**Web Scraping : Comment collecter les données du web avec Python ?**

Aujourd’hui, des millions de données circulent à travers internet. Il est d’ailleurs admis que 90% des données générées par l’humanité s’est fait sur ces 2 dernières années que sur l’ensemble des décennies précédentes. Ainsi, lorsque vous occupez un rôle de Data Analyst, ou de Data Engineer, savoir collecter ces données est la première étape indispensable de toute analyse décisionnelle. Le web scraping est l’une des techniques les plus efficaces que vous devez maîtriser pour capturer les données qui sont hors du système d’information de l’entreprise. Illustrons celà avec un exemple simple.

Supposons que vous ayez besoin d’obtenir les adresses postales de tous les services ministériels pour renchérir une analyse décisionnelle et produire un tableau de bord géographique. Que faites-vous ? Eh bien, vous pouvez faire une recherche sur Google et copier l’information du site Open Data de l’Etat et le coller dans votre propre fichier. Mais que faire si vous vous voulez alimenter en données un algorithme d’apprentissage automatique ? Dans cette situation, le copier-coller ne fonctionnera pas ! Et c’est là que vous devez utiliser le Web Scraping. Contrairement au processus long et fastidieux d’obtention manuelle de données, le Web Scraping utilise des méthodes automatisées pour obtenir des données de manière continue, et cela en un temps réduit.

Bas du formulaire

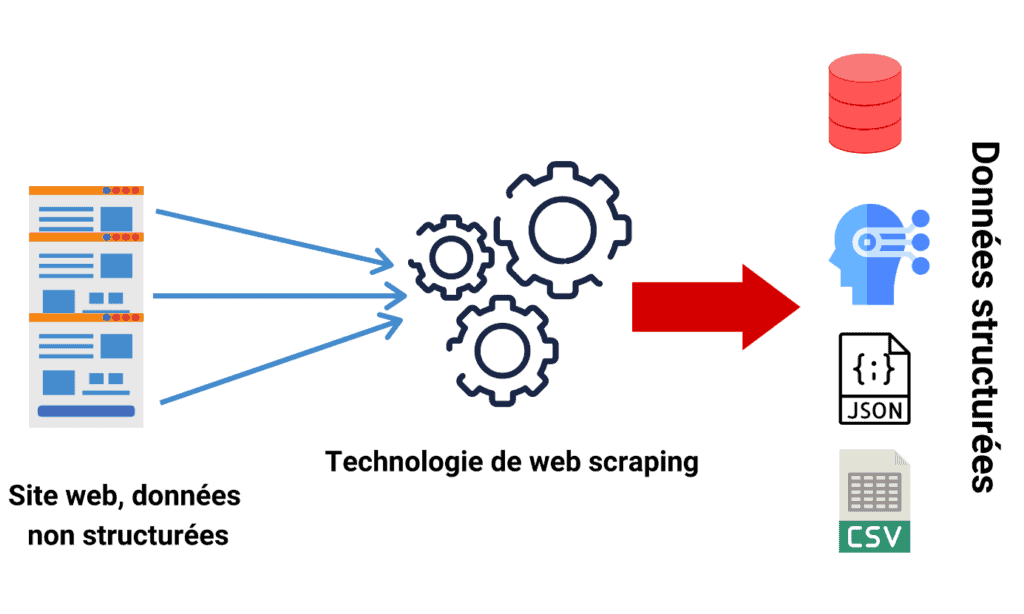
Qu’est-ce que le web scraping ?

Le Web scaping est une méthode automatique qui permet d’obtenir de grandes quantités de données à partir de sites web. La plupart de ces données sont des données non structurées au format HTML qui sont ensuite converties en données structurées dans un tableur ou une base de données afin de pouvoir être utilisées dans diverses applications.

Il existe de nombreuses façons d’effectuer du web scraping pour obtenir des données à partir de sites Web, notamment en utilisant des services en ligne, des API particulières ou même en créant votre code de web scraping à partir de zéro. Dans ce cas, vous devez écrire un script qui automatisera tout le processus, avec Python, par exemple. D’ailleurs, dans notre tutoriel programmation Python, nous vous donnons toutes les concepts à comprendre si vous souhaitez utiliser ce langage en Big Data.

