This notebook contains the Python code to extract advertisements from Facebook timeline to scale up the to conduct investigations in ads practices. The entire process is automated, but it is broken down into sections (with comments) for the clear visibility and to make it developer-friendly.

General requirements

The user must have a Facebook account and some activity on it to gather relevant information and attract advertisements. Activity means:

1. Friends
2. Likes
3. Some activity on search engines. FB also captures data (from cookies) that the user types in other tabs.
4. Groups
5. Followings/Followers

If the user has a dedicated persona-trained account, more information can be captured. The notebook is completely automated to run in the background depending on the parameters. For example, you may change the number of scrolls.

Technical requirement

The user needs to be familiar with Python and selenium framework in order to run this notebook. Also, some HTML, JavaScript and XML knowledge is required.

How to run this file?

You can run this notebook on Google Colab or on your local machine (Jupyter Notebook). It doesn't require any special configuration or system configuration. But if you're running it on Google Colab, make sure to follow these instructions:

https://stackoverflow.com/questions/51046454/how-can-we-use-selenium-webdriver-in-colab-research-google-com

In [1]:
```
# Import selenium webdriver
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
import time, os, wget

# To parse URLs from anchor tags
from urllib.parse import unquote

import pandas as pd

# To clean and beautify output obtained
from lxml import html
from bs4 import BeautifulSoup
import os.path
```

Initialise chrome driver

If you don't have one, you can download it easily from https://chromedriver.chromium.org/downloads according to the version of your Chromer browser

In [2]:
```
chrome_options = webdriver.ChromeOptions()
prefs = {"profile.default_content_setting_values.notifications" : 2}
chrome_options.add_experimental_option("prefs", prefs)
```

In [ ]:
```
# Specify the path to chromedriver.exe
driver = webdriver.Chrome('C:/Users/Pranav Bapat/chromedriver.exe', chrome_options=c

# Open the webpage
driver.get("https://www.facebook.com")
```

In [ ]:
```
# Target login area. You can get this by going throught page source
username = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECT
password = WebDriverWait(driver, 10).until(EC.element_to_be_clickable((By.CSS_SELECT

# Enter your FB username and password
username.clear()
username.send_keys("your_email_here")
password.clear()
password.send_keys("your_password_here")

# Target the login button and click it
button = WebDriverWait(driver, 2).until(EC.element_to_be_clickable((By.CSS_SELECTOR,
```

And we're logged in!

Note: If you have TFA enabled on your FB account, you'll have to manually allow the request to login

Content can be:

1. Personal post
2. Shared post
3. Friend suggestions (People you may know)
4. Sponsored ads
5. "Suggestion/Suggested for you" posts from FB
6. "Events you may like" posts from FB
7. Feelings posts (feeling exhausted, feeling happy)
8. "Suggested page to follow" posts from FB
9. Posts from groups
10. Posts about someone commenting on someone's photo/posts/video
11. Someone shared a memory
12. Someone is with someone
13. Posts from pages
14. And so on

Such varied criteria could be problematic when targeting for ads only, as ads can be found in links, comments, or as a post itself either from pages or people. Moreover, each content has a different DIV structure

On FB, everything gets loaded inside DIV tags. There is the main DIV tag with the role attribute "feed". Inside this div, content have their individual DIV tags, namely "FeedUnit_0", "FeedUnit_1", "FeedUnit_n" and so on. Inside each FeedUnit, there are fixed classes given to each element. Element refers to anything that is text or clickable. We're interested in heading, user, link of the user, link of the post, and image/video if present. Here, the user refers to either an actual user or a page or an ad.

There are two ways to get the content

1. You can get individual links, images, and headings from each DIV tag according to its classes.

Advantages:

I. It's fast.

II. No need to process further when all the information is obtained.

Disadvantages:

I. There is no fixed way for FB to show you content. Thus, you have to keep the combination of the templates for all the possible content combinations, which becomes hectic, and a slight change in any of the class names become cumbersome to manage.

II. There is no control to filter the content. You have to sort it out manually.

2. You can get the entire main DIV structure and extract links, images, and headings using XML/JSON technology.

Advantages:

I. The DIV structure is much cleaner.

II. It's swift.

Disadvantages:

I. An additional step (data cleaning and formatting) is involved in getting the content.

Facebook home page (after logging in), by default, has advertisements on the right side of the page.

Observation: They randomly change as you keep scrolling, but sometimes, they don't change at all.

