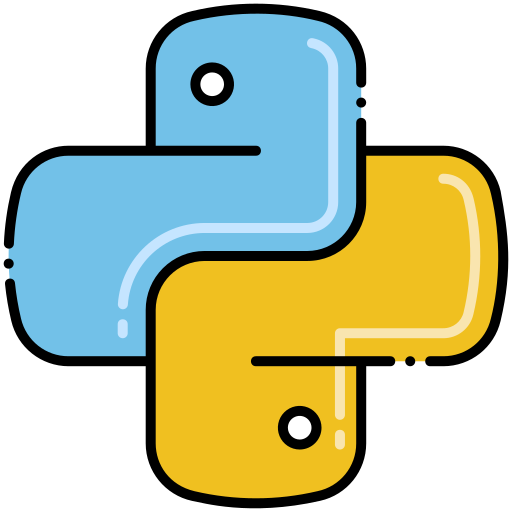
P\_Appro1 - Scrappy



Vougaz Maryline – FIN2

Lausanne

Durée : 16 périodes de 45 minutes

Schaffter Céderic

Table des matières

[1 Spécifications 3](#_Toc308526316)

[1.1 Titre 3](#_Toc308526317)

[1.2 Description 3](#_Toc308526318)

[1.3 Matériel et logiciels à disposition 3](#_Toc308526319)

[1.4 Prérequis 3](#_Toc308526320)

[1.5 Cahier des charges 3](#_Toc308526321)

[1.5.1 Objectifs et portée du projet (objectifs SMART) 3](#_Toc308526322)

[1.5.2 Caractéristiques des utilisateurs et impacts 3](#_Toc308526323)

[1.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 3](#_Toc308526324)

[1.5.4 Contraintes 3](#_Toc308526325)

[1.5.5 Travail à réaliser par l'apprenti 4](#_Toc308526326)

[1.5.6 Si le temps le permet … 4](#_Toc308526327)

[1.5.7 Méthodes de validation des solutions 4](#_Toc308526328)

[1.6 Les points suivants seront évalués 4](#_Toc308526329)

[1.7 Validation et conditions de réussite 4](#_Toc308526330)

[2 Planification Initiale 4](#_Toc308526331)

[3 Analyse 4](#_Toc308526332)

[3.1 Opportunités 4](#_Toc308526333)

[3.2 Document d’analyse et conception 4](#_Toc308526334)

[3.3 Conception des tests 5](#_Toc308526335)

[3.4 Planification détaillée 5](#_Toc308526336)

[4 Réalisation 5](#_Toc308526337)

[4.1 Dossier de Réalisation 5](#_Toc308526338)

[4.2 Modifications 5](#_Toc308526339)

[5 Tests 5](#_Toc308526340)

[5.1 Dossier des tests 5](#_Toc308526341)

[6 Conclusion 5](#_Toc308526342)

[6.1 Bilan des fonctionnalités demandées 5](#_Toc308526343)

[6.2 Bilan de la planification 5](#_Toc308526344)

[6.3 Bilan personnel 5](#_Toc308526345)

[7 Divers 6](#_Toc308526346)

[7.1 Journal de travail 6](#_Toc308526347)

[7.2 Bibliographie 6](#_Toc308526348)

[7.3 Webographie 6](#_Toc308526349)

[8 Annexes 6](#_Toc308526350)

### Spécifications

## Titre

Scrappy

## Description

Ce projet consiste en la réalisation d’un module de webscrap en python. Comme décidé dans le cahier des charge, ce module prend en charge le site de parfumdo.fr

Ma motivation principale derrière le choix de ce projet se compose d’une part de ma passion pour la parfumerie et d’autre part d’une envie de pratiquer le langage de programmation Python. En reprenant une librairie, BeautifulSoup4, que j’ai eu l’occasion d’utiliser au cours du hackathon à Zurich, auquel j’avais pris part en septembre dernier.

## Matériel et logiciels à disposition

IDE : VS Code, PyCharm & Replit

GitHub

Office 365 & LibreOffice

PC de l’école

Documentation sur internet

## Prérequis

Module I403, I404, I226A&B & C101

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART)

Réaliser un module de webscrap en Python avec la librairie BeautifulSoup4 du site Parfumdo qui écrit les informations récoltées dans un fichier .CSV en utilisant une ligne par produit, avec la présente documentation et un journal de travail, à rendre le 15 mars 2024.

### Fonctionnalités requises (du point de vue de l’utilisateur)

Pouvoir lancer le script et obtenir un fichier .CSV contenant les informations des produits du site web parfumdo.ch

A compléter par une espèce de mode d’emploi du produit. S’il s’agissait d’une montre, décrire qu’à part l’heure, il y aura la possibilité d’utiliser un chronomètre, un réveil, …

### Contraintes

Sécurité, backups, disponibilité, système utilisé, interfaces avec autres logiciels, etc.

### Travail à réaliser par l'apprenti

Un programme en Python qui utilise la librairie BeautifulSoup4 pour résolter les informations et les mettre dans un fichier .CSV.

Décrire à quoi doit ressembler le travail produit, ce qu’il faudra rendre …

### Si le temps le permet …

Objectifs complémentairesau cas où le projet n’est pas assez ambitieux dans le temps imparti…

### Méthodes de validation des solutions

Afin de valider le fonctionnement du programme, il faudra l’executer afin de constater son résultat dans le fichier .csv si retourner. Selon celui-ci et/ou le résultat et messages d’erreurs, utiliser des points d’arrêts afin de déceler les bugs et autres problèmes pouvant survenir au cours du projet.

Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…

## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Résultat du scraper retourne un fichier .csv contenant les informations appropriées aux métadonnées
* Le scraper a une fonction « GET » qui permet de récupérer le contenu d’un site passé en paramètre.
* Le scraper est capable de détecter si la page est parsable, par exemple en comparant le dom et le résultat du scrap
* Fonction de résumé pour voir combien d’éléments sont des images

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* État de fonctionnement du produit livré

### Planification Initiale

Ce paragraphe présente le planning d'origine (date de début, date de fin, vacances et congés, liste hiérarchique des tâches ou GANTT, jalons, durée totale)

Toutes les mises à jour subies par le planning sont à reporter (avec date de mise à jour) et peuvent déboucher sur plusieurs versions de plannings.

### Analyse

## Opportunités

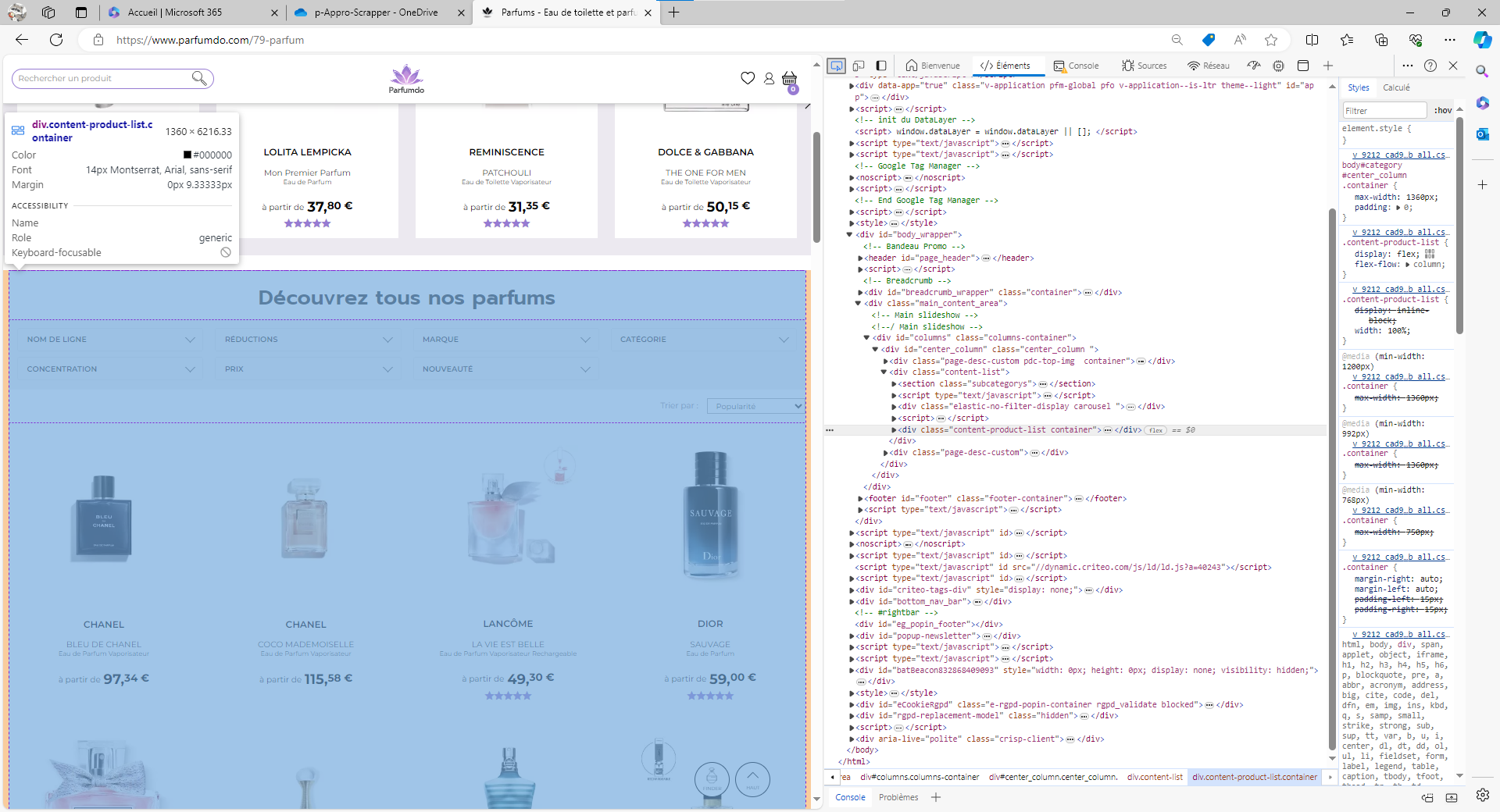
Ce projet est pour moi une opportunité de prendre un peu plus de temps afin de pratiquer un autre langage que le C#, PHP et HTML/CSS vu en cours ainsi que l’utilisation de librairie. Car au cours de notre formation nous n’avons vu ni framework, ni librairie avec les langages de programmation vu.

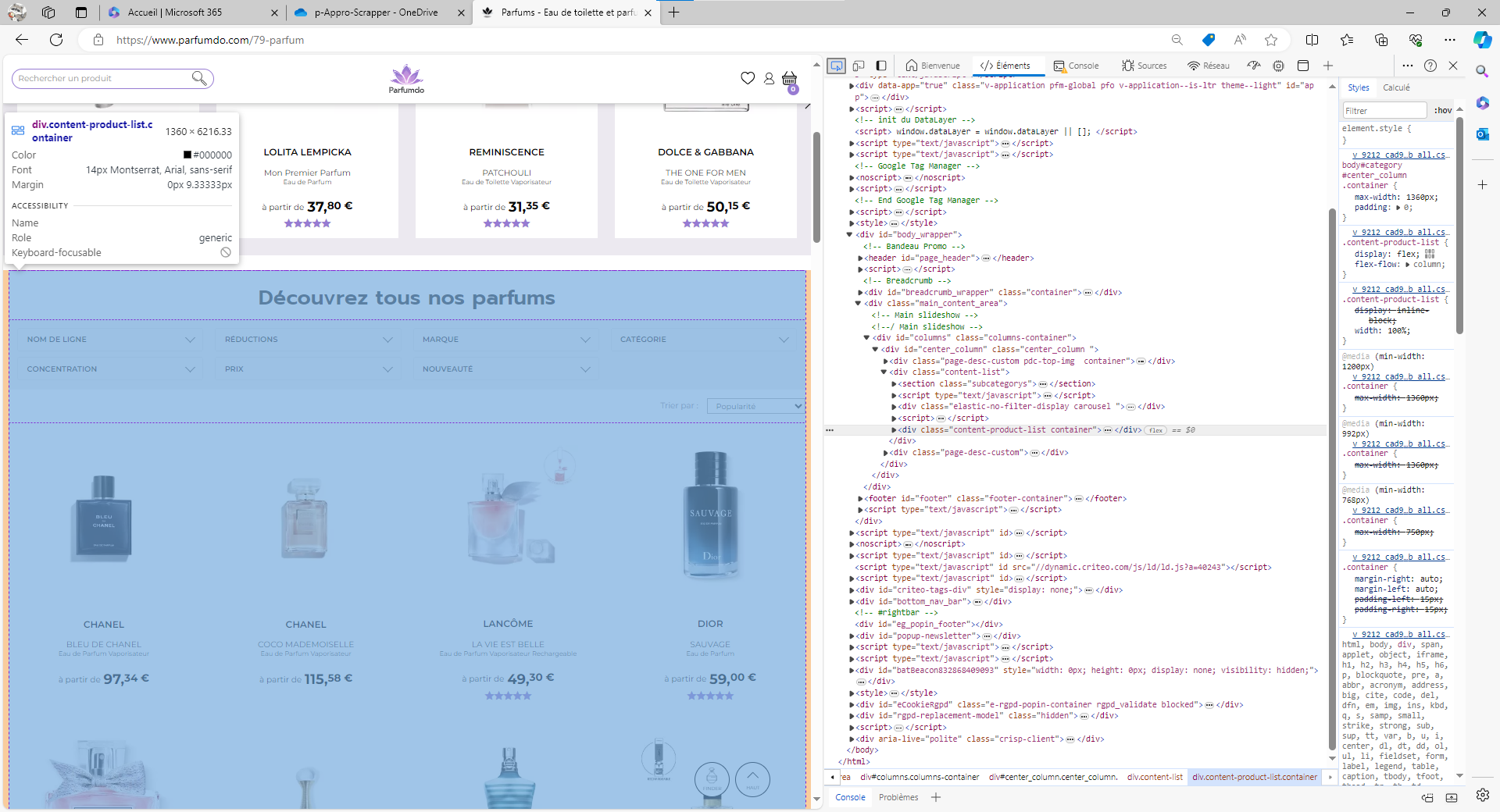
## Document d’analyse et conception

Après observations, il est ressorti que l’endroit où se trouvent les informations des produits du site web à scraper se trouvait à cet endroit dans la structure du html.

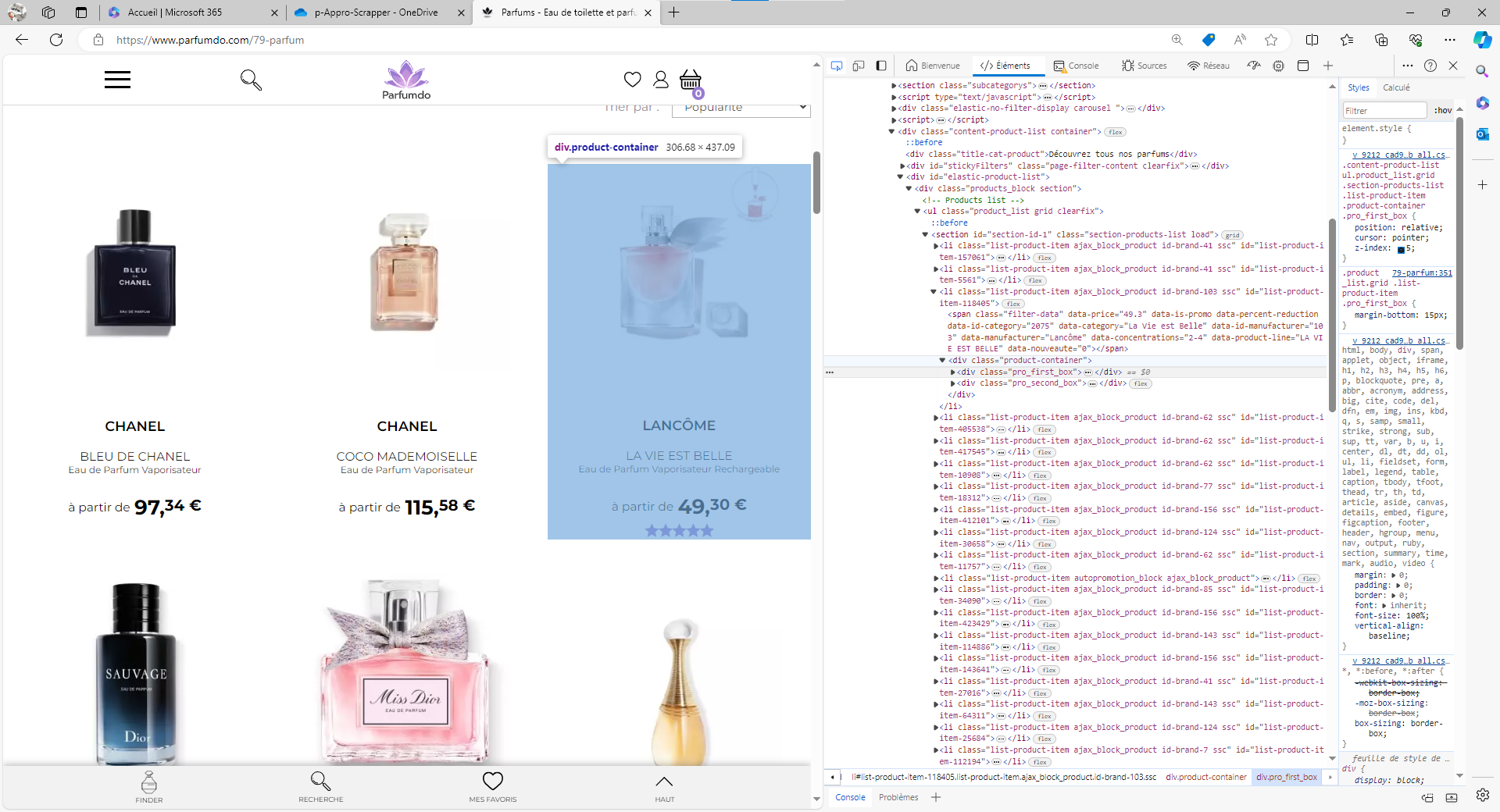
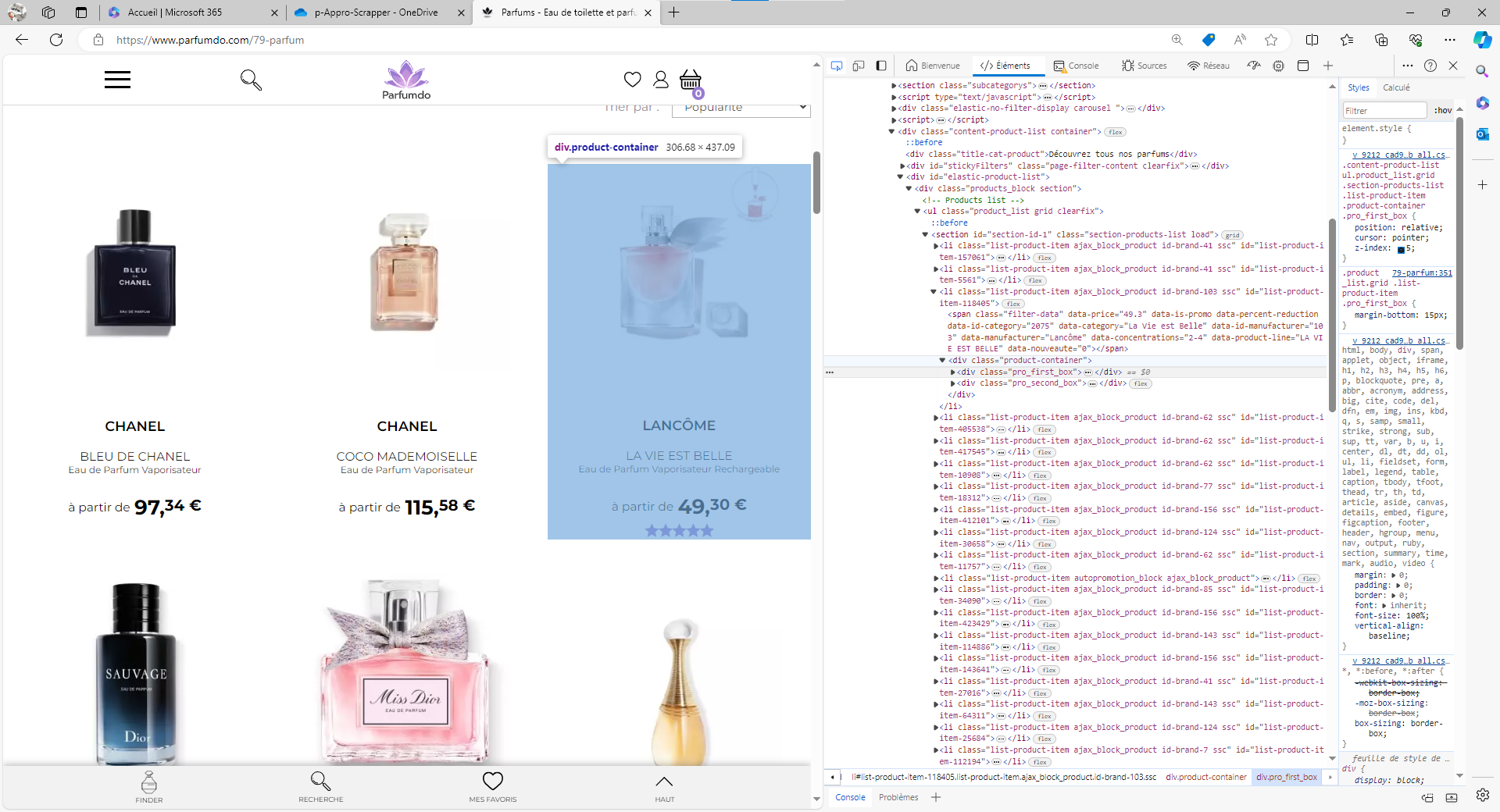
Vous les trouverez aux pages suivantes.

