

Project 6:Message Bomber

In this project we are going to build an automatic message bomber. This project will be taking the phone number which you want to bomb and also the number of times you want to send the message. For this project we will be generating the message through **Amazon's website**. We will be generating otps from amazon's website and then sending to a number.

The first thing we need to do here is **opening google**.

For this we need to use selenium's webdriver class. So first of all we will be importing the libraries. To do this run the following code:

```
from selenium import webdriver
```

Now to open google we need to create an object of the chrome driver. Here we also need to pass the **path of the chromedriver.exe**. To do this, run the following code:

```
cd='C:\\webdrivers\\chromedriver.exe'  
driver = webdriver.Chrome(cd)
```

Where **cd** is holding the path. This will be opening up google and returning an object which I am storing as **driver**. Now whatever I want to do in the browser, I can do with this object driver.

Now when we are at google, we will search for the **Amazon SignIn Page**. Which will be taking me here:



Login

Email or mobile phone number

Continue

By continuing, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#).

► [Need help?](#)

New to Amazon?

Create your Amazon account

[Conditions of Use](#) [Privacy Notice](#) [Help](#)

© 1996-2020, Amazon.com, Inc. or its affiliates

To do this I am going to take help of the function **.get()**. This function opens any url given to it as argument(placed inside parentheses) in the browser.

Now to open Amazon's SignIn page, run the following code:

```
driver.get('https://www.amazon.in/ap/signin?_encoding=UTF8&ignoreAuthState=1&openid.assoc_handle=inflex&openid.claimed_id=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&openid.identity=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&openid.mode=checkid_setup&openid.ns=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0&openid.ns.pape=http%3A%2F%2Fspecs.openid.net%2Fextensions%2Fpape%2F1.0&openid.pape.max_auth_age=0&openid')
```

d.return_to=https%3A%2F%2Fwww.amazon.in%2F%3Fref_%3Dnav_custrec_signin&switch_account=')

The big link given above is the link of the sign-in page shown above. Now here, we will usually enter the phone number at this location.



Login

Email or mobile phone number

Continue

By continuing, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#).

► [Need help?](#)

New to Amazon?

Create your Amazon account

[Conditions of Use](#) [Privacy Notice](#) [Help](#)

© 1996-2020, Amazon.com, Inc. or its affiliates

To do this we will have to look at the code in the source code of the page with which this element is stored in this page. To do this we will be **inspecting the page**. There when we hover over the above shown text box, we will get to see how is this element actually stored in the page. And this is what the code looks like



This is stored as an input element whose **id** is **“ap_email”**. Now we need to find this element through our program. To do this run the following code.

```
phone=driver.find_element_by_id("ap_email")
```

Here we are finding the element with the id “ap_email” and storing its reference as phone_number.

Now the next task is to send the phone number to this element.

But before that we need to **take the phone number as input**. To do this, run the following code:

```
phone_number =input("Enter the phone number:")
```

Here the message given inside the parentheses of **input** will be shown over the screen and whatever phone number is given as input will be stored as **mobile_number**.

We will also take the number of times the person wants to send the number as input by running the following code.

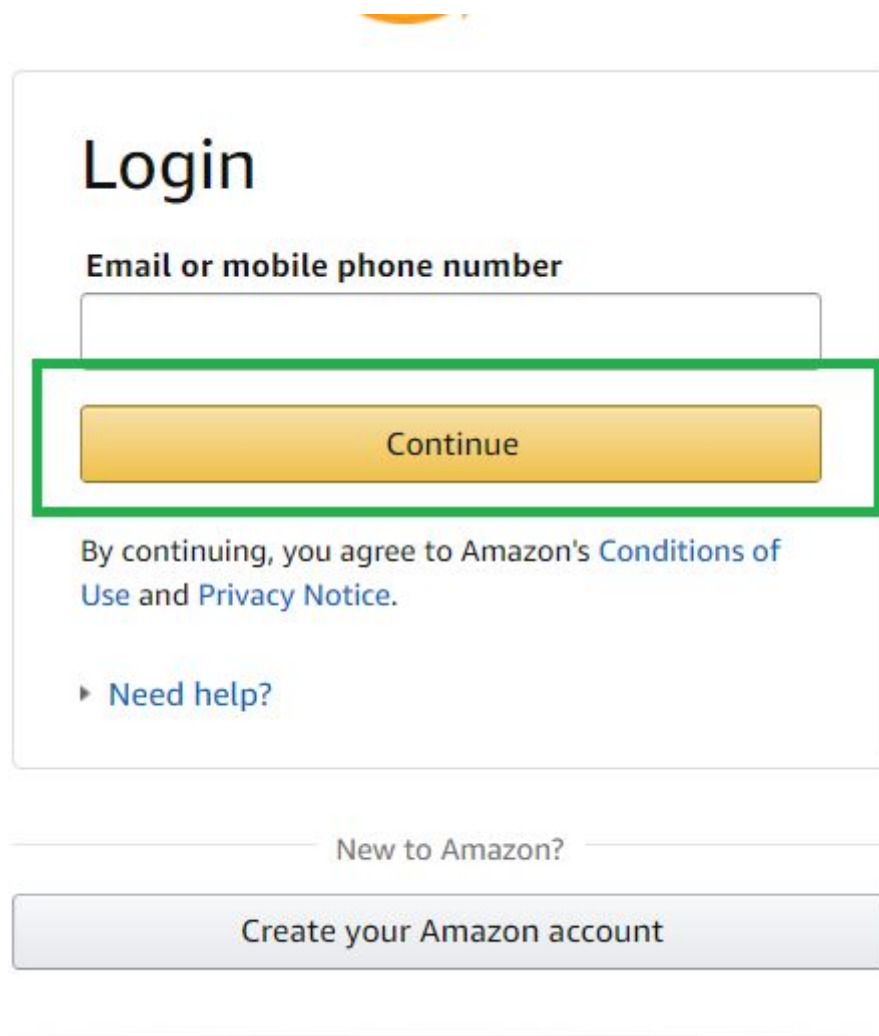
```
times=input("Enter the phone number:")
```

Now coming back to amazon, we need to enter the phone number in that text field. For this, **.send_keys()** function is going to help me, which takes an input as an argument and sends that argument to that element with respect to which it has been called. So to send the phone number, run the following code:-

```
phone.send_keys(phone_number)
```

This will send the phone number to that field.

Now once we have given the phone number, we need to press this continue button.



The image shows the Amazon login interface. At the top, there is a small orange smile logo. Below it, the word "Login" is displayed in a large, bold, black font. Underneath "Login" is the label "Email or mobile phone number" in a smaller, bold, black font. Below this label is a white text input field. A green rectangular border highlights the "Continue" button, which is a yellow-orange rectangle with the word "Continue" in black text. Below the "Continue" button, there is a line of text: "By continuing, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#)." Below this text is a link that says "► Need help?". At the bottom of the login box, there is a horizontal line with the text "New to Amazon?" in the center. Below this line is a light gray button with the text "Create your Amazon account" in black.

