2025

Mini projet d’application Système de Gestion de Bibliothèque

Réalisé par : Safae Bellafkih
Professeur : Haja Zakaria
Filière : Génie informatique 3

❖ Présentation du projet

Ce mini-projet a été réalisé dans le cadre d'un apprentissage approfondi de la **programmation orientée objet en Python**. Il consiste à développer un système de gestion de bibliothèque permettant de suivre les livres, les membres et les opérations d'emprunt ou de retour. Le projet intègre à la fois une interface en ligne de commande pour une interaction directe, et une interface graphique réalisée avec **Tkinter**. De plus, des visualisations statistiques sont générées à l'aide de la bibliothèque **Matplotlib**, afin d'analyser l'activité de la bibliothèque. Le projet est structuré en modules pour garantir une organisation claire du code, avec une attention portée à la robustesse et à la modularité.

❖ Objectifs

Ce projet vise à mettre en œuvre les principaux concepts de la programmation **Python** à travers le développement d'un système complet de gestion de bibliothèque. Les objectifs sont les suivants :

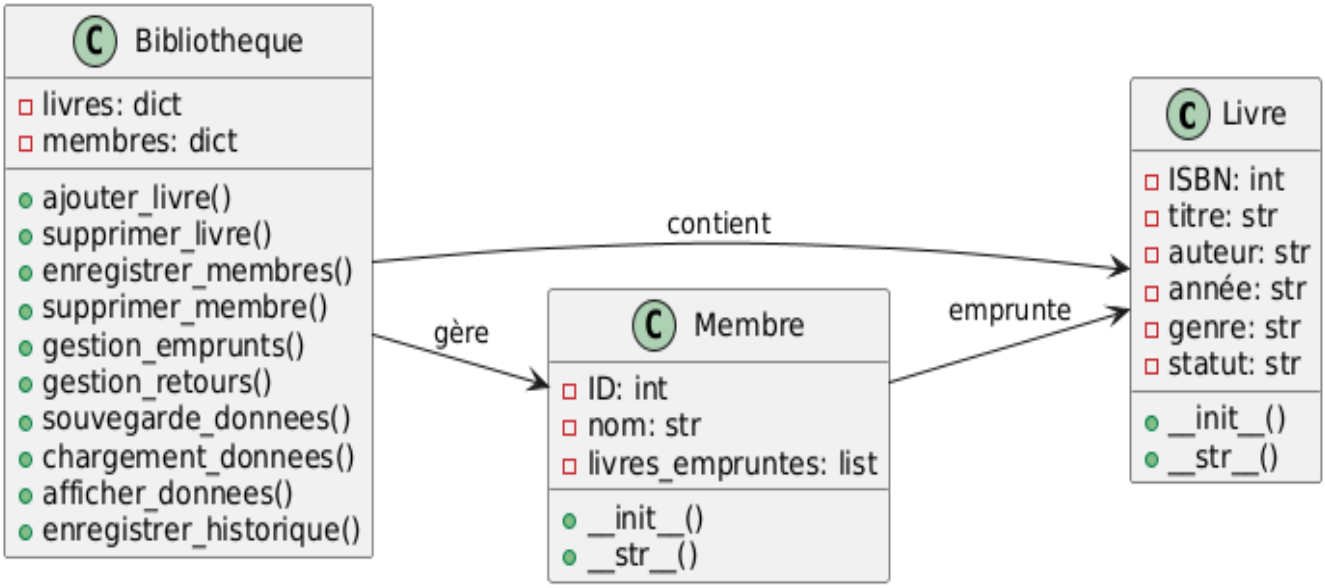
- ✓ Concevoir une application fonctionnelle et intuitive pour gérer les opérations courantes d'une bibliothèque (ajout, suppression, modification de livres et de membres).
- ✓ Implémenter un système d'emprunt et de retour de livres respectant les règles de disponibilité et de quota.
- ✓ Assurer la persistance des données à l'aide de fichiers texte (.txt) et CSV (.csv) pour conserver les informations.
- ✓ Visualiser les données statistiques de manière claire grâce à des graphiques (diagramme circulaire, histogramme, courbe temporelle).
- ✓ Renforcer les compétences en programmation orientée objet (POO), gestion des exceptions personnalisées**,** et création d'interfaces graphiques avec Tkinter.
- ✓ Favoriser une organisation modulaire et robuste du code, facilitant la maintenance et l'extension du projet.

