# Projet collaboratif Python avec Git : Gestion d'une bibliothèque

## Description

Ce projet est un **exercice collaboratif** en binôme qui combine :

- la pratique de **Python avancé** (listes, tuples, dictionnaires, boucles, map, filter, lambda, zip, sort, etc.),
- l'utilisation de **Git en équipe** (branches, commits, merge, résolution de conflits).

Vous allez construire un programme qui gère une **bibliothèque virtuelle** avec des utilisateurs et des livres.

## Organisation du projet

```
bibliotheque-projet/

- data.py          : données de départ (utilisateurs, livres)

- users.py         : code de l'Apprenant A (gestion utilisateurs)

- books.py         : code de l'Apprenant B (gestion livres)

- main.py          : fichier principal qui assemble tout

- README.md        : consignes et documentation
```

## Répartition des rôles

- **Apprenant A (branche `users`)** → Gestion des **utilisateurs**
- **Apprenant B (branche `books`)** → Gestion des **livres et statistiques**

## Données de départ (`data.py`)

Les données sont déjà fournies :

```
utilisateurs = [
    (1, "Alice", "Dupont", 25),
    (2, "Bob", "Martin", 17),
    (3, "Clara", "Durand", 32),
    (4, "David", "Petit", 20)
]


livres = [
    {"titre": "1984", "auteur": "George Orwell", "année": 1949},
    {"titre": "Le Petit Prince", "auteur": "Antoine de
Saint-Exupéry", "année": 1943},
    {"titre": "Harry Potter", "auteur": "J.K. Rowling", "année":
1997}
]


aime_livres = [
    (1, "1984"),
    (1, "Le Petit Prince"),
    (3, "Harry Potter"),
    (4, "1984")
]
```

# Travail attendu

## Apprenant A (`users.py`)

1. Filtrer les **utilisateurs majeurs** (`filter`).
2. Formater les noms complets en **majuscules** (`map`, `lambda`).
3. Créer un dictionnaire associant chaque utilisateur à ses livres aimés.

- Générer un résumé par utilisateur :

```
"ALICE DUPONT (25 ans) aime : '1984', 'Le Petit Prince'"
```

---

## Apprenant B (`books.py`)

1. Trier les livres par **année de publication** (`sort`).
2. Identifier le **plus ancien** et le **plus récent**.

- Construire un dictionnaire :

```
{ "1984": 2, "Le Petit Prince": 1, "Harry Potter": 1 }
```

3. Afficher les utilisateurs **2 par 2** (pagination avec `while`).

---

## Après fusion (`main.py`)

- Importer et appeler les fonctions de `users.py` et `books.py`.
- Afficher un récapitulatif complet.
- Bonus collaboratif :
    - Fonction de **recherche** (livre ou utilisateur).
    - Utiliser `zip` pour créer une liste `(Nom complet, âge)` triée par âge.

---

# Utilisation de Git

## 1. Mise en place

```
git clone <url-du-depot>

cd bibliotheque-projet
```

## 2. Création des branches

**Apprenant A :**

```
git checkout -b users

git push -u origin users
```

**Apprenant B :**

```
git checkout -b books

git push -u origin books
```

## 3. Workflow quotidien

```
git add .

git commit -m "Description claire du changement"

git push
```

## 4. Fusion des branches

Revenir sur `main` et fusionner :

```
git checkout main

git pull

git merge users

git merge books
```

Résoudre les conflits si nécessaire, puis :

```
git add .
git commit -m "Résolution des conflits"
git push
```