CSV

Fiche de synthèse

1 Le format CSV

Définition 1 **(Fichier CSV)** Un fichier **CSV** (valeurs séparées par des virgules) est un fichier texte qui contient une table : chaque ligne est un **enregistrement**, et ses valeurs sont séparées par un caractère, le **délimiteur**. La première ligne, l'**en-tête**, donne le nom des colonnes, appelées **descripteurs** ; un enregistrement donne une **valeur** à chaque descripteur.

Le fichier `meteo.csv` commence ainsi :

```
ville,région,température,pluie
Lyon,Auvergne-Rhône-Alpes,24,0
Brest,Bretagne,17,6
```

En France, les tableurs utilisent souvent le point-virgule, car la virgule écrit les nombres décimaux. Une valeur qui contient le délimiteur s'écrit entre guillemets : `"Paris, 5e"`.

2 Lire avec csv.reader

```
import csv
with open('meteo.csv', encoding='utf-8') as f:
    lecteur = csv.reader(f, delimiter=',')
    next(lecteur)           # passe l'en-tête
    total = 0
    nombre = 0
    for ligne in lecteur:
        total = total + int(ligne[2])
        nombre = nombre + 1
print(f'Moyenne : {total / nombre} degrés')
```

`csv.reader` donne chaque ligne sous forme de **liste** : `['Lyon', 'Auvergne-Rhône-Alpes', '24', '0']`. `next(lecteur)` saute une ligne, ici l'en-tête.

Le module `csv` donne toujours des **chaînes de caractères**, même pour des nombres : on convertit avec `int` ou `float` avant de calculer.

3 Lire avec csv.DictReader

```
with open('meteo.csv', encoding='utf-8') as f:
    lecteur = csv.DictReader(f, delimiter=',')
    for ligne in lecteur:
        ville = ligne["ville"]
        print(f'{ville} : {ligne["pluie"]} mm')
```

`csv.DictReader` donne chaque ligne sous forme de **dictionnaire** dont les clés sont les descripteurs. `ligne["pluie"]` est plus clair que `ligne[3]`, et reste juste si l'ordre des colonnes change.

4 Écrire

```
releves = [['ville', 'température'],
           ['Rennes', 18], ['Toulouse', 27]]
with open('releves.csv', 'w', encoding='utf-8',
          newline='') as f:
    ecrivain = csv.writer(f, delimiter=',')
    for ligne in releves:
        ecrivain.writerow(ligne)
```

`writerow` écrit une liste ; les nombres deviennent du texte. Le mode `'a'` ajoute des lignes à la fin d'un fichier existant ; `newline=''` évite une ligne vide après chaque enregistrement sous Windows. `csv.DictWriter(fichier, fieldnames=[...])` écrit des dictionnaires : `writeheader` écrit l'en-tête, puis `writerow` une ligne.

5 Une table en mémoire

```
def charger_csv(nom):
    with open(nom, encoding='utf-8') as f:
        return list(csv.DictReader(f))

table = charger_csv('meteo.csv')
seches = [ligne['ville'] for ligne in table
          if int(ligne['pluie']) == 0]
```

Sans `delimiter`, le délimiteur est la virgule. La table devient une **liste de dictionnaires**, une par ligne, qu'on peut parcourir, filtrer ou trier autant qu'on veut. Ici, `seches` vaut `['Lyon', 'Nice']`.

6 Rechercher avec plusieurs critères

```
ensoleillees = [
    ligne['ville'] for ligne in table
    if int(ligne['température']) > 25
    and int(ligne['pluie']) == 0
]
```

Avec `and`, une ligne est gardée si toutes les conditions sont vraies ; avec `or`, si au moins une l'est. Ici, `ensoleillees` vaut `['Nice']`.

⑦ Doublons et cohérence

Un **doublon** est un enregistrement présent deux fois : on le repère en gardant dans une liste `vus` les valeurs déjà rencontrées. Chaque descripteur a un **domaine de valeurs** : une valeur qui en sort (une pluie négative) signale une erreur de saisie.

⑧ Trier selon une colonne

```
def note(ligne):
    return ligne["note"]

par_note = sorted(eleves, key=note)
decroissant = sorted(eleves, key=note,
                     reverse=True)
```

Le paramètre `key` reçoit une fonction, écrite sans parenthèses : `sorted` l'applique à chaque ligne et trie selon les valeurs obtenues. `reverse=True` inverse l'ordre ; `max(eleves, key=note)` donne la ligne de meilleure note.

⑨ Fusionner deux tables

```
def fusionner(table1, table2, cle):
    resultat = []
    for ligne1 in table1:
        for ligne2 in table2:
            if ligne1[cle] == ligne2[cle]:
                nouvelle = {}
                for k in ligne1:
                    nouvelle[k] = ligne1[k]
                for k in ligne2:
                    nouvelle[k] = ligne2[k]
                resultat.append(nouvelle)
    return resultat
```

Les deux tables ont un descripteur commun, la **clé**, de même domaine de valeurs : deux lignes sont réunies quand elles ont la même valeur pour ce descripteur. Une ligne sans partenaire dans l'autre table n'est pas gardée.

À retenir.

- `csv.reader` donne des listes, `csv.DictReader` des dictionnaires.
- Les valeurs lues sont des chaînes : on les convertit avant de calculer.
- On recherche en combinant des conditions avec `and` et `or` ; on fusionne deux tables sur un descripteur commun.
- Avant d'exploiter une table, on vérifie les doublons et les domaines de valeurs.