

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

# Applications Android

Christophe Viroulaud

Première NSI

# Un moteur de recherche pour aider l'utilisateur à faire son choix.

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

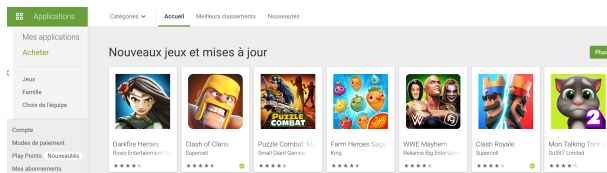


FIGURE – Magasin d'applications

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

**Options de séparateur**

☐ Largeur fixe ☒ Séparé par

☒ Tabulation ☒ Virgule ☒ Point-virgule ☐ Espace ☐ Autre

☐ Fusionner les séparateurs ☐ Espaces superflus

Séparateur de chaîne de caractères :  ▼

**Autres options**

☐ Formater les champs entre guillemets comme texte ☐ Détecter les nombres spéciaux

**Champs**

Type de colonne :  ▼

	Standard	Standard	Standard	Standard
1	Name	Category	Rating	Reviews
2	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
3	Coloring book moana	ART_AND_DESIGN	3.9	967
4	U Launcher Lite - FREE Live Cool Themes, Hide Apps	ART_AND_DESIGN	4.7	87510
5	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644
6	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967
7	Paper flowers instructions	ART_AND_DESIGN	4.4	167
8	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178
9	Infinite Painter	ART_AND_DESIGN	4.1	26815

FIGURE – Données dans un fichier texte

Comment manipuler un jeu de données ?

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

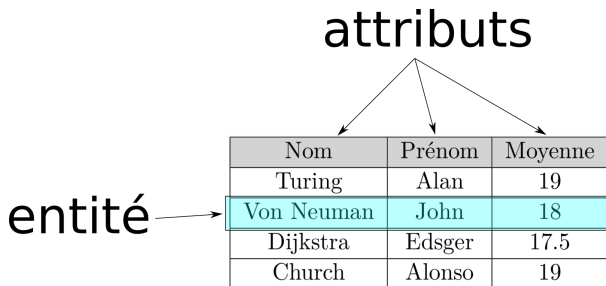


FIGURE – Table de données

# csv (Comma Separated Values)

**Options de séparateur**

☐ Largeur fixe
 ☒ Séparé par

☒ Tabulation
 ☒ Virgule
 ☒ Point-virgule
 ☐ Espace
 ☐ Autre

☐ Fusionner les séparateurs
 ☐ Espaces superflus
 Séparateur de chaîne de caractères : " ▼

**Autres options**

☐ Formater les champs entre guillemets comme texte
 ☐ Détecter les nombres spéciaux

**Champs**

Type de colonne : ▼

	Standard	Standard	Standard	Standard
	Name	Category	Rating	Reviews
1	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159
2	Coloring book moana	ART_AND_DESIGN	3.9	967
3	U Launcher Lite - FREE Live Cool Themes, Hide Apps	ART_AND_DESIGN	4.7	87510
4	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644
5	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967
6	Paper flowers instructions	ART_AND_DESIGN	4.4	167
7	Smoke Effect Photo Maker - Smoke Editor	ART_AND_DESIGN	3.8	178
8	Infinite Painter	ART_AND_DESIGN	4.1	26815

FIGURE – Fichier csv ouvert avec LibreOffice

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

# csv (Comma Separated Values)

"Turing";	"Alan";	"19"
"Von Neuman";	"John";	"18"
"Dijkstra";	"Edsger";	"17.5"
"Church";	"Alonso";	"19"

- ▶ Chaque ligne représente un nouvel élément du jeu de données.
- ▶ Virgules, point-virgules ou tabulations.
- ▶ Attributs pas forcément présents.
- ▶ Guillemets pour encapsuler les données.

## Problématique

### Données en table

#### Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Activité 1 :

1. Télécharger et décompresser l'annexe *googleplaystore.zip* sur le site <https://cviroulaud.github.io>
2. Ouvrir le fichier *googleplaystore.csv* avec un tableur (LibreOffice).
3. Repérer les *attributs*.

## Problématique

## Données en table

### Présentation

Lire un fichier csv

## Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

- ▶ *Name* : nom,
- ▶ *Category* : catégorie d'application,
- ▶ *Rating* : note (sur 5),
- ▶ *Reviews* : nombre de commentaires,
- ▶ *Installs* : nombre d'installations.



# Lire un fichier externe avec Python

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 # ouvrir le fichier
2 fichier = open("notes.csv")
3
4 # utiliser le fichier
5
6 # libérer le fichier
7 fichier.close()
```

Code 1 – Ouvrir et fermer

# Lire un fichier externe avec Python

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 import csv
2 # ouvrir le fichier
3 fichier = open("notes.csv")
4
5 lecteur = csv.reader(fichier)
6 for ligne in lecteur:
7     print(ligne)
8
9 # libérer le fichier
10 fichier.close()
```

Code 2 – Méthode pour itérer sur les données

## Activité 2 :

1. Tester le code [2](#).
2. Remplacer la méthode `reader` par `DictReader`.
3. Quel itérateur semble le plus adapté ?
4. Créer un programme *appgoogle.py*.
5. Dans le programme, ouvrir le fichier *googleplaystore.csv*.
6. Créer un tableau de dictionnaires à partir des données du fichier *csv*.

```
1 lecteur1 = csv.reader(fichier)
2 for ligne in lecteur1:
3     print(ligne)
```

['nom', 'prenom', 'moyennes']

['Turing', 'Alan', '19']

['Von Neuman', 'John', '18']

['Dijkstra', 'Edsger', '17.5']

['Church', 'Alonso', '19']

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 lecteur2 = csv.DictReader(fichier)
2 for ligne in lecteur2:
3     print(ligne)
```

OrderedDict([('nom', 'Turing'), ('prenom', 'Alan'),  
('moyennes', '19')])

OrderedDict([('nom', 'Von Neuman'), ('prenom', 'John'),  
('moyennes', '18')])

OrderedDict([('nom', 'Dijkstra'), ('prenom', 'Edsger'),  
('moyennes', '17.5')])

OrderedDict([('nom', 'Church'), ('prenom', 'Alonso'),  
('moyennes', '19')])

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

- ▶ On utilisera préférentiellement la méthode **DictReader** : chaque donnée est étiquetée.
- ▶ **reader** et **DictReader** sont des itérateurs : on ne peut les parcourir qu'une fois

## Commentaire

Une variable de type *OrderedDict* sera vue comme un simple dictionnaire.

```
1 {"nom": "Turing", "prenom": "Alan", "moyennes":  
  "19"}  
2 {"nom": "Von Neuman", "prenom": "John", "  
  moyennes": "18"}  
3 {"nom": "Dijkstra", "prenom": "Edsger", "  
  moyennes": "17.5"}  
4 {"nom": "Church", "prenom": "Alonso", "moyennes"  
  ": "19"}
```

Code 3 – OrderedDict → dict

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 import csv
2 # charge le fichier dans le programme
3 fichier = open("googleplaystore.csv")
4
5 # crée un itérateur sur les données
6 lecteur_donnees = csv.DictReader(
    fichier)
7
8 table = []
9 for ligne in lecteur_donnees:
10     table.append(ligne)
11
12 # libère le fichier externe
13 fichier.close()
```

Code 4 – Liste de OrderedDict



## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 import csv
2 # charge le fichier dans le programme
3 fichier = open("googleplaystore.csv")
4
5 # crée un itérateur sur les données
6 lecteur_donnees = csv.DictReader(
    fichier)
7
8 table = list(lecteur_donnees)
9
10 # libère le fichier externe
11 fichier.close()
```

Code 5 – Liste de OrderedDict - seconde méthode

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 import csv
2 # charge le fichier dans le programme
3 fichier = open("googleplaystore.csv")
4 # crée un itérateur sur les données
5 lecteur_donnees = csv.DictReader( fichier )
6
7 table = []
8 # Pour chaque ligne
9 for ligne in lecteur_donnees:
10     dico = {}
11     # Pour chaque couple de la ligne
12     for nom, val in ligne.items():
13         dico[nom] = val
14     table.append(dico)
15 # libère le fichier externe
16 fichier.close()
```

Code 6 – liste de dictionnaires

## Problématique

## Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

## Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

Par défaut les données chargées dans un programme par la bibliothèque `csv` sont des chaînes de caractère.

**Activité 3 :** Modifier le programme précédent pour typer correctement les informations récoltées.

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 table = []
2 for ligne in lecteur_donnees:
3     dico = {}
4     for nom, val in ligne.items():
5         # validation des données
6         if nom == "Rating":
7             val = float(val)
8         if nom == "Installs" or nom == "Reviews":
9             val = int(val)
10
11     dico[nom] = val
12     table.append(dico)
```

Code 7 – Tentative de conversion des données

ValueError : invalid literal for int() with base 10 : 'NaN'

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

**Valider les données**

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

Name	Category	Rating	Reviews	Installs
CX Network	BUSINESS	NaN	0	NaN

Tableau – Des valeurs particulières

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

**Valider les données**

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 table = []
2 for ligne in lecteur_donnees:
3     dico = {}
4     for nom, val in ligne.items():
5         if nom == "Rating":
6             if val == "NaN":
7                 val = -1.0
8             else:
9                 val = float(val)
10        if nom == "Installs":
11            if val == "NaN":
12                val = 0
13            else:
14                val = int(val)
15        dico[nom] = val
16    table.append(dico)
```

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

#### Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

Une action courante sur un jeu de données est de sélectionner certaines lignes en fonction d'un critère.

## Activité 4 :

1. Écrire la fonction `trouver_app(mot_cle: str, tab: list) → list` qui renvoie la liste des applications du tableau `tab` dont le nom contient `mot_cle`.

Indication : L'instruction `in` permet de vérifier si une sous-chaîne est présente dans une chaîne de caractère.

2. Chercher toutes les applications dont le nom contient le mot *Photo*.
3. Compter le nombre d'applications renvoyées.



## Problématique

### Données en table

Présentation  
Lire un fichier csv

### Manipuler les données

Valider les données  
Rechercher des données  
**Sélectionner**  
Agréger  
Trier des données  
Tri natif  
Clé de tri

```
1 def trouver_app(mot_cle: str, tab: list) -> list:
2     """
3     renvoie les applications contenant le 'mot_cle'
4
5     chaque mot commence par une majuscule dans la
6     table
7     """
7     res = []
8     for app in tab:
9         if mot_cle in app["Name"] :
10             res.append(app)
11     return res
```

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

**Sélectionner**

Agréger

Trier des données

Tri natif

Clé de tri

```
1 applications = trouver_app("Photo", table)
2 len( applications )
```

## Activité 4 :

4. Écrire la fonction `meilleur_app_notee(tab: list) → dict` qui renvoie l'application la mieux notée du tableau *tab*.
5. Trouver l'application photo la mieux notée.
6. Pour les plus avancés : en s'aidant de la documentation, modifier la fonction `trouver_app` pour quelle :
  - ▶ ne prenne pas en compte la casse des mots,
  - ▶ renvoie les applications contenant plusieurs mots-clés ; les mots sont passés à la fonction par le paramètre *mot\_cle* séparés par un espace.

```
1 def meilleur_app_notee(tab: list ) -> dict:
2     """
3     renvoie l' application avec la meilleure note
4     de tab
5     """
6     note_maxi = 0
7     meilleure_app = None
8     for app in tab:
9         if app["Rating"] > note_maxi:
10             note_maxi = app["Rating"]
11             meilleure_app = app
12     return meilleure_app
```

```
1 applications = trouver_app("Photo", table)
2 meilleur_app_notee( applications )
```

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

On peut imaginer proposer à l'utilisateur toutes les applications notées au-dessus de la moyenne.

## Activité 5 :

1. Écrire la fonction `moyenne_note(tab: list)` → `float` qui calcule la note moyenne des applications de *tab*. Le résultat sera arrondi à deux chiffres significatifs.
2. Dans le programme principal, construire *par compréhension* le tableau des applications photo dont la note est strictement supérieure à la moyenne.