De nombreux grands sites Web comme Google, Twitter, Facebook, StackOverflow, etc. ont des API qui vous permettent d’accéder à leurs données dans un format structuré. Mais il existe d’autres sites qui ne permettent pas aux utilisateurs d’accéder à leurs données dans un format structuré ou qui ne sont tout simplement pas aussi avancés technologiquement. Dans ce cas, il faut utiliser le Web Scraping pour extraire les données du site.

L’extraction de données sur le Web nécessite deux éléments : le **crawler** et le **scraper**. Le **crawler** est un algorithme d’intelligence artificielle qui parcourt le Web pour rechercher les données particulières requises en suivant les liens sur Internet. Le **scraper**, quant à lui, est un outil spécifique créé pour extraire les données du site web. La conception du scraper peut varier considérablement en fonction de la complexité et de l’ampleur du projet, afin qu’il puisse extraire les données rapidement et avec précision.



Maintenant que vous avez une idée de ce que c’est que le web scraping, dans la section suivante de cette chronique nous allons étudier les domaines d’applications du web scraping.

A quoi sert le web scraping au sein d’une entreprise ?

Le web scraping s’utilise dans beaucoup d’activités, si ce n’est dans toutes. Cette technique s’utilise dans de multiples domaines d’applications et ce, dans différents secteurs d’activités, par exemple :

La surveillance des prix

Le web scraping peut être utilisé par les entreprises pour récupérer les données des produits concurrents et ensuite de les utiliser pour fixer le prix optimal de leurs produits afin d’obtenir un revenu maximal.

Étude de marché

Les entreprises peuvent utiliser le web scraping pour leurs études de marché. Les données de haute qualité obtenues en grands volumes peuvent être très utiles aux entreprises pour analyser les tendances de consommation et comprendre dans quelle direction l’entreprise doit se diriger à l’avenir.

Apprentissage automatique

Les modèles d’apprentissage automatique ont besoin de données brutes pour évoluer et s’améliorer en précision. L’apprentissage automatique alimente les merveilles technologiques d’aujourd’hui, comme les voitures sans conducteur, les vols spatiaux, la reconnaissance d’images et de la parole. Cependant, ces modèles ont besoin d’énormément de données variées pour améliorer leur précision et leur fiabilité. Les outils du web scraping peuvent permettre de scraper une grande variété de données, de textes et d’images en un temps relativement court, pour alimenter automatiquement ces modèles.

Un bon projet de web scraping garantit que vous obtenez les données que vous recherchez tout en ne perturbant pas les sources de données.

Surveillance des actualités

Le web scraping des sites d’actualités peut fournir des rapports détaillés sur l’actualité à une entreprise. C’est d’autant plus essentiel pour les entreprises du secteur du journalisme ou qui dépendent de l’actualité quotidienne pour leur fonctionnement. Après tout, les rapports d’actualité peuvent faire ou défaire une entreprise en une seule journée.

Analyse des sentiments

Si les entreprises veulent comprendre le sentiment général des consommateurs à l’égard de leurs produits, l’analyse des sentiments est indispensable. Les entreprises peuvent utiliser le web scraping pour collecter des données à partir des réseaux sociaux tels que Facebook et Twitter afin de connaître le sentiment général sur leurs produits. Cela les aidera à créer des produits que les gens désirent et à prendre de l’avance sur leurs concurrents.

Marketing par courriel

Les entreprises peuvent également utiliser le Web scraping pour le marketing par e-mail. Elles peuvent collecter des identifiants d’email à partir de divers sites en utilisant le web scraping et ensuite envoyer des emails promotionnels et marketing en masse à toutes les personnes possédant ces identifiants d’email.

Il existe encore plein d’autres applications du web scraping mais nous avons vus les principales. Dans la section suivante nous parlerons des outils les plus utilisés pour le web scraping.