As you have guessed, we need to find this element in the page. On inspecting we saw it is another **input element with id as 'continue'**



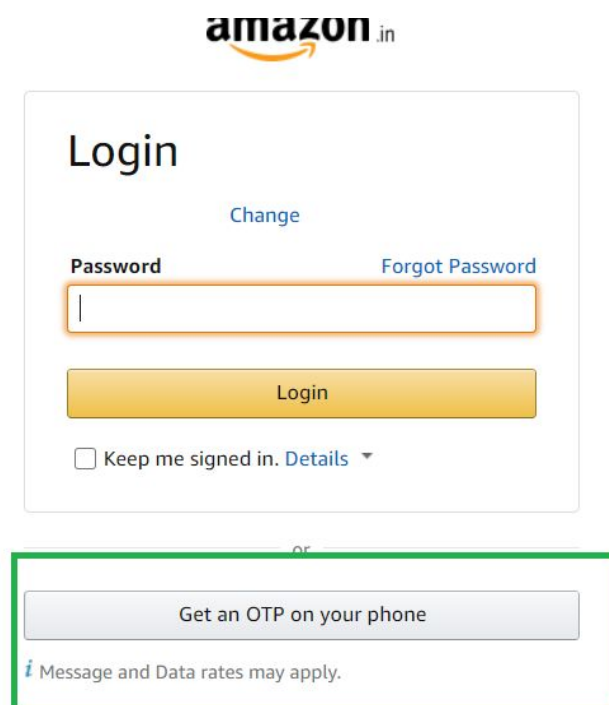
So to find this, run the following code:

```
cnt=browser.find_element_by_id('continue')
```

Now we need to click it. For that we will be running the following code:

```
cnt.click()
```

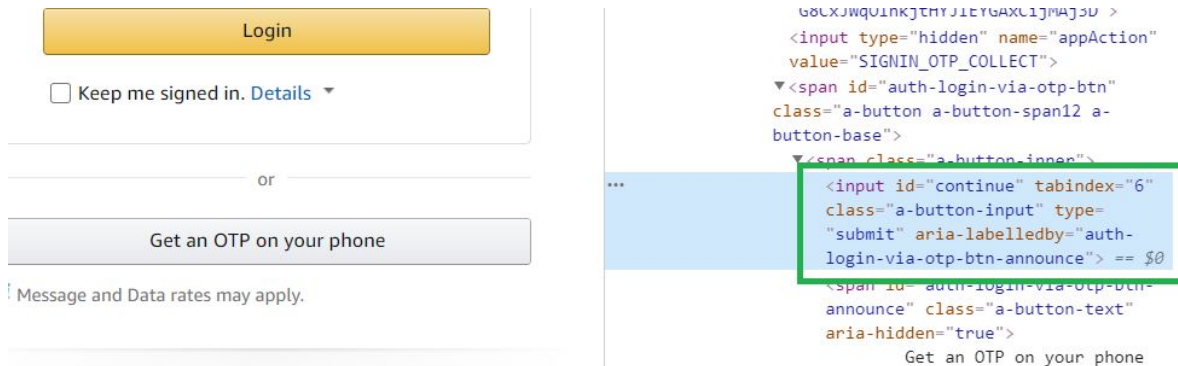
So now we will be directed to another place which looks like this:



Here we get an option called **'Get an OTP on your phone'**. To proceed further with this Message Bomber, we need to click on this option.

And as you know to do this, we need to find this element.

On inspecting the page, we got to see how it is actually stored.



This is stored as an input element with **id as 'continue'**. So now to find this and click this, run the following code.

```
sendOTP=browser.find_element_by_id('continue')
sendOTP.click()
```

On clicking this button, we are taken to this page which has this option called **'Resend OTP'**

The image shows a page titled 'Authentication required'. It includes a 'Change' link, a message stating 'We've sent an OTP to the mobile number above. Please enter it to complete verification', an 'Enter OTP' label, an input field, a yellow 'Continue' button, and a 'Resend OTP' button which is highlighted with a green box. At the bottom, there is an 'or' separator and a 'Login with your password' button.

This is because we want to send message bombs. That means **a lot of messages**. But when we clicked on **'Get an OTP on your phone'**, which was stored as **sendOTP** in the code then the OTP was sent just once. We want to send it (times-1) times more. If we manually had to send the OTP 10 times(for example) then we will have to press the **'Resend OTP'** 9 more times. So we need to do a task repeatedly and for times-1 times. To achieve this, we will have to use a **for loop which will be running for (times-1) number of times**.

We know for loops always run over a sequence. But **times** is holding a number. So **times-1** will also be a number over which we can't run a loop. But we have seen in our previous projects how to tackle this problem. We need to put (times-1) inside the **range** function which will be returning a sequence of numbers starting **from 0 and going till (times-2)** and overall having **(times-1) elements**. For example, if times=10, then times-1 is 9. And range(9) will be returning a sequence like this: [0,1,2,3,4,5,6,7,8]. This starts from 0 and goes till 8(which is equal to times-2) and has 9 elements overall. So if i run a for loop over this sequence, this loop will be running 9 times.

Similarly if I run a for loop for range(times-1), then that loop will be running for times-1 times.

Now inside this loop, we can click the **'Resend OTP'**. But to click it, we need to find it on the page. So let's see how it is stored on the page.



So this is stored as an **'a' element** with a link text as Resend OTP. So to find this and click this times-1 times, run the following code:


```
times=int(times)
n=times-1
for i in range(n):
    r=browser.find_element_by_link_text("Resend OTP")
    r.click()
```

Here, because we took times as input so it is stored as a string. But to do any mathematical operation with it, we need to convert it into an integer. Hence the first line of this block of code.

This will be sending the OTP for next times-1 times.

Once this is done, we will be **closing the browser**. And for that, we need to run the following code.

```
browser.quit()
```

This closes the browser.

Project 7: Covid 19 Dataset creator

In this project we will be building a Covid 19 Dataset creation Bot. This Bot will be able to fetch different statistics related to the current status of covid in different countries across the globe. We can get an updated covid dataset daily by just executing these few lines of code. Well let's begin.

To begin with we need to open google and we will be doing that by executing the following lines of code -

```
from selenium import webdriver

cd='C:\\webdrivers\\chromedriver.exe'
browser= webdriver.Chrome(cd)
```

Now before we move forward we will make few more imports like

```
import time
```

```
import pandas as pd
```

```
Import os
```

Now of course the question arises what are these?? Let's check them one by one

`import time` - time module contains a class `sleep()` that helps to provide a momentary delay to the execution of the code and thus helps in assuring that the web page gets ample amount of time to load.

`import os`- to work with the operating system. We will need it later on for creating the path where we wish to save our csv file

`import pandas as pd`- Well now we need to learn a bit about pandas and dataframes. First of all, what is pandas??

So basically it is a python library that helps us to perform data manipulation and analysis using different types of tools on different types of data. `pandas` stand for `panel` data, `dataframes` and `series` data.

Before we move forward on finding out more about each type lets see why we have used `as pd`??

`as` keyword is used to create an alias i.e. now instead of writing `pandas` everywhere you can use `pd` to refer to `pandas`. Thus it basically saves time and effort of writing `pandas` everytime we wish to use it.

