Activité 08 - Données, CSV

Equipe Pédagogique LU1IN0*1

Consignes : Cette activité se compose d'une première partie guidée, suivie de suggestions. Il est conseillé de traiter en entier la partie guidée avant de choisir une ou plusieurs suggestions à explorer.

L'objectif de cette activité est l'utilisation de données récupérées sous forme de fichiers .CSV, sans utiliser des bibliothèques de haut-niveau.

Le format CSV (*comma-separated values*) est un format de fichier texte permettant d'échanger facilement des jeux de données.

On pourra consulter la page *Wikipédia* afférente : `https://fr.wikipedia.org/wiki/Comma-separated_values` et retenir ces "règles" :

- les données sont présentées sous forme de tableau, donné ligne par ligne,
- dans chaque ligne, les données sont séparées par un séparateur, soit une virgule (*comma*) pour les CSV "anglophones", soit un point-virgule, pour les CSV "francophones",
- la première ligne donne les noms des colonnes.

Par exemple, voici le contenu d'un fichier CSV "francophone" `tournois.csv` (disponible sur Moodle) collectant les résultats de tournois sportifs entre amis :

```
"sport";"date";"participants";"vainqueur"
"boxe";2021-09-18;12;"Alice"
"boxe";2021-09-25;10;"Alice"
"karate";2021-09-26;19;"Carole"
"boxe";2021-10-02;8;"Bob"
"karate";2021-10-03;20;"Carole"
"tennis";2021-10-04;3;"Alice"
"boxe";2021-10-09;5;"Alice"
"karate";2021-10-10;20;"Damien"
"boxe";2021-10-16;6;"Carole"
"echecs";2021-09-17;120;"Bob"
"echecs";2021-09-24;120;"Bob"
"echecs";2021-10-01;120;"Carole"
```

On peut ouvrir les fichiers csv en python avec la fonction suivante (disponible sur Moodle) :

```
def ouvre_fichier(nom : str) -> List[str] :
    """ renvoie la liste des lignes du fichier texte ./nom.csv """
    with open("./"+nom+".csv", "r") as f:
        return f.readlines()
```

Par exemple si `tournois.csv` est présent dans le répertoire dans lequel *MrPython* a été lancé, un appel à `ouvre_fichier("tournois")` renvoie :

```
['"sport";"date";"participants";"vainqueur"\n',
 '"boxe";2021-09-18;12;"Alice"\n',
 '"boxe";2021-09-25;10;"Alice"\n',
 '"karate";2021-09-26;19;"Carole"\n',
 '"boxe";2021-10-02;8;"Bob"\n',
 '"karate";2021-10-03;20;"Carole"\n',
 '"tennis";2021-10-04;3;"Alice"\n',
 '"boxe";2021-10-09;5;"Alice"\n',
 '"karate";2021-10-10;20;"Damien"\n',
 '"boxe";2021-10-16;6;"Carole"\n',
 '"echecs";2021-09-17;120;"Bob"\n',
 '"echecs";2021-09-24;120;"Bob"\n',
 '"echecs";2021-10-01;120;"Carole"\n']
```

Le fichier a été lu comme une liste de lignes, qui finissent toutes par un retour chariot `"\n"`.

1 Partie guidée : Extraction d'information.

Dans cette partie, on travaille directement sur la liste de ligne lue avec `ouvre_fichier`. On pourra utiliser cet exemple (disponible dans Moodle) :

```
exemple1 : List[str] = ['"sport";"date";"participants";"vainqueur"\n',
                        '"boxe";2021-09-18;12;"Alice"\n',
                        '"boxe";2021-09-25;10;"Alice"\n',
                        '"karate";2021-09-26;19;"Carole"\n',
                        '"boxe";2021-10-02;8;"Bob"\n',
                        '"karate";2021-10-03;20;"Carole"\n',
                        '"tennis";2021-10-04;3;"Alice"\n',
                        '"boxe";2021-10-09;5;"Alice"\n',
                        '"karate";2021-10-10;20;"Damien"\n',
                        '"boxe";2021-10-16;6;"Carole"\n',
                        '"echecs";2021-09-17;120;"Bob"\n',
                        '"echecs";2021-09-24;120;"Bob"\n',
                        '"echecs";2021-10-01;120;"Carole"\n']
```

L'objectif est d'extraire des informations de ce fichier sous forme de dictionnaires.

