TD2A_Eco_Web_Scraping_enonce

August 22, 2017

1 Web-Scraping

Sous ce nom se cache une pratique très utile pour toute personne souhaitant travailler sur des informations disponibles en ligne, mais n'existant pas forcément sous la forme d'un tableau Excel... Bref, il s'agit de récupérer des informations depuis Internet.
Le webscraping est une technique d'extraction du contenu des sites internet, via un programme informatique : nous allons aujourd'hui vous présenter comme créer et exécuter ces robots afin de recupérer rapidement des informations utiles à vos projets actuels ou futurs.

```
In [ ]: from jyquickhelper import add_notebook_menu
        add_notebook_menu()
```

```
Out[ ]: <IPython.core.display.HTML object>
```

1.1 Un détour par le Web : comment fonctionne un site ?

Même si nous n'allons pas aujourd'hui faire un cours de web, il vous faut néanmoins certaines bases pour comprendre comment un site internet fonctionne et comment sont structurées les informations sur une page.
Un site Web est un ensemble de pages codées en HTML qui permet de décrire à la fois le contenu et la forme d'une page Web.

1.1.1 HTML

1.1.2 Les balises

Sur une page web, vous trouverez toujours à coup sûr des éléments comme < head>, < title>, etc. Il s'agit des codes qui vous permettent de structurer le contenu d'une page HTML et qui s'appellent des balises.
Citons, par exemple, les balises < p>, < h1>, < h2>, < h3>, < strong> ou < em>.
Le symbole < > est une balise : il sert à indiquer le début d'une partie. Le symbole < > indique la fin de cette partie.
La plupart des balises vont par paires, avec une «balise ouvrante" et une «balise fermante". (par exemple < p> et < /p>).

Exemple : les balise des tableaux

Balise	Description
$< table >$	Tableau
$< caption >$	Titre du tableau
$< tr >$	Ligne de tableau
$< th >$	Cellule d'en-tête
$< td >$	Cellule
$< thead >$	Section de l'en-tête du tableau
$< tbody >$	Section du corps du tableau
$< tfoot >$	Section du pied du tableau

Application : un tableau en HTML Le code HTML du tableau suivant

```
<table>
   <tr>
      <th>Prénom</th>
      <th>Nom</th>
      <th>Profession</th>
   </tr>
   <tr>
      <td>Mike</td>
      <td>Stuntman</td>
      <td>Cascadeur</td>
   </tr>
   <tr>
      <td>Mister</td>
      <td>Pink</td>
      <td>Gangster</td>
   </tr>
</table>
```

Donnera dans le navigateur

Prénom	Mike	Mister	
Nom	Stuntman	Pink	
Profession	Cascadeur	Gangster	

Parent et enfant Dans le cadre du langage HTML, les termes de parents (parent) et enfants (child) servent à désigner des élements emboîtés les uns dans les autres.
Dans la construction suivante, par exemple :

```
< div>
    < p>
       bla,bla
    < /p>
< /div>
```

On dira que l'élément < div> est le parent de l'élément < p> tandis que l'élément < p> est l'enfant de l'élément < div>.

Mais pourquoi apprendre ça pour scraper me direz-vous ?
Pour bien récupérer les informations d'un site internet, il faut pouvoir comprendre sa structure et donc son code HTML. Les fonctions python qui servent au scrapping sont principalement construites pour vous permettre de naviguer entre les balises

1.1.3 Optionnel - CSS - le style de la page WEB

Quand le bout de code html est écrit, il apaprait sous la forme d'un texte noir sur un fond blanc. Une manière simple de rendre la page plus belle, c'est d'y ajouter de la couleur.
La feuille de style qui permet de rendre la page plus belle correspond au(x) fichier(s) CSS.
Toutes les pages HTML qui font référence à cette feuille de style externe hériteront de toutes ses définitions.
Nous y reviendrons plus en détail dans le TD sur Flask (module Python de création de site internet).

