TD : Introduction au traitement des données

Une des utilisations importantes de l'informatique est le traitement des innombrables données récupérées dans différents domaines, de la santé jusqu'à la publicité. Les données sont regroupées en **bases de données**.

Des langages et des logiciels spécialisés existent pour traiter ces bases, qui peuvent contenir des millions de données. Nous allons explorer quelques fonctions de base de ces langages, en les programmant en Python.

Les bases de données sont structurées. Chaque ligne d'une base de données est appelé un **enregistrement**, et chaque enregistrement possède les mêmes champs, qui sont identifiés par des **descripteurs**.

En prologue, un peu d'exploration du format csv.

Le format CSV

Le format de fichier csv est un format standard d'échange de données. Il est reconnu par tous les tableurs notamment. CSV siginifie Comma Separated Values, c'est-à-dire "valeurs séparées par des virgules". Un fichier csv est sous format texte, ce qui le rend facilement compréhensible à la fois par les huamins et par les machines. De ce fait, Python considère automatiquement tous les champs comme des chaînes (ce qui n'est pas le cas pour un tableur)

Exemple : données sur le film "The Dark Knight Rises" (titre, année de sortie, durée, langue originale, genre, recettes)

```
The Dark Knight Rises,2012,165,en,Action,1084939099
```

Trois problèmes se posent immédiatement à la lecture de cet exemple:

- Comment différencier deux films au titres identiques (par exemple le Scarface d'Howard Hawks de 1932 du Scarface de Brian de Palma de 1983) ?
- Un film rentre en général dans plusieurs catégories de genre, commment faire ?
- En France, on écrit les nombres à virgule avec... une virgule. Comment faire pour différencier la virgule qui sépare deux champs de la virgule qui sépare partie entière et partie décimale ?

Identifiant unique

Pour différencier deux films, une idée est d'associer titre du film et réalisateur. Problème : Hitchcok a réalisé deux versions de "L'Homme qui en savait trop"... Associer titre et année ne fonctionne pas mieux.

On crée donc un entier unique associé à chaque enregistrement : c'est l'identifiant unique. L'identifiant unique permet de différencier plusieurs données dont certains champs sont identiques. Par exemple en France, il peut y avoir des homonymes complets, mais le

numéro de sécurité sociale est unique. D'autre part, faire une recherche sur entier est bien plus rapide que sur une chaîne de caractères, dans les langages spécialisés dans le traitement des données.

Exemple : L'identifiant unique du film "The Dark Knight Rises" est 49026

```
49026,The Dark Knight Rises,2012,165,en,Action,1084939099
```

Changer le délimiteur

Plutôt que d'avoir les champs séparés par une virgule, on peut utiliser un autre délimiteur. Sont utilisés fréquemment la tabulation (il y a même un format de fichier ".tsv"), ainsi que le point-virgule. Avoir un délimiteur différent résout les deux problèmes cités ci-dessus, celui des valeurs multiples pour un champ, et celui de l'écriture des nombres en France.

Dans ce TP d'introduction, le délimiteur des fichiers csv est le ";".

Exemple : on complète les genres de "The Dark Knight Rises"

```
49026;The Dark Knight Rises;2012;165;en;Action,Crime,Drama,Thriller;10849390
```

Exploration préalable d'un fichier csv

Ouvrir le fichier "films_exemple.csv" avec un tableur (par défaut avec un double-clic), et également avec un éditeur de textes (bloc-notes ou Notepad++).

Sous éditeur de textes, la forme brute avec les délimiteurs est visible.

Sous tableur, le formatage des enregistrements en ligne et colonnes est visible.

Remarque : les données proviennent de The Movie DataBase, et correspondent (probablement) aux films les plus appréciés sur une semaine donnée. Pour ce TD d'introduction, on utilisera une version restreinte aux 50 premiers films. Dans les exercices par la suite, on travaillera sur ce qui reste une petite base de données, avec 5000 films répertoriés.

Lecture d'une tables de données.

Le code suivant permet de lire une table de données et de lire les cinq premières lignes. Testez-le.

Remarques :

- il est indispensable de modifier le chemin d'accès du fichier pour pourvoir l'ouvrir !
- On a déjà utilisé les fonctions d'ouverture (open) et de fermeture (close) dans le devoir à la maison sur les dictionnaires. On précise l'encodage (utf-8) et le mode d'ouverture (`r` pour read).
- il ne faut pas oublier de fermer le fichier.

```
[ ]  import csv
```

```
from collections import OrderedDict   # structure de données
utilisée

# On importe le fichier d'exemple de base de données de films
exemple_films_csv = open('films_exemple.csv','r', encoding ='utf-
8')
lecteur_exemple = csv.DictReader(exemple_films_csv,
delimiter=';')

compteur_ligne = 0
for film in lecteur_exemple:
    print(film)
    compteur_ligne = compteur_ligne + 1
    if compteur_ligne == 5:
        break

exemple_films_csv.close()
```

Plusieurs remarques s'imposent:

- Chaque ligne est précédée de "OrderedDict" à partir de la version 3.6 de Python. Ce nouveau type de variable est comme son nom l'indique un dictionnaire ordonné. Les méthodes vues précédemment `.keys()`, `.values()` et `.items()` fonctionnent de la même manière qu'avec les dictionnaires usuels.
- Je vous déconseille l'utilisation du mot clé `break` autant que possible. Le programme suivant est plus "propre" et fait exactement la même chose.