❖ Architecture du projet

- **main.py** : Fournit une interface en ligne de commande, et permet également l'affichage des statistiques via Matplotlib.
- **Interface.py** : Implémente une interface graphique en Tkinter permettant d'effectuer toutes les opérations de manière intuitive (ajout, modification, suppression, emprunt, retour, statistiques).
- **bibliotheque.py** : Contient les classes principales du projet : Livre, Membre et Bibliotheque
- **exceptions.py** : Déclare les exceptions personnalisées utilisées pour gérer les erreurs.
- **visualisations.py** : Regroupe les fonctions de visualisation de données à l'aide de la bibliothèque Matplotlib.
- **data/** : Dossier contenant les fichiers de persistance (livres.txt, membres.txt, historique.csv), utilisés pour stocker les données.
- **requirements.txt** : Liste les dépendances nécessaires à l'exécution du projet.
- **README.md** : Fichier de présentation du projet contenant les instructions d'installation et d'exécution.

❖ Diagramme de classe

Le système repose sur trois classes principales : Livre, Membre et Bibliotheque. Ces classes sont conçues pour modéliser les entités essentielles d'une bibliothèque et les interactions entre elles.



❖ Explications des algorithmes clés

1-Gestion des emprunts

L'algorithme de gestion des emprunts commence par vérifier l'existence du livre et du membre dans la bibliothèque. Il contrôle également que le membre n'a pas dépassé le quota maximal d'emprunts **(3 livres)** et que le livre est disponible **(Pas emprunté)**. Si ces conditions sont satisfaites, le livre est marqué comme **« emprunté »** et ajouté à la liste des livres empruntés du membre. L'opération est enregistrée dans un fichier d'historique pour garder une trace des actions.

2-Gestion des retours

Lors d'un retour, l'algorithme vérifie que le livre et le membre sont valides et que le livre a bien été emprunté par ce membre. Ensuite, il remet le statut du livre à **« disponible »** et le retire de la liste des emprunts du membre. Comme pour l'emprunt, l'action est notée dans l'historique.

3-Sauvegarde des données

Pour garantir la persistance des données, l'application sauvegarde régulièrement l'état actuel de la bibliothèque dans des fichiers texte. Chaque livre est enregistré avec ses attributs (ISBN, titre, auteur, année, genre, statut) dans un fichier **livres.txt**. De même, chaque membre est enregistré dans un fichier **membres.txt** avec son ID, son nom, et la liste des livres qu'il a empruntés, identifiés par leur ISBN. Cette méthode permet de conserver les informations même après la fermeture du programme.

4-Chargement des données

Au démarrage de l'application, les fichiers de données sont lus pour reconstruire l'état de la bibliothèque. Le programme crée des objets Livre et Membre en fonction des informations contenues dans **livres.txt** et **membres.txt**. Il restaure également les relations entre membres et livres empruntés en associant chaque membre aux livres qu'il a empruntés, grâce à la liste d'ISBN enregistrée. Cette étape assure la continuité des données d'une session à l'autre.

5-Visualisation statistique

Les données collectées permettent de générer des graphiques avec Matplotlib. Par exemple, un diagramme circulaire présente la répartition des livres par genre, un histogramme montre les auteurs les plus empruntés, et une courbe temporelle illustre l'activité des emprunts au cours des 30 derniers jours. Ces visualisations offrent une vue d'ensemble simple et efficace de l'utilisation de la bibliothèque.

