Projet réseau

Ce projet **individuel** consiste à implémenter en `python` un jeu de morpion, jouable dans le terminal en réseau.

Consignes pour le rendu

L'implémentation est laissée libre : il n'y a donc pas de `doctest` à respecter.

Pour toutes vos fonctions, sauf indication contraire, vous devrez produire une **docstring** contenant au moins :

- la liste des arguments (avec leur type et s'il y a lieu leurs restrictions)
- le type de retour
- éventuellement une **doctest** contenant des tests appropriés.

Important ! Commentez de manière détaillée la partie réseau de votre projet. N'hésitez pas à faire des schémas explicatifs et une description sur des échanges de messages (ordre, format, etc.) dans un fichier annexe.

Vous déposerez sur moodle dans l'espace dédié au projet :

- Un fichier `client_morpion.py`.
- Un fichier `serveur_morpion.py`.
- (éventuellement) Un fichier d'explication de la partie réseau de votre projet.

au plus tard pour le :

Dimanche 30 avril 2023

Attention à :

- bien rendre 2 (ou 3) fichiers et non pas un dossier compressé,
- bien tester votre projet avant de le rendre,
- bien commenter votre code surtout sur la partie réseau.

Avertissement sur le travail à plusieurs

Ce projet est à rendre individuellement. Vous pouvez vous entraidez sur ce projet, **mais tout le code que vous produisez doit être personnel !**

Il y aura des vérifications de similarité de code, et en cas de suspicion de fraude, je ferais un signalement à la commission disciplinaire de l'université [1].

1. voir ce lien pour davantage d'informations

Morpion

Le morpion est un jeu à 2 joueurs sans hasard, dans lequel les joueurs placent tour à tour des symboles dans une grille (d'une vingtaine de lignes et d'une vingtaine de colonnes).

La partie se termine lorsque l'un des joueurs parvient à aligner 5 symboles (verticalement, horizontalement ou en diagonale) : dans ce cas, il remporte la partie ; ou que la grille est complétée sans qu'il y ait de vainqueur : il y a alors match nul. [2]

L'objectif de ce projet est d'implémenter ce jeu avec un affichage dans le terminal (on n'utilisera pas d'affichage graphique) et une connexion réseau.

Cahier des charges

Spécifications générales

- Votre projet utilisera un modèle client-serveur, dans lequel le serveur s'occupe :
  - ◇ de gérer l'enregistrement des 2 clients,
  - ◇ de la mécanique de jeu (tour à tour, vérification du gain),
  - ◇ de l'envoi des messages pour récupérer les coups à jouer ou afficher l'état de la grille,
  - ◇ de l'envoi du message de fin de partie.

  Le rôle des clients est d'afficher les messages reçus par le serveur et d'envoyer les messages saisis au clavier.
- Le protocole de transport utilisé pour échanger des messages sera UDP sur le port 5005 (sauf s'il est déjà utilisé par une autre application).
- Vous êtes libre de choisir les dimensions de la grille (dans l'exemple j'ai choisi 15×15).
- On ne demande pas à ce que les clients se ferment automatiquement en fin de partie.

Spécifications du programme

- Lors du lancement du client, il doit demander son nom à l'utilisateur et l'envoyer au serveur. Le serveur lui renvoie le message :
  `Bienvenue #nom_du_joueur# !`
- À la fin de la partie, le serveur doit envoyer le message :
  `#nom_du_joueur_gagnant# remporte la partie !`
  `Le jeu est terminé ! (Appuyez sur Ctrl+C pour quitter)`
  ou
  `La grille est pleine : match nul !`
  `Le jeu est terminé ! (Appuyez sur Ctrl+C pour quitter)`
- Le serveur choisit aléatoirement le premier joueur parmi les deux enregistrés.
- Au cours de la partie :
  - ◇ le serveur envoie la représentation de la grille aux deux joueurs.
  - ◇ le serveur envoie **seulement** au joueur courant le message :
    `Entrez votre prochain coup :`

2. Pour une description plus complète du jeu, voir https://fr.wikipedia.org/wiki/Morpion_(jeu)

- ◇ si le serveur reçoit un message de la part d'un autre joueur, il lui renvoie :
  `Ce n'est pas à vous de jouer !`
- ◇ si le serveur reçoit du joueur courant un coup invalide (case inexistante, case déjà utilisée), il lui renvoie :
  `"Coup Invalide !"`

Conseils généraux

Il est **indispensable** que vous ayez un projet qui compile et s'exécute sans problème.

Aussi, je vous conseille de procéder au développement de votre projet en plusieurs phases.

Phase 1 : Codage du morpion sans réseau Vous pouvez reprendre le code de votre projet d'algorithmique en adaptant les règles et la représentation.

Phase 2 : Codage des fonctionnalités réseau Vous pouvez reprendre et adapter le code de votre TP mini chat pour créer un client et serveur qui communiquent. Il faut adapter l'enregistrement des joueurs, les balises et le décodage des messages.

Phase 3 : Concevoir les échanges de messages Vous pouvez concevoir sur papier la suite des échanges de message d'une partie complète, en essayant de gérer tous les cas de figure : (coup valide, coup invalide, joueur invalide, fin de partie avec gain, fin de partie nulle, etc.). Cette conception peut vous servir d'explication de code pour le rendu final.

Phase 4 : Ajout du réseau dans le jeu Il faut adapter la boucle principale du jeu pour que la communication se fasse à travers le réseau et non plus via un seul terminal. Là encore, vous pouvez procéder par étape :

- étape 1 : Ne pas faire de vérification, considérer que les joueurs jouent uniquement à leur tour et envoient systématiquement un coup valide.
- étape 2 : Considérer que les joueurs peuvent essayer d'envoyer des coups non valides.
- étape 3 : Considérer que les joueurs peuvent essayer de jouer lors du tour de l'adversaire.