Question 1. Ecrire une fonction `decompose_ligne`, qui prend en entrée une ligne `li` de fichier .csv (donc une chaine de caractères) et un caractère `sep` qui renvoie une liste de chaînes de caractères, correspondant au découpage de `li` selon le caractère `sep`. C'est-à-dire qu'on récupère dans la liste résultat les différentes sous-chaînes de la ligne qui se trouvent entre les `sep`. En outre, on fera en sorte de supprimer le dernier caractère de `li` (le retour chariot `"\n"`) dans la liste résultat.

Par exemple,

```
>>> decompose_ligne(exemple1[0], ";")
    ['"sport"', '"date"', '"participants"', '"vainqueur"']
>>> decompose_ligne(exemple1[3], ";")
    ['"karate"', '2021-09-26', '19', '"Carole"']
>>> decompose_ligne(exemple1[3], ",")
    ['"karate";2021-09-26;19;"Carole"']
```

On constate que si on décompose une ligne de `tournois.csv` selon `";"` on récupère une liste de 4 éléments (correspondant aux colonnes `"sport"`, `"date"`, `"participants"`, `"vainqueur"`), et que si on la décompose selon `","` on récupère une liste d'un seul élément (parce qu'il n'y a pas de `","` dans ce fichier).

Question 2. Ecrire une fonction `enleve_guillemets` qui prend une chaine de caractères `s`, si `s` commence et finit par des doubles guillemets `'"'`, elle renvoie l'intérieur de la chaîne (ce qu'il y a entre les guillemets), et sinon elle renvoie la chaîne telle quelle.

```
assert enleve_guillemets('"sport"') == 'sport'
assert enleve_guillemets('sport') == 'sport'
```

Question 3. Donner une définition **avec compréhension** de la fonction `enleve_guillemets_ligne` qui prend en entrée une liste de chaînes de caractères `li`, et qui renvoie une liste correspondant à `li` dans laquelle on a enlevé les guillemets `'"'` qui entoure les éléments de `li`, quand c'est le cas.

```
assert enleve_guillemets_ligne(['"sport"', '"date"', '"participants"', '"vainqueur"'])
       == ['sport', 'date', 'participants', 'vainqueur']
assert enleve_guillemets_ligne(['"karate"', '2021-09-26', '19', '"Carole"'])
       == ['karate', '2021-09-26', '19', 'Carole']
```

Question 4. Donner une définition **utilisant des compréhensions** de la fonction `lignes_propres` qui prend en entrée une liste de chaînes de caractères `lis` correspondant à la lecture d'un fichier .csv, et qui

renvoie une liste de liste de chaînes de caractère correspondant à la décomposition de chacune des lignes de `lis` dans laquelle on a enlevé les guillemets qui entourent chaque élément, quand c'est le cas.

```
>>> lignes_propres(exemple1, ";")
[['sport', 'date', 'participants', 'vainqueur'],
 ['boxe', '2021-09-18', '12', 'Alice'],
 ['boxe', '2021-09-25', '10', 'Alice'],
 ['karate', '2021-09-26', '19', 'Carole'],
 ['boxe', '2021-10-02', '8', 'Bob'],
 ['karate', '2021-10-03', '20', 'Carole'],
 ['tennis', '2021-10-04', '3', 'Alice'],
 ['boxe', '2021-10-09', '5', 'Alice'],
 ['karate', '2021-10-10', '20', 'Damien'],
 ['boxe', '2021-10-16', '6', 'Carole'],
 ['echecs', '2021-09-17', '120', 'Bob'],
 ['echecs', '2021-09-24', '120', 'Bob'],
 ['echecs', '2021-10-01', '120', 'Carole']]
```

Question 5. Ecrire une fonction partielle `cherche_indice` qui prend en entrée un élément `e`, une liste `li` et qui renvoie le premier indice de `li` auquel apparaît `e` si c'est le cas, et `None` sinon.

```
>>> cherche_indice("sport", ['sport', 'date', 'participants', 'vainqueur'])
0
>>> cherche_indice("vainqueur", ['sport', 'date', 'participants', 'vainqueur'])
3
>>> cherche_indice("deces", ['sport', 'date', 'participants', 'vainqueur'])
None
```

