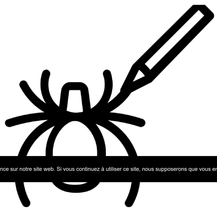
Introduction Au Web Scraping Avec Python

Par Olivier — Dernière mise à jour 22 Nov 2020 — 13,139 — 0



Created by Guilherme Simoes

Le web scraping est la récupération de données de pages web, de façons automatique. C'est une technique, basée sur un principe simple. Qui sert à de nombreuses applications : Moteurs de recherche, comparateurs de prix, outils de monitoring etc..

Dans les lignes qui suivent nous allons, à travers un exemple simple, expliquer un comment scraper un site avec Python. Le but de cet article n'étant pas de faire un tutoriel sur une bibliothèque. Mais de présenter le concept. Nous ne rentrerons donc pas dans les détails quant aux outils utilisés.

Principe du web scraping

Le scraping ou crawling se fait en deux étapes : *le téléchargement*, du code HTML de la page à scraper, et son *parsing*.

Pour obtenir le contenu de la page web (téléchargement) il suffit de faire une requête et HTTP et d'attendre le réponse (Oui c'est aussi simple que ça).

Pour l'illustration nous utiliserons la bibliothèque *requests* de Python. Disons que notre but est d'obtenir le sommaire de la page Wikipédia de la meilleure science fiction de tous les temps *Star Wars* 😍

Obtention du code source (Téléchargement)

Première étape, bien évidemment émettre une requête HTTP avec la fonction *get* de *requests*.

```
# On importe la fonction 'get' (téléchargement) de 'requests'
# Et la classe 'Selector' (Parsing) de 'scrapy'
from requests import get
from scrapy import Selector
# Lien de la page à scraper
url = "https://fr.wikipedia.org/wiki/Star_Wars"
response = get(url)
source = None # Le code source de la page
if response.status_code == 200 :
    # Si la requete s'est bien passee
    source = response.text
```

À ce niveau du script on a le code source de la page sous forme de chaîne de caractère (`str`) dans la variable source. Ceci, bien entendu, si la requête a été effectuée avec succès. C'est-à-dire la réponse à la requête a pour code 2xx (On ne prend en compte que le code 200 pour une question de simplicité).

Récupération d'informations (Parsing)

Maintenant qu'on a tout le code source de la page, il ne nous reste plus qu'à récupérer les informations qui nous intéressent.
Pour cela nous utilisons un Parser de code HTML. Il en existe plusieurs en Python mais nous utiliserons la classe 'Selector' de scrapy parce que je la trouve très simple.
Le Parser étant basé sur les balises HTML, on nous devons identifier les éléments qui correspondent aux informations qui nous intéressent. Pour faire cela, il nous faut inspecter la page à dans notre navigateur.

En faisant un clique droit sur la zone qui nous intéresse puis en cliquant sur "Inspecter l'élément".

Dans la fenêtre qui s'affiche on peut voir la partir du code source de la page qui correspond à la zone qu'on a voulu inspecté. Le but est de trouver une caractéristique des éléments qu'on recherche, en se basant sur les attributs et les relations (imbrications) entre les éléments.

Dans notre cas, nous voulons récupérer tous les titres qui apparaissent dans le sommaire avec leur numéro.
Chaque titre se trouve dans une balise "li". Celle-ci se trouve dans une balise "ul". Elle même dans la section "div" de classe "toc" qui représente la zone grise du sommaire.
Et pour chaque titre ("li"), on récupère le numéro qui le texte du "span" de classe "tocnumber" et le titre qui est le texte du "span" de classe "toctext".

```
if source :
    # Si le code source existe
    selector = Selector(text=source)
    titles = selector.css("div.toc ul > li")
    for title in titles:
        level = title.css("span.tocnumber::text").extract_first()
        name = title.css("span.toctext::text").extract_first()
        print(level + " " + name)
```

En sortie on obtient :

```
1 Univers
1.1 Histoires
1.1.1 Histoire commune
1.1.2 Suite de l'histoire dans la troisième trilogie de
1.1.3 Suite de l'histoire dans l'univers
1.2 Technologie et armes
1.3 Planètes
1.4 Espèces
2 Films
2.1 Trilogie originale (1977 à 1983)
2.1.1 Synopsis
2.1.2 Des débuts difficiles mais un succès sans précédent
2.2 Prélogie (1999 à 2005)
2.2.1 Synopsis
2.2.2 Réalisation : le retour de Lucas aux commandes
2.3 Troisième trilogie (2015 à 2019)
2.4 Projets (après 2019)
2.5 Films dérivés (depuis 2008)
3 Critiques
3.1 Accueil critique et commercial de la saga
3.2 : édition spéciale
4 Univers étendu
4.1 Séries télévisées et téléfilms
4.2 Romans
4.3 Bandes dessinées
4.4 Jeux vidéo
4.5 Autres produits dérivés
5 Inspirations
5.1 Un univers inspiré de la fiction…
5.2 … et de la réalité
6 Postérité
6.1 Impact culturel
6.2 Dans la culture populaire
6.3 Place des femmes dans la saga
7 Notes et références
7.1 Notes
7.2 Références
8 Annexes
8.1 Bibliographie
8.2 Articles connexes
8.3 Liens externes
```

Conclusion

Ceci était une introduction qui met en évidence le cas de figure le plus simple en web scraping. En effet, les informations que vous rechercherez peuvent être beaucoup moins structurées que dans notre exemple. Ce qui peut rendre votre étape de parsing moins simple.
Également, dans certains cas, la réponse qu'on attend ne peut être obtenue par un simple "get". On peut être amené à inspecter le réseau afin de simuler la requête avec les bons arguments (Headers, Cookies, Payloads) et la bonne méthode (POST ou GET dans la plupart des cas).
Une autre méthode consiste à utiliser des navigateurs sans tête (Headless browsers) tel que Selenium. Pour pouvoir simuler le comportement d'un utilisateur sur le site et récupérer le code source ensuite. Ce procédé est très utile quand on veut scraper des sites pour lesquels il faut s'authentifier.

Olivier

ARTICLE PRÉCÉDENT
Créer un modèle de Régression Linéaire avec Python

PROCHAIN ARTICLE
Boîte à outils Python du Data Scientist