1 Problématique

Devant la multitude d'applications disponibles, il peut être difficile d'effectuer son choix. Des critères (notes, commentaires…) permettent cependant de les classer et un moteur de recherche aidera l'utilisateur à faire son choix.

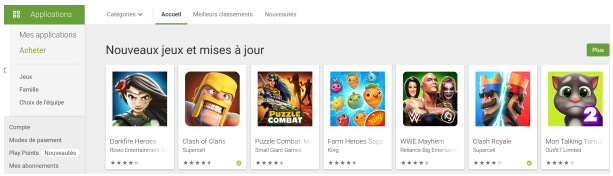

FIGURE 1 – Magasin d'applications

Les informations sont stockées dans un *jeu de données*. Cela peut être un simple fichier texte où chaque ligne contiendra les informations pour une application.

Comment manipuler un jeu de données ?

2 Données en table

2.1 Présentation

Il est courant de représenter des données sous forme de *table*.

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

Tableau 1 – Moyennes des élèves de NSI

Dans le tableau 1 la première ligne représente les **attributs** de la table. Ensuite chaque ligne représente un élément (ici un élève) du jeu de données.
Pour stocker ces informations on peut utiliser le format de fichiers *csv*.

À retenir

Le format **csv (Comma Separated Values)** permet de stocker des données tabulaires sous forme de valeurs séparées par des *virgules*. Chaque ligne représente un nouvel élément du jeu de 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*.

Christophe Viroulaud

2.2 Lire un fichier *csv*

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 1 – Méthode pour itérer sur les données

Activité 2 :

1. Tester le code 1 dans un fichier *notes.py*.
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*.

Commentaire

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

3 Manipuler les données

3.1 Valider les données

Par défaut les données chargées dans un programme par la bibliothèque **csv** sont des chaînes de caractère. Cependant, certaines des informations comme la note, sont des nombres.

Activité 3 :

1. Modifier le programme *notes.py*, pour que la moyenne soit stockée comme un flottant.
2. Modifier le programme *appgoogle.py* pour typer correctement les informations récoltées.

3.2 Rechercher des données

Une action courante sur un jeu de données est de sélectionner certaines lignes en fonction d'un critère. On peut assimiler cette action au principe général d'un moteur de recherche.

Christophe Viroulaud

3.2.1 Sélectionner

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.
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.

3.2.2 Agréger

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

3.3 Trier des données

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.

3.3.1 Tri natif

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()
2
```

— la fonction `sorted` crée un nouvel itérable trié.

```
1 nouvelle = sorted(ma_liste)
2
```


Christophe Viroulaud

3.3.2 Clé de tri

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)
```

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.

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

Christophe Viroulaud