Question 6. Ecrire une fonction `dictionnaire_compte` qui prend en entrée une liste `lis` de liste de chaînes de caractères `lis` correspondant aux lignes "propres" (décomposée et sans guillemets) d'un fichier csv, une chaine de caractère `clef`, et qui renvoie un dictionnaire dont les clefs sont les données de la colonne `clef` du fichier et les valeurs sont leurs nombre d'occurences :

```
lignes_ex1 : List[List[str]] = lignes_propres(exemple1, ";")
>>> dictionnaire_compte(lignes_ex1, "vainqueur")
{'Alice': 4, 'Carole': 4, 'Bob': 3, 'Damien': 1}
>>> dictionnaire_compte(lignes_ex1, "sport")
{'boxe': 5, 'karate': 3, 'tennis': 1, 'echecs': 3}
```

Question 7. Ecrire une fonction `dictionnaire_somme` qui prend en entrée une liste `lis` de liste de chaînes de caractères `lis` correspondant aux lignes "propres" (décomposée et sans guillemets) d'un fichier csv, une chaine de caractère `clef`, une chaine de caractère `valeur`, et qui renvoie un dictionnaire dont les clefs sont les données de la colonne `clef` du fichier et la valeur associée à la clef `k` est la somme des données de la colonne `valeur` des lignes ou `k` apparaît dans la colonne `clef` :

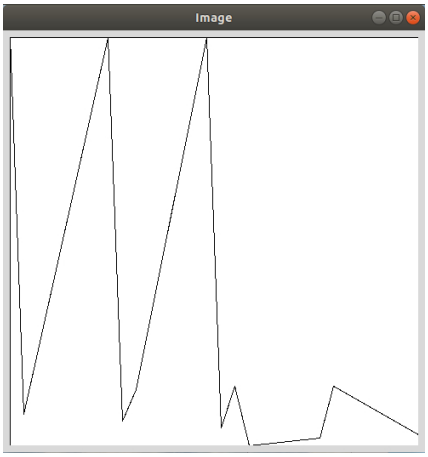
```
>>> dictionnaire_somme(lignes_ex1, "sport", "participants")
{'boxe': 41, 'karate': 59, 'tennis': 3, 'echecs': 360}
```

Ensuite. On pourra ensuite écrire des fonctions qui permettent d'utiliser ces dictionnaires (comme par exemple faire un classement des vainqueurs).

2 Suggestion : Courbes temporelles.

L'objectif de cette suggestion est de construire un graphique simpliste présentant l'évolution d'une donnée numérique en fonction du temps, à partir d'un CSV dont une colonne est une date.

Par exemple, avec les données de la partie précédente, on peut réaliser un graphique présentant l'évolution de l'affluence aux tournois en fonction de la date (tous sports confondus) :



Une démarche possible est la suivante :

- écrire des fonctions qui permettent de convertir les dates en durée (en nombre de jours) depuis une date initiale,
- convertir un dictionnaire associant date et valeur en dictionnaire associant durée depuis la première date du dictionnaire et valeur,
- convertir le dictionnaire en une liste d'associations (durée, valeur),
- trier la liste d'associations selon la durée (croissante),
- convertir les couples (durée, valeur) en points (abscisses, ordonnees) entre (-1,-1) et (1,1),
- tracer la courbe (comme dans les activités précédentes).

3 Suggestion : Projet.

Le but de cette activité est de proposer une étude sur des données réelles, récupérées sur un site d'*open data* (données ouvertes) officielles.

Par exemple, on pourra récupérer les données sur l'épidémie *Covid-19* sur le site du gouvernement : `https://www.data.gouv.fr/fr/pages/donnees-coronavirus/`

Un tel fichier est diponible directement sur Moodle, (mais beaucoup d'autres données sont à disposition gratuitement sur le Web, sous forme de CSV).

On pourra ensuite étudier cette information, et produire des *réutilisations* des données, comme des classements, ou des courbes temporelles.

Par exemple, voici la courbe temporelle du nombre de décès cumulés de l'épidémie de *Covid-19* en France depuis le début de l'épidémie jusqu'au 20 Novembre 2021, en utilisant le jeu de données téléchargeable sur Moodle :