SPÉCIFICATIONS TECHNIQUES

INTRODUCTION

L'application doit être un programme autonome et hors ligne. Le programme doit être écrit en Python et lancé depuis la console. En d'autres termes, l'exécution du programme devrait ressembler à ceci : python <nom du programme>.py. Le programme devrait fonctionner sous Windows, Mac ou Linux et avoir un fichier requirements.txt listant les dépendances nécessaires à son exécution.

PROGRAMME DES JOUEURS

Nous aimerions que l'application hors ligne contienne une base de données des joueurs dans des fichiers JSON. Le programme devrait avoir une section dédiée à l'ajout de joueurs et chaque joueur devrait contenir *au moins* les données suivantes :

- Nom de famille
- Prénom
- Date de naissance

IDENTIFIANT NATIONAL D'ÉCHECS

Les clubs d'échecs utilisent tous l'identifiant national d'échecs, géré par la Fédération des échecs. Ces identifiants sont uniques et comportent deux lettres suivies de cinq chiffres (par exemple, AB12345). Vous n'avez pas besoin de générer ces identifiants – les membres du club les connaissent !

Ne vous préoccupez pas de l'importation de nos anciennes données dans le nouveau programme. Nos managers peuvent entrer les joueurs manuellement lorsqu'ils s'inscrivent aux tournois. De plus, nous n'avons pas besoin de pouvoir supprimer des joueurs.

TOURNOIS

Le programme utilise les fichiers de données JSON pour la persistance des informations sur les tournois. Les fichiers de données sont généralement situés dans le dossier data/tournaments.

DÉROULEMENT DE BASE DU TOURNOI

- Un tournoi a un nombre de tours défini.
- Chaque tour est une liste de **matchs**.
    - Chaque match consiste en une **paire** de joueurs.
- À la fin du match, les joueurs reçoivent des points selon leurs résultats.
    - Le gagnant reçoit 1 point.
    - Le perdant reçoit 0 point.

    - Chaque joueur reçoit 0,5 point si le match se termine par un match nul.

SCHÉMA DES TOURNOIS

Chaque tournoi doit contenir au moins les informations suivantes :

- nom ;
- lieu ;
- date de début et de fin ;
- nombre de tours – réglez la valeur par défaut sur 4 ;
- numéro correspondant au tour actuel ;
- une liste des tours ;
- une liste des joueurs enregistrés ;
- description pour les remarques générales du directeur du tournoi.

TOURS / MATCHS

Un match unique doit être stocké sous la forme d'un tuple contenant deux listes, chacune contenant deux éléments : un **joueur** et un **score**. Les **matchs** doivent être stockés sous forme de liste dans l'instance du tour auquel ils appartiennent.

En plus de la liste des matchs, chaque instance du tour doit contenir un nom. Actuellement, nous appelons nos tours "Round 1", "Round 2", etc. Elle doit également contenir un champ Date et heure de début et un champ Date et heure de fin, qui doivent tous deux être automatiquement remplis lorsque l'utilisateur crée un tour et le marque comme terminé.

GÉNÉRATION DES PAIRES

- Au début du premier tour, mélangez tous les joueurs de façon aléatoire.
- Chaque tour est généré dynamiquement en fonction des résultats des joueurs dans le tournoi en cours.
    - Triez tous les joueurs en fonction de leur nombre total de points dans le tournoi.
    - Associez les joueurs dans l’ordre (le joueur 1 avec le joueur 2, le joueur 3 avec le joueur 4 et ainsi de suite.)
    - Si plusieurs joueurs ont le même nombre de points, vous pouvez les choisir de façon aléatoire.
    - Lors de la génération des paires, évitez de créer des matchs identiques (c’est-à-dire les mêmes joueurs jouant plusieurs fois l’un contre l’autre).
        - Par exemple, si le joueur 1 a déjà joué contre le joueur 2, associez-le plutôt au joueur 3.
- Mettez à jour les points de tous les joueurs après chaque tour et répétez le processus de triage et d’association jusqu'à ce que le tournoi soit terminé.
- Un tirage au sort des joueurs définira qui joue en blanc et qui joue en noir ; il n'est donc pas nécessaire de mettre en place un équilibrage des couleurs.

RAPPORTS

Nous aimerions pouvoir afficher les rapports suivants dans le programme :

- liste de tous les joueurs par ordre alphabétique ;
- liste de tous les tournois ;
- nom et dates d'un tournoi donné ;
- liste des joueurs du tournoi par ordre alphabétique ;
- liste de tous les tours du tournoi et de tous les matchs du tour.

Nous aimerions les exporter ultérieurement, mais ce n'est pas nécessaire pour l'instant. Les rapports peuvent être en texte brut, à condition qu'ils soient bien formatés et faciles à lire. Vous pouvez même utiliser des modèles HTML !

SAUVEGARDE / CHARGEMENT DES DONNÉES

Nous devons pouvoir sauvegarder et charger l'état du programme à tout moment entre deux actions de l'utilisateur. Plus tard, nous aimerions utiliser une base de données, mais pour l'instant nous utilisons des fichiers JSON pour garder les choses simples.

Les fichiers JSON doivent être mis à jour à chaque fois qu'une modification est apportée aux données afin d'éviter toute perte. Le programme doit s'assurer que les objets en mémoire sont toujours synchronisés avec les fichiers JSON. Le programme doit également charger toutes ses données à partir des fichiers JSON et pouvoir restaurer son état entre les exécutions.

VUES

Une fois le programme lancé, l'utilisateur utilisera le menu principal pour effectuer des actions. Nous vous laissons le soin de décider de la liste des menus. Tant que l'utilisateur peut effectuer les actions spécifiées ci-dessus, nous serons contents !

LA STRUCTURE ET LA MAINTENANCE DU CODE

Le code doit être aussi propre et maintenable que possible pour éviter les bugs. Nous attendons que la structure du programme suive le modèle de conception **modèle-vue-contrôleur**. En d'autres termes, le programme devrait être divisé en trois packages : modèles, vues et contrôleurs, mais vous pouvez ajouter d'autres paquets ou modules appropriés (par exemple, vous pouvez mettre en place un module principal de haut niveau).

Si vous avez le choix entre la manipulation de dictionnaires ou d'instances de classe, choisissez toujours des instances de classe pour assurer la conformité avec le modèle de conception MVC.

Pour effectuer la mise en forme et le nettoyage du code, veuillez utiliser flake8 avec l'option de longueur de ligne maximale fixée à 119. Votre repository devra également contenir un rapport généré par flake8-html, dans un répertoire appelé "flake8_rapport", qui n'affiche aucune erreur et valide que votre code est conforme aux directives PEP 8.

Veuillez utiliser pip pour définir l'environnement Python. Votre repository doit également contenir un document README.md avec toutes les instructions pour exécuter le programme et générer un nouveau rapport flake8.