[ ]
```
import csv
from collections import OrderedDict   # structure de données
utilisée

# On importe le fichier d'exemple de base de données de films
exemple_films_csv = open('films_exemple.csv','r', encoding ='utf-
8')
lecteur_exemple = csv.DictReader(exemple_films_csv,
delimiter=';')

"""
# Une méthode élémentaire pour créer la liste des enregistrement
ligne par ligne
films = []
for ligne in lecteur_exemple:
   films.append(ligne)
"""
#Une méthode plus rapide pour créer la liste des enregistrements
films = list(lecteur_exemple)

# On affiche les 5 premiers films
for i in range(5):
```

```
        print(films[i])

    # on peut aussi écrire : print(films[:5])
```

Remarque : On ne ferme pas le fichier pour coninuer à travailler dessus.

Utilisons les méthodes sur les dictionnaires. Ecrire le code permettant de lire :

- d'abord les clés du premier enregistrement ;
- puis les valeurs ;
- et enfin les items.

Affiner votre code de manière à avoir la liste des clés, valeurs et items.

Puis réjouter le code qui donne le d'enregistrements dans cette table.

```
[ ]
```

Le code suivant permet d'afficher uniquement les titres. Le modifier pour afficher titre et année de sortie.

```
[ ]  for i in range(5):
        print(films[i]['title'])
```

Ci-dessous le code pour une présentation plus sympathique, on rajoute un formatage en précisant le nombre de caractères. Le nombre maximum de caractères dans un champ de base de données est obligatoire (mais pas dans le csv).

Les méthodes `.ljust(nombre_de_caractères)` et `.rjust(nombre_de_caractères)` permettent de faire une colonne avec *nombre de caractères* caractères, complétés par des espaces si nécessaire, respectivement justifiées à gauche et à droite. `{:<nombre_de_caractères}` et `{:>nombre_de_caractères}` ont le même effet. On précise ensuite dans la méthode `.format()` les chaînes de caractères à afficher. Compléter le code pour afficher les genres entre le titre et les recettes.

```
[ ]  for i in range(5):
        print(films[i]['title'].ljust(30),films[i]
    ['revenue'].rjust(15))

    print()

    #Deuxième méthode, plus puissante mais moins lisible
    for i in range(5):
        print('{:<50} | {:>15}'.format(films[i]['title'],films[i]
    ['revenue'])) # avec en plus un séparateur vertical |
```

Remarque : on ferme le fichier pour que la suite du Notebook soit propre

```
[ ]  exemple_films_csv.close()
```

On peut bien sûr rajouter un enregistrement dans une table de données. Code :

```
[ ]  import csv
     from collections import OrderedDict   # structure de données
     utilisée

     # On importe le fichier d'exemple de base de données de films
     exemple_films_csv = open('films_exemple.csv','r', encoding ='utf-
     8')
     lecteur_exemple = csv.DictReader(exemple_films_csv,
     delimiter=';')

     films = list(lecteur_exemple)

     exemple_films_csv.close()

     # On rajoute une valeur dans notre liste
     films.append(OrderedDict([('movie_id','16130'),('title','La
     classe americaine'), ('release_year','2020'), \
                               ('runtime', '70'),
     ('original_language', 'fr'), ('genres', 'Comedy'),
     ('revenue','unknown')]))

     # On vérifie que notre film est ajouté en filtrant sur la valeur
     de son titre
     for film in films:
         if film['title'] == 'La classe americaine':
             print(film)

     exemple_films_csv.close()
```

L'année de sortie du film "La Classe Américaine" n'est pas 2020 mais 1993. Ecrire le code qui modifiera ce champ, et vérifie que le modification est bien effective.

```
[ ]
```

Ecriture de la base de données dans un nouveau fichier.

De la même manière qu'on a ouvert le fichier de données en lecture, puis créé un "lecteur", ici on va ouvrir un fichier en écriture (rappel : s'il n'existe pas il est créé à cette occasion), puis construire un "écrivain".

Faire tourner le code, ouvrir le nouveau fichier dans un tableur et vérifier que la ligne rajoutée est présente.