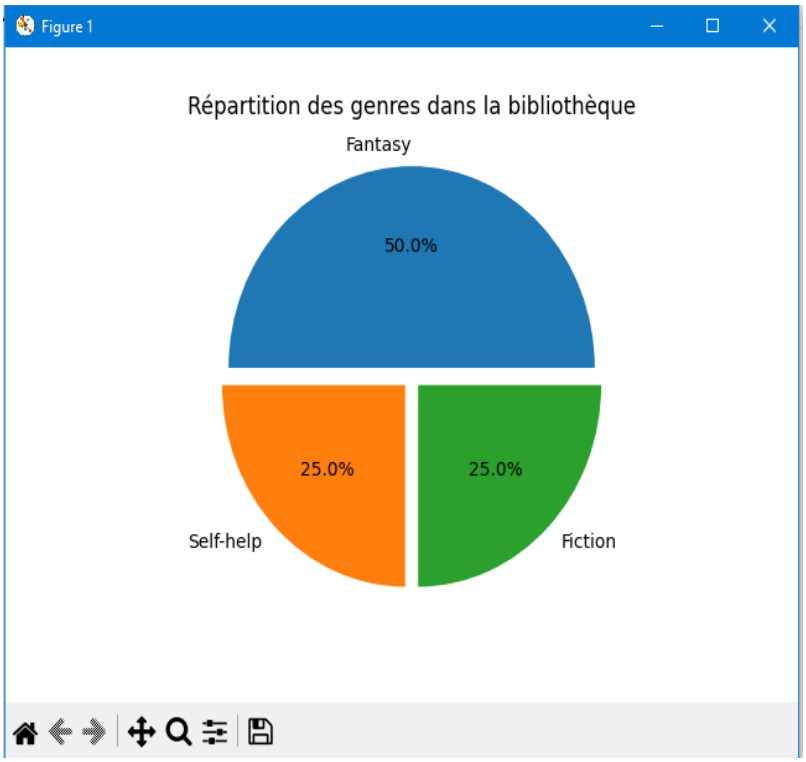
❖ Les visualisations

- **Diagramme circulaire : Répartition des genres dans la bibliothèque**

Le graphique présente la proportion de livres appartenant à différents genres dans ta bibliothèque.
Chaque portion représente un genre :

- **Fantasy** : 50%
- **Self-help** : 25%
- **Fiction** : 25%

Les couleurs permettent de distinguer visuellement chaque catégorie.

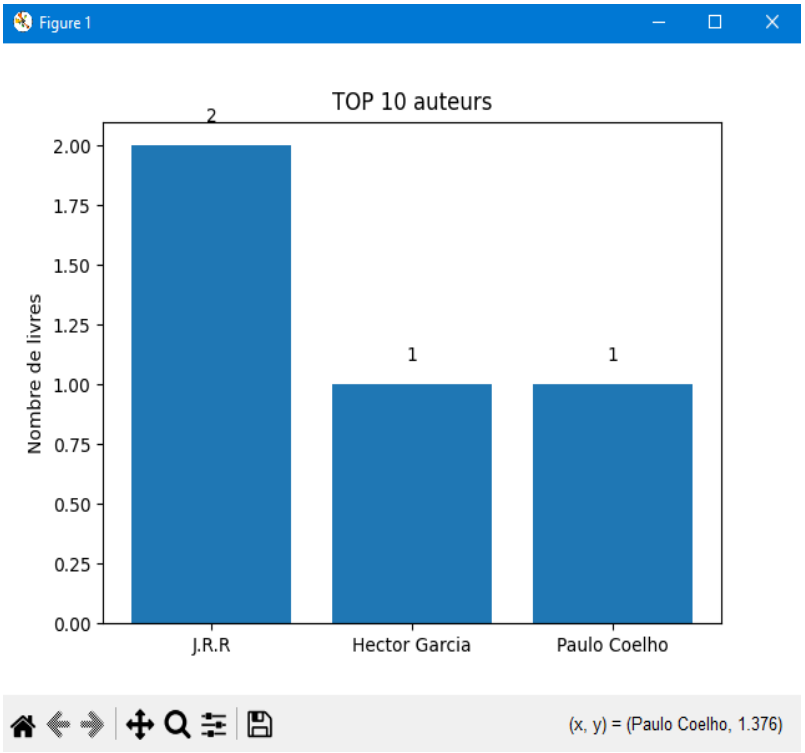


- **Histogramme : TOP 10 des auteurs les plus populaires**

Le graphique présente les auteurs les plus présents dans la bibliothèque, triés par nombre de livres.

