TD6 : Les Cesars 2016 (récupération de données via des requêtes HTTP)

Pour cette séances, vous travaillerez sur les données disponibles à l'adresse suivante :

`http://my-json-server.typicode.com/rtavenar/db_opendata/cesars2016`

Ces données concernent des films récompensés lors de la cérémonie des Césars 2016.

Préambule

— Créer sur votre disque un dossier TD6 dans PythonOpenData.
— Lancer l'éditeur Visual Studio Code.
— Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD6 et créez un fichier `td6.py`.

Exercice 1 : Un premier coup d'œil aux données

Q.1.1. À l'aide d'un navigateur web, visualisez les données disponibles à l'URL citée ci-dessus.

Q.1.2. Quel est le format de ces données ?

Q.1.3. Combien de films sont décrits dans ces données ?

Q.1.4. Dans votre fichier `td6.py`, récupérez le jeu de données en question et vérifiez que le nombre de films est conforme à ce que vous aviez noté précédemment.

Exercice 2 : Les "François" réalisateurs

Q.2.1. Écrivez une fonction qui prend en entrée un jeu de données tel que retourné par l'API qui nous intéresse et retourne la liste des réalisateurs dont le prénom est "François".

Exercice 3 : Les *biopics*

Il est à noter que l'API qui fournit les données peut être interrogée de la façon suivante :

`http://my-json-server.typicode.com/rtavenar/db_opendata/cesars2016?attribut=valeur`

auquel cas ne seront retournés que les films pour lesquels le champ `attribut` vaut `valeur`.

Q.3.1. Écrivez une fonction qui prend en entrée un genre cinématographique `genre` et un nom (de famille) d'acteur/actrice `nom_famille` et retourne la liste des titres de films du genre donné pour lesquels au moins un acteur ou une actrice a pour nom de famille `nom_famille`. La requête HTTP devra être incluse dans le corps de la fonction et vous ferez en sorte de minimiser la quantité de données à récupérer par votre script.

Q.3.2. Affichez la liste des "Biopic" dans lesquels joue une actrice ou un acteur du nom de "de Lencquesaing".

Q.3.3. Modifiez la fonction précédente pour que si l'argument `nom_famille` n'est pas fourni lors de l'appel, la fonction retourne l'ensemble des titres de films du genre voulu.

Q.3.4. Testez la fonction précédente pour afficher la liste des films du genre "Biopic".

Exercice 4 : Les actrices et acteurs

Q.4.1. Écrivez une fonction qui prend en entrée un jeu de données tel que retourné par l'API qui nous intéresse et retourne une liste avec doublons des actrices et acteurs contenus dans le jeu de données.

Q.4.2. Écrivez une fonction qui prend en entrée un acteur et retourne son nom de famille, s'il existe, ou sinon son surnom. La chaîne de caractères retournée devra être en caractères majuscules.

Q.4.3. **En utilisant les fonctions codées aux deux questions précédentes**, écrivez une fonction qui prend en entrée un jeu de données tel que retourné par l'API qui nous intéresse et retourne une liste sans doublon des actrices et acteurs contenus dans le jeu de données, triés dans l'ordre alphabétique des noms de famille (ou `surnom` si le nom de famille n'est pas spécifié).

Q.4.4. **En utilisant la fonction codée à la question 4.1**, écrivez une fonction qui prend en entrée un jeu de données tel que retourné par l'API qui nous intéresse et retourne la liste des acteurs ayant joué dans plusieurs films.

Exercice 5 : Export des données au format CSV

On souhaiterait maintenant exporter les données récupérées au format CSV. Malheureusement, le format CSV attend des données sous formes de tableaux, ce qui n'est pas très commode pour stocker des listes d'acteurs par films. La stratégie à suivre pour enregistrer ces données sera donc d'enregistrer 3 fichiers CSV :

- Un premier fichier qui contiendra les informations concernant les films du jeu de données ;
- Un deuxième fichier qui contiendra les informations concernant les actrices et acteurs du jeu de données ;
- Un troisième fichier qui listera les associations entre films et actrices/acteurs.

Q.5.1. Pour permettre de faire les liens souhaités entre films et acteurs, il faut fournir à chaque film un identifiant unique. Note : les acteurs disposent déjà d'un identifiant unique. Écrivez une fonction qui prend en entrée un jeu de données et retourne une nouvelle version de ce jeu de données dans laquelle chaque film a un nouvel attribut `id_film` unique.

Q.5.2. Appliquez cette fonction à votre jeu de données.

Q.5.3. Enregistrez les informations relatives aux acteurs dans un premier fichier `acteurs.csv` qui contiendra 4 colonnes : `id_acteur`, `prénom`, `nom`, `surnom`.

Q.5.4. Écrivez une fonction qui prend en entrée un jeu de données et retourne deux listes : la première correspondant au jeu de données dans lequel on a supprimé toutes les informations concernant les acteurs et la seconde stockant les liens entre acteurs et films sous la forme de paires stockées dans un dictionnaire `{"id_film":idf, "id_acteur":ida}`.

Q.5.5. Enregistrez les informations relatives aux films dans un fichier `films.csv` qui contiendra 7 colonnes : `id_film`, `titre`, `date_sortie`, `durée`, `réalisateur.prénom`, `réalisateur.nom` et `genre`.

Q.5.6. Enregistrez les informations permettant de lier les actrices et acteurs aux films dans un fichier `liens.csv` qui contiendra 2 colonnes : `id_film` et `id_acteur`.