In [1]:
```
import selenium
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions as ec
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.firefox.options import Options as FirefoxOptions
import time as t
from bs4 import *
from instascrape import Profile
import requests
import re
import pandas as pd
import base64
from datetime import datetime
from instagramy import InstagramUser
```

In [2]:
```
class FI_Data_Handling:
    def str_num_conv(self,x):
        total= 0
        num_map = {'K':1000, 'M':1000000, 'B':1000000000}
        if x.isdigit():
            total= int(x)
        else:
            if len(x) > 1:
                total = float(x[:-1]) * num_map.get(x[-1].upper(), 1)
        return int(total)


class Driver_Init:
    def initialize_driver(self,driver="firefox"):
        if driver.lower()=="firefox":
            options = FirefoxOptions()
            options.add_argument("--headless")
            options.add_argument('--no-proxy-server')
            options.add_argument('--disable-dev-shm-usage')
            options.add_argument('--disable-gpu')
            options.add_argument('log-level=3')
            driver = webdriver.Firefox(options=options)
            return driver

        else:
            chrome_options = webdriver.ChromeOptions()
            prefs = {"profile.default_content_setting_values.notifications" :2}
            chrome_options.add_experimental_option("prefs",prefs)
            chrome_options.add_argument('--no-proxy-server')
            chrome_options.add_argument('--disable-dev-shm-usage')
            chrome_options.add_argument('--disable-gpu')
            chrome_options.add_argument('log-level=3')
            chrome_options.add_argument('--headless')
            driver = webdriver.Chrome(options=chrome_options)
            return driver

    def get_sessionID(self):
        try:
            link = 'https://www.instagram.com/accounts/login/'
            login_url = 'https://www.instagram.com/accounts/login/ajax/'
            pas=base64.b64decode("USER_ENCODED").decode("utf-8")
            user=base64.b64decode("PASSWORD_ENCODED").decode("utf-8")
            time = int(datetime.now().timestamp())

            payload = {
                'username': user,
                'enc_password': f'#PWD_INSTAGRAM_BROWSER:0:{time}:{pas}',
                'queryParams': {},
                'optIntoOneTap': 'false'
            }

            with requests.Session() as s:
                r = s.get(link)
                csrf = re.findall(r"csrf_token\":\"(.*?)\"",r.text)[0]
                r = s.post(login_url,data=payload,headers={
                    "User-Agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.
                    "X-Requested-With": "XMLHttpRequest",
                    "Referer": "https://www.instagram.com/accounts/login/",
                    "x-csrftoken":csrf
                })

            session_id=s.cookies.get_dict()["sessionid"]
            return session_id
        except Exception as e:
            print(e)
            print("Unable to get Session ID!!")
```

```
class Face_Scraping:
    def __init__(self,driver="firefox"):
        self.driver=Driver_Init().initialize_driver(driver)

    def get_page_info(self,url=None,name=None):

        if url is None:
            url="https://www.facebook.com/"+name+"/about/"


        if name is None:
            if url[-1]=="/":
                name=re.findall(r'/[^/]+/([^/]+)/',url)[0]
            else:
                name=re.findall(r'/[^/]+/([^/]+)',url)[0]

            url="https://www.facebook.com/"+name+"/about/"


        self.driver.get(url)

        if "Page not found" in self.driver.title:
            print("Sorry, This Page is Not Available!!")
            self.driver.close()

        else:
            page_info={"pg_likes":0,
                       "pg_follows":0,
                       'pg_checked': 0,
                       'pg_about': '.'
                       }

            try:
                t.sleep(1)
                soup=BeautifulSoup(self.driver.page_source,"html.parser")

                all_info=soup.find_all("div",{"class":"dati1w0a ihqw7lf3 hv4rvrfc discj3wi d2edcug0"+
                                                      " f9o22wc5 nzypyw8j ad2k81qe tr9rh885 rq0escxv l82x9zwi"+
                                                      " uo3d90p7 pw54ja7n ue3kfks5 hybvsw6c"})

                collected_info=all_info[0].find_all("div",{"class":"qzhwtbm6 knvmm38d"})
                collected_info=[x.text for x in collected_info if x.text!="About"]
                for q in collected_info:
                    if "like" in q:
                        page_info["pg_likes"]=int("".join(re.findall(r'\d+',q)))

                    elif "follow" in q:
                        page_info["pg_follows"]=int("".join(re.findall(r'\d+',q)))

                    elif "checked" in q:
                        page_info["pg_checked"]=int("".join(re.findall(r'\d+',q)))

