

# Introduction

Dans le métier de data scientist, il est important de pouvoir se constituer des jeux de données si ceux-ci n’existent pas. Une façon de faire est d’utiliser internet et de récupérer les données sur des sites intéressants.

Le Web Scraping permet d’automatiser ce genre de procédure et de créer des routines récupérant et structurant les données au format défini par le data analist.

# Objectifs

A l’issue de cette journée , vous serez capable de :

* Scraper les données d’un site à l’aide de python avec les packages suivants :
  + Scrapy
  + Beautifoulsoup
  + Selenium

# Pré-requis

* Programmation en Python
* Ligne de commande linux

# Compétences à acquérir

* Récupérer et structurer des données venant d’un site web.

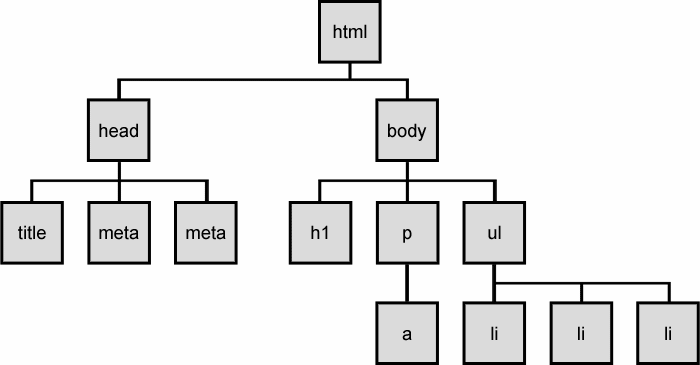
# Démarche pédagogique

Avec les packages scrapy, beautifulsoup et selenium les apprenants vont accéder aux données non structurées. Dans un premier temps, ils vont récupérer et structurer des données sur les livres référencés sur un site web. Dans un second temps, ils vont collecter des données au sujet de déchets avec beautifulsoup et selenium. Selenium permet de faire la pagination d’un url avec plusieurs pages.

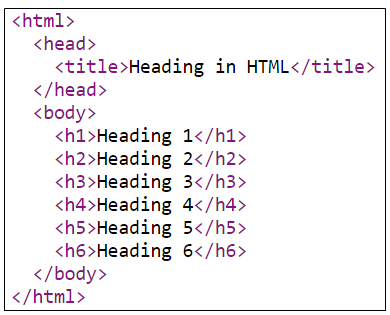
# Une introduction rapide au HTML

Pour scraper des données il est nécessaire de comprendre les bases du langage web HTML.

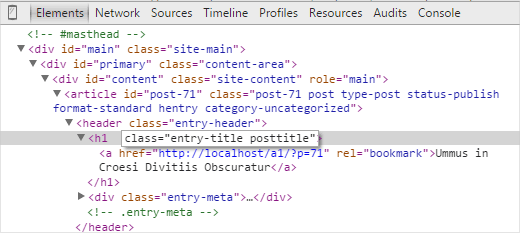
Le langage HTML décrit le contenu d’une page web, sous forme d’une arborescence, pour cela il utilise des balises qui englobent tout ou partie du contenu afin de structurer la position, le formatage et plein d’autres paramètres concernant les éléments situés à l'intérieur.



L’exemple ci-dessous correspond à une page web très basique. La page est englobée entre les balise <html> (balise ouvrante) et </html> (balise fermante). La partie contenue entre les balises <head> et </head> contient essentiellement des métadonnées, pour ce mini-projet nous nous intéresserons plus au contenu de la page contenu entre les balises <body> et </body>.



Lors de l'inspection d’une page web avec votre navigateur web, on retrouve les balises vues précédemment ainsi que des attributs (class, href, id,...) qui sont généralement utilisés pour définir plus précisément le style, l’usage ou la position du contenu de la balise.



*(Regardez comment utiliser l’inspecteur de code de votre navigateur, cela sera probablement utile)*

Maintenant que la structure d’un site est introduite, nous allons utiliser python pour récupérer des informations.

**Comment trouver les tag html? Sur n’importe quelle page, cliquez droit et choisissez “inspect”. Cela ouvre une écran avec le code html de votre page.**

# Partie 1: Scrapy

### Objectifs

* Créer un premier projet de scraping en se familiarisant avec scrapy
* Utiliser le scraping pour collecter et structurer les données d’un site

Scrapy utilise ce que l’on appelle des ‘spider’, chaque ‘spider’ a pour but d’aller récupérer et structurer les données d’un site.

