Traitement de données en tables

1. Contenu & compétences exigibles

- Indexation de tables : importer une table depuis un fichier texte tabulé ou un fichier CSV.
- Recherche dans une table : rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.
- Tri d'une table : trier une table suivant une colonne.
- Fusion de tables : construire une nouvelle table en combinant les données de deux tables.

2. Introduction

Une des utilisations principales de l'informatique de nos jours est le traitement de quantités importantes de données dans des domaines très variés : un site de commerce en ligne peut avoir à gérer des bases données pour des dizaines de milliers (voire plus) d'articles en vente, de clients, de commandes, un hôpital doit pouvoir accéder efficacement à tous les détails de traitements de ses patients, etc. Mais si les logiciels de traitement de base de données sont des programmes hautement spécialisés pour effectuer ce genre de tâches le plus efficacement possible, il est facile de mettre en œuvre les opérations de base dans un langage de programmation comme Python. Les données organisées en table correspondent à une liste de p-uplets nommés qui partagent les mêmes descripteurs.

3. Le format csv, création et lecture de fichiers csv

A] Un peu de vocabulaire…

Le format csv (pour comma separated values, soit en français valeurs séparées par des virgules) est un format très pratique pour représenter des données structurées. Dans ce format, **chaque ligne** représente un enregistrement ou **objet**.

Sur une même ligne, les différents champs de l'enregistrement sont réparés par une virgule (d'où le nom).

- les virgules correspondent aux séparations entre les colonnes
- la **première ligne du fichier** donne la **liste des descripteurs**
- les lignes suivantes donnent les **valeurs de chaque objet**, suivant les descripteurs

En pratique, on peut spécifier le caractère utilisé pour séparer les différents champs et on utilise fréquemment un point-virgule, une tabulation ou deux points.

Un exemple :

```
Sexe,Prénom,Année naissance
M,Alphonse,1952
F,Béatrice,1977
F,Charlotte,1996
```

1.1 Quels sont les descripteurs dans cet exemple ?

1.2 Donner un exemple d'objet.

1.3 Donner une valeur correspondant à cet exemple de fichier csv.

B] Création d'un fichier texte

Pour créer un fichier csv, il suffit de savoir créer un fichier texte…

- Ouvrir un éditeur de texte comme le bloc note par exemple.
- Recopier les quatre lignes de l'exemple précédent
- Enregistrer le fichier sous le nom exemple_nsi.csv en choisissant l'encodage UTF-8 s'il est disponible.
- Pour compléter votre fichier csv, il suffit d'ajouter des lignes ou d'autres colonnes.

3.1. Que se passe-t-il si vous essayez d'ouvrir ce fichier avec un tableur ? Qu'observe-t-on ?

3.2. Ajouter alors deux lignes de données supplémentaires (vous avez suffisamment d'imagination) et sauvegarder ce fichier.

C] Lecture d'un fichier csv avec python

Avec Python il y a plusieurs manières pour ouvrir un fichier CSV : lecture et écriture de fichier texte, module csv, module pandas...

Dans le même dossier que le fichier de données précédent, créer un fichier Python nommé lecture_csv.py.

3.1. Recopier le programme ci-dessous sans les lignes de commentaires.

```
# Fiche cours sur le traitement de données en tables
# Exemple 1 : lecture de fichier CSV

# Importation de la bibliothèque permettant de gérer des fichiers CSV
import csv

# Ouverture du fichier csv en mode lecture 'r'
fichier_csv = open('exemple_nsi.csv',  mode = 'r', encoding='UTF8')

# Création d'un 'lecteur' itérable
lecteur = csv.reader(fichier_csv, delimiter = ',')

# Création d'une liste à partir du fichier csv
Liste = []
for ligne in lecteur:
    Liste.append(ligne)

# Fermeture du fichier csv
fichier_csv.close()

print(Liste)
```

3.2. Indiquer les instructions permettant d'afficher dans la console :

- Le prénom du troisième objet
- L'ensemble des dates de naissance (avec une boucle)

Remarques :

Il existe différents modes pour l'ouverture d'un fichier csv. Voici quelques exemples :

- **r :** ouvre le fichier en mode lecture seule. Démarre la lecture depuis le début du fichier et constitue le mode par défaut de la fonction open ().
- **w** : s'ouvre en mode écriture seule. Le pointeur est placé au début du fichier et cela écrasera tout fichier existant avec le même nom. Il créera un nouveau fichier si un avec le même nom n'existe pas.
- **a** : ouvre le fichier pour y ajouter de nouvelles informations. Le pointeur est placé à la fin du fichier. Un nouveau fichier est créé s'il n'y en a pas avec le même nom.

