SA1.01 Implémentation
Sujet n°2
CyberAttack@IUT'O
Information pour la semaine de projet

IMPORTANT

Les instructions données dans ce document sont essentielles pour pouvoir participer au tournoi de la fin de semaine. Prenez soin de bien suivre les instructions de la section suivante afin de ne pas perdre de temps par la suite !

1 Mise à jour des fichiers Python

Afin de pouvoir participer au tournoi quelques modifications sont à apporter dans les fichiers `joueur.py` `plateau.py` et `jeu.py` . De même une nouvelle version de l'affichage est disponible.

1.1 `joueur.py`

Ce qui change

ligne 108 ajout d'une nouvelle fonction à implémenter qui permet de changer le nom du joueur

Ajoutez cette fonction à votre implémentation de `joueur.py`

1.2 `plateau.py`

Ce qui change

- **ligne 18** ajout de constantes permettant de d'indiquer les points gagnés et perdus pour chaque action. Ce sera le barème utilisé lors du tournoi.
- **ligne 262**, la docstring de `poser_protection()` a été changée. Une protection ne peut pas être poser sur une case qui contient déjà autre chose (trojans, serveur, avatar ou protection).
- **ligne 493-778** modifications mineures sur les fonctions de sauvegarde et restauration d'un plateau. La modification porte sur l'assouplissement des contrôles sur la resistance des serveurs et des ordinateurs
- **ligne 701** ajout de la fonction à implémenter `set_nom_joueur` qui permet de change le nom du joueur à qui appartient le plateau.

Si vous n'avez pas encore travaillé sur ce fichier, veuillez prendre la nouvelle version comme base à votre implémentation, sinon recopier les quatre éléments décrits ci-dessus depuis la nouvelle version `plateau.py` vers votre code.

1.3 `jeu.py`

Ce qui change

Tous les changements sauf un se situent à partir de la ligne 121

- **ligne 14** les valeurs par défaut de `resistance_serveur` et `resistance_pc` passent à `40` et `50` respectivement. Ce sont ces valeurs qui seront utilisées pour le tournoi.
- Modifications mineures dans les fonctions `joueur_aleatoire()` , `joueur_humain()` , `action_joueur()` et `action_joueur_ext()` afin de les rendre compatibles avec votre implémentation.
- Ajout des fonction de sauvegarde et de restauration d'un jeu. Ces fonctions sont essentielles pour le tournoi.
- Ajout des fonction à implémenter à partir de la ligne 303. Ces fonctions sont essentielles pour le tournoi, notamment la fonction `joueur_ia()` qui sera celle appelée lors du tournoi pour faire jouer votre IA

Si vous n'aviez pas encore travaillé sur ce fichier, veuillez prendre la nouvelle version comme base à votre implémentation, sinon, il faut remplacer les anciennes version des fonctions `joueur_aleatoire()` , `joueur_humain()` , `action_joueur()` et `action_joueur_ext()` par les nouvelles et ajouter les nouvelles fonctions.

1.4 `client.py` et `client_joueur.py`

Ces deux scripts seront utilisés lors du tournoi. Vous n'avez pas à intervenir dedans mais vous pouvez voir dans `client_joueur.py` où sera appelée votre IA.

2 Données

Des sauvegarde de jeux ont été ajoutés dans le répertoire de données. Vous pourrez vérifier ainsi que la restauration d'un jeu fonctionne bien. Des images montrent ce qui devrait être affiché pour chaque exemple vous seront fournies rapidement.

3 Tournoi

Afin d'implémenter une IA pour le jeu, il vous faut implémenter la fonction `joueur_ia(le_jeu,id_joueur)` du module `jeu.py` Cette fonction prend en paramètres l'état du jeu (c'est-à-dire des 4 plateaux) et le numéro du joueur pour lequel vous devez prendre une décision. La fonction doit retourner une chaîne de caractères décrite dans la section 3.7 du sujet initial.

Le principe du jeu en réseau est le suivant :

- un serveur va s'occuper de la gestion du jeu au travers d'une boucle qui va :
  1. envoyer l'état du jeu aux joueurs (via le réseau)
  2. attendre les réponses des joueurs (via le réseau)
  3. exécuter les ordres donnés par les joueurs
  4. mettre à jour le jeu
- les joueurs seront des clients de ce serveur. Il exécute la boucle suivante :
  1. récupérer l'état du jeu (via le réseau)
  2. exécuter la fonction `joueur_ia()`
  3. envoyer le résultat de cette fonction au serveur (via le réseau)

Vous n'avez pas à vous occuper de la partie réseau mais notez que les communications entre les joueurs et le serveur se font via de chaînes de caractères. C'est la fonction `jeu_2_str()` qui va permettre au serveur d'encoder l'état du jeu en une chaîne de caractères et c'est la fonction `creer_jeu_from_str()` qui va transformer la chaîne reçue en un jeu compatible avec votre implémentation (voir ligne 110 de `client.py`).

📎 Importance de `creer_jeu_from_str()`

Pour être sûr que votre IA fonctionne il faut s'assurer que la fonction `creer_jeu_from_str()` retourne un jeu conforme à ce que vous attendez. N'hésitez pas à demander de l'aide pour finaliser cette implémentation si vous avez des difficultés.

4 Rendu Final

Le rendu final est à effectué le 21 janvier avant 10h. Vous devrez rendre une archive `zip` du répertoire `cyber_attack` contenant tous les fichiers Python du projet ainsi qu'un fichier `EQUIPE` qui contiendra une seule ligne indiquant le nom que vous avez choisi pour votre équipe. Vous ajouterez un petit document pdf (une page recto/verso) indiquant

1. l'état de votre projet (ce qui marche ce qui ne marche pas)
2. une explication succincte de la stratégie de votre IA

L'évaluation de la SAE se basera sur ce rendu ainsi que sur l'épreuve du 21 janvier qui portera sur

- Une analyse de ce que vous avez retenu de cette SAE,
- des questions techniques sur l'implémentation.

📎 Présence obligatoire

- On vous rappelle que votre présence est obligatoire pendant toute la semaine.
- M. Limet passera dans les salles de TP lundi matin à partir de 8h30 pour bien préciser les informations de ce document