Fichiers à joindre : Auteurs.json / Livres.json / Membres.json / obj_couleur.py / obj_bibilotheque.py / obj_ihm.py / gere_biblio.py

Les notions abordées

Au cours de ce TP vous allez aborder les notions de base de la POO comme :

- Le codage de classes et de son constructeur.
- La création d'instances.
- La manipulation d'attributs d'instances et de classe.
- La création de méthodes.

*Note : Dans ce TP, tous les fichiers (ou modules) Python qui contiennent une classe, ont un nom qui commence par « **obj_** ». Ce choix a été fait pour des raisons purement pédagogiques et ne constitue pas une règle d'usage en Python…*

Présentation globale

On souhaite créer un outil de gestion d'une bibliothèque qui doit gérer :

- Une liste de livres.
- Une liste d'auteurs.
- Une liste de membres.
- Les emprunts et retours de livres que font les membres.

Une partie du code est fournie sur Moodle.

1. Téléchargez dans votre répertoire de travail le fichier : « `obj_couleur.py` ».

Partie 1 : obj_auteur.py

Pour la gestion de la bibliothèque, il est nécessaire d'avoir un module « **obj_auteur** » qui contient une classe « **Auteur** ». Celle-ci doit avoir UN attribut de classe et CINQ attributs d'instances :

- « **nombre_total_auteurs** » (*classe*) : qui est initialisé à 0 et qui est incrémenté par le constructeur à chaque création d'un nouvelle instance de l'objet « **Auteur** ».
- « **id** » : Numéro de l'auteur. Cet attribut prend la valeur de « **nombre_total_auteurs** » (après incrémentation) lors de l'instanciation de l'objet « **Auteur** ». Comme sa valeur est calculée par le constructeur, il ne fait donc pas partie de ses arguments…
- « **nom** » : Nom de l'auteur (en majuscule).
- « **prenom** » : Prénom de l'auteur (en « **capitalize()** »).
- « **pays** » : Pays d'origine. Valeur par défaut : « **None** ».
- « **date_naissance** » : Date de naissance de l'auteur. Valeur par défaut : « **None** ».

Codez le script « `obj_auteur.py` » en suivant les étapes suivantes :

2. Importez la classe « **Couleur** » du module « **obj_couleur** ». Cette classe définit des constantes de couleurs qui vont être utilisées pour tous les affichages.
3. Créez la classe « **Auteur** » qui hérite les éléments de la classe « **Couleur** ».

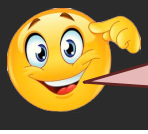
***Indications** : Si l'on souhaite créer la classe « **Objet_B** » avec un héritage des éléments de la classe « **Objet_A** », on utilise la syntaxe :* `class Objet_B(Objet_a):`

4. Ajoutez l'attribut de classe : « **nombre_total_auteurs** ».
5. Créez le constructeur qui dispose de quatre arguments dont deux ont des valeurs par défaut.

***Indications** : Si l'argument « **pays** » est égal à « **None** », on souhaite que l'attribut « **self.pays** » soit égale à « **inconnu** ». De même pour « **date_naissance** » (avec un « **e** » pour respecter le genre).*

6. Testez votre code en y ajoutant **en dehors de la classe**, le code suivant :

```
print("Création de 3 instances de Auteur et affichage...")
follett = Auteur("FOLLETT", "Ken", "Pays de Galles", "05/06/1949")
verne = Auteur("VERNE","Jules","France", "08/02/1828")
bridou = Auteur("BRIDOU", "Justin", None, None)
print(follett)
print(verne)
print(bridou)
print(bridou.pays)
print(bridou.date_naissance)
```



*Remarque : Le changement de « **None** » en « **Inconnu(e)** » dans le constructeur peut se faire avec un test sur une seule ligne:*
*Ex : **temp = "Chaud" if soleil else "Froid***

Booléen.