3.3. Recopier le programme ci-dessous qui utilise l'instruction « with open » pour récupérer les données :

```
# Exemple 2 : méthode 'with open'

# Importation de la bibliothèque permettant de gérer des fichiers CSV
import csv

Liste_bis = []

with open ('exemple_nsi.csv',  mode = 'r', encoding='UTF8') as fichier_csv:
    lecteur = csv.reader(fichier_csv, delimiter = ',')
    for ligne in lecteur:
        Liste_bis.append(ligne)

    print(Liste_bis)
```

Cette structure en liste de liste n'est pas la plus satisfaisante car le lien entre les valeurs du tableau et le nom des enregistrements, n'est pas direct. Il est préférable d'utiliser un dictionnaire…

3.4.1. Création d'un dictionnaire ordonné : recopier le programme ci-dessous.

```
# Exemple 3 : dictionnaire ordonné

# Importation de la bibliothèque permettant de gérer des fichiers CSV
import csv

Liste = []

with open ('exemple_nsi.csv',  mode = 'r', encoding='UTF8') as fichier_csv:
    lecteur = csv.DictReader(fichier_csv, delimiter = ',')

    for ligne in lecteur:
        # pour avoir un affichage plus plaisant, on ajoute l'instruction 'dict'
        Liste.append(dict(ligne))

    print(Liste)
```

3.4.2. Qu'obtient-on avec l'instruction ***print(Liste[1])*** ?

Remarques :

- Un objet de cette table csv est représenté par un dictionnaire :
  {'Sexe': 'F', 'Prénom': 'Béatrice', 'Année naissance': '1977'}
- Et un fichier csv par une liste de dictionnaires, dont les clés sont les noms des colonnes :
  ma_table = [{'Sexe': 'M', 'Prénom': 'Alphonse', 'Année naissance': '1952'},
  {'Sexe': 'F', 'Prénom': 'Béatrice', 'Année naissance': '1977'},
  {'Sexe': 'F', 'Prénom': 'Charlotte', 'Année naissance': '1996'}]

D] Création d'un fichier csv avec python à partir d'un dictionnaire

Vous allez créer un fichier csv à partir de la table ci-dessous :

```
table_spe = [{'Nom':'Baron','Prénom':'Paul','NSI':'18','Physique':'16','Maths':'15'},
             {'Nom':'Tallant','Prénom':'Greg','NSI':'10','Physique':'13','Maths':'9'},
             {'Nom':'Goury','Prénom':'Zoé','NSI':'13','Physique':'14','Maths':'11'}]
```

4.1. Recopier le programme ci-dessous et vérifier la création correcte du fichier csv.

```
1  # Exemple 4 : création d'un fichier csv
2
3  import csv
4
5  # Table à utiliser
6  table_spe = [{'Nom':'Baron','Prénom':'Paul','NSI':'18','Physique':'16','Maths':'15'},
7               {'Nom':'Tallant','Prénom':'Greg','NSI':'10','Physique':'13','Maths':'9'},
8               {'Nom':'Goury','Prénom':'Zoé','NSI':'13','Physique':'14','Maths':'11'}]
9
10 headers = table_spe[0].keys()
11
12 with open('exemple_4.csv', 'w', encoding = 'UTF-8', newline='') as f:
13     f_csv = csv.DictWriter(f, headers, delimiter = ';')
14     f_csv.writeheader()
15     f_csv.writerows(table_spe)
```

4.2. Commenter les instructions des lignes suivantes (utiliser le vocabulaire adapté) : ligne 10, ligne 14 & ligne 15.

Remarque :

La fonction « **writerows** » avec un « **s** » à la fin, permet d'écrire en une seule fois tout le contenu d'une liste contenant elle-même des sous-listes (comme dans l'exemple ci-dessus).

Par contre, pour écrire uniquement le contenu d'une liste, sans sous-liste, il faut utiliser la fonction « **writerow** » (sans le « **s** » à la fin).

