

```
# Importation des modules nécessaires
from selenium import webdriver # Importe la classe pour contrôler le navigateur via Selenium
from selenium.webdriver.common.by import By # Importe les stratégies de recherche d'éléments
from selenium.webdriver.common.keys import Keys # Importe des touches clavier pour l'interaction
from selenium.webdriver.support.ui import WebDriverWait # Importe les attentes explicites
from selenium.webdriver.support import expected_conditions as EC # Importe les conditions attendues

import pandas as pd # Importe la bibliothèque pandas pour manipuler les données tabulaires
from tabulate import tabulate # Importe une fonction pour afficher des tableaux de données
import datetime # Importe le module de manipulation de dates et heures
import time # Importe le module pour gérer les délais de temps
import json # Importe le module pour manipuler le format JSON
import os # Importe le module pour interagir avec le système d'exploitation
```

Selenium est une bibliothèque de test automatisée largement utilisée pour contrôler les navigateurs web de manière programmatique. Elle peut également être utilisée pour extraire des données à partir de sites web, comme Twitter, même en l'absence d'une API officielle.

Suivre l'évolution des followers, des likes, des retweets et des réponses sur Twitter sans utiliser d'API peut être complexe en raison de la nature dynamique du site et des limites du web scraping. Pour y parvenir, on peut extraire les premières métriques (followers, likes, retweets, réponses) lors de la première collecte, puis effectuer des collectes ultérieures à intervalles réguliers pour suivre les changements. En comparant les nouvelles métriques avec les précédentes, on peut calculer les différences et mettre à jour une base de données avec ces informations. Cependant, cette méthode n'est pas aussi précise ni en temps réel qu'une API officielle, peut rencontrer des limites et nécessite de respecter les termes d'utilisation du site. Utiliser une API officielle reste la méthode recommandée pour un suivi plus fiable et précis des métriques.

Ce code initialise un navigateur Chrome en utilisant le pilote chromedriver.exe (ou un navigateur Edge avec msedgedriver.exe en décommentant la ligne appropriée). Ensuite, il accède à la page de connexion Twitter en utilisant la méthode .get().

```
driver = webdriver.Chrome(r'C:\Users\LENOVO\Desktop\Test_DE\Section_1\chromedriver.exe')
#driver = webdriver.Edge(executable_path=r'C:\Users\LENOVO\Desktop\msedgedriver.exe')
```

```
driver.get("https://twitter.com/login")
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_12372\4254219363.py:1:  
DeprecationWarning: executable_path has been deprecated, please pass  
in a Service object
```

```
driver = webdriver.Chrome(r'C:\Users\LENOVO\Desktop\Test_DE\  
Section_1\chromedriver.exe')
```

Ce code cherche et remplit les champs de nom d'utilisateur et de mot de passe sur la page de connexion Twitter. Il localise d'abord le champ d'entrée du nom d'utilisateur en utilisant la méthode `.find_element()` avec `By.NAME`, puis utilise `.send_keys()` pour y entrer le nom d'utilisateur ("USER123" dans cet exemple). Ensuite, il trouve le bouton "Next" en utilisant une expression XPath et utilise `.click()` pour le cliquer.

```
# Find and fill in the username and password fields  
username_input = driver.find_element(By.NAME, "text")  
username_input.send_keys("YOUR_USERNAME")  
next_button = driver.find_element(By.XPATH, "//div[@role='button']  
[contains(.,'Next')]")  
next_button.click()
```

Ce code localise le champ de saisie du mot de passe sur la page de connexion Twitter en utilisant la méthode `.find_element()` avec `By.NAME`. Ensuite, il entre le mot de passe ("PASS123" dans cet exemple) dans le champ en utilisant `.send_keys()`. Enfin, il localise le bouton de connexion en utilisant une expression XPath et utilise `.click()` pour se connecter à Twitter.

```
password_input = driver.find_element(By.NAME, "password")  
password_input.send_keys("YOUR_PASSWRD")  
login = driver.find_element(By.XPATH, "//div[@role='button']  
[contains(.,'Log in')]")  
login.click()
```

Utilisation de Selenium pour ouvrir la page web twitter en utilisant le navigateur Edge ou chromedriver ainsi que spécifier le nom d'utilisateur à extraire les données

```
# Specify the username  
username_input = "elonmusk" #K22Samay  
  
# Open the Nordstrom website  
#driver.get("https://google.com")  
driver.get(f"https://twitter.com/{username_input}")
```