Series data - It is a one dimensional array of data that can be of any type like integer, string, characters etc

Example -

```
data = np.array(['England','Germany','Canada','France'])
```

```
s = pd.Series(data)
```

```
print s
```

The output will be

```
0    England
```

1 Germany

2 Canada

3 France

Dataframe - It is a 2 dimensional representation of data. Data is thus arranged in a tabular format. Example -

```
import pandas as pd
```

```
data = [['Alex',10],['Bob',12],['Clarke',13]]
```

```
df = pd.DataFrame(data,columns=['Name','Age'])
```

```
print df
```

Output -

	Name	Age
0	Alex	10
1	Bob	12
2	Clarke	13

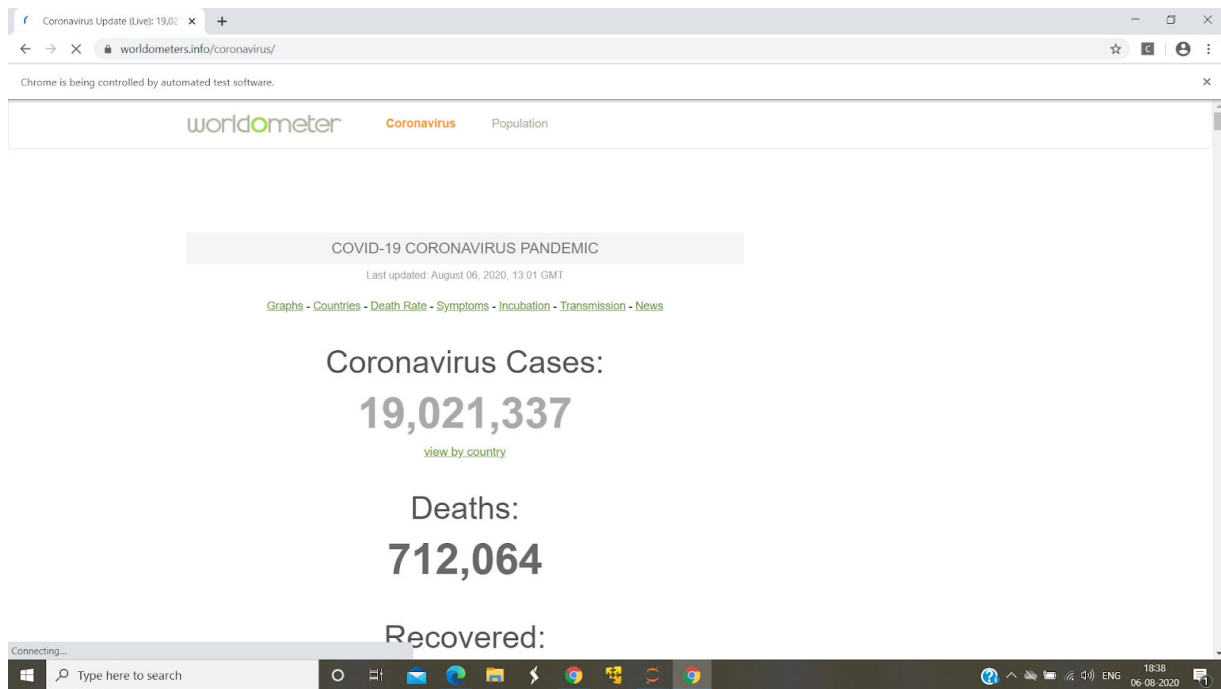
Panel Data - It is basically a 3d container of data which you can visualize as a number of dataframes piled one on top of another.

Basically here we will be using dataframes. So lets move forward and see the next steps.

Next we need to load the page . So we will type

```
browser.get("https://www.worldometers.info/coronavirus/")
```

This page contains all the updated information on covid19. The following page thus opens -



If you scroll down you will find that there is the following table which we will be fetching to create our dataset.

Report coronavirus cases

Now Yesterday 2 Days Ago Columns Search:

	Europe	North America	Asia	South America	Africa	Oceania							
#	Country, Other	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop	Total Tests	Tests/ 1M pop	Population
	World	19,021,337	+55,939	712,064	+1,778	12,203,893	6,105,380	65,429	2,440	91.4			
1	USA	4,974,278	+710	161,635	+34	2,541,459	2,271,184	18,424	15,019	488	62,384,165	188,362	331,192,837
2	Brazil	2,862,761		97,418		2,020,637	744,706	8,318	13,459	458	13,329,028	62,664	212,706,570
3	India	1,977,972	+14,733	40,888	+149	1,335,309	601,775	8,944	1,432	30	22,149,351	16,035	1,381,307,956
4	Russia	871,894	+5,267	14,606	+116	676,357	180,931	2,300	5,974	100	29,716,907	203,623	145,940,753
5	South Africa	529,877		9,298		377,266	143,313	539	8,924	157	3,113,191	52,429	59,379,523
6	Mexico	456,100	+6,139	49,698	+829	304,708	101,694	3,951	3,534	385	1,041,860	8,073	129,062,461
7	Peru	447,624		20,228		306,430	120,966	1,419	13,558	613	2,466,745	74,716	33,015,064
8	Chile	364,723		9,792		338,291	16,640	1,400	19,063	512	1,737,813	90,832	19,132,066
9	Spain	352,847		28,499		N/A	N/A	617	7,546	610	7,064,329	151,087	46,756,599
10	Colombia	345,714		11,624		186,317	147,773	1,493	6,787	228	1,761,772	34,589	50,934,781
11	Iran	320,117	+2,634	17,976	+174	277,463	24,678	4,156	3,807	214	2,612,763	31,069	84,094,686

But before proceeding further we will create the dataframe in which we will store the value. For that we use the following line of code-

```
df=pd.DataFrame(columns=['Rank','Country', 'Total Cases', 'New Cases', 'Deaths', 'New Deaths','Recovered', 'Active Cases', 'Critical'])
```

Well here we have defined our dataframe and provided the names of the columns that we wish to give.

Now we will inspect the loaded page and find out what things we need to do next.

The screenshot shows a web browser displaying a table of COVID-19 data. The table has columns for Country, Total Cases, New Cases, Total Deaths, New Deaths, Total Recovered, Active Cases, Serious/Critical, and Total Cases 1M pop. The first row is for the USA, showing 4,974,278 total cases, 710 new cases, 161,635 total deaths, 34 new deaths, 2,541,459 total recoveries, 2,271,184 active cases, 18,424 serious/critical cases, and 1 million population.

The browser's developer tools are open, showing the DOM structure. The table is identified as `table#main_table_countries_today` with the class `table-bordered table-hover main_table_countries dataTable no-footer`. The table is located within a `tbody` element.

So this region basically is the part where the data is present. So we will use and for each country we have a `tr` tag

for `i` in `browser.find_elements_by_xpath('//*[@id="main_table_countries_today"]/tbody/tr')`:

to locate this portion. Now we will have to fetch the data.

