

Université Mohammed Premier
École Nationale des Sciences Appliquées
Filière Génie Informatique

Programmation avancée en Python

Mini projet d'application

Système de Gestion de Bibliothèque

A. Objectif du projet

Vous développerez une application complète de gestion de bibliothèque en Python qui intègre :

- Un backend POO pour la logique métier
- La gestion des exceptions pour les erreurs métier
- La persistance des données dans des fichiers
- La génération de rapports statistiques avec Matplotlib
- La persistance des données (fichiers JSON/CSV)

B. Fonctionnalités Requises

1. *Module de Gestion des Livres (POO)*

- Classe Livre avec attributs :
 - ISBN, titre, auteur, année, genre, statut (disponible/emprunté)
- Classe Membre avec :
 - ID, nom, liste des livres empruntés
- Classe Bibliotheque avec méthodes :
 - Ajouter/supprimer des livres
 - Enregistrer des membres
 - Gérer les emprunts/retours
 - Sauvegarder/charger les données

2. Interface en Ligne de Commande

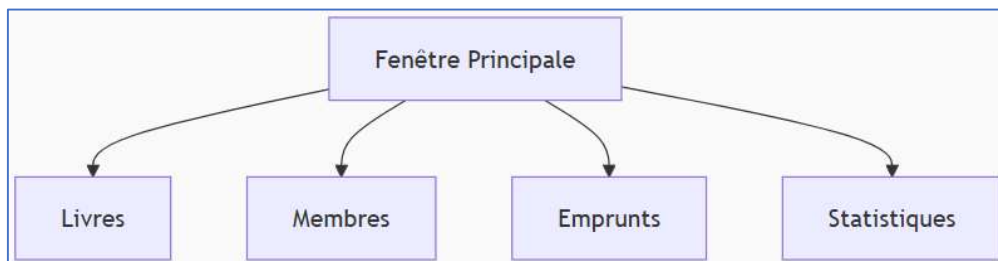
- Menu interactif avec les options :

```
=== GESTION BIBLIOTHEQUE ===  
1. Ajouter un livre  
2. Inscrire un membre  
3. Emprunter un livre  
4. Rendre un livre  
5. Lister tous les livres  
6. Afficher les statistiques  
7. Sauvegarder et quitter
```

- Validation des saisies utilisateur

3. Interface Graphique (Tkinter) (Optionnel)

- Fenêtre principale avec onglets :



- Widgets requis :
 - Tableaux
 - Formulaires de saisie
 - Boutons d'action avec gestion d'erreurs
 - Messages d'alerte personnalisés

4. Gestion des Erreurs (Exceptions)

- Créer des exceptions personnalisées :
 - LivreIndisponibleError
 - QuotaEmpruntDepasseError
 - MembreInexistantError
 - LivreInexistantError
- Intercepter ces exceptions dans le flux principal

5. Tableau de Bord Statistique

- Visualisations : Générer des graphiques avec Matplotlib
 - Diagramme circulaire : % de livres par genre

- Histogramme : Top 10 des auteurs les plus populaires
- Courbe temporelle : Activité des emprunts (30 derniers jours)
-

6. *Gestion des Données*

- Sauvegarde dans des fichiers texte/JSON :
 - livres.txt (format: ISBN;titre;auteur;année;genre;statut)
 - membres.txt (format: ID;nom;livres_empruntés)
 - historique.csv (format: date;ISBN;ID_membre;action)
- Chargement des données au démarrage

C. Organisation

- Projet individuel
- Le lien GitHub doit être déposé sur Classroom
- Livrables :
 - **Référentiel GitHub** (Voici un exemple de structure du repo Github)

```
Gestion_Bibliotheque_[VotreNom]/
├── data/
│   ├── livres.txt
│   ├── membres.txt
│   └── historique.csv
├── src/
│   ├── main.py                # Programme principal
│   ├── bibliotheque.py        # Classes POO
│   ├── exceptions.py          # Exceptions personnalisées
│   └── visualisations.py      # Génération des graphiques
├── docs/
│   └── rapport.pdf
├── assets/
│   ├── stats_genres.png
│   ├── stats_auteurs.png
│   └── presentation.mp4
├── requirements.txt
└── README.md
```

- Contenu des Fichiers
 - README.md :
 - Votre nom
 - Guide d'installation
 - Exemples d'utilisation
 - rapport.pdf (4-6 pages) :
 - Diagramme de classes UML
 - Explications des algorithmes clés
 - Captures d'écran des visualisations
 - Difficultés rencontrées et solutions
 - Vidéo de Présentation (4 min max) :
 - Démonstration des fonctionnalités
 - Explication technique des parties complexes
 - Présentation des visualisations générées
- Liens utiles :
 - <https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNeOtoC6GglhF3ncJ5rLwQrLGnV>
 - <https://www.youtube.com/watch?v=2JjQIh-sgHU>
- Date limite : 28 Juin 2025 Le lien GitHub doit être déposé sur Classroom
- NB : les projets avec des codes copier-coller ils ne seront absolument pas pris en compte.

D. Critères d'évaluation

- POO
- Gestion des erreurs
- Persistance des données
- Qualité des visualisations
- Documentation et rapport
- Vidéo de Présentation