Projet personnel :

Développement d'un « Web Scrapper » Python pour l'extraction de données sur sites web dynamiques

Sommaire :

I. Introduction

Python est un langage de programmation libre et open source, polyvalent et reconnu pour sa syntaxe simple et lisible. Il est utilisé par les développeurs et data scientists pour automatiser des tâches, analyser de grands volumes de données (Big Data), développer des applications web et concevoir des systèmes d'intelligence artificielle. Grâce à son immense bibliothèque de modules, il permet de structurer des projets complexes tout en restant accessible aux débutants.

Le « Web Scraping » est une technique d'automatisation permettant d'extraire des données structurées à partir de sites web. À l'aide de bibliothèques Python (comme BeautifulSoup ou Selenium), on simule la navigation humaine pour récupérer des informations précises (prix, textes, images) afin de les transformer en fichiers exploitables comme du Excel ou du JSON

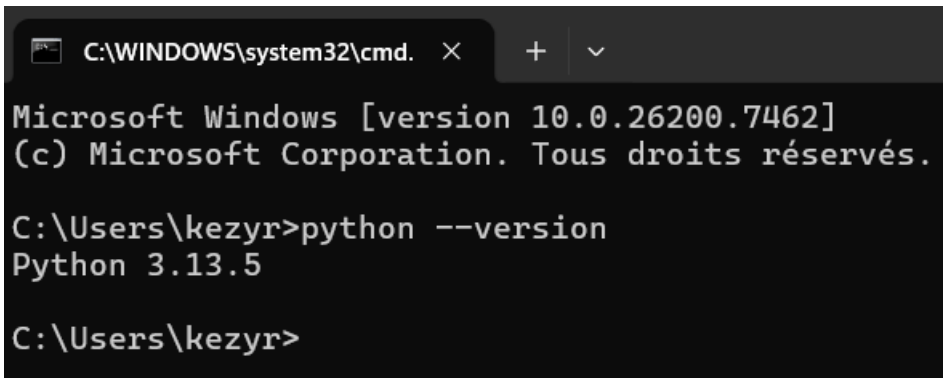
Objectif : Développer un « Web Scrapper » Python à l'aide des librairies appropriés, afin de sélectionner et extraire des produits selon des critères prédéfinis.

II. Préparation de l'environnement

1. **Installation de Python**

   Commencer par installer Python, pour cela il convient de le télécharger depuis sa source officiel : https://www.python.org.

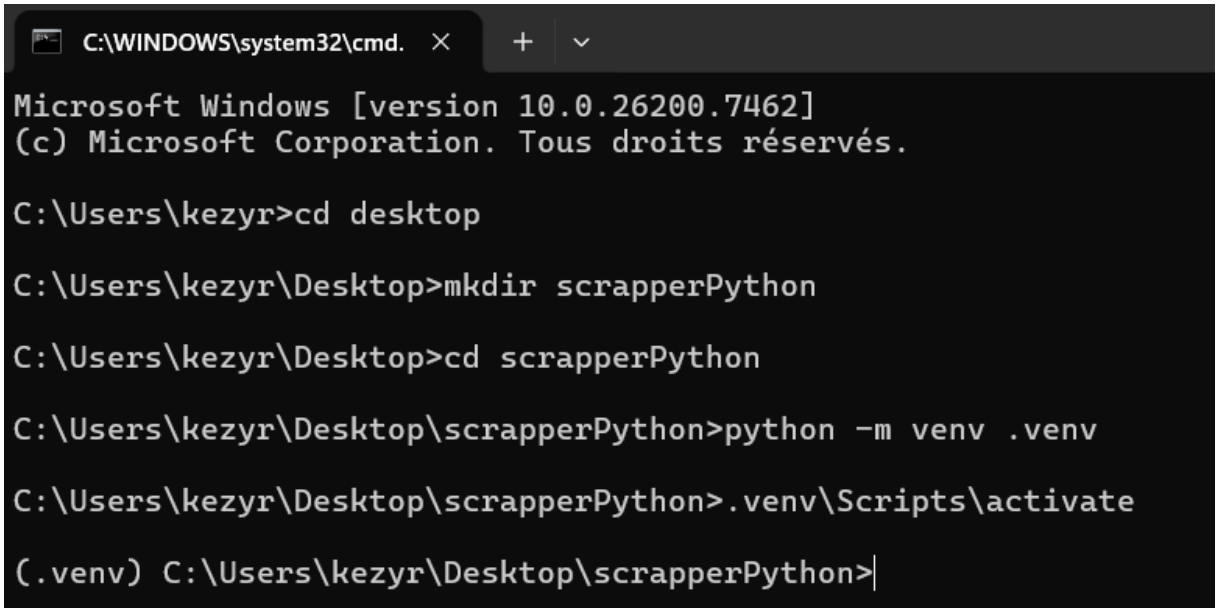
   Une fois l'installation effectuée, vérifier l'installation sur le cmd comme suit :



