TP n°2 : Format CSV et traitement

1 Le format CSV

Le format CSV pour Comma-Separated Values est un format de fichier permettant de stocker des données sous forme de tableau selon les principes suivants :

- il s'agit d'un fichier texte ;
- les lignes de texte correspondent aux lignes du tableau et se terminent par un caractère de fin de ligne afin de marquer le passage à la suivante (par exemple LF) ;
- au sein de chaque ligne, les cases du tableau et donc ses colonnes sont différenciées par un séparateur (généralement une virgule) ;
- la première ligne contient généralement les descripteurs du tableau, autrement dit les entêtes des colonnes de données.

Remarque :

- La plupart des tableurs comme LibreOffice sont capables de lire le format CSV, il est cependant aussi possible d'exploiter ce format à l'aide d'autres logiciels ou langages comme Python.
- Il existe plusieurs séparateurs et caractères de fin de ligne. Par exemple, en France, la virgule étant utilisée pour les nombres décimaux, le point-virgule peut être utilisé à la place. On retrouve la tabulation, la barre verticale, etc.

De nombreuses tables de données sont librement mise à disposition par des états, des organismes de recherche, des entreprises, etc s'inscrivant dans ce que l'on appelle l'Open Data. Ce partage permet de nombreuses études, de mieux comprendre les liens entre les différentes composantes de notre monde... En France, on peut accéder aux données partagées par l'État sur le site data.gouv.fr.

Exercice 1.

1. Récupérer le fichier pokemon.csv dans le dossier correspondant à ce TP.
2. Ouvrir le fichier dans un éditeur de texte puis dans un tableur. Identifier les descripteurs et les séparateurs.

2 Traitement des données

Exercice 2. [Taille d'une table] Écrire une fonction en Python prenant un fichier CSV en entrée et donnant le nombre de lignes et de colonnes de la table.

Exercice 3. [Descripteurs] Écrire une fonction Python prenant un fichier CSV en entrée et donnant et la liste de ses descripteurs. On laissera le `\n` sur le dernier descripteur.

Exercice 4.

1. Compléter la fonction ci-dessous afin qu'elle retire le \n à la fin du dernier élément d'une ligne importée à partir d'un fichier CSV. Elle prendra en entrée ladite ligne sous forme de liste et la renverra en sortie avec le dernier élément modifié.

```
def supprimer_slash_n(ligne : .....) -> ..... :

    dernier_element = ...................
    # on réaffecte le dernier élément sans le slash n
    ligne[.....] = ..............................

    return ligne
```

2. Modifier la fonction de récupération des descripteurs afin d'éliminer le \n de ceux-ci.

Exercice 5. [Table de données] Compléter la fonction ci-dessous prenant un fichier CSV en entrée, le séparateur le caractérisant et construisant la table de données associée sous la forme d'une liste de dictionnaire `descripteur : valeur`.

```
def construction_table(nom_fichier : str) -> list :

    with open(nom_fichier,"r",encoding="utf-8") as fichier :

        lignes = ......................... # liste de chaînes de caractères

    table = ..... # création de la table vide
    cles = ............................. # récupération des clés
    cles = ............................. # suppression du slash n

    for i in range(1, len(lignes)) :

        valeurs = ............................. # récupération des valeurs
        valeurs = ............................. # suppression du slash n

        # création du dictionnaire en compréhension
        ..........................................................
        ..............................# ajout de dico à la table


    return ..............
```

Exercice 6. [Recherche] Écrire une fonction effectuant les tâches suivantes :

1. demande à l'utilisateur un nom de fichier CSV à importer ;
2. afficher la liste des descripteurs possibles ;
3. demande à l'utilisateur de choisir un descripteur sur lequel effectuer une recherche (si le descripteur est erroné, le programme devra en informer l'utilisateur) ;
4. affiche la liste des valeurs possibles pour le descripteur choisi ;
5. demande à l'utilisateur de choisir une de ces valeurs ;
6. affiche les valeurs des autres descripteurs pour la valeur choisie.

Exercice 7. [Mass Effect] Créer une table de données à l'aide de Python avec la liste de descripteurs suivants :

```
["nom","force","endurance","vitesse","puissance biotique","endurance
  biotique","puissance technologique","endurance technologique"]],
```

pour valeurs de nom :

```
["Shepard","Liara","Garrus","Wrex","IDA","Grunt",
        "Thane","Légion","Mordin","Tali"]
```

et pour les valeurs des quatre autres descripteurs des nombres aléatoires entre 10 et 100. La table de données sera sauvegardé dans un fichier nommé mass_effect.csv.

Exercice 8. [Pokédex]

1. Écrire un programme important le pokemon.csv puis renvoyant pour chacun des descripteurs de la liste suivante

   ```
   ["attaque","défense","vitesse","attaque spéciale","défense spéciale"]
   ```

   les pokémons atteignant les valeurs minimale et maximale.
2. Écrire un programme important le fichier pokemon.csv et prenant entrée un descripteur et une valeur seuil puis renvoyant tous les pokémons ayant une valeur supérieure à la valeur seuil pour le descripteur donné. Par exemple, si on choisit « vitesse » et 50, le programme devra renvoyer la liste de tous les pokémons ayant une vitesse supérieur à 50.

Exercice 9. [Nouvelle ligne] Programmer une fonction ajoutant une nouvelle ligne à une table de données.

Exercice 10. [Nouvelle colonne] Programmer une fonction ajoutant une nouvelle colonne à une table de données.

Exercice 11. [Sélection] Programmer une fonction prenant en entrée une table de données et une liste de descripteurs puis donnant en sortant une table constituée des lignes de la première table correspondant aux descripteurs choisis.

Exercice 12. [Tri] Programmer une fonction triant les lignes d'une table de données en fonction d'un descripteur choisi.

Exercice 13. [Doublon] Programmer une fonction déterminant s'il existe deux lignes (ou plus) ayant les mêmes valeurs pour l'ensemble des descripteurs.

3 Fusion de tables

Lorsque l'on traite de grandes quantités de données, celles-ci sont souvent réparties dans plusieurs tables. On est donc souvent amené à regrouper des données dans une nouvelle table. Cette opération s'appelle la jointure de tables.

Nom	Attaque	Défense	Vitesse
Bulbizarre	49	49	45
Salamèche	52	43	65
Carapuce	48	65	43

Nom	Type 1	Type 2
Bulbizarre	Plante	Poison
Salamèche	Feu	
Carapuce	Eau	

On peut regrouper ces deux tables grâce au champ `Nom` qui est commun. On obtient le résultat suivant :

Nom	Attaque	Défense	Vitesse	Type 1	Type 2
Bulbizarre	49	49	45	Plante	Poison
Salamèche	52	43	65	Feu	
Carapuce	48	65	43	Eau	

Exercice 14. On considère les tables Membre, Prêt, Livre et Commandé ci-dessous.

Membres	
prénom	idm
Géralt	12
Triss	24
Yennefer	42
Ciri	7

Prêts	
idl	idm
12	42
11	7
3	42
4	7
8	12

Commandés	
titre	livré
Le Meilleur des Mondes	Non
Illium et Olympos	Oui

Livres	
idl	titre
1	Fondation
2	Les Robots
3	Dune
4	L'Aube de la Nuit
5	1984
6	Les Guerriers du Silence
7	Hypérion
8	Farenheit 451
9	La Saga du Commonwealth
10	La Trilogie du Vide
11	Les Naufragés du Commonwealth
12	Le Maître du Haut Château

1. Calculer à la main la jointure des tables Membres et Prêts puis la jointure des tables Prêts et Livres.
2. Pour chaque table, créer un fichier CSV correspondant.
3. Écrire un programme Python qui charge les fichiers membre.csv, prets.csv et livres.csv et qui calcule les jointures de la question 1.
4. Écrire un programme Python qui charge les trois fichiers membres.csv, prets.csv et livres.csv, qui calcule la jointure de ces trois tables et sauve le résultat dans un fichier.
5. La table Commandé liste les livres commandés et s'ils ont été livrés ou non. Proposer une méthode permettant de faire la réunion des deux tables Commandés et Livres et la coder en Python. On créera des identifiants idl pour les livres commandés reçus.