Activité - Données en table

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 en table.

Le format CSV est standardisé par la RFC 4180, mais beaucoup de fichiers ne la respectent pas. On pourra consulter la page *Wikipédia* : `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, aussi appelés *descripteurs*.

Par exemple, voici le contenu d'un fichier CSV "francophone" `tournois.csv` 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 :

```
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"`.

Extraction d'information

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

```
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']
```

Question 1. Vérifier le bon fonctionnement de la fonction `ouvre_fichier` et écrire un test.

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. Ecrire une fonction `cherche_indice` qui prend en entrée une chaine de caractères `s`, une liste `li` et qui renvoie le premier indice de `li` auquel apparaît `s` 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("football", ['sport', 'date', 'participants', 'vainqueur'])
None
```

Question 5. 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"']
```