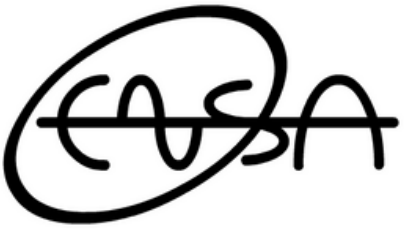


Rapport du Mini-Projet PYTHON:

Application Système de Gestion de Bibliothèque

---

nom: LAMRAOUI Ismail

classe: GI3

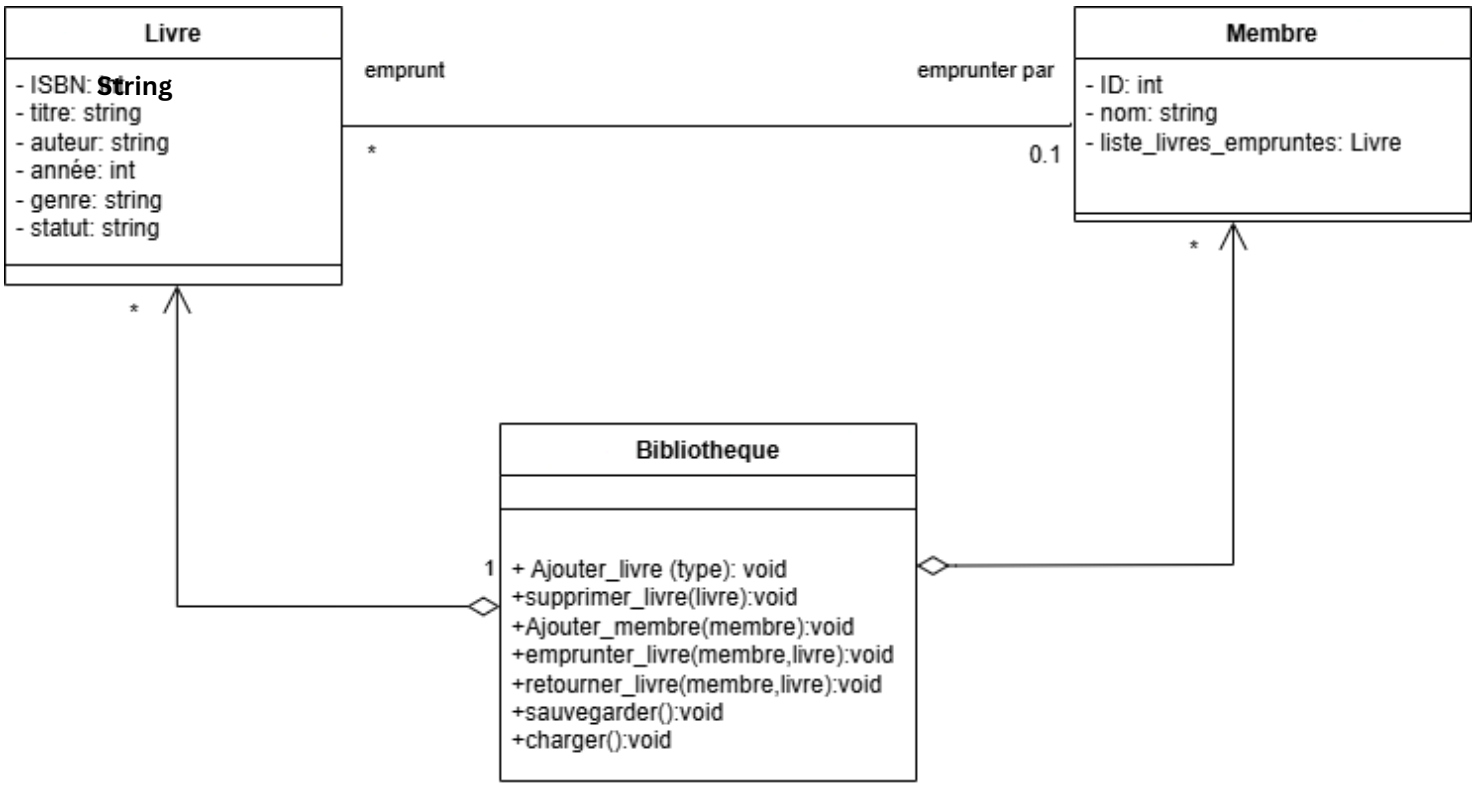
num: 40

- **Introduction:**

Ce projet a pour objectif de développer une application de gestion de bibliothèque en langage Python, en respectant les principes de la programmation orientée objet (POO). Il permet de gérer les livres, les membres et les opérations d'emprunt et de retour, tout en assurant la persistance des données via des fichiers JSON et CSV.

- L'application intègre une interface graphique simple réalisée avec Tkinter,
- ainsi qu'une visualisation statistique des données grâce à matplotlib
- Elle propose une expérience utilisateur fluide tout en conservant une structure claire et modulaire.

- **Structure du projet :**
  - UML:



- NB: le UML represente la vision avant d'avancer dans le projet(lcode), il existe quelque modif/ajout, mais tout ce qui essentiel est la.

Le projet repose sur plusieurs classes principales :

- **Livre** : représente un livre avec des attributs comme l'ISBN, le titre, l'auteur, le genre, l'année et le statut (disponible ou emprunté).
- **Membre** : représente un membre inscrit à la bibliothèque, avec son identifiant, son nom et la liste des livres empruntés.
- **Bibliotheque** : classe centrale qui gère les listes de livres et de membres, les emprunts, les retours, la sauvegarde et le chargement des données, ainsi que la génération des statistiques.

- **exceptions.py** : regroupe les exceptions personnalisées comme **LivreIndisponibleError**, **MembreInexistantError**, etc.

Le tout est organisé dans des fichiers :

- **bibliotheque.py**
- **Livre.py**
- **Membre.py**
- **exceptions.py**
- **main_tk.py**
- Dossier **data**/ : contient les fichiers **livre.json**, **membre.json** et **historique.csv**

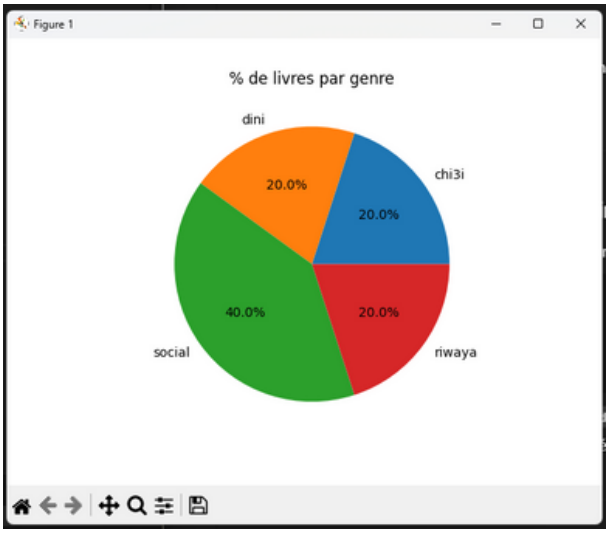
- ## Fonctionnalités réalisées:
  - ✅ Ajouter/Supprimer livres
  - ✅ Ajouter membres
  - ✅ Emprunter / retourner livres (avec exceptions)
  - ✅ Sauvegarde dans fichiers
  - ✅ Statistiques (genre, auteurs, activité)
  - ✅ **Lister les membres et les livres**
  - ✅ Interface Tkinter (mentionné même si minimal)

