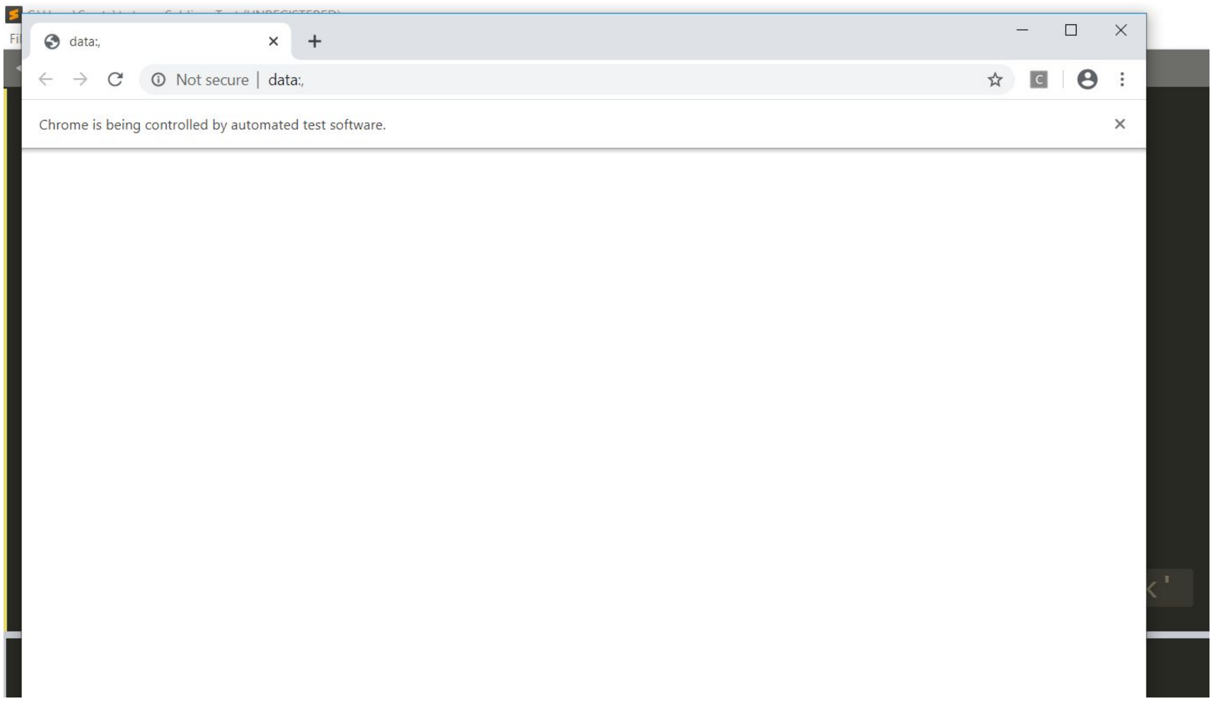
Project 11: Twitter Scraper to get trending hashtags.

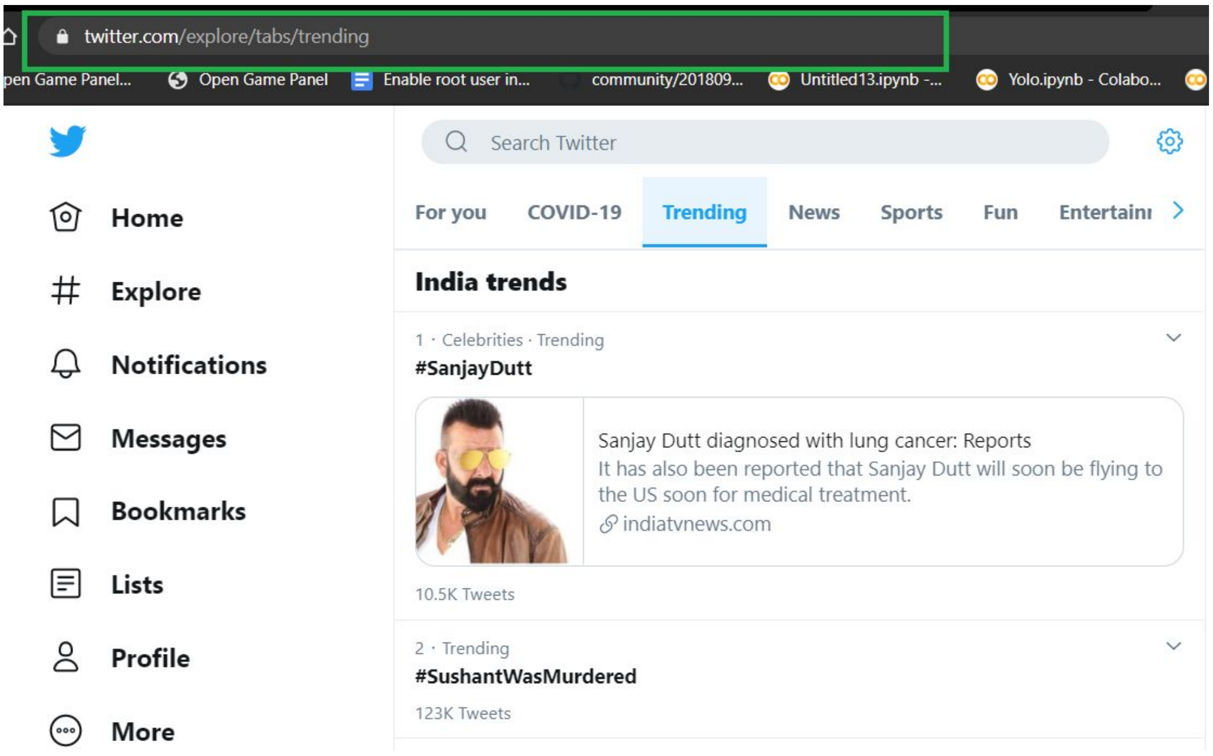
In this project we are going to write a program which will be scraping the trending hashtags(#....) from twitter. To do this, the first step will be **to open the browser.** For this we are going to use **selenium**. We will be creating an object of the **webdriver** and giving t**he path of the chromedriver.exe file in my device.** So let's begin with importing the libraries. Run the following code to do so:

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

Here **cd** is holding the path of my chromedriver.exe file. Running this will open google like this.



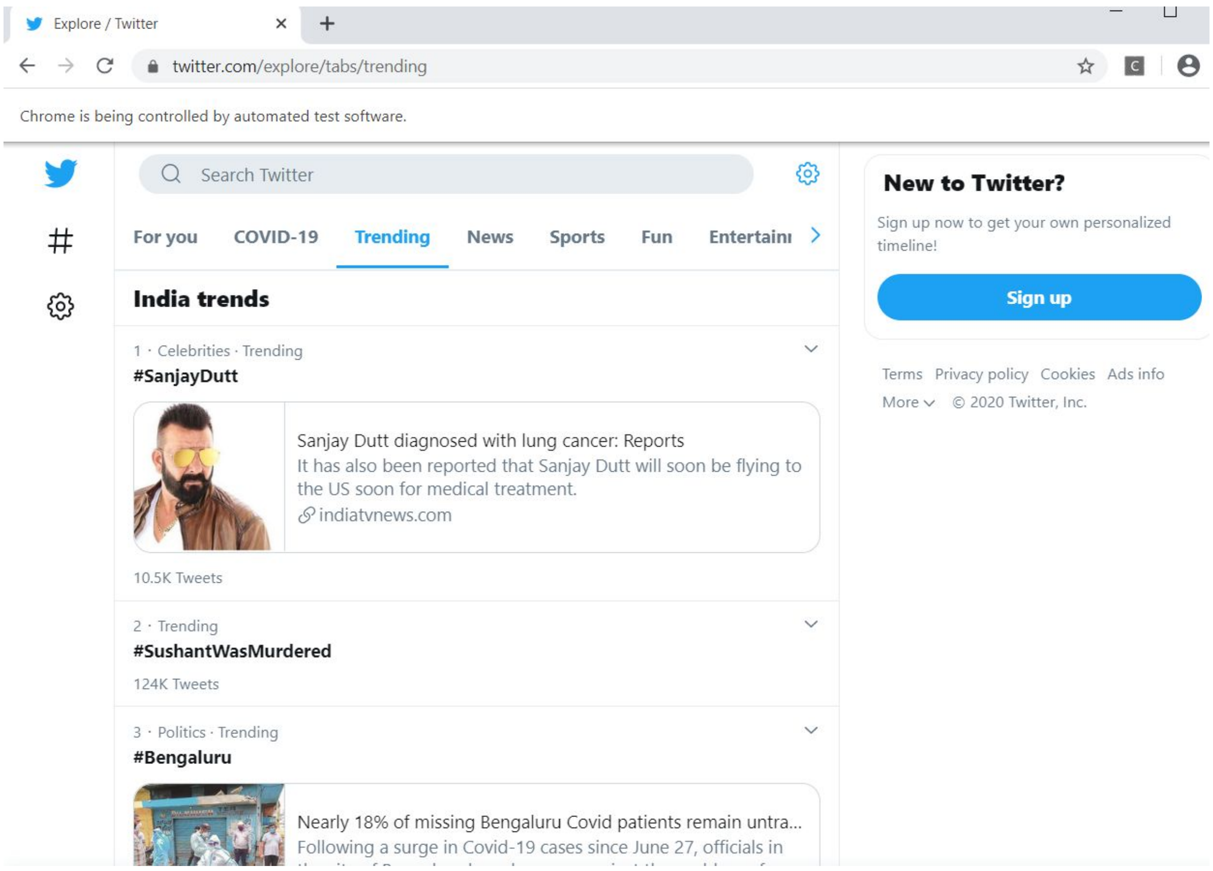
Once google has opened, we need **to open the trending page of twitter**. Let's look at the url of this page.