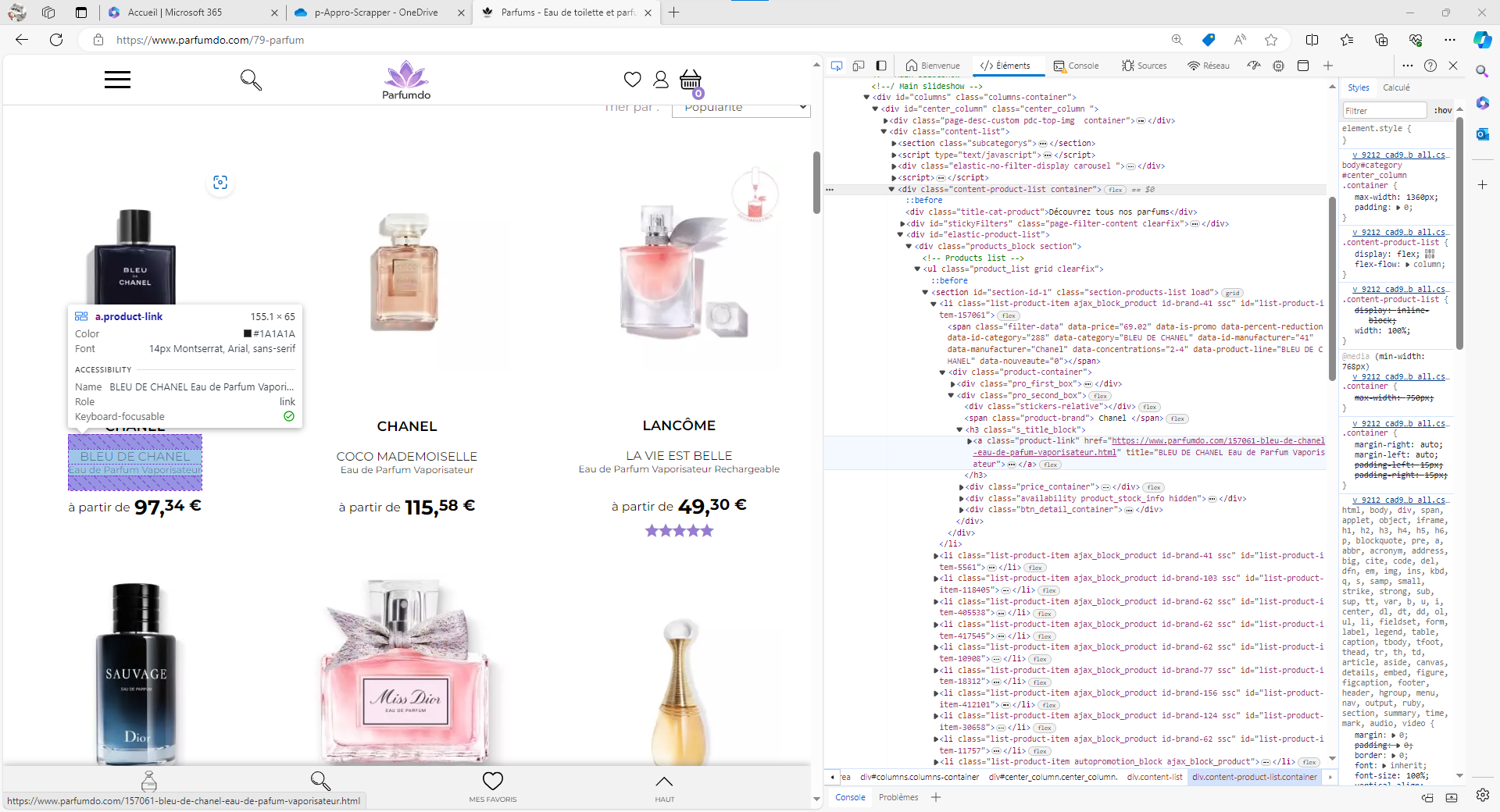
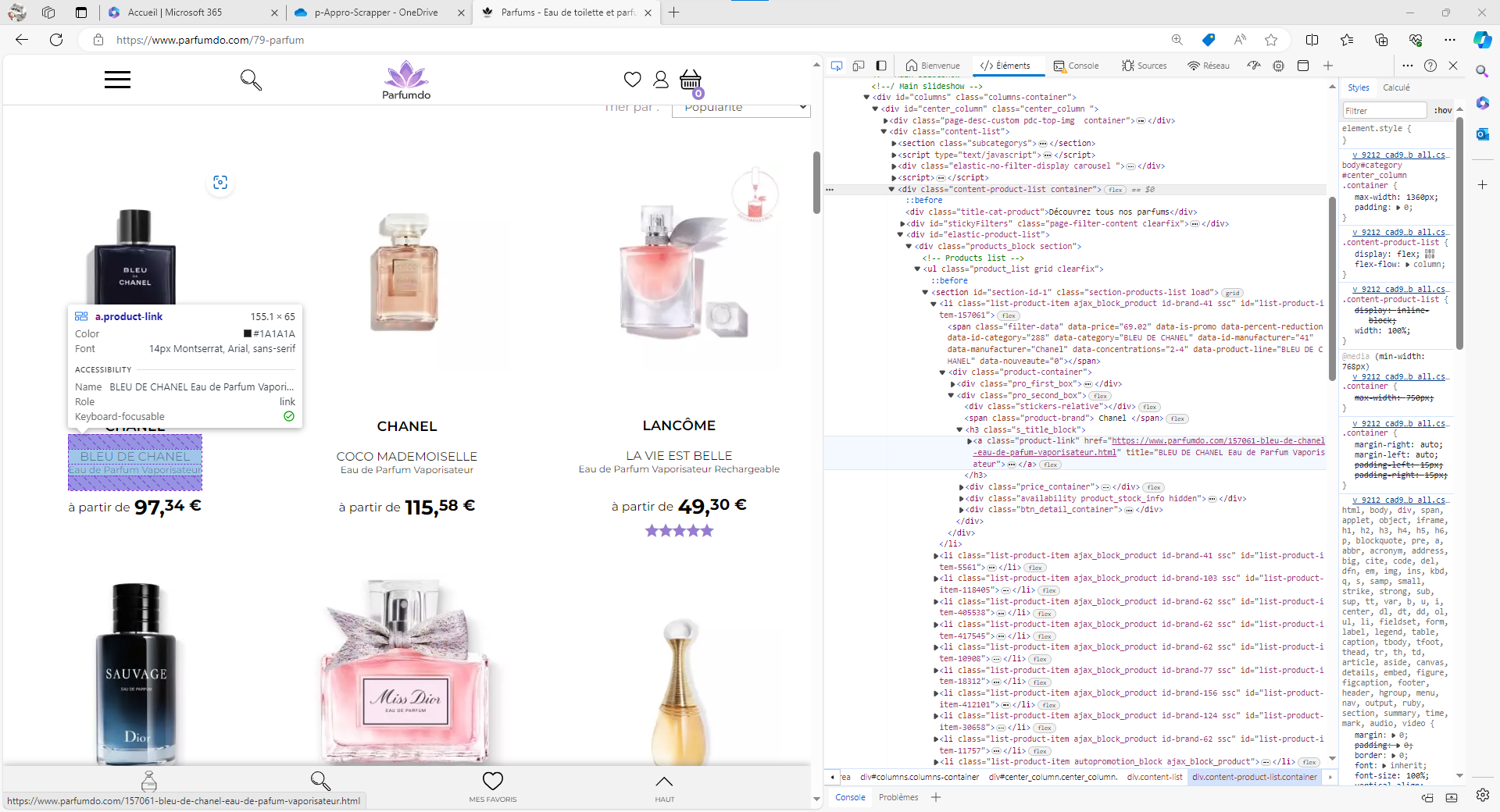
Ici se trouve la div conteneur du tableau HTML contenant les produits à parser, encadré en bleu