Les outils utilisés pour le web scraping

Il existe une multitude d’outils sous forme d’extension, de Framework ou de logiciel sur le marché pour scraper un site web. Nous allons en découvrir quelques-uns dans chacune de ces catégories.

Les extensions de navigateur pour le web scraping

Il existe un grand nombre d’outils pour scraper des sites Web que vous pouvez installer sous forme d’extensions et de modules complémentaires sur votre navigateur pour vous aider à extraire des données de sites Web. Certains d’entre eux sont présentés ci-dessous.

WebScraper

L’extension de navigateur [Webscraper.io](https://webscraper.io/) (Chrome et Firefox) présente l’un des meilleurs outils de scraping web que vous pouvez utiliser pour extraire facilement des données de pages web. Elle a été installée par plus de 250 000 utilisateurs, qui l’ont trouvée incroyablement utile.

Data Miner

L’extension [Data Miner](https://dataminer.io/) est disponible uniquement pour le navigateur Google Chrome et Microsoft Edge. Elle peut vous aider à extraire des données des pages et à les enregistrer dans un fichier CSV ou une feuille de calcul Excel. Contrairement à l’extension gratuite fournie par Webscraper.io, l’extension Data Miner n’est gratuite que pour les 500 premières pages scrappées en un mois – après cela, vous devez souscrire à un plan payant pour pouvoir l’utiliser. Avec cette extension, vous pouvez sraper n’importe quelle page sans penser aux blocages – et vos données restent privées.

Scraper

Scraper est une extension Chrome probablement conçue et gérée par un seul développeur – elle n’a même pas de site Web propre comme les autres ci-dessus. cette extension n’est pas aussi avancé que le reste des extensions de navigateur décrites ci-dessus – Cependant, elle est entièrement gratuite. Le problème majeur associé à celui-ci est qu’il exige de ses utilisateurs qu’ils sachent utiliser **XPath**, car c’est ce que vous allez utiliser. Pour cette raison, il n’est pas adapté aux débutants.

Les logiciels pour le web scraping

Après les extensions, Il existe de nombreux logiciels sur le marché que vous pouvez utiliser pour récupérer toutes sortes de données en ligne sans savoir comment coder. On présentera ci-dessous uniquement les 3 meilleurs logiciels de web scraping.

Octoparse

[Octoparse](https://www.octoparse.com/) rend le scraping web facile pour tous. Avec Octoparse, vous pouvez transformer rapidement un site Web complet en une feuille de calcul structurée en quelques clics seulement. Octoparse ne requiert aucune compétence en matière de codage. Il vous suffit de pointer et de cliquer pour obtenir les données souhaitées. Octoparse peut extraire des données de tous les types de sites Web, y compris les sites Ajaxifiés avec des techniques anti-scraping strictes. Il utilise la rotation des adresses IP pour masquer vos empreintes IP.

Heluim Scraper

[Helium Scraper](https://www.heliumscraper.com/eng/) est un autre logiciel que vous pouvez utiliser pour scraper des sites Web en tant que non-codeur. Vous pouvez capturer des données complexes en définissant vos propres actions – pour les codeurs-, ils peuvent également exécuter des fichiers JavaScript personnalisés. Avec un flux de travail simple, l’utilisation de Helium Scraper est non seulement facile mais aussi rapide car il est doté d’une interface simple et intuitive. Helium Scraper est également l’un des logiciels de scraping web offrant un bon nombre de fonctionnalités, notamment la programmation du scraper, la rotation du proxy, la manipulation de texte et les appels API, entre autres.

ParseHub

[ParseHub](https://www.parsehub.com/) se décline en deux versions : une application de bureau gratuite et une solution de grattage basée sur le cloud, payante, dotée de fonctionnalités supplémentaires et dont l’utilisation ne nécessite aucune installation. L’application de bureau ParseHub vous permet de gratter facilement n’importe quel site Web, même sans compétences en codage. En effet, le logiciel fournit une interface de type pointer-cliquer, destinée à former le logiciel sur les données à gratter. Il fonctionne parfaitement pour les sites Web modernes et vous permet de télécharger les données scrappées dans les formats de fichiers les plus courants.