The URL enclosed in the green box is the one I want to open. So we will be opening it from our program and to do this **.get()** function will help me out. So to open this page, run the following code:

```
browser.get('https://twitter.com/explore/tabs/trending')
```

When this code is ran, the same page opens in the browser like this:



Explore / Twitter

twitter.com/explore/tabs/trending

Chrome is being controlled by automated test software.

Search Twitter

For you · COVID-19 · Trending · News · Sports · Fun · Entertaini

India trends

1 · Celebrities · Trending
#SanjayDutt

Sanjay Dutt diagnosed with lung cancer: Reports
It has also been reported that Sanjay Dutt will soon be flying to the US soon for medical treatment.
indiatvnews.com

10.5K Tweets

2 · Trending
#SushantWasMurdered

124K Tweets

3 · Politics · Trending
#Bengaluru

Nearly 18% of missing Bengaluru Covid patients remain untra...
Following a surge in Covid-19 cases since June 27, officials in

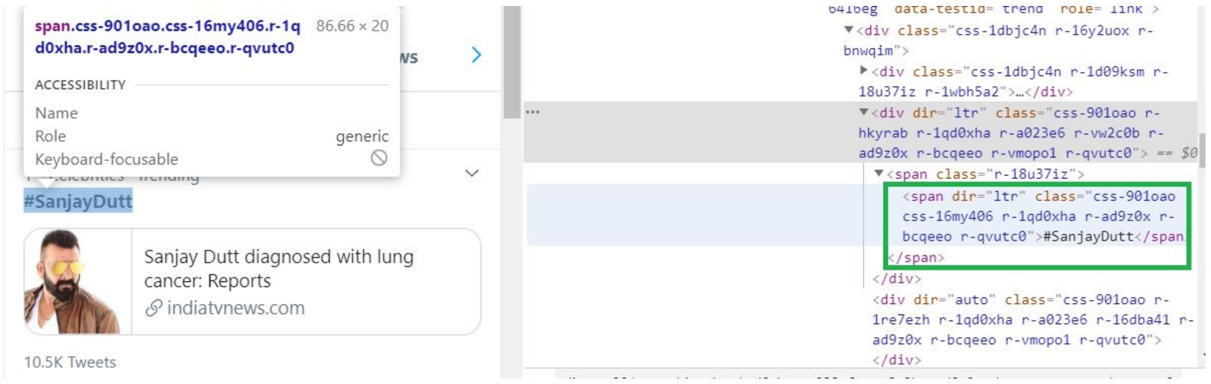
New to Twitter?

Sign up now to get your own personalized timeline!

Sign up

Terms Privacy policy Cookies Ads info
More © 2020 Twitter, Inc.

Now when this page is appearing, I need to get the hashtags from here. For that i will have to check how these hashtags are stored in the source page. For that I will have to **inspect the web page.** By doing that, this is the result I got.



span.css-901oao.css-16my406.r-1qd0xha.r-ad9z0x.r-bcqeeo.r-qvutc0 86.66 × 20

ACCESSIBILITY
Name
Role generic
Keyboard-focusable

#SanjayDutt

Sanjay Dutt diagnosed with lung cancer: Reports
indiatvnews.com

10.5K Tweets

```
b41beg" data-testid="trend" role="link">
  ▼<div class="css-1dbjc4n r-16y2uox r-
  bnwqim">
    ▶<div class="css-1dbjc4n r-1d09ksm r-
    18u37iz r-1wbh5a2">…</div>
    ▼<div dir="ltr" class="css-901oao r-
    hkyrab r-1qd0xha r-a023e6 r-vw2c0b r-
    ad9z0x r-bcqeeo r-vmopo1 r-qvutc0"> == $0
      ▼<span class="r-18u37iz">
         <span dir="ltr" class="css-901oao
         css-16my406 r-1qd0xha r-ad9z0x r-
         bcqeeo r-qvutc0">#SanjayDutt</span>
       </span>
      </div>
      <div dir="auto" class="css-901oao r-
      1re7ezh r-1qd0xha r-a023e6 r-16dba41 r-
      ad9z0x r-bcqeeo r-vmopo1 r-qvutc0">
      </div>
```

This means the #tags are stored as **the text content for the span element.** Now to find this element through my program, I will have to find the span elements. For that I am going to use the function

.find_elements_by_tag_name(). This function finds all the elements with the specified tag name (given as the argument). Here the tag name we are interested in is **span.** So to find the span tags from the page, we will be using **.find_elements_by_tag_name('span')**. But the problem here is that this will return all the span elements present in the page. Even the ones which I am not interested in. That means I need to filter this out. To do this, we will be going to each of the elements returned and check whether those elements are hashtags are not. The identity of any hashtag is presence of '#' in front of the element.

All these things lead me to using **a for loop for each of the elements** returned by the find_elements function. And **inside the for loop check whether the text content of that element is starting with a '#' or not.** Once we get such elements, we will simply append them in a list (starting from a blank list).

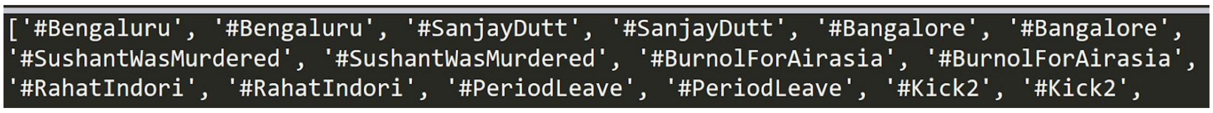
But before writing code for this task, we will have to apply a sleep just after opening the link. Sleep will **halt the program execution for the mentioned duration** at that line only. Once that duration has passed only then the execution goes to the next line of the code.

We will be using sleep because **the browser takes a few seconds to open the link.** If before the page opens, I try to find any element there, it will either give me an error or give me wrong output.

This is done using the following code:

```
import time
time.sleep(15)
sp=browser.find_elements_by_tag_name('span')
fl=[]
for i in sp:
    a=i.get_attribute('textContent')
    if (a.startswith('#')):
            if a not in fl:
                    fl.append(a)
```

After this if I try to print fl, this is the result.

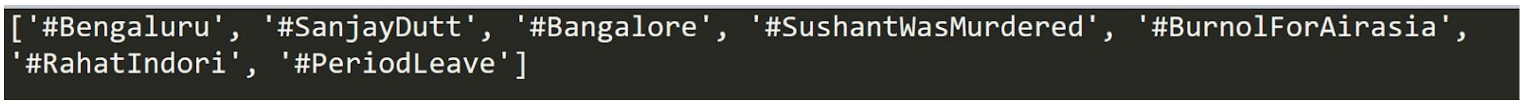


```
['#Bengaluru', '#Bengaluru', '#SanjayDutt', '#SanjayDutt', '#Bangalore', '#Bangalore',
'#SushantWasMurdered', '#SushantWasMurdered', '#BurnolForAirasia', '#BurnolForAirasia',
'#RahatIndori', '#RahatIndori', '#PeriodLeave', '#PeriodLeave', '#Kick2', '#Kick2',
```

This shows the same hashtag is being appended twice. We surely don’t want this to happen. This can be solved by a simple concept. Before appending any hashtag check if that hashtag is already in fl. **If it is not, then only append else don’t.** To do this, the above mentioned code will be changed in this way:

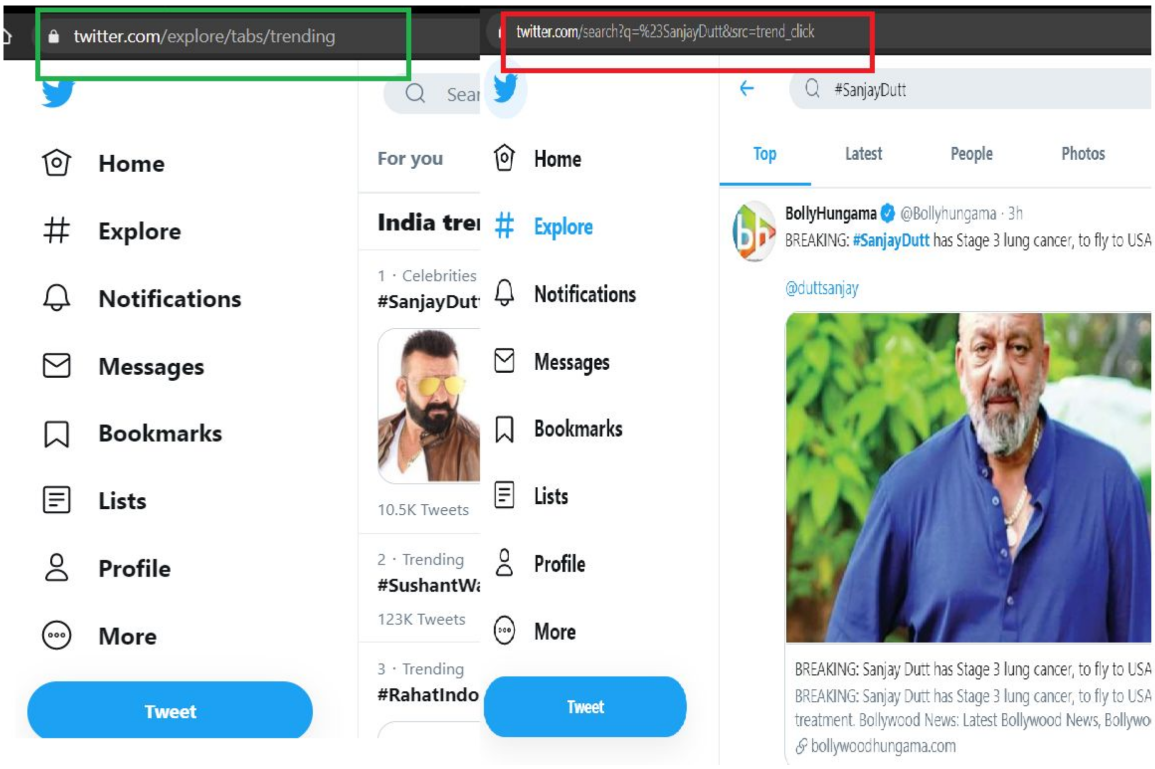
```
import time
time.sleep(15)
sp=browser.find_elements_by_tag_name('span')
fl=[]
for i in sp:
    a=i.get_attribute('textContent')
    if (a.startswith('#')):
            if a not in fl:
                    fl.append(a)

print(fl)
```

Now the output of printing fl is looking like this:



```
['#Bengaluru', '#SanjayDutt', '#Bangalore', '#SushantWasMurdered', '#BurnolForAirasia',
'#RahatIndori', '#PeriodLeave']
```

That means the problem is solved. We have reached halfway of the project. Now we just don't want to store the hashtags, we also want **to get to the tweets which are using these hashtags.** That means the same result that we get once we click on any hashtag. SDo to understand how to do this, let’s check out wj=hat happens when we click on any hashtag.