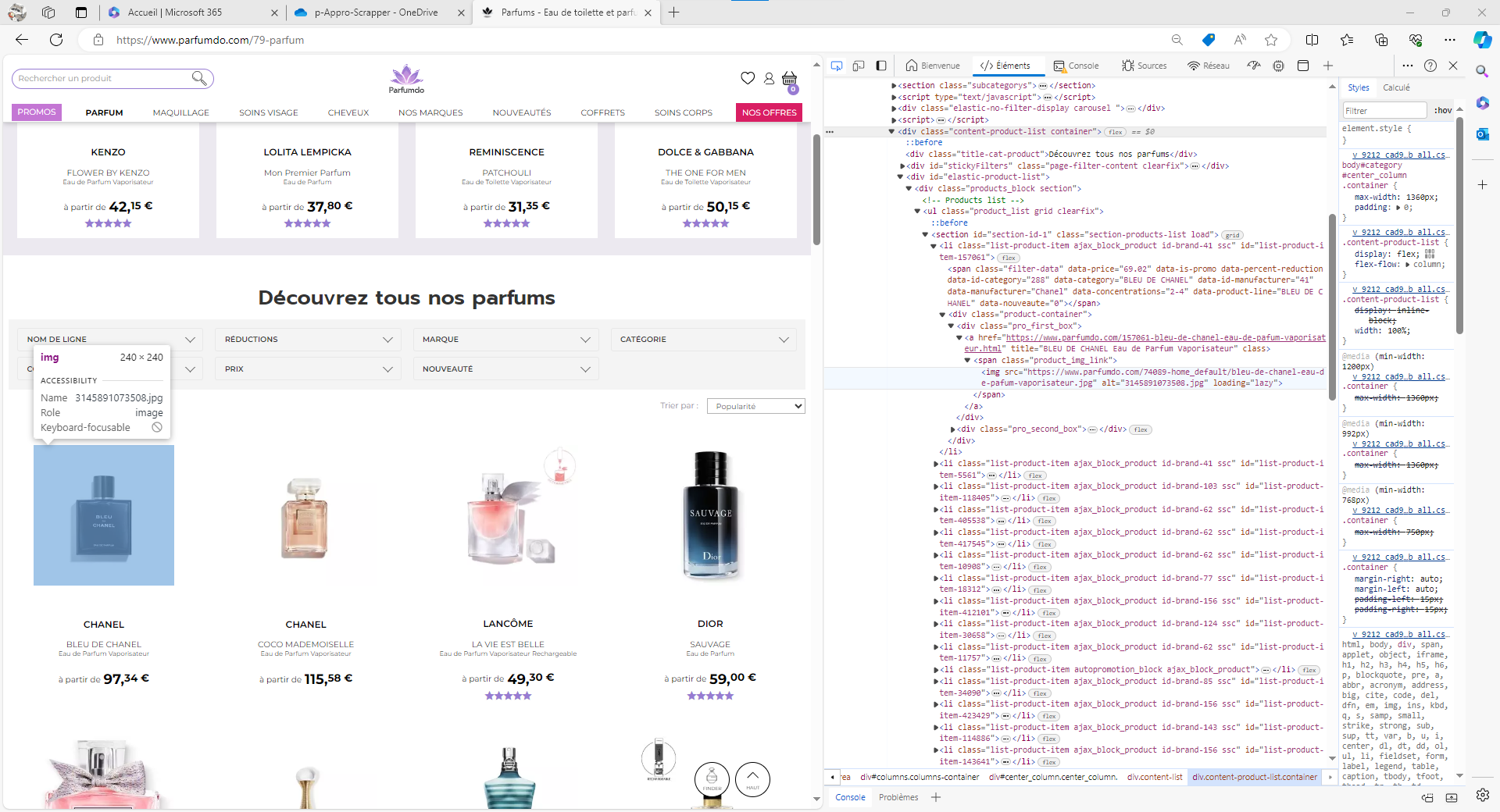


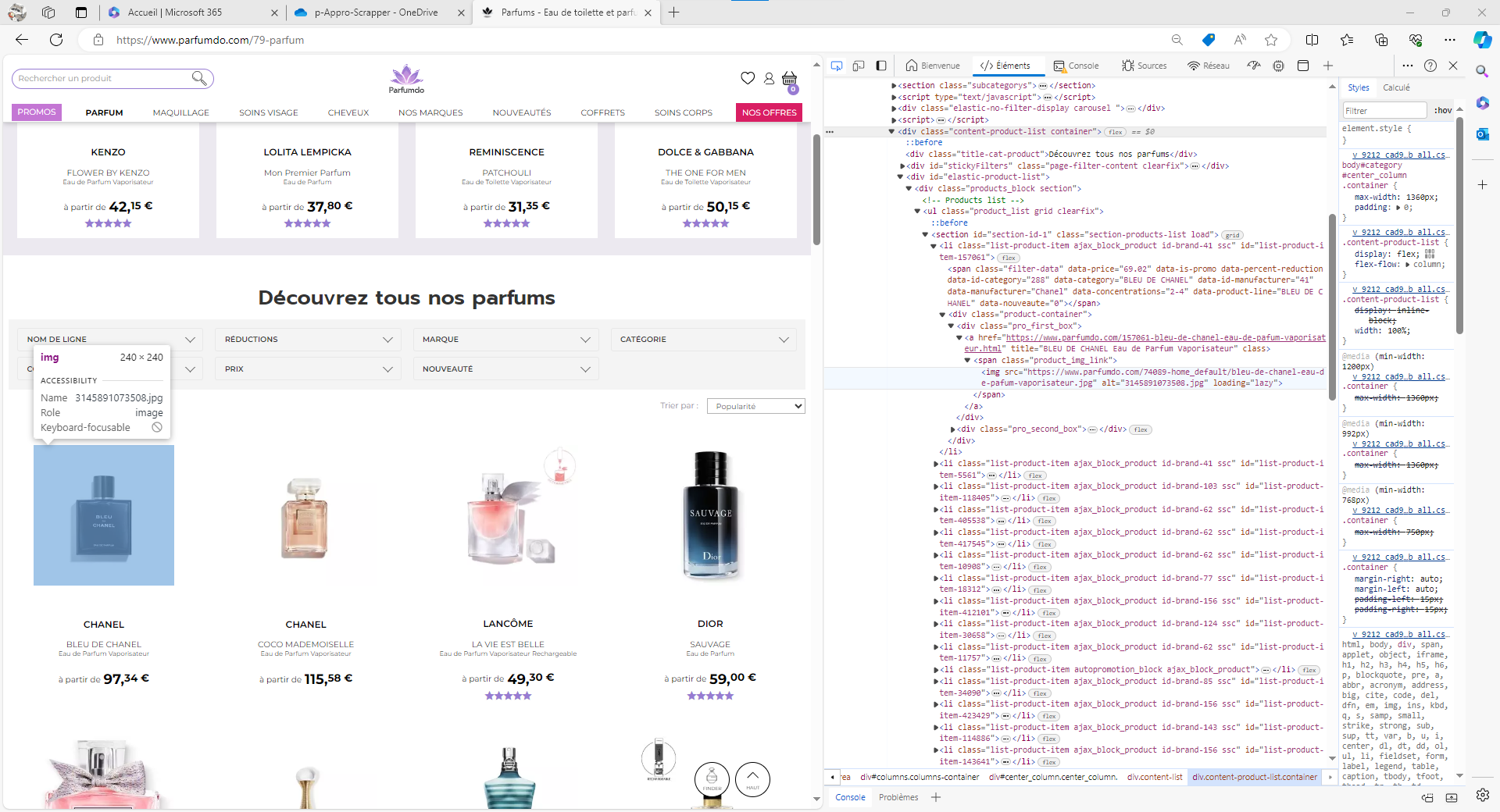


Ici se trouve la div conteneur du tableau HTML contenant les informations du produit à parser, encadré en bleu