   Ici, Python a bien été installé avec la bonne version.

2. **Création de l'environnement virtuel**

   Créer un dossier dans lequel mettre le fichier du script ainsi que l'environnement virtuel où il va être exécuté afin de ne pas polluer la machine :



3. Mise à jour des paquets

Mettre à jour les paquets afin que tout soit prêt :

```
C:\Users\kezyr\Desktop\scrapperPython>.venv\Scripts\activate

(.venv) C:\Users\kezyr\Desktop\scrapperPython>python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\kezyr\desktop\scrapperpython\.venv\lib\site-packages (25.1.1)
Collecting pip
  Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 25.1.1
    Uninstalling pip-25.1.1:
      Successfully uninstalled pip-25.1.1
Successfully installed pip-25.3
```

III. Installation de Playwright

3.1 Installation du module Python

Installer la librairie playwright :

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>pip install playwright
Collecting playwright
  Downloading playwright-1.57.0-py3-none-win_amd64.whl.metadata (3.5 kB)
Collecting pyee<14,>=13 (from playwright)
  Using cached pyee-13.0.0-py3-none-any.whl.metadata (2.9 kB)
Collecting greenlet<4.0.0,>=3.1.1 (from playwright)
  Downloading greenlet-3.3.0-cp313-cp313-win_amd64.whl.metadata (4.2 kB)
Collecting typing-extensions (from pyee<14,>=13->playwright)
  Downloading typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Downloading playwright-1.57.0-py3-none-win_amd64.whl (36.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.6/36.6 MB 56.0 MB/s  0:00:00
Downloading greenlet-3.3.0-cp313-cp313-win_amd64.whl (301 kB)
Using cached pyee-13.0.0-py3-none-any.whl (15 kB)
Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Installing collected packages: typing-extensions, greenlet, pyee, playwright
Successfully installed greenlet-3.3.0 playwright-1.57.0 pyee-13.0.0 typing-extensions-4.15.0

(.venv) C:\Users\kezyr\Desktop\scrapperPython>
```

Vérifier que c'est bien installé :

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>pip show playwright
Name: playwright
Version: 1.57.0
Summary: A high-level API to automate web browsers
Home-page: https://github.com/Microsoft/playwright-python
Author: Microsoft Corporation
Author-email:
License-Expression: Apache-2.0
Location: C:\Users\kezyr\Desktop\scrapperPython\.venv\Lib\site-packages
Requires: greenlet, pyee
Required-by:
```

Ici tout est bien en ordre.