                    elif 'About' in q:
                        page_info["pg_about"]=q[5:]

                self.driver.close()
                return page_info

            except Exception as e:
                print(e)
                self.driver.close()


    def get_page_posts(self,url=None,name=None):

        if url is None:
            url="https://www.facebook.com/pg/"+name+"/posts/"


        if name is None:
            if url[-1]=="/":
                name=re.findall(r'/[^/]+/([^/]+)/',url)[0]
            else:
                name=re.findall(r'/[^/]+/([^/]+)',url)[0]

            url="https://www.facebook.com/pg/"+name+"/posts/"


        self.driver.get(url)

        if "Page not found" in self.driver.title:
            print("Sorry, This Page is Not Available!!")
            self.driver.close()

        else:
            t.sleep(1)
            comments=[]
            reactions=[]
            shares=[]
```

```
            contents=[]
            timestamps=[]
            soup=BeautifulSoup(self.driver.page_source,"html.parser")
            all_posts=soup.find_all("div",{"class":"_427x"})

            for post in all_posts:
                try:
                    comment=post.find("a",{"class":"_3hg- _42ft"})
                    if comment is None:
                        comment=BeautifulSoup("<p>0 Comments</p>","lxml")
                except Exception:
                    comment="Not Found"
                    print(comment)

                try:
                    timestamp=post.find("abbr",{"class":re.compile(r"_5ptz")})
                except Exception:
                    timestamp="Not Found"
                    print(timestamp)

                try:
                    reaction={}
                    tot=post.find("span",{"class":"_81hb"})
                    if tot is None:
                        tot=BeautifulSoup("<p>0</p>","lxml")

                    like=post.find("a",{"class":"_1n9l","aria-label":re.compile(r'Like')})
                    if like is None:
                        like=BeautifulSoup("<a aria-label='0 Like'>0</a>","html.parser").find('a')

                    love=post.find("a",{"class":"_1n9l","aria-label":re.compile(r'Love')})
                    if love is None:
                        love=BeautifulSoup("<a aria-label='0 Love'>0</a>","html.parser").find('a')

                    sad=post.find("a",{"class":"_1n9l","aria-label":re.compile(r'Sad')})
                    if sad is None:
                        sad=BeautifulSoup("<a aria-label='0 Sad'>0</a>","html.parser").find('a')

                    angry=post.find("a",{"class":"_1n9l","aria-label":re.compile(r'Angry')})
                    if angry is None:
                        angry=BeautifulSoup("<a aria-label='0 Angry'>0</a>","html.parser").find('a')

                    haha=post.find("a",{"class":"_1n9l","aria-label":re.compile(r'Haha')})
                    if haha is None:
                        haha=BeautifulSoup("<a aria-label='0 Haha'>0</a>","html.parser").find('a')

                    fi_h=FI_Data_Handling()
                    reaction["tot"]=fi_h.str_num_conv(tot.text)
                    reaction["like"]=fi_h.str_num_conv(like["aria-label"].split()[0])
                    reaction["love"]=fi_h.str_num_conv(love["aria-label"].split()[0])
                    reaction["sad"]=fi_h.str_num_conv(sad["aria-label"].split()[0])
                    reaction["haha"]=fi_h.str_num_conv(haha["aria-label"].split()[0])
                    reaction["angry"]=fi_h.str_num_conv(angry["aria-label"].split()[0])
                    remaining_reacts=fi_h.str_num_conv(tot.text)-(fi_h.str_num_conv(like["aria-label"].split()[0]
                                                                  fi_h.str_num_conv(love["aria-label"].split()[0]
                                                                  fi_h.str_num_conv(sad["aria-label"].split()[0]]
                                                                  fi_h.str_num_conv(angry["aria-label"].split()[0
                                                                  fi_h.str_num_conv(haha["aria-label"].split()[0]
                                                                      )

                    reaction["r_reacts"]=abs(remaining_reacts)



                except Exception as e:
                    print(e)

                try:
                    share=post.find("span",{"class":"_355t _4vn2"})
                    if share is None:
                        share=BeautifulSoup("<p>0 Shares</p>","lxml")
                except Exception:
                    share="Not Found"
                    print(share)

                try:
                    content=post.find("div",{"class":"_5pbx userContent _3576"})
                    if content is None:
                        content=post.find("div",{"class":"_5_jv _58jw"})
                        if content is None:
                            content=BeautifulSoup("<p>No Content</p>","lxml")
                except Exception:
                    content="Not Found"
                    print(content)

                timestamps.append(timestamp["data-utime"])
                contents.append(content.text)
                comments.append(fi_h.str_num_conv(comment.text.split()[0]))
                shares.append(fi_h.str_num_conv(share.text.split()[0]))
```

```
                reactions.append(reaction)
            df=pd.DataFrame({
                                "timestamp":timestamps,
                                "upload_date":[datetime.fromtimestamp(int(x)) for x in timestamps],
                                "Content":contents,
                                "Num_Comments":comments,
                                "Num_Shares":shares,
                                "Total_Reacts":[x["tot"] for x in reactions],
                                "Likes":[x["like"] for x in reactions],
                                "Sad":[x["sad"] for x in reactions],
                                "Angry":[x["angry"] for x in reactions],
                                "Love":[x["love"] for x in reactions],
                                "Haha":[x["haha"] for x in reactions],
                                "Remaining_reacts":[x["r_reacts"] for x in reactions],
                               })

            df.drop_duplicates(subset="Content",inplace=True,keep="first")

            self.driver.close()

            return df

class Insta_Scraping:

    def __init__(self):
        self.sess_id=Driver_Init().get_sessionID()

    def get_page_posts_info(self,url):
        try:
            session_id=self.sess_id
            profile_page = Profile(url)
            headers = {
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
            "cookie":f'sessionid={session_id};'
            }
            profile_page.scrape(headers=headers)
            page_info=pd.DataFrame(profile_page.to_dict(),index=[x for x in range(1)])
            recent_posts =profile_page.get_recent_posts()
            posts_data = [post.to_dict() for post in recent_posts]
            recent_posts_df = pd.DataFrame(posts_data)

            return page_info,recent_posts_df

        except Exception as e:
            print(e)
            print("Unable To Get Insta Profile Info!!!")


    def insta_2_scraper(self,url):
        try:
            session_id=self.sess_id
            user = InstagramUser(url, sessionid=session_id)
            page_info={}
            page_info["number_of_followers"]=user.number_of_followers
            page_info["number_of_followings"]=user.number_of_followings
            page_info["number_of_posts"]=user.number_of_posts
            page_info["profile_picture_url"]=user.profile_picture_url
            page_info["is_verified"]=user.is_verified
            page_info["other_info"]=user.other_info
            page_info["username"]=user.username
            page_info["fullname"]=user.fullname
            page_info["is_private"]=user.is_private
            page_info["is_joined_recently"]=user.is_joined_recently
            page_info["biography"]=user.biography
            page_info["website"]=user.website
            page_info_df=pd.DataFrame(page_info,index=[x for x in range(1)])

            likes=[]
            comments=[]
            upload_date=[]
            caption=[]
            timestamp=[]
            display_url=[]
            is_video=[]
            short_code=[]
            post_url=[]
            location=[]
            for post in user.posts:
                likes.append(post.likes)
                comments.append(post.comments)
                upload_date.append(post.taken_at_timestamp)
                caption.append(post.caption)
                timestamp.append(post.timestamp)
                display_url.append(post.display_url)
                is_video.append(post.is_video)
                short_code.append(post.shortcode)
                post_url.append(post.post_url)
                location.append(post.location)
```

```
            posts_info_df=pd.DataFrame({
                                        "Likes":likes,
                                        "Num_Comments":comments,
                                        "upload_date":upload_date,
                                        "caption":caption,
                                        "timestamp":timestamp,
                                        "display_url":display_url,
                                        "is_video":is_video,
                                        "short_code":short_code,
                                        "post_url":post_url,
                                        "location":location
                                        })

            return page_info_df,posts_info_df

        except Exception as e:
            print(e)
```

