son premier coup, il faut qu'il puisse indiquer la case sur laquelle il souhaite jouer. Il faut pour cela pouvoir tester la validité du coup.

Exercice 3 : Écrire une fonction `libre(grille, i, j)` qui renvoie un booléen correspondant au fait que la case de la grille sur la ligne `i` et la colonne `j` est libre.

```
>>> libre([['x', 'o', '.'], ['.', 'x', '.'], ['.', '.', 'o']], 0, 0)
False
>>> libre([['x', 'o', '.'], ['.', 'x', '.'], ['.', '.', 'o']], 0, 2)
True
```

Exercice 4 : Faire l'exercice sur la feuille à rendre.

Exercice 5 : Compléter La fonction `demander(grille)` qui demande à l'utilisateur la case sur laquelle il veut jouer et renvoie les coordonnées `i` et `j` de la case. Si la réponse n'est pas valide, on lui redemande une autre case, jusqu'à ce qu'il donne une réponse valide.

```
def demander(grille):
    n = -1 # Pour être sûr de rentrer dans la boucle
    i, j = 0, 0 # On prend une case arbitraire
    while ...: # n est trop petit ou trop grand ou la case n'est pas libre
        n = int(input("Donner une position libre : "))
        ... = ... % 3 # Le numéro de colonne
        ... = ... // 3 # le numéro de ligne
    return i, j
```

```
>>> demander([['x', 'o', '.'], ['.', 'x', '.'], ['.', '.', 'o']])
Donner une position libre : 1
Donner une position libre : 2
(0, 2)
```

Test de fin de partie

Après le coup de chaque joueur, il faut tester si le coup a permis de gagner. Pour cela, nous allons des fonctions permettant de tester les lignes, les colonnes et les diagonales.

Exercice 6 : Faire l'exercice sur la feuille à rendre.

Exercice 7 : Compléter la fonction `tester_ligne(grille, i)` qui renvoie un booléen indiquant si la ligne `i` de la grille est composée des 3 mêmes symboles, différents de `'.'`.

```
def tester_ligne(grille, i):
    return (not libre(grille, i, 0)
            and grille[i][0] == grille[...][...]
            and grille[...][...] == grille[...][...])
```

```
>>> tester_ligne([['x', 'o', '.'], ['.', '.', '.'], ['o', 'o', 'o']], 1)
False
>>> tester_ligne([['x', 'o', '.'], ['.', '.', '.'], ['o', 'o', 'o']], 0)
False
>>> tester_ligne([['x', 'o', '.'], ['.', '.', '.'], ['o', 'o', 'o']], 2)
True
```

Exercice 8 : Faire l'exercice sur la feuille à rendre.

Exercice 9 : Écrire la fonction `tester_colonne(grille, j)` qui renvoie un booléen indiquant si la colonne `j` de la grille est composée des 3 mêmes symboles, différents de `'.'`.

```
>>> tester_colonne([['x', '.', 'x'], ['.', '.', 'x'], ['o', '.', 'x']], 0)
False
>>> tester_colonne([['x', '.', 'x'], ['.', '.', 'x'], ['o', '.', 'x']], 1)
False
>>> tester_colonne([['x', '.', 'x'], ['.', '.', 'x'], ['o', '.', 'x']], 2)
True
```

Exercice 10 : Faire l'exercice sur la feuille à rendre.

Exercice 11 : Écrire La fonction `tester_diagonale1(grille)` qui renvoie un booléen indiquant si la diagonale allant de la case 0 à la case 8 est composée des 3 mêmes symboles, différents de `'.'`.

```
>>> tester_diagonale1([['x', 'o', '.'], ['.', 'x', '.'], ['o', 'o', 'o']])
False
>>> tester_diagonale1([['x', 'o', '.'], ['.', 'x', '.'], ['o', 'o', 'x']])
True
>>> tester_diagonale1([['x', 'o', 'o'], ['.', 'o', '.'], ['o', 'o', 'x']])
False
```

Exercice 12 : Faire l'exercice sur la feuille à rendre.

Exercice 13 : Écrire La fonction `tester_diagonale2(grille)` qui renvoie un booléen indiquant si la diagonale allant de la case 2 à la case 6 est composée des 3 mêmes symboles, différents de `'.'`.

```
>>> tester_diagonale2([['x', 'o', '.'], ['.', 'x', '.'], ['o', 'o', 'o']])
False
>>> tester_diagonale2([['x', 'o', '.'], ['.', 'x', '.'], ['o', 'o', 'x']])
False
>>> tester_diagonale2([['x', 'o', 'o'], ['.', 'o', '.'], ['o', 'o', 'x']])
True
```

Exercice 14 : Compléter la fonction `tester_case(grille, i, j)` qui renvoie un booléen indiquant si la case de coordonnées `i` et `j` a permis de gagner la partie.
On ne testera les diagonales que si c'est nécessaire.

```
def tester_case(grille, i, j):
    if ...: # on teste la ligne, ainsi que la colonne
        return True
    elif ... and tester_diagonale1(grille):
        return True
    elif ... and tester_diagonale2(grille):
        return True
    else:
        return ...
```

```
>>> tester_case([['x', 'o', '.'], ['.', 'o', '.'], ['o', 'o', 'x']], 0, 0)
False
>>> tester_case([['x', 'o', '.'], ['.', 'o', '.'], ['o', 'o', 'x']], 0, 1)
True
```

Une partie complète

Exercice 15 : Faire l'exercice sur la feuille à rendre.

Exercice 16 : Compléter la fonction `partie()` qui permet de faire une partie jusqu'à ce qu'il y ait un gagnant ou match-nul.

```
def partie():
    joueur = 2
    grille = creer_grille()
    nb_coups = 0
    gagnant = 0
    while ...: # condition pour continuer
        joueur = ... - joueur # On change en début de partie
        print()
        print("**********")
        print(f"Au tour du joueur {joueur} ({SYMBOLES[joueur]})")
        print("**********")
        print()
        afficher(grille)
        print()
        i, j = demander(grille)
        grille[i][j] = ... # on place le symbole
        if ...: # On regarde si on a gagné
            gagnant = joueur
        nb_coups += 1
```

```
    print()
    print("Partie terminée")
    print()
    afficher(grille)
    print()
    if gagnant == ...:
        print("Match nul !")
    else:
        print(f"Bravo joueur {gagnant}")
```

```
>>> partie()

**********
Au tour du joueur 1 (X)
**********

+-+-+-+
|0|1|2|
+-+-+-+
|3|4|5|
+-+-+-+
|6|7|8|
+-+-+-+

Donner une position libre : 0

**********
Au tour du joueur 2 (O)
**********

+-+-+-+
|x|1|2|
+-+-+-+
|3|4|5|
+-+-+-+
|6|7|8|
+-+-+-+

Donner une position libre : 4

...

**********
Au tour du joueur 2 (O)
**********

+-+-+-+
|x|x|o|
+-+-+-+
|3|o|5|
+-+-+-+
```

```
|6|7|x|
+-+-+-+

Donner une position libre : 6
Bravo joueur 2
```

Jouer contre l'ordinateur

Nous allons maintenant rajouter un adversaire contrôlé par l'ordinateur. Il suivra les 3 principes suivants, dans l'ordre.

1. S'il peut jouer et gagner, il le fait.
2. Si l'autre joueur a une case sur laquelle il peut jouer et gagner, on la bloque.
3. Sinon, il joue au hasard.

Exercice 17 : Faire l'exercice sur la feuille à rendre.

Exercice 18 : Compléter la fonction `tester_coups_gagnants(grille, joueur)` qui cherche toutes les cases sur lesquelles le joueur `joueur` peut jouer et gagner. Elle renvoie la liste des cases permettant de gagner en un coup.

```
def tester_coups_gagnants(grille, joueur):
    liste_coups = []
    for i in range(...):
        for j in range(...):
            if libre(grille, i, j): # la case est libre
                grille[i][j] = ... # on met le symbole du joueur
                if ...: # On teste s'il a gagné
                    liste_coups.append((i, j)) # on mémorise la case
                grille[i][j] = ... # on remet la case vide
    return liste_coups
```

```
>>> tester_coups_gagnants([['o', '.', 'o'], ['x', 'o', 'x'], ['.', '.', '.']], 2)
[(0, 1), (2, 0), (2, 2)]
>>> tester_coups_gagnants([['o', '.', 'o'], ['x', 'o', 'x'], ['.', '.', '.']], 1)
[]
```

Exercice 19 : Compléter la fonction `ordinateur(grille, joueur)` qui permet à l'ordinateur de jouer selon la stratégie ci-dessus. Elle renvoie les coordonnées de la case choisie.

```
def ordinateur(grille, joueur):
    # On regarde si on peut gagner
    coups_gagnants = tester_coups_gagnants(grille, joueur)
    if len(coups_gagnants) > 0:
        return coups_gagnants[...] # On renvoie le premier coup gagnant
    # On regarde si on peut perdre
    coups_perdants = tester_coups_gagnants(grille, ...-joueur)
    if ...:
        return ... # on bloque le premier qu'on a trouvé
    # Sinon on choisit au hasard
    cases_libres = [] # On fait la liste des cases libres
    for i in range(...):
```

```
        for j in range(...):
            if ...:
                cases_libres.append((i, j))
    return random.choice(cases_libres) # On en prend une au hasard
```

```
>>> ordinateur([['o', '.', 'o'], ['x', 'o', 'x'], ['.', '.', '.']], 2)
(0, 1) # On joue pour gagner
>>> ordinateur([['o', '.', 'o'], ['x', 'o', 'x'], ['.', '.', '.']], 1)
(0, 1) # On bloque
>>> ordinateur([['o', '.', '.'], ['x', 'x', 'o'], ['.', '.', '.']], 1)
(2, 2) # Coup au hasard
>>> ordinateur([['o', '.', '.'], ['x', 'x', 'o'], ['.', '.', '.']], 1)
(0, 1) # Coup au hasard
```

Exercice 20 : Compléter la fonction `partie2()` qui permet de jouer contre l'ordinateur. On peut changer de type de joueurs en changeant les valeurs dans le dictionnaire `types`.

```
def partie2():
    joueur = 2
    grille = creer_grille()
    types = {1: "humain", 2: "ordi"} # on indique le type de chaque joueur
    nb_coups = 0
    gagnant = 0
    while ...:
        joueur = ... - joueur
        print()
        print("********************")
        print(f"Au tour du joueur {joueur} ({SYMBOLES[joueur]})")
        print("********************")
        print()
        afficher(grille)
        print()
        if types[joueur] == "humain":
            i, j = demander(grille)
        else:
            i, j = ordinateur(grille, joueur)
            print(f"Je joue en {...}.")
        grille[i][j] = ...
        if ...:
            gagnant = joueur
        nb_coups += 1
    print()
    print("Partie terminée")
    print()
    afficher(grille)
    print()
    if ...:
        print("Match nul !")
    else:
        print(f"Bravo joueur {joueur}")
```

Pour aller plus loin

Voici quelques exemples d'améliorations possibles. Vous pouvez en imaginer d'autres :

- Proposer de choisir le type des joueurs au début de la partie.
- Changer les messages en cas de victoire de l'ordinateur.
- Améliorer l'ordinateur en lui permettant de chercher sur quelle case il pourrait joueur pour pouvoir gagner au coup suivant. De préférence, il choisira une case lui offrant plusieurs possibilités de victoire.
- Ajouter la possibilité de jouer plusieurs parties, en comptant les points et en changeant le joueur qui commence. Par exemple le perdant peut commencer.
- Modifier le jeu pour qu'on puisse jouer sur une grille de taille variable (4×4 ou 5×5 par exemple). Attention, cela nécessite de modifier toutes les fonctions du projet. Vous pourrez utiliser une variable globale `TAILLE`.