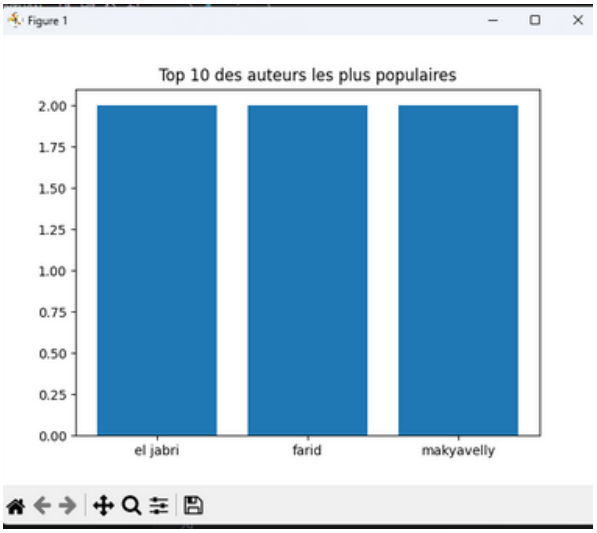
- **Statistiques :**

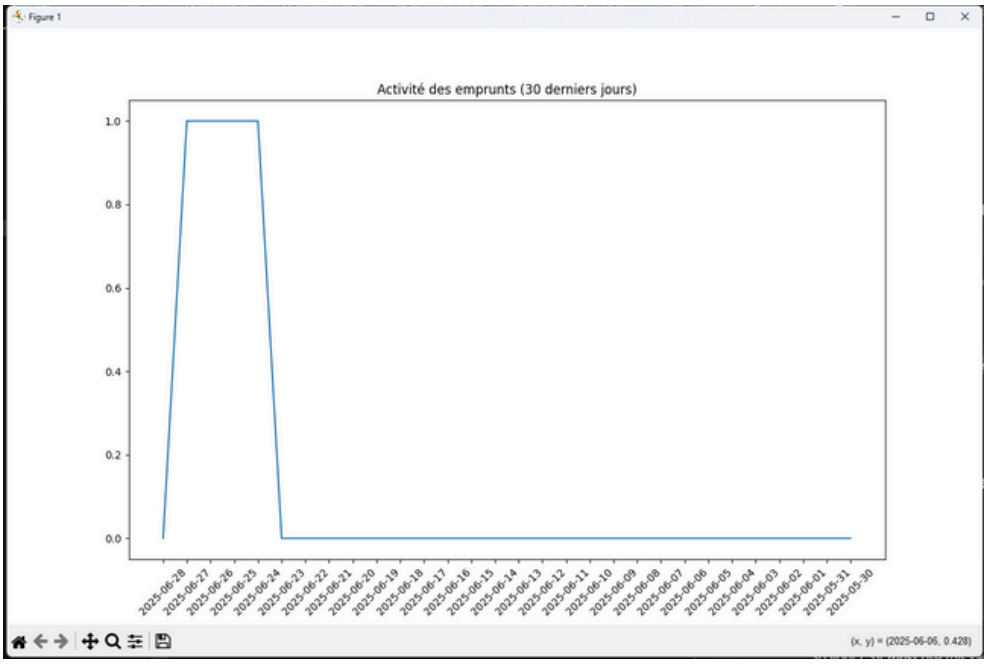
Le projet propose trois statistiques visuelles basées sur les données d'emprunt :

- **Diagramme circulaire** : pourcentage des livres par genre. Permet de visualiser la répartition des genres dans la bibliothèque.
- **Histogramme** : top 10 des auteurs les plus empruntés. Basé sur l'analyse du fichier historique.csv.
- **Courbe d'activité** : nombre d'emprunts par jour sur les 30 derniers jours. Affiche les tendances de fréquentation.