Exemple d'utilisation

Voir la vidéo sur moodle pour un exemple plus interactif.

Début de partie

```
$ python3 serveur_morpion.py
```

Serveur

```
$ python3 client_morpion.py
Enter your name : pr
Bienvenue pr !
 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . .
 5| . . . . . .
 6| . . . . . .
 7| . . . . . .
 8| . . . . . .
 9| . . . . . .
10| . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O

I7
Ce n'est pas à vous de jouer !
fmdsf
Ce n'est pas à vous de jouer !
 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . .
 5| . . . . . .
 6| . . . . . .
 7| . . . . . O . . . . . .
 8| . . . . . .
 9| . . . . . .
10| . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O

Entrez votre prochain coup :
```

Client 2

```
$ python3 client_morpion.py
Enter your name : as
Bienvenue as !
 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . .
 5| . . . . . .
 6| . . . . . .
 7| . . . . . .
 8| . . . . . .
 9| . . . . . .
10| . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O

Entrez votre prochain coup :
qsdmlfk
Coup Invalide !
Z12
Coup Invalide !
I7
 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . .
 5| . . . . . .
 6| . . . . . .
 7| . . . . . O . . . . . .
 8| . . . . . .
 9| . . . . . .
10| . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O
```

Client 1

Fin de partie

```
$ python3 serveur_morpion.py
$
```

Serveur

```
$ python3 client_morpion.py
Enter your name : pr
Bienvenue pr !

...

 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . X X X . . . . .
 5| . . . . . X O . . . . . .
 6| . . . . . O . . . . . .
 7| . . . . . O . . . . . .
 8| . . . . . O . . . . . .
 9| . . . . . O . . . . . .
10| . . . . . X . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O

as remporte la partie !
Le jeu est terminé ! (Appuyez sur
Ctrl+C pour quitter)
```

Client 2

```
$ python3 client_morpion.py
Enter your name : as
Bienvenue as !

...

 0| . . . . . .
 1| . . . . . .
 2| . . . . . .
 3| . . . . . .
 4| . . . . . X X X . . . . .
 5| . . . . . X O . . . . . .
 6| . . . . . O . . . . . .
 7| . . . . . O . . . . . .
 8| . . . . . O . . . . . .
 9| . . . . . O . . . . . .
10| . . . . . X . . . . . .
11| . . . . . .
12| . . . . . .
13| . . . . . .
14| . . . . . .
   ----------------------------
    A B C D E F G H I J K L M N O

as remporte la partie !
Le jeu est terminé ! (Appuyez sur
Ctrl+C pour quitter)
```

Client 1

Proposition de résolution

La méthode qui suit n'est qu'une ***proposition*** *de résolution, vous êtes libre de la suivre entièrement, partiellement ou pas du tout.*

Il est préférable de séparer la partie gestion du jeu et la partie réseau du projet (même si tout est contenu dans un seul fichier).

Modélisation

- Vous pouvez modéliser les joueurs par un couple (adresse IP, port UDP), stocker dans un dictionnaire `joueurs` la correspondance entre un joueur et son type de symbole (`'X'` ou `'O'`), et stocker dans un autre dictionnaire `names` la correspondance entre un joueur et son nom.

- Vous pouvez représenter la grille de jeu par un tableau de tableau contenant soit le couple (adresse IP, port UDP) correspondant au joueur, soit une valeur arbitraire pour une case vide (par exemple 0)
- Vous pouvez coder deux fonctions :
  - ◇ `choisir_premier_joueur` qui permet de choisir un joueur parmi les deux enregistrés.
  - ◇ `joueur_suivant` qui permet de retourner l'autre joueur que celui placé en paramètre.

Gestion du jeu Vous pouvez réutiliser (en adaptant pour le jeu du morpion) le code que vous aviez produit pour le puissance 4 du projet d'algorithmique, en adaptant les fonctions :

- `generer_grille_vide`
- `peut_jouer`
- `joue`
- `a_gagne_vert`, `a_gagne_hor`,`a_gagne_diag1`,`a_gagne_diag2`, `a_gagne`
- `grille_pleine`
- `get_grille_representation`

Mise en réseau

- Vous pouvez vous inspirer de ce que vous avez fait pour le TP mini chat sur la partie client (avec un *sender* et un *receiver*).
- Vous pouvez définir 3 balises de message pour :
  - ◇ les messages d'enregistrement de joueurs
  - ◇ les messages d'envoi de coup
  - ◇ les messages de fin de partie
- Au niveau du serveur :
  - ◇ Vous pouvez définir une fonction `enregistre_joueur` qui permet :
    - ■ d'enregistrer le nom d'un joueur,
    - ■ de lui affecter un symbole (parmi `'X'` et `'O'`),
    - ■ de lui envoyer un message de bienvenue.

    Elle vérifiera également que le joueur n'est pas déjà inscrit pour la partie.
- Vous pouvez définir une fonction `decode_coup` prenant en paramètre un message reçu par le serveur et retournant le coup sous un format utilisable.
- Vous pouvez coder la mécanique de votre jeu dans une fonction `boucle_principale` reprenant le principe de celle utilisée dans le puissance 4, mais en adaptant la récupération du prochain coup (avec la vérification que le coup envoyé provient bien du joueur courant, que la syntaxe est correcte et que le coup est valide).
- enfin, votre programme principal :
  - ◇ enregistre deux joueurs
  - ◇ lance la boucle principale