***Résultat** : Les trois « **print** » ne sont pas très exploitables…*

QCM

7. Créez la méthode spéciale « **__str__(self)** ». Cette méthode doit retourner une « *f-string* » qui va être utiliser par le « **print(***nom_instance***)** » pour pouvoir afficher un contenu explicite de chaque instance. On souhaite avoir le rendu suivant :

« **id** » Tab (« **\t** ») NO_COLOR MAGENTA

```
Création de 3 instances de Auteur et affichage...
1.  : Ken FOLLETT (né(e) le 05/06/1949 en Pays de Galles)
2.  : Jules VERNE (né(e) le 08/02/1828 en France)
3.  : Justin BRIDOU (né(e) le Inconnue en Inconnu)
Inconnu
Inconnue
```

***Indications** : Les couleurs possibles sont définies comme **attributs de classe** dans la classe « **Couleur** » dans « **obj_couleur** ». Pour appliquer une couleur, il faut utiliser au sein de la « f-string », la syntaxe :*

*« **{***nom_classe.NOM_COULEUR***}** ».*

***Exemple** : `f"Texte en {Auteur.BLEU}bleu{Auteur.NO_COLOR}"`…*

8. Testez à nouveau votre code et vérifiez bien qu'il produit l'affichage désiré…

Partie 2 : obj_livre.py

Pour compléter la gestion de la bibliothèque, il est nécessaire d'avoir un second module « **obj_livre** » qui contient une classe « **Livre** ». Celle-ci doit avoir UN attribut de classe et SIX attributs d'instances :

- « **nombre_total_livres** » (*classe*) : qui fonctionne comme la classe précédente. Il est initialisé à 0 et est incrémenté par le constructeur pour toute nouvelle création d'une instance de « **Livre** ».
- « **id** » : Numéro du livre. Cet attribut prend la valeur de « **nombre_total_livres** » (après incrémentation) lors de l'instanciation de l'objet « **Livre** ». Comme sa valeur est calculée par le constructeur, il ne fait donc pas partie de ses arguments…
- « **titre** » : Titre du livre.
- « **auteur** » : Instance d'un objet « **Auteur** ». Cela signifie que l'on ne peut pas créer de livre dont l'auteur n'est pas référencé par une instance de « **Auteur** »…
- « **isbn** » : Numéro ISBN (code barre en 4ème de couverture). Valeur par défaut : « **None** ».
- « **annee_publication** » : Année de la date de la publication du livre. Valeur par défaut : « **None** ».
- « **disponible** » : Booléen indiquant si le livre est disponible au prêt ou s'il a été emprunté par un membre. Valeur par défaut : « **True** ».

Codez le script « `obj_livre.py` » en suivant les étapes suivantes :

9. Importez la classe « **`Couleur`** » du module « **`obj_couleur`** » ainsi que la classe « **`Auteur`** » du module « **`obj_auteur`** ».
10. Créez la classe « **`Livre`** » qui hérite les éléments de la classe « **`Couleur`** ».
11. Ajoutez l'attribut de classe : « **`nombre_total_livres`** ».
12. Créez le constructeur qui dispose de quatre arguments dont deux ont des valeurs par défaut.

Indications *: Si l'argument « **`isbn`** » est égal à « **`None`** », on souhaite que l'attribut « **`self.isbn`** » soit égale à « **`N/A`** ». De même, si l'argument « **`annee_publication`** » est égale à « **`None`** », on souhaite que l'attribut « **`self.annee_publication`** » soit égale à « **`inconnue`** ».*

QCM

13. Testez votre code en y ajoutant **en dehors de la classe**, le code suivant :

```
follett = Auteur("FOLLETT", "Ken", "Pays de Galles", "05/06/1949")
verne = Auteur("VERNE","Jules","France", "08/02/1828")

print("Création de 3 instances de Livre et affichage...")
livre_1 = Livre("Les Piliers de la Terre", follett, "9782130428114", "1989")
livre_2 = Livre("Une Colonne de Feu", follett, "9782221157695", "2017")
livre_2.disponible = False
livre_3 = Livre("Vingt Mille Lieues sous les mers", verne, "9782070364234", "1870")
print(livre_1)
print(livre_2)
print(livre_3)
```