The screenshot shows a web browser displaying a table of COVID-19 data. The table has columns for Country, Total Cases, New Cases, Total Deaths, New Deaths, Total Recovered, Active Cases, Serious/Critical, and Total Cases 1M pop. The first row is for the USA, showing 4,974,278 total cases, 710 new cases, 161,635 total deaths, 34 new deaths, 2,541,459 total recoveries, 2,271,184 active cases, 18,424 serious/critical cases, and 1 million population.

The browser's developer tools are open, showing the DOM structure. The table is identified as `table#main_table_countries_today` with the class `table-bordered table-hover main_table_countries dataTable no-footer`. The table is located within a `tbody` element.

We use the code

```
td_list=i.find_elements_by_tagname('td')
```

As you can see the data have td tags. Now we define row=[] to append the data individually for each country. Using a loop we fetch the datas, extract their text part and append them to row

```
row=[]
```

```
for td in td_list:
```

```
    row.append(td.text)
```

For example we get the following result

```
['1', 'USA', '4,974,278', '+710', '161,635', '+34', '2,541,459', '2,271,184', '18,424', '15,019', '488', '62,384,165', '188,362', '331,192,837', '']
```

Now what we need to do is to add this data to the dataframe. So for that we define an empty dictionary

```
data={}
```

Now using the following loop we pick one data at a time at and assign them to different columns

```
for j in range(len(df.columns)):
```

```
    data[df.columns[j]]=row[j]
```

So now it looks like this

```
{'Rank': '1', 'Country': 'USA', 'Total Cases': '4,974,278', 'New Cases': '+710', 'Deaths': '161,635', 'New Deaths': '+34', 'Recovered': '2,541,459', 'Active Cases': '2,271,184', 'Critical': '18,424'}
```

Finally we append it to the main dataframe and this process continues.

```
df.append(data)
```

Thus the final dataframe is created.

But if you observe the dataframe you will find that an extra row of data crops up at the top which is irrelevant so we use slicing as df[1:] which removes the first row and considers the rest of the rows.

Now the last thing is to save it to a csv file. For that we define the path that we wish to store it to and using `os.path.join()` we join the path and the name of the csv file that we wish to give it. Thus the final path is ready. Now we have to use `.to_csv()` to convert the data frame and store it in the csv format.

So the final csv file looks as -

Rank	Country	Total Case	New Case	Deaths	New Death	Recovered	Active Cas	Critical
1	USA	48,15,776	2,129	1,58,376	11	23,80,561	22,76,839	18,623
2	Brazil	27,33,677		94,130		18,84,051	7,55,496	8,318
3	India	18,19,200	14,498	38,375	214	11,97,512	5,83,313	8,944
4	Russia	8,56,264	5,394	14,207	79	6,53,593	1,88,464	2,300
5	South Africa	5,11,485		8,366		3,47,227	1,55,892	539
6	Mexico	4,39,046	4,853	47,746	274	2,89,394	1,01,906	3,944
7	Peru	4,28,850		19,614		2,94,187	1,15,049	1,410
8	Chile	3,59,731		9,608		3,32,411	17,712	1,437
9	Spain	3,35,602		28,445		N/A	N/A	617
10	Colombia	3,17,651		10,650		1,67,239	1,39,762	1,493
11	Iran	3,12,035	2,598	17,405	215	2,70,228	24,402	4,104
12	UK	3,04,695		46,201		N/A	N/A	86
13	Saudi Arabia	2,80,093	1,258	2,949	32	2,42,055	35,089	2,017
14	Pakistan	2,80,029	331	5,984	8	2,48,873	25,172	1,064
15	Italy	2,48,070		35,154		2,00,460	12,456	42
16	Bangladesh	2,42,102	1,356	3,184	30	1,37,905	1,01,013	
17	Turkey	2,32,856		5,728		2,16,494	10,634	582
18	Germany	2,11,567	105	9,226		1,93,600	8,741	261
19	Argentina	2,01,919		3,648		89,026	1,09,245	1,122
20	France	1,87,919		30,265		81,500	76,154	371
21	Iraq	1,31,886	2,735	4,934	66	94,111	32,841	514
22	Canada	1,16,884		8,945		1,01,574	6,365	2,253

Project 8:Instagram Profile Picture Downloader

In this project, we will be building a python bot which will take any instagram user handle as input and then saves the profile picture in the device. In this project, we will not need to login to any account. The only input will be the **user-handle** whose profile pic the user wants to download.

To start with this, the first thing that we would be doing is Opening google. And that will be done by running the following code.

```
from selenium import webdriver
cd='C:\\webdrivers\\chromedriver.exe'
driver = webdriver.Chrome(cd)
```

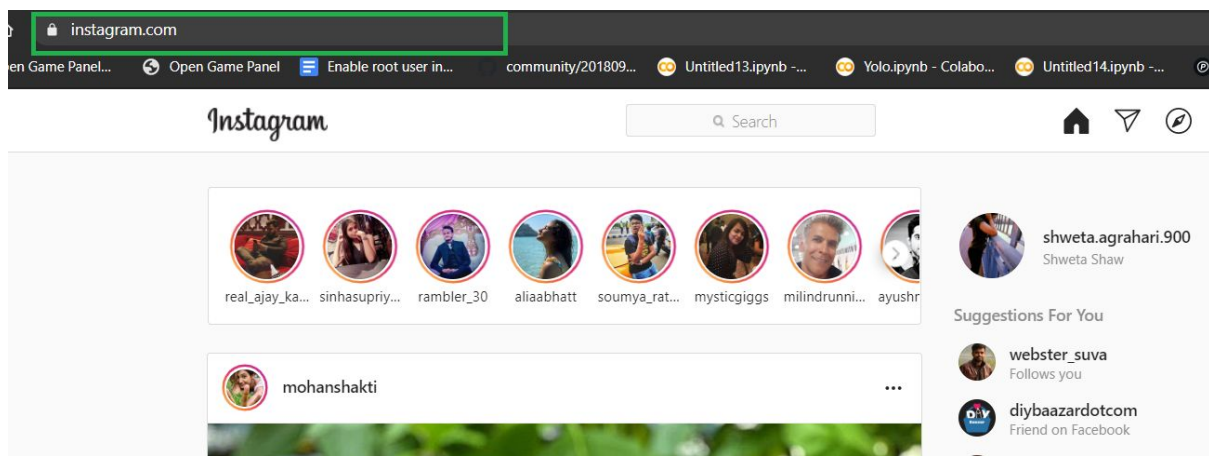
Now I will ask the user **to enter the user-handle whose profile pic he wants**. To do that we will be running the following code:

```
user_h=input("Enter the user handle of the profile: ")
```

