Programme plusieurs\_categories.py

Home\_url = url de la page d’accueil

Html\_info = contenu html de la page d’accueil

Categories : la commande html\_info.find(‘ul’, {‘class’ : ‘nav nav\_list’}).find(‘ul’).findAll(‘a’) stocke toutes les balises <a></a> contenues dans la balise <u></u> qui contien la class « nav nav\_list » et qui correspond à la liste des catégories du site.

On procède à une compréhension de liste pour créer une liste dont chaque élément correspondra à une catégorie de notre site. La commande .text.replace(‘\n’,’’).strip() permet d’isoler le texte à l’intérieur des balises, de supprimer les caractères de retours à la ligne et les espaces de chaque côté du texte.

La prochaine boucle « for » va initialiser le fichier csv où seront stockées nos données. On crée au même moment un dossier au nom de la catégorie s’il n’est pas déjà existant. Cette action est possible grâce à la bibliothèque os de python et la commande os.mkdir().

La variable i parcours la liste des catégories que nous venons de créer. A chaque itération on ouvre un fichier csv en mode écriture avec un encodage utf8 et pour titre

« page\_produit\_{i}.csv »

Ainsi la catégorie « travel » générera un fichier csv nommé : page\_produit\_Travel.csv.

On en profite pour stocker ce nom de fichier dans une liste files\_list en vue d’une utilisation future.

On va chercher dans le dictionnaire «maintenance » les entêtes des colonnes du fichier csv.

Comme il s’agit d’une seule chaine de caractère on utilise la méthode .split(‘ ;’) pour obtenir une liste où chaque entête est un élément.

On arrive alors dans une boucle for pour parcourir cette liste.

On utilise enumerate() pour stocker le numéro d’itération dans une variable n. Cette variable n nous permet de déterminer si on est arrivé en bout de liste. La variable header prendra la chaine de caractère comme valeur.

Tant que ce n’est pas le cas on écrit la valeur de la variable header suivi du séparateur de colonne. La valeur de ce séparateur est contenue dans notre variable maintenance à la clé « csv\_separator ». il est important de choisir « ; » car par défaut la séparateur utilisé est « , ». Or la virgule est très présente dans les textes de résumé.

Quand on arrive au dernier élément de la liste soit lorsque n = len(keys) – 1, on écrit la dernière valeur de header suivi cette fois- ci d’un retour à la ligne.

La prochaine étape du programme est la boucle for nb, cat in enumerate(list) où nb est le numéro d’itération et cat la valeur de l’élément liste, soit le nom de la catégorie.

Il faut maintenant initialiser le url de la page d’accueil de la catégorie considérée. Les urls du site sont construits de sorte que le nom de la catégorie est suivi d’un numéro. La première catégorie porte le numéro 2. On sélectionne alors la partie commune à toutes les urls et on ajoute un ligne de commande permettant d’ajouter le nom de la catégorie en enlevant les majuscules qui ne sont pas dans l’url et en remplaçant les espaces entre les mots par un underscore. Enfin, nb commence à zero mais la première catégorie commence à 2.

Html\_info\_cat : contenu html de la page d’accueil de la catégorie.

On procède alors à un test pour savoir si la catégorie possède une ou plusieurs pages car dans ce dernier cas la procédure change légèrement.

Une catégorie avec plusieurs page possède une balise dans son code html qu’une catégorie avec une seul page ne contient pas. On vérifie donc la présence de cette balise dans le contenu html avec un simple « if ».

Si la balise n’est pas présente on fait appel aux fonctions find\_data() et write\_data().

Si la balise est présente, un travail en amont est nécessaire. Il y a plusieurs pages et chacune possède une url à soi. Il faut construire et stocker ces urls. L’indicatif de la page est donnée simplement par page\_i.html.

Il faut donc enlever le index.html de l’url de la page d’accueil et le remplace par l’indicatif de la page. Le nombre de page nous est donnée par une balise particulière. On fait alors une compréhension de liste avec un range entre 1 et la valeur du nombre de page +1.

Chacune des urls est ajouté à la liste ‘number\_of\_pages’. On fait ensuite appel aux fonction find\_data() et write\_data().

La suite de l’algo respecte le même process peut import qu’il y ait une ou plusieurs pages.

On affect à une variable l’url qui correspond à la page affichant les différents livres.

On appelle alors la fonction listing() qui prend en argument l’url sus-mentionnée et une liste vide que nous avons appelé urls.

On charge le code html de la page.

On crée une variable balise qui va contenir toutes les balises <h3> à l’intérieur de la balise <section>.

Celles-ci correspondent aux titres des livres contenus sur la page. A partir de cette variable on construit une liste dont les éléments sont également des listes. Ces sous-liste contiennent deux éléments : le titre du livre ainsi que l’index de la page. Il est alors possible de construire les urls des pages des livres.

