

Facebook-Scraping :

Dites “*Ciao!*” au contact qui vous supprime de la liste d'amis

Lucic A., Moezzibadi, M., Mbengue M.

Techniques de programmation - Python

Mars 2021



- **Envie de savoir qui vous a supprimé de ses amis en FB!**



Figure 1: Fun!

Objectifs :

- ① Scraping des données sur Facebook
- ② Comparer la liste d'amis régulièrement
- ③ Envoi-de message automatique




			
+/-	Beautiful Soup	Selenium	Graph API
Pros	<ul style="list-style-type: none">* User friendly* Facile à apprendre et à maîtriser	<ul style="list-style-type: none">* Polyvalent* Framework conçu pour <i>Java Scraping</i>	<ul style="list-style-type: none">* Accès simple* Réalisation des scénarios plus avancés
Cons	<ul style="list-style-type: none">* Installation des <i>Dependencies</i>* Inefficace	<ul style="list-style-type: none">* Pas destiné au Web Scraping* Inefficace	<ul style="list-style-type: none">* Modifier ou désactiver tout accès par FB* Fiasco de <i>Cambridge Analytica</i>

Figure 2: Outils disponibles pour FB Scraping

Pourquoi Selenium?

- Facebook n'est pas un site régulier avec un budget limité
- Un système anti-bot très puissant
- `urllib.request` => Document HTML incomplet
- Selenium utilise un exécutable (*webdriver*) afin de contrôler *Chrome* (*Firefox*)
- Une combinaison de Beautiful Soup et Selenium pour le **Scraping dynamique**

Importer les modules utilisés dans le but du Scraping

```
import time
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

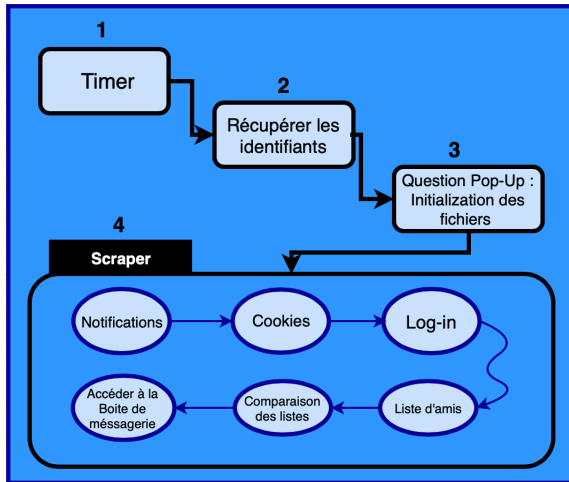



Figure 3: Différentes parties du code

Problème!

Comment éviter les tâches répétitives de changements manuels des paramètres en début du code?

- Un fichier de configurations
- Le module d'*Argparse*

```
(base) moezzibadi@MacBook-Pro-de-Moezzibadi FB % python FB.py -p -u 'Bishgani1'  
Password: 
```

Argparse (*Optparse* de *r*) nous permet de:

- créer une interface de ligne de commande
- définir une aide pour l'appel de chaque argument
- préciser son objectif et son mode d'usage



Timer

- 1ère partie :

① Faire tourner le code régulièrement:

```
t = time.time()
while True:
    TROUVE_ENVOI(user_id,password) #: La fonction principale
    time.sleep(86400)
```

② Faire tourner le code à un instant précis:

```
from datetime import datetime
from threading import Timer
x=datetime.today()
y=x.replace(day=x.day+1, hour=00, minute=2, second=0, microsecond=0)
delta_t=y-x
secs=delta_t.seconds+1
TROUVE_ENVOI(user_id,password) #: La fonction principale
t = Timer(secs, TROUVE_ENVOI)
t.start()
```

• 2ème partie : Créer une interface de ligne de commande

```
import argparse, getpass
class Password:
    DEFAULT = 'Prompt if not specified'
    def __init__(self, value):
        if value == self.DEFAULT:
            value = getpass.getpass('LDAP Password: ')
        self.value = value
    def __str__(self):
        return self.value
class PasswordPromptAction(argparse.Action):
    def __call__(self, parser, args, values, option_string=None): # If no value is given on the cmdline prompt for password.
        if values: # Ideally a security warning could be generated here.
            setattr(args, self.dest, values)
        else:
            setattr(args, self.dest, getpass.getpass())
parser = argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument('-u', '--username', help='Specify username',
                    default=getpass.getuser())

## _StoreAction(option_strings=['-u', '--username'], dest='username', nargs=None, const=None, default='moezzibadi', type=None, choices=None,
parser.add_argument("-p", "--password", type=str, action=PasswordPromptAction, nargs='?')

## PasswordPromptAction(option_strings=['-p', '--password'], dest='password', nargs='?', const=None, default=None, type=<class 'str'>, choi
args = parser.parse_args()
```


Initialiser les fichiers des contacts

- 3 ème partie : Créer un message pop-up pour initialiser les fichiers

```
import sys #Question pop up
question="Est-ce que c'est la première fois que vous lancez le code? "
valid=True
def PopUp(question):
    valid = {"oui": True, "non": False}
    while True:
        sys.stdout.write(question)
        choix = input().lower()
        if choix in valid:
            return valid[choix]
        else:
            sys.stdout.write("Veuillez répondre par 'oui' ou par 'non'. ")
    valid=PopUp(question)
```

- Autoriser l'automatisation à distance pour le navigateur

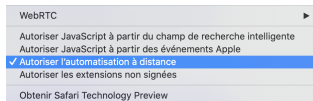


Figure 4: Automatisation dans le menu bar

4ème partie : Scraping

- Désactiver les notifications

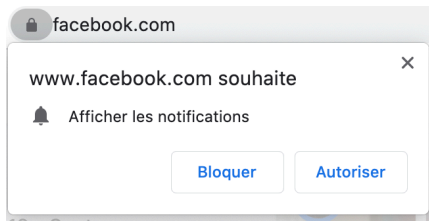


Figure 5: Demande d'affichage des notifications par FB