1.2 Scrapper avec python

Nous allons essentiellement utiliser le package BeautifulSoup4 pour ce cours, mais d'autres packages existent (Selenium, Scrapy...).
BeautifulSoup sera suffisant quand vous voudrez travailler sur des pages HTML statiques, dès que les informations que vous recherchez sont générées via l'exécution de scripts Javascipt, il vous faudra passer par des outils comme Selenium.
De même, si vous ne connaissez pas l'URL, il faudra passer par un framework comme Scrapy, qui passe facilement d'une page à une autre ("crawl"). Scrapy est plus complexe à manipuler que BeautifulSoup : si vous voulez plus de détails, rendez-vous sur la page du tutorial https://doc.scrapy.org/en/latest/intro/tutorial.html.

1.2.1 Utiliser BeautifulSoup

Les packages pour scrapper des pages HTML : - BeautifulSoup (pip install bs4) - urllib

```
In [ ]: import urllib
        import bs4
        #help(bs4)
```

1ere page HTML On va commencer facilement, prenons une page wikipedia, par exemple celle de la Ligue 1 de football :
https://fr.wikipedia.org/wiki/Championnat_de_France_de_football_2016-2017
On va souhaiter récupérer la liste des équipes, ainsi que les url des pages Wikipedia de ces équipes.

```
In [ ]: # Etape 1 : se connecter à la page wikipedia et obtenir le code source

        url_ligue_1 = "https://fr.wikipedia.org/wiki/Championnat_de_France_de_football_2016-2017
```

```
        from urllib import request

        request_text = request.urlopen(url_ligue_1).read()
        print(request_text[:1000])
```

```
b'<!DOCTYPE html>\n<html class="client-nojs" lang="fr" dir="ltr">\n<head>\n<meta charset="UTF-8"
```

```
In [ ]: # Etape 2 : utiliser le package BeautifulSoup
        # qui "comprend" les balises contenues dans la chaine de caractères renvoyée par la fonc

        page = bs4.BeautifulSoup(request_text, "lxml")

        #print(page)
```

Si on print l'objet, page créée avec BeautifulSoup, on voit que ce n'est plus une chaine de caractères mais bien une page HTML avec des balises. On peut à présenter chercher des élements à l'intérieur de ces balises.
par exemple, si on veut connaire le titre de la page, on utilise la méthode .find et on lui demande "title"

```
In [ ]: print(page.find("title"))
```

```
<title>Championnat de France de football 2016-2017 ¿  Wikipédia</title>
```

la methode .find ne renvoie que la première occurence de l'élément

```
In [ ]: print(page.find("table"))
```

```
<table>
<caption style="background:#99cc99; color: #000000;">Généralités</caption>
<tr>
<th scope="row" style="width:10.5em;">Sport</th>
<td><a href="/wiki/Football" title="Football">Football</a></td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Organisateur(s)</th>
<td><a href="/wiki/Ligue_de_football_professionnel" title="Ligue de football professionnel">LFP<
</tr>
<tr>
<th scope="row" style="width:10.5em;">Édition</th>
<td><abbr class="abbr" title="Soixante-dix-neuvième (septante-neuvième)">79<sup>e</sup></abbr></
</tr>
<tr>
<th scope="row" style="width:10.5em;">Lieu(x)</th>
<td><span class="datasortkey" data-sort-value="France"><span class="flagicon"><a class="image" h
</tr>
<tr>
```

```
<th scope="row" style="width:10.5em;">Date</th>
<td>du <time class="nowrap date-lien" datetime="2016-08-12"><a href="/wiki/12_ao%C3%BBt" title="
au <time class="nowrap date-lien" datetime="2017-05-20"><a href="/wiki/20_mai" title="20 mai">20
</tr>
<tr>
<th scope="row" style="width:10.5em;">Participants</th>
<td>20 équipes</td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Matchs joués</th>
<td>380</td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Affluence</th>
<td>7 965 830 <small>(20 963 par match)</small></td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Site web officiel</th>
<td>
<p class="plainlinks"><a class="external text" href="http://www.lfp.fr" rel="nofollow">Site offi
</td>
</tr>
</table>
```

Pour trouver toutes les occurences, on utilise .findAll()

```
In [ ]: print("Il y a", len(page.findAll("table")), "éléments dans la page qui sont des <table>"
```

```
Il y a 32 éléments dans la page qui sont des <table>
```

```
In [ ]: print(" Le 2eme tableau de la page : Hiérarchie \n", page.findAll("table")[1])
        print("----------------------------------------------------------")
        print("Le 3eme tableau de la page : Palmarès \n",page.findAll("table")[2])
```

```
 Le 2eme tableau de la page : Hiérarchie
 <table>
<caption style="background:#99cc99; color: #000000;">Hiérarchie</caption>
<tr>
<th scope="row" style="width:10.5em;">Hiérarchie</th>
<td><abbr class="abbr" title="Premier">1<sup>er</sup></abbr> échelon</td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Niveau inférieur</th>
<td><a href="/wiki/Championnat_de_France_de_football_de_Ligue_2_2016-2017" title="Championnat de
</tr>
```

```
</table>
----------------------------------------------------------
Le 3eme tableau de la page : Palmarès
 <table>
<caption style="background:#99cc99; color: #000000;">Palmarès</caption>
<tr>
<th scope="row" style="width:10.5em;">Tenant du titre</th>
<td><a href="/wiki/Paris_Saint-Germain_Football_Club" title="Paris Saint-Germain Football Club">
</tr>
<tr>
<th scope="row" style="width:10.5em;">Promu(s) en début de saison</th>
<td><a class="mw-redirect" href="/wiki/Association_sportive_Nancy-Lorraine" title="Association s
<a href="/wiki/Dijon_Football_C%C3%B4te-d%27Or" title="Dijon Football Côte-d'Or">Dijon FCO</a><b
<a href="/wiki/Football_Club_de_Metz" title="Football Club de Metz">FC Metz</a></td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Vainqueur</th>
<td><b><a class="mw-redirect" href="/wiki/AS_Monaco_Football_Club" title="AS Monaco Football Clu
</tr>
<tr>
<th scope="row" style="width:10.5em;">Deuxième</th>
<td><a href="/wiki/Paris_Saint-Germain_Football_Club" title="Paris Saint-Germain Football Club">
</tr>
<tr>
<th scope="row" style="width:10.5em;">Troisième</th>
<td><a href="/wiki/Olympique_Gymnaste_Club_Nice_C%C3%B4te_d%27Azur" title="Olympique Gymnaste Cl
</tr>
<tr>
<th scope="row" style="width:10.5em;">Relégué(s)</th>
<td><a class="mw-redirect" href="/wiki/AS_Nancy-Lorraine" title="AS Nancy-Lorraine">AS Nancy-Lor
<a class="mw-redirect" href="/wiki/SC_Bastia" title="SC Bastia">SC Bastia</a><br/>
<a class="mw-redirect" href="/wiki/FC_Lorient" title="FC Lorient">FC Lorient</a></td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Buts</th>
<td>991 <small>(2,61 par match)</small></td>
</tr>
<tr>
<th scope="row" style="width:10.5em;"><img alt="Averti" data-file-height="260" data-file-width="
<td>1297</td>
</tr>
<tr>
<th scope="row" style="width:10.5em;"><a class="image" href="/wiki/Fichier:Red_card.svg"><img al
<td>96</td>
</tr>
<tr>
<th scope="row" style="width:10.5em;">Meilleur joueur</th>
<td><span class="flagicon"><a class="image" href="/wiki/Fichier:Flag_of_Uruguay.svg" title="Drap
```

```
</tr>
<tr>
<th scope="row" style="width:10.5em;">Meilleur(s) buteur(s)</th>
<td><span class="flagicon"><a class="image" href="/wiki/Fichier:Flag_of_Uruguay.svg" title="Drap
</tr>
<tr>
<th scope="row" style="width:10.5em;">Meilleur(s) passeur(s)</th>
<td><span class="flagicon"><a class="image" href="/wiki/Fichier:Flag_of_France.svg" title="Drape
</tr>
<tr>
<th scope="row" style="width:10.5em;">Barragiste(s)</th>
<td><a class="mw-redirect" href="/wiki/FC_Lorient" title="FC Lorient">FC Lorient</a></td>
</tr>
</table>
```

1.2.2 Exercice guidé : obtenir la liste des équipes de Ligue 1

La liste des équipes est dans le tableau "Participants" : dans le code source, on voit que ce tableau est celui qui a class = "DebutCarte"
On voit également que les balises qui encerclent les noms et les urls des clubs sont de la forme suivante

```
<a href="url_club" title="nom_club"> Nom du club </a>
```

```
In [ ]: for item in page.find('table', {'class' : 'DebutCarte'}).findAll({'a'})[0:5] :
            print(item, "\n-------")
```

```
<a class="image" href="/wiki/Fichier:France_location_map-Regions-2016.svg"><img alt="France loca
-------
<a href="/wiki/Paris_Saint-Germain_Football_Club" title="Paris Saint-Germain Football Club">Pari
-------
<a href="/wiki/Association_sportive_de_Monaco_football_club" title="Association sportive de Mona
-------
<a href="/wiki/Olympique_lyonnais" title="Olympique lyonnais">Olympique lyonnais</a>
-------
<a href="/wiki/Stade_rennais_football_club" title="Stade rennais football club">Stade rennais FC
-------
```

On n'a pas envie de prendre le premier élément qui ne correspond pas à un club mais à une image.
Or cet élément est le seul qui n'ait pas de title = "".
Il est conseillé d'exclure les élements qui ne nous intéressent pas en indiquant les éléments que la ligne doit avoir au lieu de les exclure en fonction de leur place dans la liste

```
In [ ]: ### condition sur la place dans la liste >>>> MAUVAIS
        for e, item in enumerate(page.find('table', {'class' : 'DebutCarte'}).findAll({'a'})[0:5
            if  e == 0:
                pass
```

```
            else :
                print(item)
```

```
<a href="/wiki/Paris_Saint-Germain_Football_Club" title="Paris Saint-Germain Football Club">Pari
<a href="/wiki/Association_sportive_de_Monaco_football_club" title="Association sportive de Mona
<a href="/wiki/Olympique_lyonnais" title="Olympique lyonnais">Olympique lyonnais</a>
<a href="/wiki/Stade_rennais_football_club" title="Stade rennais football club">Stade rennais FC
```

```
In [ ]: #### condition sur les éléments que doit avoir la ligne >>>> BIEN
        for item in page.find('table', {'class' : 'DebutCarte'}).findAll({'a'})[0:5] :
            if item.get("title") :
                print(item)
```

```
<a href="/wiki/Paris_Saint-Germain_Football_Club" title="Paris Saint-Germain Football Club">Pari
<a href="/wiki/Association_sportive_de_Monaco_football_club" title="Association sportive de Mona
<a href="/wiki/Olympique_lyonnais" title="Olympique lyonnais">Olympique lyonnais</a>
<a href="/wiki/Stade_rennais_football_club" title="Stade rennais football club">Stade rennais FC
```

Enfin la dernière étape, consiste à obtenir les informations souhaitées, c'est à dire dans notre cas, le nom et l'url des 20 clubs.
Pour cela, nous allons utiliser deux méthodes de l'élement item : - getText() qui permet d'obtenir le texte qui est sur la page web et dans la balise < a> - get('xxxx') qui permet d'obtenir l'élément qui est égal à xxxx
Dans notre cas, nous allons vouloir le nom du club ainsi que l'url : on va donc utiliser **getText** et **get("href")**

```
In [ ]: for item in page.find('table', {'class' : 'DebutCarte'}).findAll({'a'})[0:5] :
            if item.get("title") :
                print(item.get("href"))
                print(item.getText())
```

```
/wiki/Paris_Saint-Germain_Football_Club
Paris SG
/wiki/Association_sportive_de_Monaco_football_club
AS Monaco FC
/wiki/Olympique_lyonnais
Olympique lyonnais
/wiki/Stade_rennais_football_club
Stade rennais FC
```

```
In [ ]: # pour avoir le nom officiel, on aurait utiliser l'élément <title>
        for item in page.find('table', {'class' : 'DebutCarte'}).findAll({'a'})[0:5] :
            if item.get("title") :
                print(item.get("title"))
```

```
Paris Saint-Germain Football Club
Association sportive de Monaco football club
Olympique lyonnais
Stade rennais football club
```