3.2 Installation des navigateurs

Installer les navigateurs dont Playwright a besoin pour fonctionner, à savoir Chromium, Firefox et Webkit :

```
C:\WINDOWS\system32\cmd.
(.venv) C:\Users\kezyr\Desktop\scrapperPython>playwright install
```

```
Downloading Chromium 143.0.7499.4 (playwright build v1200) from https://cdn.playwright.dev/dbazure/download/playwright/b
uilds/chromium/1200/chromium-win64.zip
(node:64576) [DEP0169] DeprecationWarning: `url.parse()` behavior is not standardized and prone to errors that have secu
rity implications. Use the WHATWG URL API instead. CVEs are not issued for `url.parse()` vulnerabilities.
(Use `node --trace-deprecation ...` to show where the warning was created)
169.8 MiB [====================] 100% 0.0s
Chromium 143.0.7499.4 (playwright build v1200) downloaded to C:\Users\kezyr\AppData\Local\ms-playwright\chromium-1200
Downloading Chromium Headless Shell 143.0.7499.4 (playwright build v1200) from https://cdn.playwright.dev/dbazure/downlo
ad/playwright/builds/chromium/1200/chromium-headless-shell-win64.zip
(node:63372) [DEP0169] DeprecationWarning: `url.parse()` behavior is not standardized and prone to errors that have secu
rity implications. Use the WHATWG URL API instead. CVEs are not issued for `url.parse()` vulnerabilities.
(Use `node --trace-deprecation ...` to show where the warning was created)
107.2 MiB [====================] 100% 0.0s
Chromium Headless Shell 143.0.7499.4 (playwright build v1200) downloaded to C:\Users\kezyr\AppData\Local\ms-playwright\c
hromium_headless_shell-1200
Downloading Firefox 144.0.2 (playwright build v1497) from https://cdn.playwright.dev/dbazure/download/playwright/builds/
firefox/1497/firefox-win64.zip
(node:12840) [DEP0169] DeprecationWarning: `url.parse()` behavior is not standardized and prone to errors that have secu
rity implications. Use the WHATWG URL API instead. CVEs are not issued for `url.parse()` vulnerabilities.
(Use `node --trace-deprecation ...` to show where the warning was created)
107.1 MiB [====================] 100% 0.0s
Firefox 144.0.2 (playwright build v1497) downloaded to C:\Users\kezyr\AppData\Local\ms-playwright\firefox-1497
Downloading Webkit 26.0 (playwright build v2227) from https://cdn.playwright.dev/dbazure/download/playwright/builds/webk
it/2227/webkit-win64.zip
(node:60580) [DEP0169] DeprecationWarning: `url.parse()` behavior is not standardized and prone to errors that have secu
rity implications. Use the WHATWG URL API instead. CVEs are not issued for `url.parse()` vulnerabilities.
(Use `node --trace-deprecation ...` to show where the warning was created)
58.2 MiB [====================] 100% 0.0s
Webkit 26.0 (playwright build v2227) downloaded to C:\Users\kezyr\AppData\Local\ms-playwright\webkit-2227
```

3.3 Vérification

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>playwright --version
Version 1.57.0
```

3.4 Installation de la librairie Beautifulsoup4

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>pip install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.14.3-py3-none-any.whl.metadata (3.8 kB)
Collecting soupsieve>=1.6.1 (from beautifulsoup4)
  Downloading soupsieve-2.8.1-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\kezyr\desktop\scrapperpython\.venv\lib\site-packages
 (from beautifulsoup4) (4.15.0)
Downloading beautifulsoup4-4.14.3-py3-none-any.whl (107 kB)
Downloading soupsieve-2.8.1-py3-none-any.whl (36 kB)
Installing collected packages: soupsieve, beautifulsoup4
Successfully installed beautifulsoup4-4.14.3 soupsieve-2.8.1
```

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>pip show beautifulsoup4
Name: beautifulsoup4
Version: 4.14.3
Summary: Screen-scraping library
Home-page: https://www.crummy.com/software/BeautifulSoup/bs4/
Author:
Author-email: Leonard Richardson <leonardr@segfault.org>
License: MIT License
Location: C:\Users\kezyr\Desktop\scrapperPython\.venv\Lib\site-packages
Requires: soupsieve, typing-extensions
Required-by:
```

IV. Développement du script

Importer les modules nécessaires :

```
1   from playwright.sync_api import sync_playwright
2   from bs4 import BeautifulSoup
3   import time
4   import re
```

Définir le tableau pour stocker les résultats :