Avant de commencer à scraper:

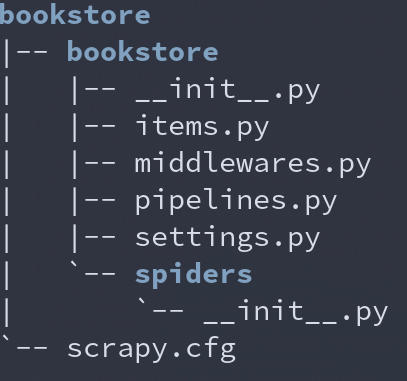
* Installer la version 2.5.1 de scrapy

La gestion d’un projet scrapy peut se faire assez facilement en ligne de commande, avant de créer notre premier ‘spider’, il est nécessaire de créer un nouveau projet scrapy que l’on nommera “bookstore” avec la commande suivante:

scrapy startproject bookstore

* Créer le projet

Cette commande génère la configuration du projet de scraping, on obtient ensuite l’arborescence suivante:



Pour nous familiariser avec l’outil nous allons créer un premier ‘spider’ pour récupérer les noms des livres de la page 1.

La commande scrapy genspider nom\_spider nom\_domaine permet d’obtenir un template à compléter pour créer un spider.

Nous allons dans un premier temps récupérer les titres des livres présents sur le site.

* Créer un premier ‘spider’ nommé “book\_names”

Avant de créer la routine pour extraire les noms de la page, il est nécessaire de compléter le fichiers items.py en créant une nouvelle classe qui va contenir un attribut nommé name. Cette structure de données va "accueillir" les données scrapées!

Pour extraire les données on utilise une notation nommée XPATH qui permet de naviguer dans des documents comme l’HTML. (pour en savoir plus voir [ici](https://www.w3schools.com/xml/xml_xpath.asp))

* Compléter le code du fichier bookstore/spiders/book\_names.py
  + Créez une fonction start\_requests permettant de faire une requête sur la page principale
  + Complétez la fonction parse afin d’utiliser la réponse du serveur pour en extraire les titres des livres.
  + Stockez les données dans la classe créer dans items.py

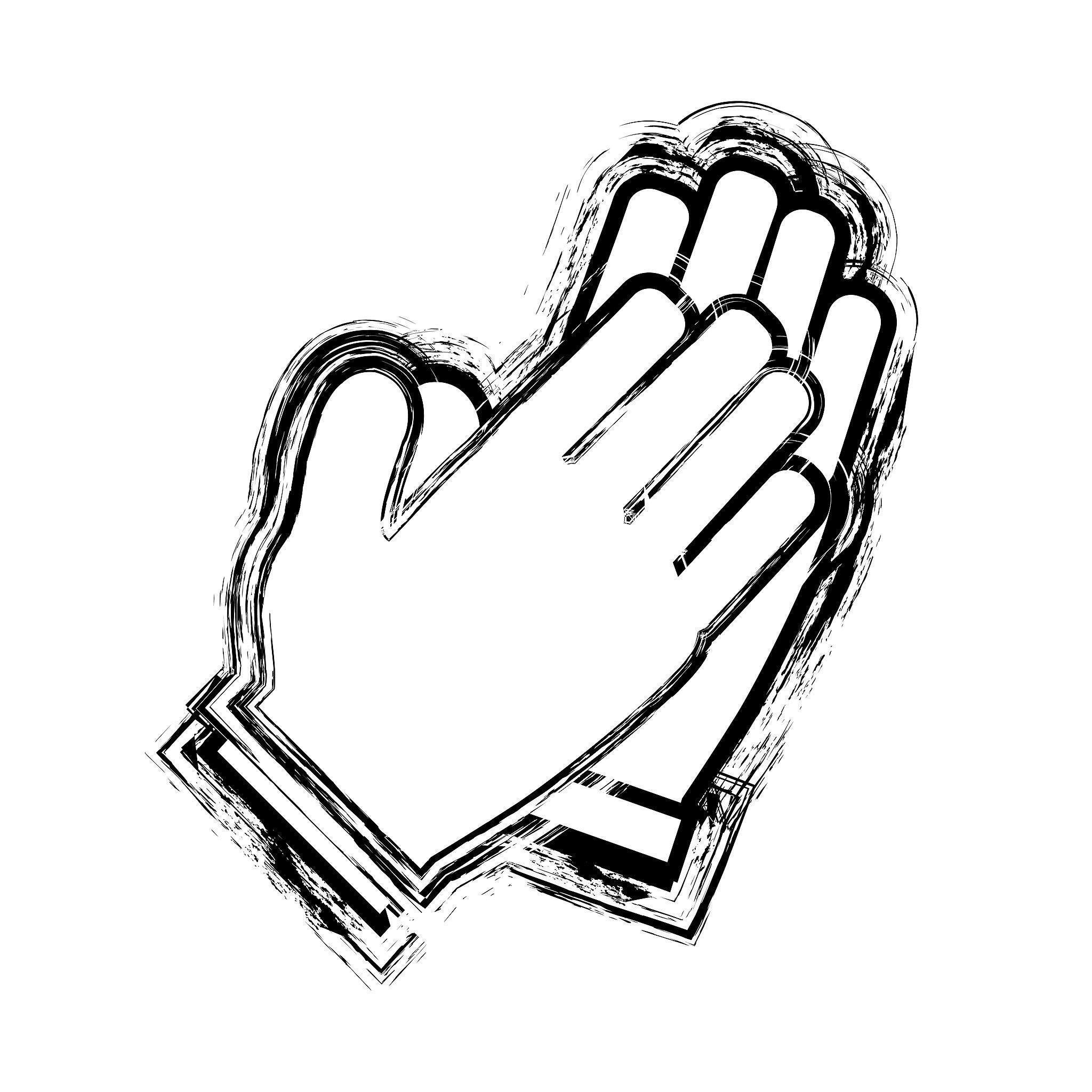
*(pour cette partie il peut être utile d’afficher seulement les titres des livres en sortie afin de s’assurer que l’on récupère bien les bonnes données!)*