E] Création d'un fichier csv avec python à partir à partir d'une liste de listes

```
# Exemple 5 : création d'un fichier csv à partir d'une liste de listes

import csv

liste = [['Nom', 'Prénom', 'NSI', 'Physique', 'Maths'],
         ['Baron', 'Paul', 18, 16, 15],
         ['Tallant', 'Greg', 10, 13, 9],
         ['Goury', 'Zoé', 13, 14, 11]]

with open('exemple_5.csv', 'w', encoding = 'UTF-8', newline = '') as f:
    f_csv = csv.writer(f, delimiter = ';')
    f_csv.writerows(liste)
```

5.1. Réaliser le programme ci-dessus puis vérifier le contenu du fichier csv ainsi créé dans le bloc-note.

5.2. Observer ce qui se passe si l'on supprime l'instruction « newline… ».

5.3. Application :

Vous devez créer un programme nommé exemple_6.py qui permet d'insérer un nouvel enregistrement dans un fichier csv (celui créé précédemment). Tester votre programme avec l'élève CMOI Arthur, qui a eu respectivement 15, 16 et 11.

4. Traitement de données en tables

A. Sélection de données suivant certains critères :

1. Analyser le programme ci-dessous. Expliquer ce que l'on va obtenir dans la console.

```
# Exemple 7 : sélection de données

import csv

tableau = []
fichier = open('exemple_5.csv', 'r', encoding = 'UTF-8', newline = '')
lecteur = csv.DictReader(fichier, delimiter = ';')

for ligne in lecteur:
    tableau.append(dict(ligne))

print(tableau, '\n')

# Liste des prénoms dans le fichier :
resultat_1 = [ligne['Prénom'] for ligne in tableau]
print(resultat_1, '\n')
```

2. Exécuter ce programme et compléter si nécessaire votre réponse précédente.
3. Quelles instructions utiliser pour obtenir la liste des noms des élèves ayant plus de 12/20 en NSI ?
4. Quelles instructions utiliser pour obtenir la liste des noms des élèves ayant plus de 15/20 en NSI et plus de 12 en maths ?
5. Quelles instructions utiliser pour obtenir la liste des notes obtenues par Zoé ?
6. Quelles instructions utiliser pour obtenir la moyenne des notes des élèves en NSI ?
7. Construire un fichier csv nommé ***moyenne.csv*** contenant les noms & prénoms des élèves avec leur moyenne pour l'ensemble des spécialités.

B. Comment trier une table ?

Utilisation de la fonction sorted(tableau, key = None, reverse = False)

En Python, la fonction sorted(tableau) permet de trier un tableau. Cette fonction dispose d'un argument optionnel nommé key permettant de préciser selon quel critère une liste doit être triée. La valeur du paramètre key devrait être une fonction qui prend un seul argument et renvoie une clef à utiliser à des fins de tri. Cette technique est rapide car la fonction clef est appelée exactement une seule fois pour chaque enregistrement en entrée. Un troisième argument optionnel reverse (un booléen) permet de préciser si on veut le résultat par ordre croissant par défaut) ou décroissant (reverse=True).

Remarque : ne confondez pas avec la procédure de tri des listes « List.sort() » qui va modifier la liste elle-même alors que la fonction sorted() renvoie une nouvelle liste correspondant la liste triée.

B.1. Analyser le programme ci-dessous et ce que l'on obtient dans la console python en exécutant ce programme. Expliquer les paramètres pour les tris effectués.

```
1  # Exemple 8.1. : tri de tables
2
3  liste_1 = ['cri', 'car', 'ce', 'bonjour', 'a', 'b']
4  liste_triee = sorted(liste_1)
5  print(liste_triee, '\n')
6
7  print('Liste non-triée', liste_1, '\n')
8
9  liste_triee = sorted(liste_1, key = len)
10 print(liste triee)
```

```
>>> %Run Exemple_8.py
  ['a', 'b', 'bonjour', 'car', 'ce', 'cri']

  Liste non-triée ['cri', 'car', 'ce', 'bonjour', 'a', 'b']

  ['a', 'b', 'ce', 'cri', 'car', 'bonjour']
```

On veut effectuer un tri sur la table de données ci-dessous par ordre alphabétique des noms.

```
table_spe = [{'Nom':'Baron','Prénom':'Paul','NSI':'18','Physique':'16','Maths':'15'},
             {'Nom':'Tallant','Prénom':'Greg','NSI':'10','Physique':'13','Maths':'9'},
             {'Nom':'Goury','Prénom':'Zoé','NSI':'13','Physique':'14','Maths':'11'}]
```