Chaque **barre** représente un auteur.
L'**hauteur de la barre** correspond au **nombre de livres** présents dans la bibliothèque écrits par cet auteur.
Les données affichées ici :

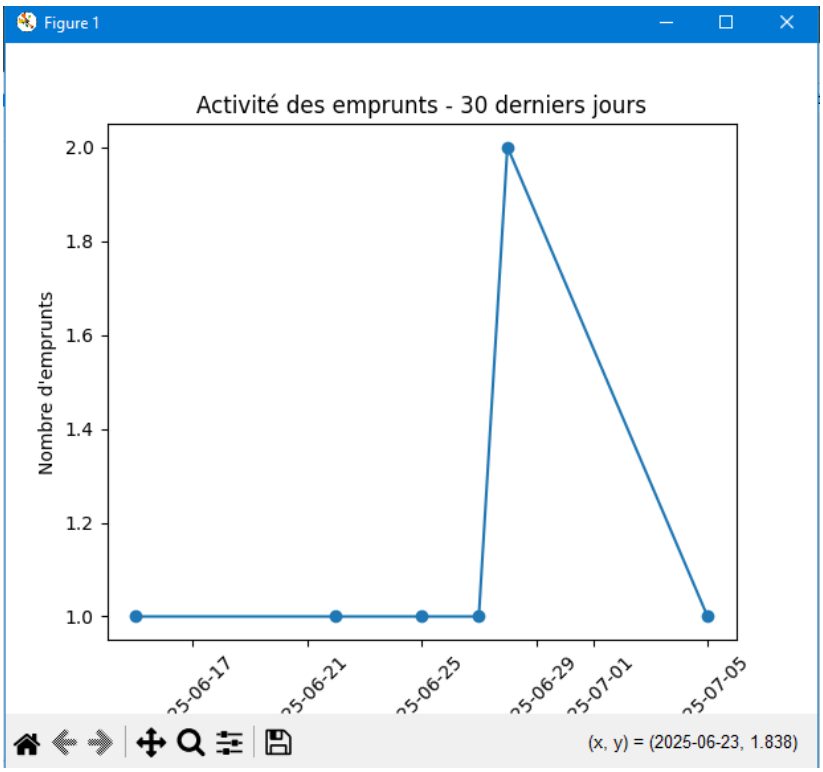
- **J.R.R** : 2 livres
- **Hector Garcia** : 1 livre
- **Paulo Coelho** : 1 livre



- **Courbe temporelle : Activité des emprunts (30 derniers jours)**

Le graphique présente le nombre de livres empruntés **chaque jour** au cours du **dernier mois**.

- Chaque **point** représente une journée où au moins un emprunt a été enregistré.
- L'activité est **faible et stable** sur la plupart des jours (1 emprunt/jour).
- Une **pointe d'activité** est visible vers le **28 juin 2025**, avec **2 emprunts** ce jour-là.
- Après ce pic, l'activité **redescend immédiatement** à 1 emprunt/jour.



❖ Difficultés rencontrées et solutions

- **Synchronisation des données**
  Modifications dans l'interface parfois non reflétées dans les fichiers, et inversement.
  Solution : Sauvegarde systématique après chaque modification et rechargement des données
- **gestion des imports relatifs dans un projet multi-dossiers**
  Configuration correcte les chemins d'accès (__file__, os.path.dirname) pour assurer l'import correct des modules.
  Solution : Utilisation d'imports relatifs et gestion des chemins dynamiques.
- **Maîtrise de Tkinter**
  Complexité de gérer onglets, tableaux dynamiques et événements.
  Solution : Apprentissage progressif et organisation claire du code.
- **Sauvegarde et chargement en format texte**
  Trouver un format fiable et compatible avec les règles métier.
  Solution : Standardisation avec fichiers texte séparés par ; et gestion des erreurs.
- **Visualisation avec Matplotlib**
  Ajustement de l'affichage et sauvegarde des graphiques.
  Solution : Tests répétés, ajustement des marges, et vérification des dossiers avant sauvegarde.
- **Manipulation des fichiers CSV pour l'historique**
  Lecture et écriture des données historiques.
  Solution : Utilisation du module csv pour gérer efficacement les fichiers.