Les libraires Python pour le web scraping

**Python** est le langage de programmation le plus populaire pour coder des web scrapers en raison de sa syntaxe simple, de sa courbe d’apprentissage et du nombre de bibliothèques disponibles qui facilitent le travail des développeurs. Nous présenterons dans cette section quelques librairies Python permettant d’effectuer le Webscraping.

Scrapy

[Scrapy](https://scrapy.org/) est un Framework python open-source conçu spécifiquement pour le scraping web par les cofondateurs de Scrapinghub, Pablo Hoffman et Shane Evans. Vous vous demandez peut-être « *Qu’est-ce que cela signifie ?*« .

Cela signifie que Scrapy est une solution de web scraping à part entière qui vous décharge d’une grande partie du travail de construction et de configuration de vos **spiders**, et surtout, il gère de manière transparente les cas limites auxquels vous n’avez probablement pas encore pensé.

Quelques minutes après l’installation du Framework, vous pouvez disposer d’un spider entièrement opérationnel pour scraper le web. Dès le départ, les spiders Scrapy sont conçus pour télécharger du HTML, analyser et traiter les données et les enregistrer dans des formats de fichiers CSV, JSON ou XML.

Il existe également un large éventail d’extensions et d’intergiciels intégrés conçus pour gérer les cookies et les sessions, ainsi que des fonctionnalités HTTP telles que la compression, l’authentification, la mise en cache, les agents utilisateurs, le fichier robots.txt et la restriction de la profondeur d’exploration.

L’un des plus grands avantages de l’utilisation du framework Scrapy est qu’il est construit sur **Twisted**, une bibliothèque de mise en réseau asynchrone. Cela signifie que les Scrapy spiders n’ont pas à attendre pour faire des requêtes une par une. Au lieu de cela, ils peuvent effectuer plusieurs requêtes HTTP en parallèle et analyser les données au fur et à mesure qu’elles sont renvoyées par le serveur. Cela augmente considérablement la vitesse et l’efficacité d’un scraper.

Un petit inconvénient de Scrapy est qu’il ne gère pas directement JavaScript comme **Selenium**. Cependant, l’équipe de **Scrapinghub** a créé **Splash**, un navigateur **headless** facile à intégrer, léger et scriptable, spécialement conçu pour le web scraping.

La courbe d’apprentissage de Scrapy est un peu plus raide que celle, par exemple, de l’utilisation de **BeautifulSoup**. Cependant, le projet Scrapy dispose d’une excellente documentation et d’un écosystème extrêmement actif de développeurs sur GitHub et StackOverflow qui publient en permanence de nouveaux plugins et vous aident à résoudre les problèmes que vous rencontrez.

Si vous souhaitez créer votre première araignée Scrapy, n’hésitez pas à consulter le site Web du projet Scrapy.

**Pyspider**

[PySpider](http://docs.pyspider.org/en/latest/) est un autre outil de web scraping que vous pouvez utiliser pour écrire des scripts en Python. Contrairement à Scrapy, il peut produire un rendu du JavaScript et, en tant que tel, ne nécessite pas l’utilisation de **Selenium**. Cependant, il est moins mature que Scrapy, qui existe depuis 2008 et dispose d’une meilleure documentation et d’une meilleure communauté d’utilisateurs. Cela ne rend pas PySpider inférieur. En fait, PySpider est livré avec des fonctionnalités inégalées telles qu’un éditeur de script en interface Web.

Requests

[Requests](https://fr.python-requests.org/en/latest/) est une bibliothèque HTTP qui permet d’envoyer facilement des requêtes HTTP. Elle est construite sur la base de la bibliothèque **urllib**. C’est un outil robuste qui peut vous aider à créer des racleurs web plus fiables. Il est facile à utiliser et nécessite moins de lignes de code.