In [ ]:
```
# Target right side panel for the links
#anchors = driver.find_elements_by_tag_name('div.cxgpxx05 div span div div a')
#anchors = [a.get_attribute('href') for a in anchors]
```

Way 1

In [ ]:
```
# Wait 5 seconds to allow your new page to load
time.sleep(5)

driver.get("https://www.facebook.com/")
```

```
scroll = 10
for j in range(0, scroll):
    # Scroll one time entire height of the screen
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    # Wait for 5 seconds to load everything
    time.sleep(5)

anchors = driver.find_elements_by_tag_name('div.pedkr2u6 div div div.l9j0dhe7 div.l9
# Get only href attribute of anchor tag
anchors = [a.get_attribute('href') for a in anchors]

fb_page_link = driver.find_elements_by_tag_name('div.pedkr2u6 div.buofh1pr h4 a')
fb_page_link = [pl.get_attribute('href') for pl in fb_page_link]

content_class = driver.find_elements_by_tag_name('div.pedkr2u6 div.buofh1pr span a')
content_class = [cc.text for cc in content_class]

headings = driver.find_elements_by_tag_name('div.pedkr2u6 div.buofh1pr h4 a strong s
headings = [h.text for h in headings]

# All the links are preceded with an FB URL. We get rid of it by
anchors = [a for a in anchors if str(a).startswith("https://l.facebook.com/l.php?u="

print('Found ' + str(len(anchors)))
```

In [ ]:

```
links = []
word = 'utm'
# UTM paramter is used in almost all of the ads for companies to make revenue and de
# UTM paramter is useful.

for i in anchors:
    x = i[31:]
    x = unquote(x)
    if (word.find(x) == -1):
        links.append(x)

df = pd.DataFrame(list(links))
df

# We get following output of the links
```

	0
0	https://www.datacamp.com/?utm_source=fb_paid&u...
1	https://www.datacamp.com/?utm_source=fb_paid&u...
2	https://www.datacamp.com/?utm_source=fb_paid&u...
3	https://buff.ly/3yxjhxq?fbclid=IwAR2ae3kqeJ1S1...
4	https://buff.ly/3yxjhxq?fbclid=IwAR2-j8a6bj3AQ...
5	https://goo.gl/RHNL0l?fbclid=IwAR0cZ3hZyHKA0Q6...
6	https://www.youtube.com/watch?v=lHgFJEJgUrg&fb...
7	https://www.youtube.com/watch?v=lHgFJEJgUrg&fb...
8	https://mashable.com/article/scripps-spelling-...
9	https://mashable.com/article/scripps-spelling-...
10	https://timesofindia.indiatimes.com/videos/tv/...
11	https://timesofindia.indiatimes.com/videos/tv/...
12	https://www.hubspot.com/crm/e010a?utm_source=f...
13	https://www.hubspot.com/crm/e010a?utm_source=f...
14	https://www.hubspot.com/crm/e010a?utm_source=f...
15	https://retool.com/?utm_source=fb&utm_medium=p...
16	https://retool.com/?utm_source=fb&utm_medium=p...
17	https://www.enlargeyourparis.fr/balades/80-des...
18	https://www.enlargeyourparis.fr/balades/80-des...
19	https://goo.gl/GfnJGF?fbclid=IwAR0aOJt4ZXi7nGW...
20	https://www.hellofresh.nl/plans?c=FRESHSTART&u...

In [ ]:
```
headings
```

```
['',
 '',
 '',
 '',
 '',
 '',
 '',
 'Marley Spoon',
 'Viraaj Braganza',
 'The REAL Russell Peters',
 'Amar Nath Agrawal',
 'BBC News',
 'BlueMovement NL',
 'The BACK Benchers']
```

And the same way, we get content_class and fb_page_link. But as mentioned earlier, if the content jumbled up, and it will certainly be, distinguishing between the heading and links and classes becomes almost impossible. Thus, "Way 2" Is preferred.