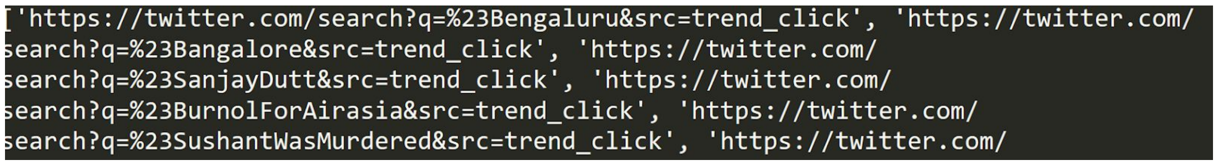

twitter.com/explore/tabs/trending
twitter.com/search?q=%23SanjayDutt&src=trend_click
Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet
For you
India tre
1 · Celebrities
#SanjayDut
10.5K Tweets
2 · Trending
#SushantW
123K Tweets
3 · Trending
#RahatIndo
#SanjayDutt
Top
Latest
People
Photos
BollyHungama @Bollyhungama · 3h
BREAKING: #SanjayDutt has Stage 3 lung cancer, to fly to USA
@duttsanjay
BREAKING: Sanjay Dutt has Stage 3 lung cancer, to fly to USA
BREAKING: Sanjay Dutt has Stage 3 lung cancer, to fly to USA treatment. Bollywood News: Latest Bollywood News, Bollywo
bollywoodhungama.com

Here we can see that the URL gets changed. And the link which I am interested in is nothing but **'https://twitter.com/search?q=%23' then the content of the hashtag excluding the '#' sign** and then **'&src=trend_click'.** That means for each hashtag, we can generate the URL by appending different things together. To do this we will have to iterate over **fl** which is holding the hashtags. And for each hashtag remove # using **slicing** and generate the URL using **string concatenation** and finally store in a list.

To do this, run the following code:

```
urls=[]
for i in fl:
  i=i[1:]
  url='https://twitter.com/search?q=%23'+i+'&src=trend_click'
  urls.append(url)
print(urls)
```

Here the slicing `i[:1]` is done to remove '#'. Now when the list **urls** is printed out, this is the result (a part of the whole result):



```
['https://twitter.com/search?q=%23Bengaluru&src=trend_click', 'https://twitter.com/
search?q=%23Bangalore&src=trend_click', 'https://twitter.com/
search?q=%23SanjayDutt&src=trend_click', 'https://twitter.com/
search?q=%23BurnolForAirasia&src=trend_click', 'https://twitter.com/
search?q=%23SushantWasMurdered&src=trend_click', 'https://twitter.com/
```

At this point, we have 2 lists.1st one **fl,** holding the hashtags and 2nd one urls, holding the urls. Now we need **to create a dataset** with these lists. By creating a dataset I meant **creating a dataframe** and then **saving it as a csv file.**

First of all I need to create the dataframe. For that I will be **creating a dictionary** with two keys which will be converted into the column names and whose key values will be stored as different data in different rows. The key values will be nothing but the lists created above, **fl** and **urls.** To do this, run the following code:

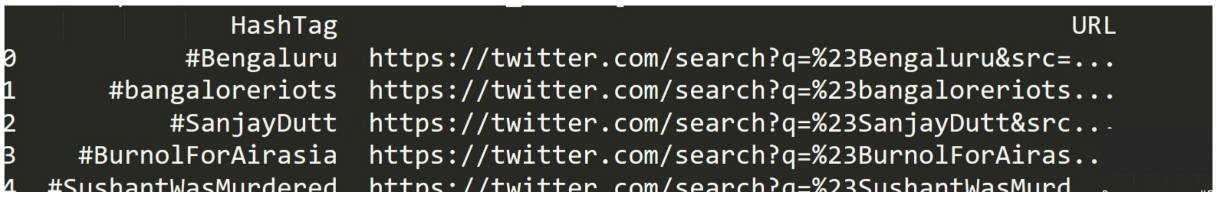
```
dic={'HashTag':fl,'URL':urls}
```

Here, I am giving the column names as **HashTag** and **URL.**

Once we have got the dictionary, we need **to create the dataframe**. To do this, run the following code:

```
import pandas
df=pd.DataFrame(dic)
print(df)
```

On printing out df, this is what the result looks like:



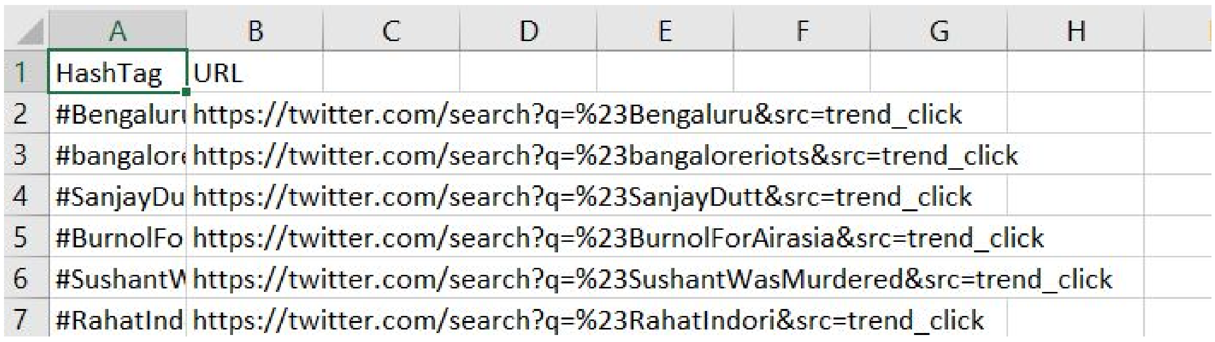
```
               HashTag                                                URL
0            #Bengaluru  https://twitter.com/search?q=%23Bengaluru&src=...
1      #bangaloreriots  https://twitter.com/search?q=%23bangaloreriots...
2          #SanjayDutt  https://twitter.com/search?q=%23SanjayDutt&src...
3    #BurnolForAirasia  https://twitter.com/search?q=%23BurnolForAiras..
4  #SushantWasMurdered  https://twitter.com/search?q=%23SushantWasMurd
```

Now the last thing is **to save this as a csv file.** To do this, run the following code:

```
df.to_csv("C:\\Users\\Sweta\\Twitter_HT.csv",index=False)
```

This will save the dataframe as the csv file at whatever location you have given to it.

This is what i got when i opened my saved file:



	A	B	C	D	E	F	G	H
1	HashTag	URL						
2	#Bengaluru	https://twitter.com/search?q=%23Bengaluru&src=trend_click						
3	#bangalore	https://twitter.com/search?q=%23bangaloreriots&src=trend_click						
4	#SanjayDu	https://twitter.com/search?q=%23SanjayDutt&src=trend_click						
5	#BurnolFo	https://twitter.com/search?q=%23BurnolForAirasia&src=trend_click						
6	#SushantW	https://twitter.com/search?q=%23SushantWasMurdered&src=trend_click						
7	#RahatInd	https://twitter.com/search?q=%23RahatIndori&src=trend_click						