Ce code crée un chemin de fichier CSV en fonction du nom d'utilisateur saisi précédemment et du répertoire de profil. Ensuite, il vérifie si le fichier CSV existe déjà à cet emplacement en utilisant `os.path.exists()`. Si le fichier existe, il charge les données existantes dans un DataFrame à l'aide de Pandas avec `pd.read_csv()`. Sinon, s'il n'existe pas, il crée un DataFrame vide.

```

csv_file_path = rf'C:\Users\LENOVO\Desktop\Test_DE\Section_1\Profiles\
{username_input}.csv'

# Check if the CSV file already exists
if os.path.exists(csv_file_path):
    existing_data = pd.read_csv(csv_file_path)
else:
    existing_data = pd.DataFrame()

```

Actualiser le navigateur

```
driver.refresh()
```

Extraire les données de l'utilisateur (nom d'utilisateur, de profil, nombre de posts, de followers, de following, photo de profile, de couverture, bio, badge de vérification, date de naissance, joining, lien, catégorie, localisation)

La méthode XPath est une technique utilisée dans Selenium pour localiser et extraire des éléments d'une page web. Elle consiste à définir un chemin d'accès spécifique à un élément en utilisant des balises, des attributs et des relations hiérarchiques entre les éléments HTML. Cela permet de cibler précisément l'élément souhaité, même si sa position ou sa structure change. Par exemple, "//div[@class='nom-de-classe']" cible tous les éléments ayant la classe spécifiée. En utilisant XPath, Selenium peut trouver et interagir avec des éléments tels que des champs de texte, des boutons et des liens sur une page web.

```

user_list = []
profile = driver.find_element(By.XPATH, f'//div[@class="css-1dbjc4n"]')
profile1 = driver.find_element(By.CSS_SELECTOR, '.css-1dbjc4n.r-
lifxtd0.r-ymttw5.r-ttdzmv')

# Get the current date and time
current_datetime = datetime.datetime.now()
current_date = current_datetime.strftime("%Y-%m-%d") # Format: YYYY-
MM-DD
current_time = current_datetime.strftime("%H:%M:%S") # Format:
HH:MM:SS

username = profile.find_element(By.XPATH,
"/html/body/div[1]/div/div/div[2]/main/div/div/div/div/div/div[3]/
div/div/div/div/div[2]/div[1]/div/div[2]/div/div/div/span").text
Profile_name = profile.find_element(By.XPATH,
"/html/body/div[1]/div/div/div[2]/main/div/div/div/div/div/div[3]/
div/div/div/div/div[2]/div[1]/div/div[1]/div/div/span/span[1]").text
posts_number = profile.find_element(By.XPATH, "//div[@class='css-
1dbjc4n r-16y2uox r-1wbh5a2 r-1pi2tsx r-1777fci']//div[@class='css-
1dbjc4n r-1habvwh']//div[@class='css-901oao css-1hf3ou5 r-1bwzh9t r-
37j5jr r-n6v787 r-16dba41 r-1cwl3u0 r-bcqeoo r-qvutc0']").text

```

```

Followers = profile.find_element(By.XPATH,
'//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div/div/div[3]/div/div/div/div/div[5]/div[1]/a/span[1]/span').text
Following = profile.find_element(By.XPATH,
'//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div/div/div[3]/div/div/div/div/div[5]/div[2]/a/span[1]/span').text

try:
    bio = profile.find_element(By.XPATH,
'//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div/div/div[3]/div/div/div/div/div[3]/div/div').text
except:
    bio = "-"

try:
    verification_badge_element = driver.find_element(By.CSS_SELECTOR,
"[data-testid='icon-verified']")
    v = 'verified'
except:
    try:
        verification_badge_element = driver.find_element(By.XPATH,
"//svg[@data-testid='icon-verified']")
        v = 'verified'
    except:
        v = 'Non verified'

try:
    joined = profile1.find_element(By.CSS_SELECTOR, "[data-
testid='UserJoinDate']").text
except:
    joined = "-"

try:
    birthdate = profile1.find_element(By.CSS_SELECTOR, "[data-
testid='UserBirthdate']").text
except:
    birthdate = "-"

try:
    link = profile1.find_element(By.CSS_SELECTOR, "[data-
testid='UserUrl']").text
except:
    link = "-"

try:
    categories = profile1.find_element(By.CSS_SELECTOR, "[data-
testid='UserProfessionalCategory']").text
except:

```

```

categories = "-"

try:
    location = profile1.find_element(By.CSS_SELECTOR, "[data-
testid='UserLocation']").text
except:
    location = "-"

try:
    profile_pic = profile.find_element(By.XPATH,
"/html/body/div[1]/div/div/div[2]/main/div/div/div/div/div/div[3]/
div/div/div/div/div[1]/div[1]/div[2]/div/div[2]/div/a/div[3]/div/
div[2]/div/img")
    image_url = profile_pic.get_attribute('src')
except:
    profile_pic = "-"

try:
    background_pic = profile.find_element(By.XPATH,
"/html/body/div[1]/div/div/div[2]/main/div/div/div/div/div/div[3]/
div/div/div/a/div/div[2]/div/img")
    image_url1 = background_pic.get_attribute('src')
except:
    background_pic = "-"

user_dict = {
    'date' : current_date,
    'time' : current_time,
    'username': username,
    'Profile_name': Profile_name,
    'account_status' : v,
    'posts_number': posts_number,
    'Followers': Followers,
    'Following': Following,
    'bio': bio,
    'joined' : joined,
    'birthdate' : birthdate,
    'link' : link,
    'categories' : categories,
    'location' : location,
    'profile_pic' : image_url,
    'background_pic' : image_url1

}

# Add the dictionary to a list

```

```

user_list.append(user_dict)
user_list

[{'date': '2023-08-11',
  'time': '00:34:52',
  'username': '@elonmusk',
  'Profile_name': 'Elon Musk',
  'account_status': 'verified',
  'posts_number': '29.2K posts',
  'Followers': '408',
  'Following': '152.5M',
  'bio': 'Blades of Glory',
  'joined': 'Joined June 2009',
  'birthdate': '-',
  'link': '-',
  'categories': '-',
  'location': 'XÐ',
  'profile_pic':
'https://pbs.twimg.com/profile_images/1509808420030844929/ifEhyqHM_200x200.jpg',
  'background_pic':
'https://pbs.twimg.com/profile_banners/3429950987/1666617954/1080x360'
}]

```

Ces lignes de code permettent de fusionner les nouvelles données extraites avec les données existantes, puis de sauvegarder le tout dans un fichier CSV. Tout d'abord, les nouvelles données (stockées dans le dictionnaire "user_dict") sont ajoutées aux données existantes (dans "existing_data") en utilisant la fonction "append". Ensuite, le tableau combiné est enregistré dans le fichier CSV spécifié par "csv_file_path" en utilisant "to_csv", et finalement, le tableau mis à jour est affiché de manière formatée dans la console à l'aide de "tabulate". Cela permet de stocker et d'afficher les informations collectées de manière ordonnée.

```

combined_data = existing_data.append(user_dict, ignore_index=True)
combined_data.to_csv(csv_file_path, index=False)
print(tabulate(combined_data, headers='keys', tablefmt='fancy_grid'))

```

	date	time	username	Profile_name	
account_status		posts_number	Followers	Following	bio
joined		birthdate	link	categories	location
profile_pic					
background_pic					

0	2023-08-10	00:00:49	@elonmusk	Elon Musk	verified
29.2K posts		408	152.4M	Blades of Glory	
Joined June 2009	-	-	-	xD	
https://pbs.twimg.com/profile_images/1683325380441128960/yRsRRjG0_200x200.jpg					
https://pbs.twimg.com/profile_banners/44196397/1690621312/1080x360					
1	2023-08-10	23:31:36	@elonmusk	Elon Musk	verified
29.2K posts		408	152.5M	Blades of Glory	
Joined June 2009	-	-	-	xD	
https://pbs.twimg.com/profile_images/1666063330010710016/MwvfI8KZ_200x200.jpg					
https://pbs.twimg.com/profile_banners/158478342/1680545848/1080x360					
2	2023-08-11	00:34:52	@elonmusk	Elon Musk	verified
29.2K posts		408	152.5M	Blades of Glory	
Joined June 2009	-	-	-	xD	
https://pbs.twimg.com/profile_images/1509808420030844929/ifEhyqHM_200x200.jpg					
https://pbs.twimg.com/profile_banners/3429950987/1666617954/1080x360					

Ce fragment de code définit le chemin vers un fichier CSV pour stocker les tweets extraits. Il vérifie d'abord si le fichier CSV existe déjà. S'il existe, il lit les données existantes à partir du fichier CSV. Sinon, il crée un DataFrame vide pour stocker les données. Cela permet de préparer l'ajout des nouveaux tweets extraits aux données existantes ou de commencer avec un DataFrame vide si c'est la première fois que les données sont collectées.

```

tweets_csv = rf'C:\Users\LENOVO\Desktop\Test_DE\Section_1\Tweets\
{username_input}_tweets.csv'

# Check if the CSV file already exists
if os.path.exists(tweets_csv):
    existing_data = pd.read_csv(tweets_csv)
else:
    existing_data = pd.DataFrame()

driver.refresh()

```

Charger plusieurs pages pour extraire leurs tweets

Ce code effectue un défilement vers le bas de la page à plusieurs reprises pour charger davantage de tweets. Il exécute une boucle qui réduit la variable scrolls à chaque itération, puis utilise la méthode `execute_script` de Selenium pour faire défiler la page vers le bas en utilisant JavaScript.

Méthode 1

```

scrolls = 7
while True:
    scrolls -= 1
    driver.execute_script("window.scrollTo(0,
document.body.scrollHeight)")
    time.sleep(3)
    if scrolls < 0:
        break

```

Méthode 2

```

driver.execute_script("window.scrollTo(0,
document.body.scrollHeight)")

```

Méthode 3

```

# Number of tweets you want to extract
desired_tweet_count = 20
current_tweet_count = 0

while current_tweet_count < desired_tweet_count:
    tweets = driver.find_elements(By.XPATH,
'//div[@data-testid]//article[@data-testid="tweet"]')
    current_tweet_count += len(tweets)

    if current_tweet_count >= desired_tweet_count:
        break # Stop if desired tweet count is reached

# Scroll down to load more tweets

```



```

driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
time.sleep(3) # Adjust the scrolling pause time as needed

```

Extraire les tweets et leurs données

Ce code extrait les données des tweets affichés sur la page actuelle de Twitter. Il parcourt la liste des éléments tweets et extrait les informations suivantes pour chaque tweet : la date et l'heure du tweet, le texte du tweet, le nombre de retweets, de likes et de réponses, ainsi que les informations sur les médias associés (images ou vidéos). Il tente d'extraire le contenu média en vérifiant d'abord s'il s'agit d'une image ou d'une vidéo, puis récupère le lien ou la source correspondant. Les informations extraites sont stockées dans une liste de dictionnaires appelée `tweets_list`.

```

tweets_list = []

#wait = WebDriverWait(driver, 10)
tweets = driver.find_elements(By.XPATH,
'//div[@data-testid="//article[@data-testid="tweet"]']')
#tweets = wait.until(tweets_loaded)

# Get the current date and time
current_datetime = datetime.datetime.now()
current_date = current_datetime.strftime("%Y-%m-%d") # Format: YYYY-MM-DD
current_time = current_datetime.strftime("%H:%M:%S") # Format: HH:MM:SS

for item in tweets[0:]:
    try:
        date = item.find_element(By.XPATH, '//*[@data-testid="tweetTime"]').text + ' ' + 'ago'
    except:
        date = '[empty]'
        #print(date)

    try:
        text = item.find_element(By.XPATH, '//*[@data-testid="tweetText"]').text
    except:
        text = '[empty]'

    retweet_count = item.find_element(By.CSS_SELECTOR, 'div[data-testid="retweet"]').text
    like_count = item.find_element(By.CSS_SELECTOR, 'div[data-testid="like"]').text
    reply_count = item.find_element(By.CSS_SELECTOR, 'div[data-testid="reply"]').text