Maintenant que les données voulues sont accessibles pour une page, le ‘spider’ va être modifié afin de parcourir les différentes pages (lors d’un clic sur le bouton next en bas de la page).

* modifiez la fonction start\_request or parse afin de parcourir toutes les pages
* stockez le résultat dans un fichier csv

Combien y a t il de livre dans les pages du site?

* ajoutez une colonne afin de numéroter les livre de façon unique par page
* stockez le résultat dans un autre fichier csv.



Bravo! Vous avez scrappé votre premier site!

**Question 1**

Maintenant créez un (ou plusieurs) nouveau(x) spider(s) pour récupérer les colonnes suivantes:

* numéro de page
* numéro unique (comme fait précédemment)
* titre
* prix
* livre en stock (booléen)

### Livrables

* Ls fichier csv demandé.

## Pour aller plus loin

Ajoutez au csv:

* Le genre du livre
* Le code UPC (présent dans la page concernant le livre)

### Ressources

* <https://docs.scrapy.org/en/latest/index.html>
* <https://realpython.com/web-scraping-with-scrapy-and-mongodb/#scrapy-project>
* https://scrapeops.io/python-scrapy-playbook/scrapy-save-csv-files/
* [https://towardsdatascience.com/a-minimalist-end-to-end-scrapy-tutorial-part-i-11e350bcdec0](https://towardsdatascience.com/a-minimalist-end-to-end-scrapy-tutorial-part-i-11e350bcdec0?gif=true)

# Partie 2: Requests and Beautifulsoup

Contrairement à scrapy, qui propose un cadre pour faire du scraping (class, object, structure, etc.), requests et beautifulsoup sont des packages pour que vous puissiez construire votre propre scraper.

Avec “requests” vous pouvez récupérer le contenu html d’une page. Mais lorsque vous regardez le résultat, cela ressemble encore à un énorme désordre. Le package “Beautifulsoup” vous permet d’analyser ou d’extraire des éléments souhaités en utilisant des tag html.

Avant de commencer:

* créer une environnement web scraping
* installer requests
* installer beautifulsoup
* pandas

Source:

* <https://realpython.com/beautiful-soup-web-scraper-python/>
* <https://beautiful-soup-4.readthedocs.io/en/latest/>

## INSEE - Production déchet 2020

* '<https://www.insee.fr/fr/statistiques/5229313#tableau-figure1'>

**Question 1 : récupérer le tableau qui s’affiche sur la page dans l’onglet “tableau”**

Utilise “pandas” pour récupérer le tableau.