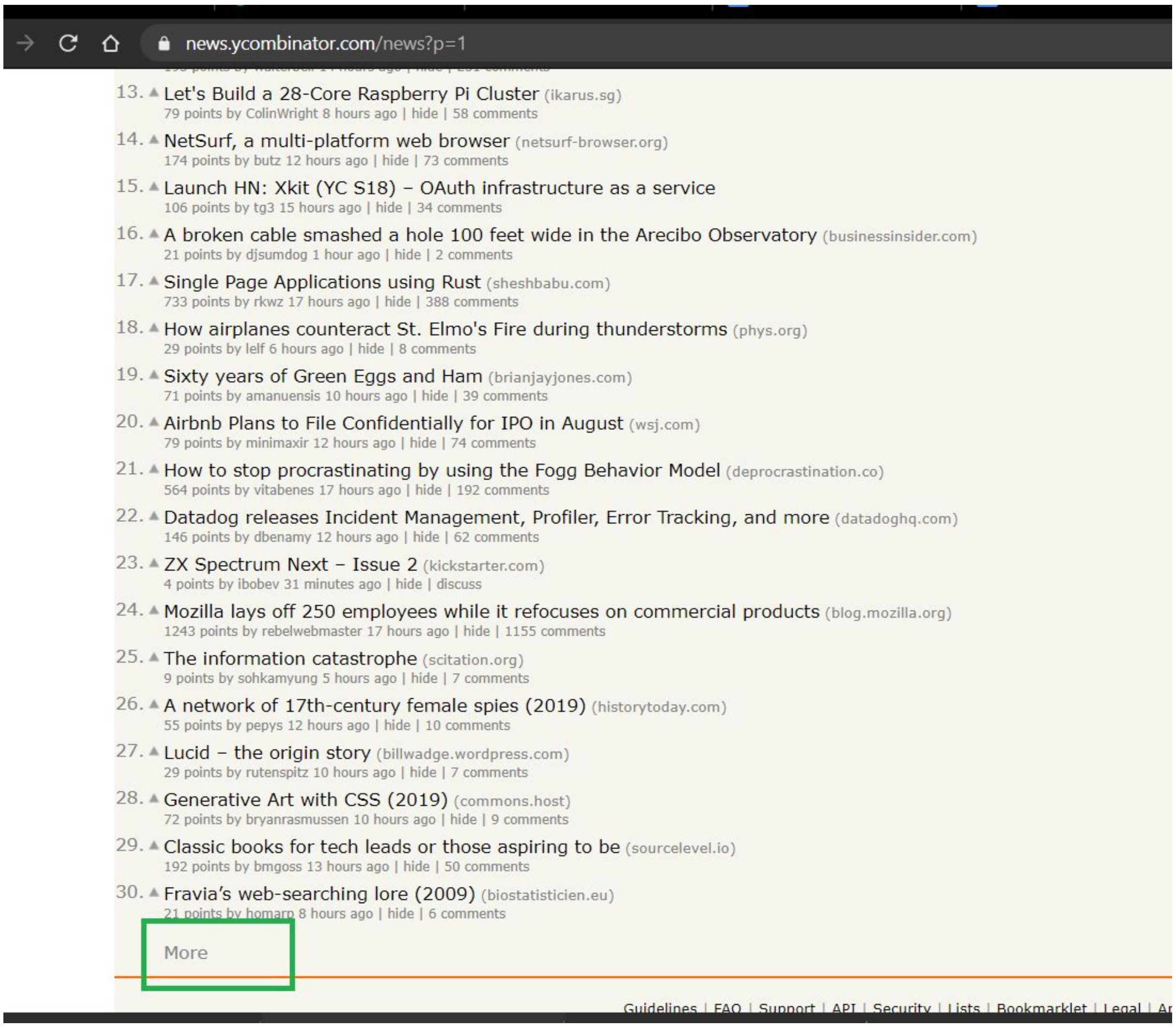
Project 12: News Scraper.

In this project we are going to write a python program which will be scraping the **News Title** and the **link to that news** from the website **Hacker News** (https://news.ycombinator.com/news?p=1).

This is another great example of **web scraping.** Here we are going to use BeautifulSoup once again. So we will start with importing all the important libraries. As I mentioned we will need **BeautifulSoup** for web scraping, we will also need **urllib.request** for getting the source code of the page and **pandas** for working with dataframes as we did in our previous projects. So run the following code to import them.

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
```

Before moving ahead let's look at the page of Hacker News once.



news.ycombinator.com/news?p=1

13. ▲ Let's Build a 28-Core Raspberry Pi Cluster (ikarus.sg)
79 points by ColinWright 8 hours ago | hide | 58 comments

14. ▲ NetSurf, a multi-platform web browser (netsurf-browser.org)
174 points by butz 12 hours ago | hide | 73 comments

15. ▲ Launch HN: Xkit (YC S18) – OAuth infrastructure as a service
106 points by tg3 15 hours ago | hide | 34 comments

16. ▲ A broken cable smashed a hole 100 feet wide in the Arecibo Observatory (businessinsider.com)
21 points by djsumdog 1 hour ago | hide | 2 comments

17. ▲ Single Page Applications using Rust (sheshbabu.com)
733 points by rkwz 17 hours ago | hide | 388 comments

18. ▲ How airplanes counteract St. Elmo's Fire during thunderstorms (phys.org)
29 points by lelf 6 hours ago | hide | 8 comments

19. ▲ Sixty years of Green Eggs and Ham (brianjayjones.com)
71 points by amanuensis 10 hours ago | hide | 39 comments

20. ▲ Airbnb Plans to File Confidentially for IPO in August (wsj.com)
79 points by minimaxir 12 hours ago | hide | 74 comments

21. ▲ How to stop procrastinating by using the Fogg Behavior Model (deprocrastination.co)
564 points by vitabenes 17 hours ago | hide | 192 comments

22. ▲ Datadog releases Incident Management, Profiler, Error Tracking, and more (datadoghq.com)
146 points by dbenamy 12 hours ago | hide | 62 comments

23. ▲ ZX Spectrum Next – Issue 2 (kickstarter.com)
4 points by ibobev 31 minutes ago | hide | discuss

24. ▲ Mozilla lays off 250 employees while it refocuses on commercial products (blog.mozilla.org)
1243 points by rebelwebmaster 17 hours ago | hide | 1155 comments

25. ▲ The information catastrophe (scitation.org)
9 points by sohkamyung 5 hours ago | hide | 7 comments

26. ▲ A network of 17th-century female spies (2019) (historytoday.com)
55 points by pepys 12 hours ago | hide | 10 comments

27. ▲ Lucid – the origin story (billwadge.wordpress.com)
29 points by rutenspitz 10 hours ago | hide | 7 comments

28. ▲ Generative Art with CSS (2019) (commons.host)
72 points by bryanrasmussen 10 hours ago | hide | 9 comments

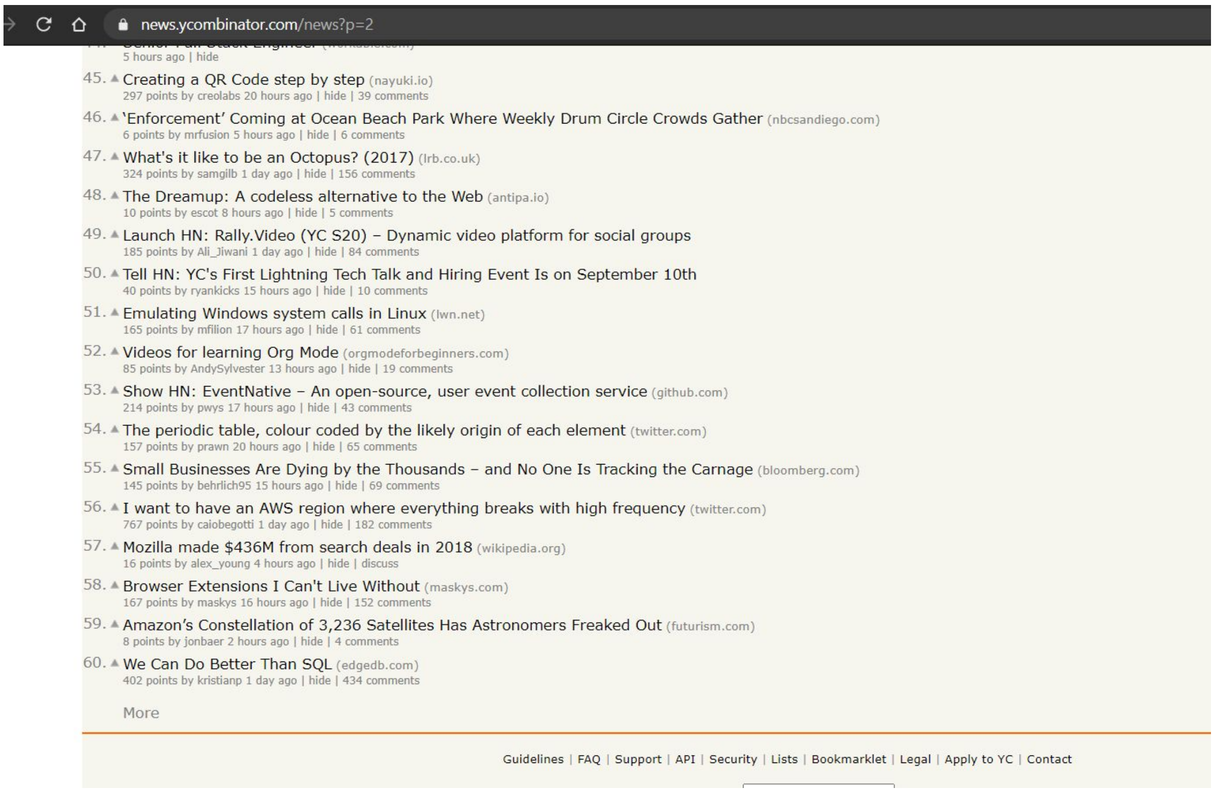
29. ▲ Classic books for tech leads or those aspiring to be (sourcelevel.io)
192 points by bmgoss 13 hours ago | hide | 50 comments

30. ▲ Fravia's web-searching lore (2009) (biostatisticien.eu)
21 points by homarp 8 hours ago | hide | 6 comments

More

Guidelines | FAQ | Support | API | Security | Lists | Bookmarklet | Legal | Ap

So this is the homepage and also the first page of this website which has 30 news in it. If we click on the **More** button at the bottom, it will take me to the next page which has the next 30 news i.e. from 31 to 60.



news.ycombinator.com/news?p=2

5 hours ago | hide

45. ▲ Creating a QR Code step by step (nayuki.io)
297 points by creolabs 20 hours ago | hide | 39 comments

46. ▲ 'Enforcement' Coming at Ocean Beach Park Where Weekly Drum Circle Crowds Gather (nbcsandiego.com)
6 points by mrfusion 5 hours ago | hide | 6 comments

47. ▲ What's it like to be an Octopus? (2017) (lrb.co.uk)
324 points by samgilb 1 day ago | hide | 156 comments

48. ▲ The Dreamup: A codeless alternative to the Web (antipa.io)
10 points by escot 8 hours ago | hide | 5 comments

49. ▲ Launch HN: Rally.Video (YC S20) – Dynamic video platform for social groups
185 points by Ali_Jiwani 1 day ago | hide | 84 comments

50. ▲ Tell HN: YC's First Lightning Tech Talk and Hiring Event Is on September 10th
40 points by ryankicks 15 hours ago | hide | 10 comments

51. ▲ Emulating Windows system calls in Linux (lwn.net)
165 points by mfilion 17 hours ago | hide | 61 comments

52. ▲ Videos for learning Org Mode (orgmodeforbeginners.com)
85 points by AndySylvester 13 hours ago | hide | 19 comments

53. ▲ Show HN: EventNative – An open-source, user event collection service (github.com)
214 points by pwys 17 hours ago | hide | 43 comments

54. ▲ The periodic table, colour coded by the likely origin of each element (twitter.com)
157 points by prawn 20 hours ago | hide | 65 comments

55. ▲ Small Businesses Are Dying by the Thousands – and No One Is Tracking the Carnage (bloomberg.com)
145 points by behrlich95 15 hours ago | hide | 69 comments

56. ▲ I want to have an AWS region where everything breaks with high frequency (twitter.com)
767 points by caiobegotti 1 day ago | hide | 182 comments

57. ▲ Mozilla made $436M from search deals in 2018 (wikipedia.org)
16 points by alex_young 4 hours ago | hide | discuss

58. ▲ Browser Extensions I Can't Live Without (maskys.com)
167 points by maskys 16 hours ago | hide | 152 comments

59. ▲ Amazon's Constellation of 3,236 Satellites Has Astronomers Freaked Out (futurism.com)
8 points by jonbaer 2 hours ago | hide | 4 comments

60. ▲ We Can Do Better Than SQL (edgedb.com)
402 points by kristianp 1 day ago | hide | 434 comments

More

Guidelines | FAQ | Support | API | Security | Lists | Bookmarklet | Legal | Apply to YC | Contact

And this will go on. If you notice the url of different pages then you will notice that the URL actually has 2 parts. One which is **static** i.e. this part doesn't change with each page. This is the part I am talking about 'https://news.ycombinator.com/news?p=' . The next part is just a number which **keeps on changing** as the page changes. And this actually **represents the page number**. i.e. for the first page, it is 1, for the second page, it is 2 and so on.

So we can generalize the **URL** as **the static part + the page number,** where the page '+' sign means the **string concatenation**.

So now we need to take the number of pages, the user is interested in scraping news from, as input. Run the following code for this:

```
p=int(input("Enter the number of pages: "))
```

We are taking the input and storing it in **p**. But before that we are also changing it to integer because I am going to use it numerically.

Now once I have the number of pages, I need to go to each page and scrape the information from that. For this, I need to create a sequence of length equals to the page. The most easy way of doing it is using **range(p).** This will create a sequence of numbers starting from **0** and going till **(p-1)** and the length of this sequence is going to be equal to **p**. Now for each number in this sequence we will **add 1 to it and this will return me the page number**.

Let's understand this with the help of one **example**. Let's say the user entered **p=3.** Now if I run a loop for the sequence **range(p),** the iterator is going to have values **0,1 & 2** for each time the loop is running. If we try to find the page number each time the loop is running, we simply need to add 1 to these numbers. Now if add 1 to the value of the iterator, then the value returning will be **1,2 & 3** which can be used as the page numbers.

So we have understood that we need to run a for loop for range(p) and inside the loop increase the value of the iterator by 1 to get the page number.

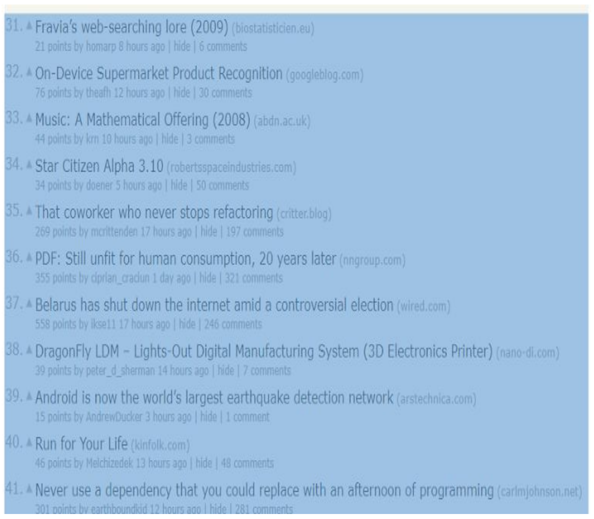
Let's see what we are going to do inside the loop. For each page, first of all we will open that page. I.e. **1st task is to create the url, 2nd is to get the source code of that page. 3rd will be to gather the information.**

So let's look at the code for the first 2 steps.

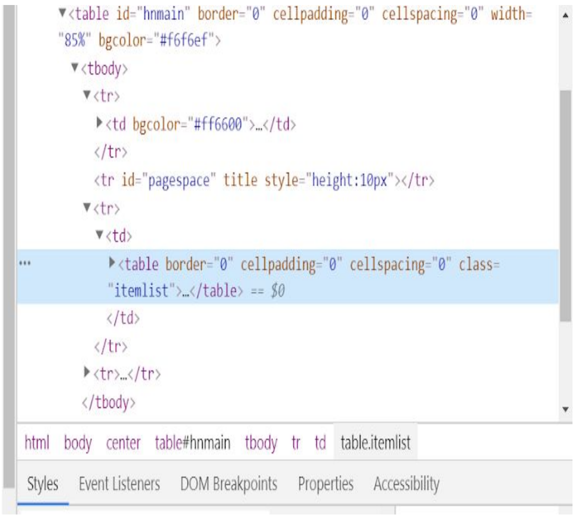
```
for i in range(p):
        url='https://news.ycombinator.com/news?p='+str(i+1)
        r=urllib.request.urlopen(url)
        c=r.read()
        soup=BeautifulSoup(c,'html.parser')
```

Here in the first line we are creating the url. In the next two lines, we are getting the source code of the page and in the last line, we are passing the source code to the BeautifulSoup where the parser will parse it. Now using **soup** we can get all the information from the source code.

Now to gather the information from each page, let's look at the source code of the page.



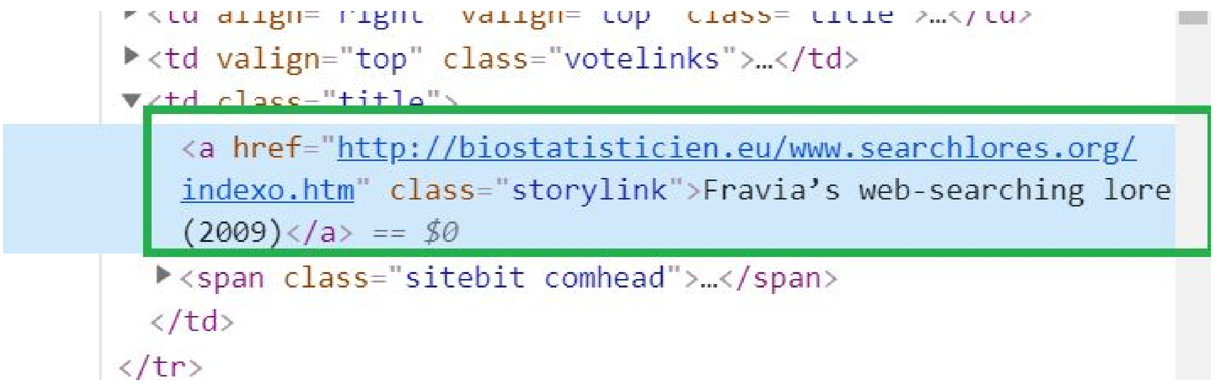
```
31. ▲ Fravia's web-searching lore (2009) (biostatisticien.eu)
      21 points by homarp 8 hours ago | hide | 6 comments
32. ▲ On-Device Supermarket Product Recognition (googleblog.com)
      76 points by theafh 12 hours ago | hide | 30 comments
33. ▲ Music: A Mathematical Offering (2008) (abdn.ac.uk)
      44 points by krn 10 hours ago | hide | 3 comments
34. ▲ Star Citizen Alpha 3.10 (robertsspaceindustries.com)
      34 points by doener 5 hours ago | hide | 50 comments
35. ▲ That coworker who never stops refactoring (critter.blog)
      269 points by mcrittenden 17 hours ago | hide | 197 comments
36. ▲ PDF: Still unfit for human consumption, 20 years later (nngroup.com)
      355 points by ciprian_craciun 1 day ago | hide | 321 comments
37. ▲ Belarus has shut down the internet amid a controversial election (wired.com)
      558 points by ikse11 17 hours ago | hide | 246 comments
38. ▲ DragonFly LDM – Lights-Out Digital Manufacturing System (3D Electronics Printer) (nano-di.com)
      39 points by peter_d_sherman 14 hours ago | hide | 7 comments
39. ▲ Android is now the world's largest earthquake detection network (arstechnica.com)
      15 points by AndrewDucker 3 hours ago | hide | 1 comment
40. ▲ Run for Your Life (kinfolk.com)
      46 points by Melchizedek 13 hours ago | hide | 48 comments
41. ▲ Never use a dependency that you could replace with an afternoon of programming (carlmjohnson.net)
      301 points by earthboundkid 12 hours ago | hide | 281 comments
```



```
▼<table id="hnmain" border="0" cellpadding="0" cellspacing="0" width=
"85%" bgcolor="#f6f6ef">
  ▼<tbody>
    ▼<tr>
      ▶<td bgcolor="#ff6600">…</td>
     </tr>
     <tr id="pagespace" title style="height:10px"></tr>
    ▼<tr>
      ▼<td>
        ▶<table border="0" cellpadding="0" cellspacing="0" class=
         "itemlist">…</table> == $0
        </td>
      </tr>
    ▶<tr>…</tr>
    </tbody>
```

html body center table#hnmain tbody tr td table.itemlist
Styles Event Listeners DOM Breakpoints Properties Accessibility

Here we can see, the list of news is stored inside a **table element with class name 'itemlist'**. Now let's see how one news is stored.