```

```

        views_count = item.find_element(By.XPATH, "//span[@data-
testid='app-text-transition-container']//span[@class='css-901oao css-
16my406 r-poiln3 r-n6v787 r-1cwl3u0 r-1k6nrdrp r-1e081e0 r-
qvutc0']").text

media_info_list = []
media_type = " "

    try:
        # Try to find a div element with data-testid="tweetPhoto"
for pictures
        if item.find_elements(By.XPATH, "//div[@data-
testid='cellInnerDiv']//div[@class='css-1dbjc4n r-1adg3ll r-
1ny4l3l']//div[@class='css-1dbjc4n r-eqz5dr r-16y2uox
r-1wbh5a2']//img[@class='css-9pa8cd']"):
            media_src = item.find_elements(By.XPATH, "//div[@data-
testid='cellInnerDiv']//div[@class='css-1dbjc4n r-1adg3ll r-
1ny4l3l']//div[@class='css-1dbjc4n r-eqz5dr r-16y2uox
r-1wbh5a2']//img[@class='css-9pa8cd']")[0].get_attribute("src")
            media_type = "picture"
            media_dict = {
                'media_type': media_type,
                #'media_url': media_src
            }
            media_info_list.append(media_dict)
    except:
        # If the picture div element is not found, try to find a
div element with data-testid="videoComponent" for videos
        if item.find_elements(By.XPATH, "//div[@data-
testid='cellInnerDiv']//div[@class='css-1dbjc4n']//div[@class='css-
1dbjc4n r-eqz5dr r-16y2uox r-1wbh5a2']//video[@aria-label='Embedded
video']"):
            media_src = item.find_elements(By.XPATH,
"//div[@data-testid='cellInnerDiv']//div[@class='css-1dbjc4n']//div[@c
lass='css-1dbjc4n r-eqz5dr r-16y2uox r-1wbh5a2']//video[@aria-
label='Embedded video']")[0].get_attribute("src")
            media_type = "video"
            media_dict = {
                'media_type': media_type,
                #'media_url': media_src
            }
            media_info_list.append(media_dict)

# Create the main dictionary for this tweet

```

```

        tweets_dict = {
            'actual_date': current_date,
            'actual_time': current_time,
            'datetime of post': date,
            'text': text,
            'media_info_list': media_info_list, # Store media
information as a list of dictionaries
            'retweet_count': retweet_count,
            'like_count': like_count,
            'reply_count': reply_count,
            # 'views_count': views_count
        }

        # Add the dictionary to the list
        tweets_list.append(tweets_dict)

tweets_list

[{'actual_date': '2023-08-11',
  'actual_time': '00:36:25',
  'datetime of post': '3h ago',
  'text': 'This platform is by far the best way to reach world
leaders, CEOs and the cognoscenti in general',
  'media_info_list': [],
  'retweet_count': '5,700',
  'like_count': '44.2K',
  'reply_count': '5,612'},
 {'actual_date': '2023-08-11',
  'actual_time': '00:36:25',
  'datetime of post': '23h ago',
  'text': 'Great product engineering & design sessions today with
the X team!\n\nFuture is looking bright.',
  'media_info_list': [],
  'retweet_count': '15.8K',
  'like_count': '181.8K',
  'reply_count': '12.8K'},
 {'actual_date': '2023-08-11',
  'actual_time': '00:37:12',
  'datetime of post': '3h ago',
  'text': 'This platform is by far the best way to reach world
leaders, CEOs and the cognoscenti in general',
  'media_info_list': [],
  'retweet_count': '5,705',
  'like_count': '44.2K',
  'reply_count': '5,614'},
 {'actual_date': '2023-08-11',
  'actual_time': '00:37:12',
  'datetime of post': '23h ago',
  'text': 'Great product engineering & design sessions today with
the X team!\n\nFuture is looking bright.'}]

```

```

    'media_info_list': [],
    'retweet_count': '15.8K',
    'like_count': '181.8K',
    'reply_count': '12.8K'},
{'actual_date': '2023-08-11',
 'actual_time': '00:37:12',
 'datetime of post': 'Aug 9 ago',
 'text': 'Highly recommend doing this!',
 'media_info_list': [],
 'retweet_count': '7,268',
 'like_count': '52.2K',
 'reply_count': '4,198'}}]

```

Ce code ajoute les données extraites des tweets à l'ensemble de données existant (`existing_data`). Ensuite, il sauvegarde les données combinées dans un fichier CSV spécifié par la variable `tweets_csv`. Enfin, il imprime le tableau mis à jour dans la console en utilisant la fonction `tabulate` pour afficher les données de manière formatée.

```

combined_data = existing_data.append(tweets_list, ignore_index=True)
combined_data.to_csv(tweets_csv, index=False)
print(tabulate(combined_data, headers='keys', tablefmt='fancy_grid'))

```