INSTABOT

Code & notes

INSTABOT-1

- We will import all the Important libraries for the project.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException, Element
from bs4 import BeautifulSoup as bs
import pandas as pd
import numpy as np
import requests, time
```

- We will mention the URL of the website to access.

```
#accessing website
url = "https://www.instagram.com/"
```

- we will establish the driver perform all the selenium operation on the website.

```
driver = webdriver.Chrome()
driver.get(url)
driver.maximize_window()
wait = WebDriverWait(driver,10)
```

- Mentioning the username and password

```
username = "username"
password = "password"
```

- We will perform the First operation of Login .

1. **Log In**

```
 def login(username,password):
    try:
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/
        #Going through popups
        pop = wait.until(EC.presence_of_element_located((By.XPA
        pop.click()
        #closing the notification popup
        notify = wait.until(EC.presence_of_element_located((By.
        notify.click()
    except TimeoutException:
        print("Something is Wrong")
```

```
login(username,password)
```

2. **<u>This Search option Script will help you get all the profile name with specific keywords.</u>**

```
def search(s):
    try:
        #opening search box for search operation
        wait.until(EC.presence_of_element_located((By.PARTIAL_L
        search = wait.until(EC.presence_of_element_located((By.
        search.send_keys(s)
        #extracting usernames
        wait.until(EC.presence_of_element_located((By.XPATH,"./
        handel_name = driver.find_elements(By.XPATH,".//div[(@c
        names = []
        for i in handel_name:
            if i.text[0] != "#":
                names.append(i.text.split('\n')[0])
        time.sleep(3)
        driver.refresh()
        return names
    except TimeoutException:
        print("Something is Wrong")
```

- printing the names gathered from the search function operation.

```
profile_name = search('Keyword')
for i in profile_name:
    print(i)

print(f"Total Number of profile : {len(profile_name)}")
```

3. **<u>Searching and Opening a Profile.</u>**

- here we will open the profile with the manually inputting the user name to access there information and continue the operation to next step.

```
def search_open(s):
    try:
        #locating Search Box and Sending text for the search
        wait.until(EC.presence_of_element_located((By.PARTIAL_LI
        search = wait.until(EC.presence_of_element_located((By.X
        search.send_keys(s)

        #Checking the result
        result = wait.until(EC.presence_of_element_located((By.X
        result.click()

        time.sleep(5)
    except TimeoutException:
        print('Something Wrong with the search and open OP')
```

```
search_open('Profile name')
```

4. **Follow/Unfollow given handle.**

- here we will perform the basic operation following & unfollowing the profile using the script.
- this script is useful if we want to follow/unfollow multiple profile on Instagram.

```
def follow():
    try:

        #Locating follow button
        follow_btn = wait.until(EC.presence_of_element_located(

        #checking for the handle follow followed by a message fo
        if follow_btn.text == 'Follow':
            follow_btn.click()
```

```
            time.sleep(3)
        else:
            print("Already Following")
    except TimeoutException:
        print("something Wrong with the Follow OP")
```

```
def unfollow():
    try:
        follow_btn = wait.until(EC.presence_of_element_located(
        if follow_btn.text != 'Follow':
            follow_btn.click()
            time.sleep(3)
            #going Through the popup for conformation to unfollo
            pop_up = wait.until(EC.presence_of_element_located(
            pop_up.click()
            time.sleep(3)
        else:
            print("Already Unfollowed")
    except TimeoutException:
            print("Something Wrong with the Unfolow OP")
```

```
follow()
```

```
unfollow()
```

5. **Liking / Un-liking the posts.**

- here we can set the number of post we want to like/ un-follow and let the script perform the action on our behalf .
- By default the value is set to like/ un-follow 30 post at this time.