- Pour cela il faut créer une fonction qui va renvoyer la valeur de la clé 'Nom' de chaque dictionnaire de la liste.
- Dans sorted(), on va passer la fonction créée en paramètre key.

```
# Exemple 8.2. : tri de tables par ordre alphabétique

table = [{'Nom':'Baron','Prénom':'Paul','NSI':'18','Physique':'16','Maths':'15'},
        {'Nom':'Tallant','Prénom':'Greg','NSI':'10','Physique':'13','Maths':'9'},
        {'Nom':'Goury','Prénom':'Zoé','NSI':'13','Physique':'14','Maths':'11'}]

def nom(ligne):
    return ligne['Nom']

tri_nom = sorted(table, key = nom, reverse = False)
print(tri_nom)
```

B.2. Que se passe-t-il si l'on remplace l'instruction False par True ?

B.3. Effectuer un tri selon les notes en Maths (ordre décroissant). Attention, convertissez les notes en flottants dans la fonction sinon ...

C. Fusion de deux tables

Il faut que ces deux tables aient au moins un descripteur en commun.

```
# Exemple C : fusion de deux tables

table_1 = [{'Nom':'Baron','Prénom':'Paul','NSI':'18','Physique':'16','Maths':'15'},
        {'Nom':'Tallant','Prénom':'Greg','NSI':'10','Physique':'13','Maths':'9'},
        {'Nom':'Goury','Prénom':'Zoé','NSI':'13','Physique':'14','Maths':'11'}]

table_2 = [{'Nom':'Baron','Age': '16', 'Anglais':'12'},
           {'Nom':'Tallant','Age': '17', 'Anglais':'13'},
           {'Nom':'Goury','Age': '15', 'Anglais':'11'}]

def fusion(table_1,table_2,cle) :
    table_fusion=[]
    for ligne_1 in table_1 :
        for ligne_2 in table_2 :
            if ligne_1[cle] == ligne_2[cle] :
                new_ligne = ligne_1
                for key in ligne_2 :
                    if key != cle :
                        new_ligne[key] = ligne_2[key]
                table_fusion.append(new_ligne)
    print(table_fusion)
    return table_fusion

fusion(table_1, table_2, 'Nom')
```

5. TD

Exercice 1 :

On considère les tables Membre (idm est le numéro d'identification de la personne) , Prêt et Livre (idl est le numéro d'identification du livre) ci-dessous :

prénom	idm
Bertrand	12
Lucie	24
Jean	42
Sara	7

Table Membre

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

Table Prêt

idl	titre
1	Fondation
2	Les Robots
3	Dune
4	La Stratégie Ender
5	1984
6	Santiago
7	Hypérion
8	Farenheit 451
9	Les Monades urbaines
10	Star Wars - épisode IX
11	L'étoile et le fouet
12	Ubik

Table Livre

1. Pour chaque table, écrire le contenu d'un fichier CSV correspondant : membre.csv, pret.csv et livre.csv
2. Construire ci-dessous, à la main, la fusion des trois tables. On la nomme fusion_totale.
3. Écrire un programme en Python nommé ***fusion_globale.py*** qui charge les trois fichiers membre.csv, pret.csv et livre.csv et qui construit la fusion globale précédente en créant un fichier csv nommé ***fusion_globale.csv***

Exercice 2 :

Récupérer le fichier csv correspondant aux prénoms donnés en France depuis 1900 (hors Mayotte) jusqu'à 2019 à partir du site web de l'INSEE. Ce fichier devrait être nommé « nat2019.csv ».

Vous devez réaliser un programme python nommé « prenoms_2019.py » qui permette de répondre à l'ensemble des questions ci-dessous.

1. Afficher sous forme d'une liste de liste l'ensemble des données du fichier csv.
2. Déterminer le nombre d'objets dans ce fichier.
3. Donner la liste des descripteurs du fichier csv.
4. Déterminer le nombre de NICOLAS nés en 1982.
5. Demander à l'utilisateur un prénom (en majuscule) et une année et qui renvoie le nombre de naissances correspondant à ces caractéristiques.
6. En quelle année le prénom NOEMIE a-t-il été donné 184 fois ?
7. Demander à l'utilisateur un prénom puis déterminer le nombre total de fois où ce prénom a été donné depuis 1900.
8. Est-ce que des filles ont été prénommées NICOLAS ? Si oui en quelle(s) année(s) ?
9. Demander à l'utilisateur un prénom et donner l'année en laquelle ce prénom a été le plus donné. Le programme renverra également ce nombre maximal.
10. Demander à l'utilisateur une année et renvoyer le prénom le plus donné en cette année.