```
▶<td align="right" valign="top" class="title">…</td>
▶<td valign="top" class="votelinks">…</td>
▼<td class="title">
   <a href="http://biostatisticien.eu/www.searchlores.org/
   indexo.htm" class="storylink">Fravia's web-searching lore
   (2009)</a> == $0
  ▶<span class="sitebit comhead">…</span>
  </td>
</tr>
```

You will find out that the news element is actually stored as an **'a' element with class name 'storylink' where the text content is the title of the news and inside the attribute href is the link of that news.** This means in each page we need to find this table with class name 'itemlist' and then each of the a elements with class name 'storylink'

To do this, inside the for loop, I will run the following code:

```
d=soup.find('table',{'class':'itemlist'}).find_all('a',{'class':'storylink'})
```

After running this **d** will be holding all the 'a' elements I am interested in. Now we can iterate over **d** to extract all the information. That means another for loop. Inside this we will extract the text content using **.text** and then the value stored inside the href attribute using **.get('href').** The code for this will look like this.

```
for j in d:
        title=j.text
        link=j.get('href')
```

This will return me all the titles and all the links. Now I will be appending them in lists so that I can create a dataframe with them. So **to go to each page and extract all the information and store them inside list**, run the following code:

```
titles=[]
links=[]
for i in range(p):
    url='https://news.ycombinator.com/news?p='+str(i+1)
    r=urllib.request.urlopen(url)
    c=r.read()
    soup=BeautifulSoup(c,'html.parser')
    d=soup.find('table',{'class':'itemlist'}).find_all('a',{'class':'storylink'})

    for j in d:
        title=j.text
        link=j.get('href')
        titles.append(title)
        links.append(link)
```

With this, **titles will be holding all the news titles and links will be holding all the news links.**

Now I need to make a **dictionary** with these lists which will help me in creating the **dataframe**.

In this dictionary, the key names will be given in such a way that we can make them column names.

So to create a dictionary with these lists, run the following code:

```
dic={'news_title':titles,'URL':links}
```

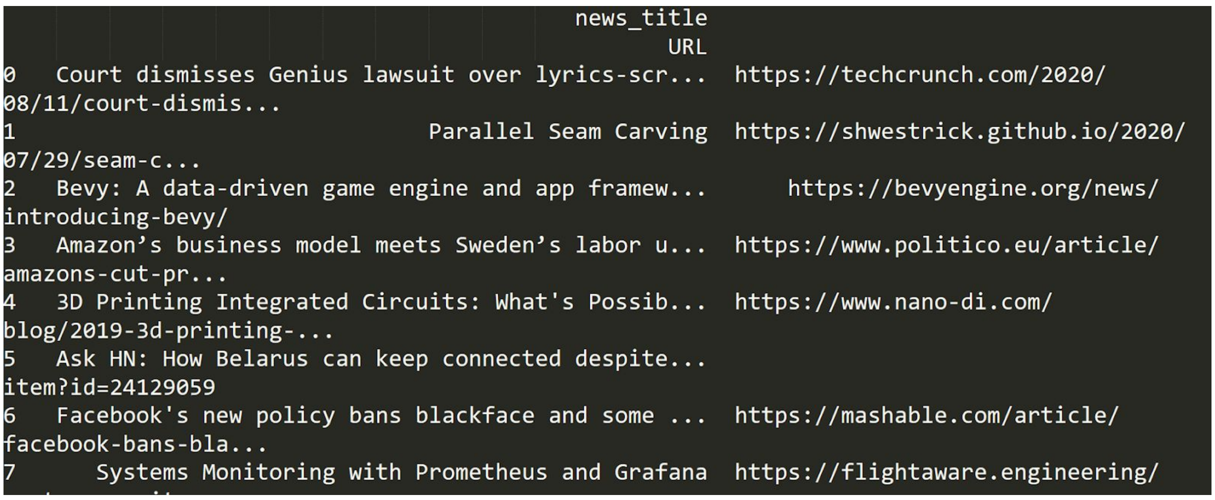
Here I am giving the key names as **news_title** and **URL**. These will be the column names in the dataframe.

Once we have this dictionary we can go on and create the dataframe and then save it as a csv file as done in previous projects. This is the last step of the project

To create the dataframe we will use the function **pd.DataFrame()** and to save it we will use **df.to_csv**. Now to do this, run the following code:

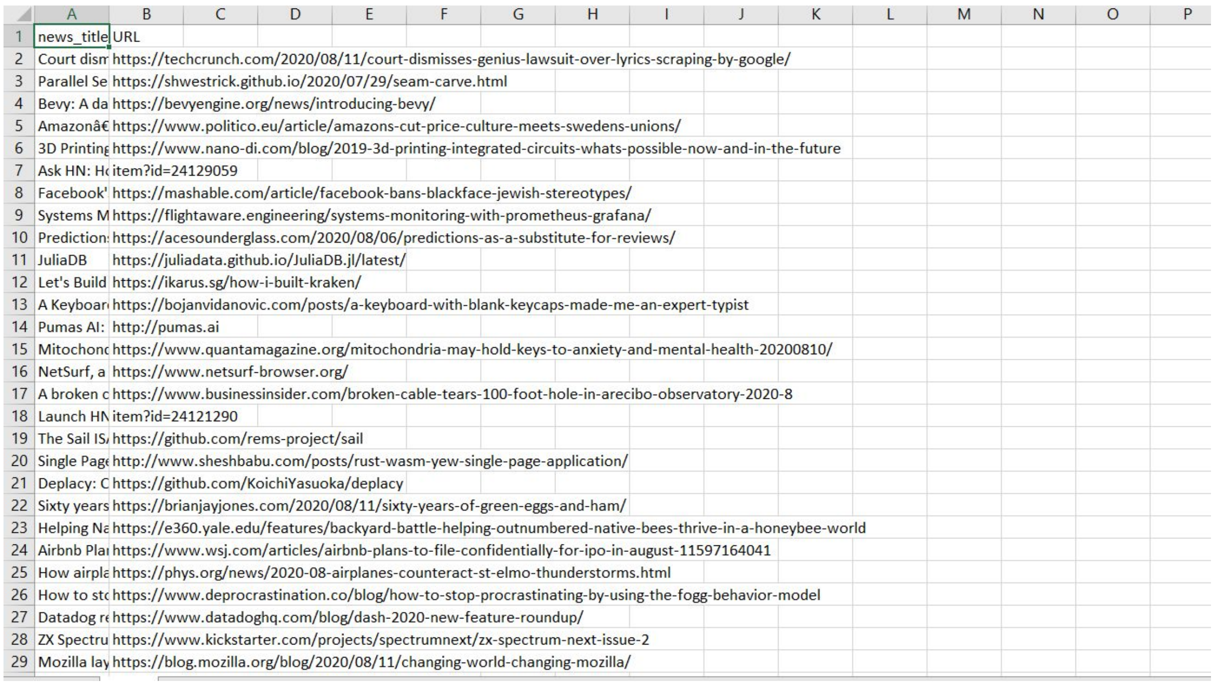
```
df=pd.DataFrame(dic)
print(df)

df.to_csv("C:\\Users\\Sweta\\news.csv",index=False)
print("Done")
```

When we try to print the dataframe for **p=1,** this is the result(a part only):



```
                                           news_title
                                                  URL
0   Court dismisses Genius lawsuit over lyrics-scr...  https://techcrunch.com/2020/
08/11/court-dismis...
1                              Parallel Seam Carving  https://shwestrick.github.io/2020/
07/29/seam-c...
2   Bevy: A data-driven game engine and app framew...      https://bevyengine.org/news/
introducing-bevy/
3   Amazon’s business model meets Sweden’s labor u...  https://www.politico.eu/article/
amazons-cut-pr...
4   3D Printing Integrated Circuits: What's Possib...  https://www.nano-di.com/
blog/2019-3d-printing-...
5   Ask HN: How Belarus can keep connected despite...
item?id=24129059
6   Facebook's new policy bans blackface and some ...  https://mashable.com/article/
facebook-bans-bla...
7      Systems Monitoring with Prometheus and Grafana  https://flightaware.engineering/
```

And if you go to the file that u created, **news.csv** in this case, this is what it will look like:



	A	B
1	news_title	URL
2	Court dism	https://techcrunch.com/2020/08/11/court-dismisses-genius-lawsuit-over-lyrics-scraping-by-google/
3	Parallel Se	https://shwestrick.github.io/2020/07/29/seam-carve.html
4	Bevy: A da	https://bevyengine.org/news/introducing-bevy/
5	Amazonâ€	https://www.politico.eu/article/amazons-cut-price-culture-meets-swedens-unions/
6	3D Printing	https://www.nano-di.com/blog/2019-3d-printing-integrated-circuits-whats-possible-now-and-in-the-future
7	Ask HN: Hc	item?id=24129059
8	Facebook'	https://mashable.com/article/facebook-bans-blackface-jewish-stereotypes/
9	Systems M	https://flightaware.engineering/systems-monitoring-with-prometheus-grafana/
10	Prediction:	https://acesounderglass.com/2020/08/06/predictions-as-a-substitute-for-reviews/
11	JuliaDB	https://juliadata.github.io/JuliaDB.jl/latest/
12	Let's Build	https://ikarus.sg/how-i-built-kraken/
13	A Keyboar	https://bojanvidanovic.com/posts/a-keyboard-with-blank-keycaps-made-me-an-expert-typist
14	Pumas AI:	http://pumas.ai
15	Mitochonc	https://www.quantamagazine.org/mitochondria-may-hold-keys-to-anxiety-and-mental-health-20200810/
16	NetSurf, a	https://www.netsurf-browser.org/
17	A broken c	https://www.businessinsider.com/broken-cable-tears-100-foot-hole-in-arecibo-observatory-2020-8
18	Launch HN	item?id=24121290
19	The Sail IS	https://github.com/rems-project/sail
20	Single Page	http://www.sheshbabu.com/posts/rust-wasm-yew-single-page-application/
21	Deplacy: C	https://github.com/KoichiYasuoka/deplacy
22	Sixty years	https://brianjayjones.com/2020/08/11/sixty-years-of-green-eggs-and-ham/
23	Helping Na	https://e360.yale.edu/features/backyard-battle-helping-outnumbered-native-bees-thrive-in-a-honeybee-world
24	Airbnb Pla	https://www.wsj.com/articles/airbnb-plans-to-file-confidentially-for-ipo-in-august-11597164041
25	How airpla	https://phys.org/news/2020-08-airplanes-counteract-st-elmo-thunderstorms.html
26	How to sto	https://www.deprocrastination.co/blog/how-to-stop-procrastinating-by-using-the-fogg-behavior-model
27	Datadog re	https://www.datadoghq.com/blog/dash-2020-new-feature-roundup/
28	ZX Spectru	https://www.kickstarter.com/projects/spectrumnext/zx-spectrum-next-issue-2
29	Mozilla lay	https://blog.mozilla.org/blog/2020/08/11/changing-world-changing-mozilla/

Project 13:PDF to text converter bot.

In this project we will be creating a python bot which will convert any pdf file to text file.

Well first we need to search for a module that will help us in converting pdfs. Let's search for it on the internet. Well here we get a module called pdfminer
Well pdfs are actually difficult to handle as compared to text files because they are somewhat complex and tricky to deal with. pdfminer comes to a rescue in this case.
PDFMiner is basically a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting

and analyzing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. So as we are going to use a new module the first step should be to install it right?? Let's use pip for that