```
def like_post():
    try:
        driver.execute_script('window.scrollTo(0,document.body.s
```

```
        time.sleep(2)
        driver.execute_script('window.scrollTo(0,document.body.s
        time.sleep(2)
        post = driver.find_elements(By.CLASS_NAME,"_aagu")
        for i in range(30):
            time.sleep(2)
            post[i].click()

            #locating the link button/ Unlike button
            wait.until(EC.presence_of_element_located((By.XPATH,
            like_btn = driver.find_element(By.XPATH,"//section/s
            status = bs(like_btn.get_attribute('innerHTML'),"htm
            if  status == 'Like':
                like_btn.click()
                like_btn.click()
                time.sleep(2)
                print("Post",i," is Liked")
            else:
                print("You Have already Liked the post",i+1)
                time.sleep(1)
                #locating the 'X' button to close the post.
            wait.until(EC.presence_of_element_located((By.XPATH,
    except TimeoutException:
        print("Something Is wrong with Like OP")
```

```
def unlike_post():
    try:
        driver.execute_script("window.scrollTo(0, document.body
        time.sleep(3)
        driver.execute_script("window.scrollTo(0, document.body
        time.sleep(2)
        post = driver.find_elements(By.CLASS_NAME,"_aagu")

        for i in range(30):
            time.sleep(2)
```

```
            post[i].click()


            #locating the Like/Dislike Button For Further OP
            wait.until(EC.presence_of_element_located((By.XPATH,
            like_btn = driver.find_element(By.XPATH,"//section/s
            status = bs(like_btn.get_attribute('innerHTML'),"htm

            if status == 'Unlike':
                like_btn.click()
                time.sleep(2)
                print("Post",i," is UnLiked")

            else:
                print("You Have Already UnLiked The Post",i+1)
                time.sleep(2)
            #locating the close button
            wait.until(EC.presence_of_element_located((By.XPATH,
    except TimeoutException:
        print("Something Wrong with the Dislike OP")
```

```
like_post()
```

```
unlike_post()
```

6. **Extracting list of followers.**

- Here we will extract the names of followers from the followers list of specified user.
- The default value of names extracted will be 500 which can be manually increased/ decreased in the function.

```
def extract_followers():
    try:
```

```
        #locating the followers button for getting the followers

        followers_btn = wait.until(EC.element_to_be_clickable((B
        driver.execute_script("arguments[0].scrollIntoView(true)
        time.sleep(1)
        followers_btn.click()


        #followers list frame
        try:
            frame = wait.until(EC.presence_of_element_located((B
        except StaleElementReferenceException:
            print("StaleElementReferenceException: Re-locating t
            frame = wait.until(EC.presence_of_element_located((B
        for i in range(50):
            driver.execute_script("arguments[0].scrollTop = argu
            time.sleep(2)

        names = []

        #extracting the Names of the followers
        followers_name = wait.until(EC.presence_of_all_elements_

        for i in followers_name[:500]:
            names.append(i.text.split('\n')[0])
        return names
    except TimeoutException:
        print("Something Wrong with the OP")
    except Exception as e:
        print(f"An error occurred: {e}")
```

```
users = extract_followers()
```

- Here we will print the names using the for loop to get it in the ordered form for visual check.

```
ind = 1
for usernames in users:
    print(ind,usernames)
    ind +=1
```

7. **Checking for the profile that follows back**

- Here we will check whether the users we are following are following us back or not.

```
def follow_check():
    try:
        following_btn = wait.until(EC.element_to_be_clickable((B
        driver.execute_script("arguments[0].scrollIntoView(true
        time.sleep(1)
        following_btn.click()


        #followers list frame
        try:
            follow_frame = wait.until(EC.presence_of_element_loc
        except StaleElementReferenceException:
            print("StaleElementReferenceException: Re-locating t
            follow_frame = wait.until(EC.presence_of_element_loc
        for i in range(50):
            driver.execute_script("arguments[0].scrollTop = argu
            time.sleep(1)

        names = []

        #extracting the Names of the followers
        following_name = wait.until(EC.presence_of_all_elements_

        for i in following_name:
            names.append(i.text.split('\n')[0])
```

```
        return names
    except TimeoutException:
        print("Something Wrong with the OP")
    except Exception as e:
        print(f"An error occurred: {e}")
```

- Here is the example code of checking the follow_back function in real time.

```
# Extracting followers using Extract_Followers() function
followers_of_sumit = extract_followers()
#casting into set
followers_of_sumit = set(followers_of_sumit)
```

```
#now finding all user followed by me for that I'll use search_op
#followed by me
search_open('exps_sumit')
followed_by_me = follow_check()
followed_by_me = set(followed_by_me)
```

```
#taking intersection so s1 contains only that user who followed
s1=(followers_of_foodtalkindia).intersection(followed_by_me)
if len(s1) == 0:
    print('No such users found')
else:
    #now extracting my followers
    my_follower = follow_check()
    my_follower = set(my_follower)
    #taking intersection with s1, so s2 contains only users that
    s2=s1.intersection(my_follower)
    if len(s2) == 0:
        print('No such users found')
    else:
```

```
        for user in s2:
            print(user)
```

8. **Checking the story of Instagram profile.**

- here we will check the story from Instagram profile using the script

```
def check_story():
    try:
        #locating the profile Pic to view the story
        profile_pic = wait.until(EC.element_to_be_clickable((By
        driver.execute_script("arguments[0].scrollIntoView(true
        time.sleep(1)
        #checking the profile pic height to check whether story
        pic_height = driver.find_element(By.XPATH,".//canvas[@cl

        if int(pic_height) == 229:
            print("Already Seen the Story")
        else:
            print("Viewing the story")
            profile_pic.click()
    except:
        print("No story is available to view")
```

```
#for search and open 'coding.ninjas' instagram handle
search_open('coding.ninjas')
```

```
follow()
```

```
check_story()
```

INSTABOT - 2

.

- Importing all the necessary library for the project.

```
#importing the libraries
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.common.exceptions import NoSuchElementException, T
from selenium.webdriver.support import expected_conditions as EC
import time
from datetime import datetime,timedelta
from collections import OrderedDict
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

- Mentioning the website if INSTABOT 2 is being used separately and not with continuation of INSTABOT 1

```
#accessing website
url = "https://www.instagram.com/"
```

.

```
#creating the driver for automation process
driver = webdriver.Chrome()
driver.maximize_window()
driver.get(url)
wait = WebDriverWait(driver,10)
```

```
#mentioning the username and password
username = "username"
```

```
password = "password"
```

1. **Log In**

```
def login(username,password):
    try:
        time.sleep(1)
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/d
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/d
        driver.find_element(By.XPATH,"//*[@id='loginForm']/div/d
        #Going through popups
        pop = wait.until(EC.presence_of_element_located((By.XPAT
        pop.click()
        #closing the notification popup
        notify = wait.until(EC.presence_of_element_located((By.X
        notify.click()
    except TimeoutException:
        print("Something is Wrong")
```

```
login(username,password)
```

2. **Getting profile name from search .**

```
def search(s):
    try:
        #opening search box for search operation
        time.sleep(3)
        wait.until(EC.presence_of_element_located((By.PARTIAL_LI
        search = wait.until(EC.presence_of_element_located((By.X
        search.send_keys(s)
        #extracting usernames
        wait.until(EC.presence_of_element_located((By.XPATH,"./
        handel_name = driver.find_elements(By.XPATH,".//div[(@cl
        names = []
        for i in handel_name:
```

```
            if i.text[0] != "#":
                names.append(i.text.split('\n')[0])
        time.sleep(3)
        driver.refresh()
        return names
    except TimeoutException:
        print("Something is Wrong")
```

```
#searching for profile with food in there name
search('keyword')
```

3. **Searching and opening the profile.**

- we will use this function later inside other function for there process.

```
def search_open(s):
    try:
        #locating Search Box and Sending text for the search
        wait.until(EC.presence_of_element_located((By.PARTIAL_LI
        search = wait.until(EC.presence_of_element_located((By.X
        search.send_keys(s)

        #Checking the result
        result = wait.until(EC.presence_of_element_located((By.X
        result.click()

        time.sleep(5)
    except TimeoutException:
        print('Something Wrong with the search and open OP')
```

- Creating the list of top n profile on whom we will perform the operation.
- example of a list here is given with top 10 profile with food in there name.

```
#first 10 profile from the search
handles = ['foodtalkindia','foodie_flatlay','foodelhi','fooodie_
```

4. **Creating function for getting top 5 handle which have highest number of followers from the list of handles.**

```
#function for getting top 5 handle which have highest number of
def top_handle(lst):
    try:
        #creating a Dictionary that will contain the profile nar
        dct = {}
        for i in lst:
            time.sleep(2)
            search_open(i)

           #getting the follow count
            followers_val_count = wait.until(EC.presence_of_elem
            dct[i] = int(followers_val_count.get_attribute('titl
        #top 5 most followed followers being sorted here
        sorted_handles = OrderedDict(sorted(dct.items(), key=lar

        return sorted_handles
    except TimeoutException:
        print("Timeout, TRY AGAIN")
```

```
#storing the names of handle in a variable
top_handle = top_handle(handles)
```

```
print('Top 5 Instagram handles which have the highest number of
for user_name, follower_count in top_handle.items():
    print(f"{user_name} -> {follower_count}")
```

5. **Finding the number of posts these handles have done in the previous 3 days.**

```
#This function will return the count of post posted by the user
def count_of_post(day):
    try:
        #locating the first Post
        driver.execute_script('window.scrollTo(0,document.body.s
        time.sleep(2)
        driver.execute_script('window.scrollTo(0,document.body.s
        time.sleep(2)
        post = driver.find_element(By.CLASS_NAME,"_aagu")
        post.click()
        count = 0

        while True:
            #Getting info of the posted media
            info = driver.find_element(By.CLASS_NAME,"x1p4m5qa")

            #getting the date in string format
            date_string = info.get_attribute('datetime')

            #converting the date into acceptable format for anay

            date = datetime.strptime(date_string[0:19],"%Y-%m-%d

            #calculating the Diffrence in posted date and curren

            if (datetime.now()-date).days>day:
                break
            else:
                count += 1
            #locating next button
            time.sleep(1)
            next_button = wait.until(EC.presence_of_element_loca
```

```
            next_button.click()
        #closing the post
        time.sleep(1)
        close_post = driver.find_element(By.XPATH,".//div[@class
        close_post.click()
        return count
    except TimeoutException:
        print("Timeout, TRY AGAIN...")
```

```
total_post_count = {}
for i in top_handle :
    time.sleep(3)
    search_open(i)
    count = count_of_post(3)
    total_post_count[i] = count
```

```
print('Number of posts these handles have done in the previous 3
for i in total_post_count:
    print(i,total_post_count[i])
```

6. **Depicting gathered information using a suitable graph**

```
# using total_post_count to get username and number of post
x = total_post_count.keys()
y = total_post_count.values()
plt.bar(x,y,color='grey',alpha=0.6)
plt.ylabel("Number of posts in the previous 3 days",size=10)
plt.xlabel("Username",size=10)
plt.title('Most Followed user vs Number of recent posts ',fonts
plt.grid()
plt.show()
```

7. **Opening the 5 handles we obtained in the last operation and scraping the hashtag content of the first 10 posts of each handle.**

```
def extract_contents(lst):
    try:
        hashtag_list = []
        for users in lst:
            search_open(users)
            time.sleep(1)
            #scrolling the page to get the view
            driver.execute_script('window.scrollTo(0,document.bo
            time.sleep(1)
            driver.execute_script('window.scrollTo(0,document.bo
            time.sleep(1)

            #locating the post of the user
            post = wait.until(EC.presence_of_all_elements_locate
            for i in post[:10]:
                i.click()
                time.sleep(2)
                #locating all the hashtag used in the post of th
                hashtag = driver.find_elements(By.XPATH,".//a[@c
                for tagname in hashtag:
                    hashtag_list.append(tagname.text)

                #locating the close button
                time.sleep(1)
                wait.until(EC.presence_of_element_located((By.XP
        return hashtag_list
    except TimeoutException:
        print("Timeout....TRY AGAIN")
```

```
handles = list(total_post_count.keys())
hashtags = extract_contents(handles)
```

```
print(hashtags)
```

8. **Prepare a list of all words used in all the scraped posts and calculating the frequency of each word.**

```
dct = {}
words_list = []
for word in hashtags:
    dct[word] = dct.get(word,0) + 1

for i in sorted(dct, key=dct.get, reverse=True):
    words_list.append([i,dct[i]])
    print(i,dct[i])
```

9. **Creating a csv file with two columns : the word and its frequency .**

```
df = pd.DataFrame(words_list,columns = ['Word','Frequency'])
#using .to_csv inbuilt pandas function for converting dataframe
df.to_csv(r'word_frequency.csv',index = False)
df.head(10)
```

10. **checking for the hashtags that were most popular among these list.**

```
print(words_list[0][0])
```