Il est très important de noter qu’il peut vous aider à gérer les cookies et les sessions, ainsi que l’authentification et la mise en commun automatique des connexions, entre autres choses. Son utilisation est gratuite et les développeurs Python l’utilisent pour télécharger des pages avant d’utiliser un analyseur syntaxique pour extraire les données requises.

BeautifulSoup

**BeautifulSoup** facilite le processus d’analyse syntaxique des données contenues dans les pages Web. Il se place au-dessus d’un analyseur HTML ou XML et vous offre des méthodes Python permettant d’accéder aux données. ***BeautifulSoup est devenu l’un des outils de scraping Web les plus importants du marché en raison de la facilité d’analyse syntaxique qu’il offre.***

En fait, la plupart des didacticiels de scraping Web utilisent BeautifulSoup pour enseigner aux débutants comment écrire des scrapeurs Web. Lorsqu’il est utilisé avec Requests pour envoyer des requêtes HTTP, les web scrapers deviennent plus faciles à développer – beaucoup plus faciles qu’avec Scrapy ou PySpider.

Selenuim

Scrapy, Requests et BeautifulSoup ne vous aideront pas si un site Web est Ajaxifié, c’est-à-dire s’il dépend de requêtes AJAX pour charger certaines parties d’une page via JavaScript. Si vous accédez à une telle page, vous devez utiliser **Selenium**, qui est un outil d’automatisation du navigateur Web. Il peut être utilisé pour automatiser les navigateurs sans tête tels que Chrome et Firefox sans tête. Les versions plus anciennes peuvent automatiser **PhantomJS**.

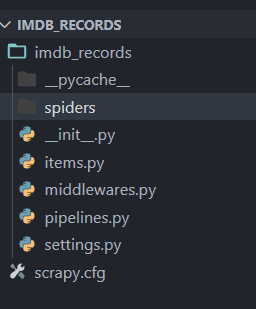
## Le web scraping avec Python scrapy

Dans cette section, nous allons vous montrer comment extraire des noms, des liens et des évaluations de films sur IMDB. Nous stockerons les données extraites dans un fichier CSV.

Avant de pouvoir effectuer du web scraping avec Scrapy, vous devez installer la *bibliothèque Scrapy*. Pour ce faire, exécutez la commande suivante dans votre invite de commande :

pip install scrapy

Dans le répertoire où vous avez créé votre projet *Scrapy imdb\_records*, vous devriez voir un nouveau dossier appelé « *imdb\_records* » avec la structure de répertoire suivante :



Une fois que vous avez créé un projet Scrapy, l’étape suivante consiste à créer une araignée qui va explorer le site Web que vous souhaitez mettre à jour. Le script ci-dessous crée un robot nommé *imdb\_spider* qui explore la page Web du Top 250 de l’IMDB. Notez que le script suivant doit être exécuté dans le dossier « *spiders* » de votre projet « *imdb\_records*« .

scrapy genspider imdb\_spider www.imdb.com/search/title/?groups=top\_250

Une fois que vous aurez créé un spider, un fichier Python nommé « *imdb\_spider* » apparaîtra dans votre dossier « *spiders*« . Le nouveau script Python ressemblera à ceci :

import scrapy

class ImdbSpiderSpider(scrapy.Spider):

name = 'imdb\_spider'

allowed\_domains = ['www.imdb.com/titlesearch//?groups=top\_250']

start\_urls = ['http://www.imdb.com/search/title/?groups=top\_250/']