```
pip install pdfminer
```

Well now we need some important modules that we are going to use. Let's first have a look at them

PDFResourceManager - It is used to store shared resources of a file like fonts or images.
PDFPageInterpreter - It is used to process the page contents .You know that here we are going to work page by page , so the PDFPageInterpreter becomes important in this case.
PDFPage - It is used to create an object that is going to hold information about the page. Again this becomes important since we will be dealing with the pages of the original file.
TextConverter- It is used to convert any pdf into text.This will be passed to the PageInterpreter object.
LAParams - It takes care of the layout of the pdf file.

Now let's import them and start our coding

Now we will import PDFResourceManager and OPDFPageInterpreter from pdfminer.pdfinterp, PDFPage from pdfminer.pdfpage, TextConverter from pdfminer.converter and finally LAParams from pdfminer.layout

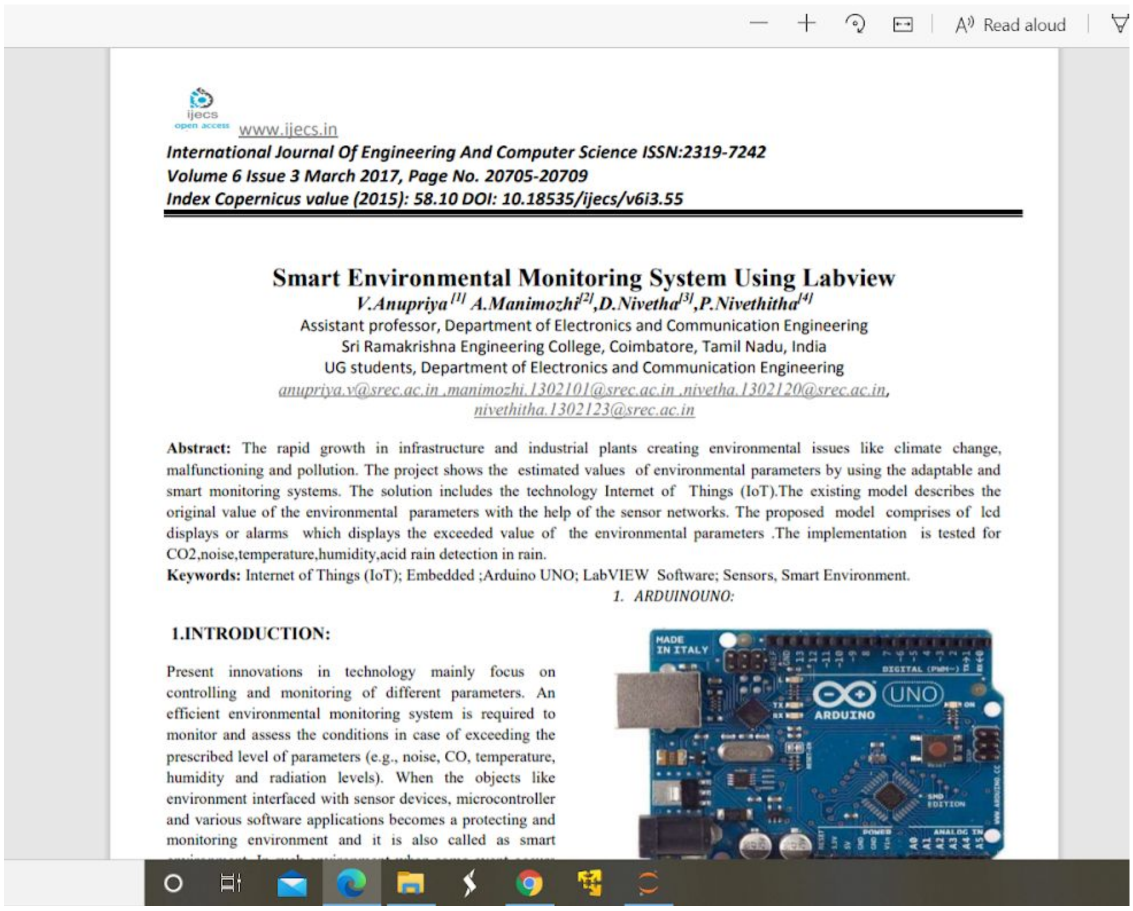
Let's write the code

```
from pdfminer.pdfinterp import PDFResourceManager,
PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
import io
```

Now you might be wondering what io is. Let’s move forward for the time being and in the process find out the use of io as well. We need to open the PDF file now.

```
path='C:\Users\Aoishi\Downloads\literature_survey2.pdf'
pdf = open(path, 'rb')
```

So the pdf file looks like



www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 3 March 2017, Page No. 20705-20709
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i3.55

Smart Environmental Monitoring System Using Labview

V.Anupriya [1] A.Manimozhi[2],D.Nivetha[3],P.Nivethitha[4]
Assistant professor, Department of Electronics and Communication Engineering
Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India
UG students, Department of Electronics and Communication Engineering
anupriya.v@srec.ac.in ,manimozhi.1302101@srec.ac.in ,nivetha.1302120@srec.ac.in, nivethitha.1302123@srec.ac.in

Abstract: The rapid growth in infrastructure and industrial plants creating environmental issues like climate change, malfunctioning and pollution. The project shows the estimated values of environmental parameters by using the adaptable and smart monitoring systems. The solution includes the technology Internet of Things (IoT).The existing model describes the original value of the environmental parameters with the help of the sensor networks. The proposed model comprises of lcd displays or alarms which displays the exceeded value of the environmental parameters .The implementation is tested for CO2,noise,temperature,humidity,acid rain detection in rain.

Keywords: Internet of Things (IoT); Embedded ;Arduino UNO; LabVIEW Software; Sensors, Smart Environment.

1. ARDUINOUNO:

1.INTRODUCTION:

Present innovations in technology mainly focus on controlling and monitoring of different parameters. An efficient environmental monitoring system is required to monitor and assess the conditions in case of exceeding the prescribed level of parameters (e.g., noise, CO, temperature, humidity and radiation levels). When the objects like environment interfaced with sensor devices, microcontroller and various software applications becomes a protecting and monitoring environment and it is also called as smart

So we basically provide the path and opens it in the rb mode
Now we need to generate the interpreter. So it follows the syntax

```
object_name=PDFPageInterpreter(resource_manager_object,converter_object)
```

Thus we need to create objects of PDFResourceMangaer and TextConverter first

So we create the object of PDFResourceManager as

```
rm=PDFResourceManager()
```

For TextConverter object the syntax is
object_name=TextConverter(resource manager object, in memory stream, laparam_object)

Well now how to create an in memory stream object. Remember we imported io at the beginning. This comes handy here
So the object can be simply created using io.StringIO().

Now to create a LAParam object we need to simple type
lp=LAParams()
So now the code looks like

```
mem=io.StringIO()
lp=LAParams()
cnv = TextConverter(rm,mem ,  Laparams=lp)
```

Now we create the PDFPageInterpreter object

```
ip = PDFPageInterpreter(rm, cnv)
```

Now we will proceed towards dealing with individual pages.