Pour créer des instances de « **`Livre`** », il faut disposer d'instances de « **`Auteur`** »…

Résultat 1 *: Les « **`print`** » ne sont pas très exploitables car il manque la méthode « **`__str__(self)`** ».*

Note : Pour que le code qui se trouve en dehors d'une classe ne soit pas exécuté lors de l'importation du module correspondant, il faut le placer dans le test :
`if __name__ == "__main__":`

14. Ajoutez pour les modules « **`obj_auteur`** » et « **`obj_livre`** » et au début du code qui est externe à la classe, le test « **`if __name__ == "__main__":`** »

QCM

15. Créez la méthode « **`__str__(self)`** » pour que le « **`print(`***`nom_instance`***`)`** » affiche un contenu explicite de chaque instance conforme à ceci :

```
Création de 3 instances de Livre et affichage...
1.  : 'Les Piliers de la Terre' de Ken FOLLETT (ISBN: 9782130428114, publié en 1989) - Dispo
2.  : 'Une Colonne de Feu' de Ken FOLLETT (ISBN: 9782221157695, publié en 2017) - NON Dispo
3.  : 'Vingt Mille Lieues sous les mers' de Jules VERNE (ISBN: 9782070364234, publié en 1870) - Dispo
```

QCM

16. Testez à nouveau votre code et vérifiez bien qu'il produit l'affichage désiré…

Partie 3 : obj_membre.py

Pour finir la gestion de la bibliothèque, il est nécessaire d'avoir un troisième module « **obj_membre** » qui contient une classe « **Membre** ». Celle-ci doit avoir UN attribut de classe et CINQ attributs d'instances :

- « **nombre_total_membres** » (*classe*) : qui fonctionne comme les deux classes précédentes.
- « **id** » : Numéro du membre.
- « **nom** » : Nom du membre.
- « **prenom** » : Prénom du membre.
- « **date_naissance** » : Sa date de naissance au format : jj/mm/aaaa.
- « **livres_empruntes** » : Liste d'instances d'objets « Livre ». Le constructeur doit lui affecter une liste vide lors de l'instanciation…

Codez le script « `obj_membre.py` » en suivant les étapes suivantes :

17. Importez la classe « **Couleur** » du module « **obj_couleur** », la classe « **Auteur** » du module « **obj_auteur** » ainsi que la classe « **Livre** » du module « **obj_livre** ».
18. Créez la classe « **Membre** » qui hérite les éléments de la classe « **Couleur** ».
19. Ajoutez l'attribut de classe : « **nombre_total_membres** ».
20. Créez le constructeur qui dispose de trois arguments.
21. Créez la méthode « **__str__(self)** » pour que le « **print(***nom_instance***)** » affiche un contenu explicite de chaque instance conforme à ceci :

```
Création de 2 instances de Membre et affichage...
1.    : Albert EINSTEIN (né(e) le 14/03/1879)
2.    : Marie CURIE (né(e) le 07/11/1867)
```

22. Testez votre code en y ajoutant **en dehors de la classe**, le code suivant :

```
if __name__ == "__main__":
    print("Création de 2 instances de Membre et affichage...")
    albert = Membre("EINSTEIN", "Albert", "14/03/1879")
    marie = Membre("CURIE", "Marie", "07/11/1867")
    print(albert)
    print(marie)
```

Il faut à présent enrichir cette classe de méthodes qui permettent de gérer les emprunts et restitutions de livres que peut fait chaque membre.