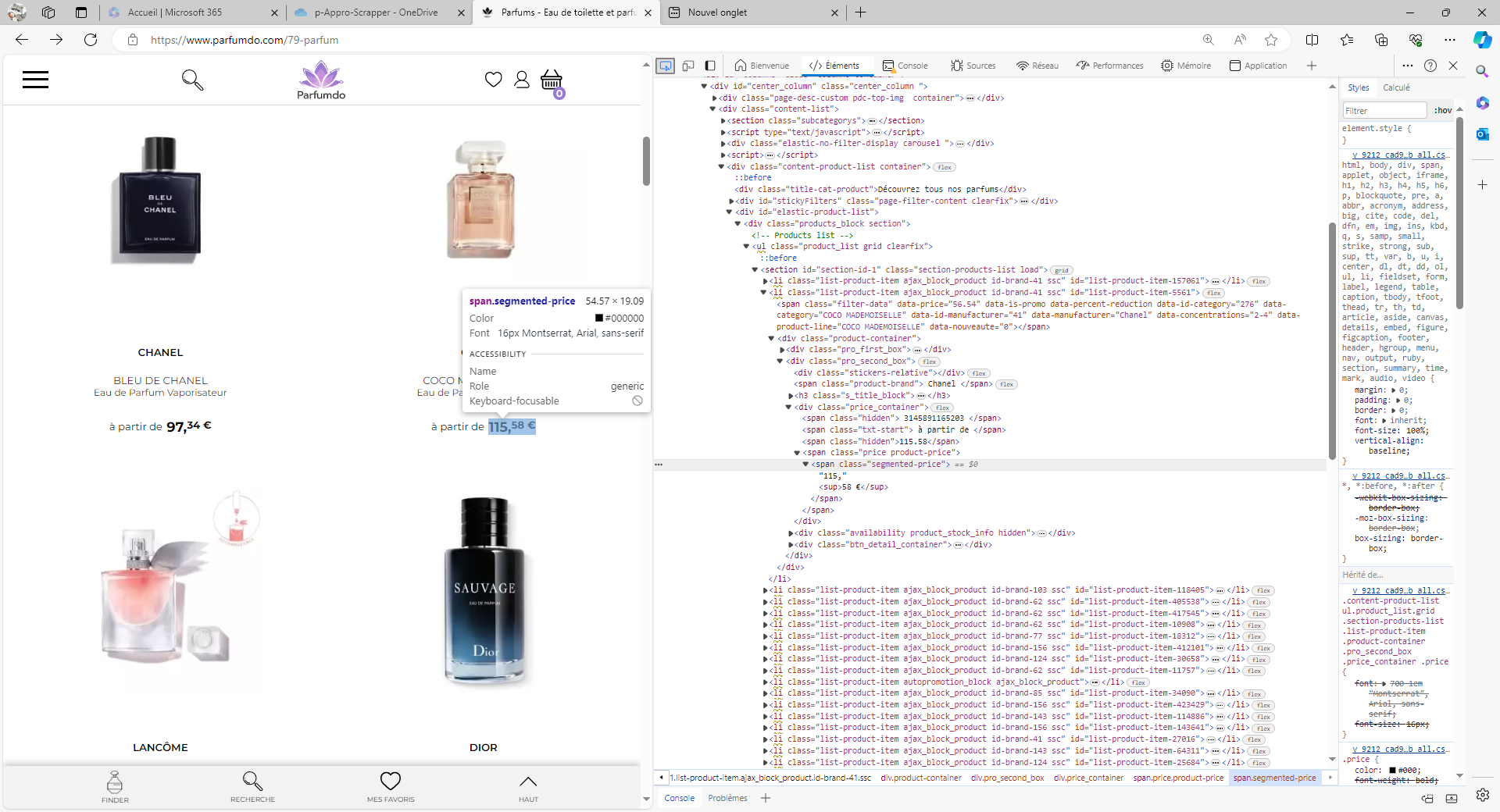


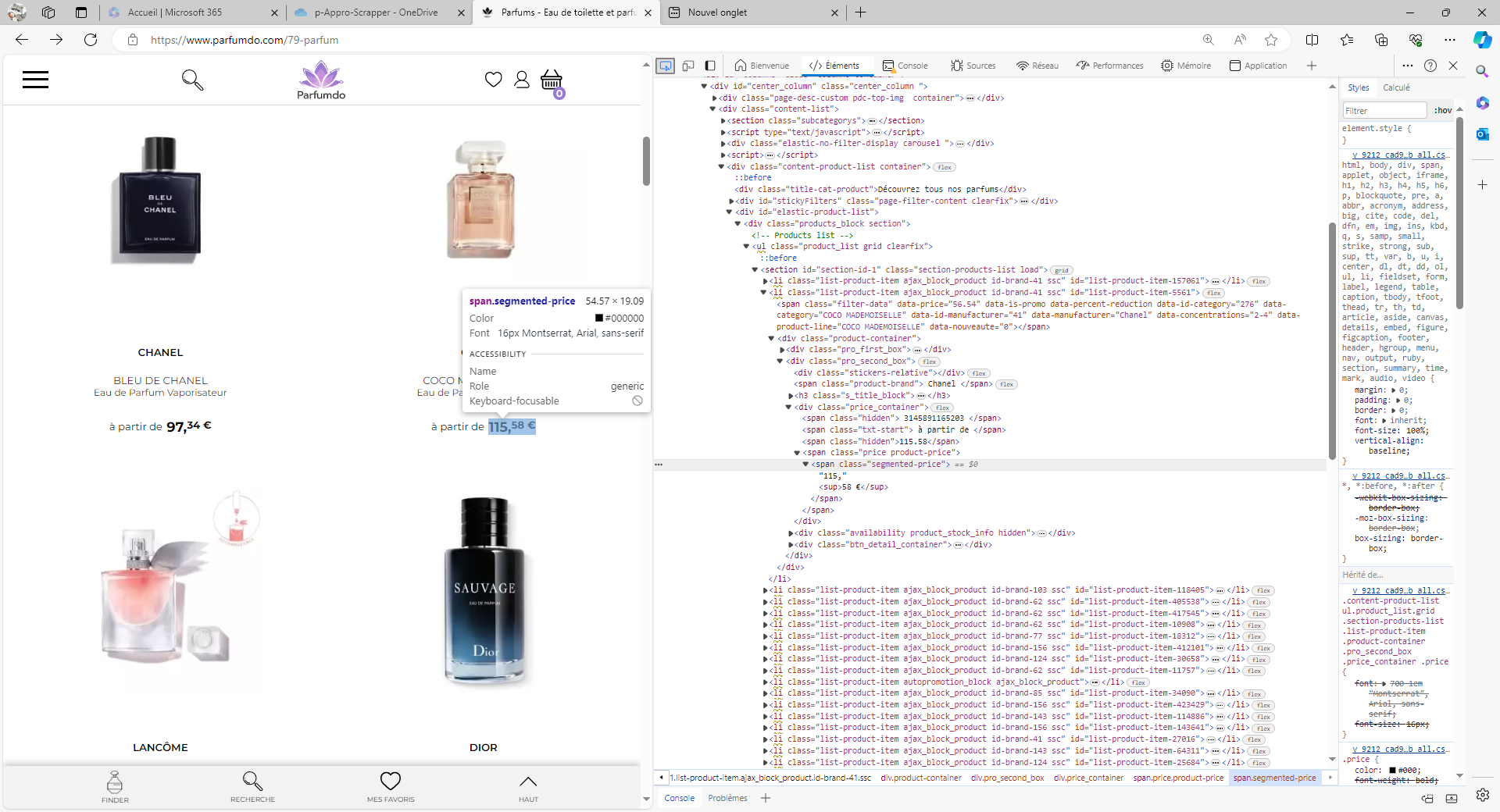
Ici se trouve la balise <a></a> contenant les informations du nom du produit et son lien à récupérer, encadré en bleu

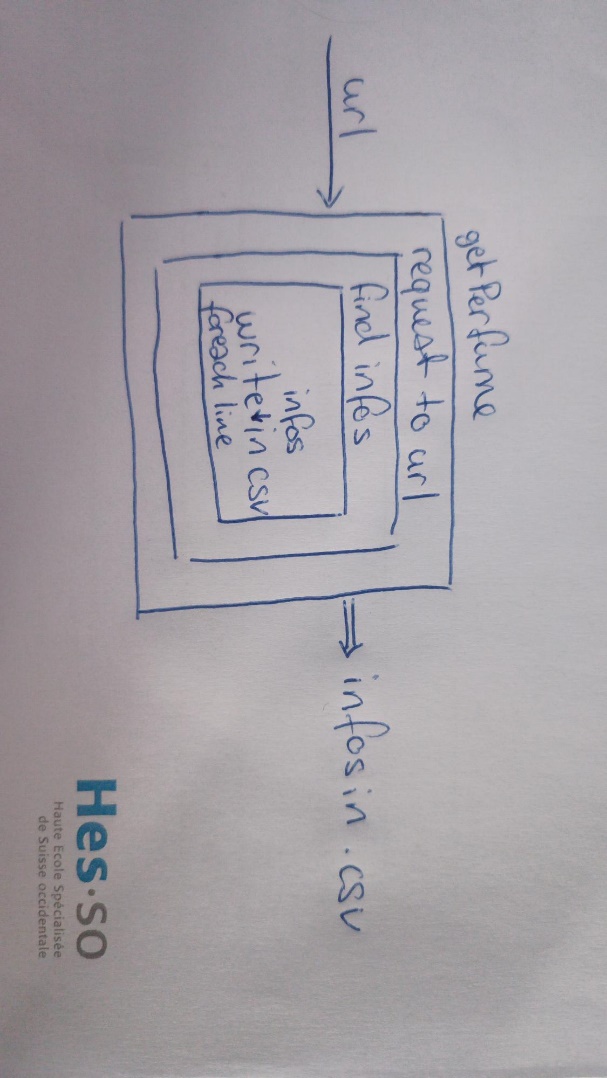
Ici se trouve la balise <img> contenant le lien de l’image du produit à récupérer, encadré en bleu



Ici se trouve la balise <span> contenant le prix du produit à récupérer, encadré en bleu





Ces informations ont donc constitué de base pour construire mon programme, car chaque site web à une structure différente car elle est propre à chaque site web. Par conséquent ce sont des informations cruciales à obtenir afin de réaliser un scraper.

Ci-dessous se trouve mon schéma pour réaliser mon code :

## Conception des tests

Le programme s’exécute sans erreur.

Le programme envoie une requête au site web à scraper.

Le programme retourne un fichier .CSV.

Le programme a rempli le fichier .CSV avec des informations du site web.

Les informations contenues dans le fichier .CSV correspondent aux métadonnées du programme.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

### Réalisation

## Dossier de Réalisation

Comme spécifié dans le cahier des charges, j’ai commencé mon projet en utilisant GitHub comme outil de versionnage, VS Code comme environnement de développement et le langage de programmation Python.

Néanmoins au cours du projet, j’ai rencontré quelques problèmes (voir point 5.2) qui m’ont fait changer d’environnement de développement pour me tourner d’abord vers Replit qui m’a permis de contourner les problèmes précédents tout en sauvant mes modifications faites de manière à ce que je puisse y accéder depuis n’importe quel appareil sur lequel je me serai connectée avec mon compte Replit.

Ne pouvant envoyer de requête au site web cible depuis Replit, je me suis finalement tournée, vers PyCharm.

La version de Python est la suivante 3.6.8 et les librairies utilisées sont :

* BeautifulSoup4
* CSV
* Requests

Mon choix c’est porté sur ces librairies, car la première est une que j’ai déjà survolé comme expliquez au point 1.2 et qu’en cherchant des informations sur des forums et donc internet, ce sont celles qui revenaient le plus souvent pour effectuer les tâches dont j’avais besoin. À savoir donc, parser un site web (BeautifulSoup4), créer et écrire dans un fichier CSV (CSV), accéder au site web en question en envoyant une requête (Requests).