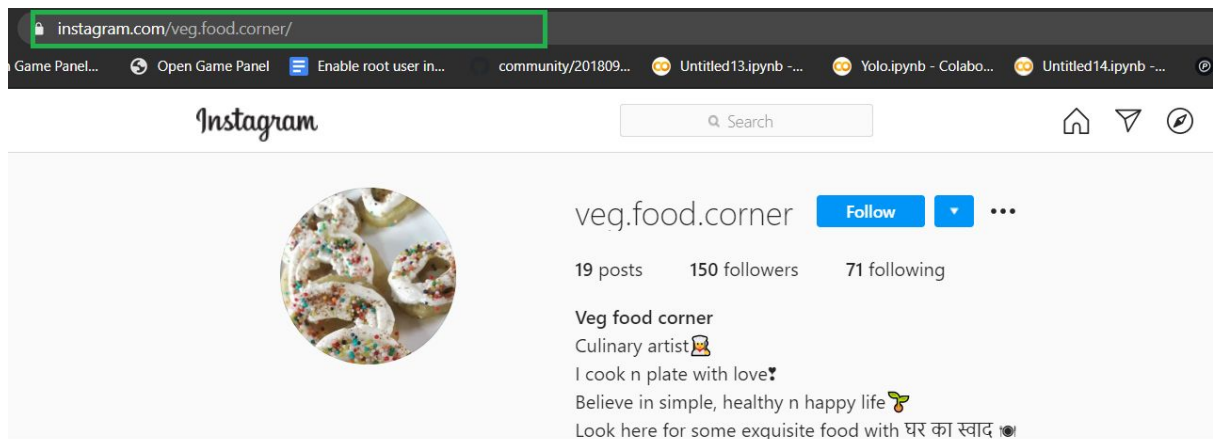
This will be printing the message over the user's screen and whatever keyboard input will be received, will be saved as **user_h**.

Now our task is to open the profile on instagram.

When we open instagram, this is the link of the page



But from here, if we go to any instagram profile, for example if I go to the page of **veg.food.corner**, so this will be the link to that page



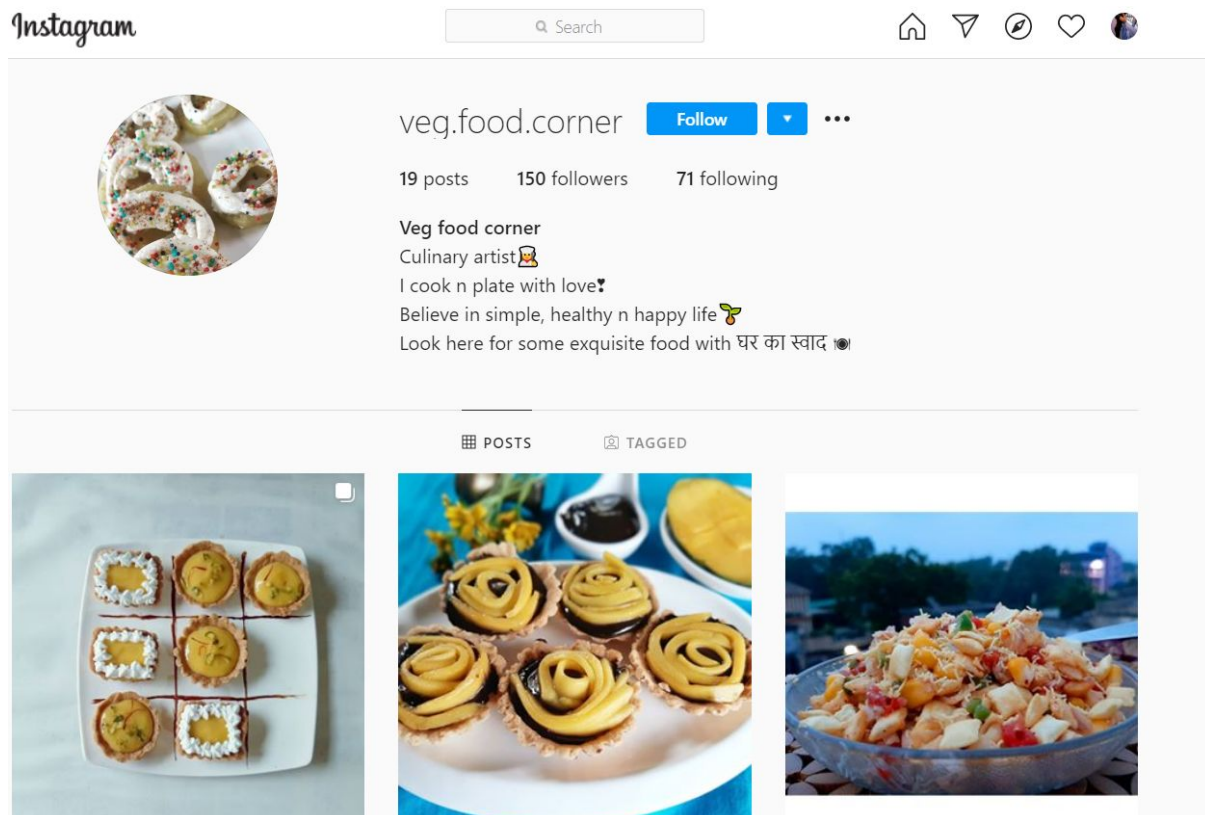
That means whatever user-handle we want to open, we need to put that after '<https://www.instagram.com/>' and the new link will be the profile itself. Using the same concept, we will be developing the link of the profile whose **user_handle** is entered to the program. To do this run the following code.

```
url='https://www.instagram.com/'  
url_p=url+user_h
```

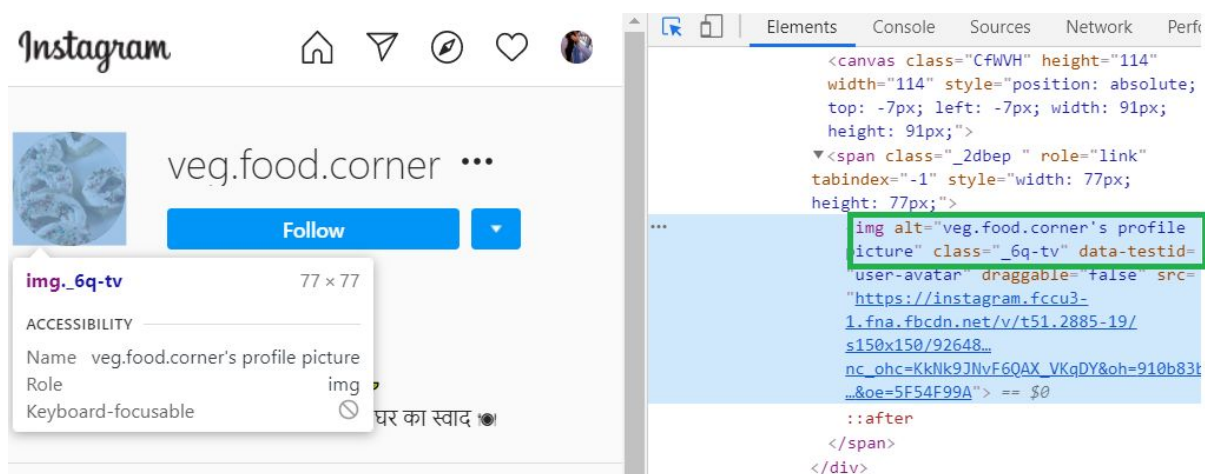
Now our next task is to open this page in the browser. So for that, run the following code.

```
driver.get(url_p)
```

Now once we do this, the following page appears(for veg.food.corner).



So from here I need to download the profile picture. For that I need to see how this element is stored in the code. So next we will be inspecting the page.