Codez la méthode « **lister_emprunts()** » en suivant les consignes suivantes :

23. Ajoutez pour commencer le code suivant à la fin du « *main* ». Ce code va vous permettre de tester la méthode que vous êtes sur le point d'écrire…

```
    follett = Auteur("FOLLETT", "Ken", "Pays de Galles", "05/06/1949")
    livre_1 = Livre("Les Piliers de la Terre", follett, "9782130428114", "1989")
    livre_2 = Livre("Une Colonne de Feu", follett, "9782221157695", "2017")

    albert.livres_empruntes.append(livre_1)
    albert.livres_empruntes.append(livre_2)

    print("\n*** 1er affichage des livres empruntés...")
    albert.lister_emprunts()
    marie.lister_emprunts()
```

Pour créer des emprunts, il faut des livres et chacun d'eux a besoin de l'instance de son auteur…

Pour le test, on ajoute « *à la main* » des emprunts…

24. Ajoutez dans la méthode, le code qui affiche le texte suivant **si la liste des emprunts est vide** :

```
    /// Marie CURIE n'a aucun livre emprunté.
```

Tab (« \t ») — Prénom et nom du membre.

25. Ajoutez le code qui affiche les lignes de texte suivantes **si la liste des emprunts n'est pas vide** :

Tab (« \t ») — Prénom et nom du membre.

```
-> Albert EINSTEIN a emprunté les livres suivants :
    - 1. Les Piliers de la Terre de Ken FOLLETT
    - 2. Une Colonne de Feu de Ken FOLLETT
```

Une ligne affichée pour chaque élément de la liste des livres empruntés…

Tab (« \t ») — Titre du livre. — Prénom et nom de l'auteur.

26. Vérifiez que la méthode « **`lister_emprunts()`** » génère bien les affichages précédents…

Codez à présent la méthode « **`emprunter_livre(…)`** » qui admet pour argument une instance de l'objet « **`Livre`** ». Votre code doit respecter les contraintes suivantes :

27. Testez si l'instance de l'objet « **`Livre`** » est disponible (attribut : « **`disponible`** »).
28. Affichez le texte suivant **si le livre n'est pas disponible** :

```
    ----> Le livre 'Une Colonne de Feu' est déjà emprunté.
```

Tab (« \t ») — Titre du livre.

29. Réalisez les étapes suivantes **si le livre est disponible** :
    - Rendre le livre indisponible.
    - Ajouter le livre à la liste des livres empruntés du membre.
    - Afficher le texte suivant :

```
    ----> Albert EINSTEIN a emprunté 'Les Piliers de la Terre'.
```

Tab (« \t ») — Prénom et nom du membre. — Titre du livre.

30. Tester cette seconde méthode avec le code du « *main* » suivant (complément et modification) :

```
albert.livres_empruntes.append(livre_1)
albert.livres_empruntes.append(livre_2)

print("\n*** 1er affichage des livres empruntés...")
albert.lister_emprunts()
marie.lister_emprunts()

print("\nEmprunts de livres...")
albert.emprunter_livre(livre_1)
albert.emprunter_livre(livre_2)
marie.emprunter_livre(livre_2)

print("\n*** 2ème affichage des livres empruntés...")
albert.lister_emprunts()
marie.lister_emprunts()
```

A supprimer…

Existe déjà…

A ajouter…

Codez pour finir la méthode « **restituer_livre(…)** » qui admet pour argument une instance de l'objet « **Livre** ». Votre code doit respecter les contraintes suivantes :

31. Rendre le livre disponible.
32. Supprimer le livre à la liste des livres empruntés du membre.
33. Affichez le texte suivant :

```
--------> Albert EINSTEIN a restitué 'Les Piliers de la Terre'.
```

Tab (« \t ») | Prénom et nom du membre. | Titre du livre.