Quant aux infrastructures hardware, mon pc personnel et celui que l’école me met à disposition ont été mes supports principaux.

Afin de réussir à produire ce code, j’ai d’abord commencé par effectué les importations des librairies après avoir installé python sur mon pc.

import csv

import requests

import os

from bs4 import BeautifulSoup

Puis j’ai procédé à la construction de la partie qui envoie une requête au site web, en définissant une nouvelle fonction et enutilisant la fonction « print() », pour voir s’il me retournait bien l’url que j’essayais d’atteindre.

#fonction GET

#param 1 = url de la page

#param 2 = page à scraper

def getPerfume(url, page):

    #variable qui lance une requête au site web avec la page en paramètre

    req = requests.get(url + '?p=' + str(page))

    print(req)

Puis j’ai donné la requête à la fonction de BeautifulSoup pour me parser la page html en la mettant dans une variable pour pouvoir réutiliser le contenu plus tard dans mon code. Toujours en utilisant le print() pour vérifier le contenu de la variable.

#initialisation de la variable soup qui contient la page html

    soup = BeautifulSoup(req.text, "html.parser")

print(soup)

Afin de pouvoir commencer à construire le bloc de recherche d’informations en reprenant les informations trouvées pendant mon analyse de l’HTML du site web au point 3.2, en vérifiant que les balises et les attributs me donnent les informations recherchées en utilisant à nouveau le « print() » pour avoir un visuel de ma fonction.

# boucle qui va chercher chaque élément correspondant aux métadonnées

        for perfume in perfumes:

            name = perfume.find('a', class\_='product-link')['title']

print(name)

Par la suite, j’ai fait une fonction if else récursive qui me permettait d’incrémenter le numéro de la page en l’insérant dans la fonction de requête et de parsage, tant qu’il y avait des produits sur la page, avec l’aide de quelqu’un.

# fonction récursive pour continuer tant qu'il y a des produits

    if(len(perfumes) == 0):

        # si plus de parfum, ne rien faire

        return

    else:

        # appel de la fonction pour passer à la page suivante

        getPerfume(url, page + 1)

Par la suite, j’ai mis un garde-fou inspiré de mon module sur les requête SQL qui permet de ne pas recréer plusieurs fois le même fichier. Car au début des scripts construisant la base de donnée, il y a en général un « IF databaseName EXIST DROP databaseName ».

# si le fichier n'existe pas, création du fichier et rajout des headers

if not os.path.exists('scrappyCSV.csv'):

    #création du fichier

    with open('scrappyCSV.csv', 'w', newline='') as file:

        # initalisation de l'écriture du fichier avec comme délimiteur de colonne, le ;

        writer = csv.writer(file, delimiter=';')

        # définition des métadonnées

        writer.writerow(['Nom', 'Prix', 'Marque', 'Image', 'Lien'])

Pour finir, j’ai appelé la fonction créée comme cela :

#appel de la fonction GET, getPerfume

getPerfume('https://www.parfumdo.com/en/79-perfume', 1)

Cette partie permet de reproduire ou reprendre le projet par un tiers.

Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)

Code source commenté des éléments logiciels développés.

Modèle physique d’une base de données.

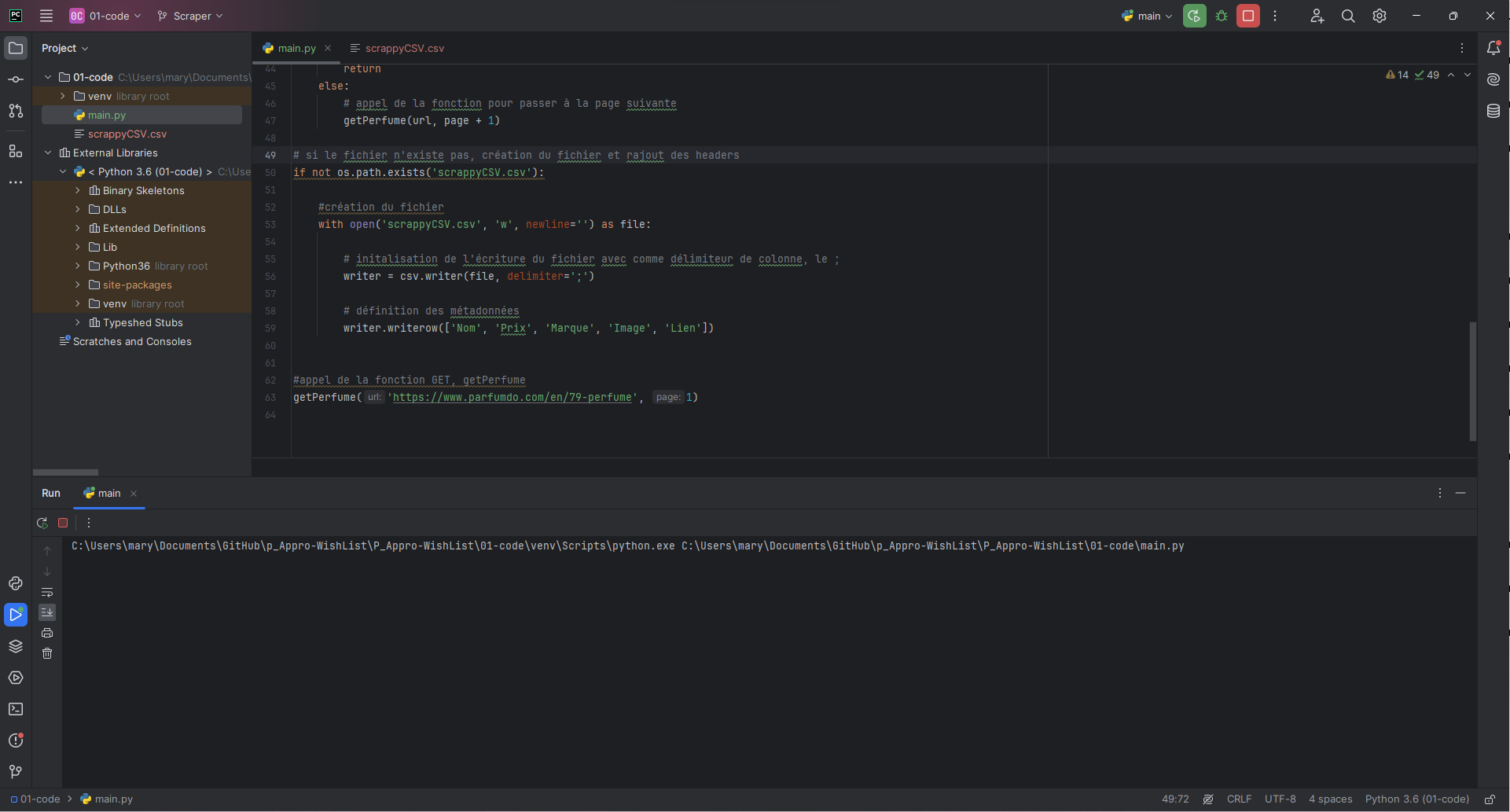
Arborescences des documents produits.

Il faut décrire le parcours de réalisation et justifier les choix.

### Tests

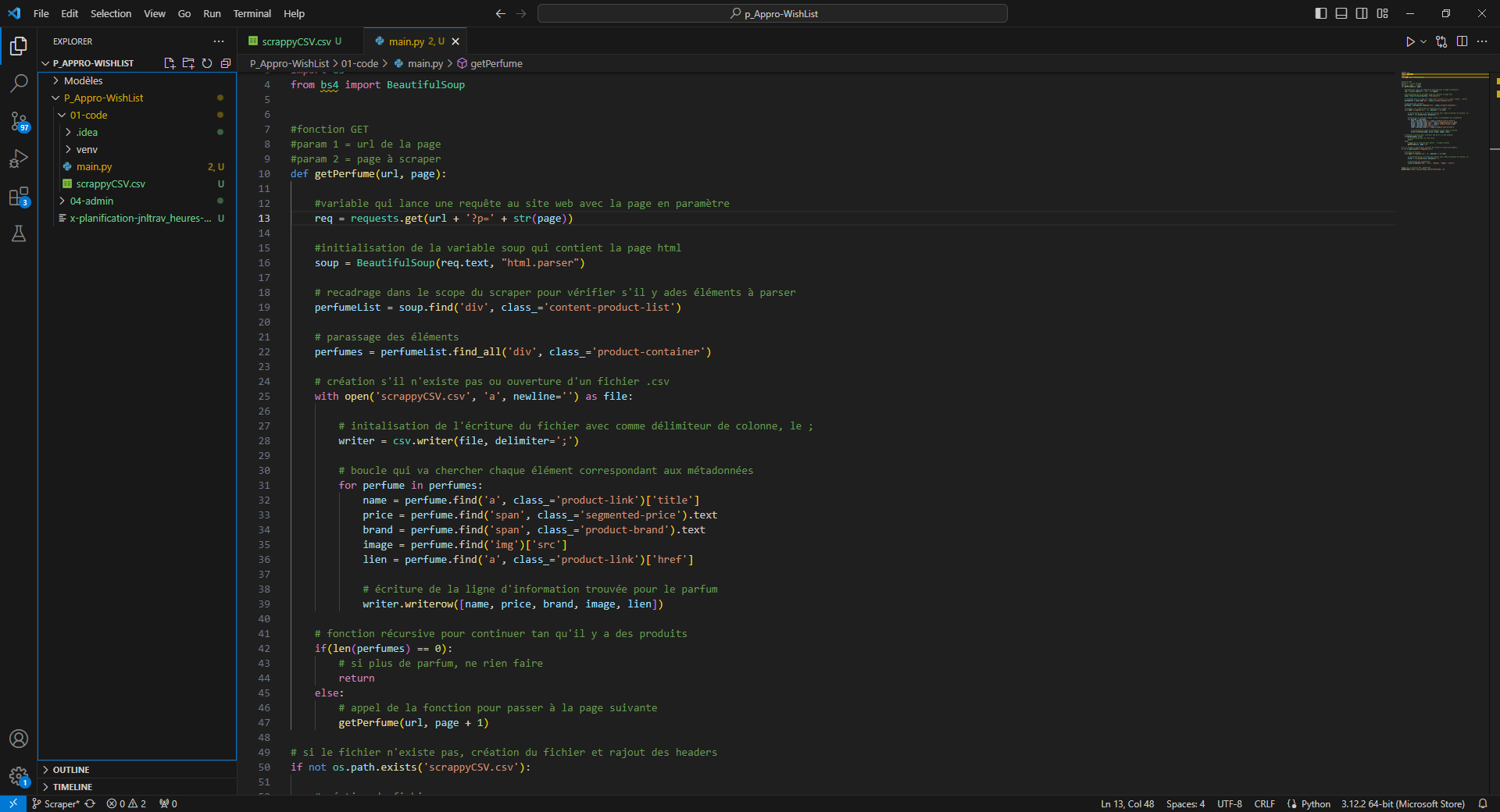
## Dossier des tests

Selon le point 3.3 ci-dessous se trouve les tests, leur réalisation et résultats :

Le programme s’exécute sans erreur.

