
	Séquence 7 Données structurées au format csv Lecture Ecriture	1 / 3 
---	--	---

1 Objectif

Traiter les données issues de fichiers textes au format csv.

2 Description

Nous avons vu précédemment comment lire et écrire des données dans un fichier texte.

Le principe général reste le même mais globalement seules les instructions à utiliser changent.

Nous verrons aussi comment gérer quelques erreurs qui pourraient survenir lors du traitement de ces fichiers.

3 Travail demandé

3.1 Lecture d'un fichier csv

3.1.1 Méthode csv.reader

Vous disposez de deux fichiers au format csv :

- personnes.csv
- personnesCopie.csv

On donne le code python suivant :

```
import csv
with open('personnes.csv', mode = "r" , encoding='utf-8' , newline = '') as csvFile :
    contenu = csv.reader(csvFile)
    contenuLu = [ligne for ligne in contenu]

# Affichage du résultat :
print(contenuLu)
print("-----")
for ligne in contenuLu :
    print(ligne)
```

Question 1. Copier le code après avoir créé un fichier python vierge.

Question 2. Exécuter ce code et indiquer :

- Le format de la variable « contenuLu »
- Comment est exécuté le code

Voici quelques explications :

- **import** nous importons la bibliothèque csv.
- **open** ouvre le fichier en lecture (mode = "r" pour read) -> csvFile (objet de type objFile)
- **initialisation** du lecteur (reader) de csvFile.
- **lecture ligne par ligne** la boucle for permet d'énumérer les différentes lignes de csvFile.
- La première liste affichée correspondant au nom des entêtes des colonnes (les **descripteurs**)

Question 3. Modifier le programme afin de **créer** une fonction qui permet d'obtenir le même résultat.

Utiliser la variable « personnes » pour stocker les données du fichier. **Valider** la modification.

Question 4. Indiquer par quelles lignes de code on aurait pu remplacer :

« contenuLu = [ligne for ligne in contenu] »

Vous pouvez vous aider de la ressource ci-dessous pour répondre à la question.

Ressources : [accès](#)

On met à votre disposition un deuxième fichier texte « *personnesCopie.csv* ».

Question 5. **Modifier** votre programme afin de lire ce fichier csv tout en conservant la lecture du premier. **Stocker** les données lues dans la variable « *personnesCopie* ».

Question 6. **Ajouter** ensuite la ligne de code suivante :
`assert personnes == personnesCopie`

Question 7. **Valider** la modification. Que peut-on constater ? Quel type d'erreur est affiché ? **Justifier** cette erreur.

Question 8. **Apporter** les modifications pour que cette erreur disparaisse. **Valider**.

3.1.2 Méthode csv.DictReader

On donne le code python suivant :

```
import csv
with open('personnes.csv', mode = "r" , newline = '') as csvFile :
    reader = csv.DictReader(csvFile)
    contenuLu = [dict(ligne) for ligne in reader]

# Affichage du résultat :
print(contenuLu)
```

Question 9. **Copier** le code après avoir créé un fichier python vierge.

Question 10. **Exécuter** ce code et indiquer le format de la variable « contenuLu ».

Question 11. **Modifier** le programme afin d'afficher le détail de cette variable.

Question 12. **Modifier** le programme afin de **créer** une fonction qui permet d'obtenir le même résultat. **Utiliser** la variable « personnes » pour stocker les données du fichier. **Valider** la modification.

Question 13. **Comparer** les résultats obtenus avec les deux méthodes « csv.reader » et « csv.DictReader ».

3.1.3 Quelques commentaires

La première méthode permet de récupérer les données contenues dans le fichier csv sous la forme d'une liste de listes [[.....], [.....], [.....]...]. La deuxième permet quant à elle de récupérer les données contenues dans le fichier csv sous la forme d'une liste de dictionnaires [{.....}, {.....}, {.....} ...].

Une telle structure est mieux adaptée. D'une part, les en-têtes (appelés **descripteurs**) ne sont plus lus comme une ligne banale mais sont utilisées comme clefs dans les dictionnaires.

Dans un fichier comportant beaucoup de colonnes, il est assez fastidieux de se souvenir à quoi correspond l'indice d'une colonne.

D'autre part, si chaque élément listeDico[i] est un dictionnaire dont les clés sont les entêtes de colonnes, il sera bien plus facile de les manipuler.

3.1.4 Extraire des données suivant certains critères

Question 14. **Compléter** la ligne de code suivante :

`contenuLu = [dict(ligne) for ligne in reader]` par
`if dict(ligne)['age']>'50'` que vous placerez avant le crochet de fermeture.

Question 15. **Valider** la modification, que constatez-vous ?

Question 16. **Modifier** la condition pour que seules les personnes dont le nom de famille soit *Lenard* soient extraites. Pour cela **copier** puis **coller** la ligne de code modifiée précédemment, et la **modifier**. **Passer** la ligne de code précédente en commentaire. **Valider** la modification.

Question 17. **Rajouter** à la suite de `dict(ligne)` le critère suivant `['prenom']`. **Valider** la modification, que constatez-vous ?

3.1.5 Manipuler les données

Question 18. **Extraire** une liste des âges de toutes les personnes du fichier. **Calculer** puis **afficher** l'âge moyen en utilisant les deux méthodes suivantes :

- Méthode classique par des opérations mathématiques d'addition et de division.
- Méthode plus évoluée en utilisant l'instruction `sum()` et une compréhension de liste pour convertir en une seule fois les éléments chaîne de caractère en entier.