34. Tester cette troisième méthode avec le code du « *main* » suivant :

```
print("\n*** 2ème affichage des livres empruntés...")
albert.lister_emprunts()
marie.lister_emprunts()

print("\nRestitution d'un livre...")
albert.restituer_livre(livre_1)

print("\n*** 3ème affichage des livres empruntés...")
albert.lister_emprunts()
marie.lister_emprunts()
```

Existe déjà…

A ajouter…

Partie 4 : test global

A votre code écrit dans ce qui précède, on va ajouter DEUX modules partiellement codés, un programme principal ainsi de trois fichiers JSON contenant des données. L'idée est d'ajouter le code qui est nécessaire pour faire un test en situation quasiment réelle.

35. Téléchargez dans votre répertoire de travail les SIX fichiers suivants :
    - « `obj_ihm.py` » : Module qui permet un interface texte (non graphique).
    - « `obj_bibliotheque.py` » : Module qui permet, à partir de fichiers JSON, de créer des listes d'instances pour les trois objets que vous avez créé précédemment .
    - « `gere_biblio.py` » : Script principal (« main ») de l'outil créé dans ce TP.
    - « `Auteurs.json` » : Données qui permettent de créer 11 instances de l'objet « Auteur ».
    - « `Livres.json` » : Données qui offrent la possibilité de créer 23 instances de l'objet « Livre » compatibles avec la liste des auteurs…
    - « `Membres.json` » : Données permettent de créer 7 instances de l'objet « Membre ».

En regardant le code de la classe « **Bibliotheque** » du module « **obj_bibliotheque** » et en particulier son constructeur, on peut voir qu'il est simpliste. Il contient six attributs : trois contiennent les noms des trois fichiers JSON et les trois autres sont juste des listes vides destinées à recevoir des instances de « **Auteur** », « **Livre** » et « **Membre** ».
Cette classe est également dotées de trois méthodes « **charger_auteurs(…)** », « **charger_livres(…)** » et « **charger_membres(…)** » qui permettent de charger dans ces trois listes les données des fichiers JSON…

***Note** : Un contrôle de l'existence de l'auteur dans la liste des auteurs est fait avant d'ajouter un livre à la liste des livres… Il en résulte qu'il faut toujours peupler la liste des auteurs AVANT celle des livres !!!*

Dans le module « **obj_bibliotheque** » se trouve également un « *main* » qui permet de le tester.

36. Effectuez ce test en exécutant « `obj_bibliotheque.py` ».

> ***Résultat*** *: Pour une première instance de l'objet « **Bibliotheque** », le peuplement des trois listes d'auteurs, de livres et de membres se réalise normalement à partir des trois fichiers JSON. Le résultat est affiché…*
>
> *Pour une seconde instance de l'objet « **Bibliotheque** », seul un auteur est ajouté « à la main » avant de lancer le peuplement de la liste des livres. Des problèmes apparaissent alors car de nombreux livres n'ont pas leur auteur dans la liste des auteurs !!!*

Le module « **obj_ihm** » contient une classe « **Ihm** » qui offre une « *Interface Homme-Machine* » (d'où IHM) permettant à l'utilisateur d'accéder aux différentes données de l'instance de l'objet « **Bibliotheque** ». Ainsi, cette IHM propose différents menus que l'on choisit à l'aide du clavier comme on le fait à la souris pour une IHM graphique… La classe « **Ihm** » deux attributs :

- « **biblio** » : Instance de l'objet « **Bibliotheque** ».
- « **choix** » : Variable qui donne le numéro du menu choisi par l'utilisateur.