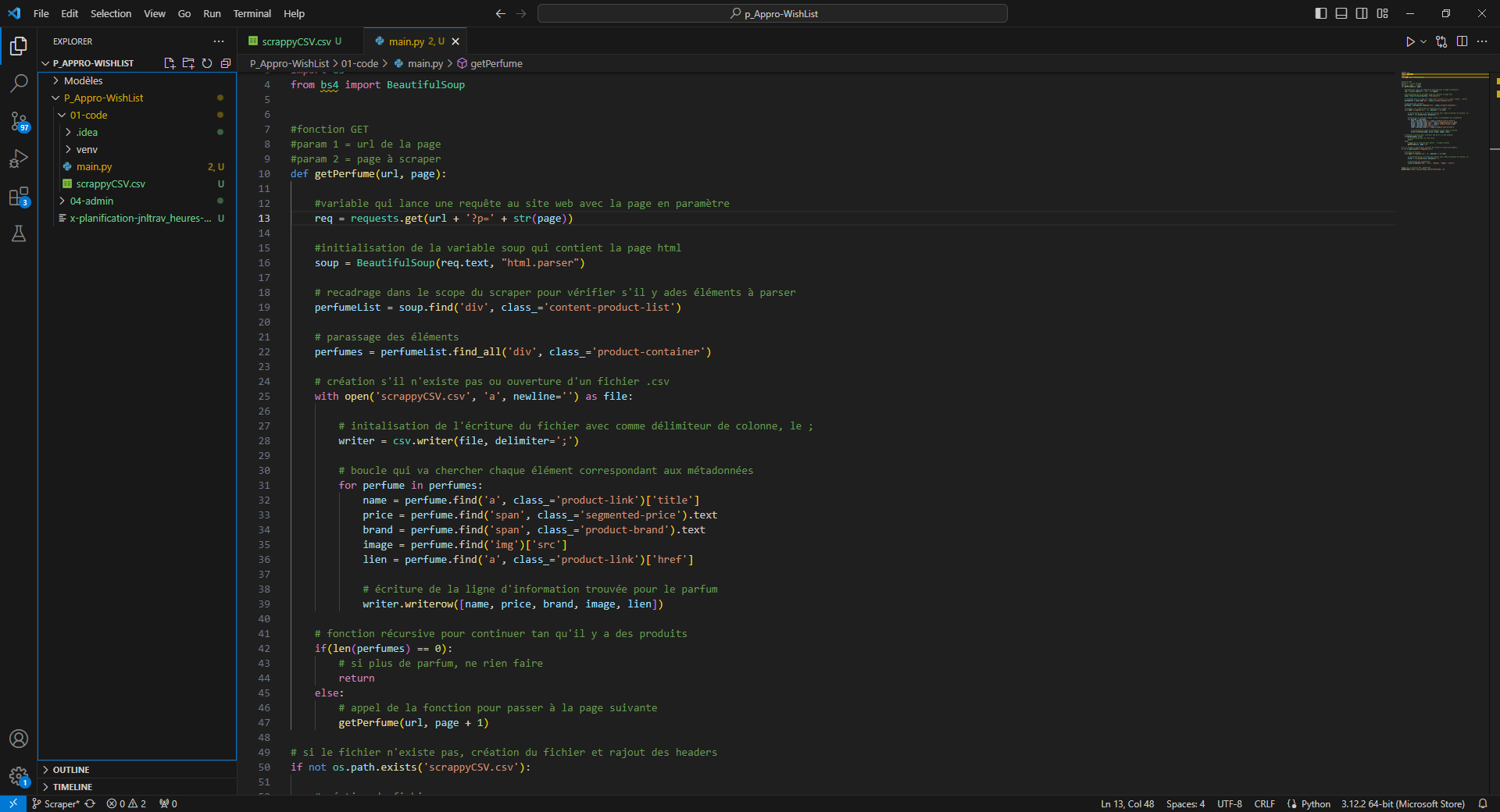
*On regarde si le programme a trouvé une erreur, si les tests suivant de ce chapitre sont réussi.*

Le programme envoie une requête au site web à scraper.

*Dans le code, il y a une variable pour faire une requête en passant toutes les pages existantes contenant des produits.*

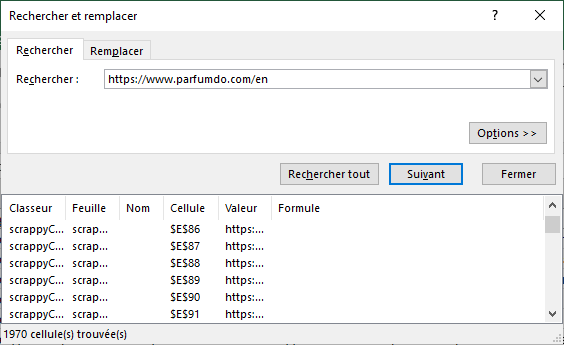
Le programme retourne un fichier .CSV.

*Dans l’explorateur de fichier, l’extension du fichier est bien en .csv*

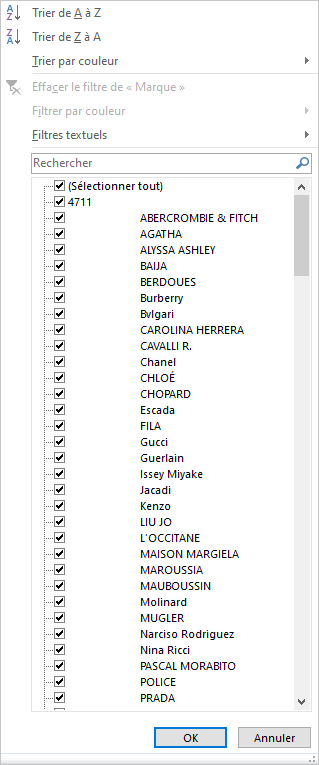


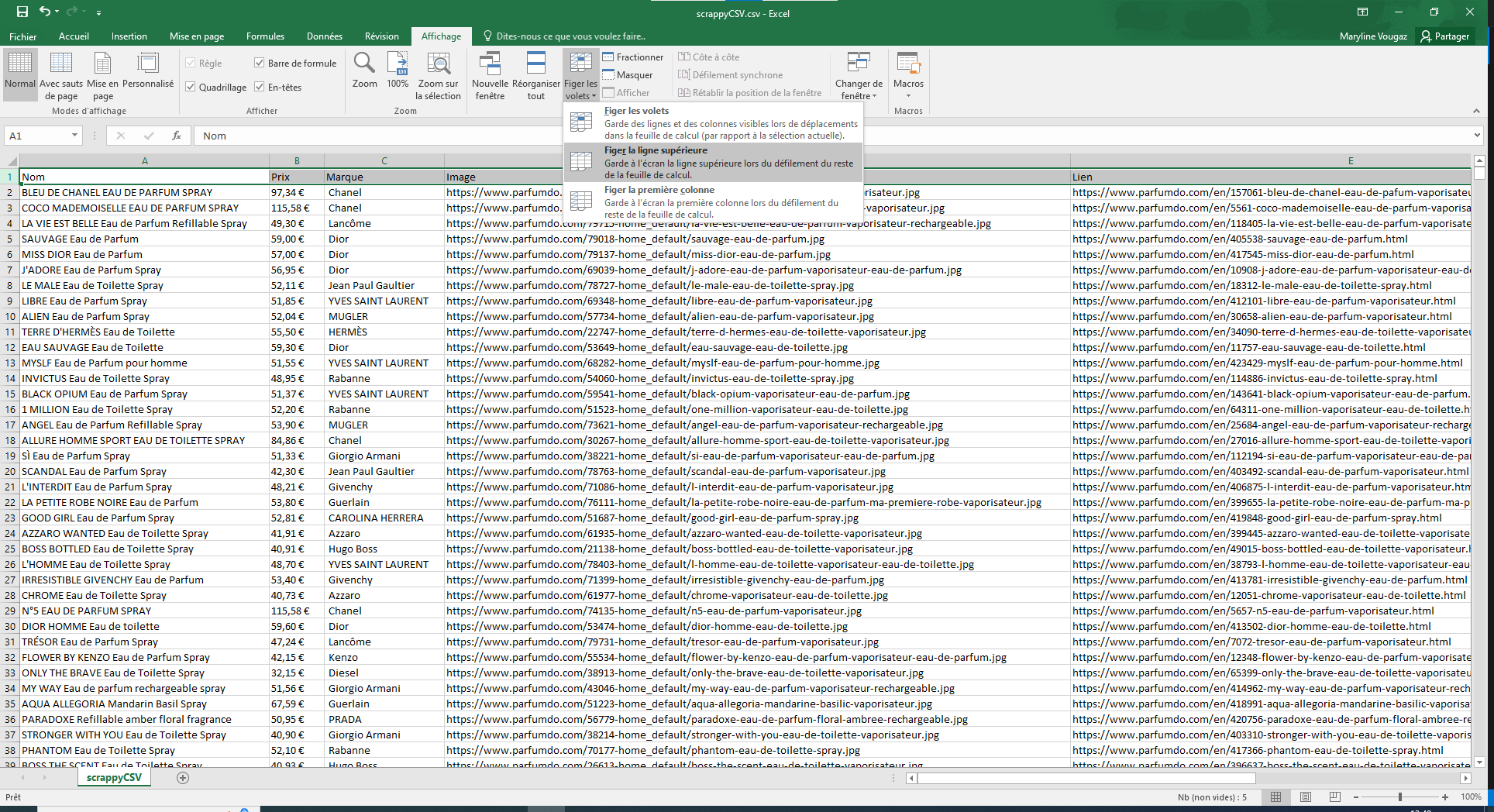
Le programme a rempli le fichier .CSV avec des informations du site web.