```
6   resultats = []
```

Identifier au préalable les moyens de naviguer entre les pages, et appliquer :

```
8   def get_next_page_url(soup):
9       suivant = soup.find("a", string=lambda t: t and "Suivant" in t)
10      return suivant["href"] if suivant else None
```

Définir ensuite comment lancer la page, avec le navigateur de votre choix :

```
12  with sync_playwright() as p:
13      browser = p.chromium.launch(headless=False)
14      page = browser.new_page()
```

A ce moment récupérer l'url du site afin de naviguer dessus, ici un site de rachat de catalyseurs :

```
16      url = "https://leboncata.fr/constructeur/9a88661f-caae-4dca-a676-b96a50b38903/renault"
17      while url:
18          page.goto(url)
```

Mettre un temps d'attente et un scroll afin de charger les pages en entiers :

```
21              page.mouse.wheel(0, 3000)
22              page.wait_for_timeout(2000)
```

Ensuite « parser » le html récupéré grâce à BeautifulSoup4 :

```
25              html = page.content()
26              soup = BeautifulSoup(html, "html.parser")
```

Identifier les produits, leur référence et leur prix :

```
28          produits = soup.select("a[href*='/catalyseur/']")
29          for produit in produits:
30              ref_bloc = produit.select_one("div.text-xl.font-semibold")
31              ref = ref_bloc.get_text(strip=True) if ref_bloc else None
32
33              prix_bloc = produit.select_one("div.text-base.font-extrabold")
34              prix_txt = prix_bloc.get_text(strip=True) if prix_bloc else None
```

Ensuite les filtrer selon des critères prédéfinis, ici ne sont retenus que les catalyseurs valant plus de 200€ à la revente :

```
37              if prix_txt:
38                  prix_num = float(prix_txt.replace("€", "").replace(",", ".").replace("\u202f", "").strip())
39                  if prix_num >= 200:
40                      resultats.append({"référence":ref, "prix":prix_num})
```

Chercher le lien vers la page suivante :

```
43              next_url = get_next_page_url(soup)
44              url = next_url if next_url else None
```

Enfin fermer le navigateur et afficher le résultat :

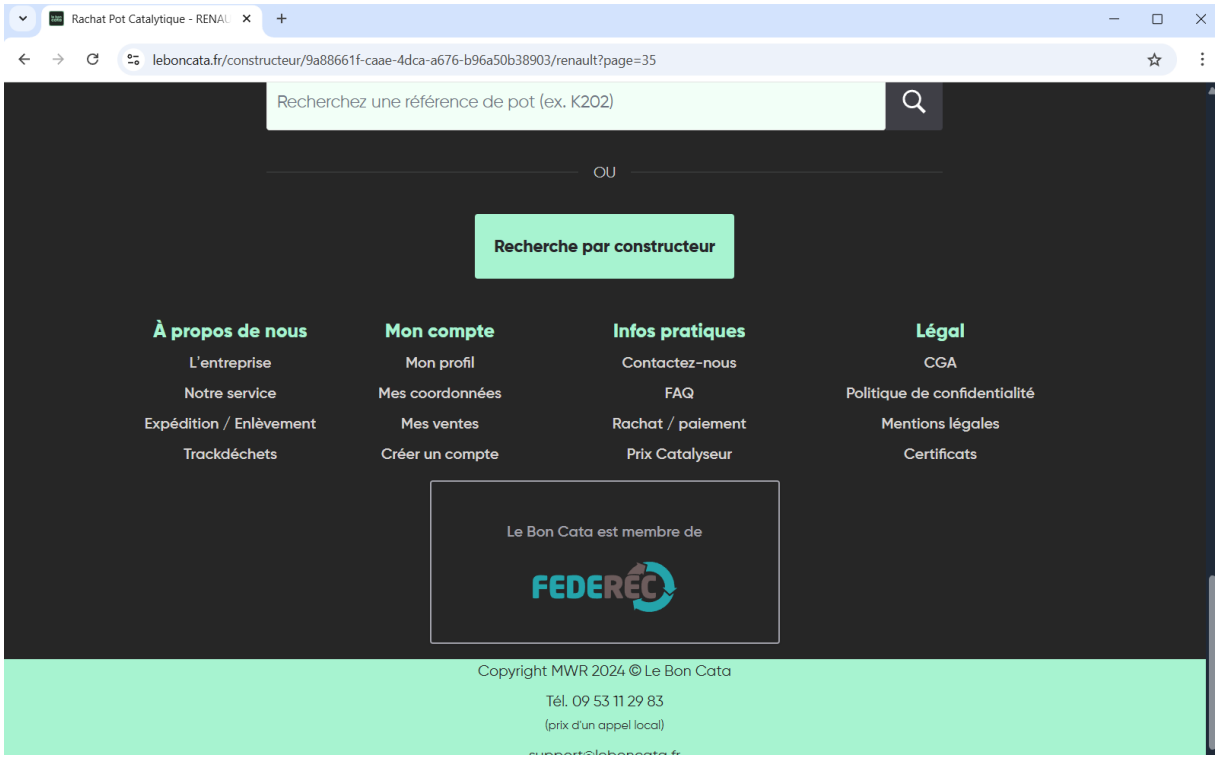
```
46      browser.close()
47
48  for produit in resultats:
49      print(produit)
```

V. Exécution du Web Scrapper

Lancement du script :