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

**Agréger**

Trier des données

Tri natif

Clé de tri

```
1 def moyenne_note(tab: list) -> float:
2     """
3     renvoie la note moyenne des apps de tab
4     """
5     somme = 0
6     nb = 0
7     for app in tab:
8         # Si l'app a déjà été notée
9         if app["Rating"] >= 0 :
10             somme += app["Rating"]
11             nb += 1
12     return round(somme/nb, 2)
```

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

**Agréger**

Trier des données

Tri natif

Clé de tri

```
1 applications = trouver_app("Photo", table)
2
3 moyenne = moyenne_note(applications)
4
5 meilleur_app = [app for app in applications if
                  app["Rating"] > moyenne]
```

Code 9 – Sélection par compréhension

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

#### **Trier des données**

Tri natif

Clé de tri

Le tri est une autre opération fréquemment exécutée sur un jeu de données. On peut imaginer dans notre cas, ordonner les applications photo en fonction de leur note.



En tant que langage de haut-niveau Python offre un outil de tri efficace :

- ▶ la méthode `sort` trie *en place* un itérable,

```
1 ma_liste.sort()
```

- ▶ la fonction `sorted` crée un nouvel itérable trié.

```
1 nouvelle = sorted(ma_liste)
```

Dans notre cas les objets triés ne sont pas des simples entiers mais des dictionnaires. Il faut alors préciser la **clé de tri**.

```
1 def parametres_tri(eleve : dict) -> float:
2     """
3     renvoie le paramètre utilisé pour le tri
4     """
5     return eleve["moyennes"]
6
7 eleves.sort(key=parametres_tri)
```

```
'nom' : 'Dijkstra', 'prenom' : 'Edsger', 'moyennes' : 17.5
'nom' : 'Von Neuman', 'prenom' : 'John', 'moyennes' : 18.0
'nom' : 'Turing', 'prenom' : 'Alan', 'moyennes' : 19.0
'nom' : 'Church', 'prenom' : 'Alonso', 'moyennes' : 19.0
```

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Activité 6 :

1. Écrire la fonction `parametres_tri_1(app: dict) → float` qui renvoie la note de l'application.
2. Trier le tableau des applications photo en fonction de leur note.
3. Afficher les cinq meilleures applications.

## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 def parametres_tri_1(app: dict) -> float:
2     """
3     renvoie le Rating de l'app
4     """
5     return app["Rating"]
```

```
1 apps_triees = sorted( applications , key=parametres_tri_1)
2 for i in (range(5)):
3     print(f"Top {i+1} :", apps_triees[i][ "Name"])
```

```
1 Top 1: Simple Photo BG Changer
2 Top 2: FG Autumn Photo Puzzle
3 Top 3: Blender BG - Photo Blend With
    Background
4 Top 4: Best CG Photography
5 Top 5: Life Made WI-Fi Touchscreen
    Photo Frame
```

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Problématique

## Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

**Activité 6 :** Pour départager les applications avec la même note, on choisit de définir un second paramètre de tri : le nombre de commentaires.

4. Écrire la fonction `parametres_tri_2(app: dict) → tuple` nouvelle clé de tri.
5. Appliquer cette nouvelle clé.

```
1 def parametres_tri_2(app: dict) -> tuple:
2     """
3     renvoie le tuple (Rating, Reviews) de l'app
4     """
5     return (app["Rating"], app["Reviews"])
```

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 Top 1: FG Autumn Photo Puzzle
2 Top 2: Blender BG - Photo Blend With
   Background
3 Top 3: Simple Photo BG Changer
4 Top 4: Best CG Photography
5 Top 5: Life Made WI-Fi Touchscreen
   Photo Frame
```

Problématique

Données en table

Présentation

Lire un fichier csv

Manipuler les  
données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

## Activité 6 :

6. Pour les plus avancés : Reprendre la fonction `tri_insertion` et la modifier pour effectuer le tri de la première question.



## Problématique

### Données en table

Présentation

Lire un fichier csv

### Manipuler les données

Valider les données

Rechercher des données

Sélectionner

Agréger

Trier des données

Tri natif

Clé de tri

```
1 def tri_insertion (tab: list ) -> None:
2     """
3     tri le tableau dans l'ordre croissant des notes
4     """
5     for i in range(len(tab)):
6         # mémoriser
7         en_cours = tab[i]
8         pos = i
9         # décaler
10        while pos > 0 and en_cours["Rating"] < tab[pos
11            -1]["Rating"] :
12            tab[pos] = tab[pos-1]
13            pos = pos-1
14        # insérer
15        tab[pos] = en_cours
```