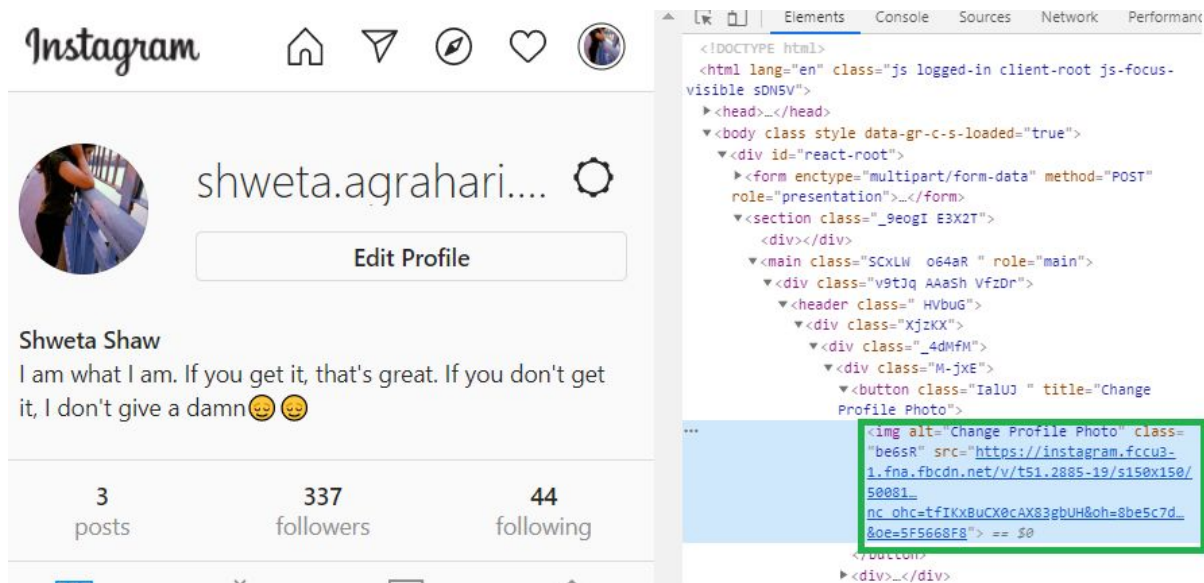


Here I see that the profile pic is actually stored in an **img element** with **class name '_6q-tv'**. It also has the link of the image in its **src attribute**. So now our task is to find this element and retrieve the image link.

To find the element I am going to use **.find_element_by_xpath()** method as usual. And for retrieving the image link I will be using **.get_attribute()**

function. This will return the value of the attribute given to this function as input.

But the problem here is, in a private account, the same element is stored with a different class name. Look here.



Here I see that the profile pic is actually stored in an **img element with class name 'be6sr'**. It also has the link of the image in it's **src attribute**.

That means for doing the same task we need to run two different codes depending on whether the profile is public or private. Because if i run a code with the class name = '**_6q-tv**' so it will be working properly with public profiles but start giving error for private profile and vice-versa

Here **try-except block** is going to help me here. Try-except block helps in exception handling. When you are aware of the fact that some code can throw an error then you put that code in the **try block**. And you put another line of code inside the **except block** which will run instead of the code put in the try block if that code throws an error. If the code staying inside the try block doesn't throw an error then that code will be executed only.

So now, to get this profile pic element, we will run the following code:

```
try:  
    image=driver.find_element_by_xpath("//img[@class='_6q-tv']")  
except:  
    image=driver.find_element_by_xpath("//img[@class='be6sr']")
```

The code inside the try block will be running. For public profiles, this will run and not throw an error so things will be fine. If the profile is private, so this code inside try will be throwing an error because in that page **no such element is present**. So in case of any error, the code of the except block will be running instead of the one in the try block. So be it a private or public profile, after running the above code, **image** is holding the element. Now to get the link, run the following code:

```
img_link=image.get_attribute('src')
```

This code will return whatever the value of the 'src' attribute is. The value is nothing but the link of an image which is stored in **img_link**. Now we have the link of the image we want to download.

So my next step is going to be downloading an image from a link using python. Now if you don't know how to do this then just do a simple search over google for this and you will get various options.

One of those options is using **urllib.request.urlretrieve()**. This method takes the link of the image to be downloaded and also the path where

The image will be downloaded as input and then downloads the image at that path. Right now in the program, I am going to create the path with the help of the user `_handle` and then download the photo with that path. To do this, run the following code:

```
path="D:\\"+user_h+".jpg"
import urllib.request
urllib.request.urlretrieve(img_link,path)
print("The profile pic has been downloaded at: "+path)
```

Here a new name or path of the file will be created. For example, for the page **veg.food.corner**, the path will be **"D:\\veg.food.corner.jpg"**. The image will be downloaded at this path. After the download a confirmation message will be printed out over the user's screen.s

Project 9: Facebook Friend list Scraper

Now we will be discussing facebook friend list scraper. So basically this program will return the names of all the facebook friends that we have. So let's begin with the coding part

So the first thing that we need to do is to import the required modules.

```
from selenium import webdriver
```

```
import time
```

```
from selenium.webdriver.common.keys import Keys
```

```
from bs4 import BeautifulSoup
```

Well you already know the first three. The question arises: **what is BeautifulSoup?**

Beautiful Soup is a library that makes it easy to scrape information from web pages. It helps to pull the data out of HTML and XML files. Here we will use it to pull data out from the HTML page.

The next step is to login to your fb account. Well you already know how to do that. Just a quick recap.

```
user_id=input('Enter User Id of your Fb Account :')
```

```
password=input('Enter the password :')
```

```
print(user_id)
```

```
print(password)
```

```
cd='C:\\webdrivers\\chromedriver.exe'
```

```
browser= webdriver.Chrome(cd)
```

```
browser.get('https://www.facebook.com/')
```

```
user_box = browser.find_element_by_id("email")
```

```
user_box.send_keys(user_id)
```

```
password_box = browser.find_element_by_id("pass")
```

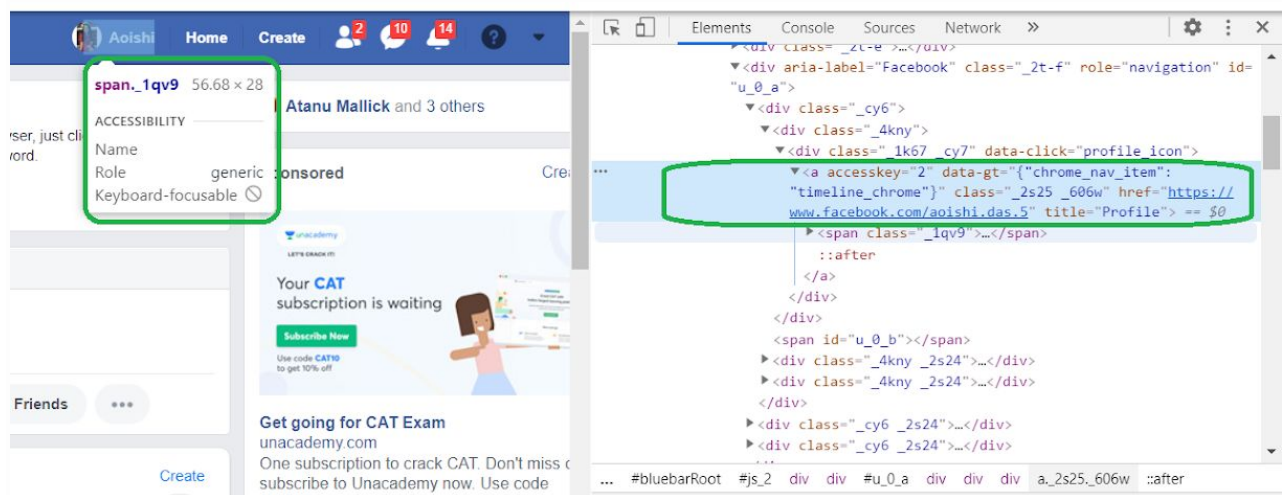
```
password_box.send_keys(password)
```

```
login_box = browser.find_element_by_id("u_0_b")
```

```
login_box.click()
```

```
time.sleep(20)
```