```
(.venv) C:\Users\kezyr\Desktop\scrapperPython>python dynamique.py
```

La fenêtre du navigateur Chromium s'ouvre et le script agit comme si un utilisateur lambda navigue dessus :



Une fois toutes les pages passées en revue, le script s’arrête et le résultat est récupéré :

```
{'référence': '8200756673 H8200787005', 'prix': 255.96}
{'référence': 'H8200959456 8200959427', 'prix': 212.74}
{'référence': '8200390853 8200397416', 'prix': 211.07}
{'référence': '8200787005 8200756673C', 'prix': 255.96}
{'référence': '8200506654 H8200389549', 'prix': 248.0}
{'référence': '8200526838 H8200540278', 'prix': 236.63}
{'référence': '8200578825 H8200540278', 'prix': 245.68}
{'référence': '8200693046 H8200693048', 'prix': 335.75}
{'référence': '8200956313', 'prix': 203.81}
{'référence': 'C328 4408 208020013R', 'prix': 202.24}
{'référence': 'H8220540278 8200578825', 'prix': 244.22}
{'référence': '8200747077 8200389549', 'prix': 248.0}
{'référence': 'C496 208029814R', 'prix': 200.33}
{'référence': 'C498 208027269R', 'prix': 204.92}
{'référence': 'C616 GM93455570 208026742R', 'prix': 276.76}
{'référence': 'C622 208027825R', 'prix': 251.51}
{'référence': 'C497 208026513R', 'prix': 226.19}
{'référence': '208A04643R', 'prix': 345.58}
{'référence': '208A02747R HMLGT6978R', 'prix': 253.96}
{'référence': 'C414', 'prix': 228.7}
{'référence': 'C505', 'prix': 215.63}
{'référence': 'C617', 'prix': 295.47}
{'référence': 'C628', 'prix': 343.69}
{'référence': 'C638', 'prix': 322.12}
{'référence': 'C656', 'prix': 371.38}
{'référence': 'C80', 'prix': 230.5}
{'référence': 'H8201599992 208A04010R', 'prix': 214.29}

(.venv) C:\Users\kezyr\Desktop\scrapperPython>
```

VI. Conclusion

Ce tutoriel montre comment mettre en place un environnement propre, écrire un script fonctionnel et comprendre les bases de l'automatisation web. Il constitue une excellente base pour développer des scrapers plus avancés, ajouter de la gestion d'erreurs, exporter les données ou intégrer ces scripts dans des projets plus larges.