11. **Plotting Pie Chart of the top 5 hashtags obtained and the number of times they were used by these bloggers in the scraped posts.**

```
hashtag_name = [words_list[0][0],words_list[1][0],words_list[2]
count = [words_list[0][1],words_list[1][1],words_list[2][1],word
plt.figure(figsize = (12,6))
plt.pie(count,labels = hashtag_name,autopct = '%0.2f%%',radius =
plt.title('Top 5 Hashtags used by bloggers', fontsize = 20,colo
plt.show()
```

12. **Find out the number of likes of the top 10 posts of the 5 handles obtained earlier.**

```
def get_likes(user):
    try:
        likes = []
        #using this function to search and open the profile of t
        search_open(user)
        time.sleep(1)

        #scrolling the page to get the post in the view range fo
        driver.execute_script("window.scrollTo(0,document.body.s
        time.sleep(1)
        driver.execute_script("window.scrollTo(0,document.body.s
        time.sleep(1)

        #locating the post of the user
        posts = wait.until(EC.presence_of_all_elements_located(
        for i in posts[:10]:
            i.click()
            time.sleep(1)
            try:
                like = driver.find_element(By.XPATH,".//section
                no_of_likes = (like.text).replace(",","")
                likes.append(int(no_of_likes))
                time.sleep(1)
            #if any post has there likes hidden then we will mer
            except NoSuchElementException :
                #appending the value of likes as 1 so we can get
                likes.append(int(1))
                time.sleep(1)
            except ElementClickInterceptedException as e:
                print(e)
            finally:
                #locating the close button
                time.sleep(1)
                wait.until(EC.presence_of_element_located((By.XP
                time.sleep(1)
```

```
        return likes
    except TimeoutException as e:
        print(e)
```

```
#dct will store username with list of likes
dct = {}
for user in handles :
    likes = get_likes(user)
    dct[user] = likes
    print('Number of likes of the top 10 posts of',user,':')
    for i in likes :
        print(i,'Likes')
    print()
```

13. **Calculating the average likes for a handle.**

- here we will calculate the average number of likes for a handle.

```
#This function will return a dictionary of {'username':average l
def get_avg_likes(d):
    avg_likes = {}
    for key,val in d.items():
        avg = sum(val) // 10
        avg_likes[key] = avg
    return avg_likes
```

```
avg_likes = get_avg_likes(dct)

print('Average likes for each handle are :')
for user,likes in avg_likes.items():
    print(user, ':', likes)
```

14. **Dividing the average of likes obtained from the number of followers of the handle to get the average followers : like ratio of each handle.**

```
def average_to_like_ratio(top_handle,avg_likes,user):
    follow_like = {}
    for i in range(5):
        follow_like[user[i]] = top_handle[user[i]]//avg_likes[us
    return follow_like
```

```
#using the value from top_handle which contains a dictionary of
#extracing username from top_handles
top_handles_names = list(top_handle.keys())
#now calling above function
avg_fol_like_ratio = average_to_like_ratio(top_handle, avg_likes
print('Average followers:like ratio of each handle :')
for user,ratio in avg_fol_like_ratio.items():
    print(user, ':', ratio)
```

14. **Create a bar graph to depict the above obtained information.**

```
#Plotting Bar Graph for average likes of each handle
username = list(avg_likes.keys())
avg_like = list(avg_likes.values())
plt.figure(figsize=(12, 8))  # width=12 inches, height=8 inches
plt.bar(username,avg_like,color='g',alpha=0.6)
plt.ylabel("Average Likes",size=10)
plt.xlabel("Handle's Name",size=10)
plt.xticks(fontsize=7)  # Decrease font size of x-axis labels
plt.title('Average like for each handle ',fontsize=20, color='gr
plt.grid()
plt.show()
```

```
# Plotting Bar Graph for average followers:like ratio of each ha
#using average likes obtained from the number of followers of th

username = list(avg_fol_like_ratio.keys())
```

```
avg_like = list(avg_fol_like_ratio.values())
plt.figure(figsize=(12, 8))  # width=12 inches, height=8 inches
plt.bar(username,avg_like,color='Indigo',alpha=0.6)
plt.ylabel("Average Followers:Like Ratio",size=12)
plt.xlabel("Handle's Name",size=12)
plt.title('Average Followers:Like Ratio of Each Handle ',fontsiz
plt.xticks(fontsize = 15)
plt.grid()
plt.show()
```