[ ]
```
def creer_csv(table_donnees, nom_fichier):
    """
    Crée un fichier csv à partir d'une table de données.
    @param table_donnees : liste de dictionnaires ordonnés
    @param nom_fichier : chaine de caractères, la logique veut
qu'elle finisse par l'extension .csv
    """
    en_tete = list(table_donnees[0].keys())
    csv_bis = open(nom_fichier, mode="w", encoding="utf-8",
newline = "")
    csv_ecrivain = csv.DictWriter(csv_bis, delimiter
=";",fieldnames = en_tete)

    csv_ecrivain.writeheader()
    for ligne in table_donnees:
        csv_ecrivain.writerow(ligne)

    csvfile.close()
    return

creer_csv(films,"films_exemples_bis.csv")
```

On peut aussi supprimer un film avec la méthode `.remove()`. Le code suivant supprime le film que l'on vient de rajouter.

[ ]
```
# On enlève le film qu'on a ajouté
for film in films:
    if film['title'] == 'La classe americaine':
        films.remove(film)

# On vérifie qu'il n'est bien plus présent
for film in films:
    if film['title'] == 'La classe americaine':
        print(film)
print("vérification effectuée")
```

Recherches et calculs dans une table de données

Ecrire une fonction qui admet comme paramètre d'entrée la liste des films, et donne en retour la liste des titres de films sortis en 2010.

Indice avec les lignes déjà écrites

```
[ ]  film_0 = films[0]
     print('le champ des recettes "revenue" a pour type
     ',type(film_0['revenue']))
```

Ecrire une fonction qui donne les recettes moyennes des films de cette table, ainsi que le titre du film ayant eu la recette minimale (et ladite recette).

```
[ ]
```

Tri d'une table

On utilisera les fonctions et méthodes de tri intégrées à Python. On dispose de:

- `truc.sort()` qui a la particularité de modifier *truc*
- `bidule = sorted(truc)` où l'on crée une nouvelle variable *bidule* à partir de *truc*, qui est inchangée. On utilisera la deuxième méthode ; l'appliquer sans aucun préalable sur la liste et afficher les dix premiers éléments. Que constatez-vous ?

```
[ ]  bidule = sorted(films)
     print(bidule[:5])
```

Pour trier les enregistrements de la table, on a besoin d'une **clé** de tri. On crée cette clé en extrayant les valeurs correspondantes, et elle est passée en argument dans l'appel de la fonction.

```
[ ]  def cle_titre(ligne):
         """
         Renvoie la valeur du champ 'title' d'un enregistrement de la
     table
         """
         return ligne['title']
```

```
films_tries = sorted(films, key = cle_titre)
for i in range(10):
    film = films_tries[i]
    print(film['title'])
```

L'argument `reverse = True` permet d'inverser le tri. Le tester dans la fonction précédente, puis donner ci-dessous les dix films les plus récents. On donnera l'année de sortie et le titre du film.

```
[ ]
```

Jointure de tables (fusion)

On souhaite associer à chaque film son réalisateur. On réalise une **jointure** de deux tables, en utilisant l'identifiant unique du film. Comprenez le code ci-dessous, qui sera utile comme exemple pour les exercices.

```
[ ]  # On veut associer les réalisateurs aux films, pour ça on doit
     faire une jointure entre deux fichiers et on ne garde que
     l'identifiant du film, le titre et le nom du réalisateur
     # On importe les données de réalisateurs
     exemple_realisateurs_csv = open('realisateurs_exemple.csv','r',
     encoding ='utf-8')
     lecteur_realisateurs_exemple =
     csv.DictReader(exemple_realisateurs_csv, delimiter=';')

     realisateurs = []

     for ligne in lecteur_realisateurs_exemple:
        realisateurs.append(ligne)

     # On fait la jointure en utilisant des identifiants internes (id)
     car plusieurs films peuvent avoir le même titre
     films_avec_realisateurs = []
     for film in films:
        for real in realisateurs:
           if film['movie_id'] == real['movie_id']:
              films_avec_realisateurs.append(
     OrderedDict([('Identifiant',film['movie_id']),
     ('Titre',film['title']), \

     ('Réalisateur',real['director'])]) )
```

```
for ligne in films_avec_realisateurs:
    print('{:<50} | {:
<50}'.format(ligne['Titre'],ligne['Réalisateur']))

exemple_realisateurs_csv.close()
```

Validation des données

Reprendre le calcul de la recette moyenne avec le fichier `films_exemples_bis.csv`.

Que constatez-vous ? Réglez le problème !

Remarque : on ne change pas les données, en effet le spécialiste en base de données traite l'information mais ne la recueille pas.

```
[ ]
```

[**Frederic Mandon**](mailto:frederic.mandon@ac-montpellier.fr), Lycée Jean Jaurès - Saint Clément de Rivière - France (2020)