**METRO GRENOBLE - POUBELLE POINTS D’APPORTS**

* <https://data.metropolegrenoble.fr/ckan/dataset/points-d-apports-volontaire/resource/9be5666b-f050-4584-b87e-e937e5be080a>

**Question 2: téléchargez le fichier csv sur**

A l’aide de requests, beautifulsoup et pandas écrire un script pour :

* Récupérer le contenu du fichier csv dans un pandas dataframe. Le but est de construire/chercher l’url csv avec beautifulsoup (donc pas copy coller d’url).
* Téléchargez le le fichier et enregistrez le dans un dossier “data”

# Partie 3: Selenium Pagination

Une rubrique d’un site web peut contenir plusieurs pages. Dans le cas où l’url de page change en passant d’une page à l’autre, nous pouvons continuer à utiliser beautifulsoup. Imagine que l’url des pages est construit comme: www.nom.com/page=0 (1,2,3, …..). Avec beautifulsoup cela suffit d’écrire un loop de 0 à la dernière page.

Par contre, si l’url du page ne change pas, et que le site utilise JavaScript, vous ne pouvez plus continuer avec Beautifulsoup. Le package Selenium est capable de faire de la pagination quand l’url ne change pas.

Avec selenium vous pouvez simuler l’action “clique sur un bouton” comme le fait vous quand vous voulez passer d’une page à l’autre.

Avant de commencer:

* selenium

A importer de selenium:

* from selenium import webdriver
* from selenium.webdriver.common.by import By
* from selenium.webdriver.support import expected\_conditions as EC
* from selenium.webdriver.support.ui import WebDriverWait
* from selenium.common.exceptions import TimeoutException

Source:

* <https://selenium-python.readthedocs.io/>
* <https://levelup.gitconnected.com/efficiently-scraping-multiple-pages-of-data-a-guide-to-handling-pagination-with-selenium-and-3ed93857f596>

**ADEME - AGIR POUR LA TRANSITION**

Le site web suivant d'Ademe contient deux pages avec sur chaque page plusieurs articles.

* <https://agirpourlatransition.ademe.fr/particuliers/maison/dechets?page=0>

**Question 1 : récupère le titre du premier article sur chaque page**

Utilise sélénium pour la pagination et récupère le premier article de chaque page. Ceci sans utiliser un loop mais en appuyant sur l'élément dans.

Astuce - suit le schéma suivant:

*driver = webdriver.Firefox()*

*driver.get(url)*

*while True:*

*define soup*

*extraire l'élément souhaité*

*récupère / print info*

*xpath (le pas/tag de l'élément à cliquer pour aller à la page suivante)*

*try:*

*définit l’element à cliquer*

*clic*

*except TimeoutException::*

*print(‘no more pages’)*

*pass*

*driver.close()*

# 

# 

# 

# BONUS: GET TABLE WITH HYPERLINK

Les développeurs de projets industriels (parc éolien, usine chimique, etc.) doivent soumettre une demande d'autorisation au département. Les informations et le statut (refusé ou autorisé) de ces demandes sont accessibles au public.

Sur la page suivante, vous trouverez plusieurs liens pour différentes années ("Tableau ICPE - Année 20XX). Chaque année contient un tableau avec des projets de tous types.

* <https://www.aisne.gouv.fr/Actions-de-l-Etat/Environnement/Installations-classees-pour-la-protection-de-l-environnement/Autorisation-environnementale>

Question : écrire un scraper pour récupérer les tableaux des années 2023, 2022 et 2021 et les renvoyer sous forme de fichier csv.

Pour des cellules qui contiennent un hyperlink, ne récupérer que l'url (tag, href) du lien hypertexte. Pour cela, vous ne pouvez pas utiliser pd.read\_html, car cette fonction récupère toutes les informations présentes dans la cellule spécifique et non l'url complète.

Essayez d'abord d'utiliser pd.read\_html et vérifiez les résultats. Le scraper à écrire doit retourner l'url sous l'hyperlien (par exemple https://www.aisne.gouv.fr/contenu/telechargement/39008/297615/file/APREJET\_IC2023\_086.pdf) et non " Télécharger APREJET\_IC2023\_086 PDF - 0,19 Mb - 24/04/2023 ".