Chacune respecte la même architecture :

<http://book.toscrape.com/catalogue/titredulivre/indexdelapage>

On parcours alors la liste que nous juste de créer. On construit alors chaque url une à une et on la stocke dans la liste vide que nous avions passé en deuxième argument de la fonction listing().

A ce stade de la fonction, on possède une liste appelée inventory qui contient toutes les urls des livres présents sur la page étudiée. Dans le cas d’une catégorie avec plusieurs pages, on répète l’opération avec l’url de la page suivante et ainsi de suite jusqu’à ce qu’inventory contienne les urls de la totalité des livre de la catégorie.

On initialise un dicitonnaire vide.

On parcours avec enumerate() la liste des urls.

On ajoute un couple (clé, valeur) pour chaque url où la clé est « book {n° d’itération} » et la valeur l’url du livre.

La fonction listing se finie et renvoie le dictionnaire qui vient d’être construit.

On assigne ce dictionnaire à la variable link\_dictionary : il s’agit donc d’un dictionnaire qui contient l’url de tous les livres d’une catégorie.

La prochaine étape consiste à parcourir tous les couples du dictionnaire.

Cette action est réalisée avec la ligne for k, v in link\_dictionary.items() où k prend pour valeur la clé du dictionnaire et v la valeur qui y est associée.

Le reste de l’explication expose le processus pour une seul url. Celui-ci se répétant simplement pour chacune des urls des livres de la catégorie.

On fait ensuite appel à la fonction find\_book\_data( ) qui prend en premier argument l’url du livre et en deuxième argument la clé du dictionnaire correspondante.

Le but de cette fonction est de créer un dictionnaire dans lequel les clés correspondront aux entêtes de notre fichier csv.

Pour cela on initialise ce dictionnaire avec pour première clé ‘product\_page\_url’ et en valeur correspondante l’url du livre que l’on a passé en premier argument de la fonction.

On se procure ensuite le code html de la page avec beautifulsoup et on part à la chasse à nos informations.

On procède de la manière suivante :

Dictionary[‘entête’] = commande.

Cette commande est établie de manière empirique. Il faut sélectionner le texte sur la page web, clique droit souris puis cliquer sur examiner l’élément. Le navigateur va alors faire apparaitre le code html qui génère la page qui, coté algo est obtenu avec beautifulsoup, en surlignant la balise qui génère le texte qui nous intéresse. On utilise ensuite les méthodes internes à beautifulsoup pour isoler cette balise puis récupérer soit le text à l’intérieur soit un attribut de la balise elle-même.

Du fait de l’empirisme de la méthode, l’algo est vulnérable face à des méthodes de codage html différentes de celle du site étudié ici.

Une fois notre dictinnaire complété la fonction le renvoie.

Dans notre algorithme on crée un dictionnaire product\_description avec cette fonction.

Il est maintenant temps de passer à la dernière étape et d’appeler la fonction wrtie\_data( ). Cette fonction prend en premier argument la dictionnaire qui vient d’être créé, et en deuxième argument l’élément nb de la liste files\_list.

Cette liste contient les noms des fichiers csv que nous avons initiés en début d’algorithme et va nous être utile pour empêcher le risque d’une erreur d’orthographe lors de la saisie du nom du fichier csv dans lequel on veut écrire.

Ainsi le fonction write\_data ouvre le fichier csv correspondant à la catégorie. Il ouvre ce fichier en mode ‘a’ ce qui signifie qu’on va ajouter du texte en dessous de celui déjà existant. On crée ensuite une liste keys à partir de la variable maintenance. Cette liste est composée des entêtes du fichier csv.

On parcourt ensuite cette liste et on écrit la valeur correspondante du dictionnaire à chacune des clés.

On ajoute un retour à la ligne une fois le dictionnaire entièrement parcouru pour que les infos du prochain livre soient bien mises en dessous.

Une fois ces opérations terminées, on passe au livre suivant dans le dictionnaire link\_dictionary et on réalise les même étapes.

Ainsi de suite jusqu’à avoir parcourus tous les livres du dictionnaire.

Une fois fait, on passe à la catégorie suivante et on répète les mêmes opérations jusqu’à avoir parcourus toutes les catégories du site.

Une fois la dernière catégorie parcourue, le programme s’arrête tout seul.

Supplément pour l’enregistrement du fichier image :

Celui-ci s’effectue avec une librairie spéciale qui s’appelle urllib.request et la méthode .urlretrieve de cette librairie. Cette méthode a besoin de l’url exacte de l’image en premier argument et du nom sous lequel on veut enregistrer l’image en deuxième argument.