Mini Projet 3 : Scrapper un Shopify

Consignes : cette évaluation sera réalisée sous la forme d'un mini-projet à terminer à la maison. Vous remettrez à votre enseignant(e) un lien vers un **dépôt GitHub privé** ayant pour nom **B2P2_Python_TE02_NOM_Prénom** et **l'ajout de l'utilisateur mjouan23 devra être fait**.
Le dépôt devra contenir les éléments suivants : un **dossier csv** contant le fichier csv généré, un **fichier script.py** et un **fichier requirements.txt** contenant la liste des bibliothèques utilisées **sans votre dossier venv. Le bon respect des consignes rapportera 1 point de bonus.**

Pour un souci de compétences en interne, votre client souhaite migrer sa boutique Shopify en boutique WordPress/WooCommerce. Pour ce faire, vous allez devoir **créer un script de migration Python permettant de parcourir le JSON products.json du Shopify de votre choix afin de le formater au format CSV** à importer dans WooCommerce.

Shopify libre

Choisissez le Shopify pour lequel vous souhaitez migrer la liste des produits parmi les liens Shopify référencés sur le lien ci-dessous :

https://www.shopify.com/fr/blog/48153989-40-superbes-boutiques-en-ligne-creees-avec-shopify

Une fois votre choix effectué, rendez-vous à l'adresse suivante : [*lien de votre Shopify*]/products.json afin d'accéder à la liste des produits du Shopify sous la forme d'un JSON.

Exercice 1

Créer votre projet avec un environnement et installez les bibliothèques externes suivantes :

- pandas
- requests

Pensez à générer un fichier requirements.txt afin de lister les bibliothèques à installer.

Vous devrez importer les bibliothèques installées ainsi que bibliothèques internes : json et csv.

Pensez à générer à votre dépôt et à ajouter le dossier venv dans le .gitignore

Exercice 2

En utilisant la bibliothèque « requests », écrivez une fonction « get_json » permettant de charger le JSON de n'importe quelle URL Shopify passé en paramètre. Cette fonction possédera 3 paramètres :

- **url** : le lien du Shopify (sans le products.json)
- **limit** : le nombre de produits à charger
- **page** : le numéro de la page à partir de laquelle vous souhaitez récupérer les produits

Cette fonction prendra en compte les erreurs suivantes et affichera les messages d'erreur en conséquent :

- **Erreur HTTP** => « Erreur http : » + error
- **Erreur Connexion** => « Connexion impossible : » + error
- **Erreur Timeout** => « Délai dépassé » + error
- **Erreur autre** => « Erreur : » + error

Si aucune erreur détectée, renvoyez un objet JSON avec les résultats.

Pour la gestion des erreurs pensez à la méthode raise_for_status()

Exercice 3

Ecrivez une fonction « json_to_df » permettant de retourner un dataframe à partir d'un JSON passé en paramètre à l'aide de la bibliothèque pandas et json.

Exercice 4

Ecrivez une fonction « get_products » permettant de renvoyer un dataframe contenant TOUS les produits d'un Shopify passé en paramètre.

Cette fonction devra appeler les 2 fonctions précédentes et boucler jusqu'à ce qu'il n'y ait plus aucun produit à récupérer sur le Shopify.

Elle devra « concaténer » tous les résultats dans un seul et unique dataframe « df_products » en prenant soin de réinitialiser les index afin de garder une certaine cohérence.

Vous allez devoir utiliser une variable « page »

Exercice 5

A l'aide de la documentation « documentation_shopify_to_woocommerce.csv », écrivez une fonction « get_variants » permettant de retraiter le dataframe « df_products » afin d'en extraire tous les variants pour chaque produit Shopify et récupérer toutes les informations dont le client aura besoin pour son importation dans WooCommerce en suivant la documentation fournie.

Les champs avec l'indication « Valeur par défaut » n'auront pas besoin d'être présents dans le dataframe. Cette méthode devra retourner un dataframe « df_variants »

Exercice 6

Ecrivez la fonction « get_csv » permettant de générer un fichier CSV à partir d'un dataframe « df_variants » passé en paramètre.

Exercice 7

1. Demander à l'utilisateur de saisir une url puis la récupérer
2. Appelez la fonction « get_products » en passant en paramètre l'url saisie par l'utilisateur
3. Appelez la fonction « get_variants » en passant en paramètre le dataframe « df_products » renvoyé par la fonction « get_products »
4. Appelez le fonction « get_csv » en passant en paramètre le dataframe « df_variants » renvoyé par la fonction « get_variants »

Une fois votre fichier généré, testez-le dans un WooCommerce fraîchement installé en suivant la procédure suivante :

https://woocommerce.com/document/importation-exportation-de-produits-au-format-csv/

Si le fichier a correctement été généré, le mappage des champs se fera automatiquement.

Lien Utiles

Documentation des champs WooCommerce :
https://github.com/woocommerce/woocommerce/wiki/Product-CSV-Import-Schema

Exemple d'un lien vers la liste des produits Shopify :
https://www.pandatea.fr/products.json

Documentation Python Pandas sur les DataFrame :
https://pandas.pydata.org/docs/reference/frame.html