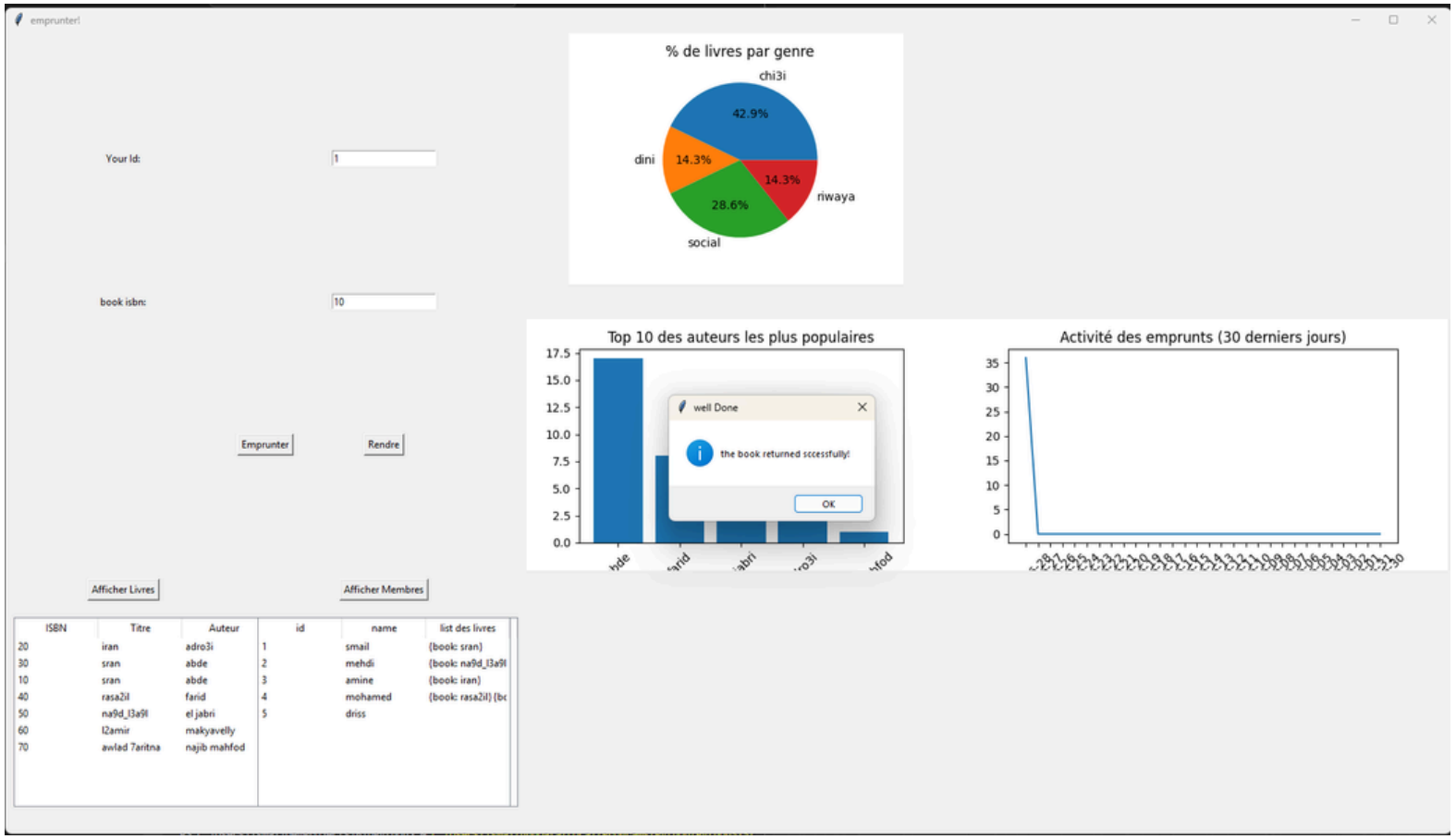
NB: Chaque graphique est généré par une méthode de la classe Bibliotheque, en utilisant matplotlib.



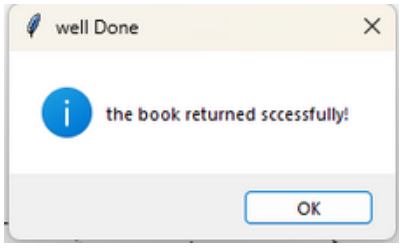


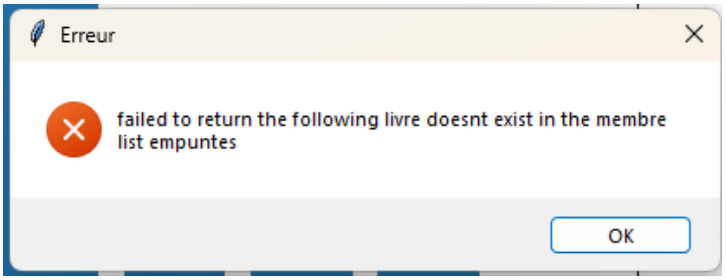


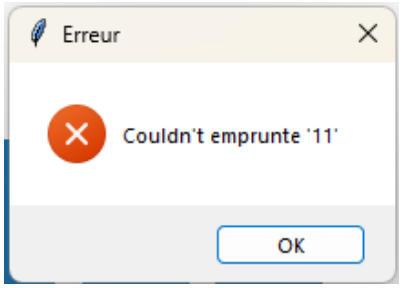
- **Interface Graphique *Tkinter*:**

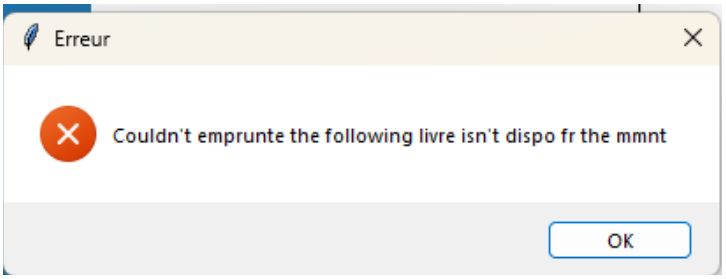


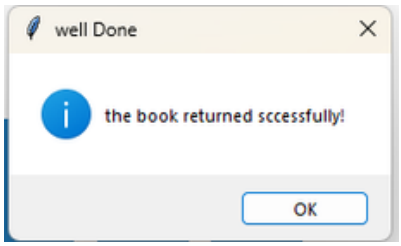
- Alerts:



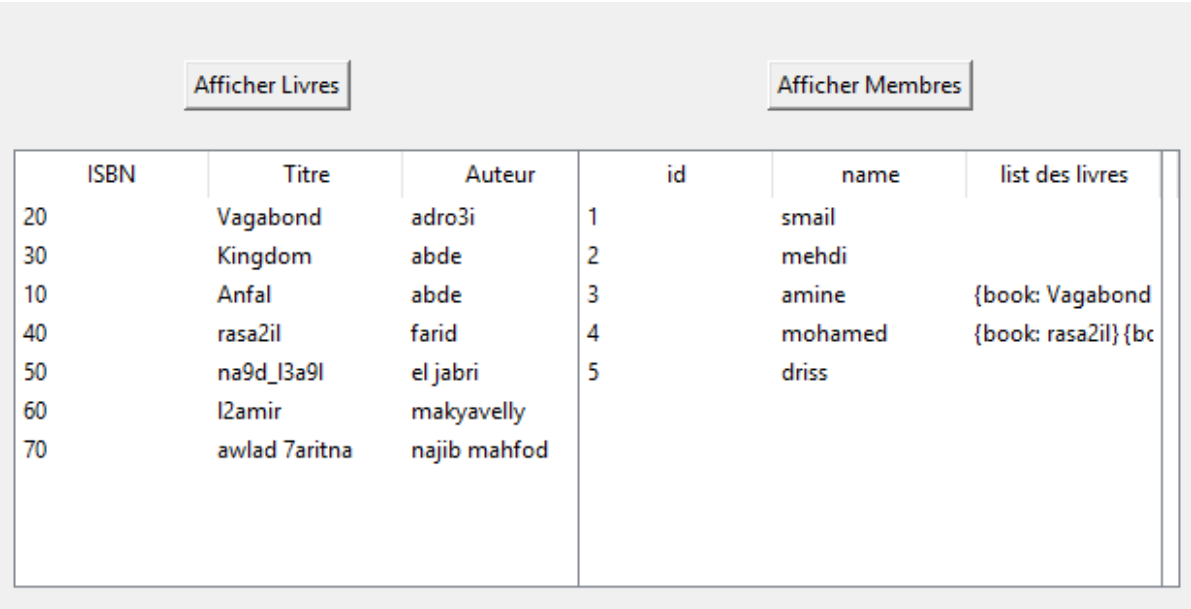








✅ Lister les membres et les livres



dynamique se refresh apres chaque click sur les butons

ISMAIL LAMRAOUI

- # Algorithmes clés:

Dans la methode **chager_data()**, pour charger les membres exactement la list des livres empruntes, alors que .json a juste le isbn du livre; on le recupere alors et on l'utilise comme **cle** pour chercher dans le **dict** des livres (qu'on a juste fini de charger dans la mm fnct.

ET le deuxieme pour **sauvegarder_data()** methode, meme partie save membre; or notre list de emprunt contient les livre**(object)**, on cree une temporaire liste pour stocker juste leurs **Isbn** avant de stoker in data.

ISMAIL LAMRAOUI

```
with open(r"C:\Users\ismai\DevProjects\BibPython\data\membre.json","r") as f:
    membrdata= json.load(f)
    for i in membrdata:
        temp_membre= Membre(int(i["id"]),i["nom"])
        for j in i["livres_empruntes"]: # j represent alors le isbn
            # var=i["livres_empruntes"][j] ///false cz i[livre_emprnt] machi dict its a list
            livre1= self.__listLivres[str(j)]
            temp_membre._livres_empruntes.append(livre1)
        self.__listMembres[int(i["id"])]=temp_membre

    # debug helper nade
# for id, m in self.__listMembres.items():
#     print(f"[{id}] empruntés:", [liv._isbn for liv in m._livres_empruntes])
```

```
with open(r"C:\Users\ismai\DevProjects\BibPython\data\membre.json","w") as f:
    membre_data=[]
    for id in self.__listMembres:
        temp_membre= self.__listMembres[id]
        emprunts_isbn=[] # bach ncollectiw only isbns of livre
        for lvr_empr in temp_membre._livres_empruntes:
            emprunts_isbn.append(str(lvr_empr._isbn))
        data={
              "id":int(temp_membre._id),
              "nom":temp_membre._nom,
              "livres_empruntes": emprunts_isbn
              }
        membre_data.append(data)
    # debug helper nade
    # for data in membre_data:
    #     print(f"ID: {data['id']}, emprunts: {data['livres_empruntes']}")

    json.dump(membre_data,f,indent=2)
```

Pour la methode stats du dernier 30 jours, or j'ai pas d'experience en utilisation du datetime, j'ai pris beaucoup de nouveaux choses, en cette etape j'ai utilise' l'aide de chatgpt, j'ai cree/ une liste des 30 derniers jours utilisant .today() et timedelta() et strftime (strptime() aussi dans autre partie) cree un dict des 30 jour initialiser a 0, recuperer les donnes separee par ";", et checker l existance de ce jour pour incrementer ou l initialiser a un dans notre dictionnaire.

ISMAIL LAMRAOUI

```python
def stats_30_days (self):
    today= datetime.today()
    last_30_days=[]
    for i in range(0,30):
        temp_day= today - timedelta(days=i)
        last_30_days.append(temp_day.strftime("%Y-%m-%d"))
    with open(r"C:\Users\ismai\DevProjects\BibPython\data\historique.csv","r") as f :
        line=f.readline()
        last_30_day={d: 0 for d in last_30_days} # bach ta ila kan nhar khawi ikon fih 0
        while line:
            elem= line.split(";")

            if len(elem) != 4 or elem[3].strip() != "emprunt" or elem[0] not in last_30_days :
                line = f.readline()# to read thenext line before going to next iteration
                continue
            else:
                if elem[0] not in last_30_day:
                    last_30_day[elem[0]]=1
                else:
                    last_30_day[elem[0]] +=1

            line=f.readline()
```

- # Conclusion :

Ce projet m'a permis de consolider mes compétences en **POO**, en **gestion de fichiers**, en manipulation de données, et en création d'interfaces utilisateur. L'intégration des statistiques avec **matplotlib** et la conception de l'interface avec **Tkinter** ont ajouté une dimension pratique et visuelle au projet.
Parmi les difficultés rencontrées : le temps etait trop court :), la synchronisation entre objets et fichiers, et **l'alignement des données temporelles pour les statistiques**.
Si plus de temps était disponible, j'aurais aimé :

- Améliorer l'interface (affichage tabulaire, recherches dynamiques)
- Implémenter la suppression des membres/livres

Ce projet reste une expérience riche, concrète et valorisante pour ma progression en informatique.