Cette même classe dispose de 12 méthodes publiques : une pour chaque menu de l'IHM…
Découvrons à présent cette IHM :

37. Lancez cette IHM en exécutant « `gere_biblio.py` » et testez-là…

> ***Note*** *: Seuls les menus 1 à 6 et 12 sont réellement opérationnels. Les menus 7 à 11 sont à coder dans l'activité complémentaire…*
>
> ***Note importante*** *: Ne perdez pas trop de temps sur cela, il reste encore des choses à faire dans ce TP…*

Partie 5 : création d'un package…

La création d'un **package** (à usage local ou privé) est une opération particulièrement simple. Pour la réaliser, il faut suivre les étapes suivantes :

38. Créez dans le répertoire courant (où se trouve les « `.py` »), un répertoire « `mediatheque` » qui porte le nom que va avoir le **package**,.
39. Déplacez (***pas copiez***) les SIX fichiers « `obj_xxxxx.py` » dans ce répertoire « `mediatheque` ».
40. Créez dans ce répertoire « `mediatheque` » un nouveau script qu'il faut nécessairement nommer « `__init__.py` ».
41. Placez-y le code suivant :

```
# __init__.py pour le package 'mediatheque'

# Importation des classes principales du package
from .obj_auteur import Auteur
from .obj_livre import Livre
from .obj_membre import Membre
from .obj_bibliotheque import Bibliotheque
from .obj_ihm import Ihm
from .obj_couleur import Couleur

# Liste des classes et objets exposés par le package
__all__ = [
    'Bibliotheque',
    'Ihm',
]
```

Importation des tous les objets présents dans le package…

Liste des modules qui sont importés par un « **from mediatheque import *** »…

Il reste à faire quelques modifications mineures liés aux importations :

- Dans le code principale « `gere_biblio.py` »…

42. Apportez les modifications suivantes :

```
from mediatheque import Bibliotheque, Ihm
#from obj_bibliotheque import Bibliotheque
#from obj_ihm import Ihm
```

A ajouter…

A mettre en commentaires…

*Note : On importe bien les objets « **Bibliotheque** » et « **Ihm** » directement depuis le package et non plus depuis les deux modules « **obj_bibliotheque** » et « **obj_ihm** »…*

- Dans les SIX modules qui se trouvent à présent dans le répertoire « `mediatheque` », il faut faire des imports « ***relatifs*** ». Cela se fait en plaçant un « **.** » devant le nom du module…

Exemple pour « `obj_livre.py` » :

```
from .obj_couleur import Couleur
from .obj_auteur import Auteur
```

Ici le « point »…

43. Apportez ces modifications aux SIX modules de répertoire « `mediatheque` »…

***Note** : L'import relatif ne s'applique que pour les modules du package. Pour les autres imports (datetime par exemple), il n'y a a pas d'import relatif et donc rien à changer !!!*

Activité complémentaire

Partie 6 : finalisation des méthodes de « obj_ihm.py »

La méthode « **`menu_ajouter_auteur()`** » de la classe « **`Ihm`** » n'est pas terminée…

44. Complétez son code en utilisant le descriptif donné dans sa « ***docstring*** » et pour obtenir le rendu dans l'interface texte suivant :

```
--> Entrez le nom de l'auteur(e) ? :
********** Erreur : Valeur saisie incorrecte...
--> Entrez le nom de l'auteur(e) ? : Humm
--> Son prénom ? :
********** Erreur : Valeur saisie incorrecte...
--> Son prénom ? : philibert
--> Son pays de naissance (rien si inconnu) ? : France
--> Et sa date de naissance (1/1/1900 si inconnue) (JJ/MM/AAAA) ? :
********** Erreur : Valeur saisie incorrecte...
--> Et sa date de naissance (1/1/1900 si inconnue) (JJ/MM/AAAA) ? : 2df5fdg
********** Erreur : Valeur saisie incorrecte...
--> Et sa date de naissance (1/1/1900 si inconnue) (JJ/MM/AAAA) ? : 27/10/1991
----> L'auteur(e) 'Philibert HUMM (né(e) le 27/10/1991 en France)' a bien été créé(e)...

      Appuyer sur Entrée pour continuer...
```

Saisie vide…

Saisie vide…

Saisie vide…

Saisie d'une donnée qui n'est pas une date…

Il y a formatage du nom et prénom…

Même chose pour la méthode « **`menu_ajouter_membre()`** » de la classe « **`Ihm`** »…