```
for i in PDFPage.get_pages(pdf):
    ip.process_page(i)
    text =  mem.getvalue()
```

So basically we are iterating over all the pages of the pdf taking them one at a time and processing them using .process_page(). For each page after the conversion takes place the content is converted to text and its stored in mem i.e the in memory string. So we need to get the value stored and for that we use .getvalue() on mem. Now this will be repeated for all the pages and now finally we need to save it in a txt format. So we use

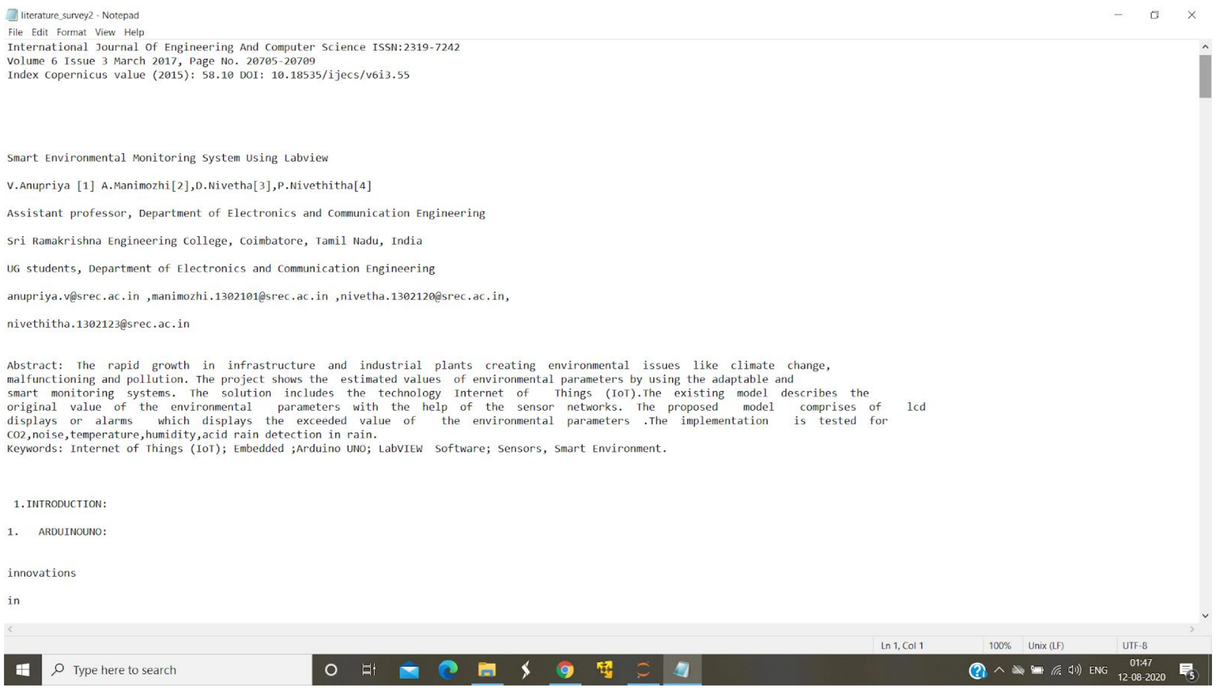
```
file=open(‘C:\Users\Aoishi\Downloads\literature_survey2.txt',’wb’)
file.write(text)
```

Now the problem with this is that it will throw an exception because the text is in str form but for write we need to pass a str like object. So we need to encode this

So we will update the code as

```
file=open(‘C:\Users\Aoishi\Downloads\literature_survey2.txt',’wb’)
file.write(text.encode('utf-8'))
```

Now after successful conversion we get



```
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 3 March 2017, Page No. 20705-20709
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i3.55



Smart Environmental Monitoring System Using Labview

V.Anupriya [1] A.Manimozhi[2],D.Nivetha[3],P.Nivethitha[4]

Assistant professor, Department of Electronics and Communication Engineering

Sri Ramakrishna Engineering College, Coimbatore, Tamil Nadu, India

UG students, Department of Electronics and Communication Engineering

anupriya.v@srec.ac.in ,manimozhi.1302101@srec.ac.in ,nivetha.1302120@srec.ac.in,

nivethitha.1302123@srec.ac.in


Abstract:  The  rapid  growth  in  infrastructure  and  industrial  plants  creating  environmental  issues  like  climate  change,
malfunctioning and pollution. The project shows the  estimated values  of environmental parameters by using the adaptable and
smart  monitoring  systems.  The  solution  includes  the  technology  Internet  of   Things  (IoT).The  existing  model  describes  the
original  value  of  the  environmental   parameters  with  the  help  of  the  sensor  networks.  The  proposed   model   comprises  of   lcd
displays  or  alarms   which  displays  the  exceeded  value  of   the  environmental  parameters  .The  implementation   is  tested  for
CO2,noise,temperature,humidity,acid rain detection in rain.
Keywords: Internet of Things (IoT); Embedded ;Arduino UNO; LabVIEW  Software; Sensors, Smart Environment.


 1.INTRODUCTION:

1.   ARDUINOUNO:


innovations

in
```

See its converted to the text format.

You can try out the similar conversion using HTMLConverter and XMLConverter to convert to HTML and XML files respectively

Project 14:Email Checker.

In this project we are going to write a program to return me the number of unread mails in the gmail account.

So let’s begin. The first step is to connect with **Gmail.** For this I am going to use the library **imaplib**. Let’s first see **what IMAP is.** IMAP stands for **Internet Mail Access Protocol.** imaplib is an **in-built python library.** That means no need to install it. We will simply import it to use.imaplib is used for accessing emails over IMAP. So first of all we will have to import it. To do so, run the following code:

```
import imaplib
```

Now coming back to the first step, we need to establish a connection with Gmail. For that we are going to use **imaplib.IMAP4_SSL()** method of imaplib. This method sets up the socket connection with the imap gmail server. Here we need to mention the host address of the server. Here our host address is **‘imap.gmail.com’.** Now to make the connection through our program, run the following code:

```
g=imaplib.IMAP4_SSL('imap.gmail.com')
```

Once the connection is established, we need to log in to the user’s account for which we require **the email id and password of the gmail account**. So it’s time to take those as input from the user. For that run the following code:

```
em=input("Enter the Mail ID: ")
pw=input("Enter the Password: ")
```

Here whatever inputs are given by the user are stored inside the variables em and pw. Using this information, we will be logging in to the account through our program.

Now to login with the gmail through our program, we will have to use the method **.login().** This function takes the mail id and password as arguments and logs in to the account. Run the following code to log in to the user’s gmail account:

```
g.login(em,pw)
```

This will log in to the user’s account. Now we need to select the ***mailbox.*** Here i=our mailbox is the **‘INBOX’** and the code for this is pretty simple:

```
g.select('INBOX')
```

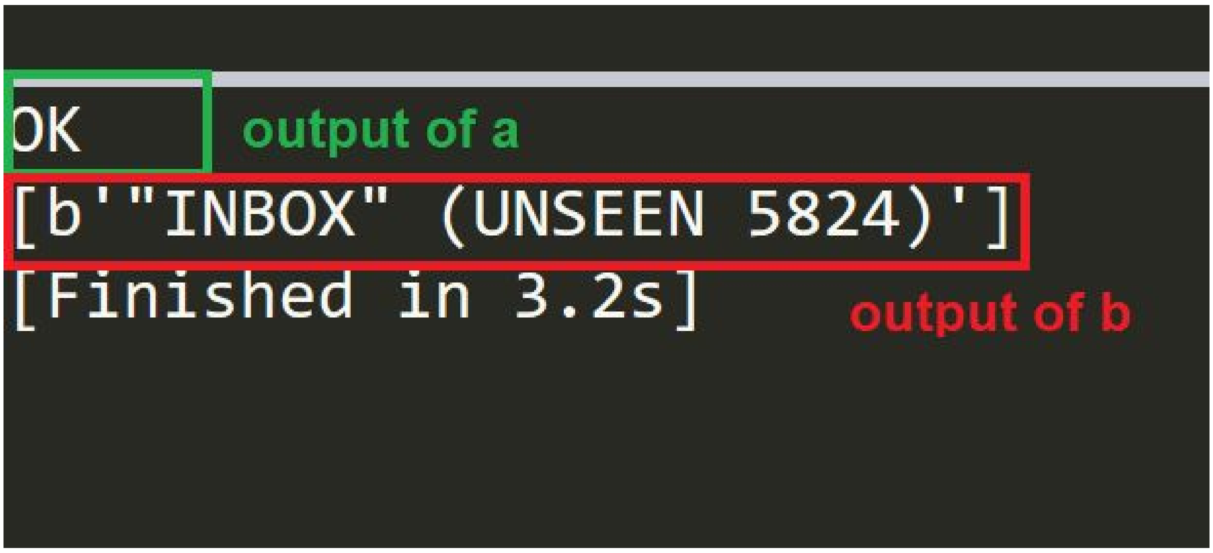
This will select the INBOX and now whatever you do with ***g,*** is going to be executed on INBOX. Now comes the final step of the project. Now we need to find **the number of unread messages.** To do this, we will be using the function **.status(‘mailbox’,’names’).** This requests the named status conditions for the mailbox. In this case, the mailbox is **‘INBOX**’ and the named condition is **‘(UNSEEN)’.** Now to get the number of unread messages, run the following code:

```
a,b=g.status('INBOX','(UNSEEN)')
```

This function will be returning me **2 values** which I am holding in **a** & **b.** The first returned value is actually **the status of this code execution**. If their code was executed properly then it will return **OK.** The second returned value holds **the information about the unread messages**. Now to see the actual result, let’s try to print **a** & **b** using the following code:

```
print(a)
print(b)
```

The output of it will look something like this:



```
OK
[b'"INBOX" (UNSEEN 5824)']
[Finished in 3.2s]
```

Here a is printing out **OK,** this means the code execution was successful. Now let's look at the output of b. This is coming out as **a list with one element.** Now to extract the number of unread messages from this whole output, first of all we have to take this element and convert this to string and then use slicing to just get the number at the end. For this run the following code:

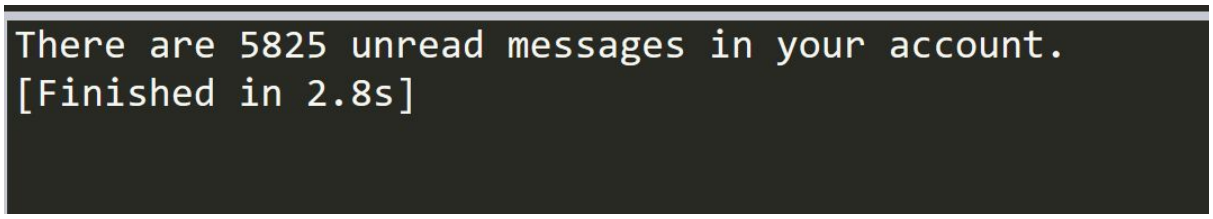
```
c=str(b[0])
n=c[18:22]
```

Here in the first line of code, we are converting the element at 0th index in b to string. In the second line of code, we are doing the meaningful slicing. Now **n** will be holding the number here.

Now as the last step, we will be showing one statement as the output to return the number of unread messages. To do this, run the following code:

```
print("There are "+n+" unread messages in your account.")
```

And here is the result of this line of code:



```
There are 5825 unread messages in your account.
[Finished in 2.8s]
```