Now the next thing is to find the friends list right?? So at first we will inspect and find out

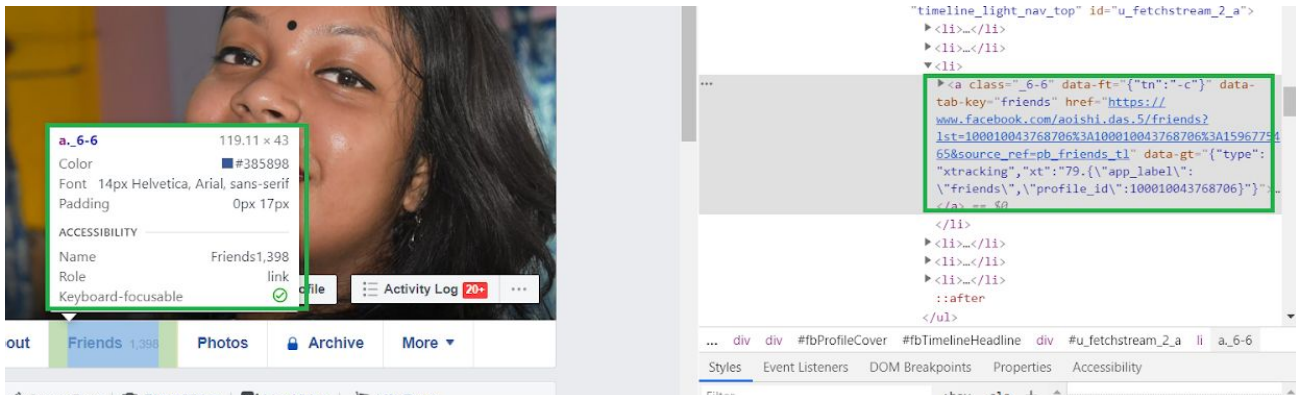


So at first we need to click on the profile. For that you can see that we need to give the path of this element.

```
pro=browser.find_element_by_xpath('//a[@class="_2s25_606w"]')
```

```
pro.click()
```

So we have detected the profile. Then we need to access friends



Now we will use the following lines of code

```
fr=browser.find_element_by_xpath('//ul[@class="_6_7 clearfix"]/li[3]/a/
```

```
fr.click())
```

Now we will have to ensure that all the friends' names are loaded before we proceed to extracting the data. For that we will be using

```
while True:
```

```
browser.execute_script('window.scrollTo(0,document.body.scrollHeight);')
```

```
time.sleep(0.1)
```

```
browser.execute_script('window.scrollTo(0,0);')
```

```
time.sleep(0.1)
```

```
try:
```

```
exit_control=browser.find_element_by_xpath("//*[@contains(text(), 'More
about you')]")
```

```
break
```

```
except:
```


continue

window.scrollTo(x,y): Scrolls the page to that location (xpos,ypos)

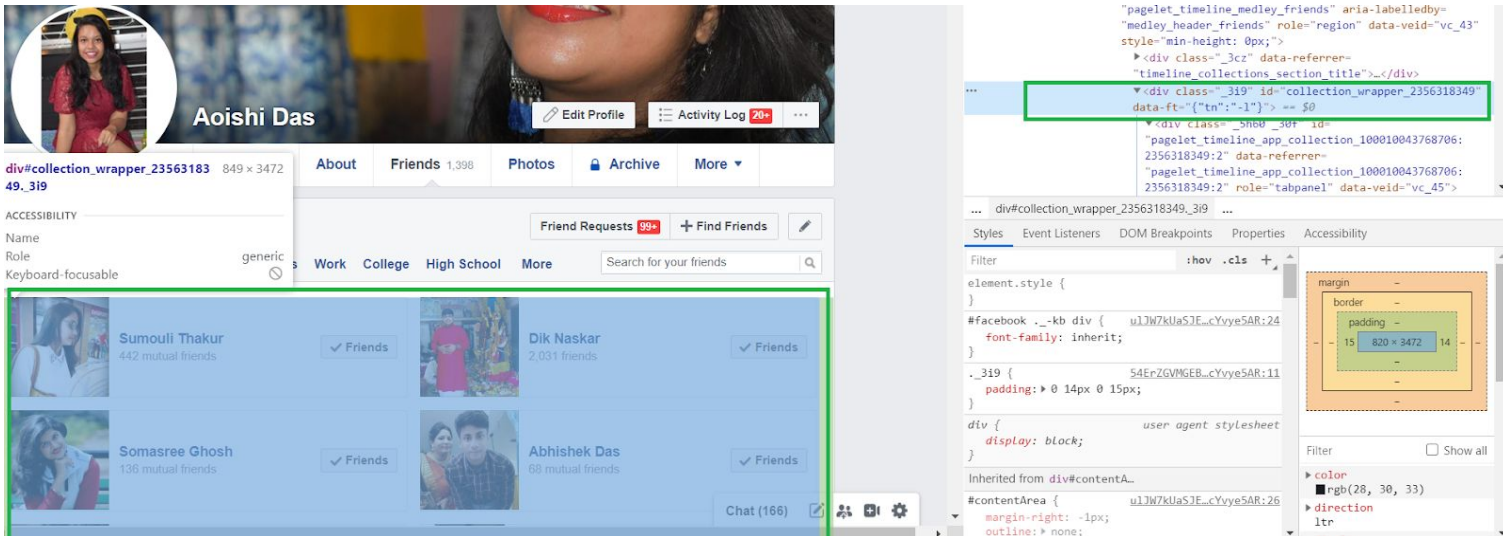
`.execute_script()` - Now this is a JS command and for in order to execute that we will be using `.execute_script()`. Now we have used a while that when True performs the scrolling operation - move to the full document height i.e the bottom then again move to the (0,0) position and like this it continues till we reach the end of the friend list. This occurs when the message 'More about you' is displayed. Thus we put it in a try block and on meeting that condition we break out from the while loop.