45. Complétez son code en utilisant le descriptif donné dans sa « ***docstring*** » et pour obtenir le rendu dans l'interface texte suivant :

```
--> Entrez le nom du membre ? :
********** Erreur : Valeur saisie incorrecte...
--> Entrez le nom du membre ? : einstein
--> Son prénom ? :
********** Erreur : Valeur saisie incorrecte...
--> Son prénom ? : albert
--> Et sa date de naissance (JJ/MM/AAAA) ? :
********** Erreur : Valeur saisie incorrecte...
--> Et sa date de naissance (JJ/MM/AAAA) ? : 14/03/1879
----> Le membre 'Albert EINSTEIN (né(e) le 14/03/1879)' a bien été créé(e)...

        Appuyer sur Entrée pour continuer...
```

Saisie vide…

Saisie vide…

Saisie vide ou d'une donnée qui n'est pas une date…

Il y a formatage du nom et prénom…

Et aussi pour la méthode « **`menu_supprimer_membre()`** » de la classe « **`Ihm`** »…

46. Complétez son code en utilisant le descriptif donné dans sa « ***docstring*** » et pour obtenir les rendus dans l'interface texte suivant :

```
-------------------- Membres --------------------
1.    : Arman BEDROSSIAN (né(e) le 12/03/1998)
...
8.    : Albert EINSTEIN (né(e) le 14/03/1879)

      --> Choisissez le ou la membre à radier (numéro) : 8
      ----> Le membre 'Albert EINSTEIN' ne peut pas être radié car il a des emprunts en cours...

            Appuyer sur Entrée pour continuer...
```

```
-------------------- Membres --------------------
1.    : Arman BEDROSSIAN (né(e) le 12/03/1998)
...
8.    : Albert EINSTEIN (né(e) le 14/03/1879)

      --> Choisissez le ou la membre à radier (numéro) : 8
      ----> Le membre 'Albert EINSTEIN' peut être radié des membres de la bibliothèque...
      ------> Veuillez confirmer sa suppression (OUI) ? :
      ------> Suppression annulée...

            Appuyer sur Entrée pour continuer...
```

Saisie vide ou différente de `'OUI'`…

```
-------------------- Membres --------------------
1.    : Arman BEDROSSIAN (né(e) le 12/03/1998)
...
8.    : Albert EINSTEIN (né(e) le 14/03/1879)

      --> Choisissez le ou la membre à radier (numéro) : 8
      ----> Le membre 'Albert EINSTEIN' peut être radié des membres de la bibliothèque...
      ------> Veuillez confirmer sa suppression (OUI) ? : OUI
      --------> Le membre 'Albert EINSTEIN' a été radié...

            Appuyer sur Entrée pour continuer...
```

Idem pour la méthode « **`menu_supprimer_livre()`** » de la classe « **`Ihm`** »…

47. Complétez son code en utilisant le descriptif donné dans sa « ***docstring*** » et pour obtenir les rendus dans l'interface texte suivant :

```
-------------------- Livres --------------------
1.  : 'Harry Potter à l'école des sorciers' de J.K. ROWLING (ISBN: 9780747532743, publié en 1997) - Dispo
2.  : 'Le Seigneur des Anneaux' de J.R.R. TOLKIEN (ISBN: 9780261103573, publié en 1954) - Dispo
3.  : 'Les Misérables' de Victor HUGO (ISBN: 9782070408501, publié en 1862) - Dispo
...
22. : 'Le Scarabée d'or' de Edgar Allan POE (ISBN: 9782070362557, publié en 1843) - Dispo
23. : 'La Peste' de Albert CAMUS (ISBN: 9782070362564, publié en 1947) - NON Dispo
24. : 'Roman Fleuve' de Philibert HUMM (ISBN: 9782073019448, publié en 2022) - Dispo

--> Choisissez le livre à supprimer (numéro) : 23
----> Le livre 'La Peste' ne peut pas être supprimé car il est emprunté...

        Appuyer sur Entrée pour continuer...
```

```
-------------------- Livres --------------------
1.  : 'Harry Potter à l'école des sorciers' de J.K. ROWLING (ISBN: 9780747532743, publié en 1997) - Dispo
2.  : 'Le Seigneur des Anneaux' de J.R.R. TOLKIEN (ISBN: 9780261103573, publié en 1954) - Dispo
3.  : 'Les Misérables' de Victor HUGO (ISBN: 9782070408501, publié en 1862) - Dispo
...
22. : 'Le Scarabée d'or' de Edgar Allan POE (ISBN: 9782070362557, publié en 1843) - Dispo
23. : 'La Peste' de Albert CAMUS (ISBN: 9782070362564, publié en 1947) - Dispo
24. : 'Roman Fleuve' de Philibert HUMM (ISBN: 9782073019448, publié en 2022) - Dispo

--> Choisissez le livre à supprimer (numéro) : 23
----> Le livre 'La Peste' peut être supprimé de la bibliothèque...
------> Veuillez confirmer sa suppression (OUI) ? :
------> Suppression annulée...

        Appuyer sur Entrée pour continuer...
```

Saisie vide ou différente de `'OUI'`…

```
-------------------- Livres --------------------
1.  : 'Harry Potter à l'école des sorciers' de J.K. ROWLING (ISBN: 9780747532743, publié en 1997) - Dispo
2.  : 'Le Seigneur des Anneaux' de J.R.R. TOLKIEN (ISBN: 9780261103573, publié en 1954) - Dispo
3.  : 'Les Misérables' de Victor HUGO (ISBN: 9782070408501, publié en 1862) - Dispo
...
22. : 'Le Scarabée d'or' de Edgar Allan POE (ISBN: 9782070362557, publié en 1843) - Dispo
23. : 'La Peste' de Albert CAMUS (ISBN: 9782070362564, publié en 1947) - Dispo
24. : 'Roman Fleuve' de Philibert HUMM (ISBN: 9782073019448, publié en 2022) - Dispo

--> Choisissez le livre à supprimer (numéro) : 23
----> Le livre 'La Peste' peut être supprimé de la bibliothèque...
------> Veuillez confirmer sa suppression (OUI) ? : OUI
--------> Le livre 'La Peste' a été supprimé...

        Appuyer sur Entrée pour continuer...
```

Et pour finir, il reste encore la méthode « **`menu_ajouter_livre()`** » de la classe « **`Ihm`** » à compléter…

48. Complétez son code en utilisant le descriptif donné dans sa « ***docstring*** » et pour obtenir le rendu dans l'interface texte suivant :

```
---------- Auteurs ----------
1.    : J.K. ROWLING (né(e) le 31/07/1965 en Angleterre)
...
12.   : Philibert HUMM (né(e) le 27/10/1991 en France)

      --> Choisissez un auteur (numéro) : 0
      ********** Erreur : choix invalide...
      --> Choisissez un auteur (numéro) : 12
      --> Entrez le titre du livre ? :
      ********** Erreur : Valeur saisie incorrecte...
      --> Entrez le titre du livre ? : Roman Fleuve
      --> Son numéro ISBN (rien si inconnu) ? : 9782073019448
      --> Son année de publication (0 si inconnue) ? : a
      ********** Erreur : Valeur saisie incorrecte...
      --> Son année de publication (0 si inconnue) ? : 2022
      ----> Le livre 'Roman Fleuve' de Philibert HUMM (ISBN: 9782073019448, publié en 2022) a bien été créé...

            Appuyer sur Entrée pour continuer...
```

Saisie vide ou incorrecte…

Saisie vide…

Saisie vide ou d'une donnée qui n'est pas un entier…