TD3 : web scraping

L'objectif du travail de ce TD est de récupérer des informations textuelles à partir d'une page de résultats de recherche sur le site des collections du Louvre : https://collections.louvre.fr/recherche

Télécharger le fichier source depuis l'espace-cours Cursus dans un répertoire de travail et ouvrir ce répertoire dans VSCode.

Partie 1 : récupération du contenu d'une page d'œuvre

L'objectif de cette section est de récupérer le titre et les informations complètes relatives à **une** œuvre à partir de l'url de la page présentant cette œuvre.

Vous pourrez l'URL suivante comme page de test :
`https://collections.louvre.fr/ark:/53355/cl010096624`

Q 1.1. Dans le fichier *td_scraping.py*, écrire une fonction `get_bsobj_from_url` qui prend en entrée une url, ouvre la page associée et renvoie l'objet BeautifulSoup correspondant au contenu html de cette page.

Q 1.2. Écrire une fonction `get_name` qui prend en entrée un objet BeautifulSoup et renvoie le nom de l'œuvre. Pour cela, identifier la balise correspondante grâce à l'inspecteur d'élément du navigateur et une éventuelle classe pertinente permettant de sélectionner précisément cette balise.

Vérifier le bon fonctionnement de cette fonction avec la page de test données ci-dessus.

Q 1.3. Écrire une fonction `get_section_info` qui prend en entrée un objet BeautifulSoup d'une page de présentation d'une œuvre, et renvoie le code html (sous forme d'un objet Tag de BeautifulSoup) de la section présentant les informations complètes de l'œuvre (partie principale de la page, sur fond blanc). Dans le code HTML cette zone de la page correspond à une balise `<section>` : trouver une classe pertinente permettant de sélectionner précisément cette section.

Vérifier le fonctionnement de votre fonction sur la page de test.

Q 1.4. Dans le fichier *td_scraping.py* le code de la fonction `extract_text` vous est fourni. Celle-ci prend en entrée un objet Tag de BeautifulSoup, tel que renvoyé par la fonction précédente par exemple, et extrait puis renvoie son contenu textuel sous forme d'une chaîne de caractères. Tester ce que fait cette fonction sur le contenu renvoyé par la question précédente.

Q 1.5. En utilisant toutes les fonctions précédentes, écrire une fonction de synthèse `parse_url` qui prend en entrée une URL et qui renvoie un tuple contenant trois éléments : l'url, le titre de l'œuvre, et le contenu textuel de la section d'informations.

Tester avec la page d'œuvre de test utilisée ci-dessus. Vérifier également que votre fonction permet d'extraire les informations à partir **d'autres pages d'œuvre** accessibles depuis la page de résultats de recherche suivante (choisir les œuvres que vous voulez et récupérer l'URL des pages correspondantes) :

https://collections.louvre.fr/recherche?limit=20&location%5B0%5D=141080&typology%5B0%5D=11

Partie 2 : Construction d'un corpus d'information pour un ensemble de pages

L'objectif de cette section est de regrouper les textes et informations de plusieurs pages web dans un unique jeu de données.

Pour cela, nous allons utiliser la page de recherche du site des collections du Louvre.

Par exemple, la page de résultats suivantes :

https://collections.louvre.fr/recherche?limit=20&location%5B0%5D=141080&typology%5B0%5D=11

présente les 44 œuvres de la catégorie « Art du livre » du musée du Louvre. Ou plus précisément, cette page de résultats contient les liens vers les pages de présentation des 20 premières œuvres, ainsi qu'un lien de navigation vers la page de résultats suivante et une indication sur le nombre total de pages de résultats.

Q 2.1. Dans la première page de résultats « Art du livre » (url donnée ci-dessus), identifier le type de balise qui contient chaque bloc de présentation résumée d'une œuvre.

Indication : balises de la forme `<article class="c????????h">`*.*

Écrire une fonction `build_url_list` qui prend en premier paramètre l'url d'une page de résultats de recherche (comme celle donnée en exemple à la question précédente), en second paramètre l'url de base du site des collections du Louvre (`https://collections.louvre.fr`) et qui renvoie la liste de toutes les url **complètes** des pages de présentation d'œuvre disponibles sur cette page de résultat (il y aura par exemple 20 url exactement pour la première page de résultat « Art du livre » au musée du Louvre).

Indication : il faudra ajouter à chaque lien récupéré l'url de base du site du Louvre afin de former l'url complète de la page de présentation d'œuvre correspondante.

Q 2.2. Écrire maintenant une fonction `build_all_url_list` qui prend en premier paramètre l'url de la **première** page de résultats d'une recherche, en second paramètre l'url de base du site des collections du Louvre et qui renvoie sous forme d'une liste

l'ensemble des url des pages de présentation des œuvres correspondant à la recherche (il faudra par exemple récupérer 44 url au total pour la recherche « Art du livre » au musée du Louvre).

Pour cela vous récupérerez **automatiquement** l'information donnant le nombre total de pages de résultats et vous formerez les urls des différentes pages de résultats en ajoutant « `&page=X` » à la suite de l'url de la première page (ou `X` est le numéro de la page de résultat, `X ≥ 2`). A partir de ces url de page de résultats, vous utiliserez sur chacune d'entre elle la fonction `build_url_list` précédente afin d'obtenir la liste complète des pages de présentation d'œuvre.

Q 2.3. Dans le programme principal, utiliser la programmation parallèle pour récupérer une liste de tuples de 3 informations `(url, titre, texte_info)`, pour l'ensemble des url récupérées à la question précédente.

Q 2.4. Enregistrer cette liste dans un fichier `arts_du_livre.pick` à l'aide du module `pickle`.

Q 2.5. En utilisant les mêmes appels de fonction, créer un fichier `peintures.pick` contenant les informations `(url, titre, texte_info)` des œuvres issues de l'url suivante :

`https://collections.louvre.fr/recherche?limit=20&location[0]=190540&typology[0]=22`

(résultats de recherche « Peintures » au Louvre-Lens)