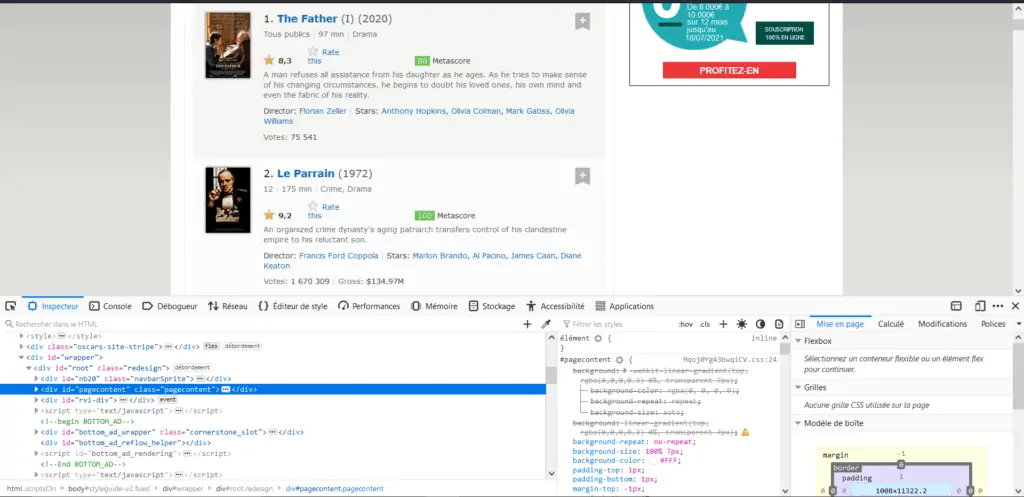
def parse(self, response):

pass

Dans le fichier ci-dessus, la variable **allowed\_domains** est une liste de domaines qui seront recherchés pour vos données souhaitées et la variable **start\_urls** est une liste d’urls où l’araignée commence à chercher les données que vous voulez. Les données à extraire sont spécifiées dans la méthode **parse().**

Toutes les données d’une page Web se trouvent dans les balises HTML. Comme Scrapy explore aveuglément le code HTML, vous devez avoir des connaissances de base en HTML pour savoir quels éléments HTML d’une page Web contiennent les données souhaitées.

L’une des façons de vous familiariser avec les éléments HTML d’un site Web est d’ouvrir le site dans un navigateur, de faire un clic droit sur la page Web et de cliquer sur « *Inspecter*« . Par exemple, si vous ouvrez la page IMDB que nous essayons d’extraire dans Firefox, puis faites un « *clic droit -> Inspecter*« , vous verrez un code HTML comme celui-ci :



À partir des éléments HTML de droite, vous pouvez voir que les noms des films se trouvent à l’intérieur de balises d’ancrage, qui se trouvent à l’intérieur de balises <h3>. Ces balises sont ensuite imbriquées dans une balise div avec une classe « lister-item-content ».

Avec Scrapy, il existe deux façons de rechercher des données dans une page HTML : Vous pouvez soit utiliser des **sélecteurs CSS**, soit utiliser des **sélecteurs XPath**. Dans ce tutoriel, nous allons utiliser les sélecteurs XPath. Expliquons brièvement la fonctionnalité de chaque sélecteur « **XPATH**« .

Tout d’abord, modifiez le fichier « *imdb\_spider.py* » que vous avez généré précédemment de manière à ce qu’il ressemble à ceci :

import scrapy

class ImdbSpiderSpider(scrapy.Spider):

name = 'imdb\_spider'

allowed\_domains = ['imdb.com']

start\_urls = ["https://www.imdb.com/search/title/?groups=top\_250"]

def parse(self, response):

movies = response.xpath("//div[@class='lister-item-content']")

for movie in movies:

movie\_name = movie.xpath(".//h3//a//text()").get()

movie\_link = "https://www.imdb.com" + movie.xpath(".//h3//a//@href").get()

movie\_rating = movie.xpath(".//div[@class='ratings-bar']//strong//text()").get()

yield{

'movie name':movie\_name,

'movie link':movie\_link,

'movie\_rating':movie\_rating

}

Dans le script ci-dessus, l’objet de réponse récupère toutes les balises <div> avec la classe « **lister-item-content**« . Ensuite, une boucle itère sur les balises <div>. Pour récupérer le nom du film à chaque itération de la boucle, la balise <div> récupérée par l’objet réponse est complétée par **.//h3//a//text()**.

Cette étrange séquence de caractères indique à Scrapy qu’à l’intérieur de la balise h3, il faut chercher la balise d’ancrage et récupérer le texte de la balise d’ancrage qui contient essentiellement des noms de films. De même, pour récupérer le titre, la requête **« XPATH » .//h3//a//@href** est ajoutée à la balise div.