Toutes ces informations, on souhaite les conserver dans un tableau Excel pour pouvoir les réuitiliser à l'envie : pour cela, rien de plus simple, on va passer par pandas, parce qu'on le maitrise parfaitement à ce stade de la formation.

```
In [ ]: import pandas

        liste_noms = []
        liste_urls = []

        for item in page.find('table', {'class' : 'DebutCarte'}).findAll({'a'}) :
            if item.get("title") :
                liste_urls.append(item.get("href"))
                liste_noms.append(item.getText())

        df = pandas.DataFrame.from_dict( {"clubs" : liste_noms, 'url' : liste_urls})

In [ ]: df.head()
```

Out[ ]:

	clubs	url
0	Paris SG	/wiki/Paris_Saint-Germain_Football_Club
1	AS Monaco FC	/wiki/Association_sportive_de_Monaco_football_...
2	Olympique lyonnais	/wiki/Olympique_lyonnais
3	Stade rennais FC	/wiki/Stade_rennais_football_club
4	OGC Nice	/wiki/Olympique_gymnaste_club_Nice_C%C3%B4te_d...

1.2.3 Exercice de web scraping avec BeautifulSoup

Pour cet exercice, nous vous demandons d'obtenir 1) les informations personnelles des 721 pokemons sur le site internet pokemondb.net
Les informations que nous aimerions obtenir au final pour les pokemons sont celles contenues dans 4 tableaux : - Pokédex data - Training - Breeding - Base stats
Pour exemple : http://pokemondb.net/pokedex/nincada

2) Nous aimerions que vous récupériez également les images de chacun des pokémons et que vous les enregistriez dans un dossier (indice : utilisez les modules request et shutil) *pour cette question ci, il faut que vous cherchiez de vous même certains éléments, tout n'est pas présent dans le TD*

1.2.4 Aller sur internet avec Selenium

L'avantage du package Selenium est d'obtenir des informations du site qui ne sont pas dans le code html mais qui apparaissent uniquement à la suite de l'exécution de script javascript en arrière plan.

Selenium se comporte comme un utilisateur qui surfe sur internet : il clique sur des liens, il remplit des formulaires etc.
Dans cet exemple, nous allons essayer de aller sur le site de Bing Actualités et entrer dans la barre de recherche un sujet donné.

```
In [ ]: import selenium #pip install selenium
        # télécharger le chrome driver http://chromedriver.storage.googleapis.com/index.html?pat
        path_to_web_driver = "./chromedriver"
```

```
In [ ]: import os, sys
        from pyquickhelper.filehelper import download, unzip_files
        if "win" in sys.platform:
            if not os.path.exists("chromedriver_win32.zip"):
                d = download("http://chromedriver.storage.googleapis.com/2.24/chromedriver_win32
            if not os.path.exists("chromedriver.exe"):
                unzip_files("chromedriver_win32.zip", where_to=".")
        elif sys.platform.startswith("linux"):
            if not os.path.exists("chromedriver_linux64.zip"):
                d = download("http://chromedriver.storage.googleapis.com/2.24/chromedriver_linux
            if not os.path.exists("chromedriver"):
                unzip_files("chromedriver_linux64.zip", where_to=".")
        elif sys.platform.startswith("darwin"):
            if not os.path.exists("chromedriver_mac64.zip"):
                d = download("http://chromedriver.storage.googleapis.com/2.24/chromedriver_mac64
            if not os.path.exists("chromedriver"):
                unzip_files("chromedriver_mac64.zip", where_to=".")
```

On soumet la requête.

```
In [ ]: import time

        from selenium import webdriver
        from selenium.webdriver.common.keys import Keys

        browser = webdriver.Chrome(path_to_web_driver)
        browser.get('https://www.bing.com/news')

        # on cherche l'endroit où on peut remplir un formulaire en utilisant les outils du navig
        # on voit que la barre de recherche est un élement du code appelé 'q' comme query
        # on lui demande de chercher cet élément
        search = browser.find_element_by_name('q')

        # on envoie à cet endroit le mot qu'on aurait tapé dans la barre de recherche
        search.send_keys("alstom")

        # on appuie sur le bouton "Entrée" Return en anglais
        search.send_keys(Keys.RETURN)
```

On extrait les résultats.

```
In [ ]: links = browser.find_elements_by_xpath("//div/a[@class='title'][@href]")

        results = []
        for link in links:
            url = link.get_attribute('href')
            results.append(url)

        len(results)
```

```
Out[ ]: 26
```

```
In [ ]: # on a une pause de 10 secondes pour aller voir ce qui se passe sur la page internet
        # on demande de quitter le navigateur quand tout est fini
        browser.quit()
```

```
In [ ]: print(results)
```

```
['http://www.boursier.com/actions/actualites/rumeurs/alstom-et-vinci-lorgnent-l-extension-du-met
```

1.2.5 Obtenir des informations entre deux dates sur Google News

En réalité, l'exemple de Google News aurait pu se passer de Selenium et être utilisé directement avec BeautifulSoup et les url qu'on réussit à deviner de Google.
Ici, on utilise l'url de Google News pour créer une petite fonction qui donne pour chaque ensemble de (sujet, debut d'une période, fin d'une période) des liens pertinents issus de la recherche Google.

```
In [ ]: import time
        from selenium import webdriver


        def get_news_specific_dates (beg_date, end_date, subject, hl = "fr", gl = "fr", tbm = "n
            '''Permet d obtenir pour une requete donnée et un intervalle temporel
            précis les 10 premiers résultats
            d articles de presse parus sur le sujet'''
            get_string = 'https://www.google.com/search?hl={}&gl={}&tbm={}&authuser={}&q={}&tbs=
            print(get_string)
            browser.get(get_string)

            # La class peut changer si Google met à jour le style de sa page.
            # Cela arrive régulièrement. Dans ce cas, il faut utiliser des
            # outils de débuggage web (Chrome - Outils de développement)
            links = browser.find_elements_by_xpath("//h3[@class='r _gJs']/a[@href]")

            results = []
            for link in links:
                url = link.get_attribute('href')
                results.append(url)
            browser.quit()
            return results
```

```
In [ ]: ### On appelle la fonction créée à l'instant
        browser = webdriver.Chrome(path_to_web_driver)
        articles_mai_2015 = get_news_specific_dates("5/1/2015","5/31/2015","société générale jer
```

```
https://www.google.com/search?hl=fr&gl=fr&tbm=nws&authuser=0&q=société générale jerome kerviel&t
```

```
In [ ]: print(articles_mai_2015)
```

```
['http://www.lemonde.fr/les-decodeurs/article/2015/05/19/affaire-kerviel-sept-ans-de-mysteres_46
```

1.2.6 Utiliser selenium pour jouer à 2048

Dans cet exemple, on utilise le module pour que python appuie lui même sur les touches du clavier afin de jouer à 2048.
Note : ce bout de code ne donne pas une solution à 2048, il permet juste de voir ce qu'on peut faire avec selenium

```
In [ ]: from selenium import webdriver
        from selenium.webdriver.common.keys import Keys

        # on ouvre la page internet du jeu 2048

        browser = webdriver.Chrome(path_to_web_driver)
        browser.get('https://gabrielecirulli.github.io/2048/')

        # Ce qu'on va faire : une boucle qui répète inlassablement la même chose : haut / droite

        # on commence par cliquer sur la page pour que les touches sachent
        browser.find_element_by_class_name('grid-container').click()
        grid = browser.find_element_by_tag_name('body')

        # pour savoir quels coups faire à quel moment, on crée un dictionnaire
        direction = {0: Keys.UP, 1: Keys.RIGHT, 2: Keys.DOWN, 3: Keys.LEFT}
        count = 0

        while True:
            try: # on vérifie que le bouton "Try again" n'est pas là - sinon ça veut dire que le
                retryButton = browser.find_element_by_link_text('Try again')
                scoreElem = browser.find_element_by_class_name('score-container')
                break
            except:
                #Do nothing.  Game is not over yet
                pass
            # on continue le jeu - on appuie sur la touche suivante pour le coup d'après
            count += 1
            grid.send_keys(direction[count % 4])
            time.sleep(0.1)
```

```
        print('Score final : {} en {} coups'.format(scoreElem.text, count))
        browser.quit()
```

```
Score final : 1484 en 166 coups
```

In [ ]: