mettre googleplaystore.zip sur site

googleplaystore.csv, specifications-app.csv, notes.csv

Applications Android
Christophe Viroulaud
Première NSI

Applications Android

Christophe Viroulaud

Première NSI

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

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

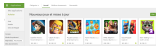


FIGURE – Magasin d'applications

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

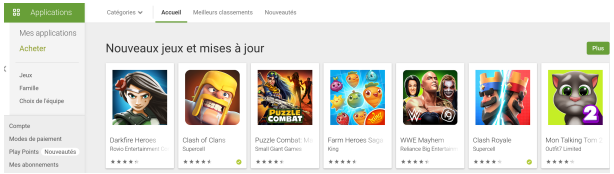


FIGURE – Magasin d'applications

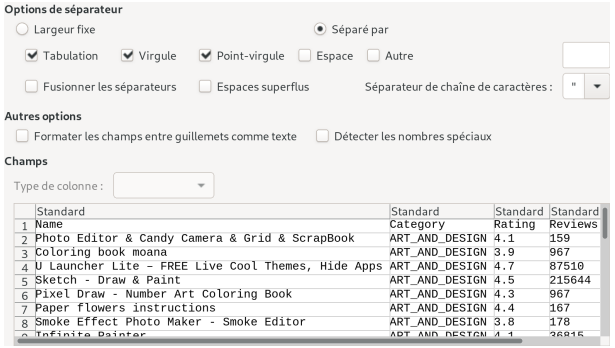
dataset

Jeu de données

Figure – Données dans un fichier texte

Comment manipuler un jeu de données ?

Jeu de données



	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	36815

Figure – Données dans un fichier texte

Comment manipuler un jeu de données ?

Vocabulaire des bases de données

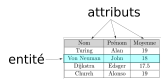


FIGURE – Table de données

Vocabulaire des bases de données

terme BDD : entité, table (ou relation), attributs

attributs

entité

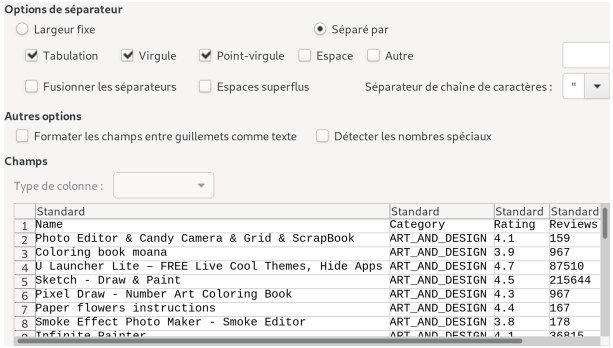
Nom	Prénom	Moyenne
Turing	Alan	19
Von Neuman	John	18
Dijkstra	Edsger	17.5
Church	Alonso	19

FIGURE – Table de données

csv (Comma Separated Values)

Figure – Fichier csv ouvert avec LibreOffice

csv (Comma Separated Values)



	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	36815

Figure – Fichier csv ouvert avec LibreOffice

Que se passe-t-il s'il y a des virgules dans une cellule ? → guillemets

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.

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.

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*.

Correction

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

Correction

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

Lire un fichier externe avec Python

Code 1 – Ouvrir et fermer

Lire un fichier externe avec Python

```
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

Code 2 – Méthode pour itérer sur les données

Lire un fichier externe avec Python

Pour lire un fichier externe dans un programme Python, il faut d'abord ouvrir ce fichier à l'aide de la commande **open**. Ensuite la bibliothèque csv propose plusieurs méthodes pour itérer sur les lignes du fichier ouvert.

```
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*.

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*.

Correction

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

['nom', 'prenom', 'moyennes']
['Turing', 'Alan', '19']
['Von Neuman', 'John', '18']
['Dijkstra', 'Edsger', '17.5']
['Church', 'Alonso', '19']

Correction

```
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']

Correction

```
lecteur2 = csv.DictReader(fichier)
for ligne in lecteur2:
    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')])

Correction

```
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')])

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

Correction

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

Correction
Commentaire
Une variable de type *OrderedDict* sera vue comme un simple dictionnaire.
Code 3 – OrderedDict → dict

Correction

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

Correction

```
import csv
# charge le fichier dans le programme
fichier = open("googleplaystore.csv")

# crée un itérateur sur les données
lecteur_donnees = csv.DictReader(
    fichier)

table = []
for ligne in lecteur_donnees:
    table.append(ligne)

# libère le fichier externe
fichier.close()
```

Code 4 – Liste de OrderedDict

Correction

```
import csv
# charge le fichier dans le programme
fichier = open("googleplaystore.csv")

# crée un itérateur sur les données
lecteur_donnees = csv.DictReader(
    fichier)

table = []
for ligne in lecteur_donnees:
    table.append(ligne)