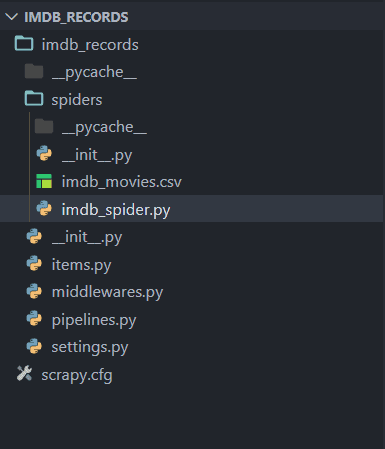
Enfin, les évaluations sont récupérées en ajoutant la requête **.//div[@class=’ratings-bar’]//strong//text()**à la balise <div> récupérée par l’objet de réponse.

Je sais que cela semble un peu compliqué, mais une fois que vous serez à l’aise avec les modèles de la mise en page HTML de la page Web que vous essayez d’extraire, cela commencera à avoir plus de sens.

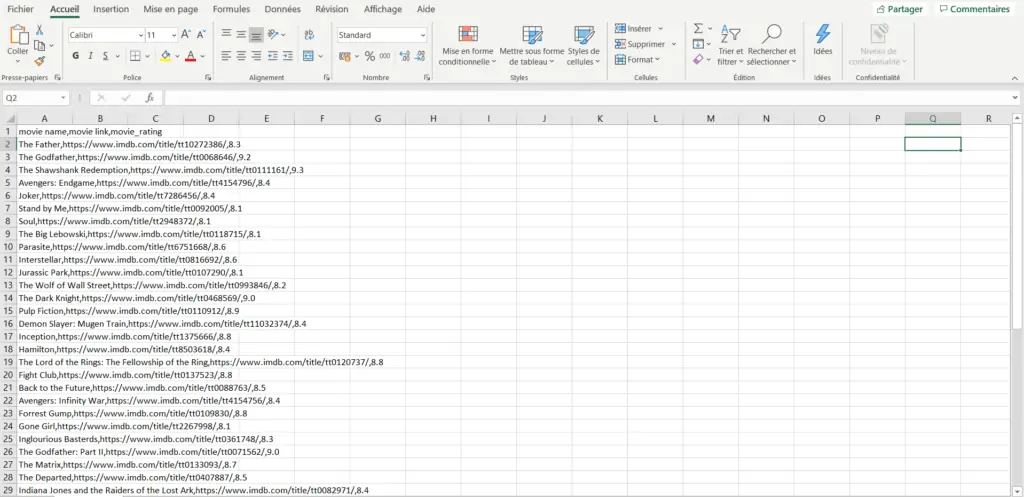
La dernière étape consiste à exécuter la méthode parse du fichier « *imdb\_spider.py* » et à stocker les données extraites dans un fichier CSV. Pour ce faire, exécutez le script suivant.

scrapy crawl imdb\_spider -o imdb\_movies.csv

Une fois que vous aurez exécuté le script ci-dessus, vous verrez qu’un fichier CSV nommé *imdb\_movies.cs* est créé dans le dossier « *spiders* » du projet *imdb\_records*.



Si vous ouvrez le fichier CSV dans Excel ou tout autre tableur, vous devriez constater qu’il contient une liste propre de tous les noms, liens et classements de films que vous venez de récupérer sur le site web.



Voilà ! Nous sommes arrivés au terme de ce tutoriel complet sur le web scraping. Nous espérons que vous avez compris l’intérêt de cette technique dans le marché et son intérêt dans votre carrière Data. Clairement le web scraping et une technique à prendre en considération si vous souhaitez travailler dans le Big Data. Si vous avez des questions par rapport à la création d’un scraper avec Python ou par rapport au web scraping en général, n’hésitez pas à la poser dans les commentaires. Pour aller plus loin dans vos compétences dans le Big Data, nous vous offrons une formation qui vous aidera à maitriser Hadoop.