In [3]:
```
fi=Face_Scraping()
df=fi.get_page_posts(url="https://www.facebook.com/google")
```

In [4]:
```
fi=Face_Scraping()
df1=fi.get_page_info(url="https://www.facebook.com/google")
```

In [5]:
```
ins=Insta_Scraping()
cur_info,cur_posts=ins.insta_2_scraper("google")
```

In [6]:
```
df.head()
```

Out[6]:

	timestamp	upload_date	Content	Num_Comments	Num_Shares	Total_Reacts	Likes	Sad	Angry	Love	Haha	Remaining_reacts
0	1652299670	2022-05-11 22:07:50	At #GoogleIO today, Sundar Pichai talked about...	163	38	611	553	0	0	48	0	10
1	1653001245	2022-05-20 01:00:45	Engineering lead Gordon Kuo talks about landin...	7	3	110	100	0	0	8	0	2
2	1652982521	2022-05-19 19:48:41	On Global Accessibility Awareness Day (#GAAD),...	23	11	204	168	0	0	20	0	16
3	1652975099	2022-05-19 17:44:59	Today's #GoogleDoodle celebrates Asian Pacific...	11	36	199	159	0	0	27	0	13
4	1652904126	2022-05-18 22:02:06	Our new Bay View campus features a first-of-it...	75	54	739	640	0	0	76	0	23

In [7]:
```
pd.DataFrame(df1,index=[x for x in range(1)])
```

Out[7]:

	pg_likes	pg_follows	pg_checked	pg_about
0	28180257	33029117	0	Organizing the world's information and making ...

In [8]:
```
cur_info
```

Out[8]:

	number_of_followers	number_of_followings	number_of_posts	profile_picture_url	is_verified	other_info	username	fullname	is_private	is
0	13031372	33	1798	https://instagram.fcai2-1.fna.fbcdn.net/v/t51....	True	NaN	google	Google	False	

In [9]:
```
cur_posts.head()
```

Out[9]:

	Likes	Num_Comments	upload_date	caption	timestamp	display_url	is_video	short_code	
0	7750	86	2022-05-19 21:29:55	An aerial view of the Google Bay View	1652988595	https://instagram.fcai2-1.fna.fbcdn.net/v/t51....	False	CdwFxosOdvb	https://www.instagram.com/p/CdwFx

				Campus, ...					
1	6066	101	2022-05-19 17:56:34	Google logo artwork with Stacey Milbern on the...	1652975794	https://instagram.fcai2-1.fna.fbcdn.net/v/t51....	False	CdvtXC7O3Xj	https://www.instagram.com/p/CdvtX
2	4363	87	2022-05-19 00:42:50	None	1652913770	https://instagram.fcai2-1.fna.fbcdn.net/v/t51....	True	Cdt3Bi-DzAK	https://www.instagram.com/p/Cdt3E
3	38132	309	2022-05-17 23:12:24	A building at the Google Bay View campus, phot...	1652821944	https://instagram.fcai2-2.fna.fbcdn.net/v/t51....	False	CdrH6kiudWa	https://www.instagram.com/p/CdrH6
4	21376	309	2022-05-12 04:46:36	Image of a colorful sculpture that spells out ...	1652323596	https://instagram.fcai2-1.fna.fbcdn.net/v/t51....	False	CdcRZE4LWZX	https://www.instagram.com/p/CdcRZE