*On vérifie que les liens du fichier .CSV soit des liens qui mènent sur le site internet de parfumdo, en faisant un « crtl+f » sur la colonne sensée contenir l’information en entrant cela : « https://www.parfumdo.com/en » dans la barre de recherche.*

**

Les informations contenues dans le fichier .CSV correspondent aux métadonnées du programme.

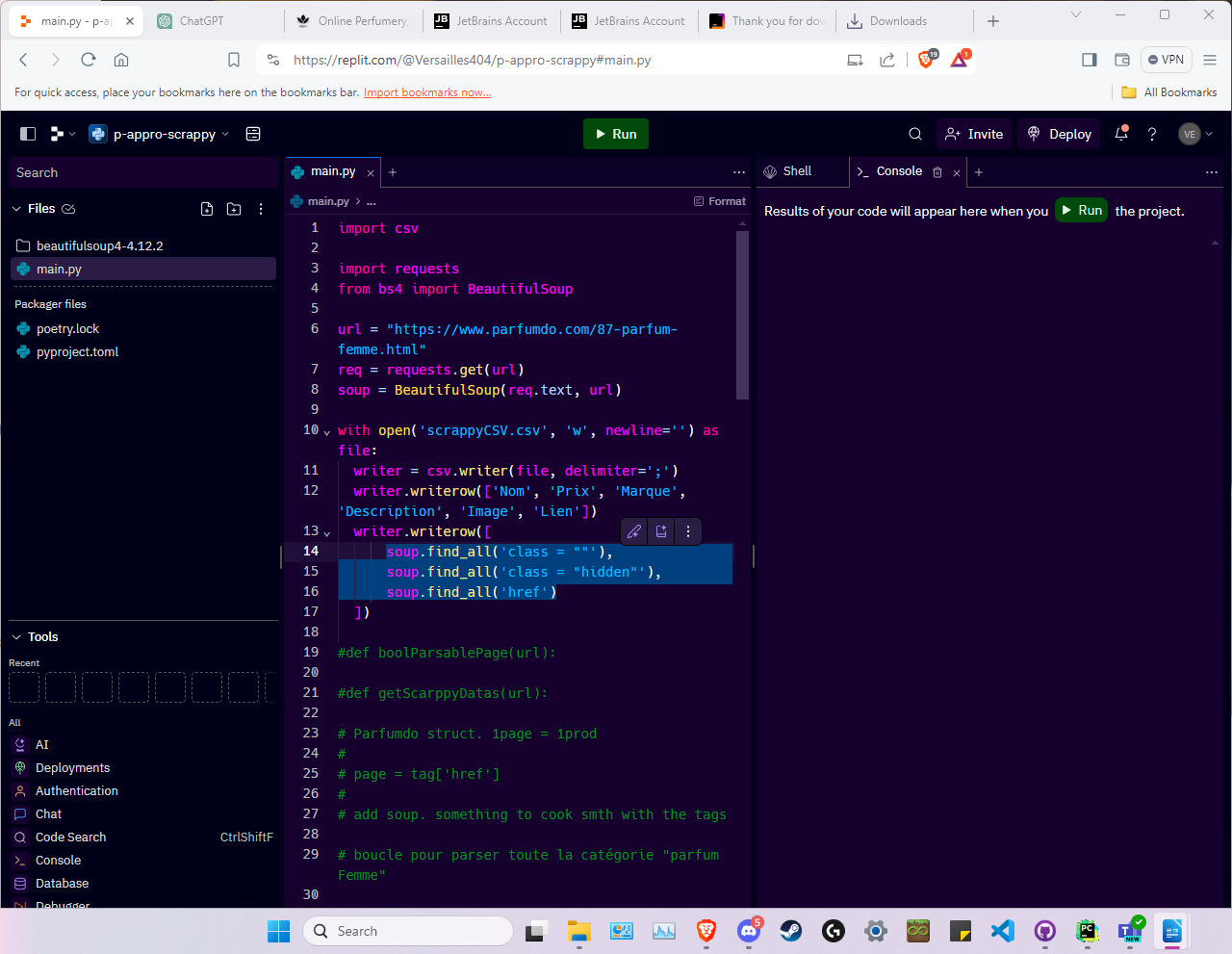
*Afin de vérifier cela, on ouvre le fichier .csv sur Excel en mettant le délimiteur « ; » comme séparateur de colonnes. On ajuste la largeur de colonne afin d’avoir une meilleure lisibilité. On fige la première ligne des métadonnées, afin de pouvoir voir plus rapidement la compatibilité et on compare sur les 38 premières lignes si les informations correspondent. Voir le résultat à la page suivante. Sinon en mettant des filtres sur les métadonnées et en regardant les filtres possibles par header, s’il corresponde comme montré ci-contre, à droite.*

**

## Difficultés & problèmes rencontrés

Initialement, il était prévu que j’utilise VS Code comme environnement de développement. Néanmoins, j’ai eu énormément de problèmes avec npm et pip, qui n’étaient pas détecté et m’empêchaient donc d’installer les libraires que je souhaitais utiliser.

Par conséquent, je me suis tournée vers Replit, car l’accès aux librairies est beaucoup plus simple car avec simplement cela :



Je pouvais importer les librairies nécessaires à mon programme.

Néanmoins, étant un environnement de développement sur un site web, cela empêchait d’envoyer une requête au site web cible, rendant impossible la progression du projet.

Résultant de cela, j’ai décidé d’utiliser PyCharm de JetBrains, car M. Bourahla nous avait déjà parlé de JetBrains et de leurs environnements de développements ainsi que de leur qualité. En utilisant à nouveau mon GitHub afin de faire du versionnage du projet autre que pour les fichiers de rendu administratif.

Un autre problème que j’ai rencontré, est cette fois-ci au niveau du code.

Le problème se situait au niveau de l’incrémentation de la page du site dans l’url en elle-même. Pour être honnête, cela a été je pense la plus grosse embûche car les seules options qui se présentaient à moi n’était pas terrible et provoquaient beaucoup d’erreurs. Dont une boucle infinie car le site web est fait de sorte à ce que si on est par exemple à la page 500, même s’il n’y a pas de produit, elle s’affichera quand même correctement. J’ai donc demandé de l’aide extérieur à un des membres de mon équipe de HackZurich, Alexandre Breuil. Qui m’a donc expliqué comment faire une fonction récursive, car c’était la meilleure solution à mon problème selon lui.

### Conclusion

## Bilan des fonctionnalités demandées

En reprenant point par point, les points du cahier des charges, ci-dessous se trouve le bilan pour chaque fonctionnalité.

* Résultat du scraper retourne un fichier .csv contenant les informations appropriées aux métadonnées

Atteinte à 100%.

* Le scraper a une fonction « GET » qui permet de récupérer le contenu d’un site passé en paramètre.

Atteinte à 100%.

* Le scraper est capable de détecter si la page est parsable, par exemple en comparant le dom et le résultat du scrap

Si des informations sont retournées dans le fichier .csv, sous les métadonnées, alors le site est considéré comme scrapable. Si néanmoins les informations ne correspondent pas aux métadonnées, cela signifie que les attributs ou balises ne sont pas celles contenant les informations recherchées.  
Par conséquent, grâce aux tests effectués, cette fonctionnalité est démontrée au point 5.1 contenant les tests et leur résultat.

Atteinte à 100%.

* Fonction de résumé pour voir combien d’éléments sont des images

Il suffit d’ouvrir le fichier .csv résultant du programme dans Excel et d’utiliser la fonction NBVAL de Excel et donc atteinte, à 75%. Car je suppose que le résultat voulu aurait dû être fournit directement par le programme en lui-même.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

J’ai clairement pris sur mon temps en dehors des cours pour avancer mon projet, car j’ai eu à lire énormément de documentation afin de mieux comprendre certaines utilisations

J’aurais aimé avoir plus de temps afin de pouvoir tout faire en cours. Néanmoins, je suis tout de même contente d’avoir pu utiliser un autre langage de programmation que ceux vu en cours dans le cadre d’un projet. Même si cela voulait dire creuser quelque chose que j’avais déjà fait de manière très sommaire et assistée. Le fait de le réaliser sans toute l’assistance que j’avais eu m’a énormément appris.

Remerciements

À Alexandre Breuil, élève de 4ème année en CFC développement chez Nagravision Sàrl pour son aide sur mon problème de boucle infinie.

### Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Webographie

<https://beautiful-soup-4.readthedocs.io/en/latest/#>

<https://code.tutsplus.com/fr/scraping-webpages-in-python-with-beautiful-soup-the-basics--cms-28211t>

<https://code.tutsplus.com/scraping-webpages-in-python-with-beautiful-soup-search-and-dom-modification--cms-28276t>

<https://www.flaticon.com/free-icon/hot-soup_2388080?term=soup&page=1&position=1&origin=search&related_id=2388080>

<https://www.flaticon.com/free-icon/python_3098090?term=python&page=1&position=4&origin=search&related_id=3098090>

Annexes

En annexe se trouve mon code, le résultat de ce dernier et mon journal de travail