# libère le fichier externe
fichier.close()
```

Code 4 – Liste de OrderedDict

Correction

```
import csv
# charge le fichier dans le programme
fichier = open("googleplaystore.csv")

# crée un itérateur sur les données
lecteur_donnees = csv.DictReader(
   fichier)

table = list(lecteur_donnees)

# libère le fichier externe
fichier.close()
```

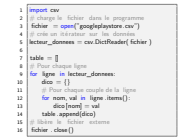
Code 5 – Liste de OrderedDict - seconde méthode

Correction

```
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

Correction



Code 6 – liste de dictionnaires

Correction

```
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

note → flottants, Installs et Reviews →entiers
booléan possible aussi (payante/gratuite)

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.

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.

Correction

Code 7 – Tentative de conversion des données

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

Correction

```
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'

Correction

Name	Category	Rating	Reviews	Installs
CX Network	BUSINESS	NaN	0	NaN

Tableau – Des valeurs particulières

1. Installs : NaN →0
2. Rating : NaN → -1 (pas notée)

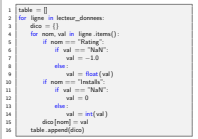
Correction

Name	Category	Rating	Reviews	Installs
CX Network	BUSINESS	NaN	0	NaN

Tableau – Des valeurs particulières

non noté = -1

Correction



Code 8 – Conversion des données

Correction

```
table = []
for ligne in lecteur_donnees:
    dico = {}
    for nom, val in ligne.items():
        if nom == "Rating":
            if val == "NaN":
                val = -1.0
            else:
                val = float(val)
        if nom == "Installs":
            if val == "NaN":
                val = 0
            else:
                val = int(val)
        dico[nom] = val
    table.append(dico)
```

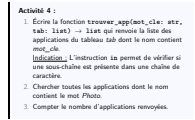
Code 8 – Conversion des données

Imiter un moteur de recherche

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

Imiter un moteur de recherche

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.

Activité 4 :

1. Écrire la fonction **`trouver_app(mot_cle: str, tab: list)`** $\rightarrow$ **`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.

Correction

Correction

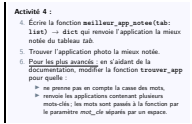
```
1  def trouver_app(mot_cle: str, tab: list) -> list:
2      """
3      renvoie les applications contenant le 'mot_cle'
4
5      chaque mot commence par une majuscule dans la
           table
6      """
7      res = []
8      for app in tab:
9          if mot_cle in app["Name"]:
10             res.append(app)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

Correction

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

Correction

```
applications  = trouver_app("Photo", table)
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.

Activité 4 :

4. Écrire la fonction `meilleur_app_notee(tab: list)` $\rightarrow$ `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.

Correction

```
def meilleur_app_notee(tab: list ) -> dict:
    """
    renvoie l'application avec la meilleure note de tab
    """
    note_maxi = 0
    meilleure_app = None
    for app in tab:
        if app["Rating"] > note_maxi:
            note_maxi = app["Rating"]
            meilleure_app = app
    return meilleure_app
```

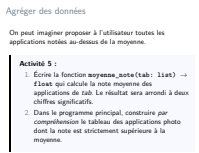
```
applications = trouver_app("Photo", table)
meilleur_app_notee(applications)
```

Correction

```
def meilleur_app_notee(tab: list ) -> dict:
    """
    renvoie l'application avec la meilleure note de tab
    """
    note_maxi = 0
    meilleure_app = None
    for app in tab:
        if app["Rating"] > note_maxi:
            note_maxi = app["Rating"]
            meilleure_app = app
    return meilleure_app
```

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

La sélection précédente étant très restrictive, il peut être intéressant d'offrir un choix plus large.


Agréger des données

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.

Agréger des données

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.

Correction

Correction

```
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)
```

Correction

Code 9 – Sélection par compréhension

Correction

```
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

Trier

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.

Trier

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)
```

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)
```

Certains élèves ont la même note →second critère ?

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 utilis é 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

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.

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.

Correction

Correction

```python
def parametres_tri_1(app: dict) -> float:
    """
    renvoie le Rating de l'app
    """
    return app["Rating"]
```

Correction

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

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

Correction

```
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
```

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.

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.

Correction

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

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

Correction

```
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"])
```

```
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
```

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

Activité 6 :

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

Correction

Correction

```python
def tri_insertion (tab: list ) -> None:
    """
    tri le tableau dans l'ordre croissant des notes
    """
    for i in range(len(tab)):
        # mémoriser
        en_cours = tab[i]
        pos = i
        # décaler
        while pos > 0 and en_cours["Rating"] < tab[pos
            -1]["Rating"] :
            tab[pos] = tab[pos-1]
            pos = pos-1
        # insérer
        tab[pos] = en_cours
```