```
# Créer une instance de ChromeOptions class :  
options = webdriver.ChromeOptions()  
# Ajouter chrome switch pour désactiver la notification :  
options.add_argument('--disable-notifications')  
# Passer l'instance de ChromeOptions à ChromeDriver Constructor :  
browser = webdriver.Chrome(ChromeDriverManager().install(), options=options)
```

4ème partie : Scraping

- Se connecter avec les identifiants:

```
browser.implicitly_wait(10)
browser.get("https://www.facebook.com")
time.sleep(3)
browser.find_element_by_xpath('//*[@title="Tout accepter"]').click()
WebDriverWait(browser, 5).until(EC.element_to_be_clickable((By.XPATH, \
... '//*[@title="Tout accepter"]')))).click()
element=browser.find_element_by_id("email")
element.send_keys(user_id)
passwor=browser.find_element_by_id("pass")
passwor.send_keys(password)
time.sleep(3)
clicker=browser.find_element_by_xpath('//*[@name="login"]')
clicker.click()
```

Note : *Ça y est! On a réussi à se connecter.*



Défiler la page

- Charger la page de la liste d'amis:

```
time.sleep(3)
browser.get("https://www.facebook.com/me/friends")
```

- `execute_script`: Une interface qui permet d'exécuter les commandes de JavaScript:

```
time.sleep(3)
##----Défiler vers le bas-----
while True:
    browser.execute_script('window.scrollTo(0,document.body.scrollHeight);')
    time.sleep(1)
    browser.execute_script('window.scrollTo(0,0);')
    time.sleep(1)
    try:
        exit_command=browser.find_element_by_xpath("//*[@contains(text(),'Photos de vous')]")
        break
    except:
        continue
```

- `find_element_by_xpath` une méthode pour localiser des éléments

- stocker les noms avec les identifiants des comptes:

```
source_page=browser.page_source
soup=BeautifulSoup(source_page, 'html.parser')
friendList=soup.find_all('div',{'class':'buofh1pr hv4rvrfc'})#<div class="buofh1pr hv4rvrfc">
nn=str(friendList).count('buofh1pr hv4rvrfc')
amis2=[None] * nn
Ids2=[None] * nn
for i in range(0,nn):
    amis2[i]=re.findall('dir="auto">(.*?)</span></a></div><div',str(friendList[i]))
    Ids2[i]=re.findall('w" href="https:\\\\www\\.facebook\\.com\\/(.*?)\\/friends_mutual',\
...str(friendList[i]))
    #Le cas de ID
    if not Ids2[i] : Ids2[i]=re.findall\
...('w" href="https:\\\\www\\.facebook\\.com\\/profile\\.php?id=(.*?)&',str(friendList[i]))
    if not Ids2[i] : Ids2[i]=re.findall\
...('wi8" href="https:\\\\www\\.facebook\\.com/(.*?)" role="link" tabindex="0"><span',\
...str(friendList[i]))
```

Stockage & comparaison des noms des listes

- Lecture des fichiers:

```
if valid==False:
    with open("List_amis.txt", "r") as ListAm:
        amis1 = json.load(ListAm)
    with open("List_ID.txt", "r") as ListId:
        Ids1 = json.load(ListId)
```

- Mis à jour des fichiers:

```
with open("List_amis.txt", "w") as ListAm:
    json.dump(amis2, ListAm)
with open("List_ID.txt", "w") as ListAm:
    json.dump(Ids2, ListAm)
if valid==True:
    print('Les fichiers sont créés')
    sys.exit()
ID=['']
for i in amis1:
    if i not in amis2: print(i, "t'a retiré de sa liste")
for i in Ids1:
    if i not in Ids2: ID=i
ID=str(ID)
ID=ID.replace("'", "")
ID=ID.replace('[', '')
ID=ID.replace(']', '')
```

Envoie de message

```
browser.get('https://www.facebook.com/messages/t/'+ID)
message="Ciao mon ami(e), Merci de ne pas répondre à ce message automatique."
time.sleep(3)
message_box=browser.find_element_by_css_selector('div.notranslate._5rpu')
message_box.send_keys(message)
time.sleep(10)
Envoi=browser.find_element_by_xpath('/html/body/div[1]/div/div[1]/div/div[3]/div/div/div[1]/div[1]\
.../div[2]/div/div/div/div/div/div[1]/div[2]/div/div/div/div[2]/div/form/div/div[3]/span[2]/div')
Envoi.click()
time.sleep(5)
TROUVE_ENVOI(user_id,password)
```

```
▼ <div aria-describedby="placeholder-199g3" aria-label="Aa" class=
"notranslate _5rpu" contenteditable="true" role="textbox" spellcheck=
"true" style="outline: none; -webkit-user-select: text; white-space: pre-
wrap; word-wrap: break-word;" tabindex="0"> = $0
```

Figure 6: Méthode de *find_element_by_css_selector* pour trouver une *Class*



- Présence des risques lors du FB Scraping : se faire bloquer, des fautes graves, pénalité juridique
- Beautiful Soup est très utile pour *HTML* mais pas suffisant quant à *JavaScript*.
- Extraction des données des réseaux sociaux est compliqué que celle d'autres sites.
- Problème rencontré avec l'*API* de *Facebook*