Way 2

In [ ]:

```
# Wait 5 seconds to allow your new page to load
time.sleep(5)

driver.get("https://www.facebook.com/")

scroll = 10
for j in range(0, scroll):
    # Take the screenshot, just in case, if there is an ad
    driver.save_screenshot("screenshot"+str(j)+".png")
    # Scroll one time entire height of the screen
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    # Wait for 5 seconds to load everything
    time.sleep(5)


# Get all the contents of the main DIV tag. The main DIV tag has following classes
fb_code = driver.find_element_by_xpath("//div[@class='pedkr2u6 tn0ko95a pnx7fd3z']")
fb_code = fb_code.get_attribute('innerHTML')

# Save the contents to an HTML file for further processing
html_file = 'fb_code1.html';
check_file_exists = os.path.isfile(html_file)
if(check_file_exists):
    with open(html_file, "r+", encoding="utf-8") as f:
        data = f.read()
        f.seek(0)
        f.write(fb_code)
        f.truncate()
        f.close()
else:
    f = open("fb_code1.html", "x", encoding="utf-8")
    f.write(fb_code)
    f.close()


# Correct the encoding and beautify the HTML file
```

```
contents = ''
with open(html_file, 'r', encoding="utf-8") as f:
    contents = f.read()
    f.close()

contents = BeautifulSoup(contents, 'lxml')

with open(html_file, "r+", encoding="utf-8") as f:
    data = f.read()
    f.seek(0)
    f.write(contents.prettify())
    f.truncate()
    f.close()
```

Once the files is saved, convert it to XML format using any online XML converter. And you'll get following structure:

```
<sect2>
  <title>
    <anchor id="jsc_c_6u"/>
    <ulink url=
    "https://www.facebook.com/5min.crafts/?__cft__[0]=AZVlbvWtdCoaav5FToSnpS3pZaLhm344MHACm3sAq7pVoMlqFEw95vlP47Of2w-mUMe1o-Zb5uDMh_TCV9EsMNLJjli65_iccWVWd6sinw_vF4KnbLo0XZhOivEkSxFpQ84Fm6YWG674EmDD2noM8iA8imAiL30_nk--YvabZX3uADRwAUkyqdUtNj1ukuPnTsM&amp;__tn__=-UC%2CP-R">5-Minute Crafts </ulink>
  </title>
  <ulink url=
  "https://www.facebook.com/5min.crafts/?__cft__[0]=AZVlbvWtdCoaav5FToSnpS3pZaLhm344MHACm3sAq7pVoMlqFEw95vlP47Of2w-mUMe1o-Zb5uDMh_TCV9EsMNLJjli65_iccWVWd6sinw_vF4KnbLo0XZhOivEkSxFpQ84Fm6YWG674EmDD2noM8iA8imAiL30_nk--YvabZX3uADRwAUkyqdUtNj1ukuPnTsM&amp;__tn__=-UC%2CP-R
  ">5-Minute Crafts </ulink>
  <para><anchor id="jsc_c_6v"/>· <link linkend="">1 t d S p o l n 9 s o r e t h s d a </link>· </para>
  <sect1 id="jsc_c_6w">
    <para>Ingenious camping hacks that will change your life <inlinegraphic fileref=
    "https://static.xx.fbcdn.net/images/emoji.php/v9/tdc/1/16/1f3d5.png" width="0.1665inch" depth="0.1665inch"/> </para>
  </sect1>
  <sect1 id="jsc_c_6x">
    <para><inlinegraphic fileref=
    "https://scontent-amt2-1.xx.fbcdn.net/v/t15.5256-10/p960x960/238747703_327558725719237_6249501364118560366_n.jpg?_nc_cat=1&amp;ccb=1-5&amp;_nc_sid=ad6a45&amp;_nc_ohc=mmfIiJQSFwsAX9RS6LM&amp;_nc_ht=scontent-amt2-1.xx&amp;oh=8fb23bacd8d0e801161914abb43a737e&amp;oe=61326A6C" width="13.3335inch" depth="7.5inch"/> </para>
    <para>0:00 / 12:07 </para>
  </sect1>
  <para><anchor id="jsc_c_6z"/><inlinegraphic fileref="embedded:Image13" width="0.1874inch" depth="0.1874inch"/> </para>
  <para>7.9K 7.9K </para>
  <sect1 id="jsc_c_6y">
    <para>245 Comments </para>
  </sect1>
  <para>712 Shares </para>
  <para>Like </para>
  <para>Comment </para>
  <para>Share </para>
  <para>Suggested for you </para>
</sect2>
```

As you could see in the above image, each DIV content is within <//sect2> tag with a unique anchor tag id.

The last <//para> tag contains "Suggested for you" text, indicating that the next DIV tag is either an ad or a FB suggestion for the user. That way, we can filter which block contains ad.

Thus, instead of segregating the entire content, we just look for the DIV tags with the possibility of an ad, and neglect all the other information.

In [ ]:

In [ ]:

In [ ]:

In [ ]: