Projet Bibliothèque

Alexandre et Noé

Objectifs

L'objectif est de concevoir une application pour constituer et suivre une bibliothèque de livres. L'idée est de pouvoir collecter des livres sur le web (web scraping) pour constituer une bibliothèque, et générer divers catalogues de cette bibliothèque. On s'intéresse ici aux livres au format EPUB et PDF. Mais l'application doit être extensible de façon à pouvoir facilement ajouter d'autres formats.

Fonctionnalités

- Créer un livre à partir d'un fichier (PDF ou EPUB), en accédant aux métadonnées des fichiers.
- Créer une bibliothèque à partir d'un dossier.
- Créer une bibliothèque à partir d'un site internet (web scrapping).
- Possibilité d'ajouter un nouveau format de livre simplement.

Etapes de réalisation

Etape I

- Création du dépot privé sur GitHub et affectation des droits de lecture et d'écriture au binôme.
- Créations des classes.
- Implémentation des fonctions.

Etape II

Consultations des projets des autres groupes. Nous avons conservé l'intégralité de notre code.

3. Etape III

- Implémentation des fonctions pour générer des rapports sur le contenu de la bibliothèque.
- Création de l'application principale.
- Prise en charge des différents arguments dont les fichiers de configuration.

Application

Ce programme est conçu pour gérer une bibliothèque de livres en ligne de commande. Il prend en charge plusieurs options pour alimenter la bibliothèque, générer des rapports sur les livres et les auteurs, en fonction des arguments fournis lors de son exécution.

Modules nécessaires

Pour utiliser le programme, il faut installer les modules suivants :

- `requests` pour accéder aux pages web.
- `bs4` from `beautifulsoup4` pour analyser les pages web.
- `urllib3` pour extraire les liens des pages web.
- `fitz` from `PyMuPDF` pour extraire les métadonnées des fichiers PDF.
- `detect` from `langdetect` pour détecter la langue du fichier.
- `re` permet de créer des expressions régulières
- `epub` from `ebooklib` pour extraire les métadonnées des fichiers EPUB.
- `warnings` pour gérer les avertissements.
- `os` pour gérer les fichiers et dossiers.
- `shutil` pour copier des fichiers
- `configparser` pour lire les fichiers de configuration
- `sys` permet d'avoir accès aux options données au programme
- `fpdf` permet de créer des fichiers PDF

Vous pouvez les installer en utilisant la commande suivante :

- `pip install requests beautifulsoup4 urllib3 PyMuPDF langdetect ebooklib fpdf`

Options disponibles :

1. **Aucun argument fourni :** Si aucun argument n'est fourni lors de l'exécution du programme, il affichera des exemples d'options disponibles :

```
Veuillez indiquer les options à utiliser quelques exemples ci-dessous :
https://math.univ-angers.fr/~jaclin/biblio/livres/
rapports
-c config.conf
-c config.conf rapports
-c config.conf https://math.univ-angers.fr/~jaclin/biblio/livres/
-c config.conf https://math.univ-angers.fr/~jaclin/biblio/livres/ 10
```

2. **Un argument fourni :**
   - Si l'argument est `-c`, le programme attend un fichier de configuration à indiquer.
   - Si l'argument est "rapports", des rapports sur les livres et les auteurs seront générés au format PDF et EPUB.
   - Si l'argument est un lien web, le programme alimentera la bibliothèque à partir de ce lien.
3. **Deux arguments fournis :**
   - Si les deux arguments sont `-c` suivi d'un fichier de configuration, le programme utilisera ce fichier pour initialiser la bibliothèque.
   - Si le premier argument est un lien web et le deuxième un nombre, le programme effectuera un scraping de la bibliothèque à partir du lien web avec la profondeur de recherche donnée.
4. **Trois arguments fournis :**
   - Si les trois arguments sont `-c`, un fichier de configuration, et "rapports", le programme générera des rapports en utilisant le chemin spécifié dans le fichier de configuration.
   - Si le premier argument est `-c` avec un fichier de configuration et le troisième un lien web, le programme alimentera la bibliothèque à partir du lien spécifié.
5. **Quatre arguments fournis :**
   - Si les quatre arguments sont `-c` avec un fichier de configuration, un lien web, et un nombre, le programme effectuera un scraping à partir du lien web pour récupérer les livres avec la profondeur de recherche spécifié.

Exemples d'utilisation :

- Pour générer des rapports de livres et d'auteurs :

```
python nom_du_programme.py rapports
```

- Pour utiliser un fichier de configuration spécifique et alimenter la bibliothèque depuis un lien web :

```
python nom_du_programme.py -c config.conf https://math.univ-angers.fr/~jaclin/biblio/livres/
```

- Pour effectuer un scraping de la bibliothèque depuis un lien web avec une profondeur de recherche donnée : ```bash python nom_du_programme.py https://math.univ-angers.fr/~jaclin/biblio/livres/ 2

Ajout d'un nouveau format de livre

Le programme de base permet de gérer les livres au format PDF et EPUB. Il est cependant possible de rajouter de nouveaux formats de livres. Pour ajouter un nouveau format de livre, il faut :

- Importer les modules nécessaires pour gérer le nouveau format de livre.
- Pour ajouter un nouveau format de livre, il faut créer une nouvelle classe héritant de la classe `Livre`.
- Implémenter la fonction `recup_format(path)` dans `fonctions_fichier.py` qui permet de récupérer les métadonnées du livre à partir du chemin d'accès du livre.
- Rajouter un élément dans le tuple `extensions` dans `fonctions_fichier.py` contenant l'extension du nouveau format de livre.
- Ajouter un `case '.format'` dans la fonction `telecharger de bibli` et dans le constructeur de `simple_bibli`.
- Pour ajouter un nouveau format de rapport, il faut implémenter une nouvelle fonction `rapport_format(dossierArrive, contenu, sortie)` dans `fonctions_fichier.py` qui permettra d'écrire ce nouveau rapport. Ainsi que rajouter un *case* dans les fonctions `rapport_livres(self, format, fichier='./rapport')` et `rapport_auteurs(self, format, fichier='./rapport')` avec le format correspondant.

Détails techniques

1. Classe : `simple_blibli`

La classe `simple_bibli` hérite de `base_bibli` permet de créer une bibiliothèque à partir de livres stockés sur l'ordinateur. Elle s'utilise comme suit :

- Création de la bibliothèque avec le chemin d'accès du dossier qui sera utilisé pour stocker les livres. S'il n'existe pas, il est créé et s'il n'est pas renseigné un dossier *défaut* sera créé.
- On y ajoute des livres en utilisant la fonction `ajouter(self, livre)` qui prend en paramètre obligatoire, le chemin d'accès du livre que l'on veut ajouter au dossier.
- `rapport_livres(self, format, fichier='./rapport')` : génère un rapport au format donné en paramètre listant tous les livres présents dans la bibliothèque, un par un. Ce rapport est enregistré dans le dossier donné en paramètre.
- `rapport_auteurs(self, format, fichier='./rapport')` : génère un rapport au format donné en paramètre listant tous les livres présents dans la bibliothèque, triés par auteur. Ce rapport est enregistré dans le dossier donné en paramètre.

2. Classe : `bibli`

La classe `bibli` hérite de `simple_bibli` et permet en plus d'alimenter la bibliothèque avec des livres téléchargés sur internet. Elle s'utilise comme suit :

- On crée la bibliothèque de la même manière que `simple_blibli` .
- `telecharger(self, url)` : télécharge le livre de l'url donné et l'ajoute à la bibliothèque.
- `alimenter(self, url, nbmax=10)` : télécharge les livres présent dans la page web correspondant à l'url donné, au maximum *nbmax*.

3. Classe : `bibli_scrap`

La classe `bibli_scrap` hérite de `bibli` et permet en plus de récupérer les livres d'une page web à partir d'un web scrapping. Elle s'utilise comme suit :

- On crée la bibliothèque de la même manière que `bibli` .
- `scrap(self, url, profondeur=0, nbmax=10)` récupère les livres de la page web correspondant à l'url donné, au maximum *nbmax*. Si *nbmax* n'est pas atteint et que *profondeur* est supérieure à 0, on récupère les livres des pages web récupérés sur la page web de l'*url* donné. On effectura cette opération au maximum *profondeur* fois.

4. Fichier : `fonctions_fichier.py`

Ce fichier contient les fonctions permettant de récupérer les métadonnées des fichiers PDF et EPUB. Il contient aussi les fonctions permettant de récupérer les livres d'une page web.

- `extensions` : tuple contenant les extensions des fichiers supportés.
- `telecharger(url)` : télécharge le livre de l'url donné et l'enregistre dans le dossier *telechargements*.
- `recup_date_langue(pdf_path, numero_page)` : récupère la date et la langue du livre PDF à partir du chemin d'accès du livre et du numéro de la page.
- `recup_pdf(pdf_path)` : récupère les métadonnées du livre PDF à partir du chemin d'accès du livre.
- `recup_EPUB(epub_path)` : récupère les métadonnées du livre EPUB à partir du chemin d'accès du livre.
- `recup_liens_livres(url)` : récupère les liens des livres d'une page web à partir de l'url de la page web.
- `recup_liens_externes(url)` : récupère les liens des pages web externes à partir de l'url de la page web.
- `est_lien_web(chaine)` : vérifie si la chaîne de caractère est un lien web.
- `est_url_valide(url)` : vérifie si l'url est valide, à savoir s'il dirige effectivement vers une page web différente.
- `lire_config(chemin_fichier)` : récupère le contenu du fichier de configuration donné en paramètre.
- `config_defaut()` : récupère les paramètres donnés par le fichier de configuration.
- `rapport_EPUB(dossierArrive, contenu, sortie)` : créer un document au format EPUB à partir du contenu dans le dossier *dossierArrive* et ayant pour nom *rapport_'sortie'.epub*.
- Classe `PDF`
- `rapport_PDF(dossierArrive, contenu, sortie)` : créer un document au format PDF à partir du contenu dans le dossier *dossierArrive* et ayant pour nom *rapport_'sortie'.pdf*.