Introduction au Web Scraping avec Python et Beautiful Soup

Objectifs

Bienvenue dans cette aventure où vous allez découvrir le web scraping ! Ce cours vous guide, pas à pas, dans l'univers de l'extraction de données sur le web. Vous apprendrez à :

- Comprendre ce qu'est le web scraping et pourquoi il est utile
- Explorer la structure d'une page web et son langage (**HTML**)
- Créer votre propre page **HTML** pour bien comprendre son fonctionnement
- Naviguer dans la structure d'une page web existante pour récupérer des informations
- Utiliser **Python** et **BeautifulSoup** pour automatiser l'extraction de données
- Respecter les aspects légaux et éthiques du scraping

Pré-requis

- Programmation en Python

1. Introduction au Web Scraping

1.1 Qu'est-ce que le Web Scraping ?

Imaginez que vous êtes un explorateur à la recherche d'un trésor caché dans une jungle numérique : Internet. Le web scraping est votre carte et votre boussole, qui vous permettent d'extraire automatiquement des informations utiles des pages web.

Mais pourquoi voudrions-nous extraire des données ? Voici quelques exemples :

- Un journaliste veut récupérer les derniers titres d'actualités d'un site d'information.
- Un analyste surveille les prix des produits sur plusieurs sites e-commerce pour comparer les tendances.
- Une entreprise suit les avis de ses clients sur divers forums et sites d'évaluation.

Plutôt que de copier les informations à la main, nous allons utiliser un programme qui fait le travail pour nous, et ce, bien plus rapidement !

1.2 Pourquoi le Web Scraping peut poser des questions légales ?

Pour faire du web scraping, il y a des règles à respecter ! Le web scraping peut poser des problèmes si vous récupérez des informations sans respecter les conditions d'utilisation du site.

Voici les bonnes pratiques légales :

- **Consulter le fichier `robots.txt`** d'un site (situé à `https://<nomdusite.com>/robots.txt`). Ce fichier indique quelles parties du site sont autorisées ou interdites au scraping.
- **Ne pas surcharger un site** en envoyant trop de requêtes à la fois (cela peut le ralentir ou le bloquer).
- **Ne pas collecter de données personnelles** sans consentement (ex. : adresses e-mail, noms de personnes).
- **Vérifier les conditions d'utilisation du site** pour voir si le scraping est autorisé.

En respectant ces règles, vous pouvez scraper en toute tranquillité et rester dans la légalité !

2. Comprendre le HTML : Le Langage des Pages Web

Avant de pouvoir extraire des informations, nous devons comprendre comment elles sont organisées dans une page web.

Sources:

- https://developer.mozilla.org/fr/docs/Learn_web_development/Getting_started/Your_first_website/Creating_the_content
- https://developer.mozilla.org/en-US/docs/Web/HTML/Element

2.1 Les bases du HTML

Le HTML (*HyperText Markup Language*) est le langage utilisé pour structurer une page web. Il fonctionne avec des **balises**, qui délimitent différents éléments de la page.

Une balise HTML s'écrit sous la forme :

```
<balise> Contenu </balise>
```

Par exemple :

```
<p>Ceci est un paragraphe.</p>
```

Chaque balise HTML a :

- **Une ouverture** `<nom_de_balise>`
- **Un contenu** (texte, image, lien, etc.)
- **Une fermeture** `</nom_de_balise>`

Certaines balises comme `<img>` (image) et `<br>` (saut de ligne) n'ont pas besoin de fermeture.

2.2 Création d'une page HTML

Créez un fichier `ma_page.html` et collez-y le code suivant :

```
<!DOCTYPE html>
<html>
<head>
    <title>Ma Première Page</title>
</head>
<body>
    <h1>Bienvenue sur ma page</h1>
    <p>Ceci est un paragraphe d'exemple.</p>
    <a href="https://example.com">Visitez ce site</a>
    <ul>
        <li>Élément 1</li>
        <li>Élément 2</li>
        <li>Élément 3</li>
    </ul>
</body>
</html>
```

Explication du code :

1. `<!DOCTYPE html>` : Indique que le document est une page web HTML.

2. `<html>` : Balise racine qui englobe tout le contenu.
3. `<head>` : Contient les informations sur la page (titre, styles, etc.).
4. `<title>` : Définit le titre de l'onglet du navigateur.
5. `<body>` : Contient le contenu visible de la page.
6. `<h1>` : Un gros titre.
7. `<p>` : Un paragraphe de texte.
8. `<a href="">` : Un lien vers un autre site.
9. `<ul>` et `<li>` : Une liste non ordonnée avec des éléments.

Ouvrez ce fichier dans un navigateur et observez le rendu. Vous venez de créer votre première page Web.

2.3 Les classes et identifiants en HTML

En HTML, les éléments peuvent être différenciés et stylisés grâce aux classes (**class**) et aux identifiants (**id**).

- Une classe (**class**) permet d'appliquer le même style à plusieurs éléments.
- Un identifiant (**id**) est unique et permet d'identifier un seul élément spécifique.

Exemple :

```
<p class="texte-important">Ceci est un texte important.</p
<p id="unique-paragraphe">Ceci est un paragraphe unique.</
```

3. Explorer une Page Web et Extraire des Informations

1. Allez sur https://books.toscrape.com/.
2. Faites **cliquer droit > Inspecter l'élément** (ou appuyez sur F12).
3. Survolez les titres des livres et observez leur structure HTML.
4. Identifiez les balises qui contiennent les noms, les images, ainsi que les prix des livres.
5. Remarquez que tous les livres appartiennent à une même **class**. Identifiez-la.

4. Mini-Projet : Extraction des Titres et Prix des Livres

Dans cette section, nous allons extraire les titres des livres et leurs prix affichés sur la première page de *Books to Scrape*.

4.1 Installation des outils:

Créer un environnement virtuel dans lequel vous installez

- beautifulsoup
- requests

4.2 Récupération de la page HTML avec Python

Avec la bibliothèque “requests” vous pouvez récupérer le contenu html d’une page.

```
import requests

url = "https://books.toscrape.com/"
response = requests.get(url)
```

Observez la réponse obtenue. Lorsque vous regardez le résultat de la page HTML, cela ressemble à un énorme désordre. Le package “Beautifulsoup” vous permet d’analyser ou d’extraire des éléments souhaités en utilisant des balises html.

4.3 Extraction des données avec BeautifulSoup

Beautifulsoup est une bibliothèque Python très utilisée en web scraping car elle facilite la navigation et la recherche d’éléments spécifiques dans une page.

Sources:

- https://realpython.com/beautiful-soup-web-scraper-python/#step-3-parse-html-code-with-beautiful-soup
- https://beautiful-soup-4.readthedocs.io/en/latest/

Question:

Utilisez requests et Beautifulsoup pour extraire **uniquement les livres les mieux notés** (>4 étoiles) et **moins chers que 30£ présentés sur les 3 premières pages.** Enregistrez le résultat dans un fichier **csv**

Question BONUS:

Créez un classement des 10 catégories ayant les livres les plus chers en moyenne.