Now we need to fetch the data right?? For that we will use BeautifulSoup

```
ps=browser.page_source  
soup=BeautifulSoup(ps,'html.parser')
```

`.page_source()` returns the present page and in the next line we create a soup object. The first argument passed is the page and second is the type of parser that we wish to use (lxml or html)

Now we need to inspect this page. So let's see what we get.



We find that the area where our data is present comes under the class '3i9'. So now we will use the following line -


```
flist=soup.find('div',{'class': '_3i9'})
```

So we are passing the soup object and locate this particular div and store in flist

Now we declare an empty list that will be used to store the name of the friends.

```
friends=[]
```

Now lets take a close look at the data

You will observe that the names are in 'a' tags. So using a loop we will find out all the text content of such tags. So findAll() basically helps to return all such elements. Using i.text we extract the text content and append it to our list.

```
for i in flist.findAll('a'):
```

```
    friends.append(i.text)
```

Now if you print the list you will find that it looks something like this-

```
print(friends)
```

```
[['', 'FriendFriends', 'Sumouli Thakur', '442 mutual friends', '', 'FriendFriends', 'Dik Naskar', '2,031 friends', '', 'FriendFriends', 'Somasree Ghosh', '136 mutual friends', '', 'FriendFriends', 'Abhishek Das', '68 mutual friends', '', 'FriendFriends', 'Aritra Sarkar', '17 mutual friends', '', 'FriendFriends', 'Sneha Dutta', '13 mutual friends', '', 'FriendFriends', 'Trip arna Jana', '101 mutual friends', '', 'FriendFriends', 'Soham Dhar Chowdhury', '44 mutual friends', '', 'FriendFriends', 'Sus mita Das', '127 friends', '', 'FriendFriends', 'Sanjana Das', '787 friends', '', 'FriendFriends', 'Tinanjali Dutta', '1,719 friends', '', 'FriendFriends', 'Shruti Mitra', '77 mutual friends', '', 'FriendFriends', 'Pritha Sengupta', '1,650 friends', '', 'FriendFriends', 'Bishakha Das', '692 friends', '', 'FriendFriends', 'Shrestha Mallick', '7 mutual friends', '', 'FriendFriends', 'Suprakash Das', '313 friends', '', 'FriendFriends', 'Sayan Kundu', '100 mutual friends', '', 'FriendFriends', 'Ruma Mallick', '54 friends', '', 'FriendFriends', 'Ayan Dhar', '4,969 friends', '', 'FriendFriends', 'Sudipta Das', '1,133 friends', '', 'FriendFriends', 'Triya Chakraborty', '1,517 friends', '', 'FriendFriends', 'Kumar Mozumder', '745 friends', '', 'FriendFriends', 'Subhasree Manna', '404 mutual friends', '', 'FriendFriends', 'Tithi Ghosh', '924 friends', '', 'FriendFriends', 'Lalita Dutta', '132 friends', '', 'FriendFriends', 'Debangana Dutta', '882 friends', '', 'FriendFriends', 'Sunetra Bhat
```

```
names_list=[]

for name in friends:

    if(name=='FriendFriends'):

        continue

    if('friends' in name):

        continue

    if(name==""):

        continue

    else:

        names_list.append(name)
```

```
In [10]: print(names_list)
```

```
['Sumouli Thakur', 'Dik Naskar', 'Somasree Ghosh', 'Abhishek Das', 'Aritra Sarkar', 'Sneha Dutta', 'Triparna Jana', 'Soham Dh  
ar Chowdhury', 'Susmita Das', 'Sanjana Das', 'Tinanjali Dutta', 'Shruti Mitra', 'Pritha Sengupta', 'Bishakha Das', 'Shrestha  
Mallick', 'Suprakash Das', 'Sayan Kundu', 'Ruma Mallick', 'Ayan Dhar', 'Sudipta Das', 'Triya Chakraborty', 'Kumar Mozunder',  
'Subhasree Manna', 'Tithi Ghosh', 'Lalita Dutta', 'Debangana Dutta', 'Sunetra Bhattacharyya', 'Sampa Jana', 'Raka Sen', 'Anam  
ika Singh', 'Anand Kumar', 'Biswadi Basu Mallick', 'Sarajit Dutta', 'Utsav Sarkar', 'Sreyasi Sorcer', 'Deep Das', 'Ayantika K  
undu', 'Pinkhi Ghosh', 'Sumit Paul', 'Sucheta Saha', 'Anuran Basu', 'Mohana Ghosh', 'Anshuman Banarjee', 'Tanisha Banerjee',  
'Anamitra Biswas', 'Maitreyee Banerjee', 'Rik Mondal', 'Debanjan Ghosh', 'Banasee Mitra', 'Debrun Nandy', 'Popi Das', 'Monal  
isa Mazumder', 'Ipsitha Pal', 'Satavisha Mallick', 'Dishananti Ganguly', 'Subhrajyoti Banerjee', 'Andrew D'costa', 'Sayantan  
Sen', 'Tirthya Chakraborty Sarkar', 'Mou Kundu', 'Ayantika Dutta', 'Soham Ghosh', 'Monalisa Nayak Jena', 'Arpita Dey', 'Banas  
ree Mitra', 'Soma Das', 'Ghantu da Chowdhury', 'Jayita Poddar', 'Sohini Chakraborty', 'প্রিতিকাজ সরকার', 'Madhuchanda Sarkar',  
'Banasee Mitra', 'Biplab Das', 'Arindam Ghosal', 'Debayjoti Roy', 'Krishnendu Chakraborty', 'Parijat Das', 'Aiyush Ganguly',  
'Ratul Das', 'Ushashi Samanta', 'Barsha Chakraborty', 'Rouble Goswami', 'Shreya Ray', 'Priyanka Mallick', 'Picasso Baral', 'S  
ohini Chaudhuri', 'Prarahshini Biswas', 'Sresta Biswas', 'Anusree Nundy', 'Aparatarka Banerjee', 'Paramita Roy', 'Moumita Sah  
a', 'Roshni Dutta', 'Sampurna Ghosh', 'Suchismita Bhattacharya', 'Piyaali Mukherjee', 'প্রাচীন ঘোষ', 'Saroni Ghoshal', 'Ani Rud  
dha Ghosal', 'Nilanjana Hazra', 'Atmaja V Acharya', 'Rama Guha', 'Purbali Roy Das', 'Anushka Guha Ray', 'Aditi Ghosal', 'Suma  
n Basu', 'Sneha Ray', 'Sayani Saha', 'Mrityika Dutta', 'Aparajita Roy', 'Aindrita Basak', 'Taneshta Kabasi', 'Poulami Kargupt  
a', 'Tiyasha Guha', 'Torsa Talukdar', 'Oindrila Ghosh', 'Vaswati Kundu', 'Ritobina Sinha', 'Puza Mallick', 'Sanchari Das', 'A  
njanika Poddar', 'Shalini Mondal', 'Anwsha Chandra', 'Ritu Gurung', 'Rishit Mitra', 'Adrija Ray Chaudhuri', 'Anisha Pal', 'S  
nigdha Paul', 'Sarbani Roy Mitra', 'Anushka Chakraborty', 'Akash Gupta', 'Raktim Rakshit', 'Rohit Sarkar', 'Shreyoshi Dutta',
```