PROJET OSINT EBAY

Groupe 7

Donovan FERRE, Nathan MICHEL, Nassem AHMED, Bérenger DESGARDIN

Sujet choisi :

Récupérer les données liées aux automobiles sur le site d'Ebay

In [1]:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
from datetime import datetime
from dateutil import parser
import time
from urllib import request
import csv
import re


from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
```

In [2]:

```
# Pour DL Chromedriver  : https://sites.google.com/a/chromium.org/chromedriver/
# Pour DL Chromedriver : https://github.com/mozilla/geckodriver/releases
```

In [3]:

```
vehicules = []
attributs = []
attributs_labels = []
en_tete = True
```

In [4]:

```
label_validators = {
    'Prix': float,
    'État': str,
    'Kilométrage': int,
    'Nombre de portes': int,
    'Année' : int,
    'Nombre de places': int,
    'Type': str,
    'Equipements de Confort': str,
    'Couleur': str,
    'Boîte de Vitesse': str,
    'Equipements de Sécurité': str,
    'Puissance (ch DIN)': int,
    'Equipements Extérieurs': str,
    'Modèle': str,
    'Carnet d\'entretien disponible': str,
    'Equipements Audio & Navigation': str,
    'Date de 1ère immatriculation': int,
    'Constructeur': str,
    'Marque': str,
    'Objet modifié': str,
    'Carburant': str,
    'Lien': str
}

def validator(label, value):
    if label in label_validators.keys():
        if type(label_validators[label](value)) is label_validators[label]:
            return label_validators[label](value)
    return "NaN"
```

In [5]:

```
def SaveInCsv(en_tete):
    for x in range(len(vehicules)):
        ligne = []
        vehicule_clean = {}
        vehicule = vehicules[x]

        for attribut_label in attributs_labels:
            ok = False
            for label in vehicule:
                if attribut_label == label:
                    ligne.append(vehicule[label])
                    ok = True
                    break

            if ok == False:
                ligne.append("")
#                 print(None)


        with open('ebayData.csv', 'a', encoding="utf-8", newline='') as csvfile:
            spamwriter = csv.writer(csvfile, delimiter=';')
            if en_tete==True:
                spamwriter.writerow(attributs_labels)
                en_tete = False
            spamwriter.writerow(ligne)
    return en_tete
```

In [6]:

```
html = None

# TROP SECURE
# url = 'https://www.leboncoin.fr/'

url = 'https://www.ebay.fr/'

# https://www.ebay.fr/b/Automobiles-et-motocyclettes/9800/bn_16549400
id = 'gh-ac'
selector = 'title'

# PATH a modifier en fonction de la machine
chrome_options = webdriver.ChromeOptions()
# chrome_options.add_argument('--headless')

chrome_options.add_argument('--disable-extensions')
chrome_options.add_argument('--profile-directory=Default')
chrome_options.add_argument("--incognito")
chrome_options.add_argument("--disable-plugins-discovery");
chrome_options.add_argument("--start-maximized")
browser = webdriver.Chrome('C:/Users/beren/Desktop/chromedriver.exe', options=chrome_op
tions)

#  PATH a modifier en fonction de la machine
# from selenium.webdriver.firefox.firefox_binary import FirefoxBinary
# monExecutable = FirefoxBinary('C:/Program Files/Mozilla Firefox/firefox.exe')
# browser = webdriver.Firefox(firefox_binary= monExecutable)

browser.get(url)
search = browser.find_element_by_id(id)
# RECHERCHE SUR 'AUTO'
search.send_keys('auto')
search.submit()

# time.sleep(3)
html = browser.page_source

soup = BeautifulSoup(html, 'lxml')



attributs_labels.append('Lien')
attributs_labels.append('Prix')

# ON BOUCLE POUR PARCOURIR PLUSIEURS PAGES
nb_pages_max = 2
page_actuelle = 1

link_pages = soup.find_all(class_='pagination__item', limit=nb_pages_max)
# print(link_pages)

for link_page in link_pages:
#     print(link_page)
    #     page = soup.find(class_='pagination__item', text=page_actuelle)
    #     print(page['href'])
    link_page_href = link_page['href']

    #     On attends que le lien soit clickable
    WebDriverWait(browser, 1).until(EC.element_to_be_clickable((By.XPATH, '//a[@href="'
```

```
+link_page_href+'"]')))
    link = browser.find_element_by_xpath('//a[@href="'+link_page_href+'"]')

    print("Lien de la page : " + str(link_page_href))
    link.click()

#     soup = BeautifulSoup(page)
#     html = browser.page_source
#     WebDriverWait(browser, timeout).until(EC.presence_of_element_located((By.XPATH,
 '//a[@href="'+link_page_href+'"]')), 'Time out')
#     WebDriverWait(browser, 10).until(EC.presence_of_element_located((By.ID, 'mainCont
ent')))


    html = request.urlopen(link_page_href)
    time.sleep(3)
    soup = BeautifulSoup(html, 'lxml')
    vehicule_on_page = soup.find_all(class_='s-item__info clearfix')

#     print("Nombre de véhicules : "+ str(len(vehicule_on_page)))

    for item in vehicule_on_page:
        a = item.find('a', href=True)
        link_href = a['href']
#         print(link_href)

         #     AFTER PAGE CHANGE
#         timeout = 10
#         WebDriverWait(browser, timeout).until(EC.presence_of_element_located((By.XPAT
H, '//a[@href="'+link_page_href+'"]')), 'Time out')

        #     On attends que le lien soit clickable
        try:
            link = WebDriverWait(browser, 10).until(EC.element_to_be_clickable((By.XPAT
H, '//a[@href="'+link_href+'"]')), 'Time Out')
    #           browser.find_element_by_xpath('//a[@href="'+link_href+'"]')

            vehicule = {}
            vehicule_name = item.find('h3').text

            action=ActionChains(browser)
            action.move_to_element(link).perform()

    #         print("Element is visible? " + str(link.is_displayed()))
#             print("Lien du vehicule : " + str(link_href))
            link.click()

            try:
        #         time.sleep(1)

                x = []
                y = []

                html = browser.page_source
                soup = BeautifulSoup(html, 'lxml')
        #         print(soup.find(class_='section'))
                section = soup.find(class_='section')
        # #         print(section.table)

                prix_span = soup.find(attrs={'itemprop': 'price'})
                prix = prix_span['content']
```

```
                prix = float(re.sub('\s', '', prix.replace(' EUR', '').replace(',', '.'
)))

                y.append('Lien')
                x.append(link_href)

                y.append('Prix')
                x.append(prix)

                presentation = section.find('table', attrs={'role': 'presentation'}, id
=None)
        #         print(presentation)

                attrLabels = presentation.find_all(class_='attrLabels')


                values = presentation.find_all("td")
                for value in values:
                    attr = value.find("span")
                    if attr:
                        x.append(attr.text)

                for labels in attrLabels:
                    string = labels.text
                    string_clean = string.replace("\t", "")
                    string_clean2 = string_clean.replace("\n", "")
                    string_clean3 = string_clean2.replace(":", "")
                    string_clean4 = string_clean3.replace(";", ".")
                    string_clean5 = string_clean4[1:-1]
                    y.append(string_clean5)

                    if string_clean5 not in attributs_labels:
                        attributs_labels.append(string_clean5)

        #         print(len(y))
        #         print(y)
        #         print(len(x))
        #         print(x)

                for z in range(0, len(x)):
                    label = y[z]
                    value = x[z]

                    vehicule[label] = validator(label, value)
#                     vehicule[label] = value

#                     print(vehicule)
                vehicules.append(vehicule)

        #         time.sleep(1)
            except:
                  print("Something went wrong")
            finally:
                browser.back()


        except:
              print("Time Out")


    en_tete = SaveInCsv(en_tete)
```

```
# print(attributs_labels)
# print(attributs)
# print(vehicules)



browser.quit()
```

```
Lien de la page : https://www.ebay.fr/sch/i.html?_from=R40&_nkw=auto&_saca
t=0&_pgn=1
Something went wrong
Something went wrong
Something went wrong
Something went wrong
Something went wrong
Something went wrong
Something went wrong
Lien de la page : https://www.ebay.fr/sch/i.html?_from=R40&_nkw=auto&_saca
t=0&_pgn=2
Time Out
Time Out
Something went wrong
Something went wrong
Something went wrong
Something went wrong
Time Out
```

In [7]:

```
vehicules_clean = []


# attributs_labels

for x in range(len(vehicules)):
    vehicule_clean = {}
    vehicule = vehicules[x]
#     print(vehicule)

    for attribut_label in attributs_labels:
        ok = False

        for label in vehicule:
            if attribut_label == label:
                vehicule_clean[attribut_label] = vehicule[label]
                ok = True
                break

        if ok == False:
            vehicule_clean[attribut_label] = None

#     print(vehicule_clean)
    vehicules_clean.append(vehicule_clean)

# print(vehicules_clean)
```

In [8]:

```
print(attributs_labels)
```

```
['Lien', 'Prix', 'État', 'Kilométrage', 'Nombre de portes', 'Année', 'Nomb
re de places', 'Type', 'Equipements de Confort', 'Couleur', 'Boîte de Vite
sse', 'Nombre de propriétaires précédents', 'Carburant', 'Equipements de S
écurité', 'Puissance (ch DIN)', 'Equipements Extérieurs', 'Modèle', "Carne
t d'entretien disponible", 'Equipements Audio & Navigation', 'Date de 1ère
immatriculation', 'Constructeur', 'Marque', 'Durée garantie occasion', 'Da
te du Contrôle Technique', 'Objet modifié', 'Date expiration garantie cons
tructeur', 'CO2 (g/km)', 'brand', 'mpn', 'Description de la modification',
'Pièces classiques', 'Alimentazione', "Veicolo d'epoca", 'Marca', 'Cambi
o', 'Tipo', 'Posti', 'Data di immatricolazione', 'Venditore', 'Modello',
'Type du véhicule']
```

In [9]:

```
records = []

for x in range(len(vehicules_clean)):
#     vehicule = vehicules_clean[x]
#     print(vehicule)

#     vehicule_attributs = []
#     for attribut in vehicule:
#         print(attribut)
#         vehicule_attributs.append(attribut)
    records.append(vehicules_clean[x])


df = pd.DataFrame(records, columns=attributs_labels)
```

In [10]:

```
df.head()
```

Out[10]:

	Lien	Prix	État	Kilométrage	Nombre de portes	Année	Nombre de places
0	https://www.ebay.fr/itm/Mercedes-Benz-Classe-S...	3000.0	Objet ayant été utilisé. Consulter la descript...	249000.0	4.0	2002.0	5.0
1	https://www.ebay.fr/itm/Mercedes-C220/27440717...	9600.0	Objet ayant été utilisé. Consulter la descript...	136000.0	NaN	2008.0	NaN
2	https://www.ebay.fr/itm/voiture-Toyota-RAV-4-I...	4150.0	Objet ayant été utilisé. Consulter la descript...	94000.0	NaN	2007.0	NaN
3	https://www.ebay.fr/itm/Mehari-CT-OK/224101793...	15000.0	Objet ayant été utilisé. Consulter la descript...	10.0	2.0	1977.0	4.0
4	https://www.ebay.fr/itm/FORD-MUSTANG-5-0-GT-FA...	6500.0	None	44000.0	NaN	1996.0	NaN

5 rows × 41 columns

In [11]:

```
df.shape
```

Out[11]:

```
(86, 41)
```

In [12]:

```
# # DEBUT DE STATS - LAISSE DE COTE
# from statistics import mean, median, mode, stdev

# prixs = df['Prix'].to_numpy()
# prix_moyenne = mean(prixs)
# prix_mediane = median(prixs)
# km = df['Kilométrage'].to_numpy()
# km = list(map(int, km))
# km_moyenne = mean(km)
# km_mediane = median(km)
# année = df['Année'].to_numpy()
# année = list(map(int, année))
# année_moyenne = mean(année)

# d = {'Prix moyen': [prix_moyenne], 'Prix médian': [prix_mediane], 'Kilometrage moye
n': [km_moyenne], 'Kilometrage médian': [km_mediane], 'Année moyenne': [année_moyenne]}
# df2 = pd.DataFrame(data = d)

# df2.head()
```

In [ ]:

```

```