Problem statement:

Need to develop a web crawler completely from scratch, below is the guidelines:

- Given a starting URL, program will extract all links on the page.
- Then it will iterate over all the new links and gather new links from the new pages. Limit to the web site that are parsing and do not download links pointing to other websites.
- Store the webpages in a directory with the date.
- Based on the links generate a network graph of the downloaded web site and visualize it. Nodes are web pages and edges are links between them.

- Write a small program that counts the number of occurrences of a given word in the paragraphs of the web pages downloaded. A paragraph in HTML code is given by Some text .
- Modify the program so that when run on another date, it will onlyDownload fresh pages (pages that were not changed since last visit; note that all pages are stored in a directory).
- Monitor the response time of the web site using ping. Write a program that pings the web site every 10 minutes. The results of the ping latency will be displayed.

Prescription:

To achieve the problem statement the solution has divided into several steps where it is considered that given URL is 'https://www.newegg.com/global/no-en/p/pl?d=processor '

Below is the high level steps:

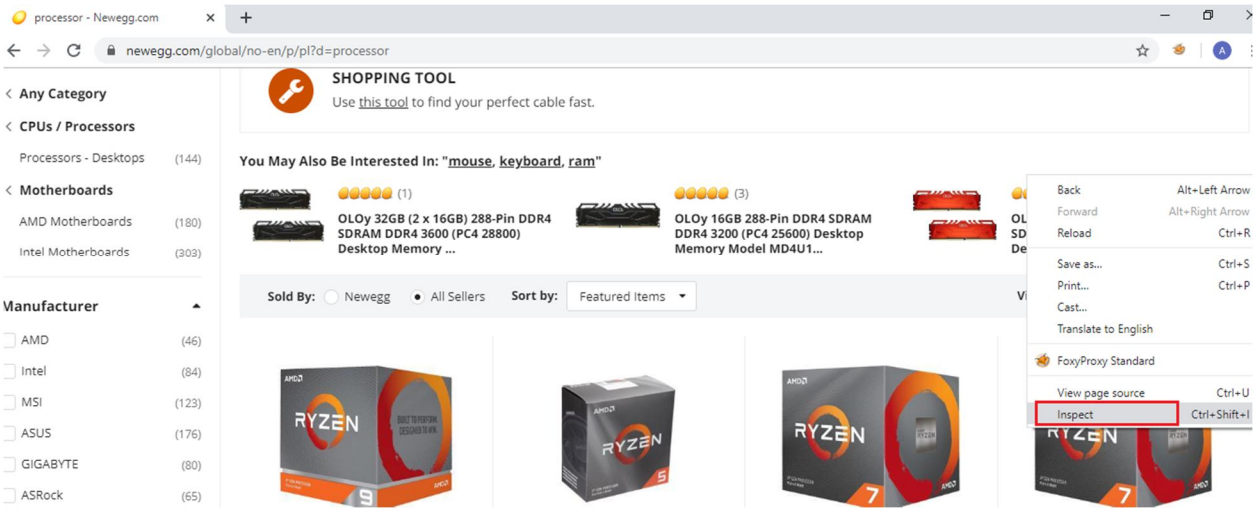
- Inspect the website and find unique location of data
- Then , create a web crawlaer to pull out desired weblinks from given URL, storage them into local directory file where the filename is consists with date
- After that, create a web crawlaer to iterate over the storage links from local directory file and search for a given word 'promo@promo.global.newegg.com' and display the number of count of given word.
- Finally, revisit and modify program based on website 'last-modified' date option.
- Besides, monitor the response time of the given website in every 10minute and save the output in a file for display.
- Visulaize a network graph using networkx and matplotlib.pyplot

Section 2

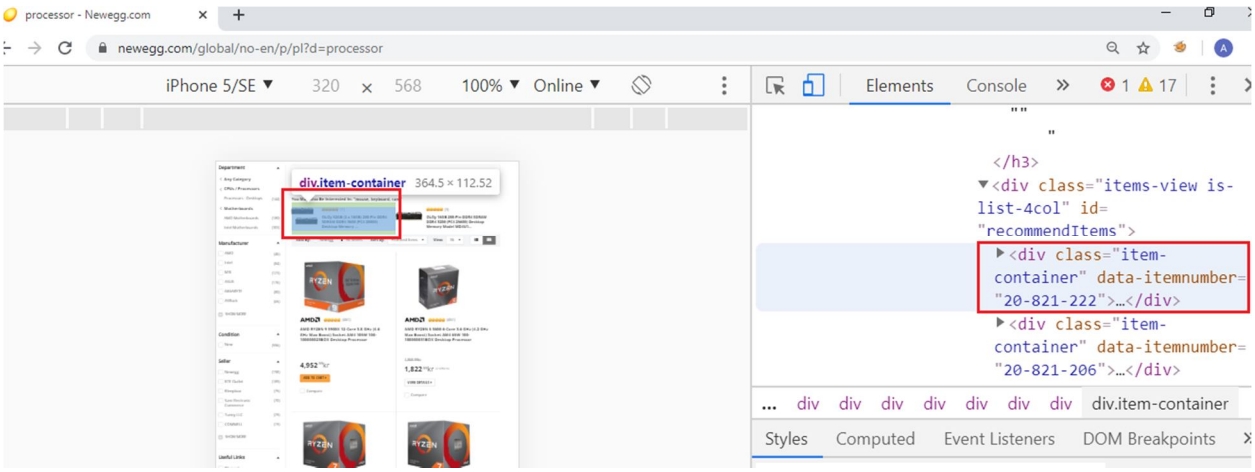
Implementation:

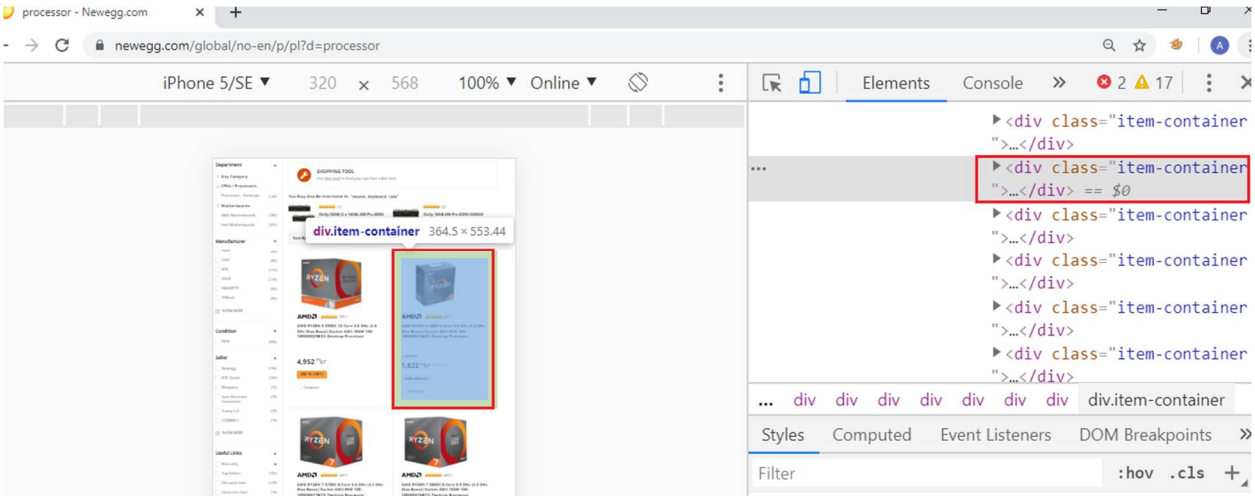
1. **Inspecting the Webpage**

To be started, let inspect the webpage (https://www.newegg.com/global/no-en/p/pl?d=processor) like the screenshot



At inspection HTML code , once hovered on the inspect page code then corresponding blue box sorrounded the product on the web page as showing below screenshot. **From inspection it is found that product links is inside of HTML div tag which is div class=”item-container”**





Now it is known that the unique location of data which is div class=”item-container” with the help of class tag.

2. Jump into the Code

Step 2.1 Ensure that necessary package are installed at OS and required library module has been imported at python script file, as showing below:

#installs pthon & python-bs4

```
apt-get update
apt-get install –y python
apt-get install –y python-bs4
```

---declare the source header at the start----

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

#urllib2 is a Python module for fetching URLs

```
from urllib2 import urlopen as Urequest
```

#Beautiful Soup is a Python library for pulling data out of HTML and XML files.

```
from bs4 import BeautifulSoup as soup
```

#datetime & time module provides various time-related functions

```
import datetime
import time
```

#The subprocess module allows us to spawn processes, connect to their input/output/error pipes,and obtain their return codes.

```
import subprocess
```

#getwcd, chdir and listdir are miscellaneous operating system interfaces to explore directories and files

```
from os import getcwd
from os import chdir
from os import listdir
```

Step 2.2 declares a variable for the url of the page.

#specify the URL

```
min_url="https://www.newegg.com/global/no-en/p/pl?d=processor"
```

#make use of the Python urllib2 urlopen to get the HTML page read of the url declared and put the value in a variable

N.B: urllib2 urlopen module was imported as name 'Urequest'

```
uclient = Urequest(min_url)
page_html= uclient.read()
uclient.close()
```

Step 2.3 Use python HTML parser module BeautifulSoup to extract data

From inspection result it was found that product links is inside of HTML div tag which is div class="item-container". So use beautiful soup **find_all()** method to collect all elements from html page which tag as **div class="item-container"** and saved in a variable **'div_processor'**

N.B: BeautifulSoup module imported as name 'soup'

```
page_soup=soup(page_html,"html.parser")
div_class=page_soup.findAll("div",{"class":"item-container"})
div_processor = div_class
```

Step 2.4 Export data to storage

Now data is ready, it is time to save it in a local directory file where filename is consists with date.

To do that, first need to record the current date and record it then need to open a file at write mode and do a iteration over the variable **'div_processor'** who contains all the data of div class="item-container".

#check the date realtime output of linux OS and saved it as format 'Day Date Month Year' format in a variable

```
dtdate2=subprocess.Popen(["date | awk -F' ' '{print
$1,$3,$2,$6}'"],shell=True,stdout=subprocess.PIPE)
line2 = dtdate2.stdout.readline().rstrip()
```

#open two file as write mode in a local directory name as " f " & "g " where

- **file " f " name being 'Day , Date Month Year.txt' format , this file save gathered web links for storage and further crawling and searching**
- file " g " name being 'links_plot.txt' and save gathered web links for network graph purpose later.

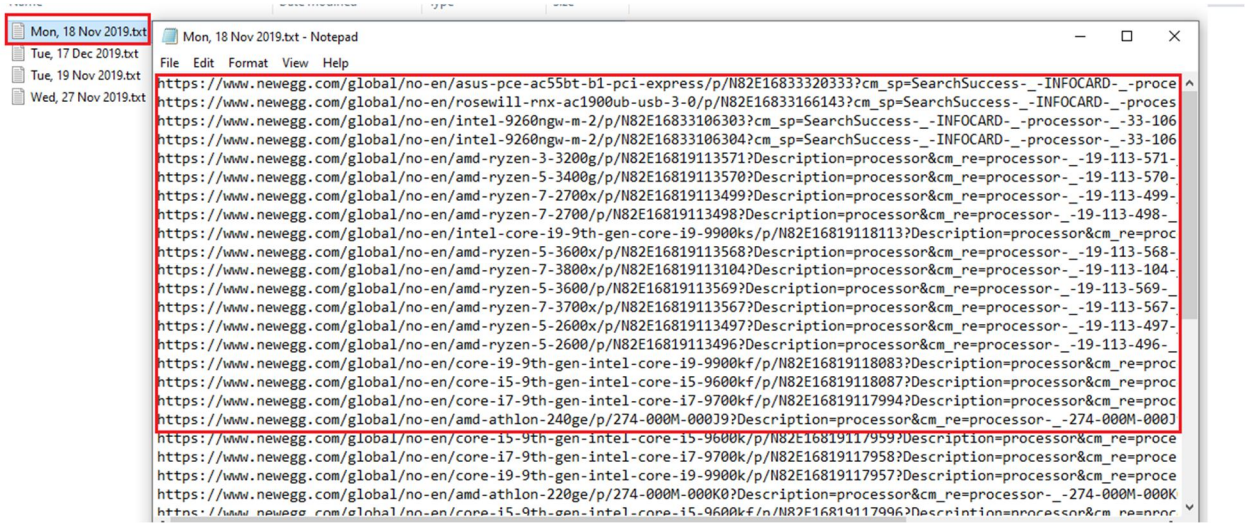
```
f =open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','w')
g =open ('/mnt/d/projectlinks/links_plot.txt','w')
```

Do a iteration over the variable **div_processor** who contains all the data of div class="item-container" and search for all **a** element with **href** attribute and saved in a variable name '**webpages**'. Finally, do a iteration over variable '**webpages**' and write the weblinks information in above two files and close the files accordingly.

```
    for container in div_processor:
      product_link=container.a["href"]
      webpages = product_link+'\n'
      for i in webpages:
         f.write(i)
         g.write (i)
    f.close()
```

```
    g.close()
```

When, finally run the script then it being able to see the all the gathered weblinks data into the file like below screenshots:



Step 2.5 Crawling over saved web links and sum how many links downloaded and search for a particular word form paragraph from each weblink and sum the count of word.

Here, first Open file “h” at read mode, **this file already been created in step 2.4 and contains all crawled weblinks from webpage**.

Then, Define given word **'promo@promo.global.newegg.com'** in a variable . this word will be crawled over gathered weblinks. And array **‘word_list []’** used to save the number of occurrence of given word and finally **list.count()** method use to get the sum of count.

Next, define another word ‘https’ and list download_count[] has been used to count how many links have been downloaded during step 2.4

```
h = open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','r')
    word='promo@promo.global.newegg.com'
    word_list=[]
    word2='https'
    download_count=[]
```

After define part complete , a iteration operated over opened file **“h”** and saved each line in a variable **‘min_text’** where check if word **‘https’** is in the variable, then it first append the word ‘https’ in list ‘download_count[]’ . Then it did urllib urlopen to read the link page and use beautiful soup method **findAll('p')** to search paragraph on the website and **join()** method takes all items in an iterable and joins them into one string and saved in a variable **“k”**. After that, it was checked that whether the string has given word 'promo@promo.global.newegg.com' or not if have then it append the word in the list **‘word_list []’** . Finally when iteration finish then list.count() method has been used to count the quantity of given word ‘promo@promo.global.newegg.com ‘ and word ‘https’ in order to print the number of occurrence of given word ‘promo@promo.global.newegg.com’ and number of downloaded links respectively.

```
     for i in h :
      min_text=i
      if word2 in min_text:
         download_count.append(word2)
         uclient = Urequest(min_text)
         page= uclient.read()
         uclient.close()
         page_text=soup(page,"html.parser")
         for i in page_text.findAll('p'):
           k = ''.join(i.findAll(text=True))
           if word in k:
            word_list.append(word)
    a=word_list.count(word)
    b=download_count.count(word2)
    print "Given word is: "+word+" The number of counts of given word are: " +str(a)
    print "Number of downloaded links are:"+str(b)+". The download path is
/mnt/d/python/project/"+line2[0:3]+','+line2[3:]+'.txt'
    h.close()
```

When, finally run the script then it being displayed information like below screenshot:

Number of downloaded links and download path and filename

Number of occurrence of given word ‘promo@promo.global.newegg.com’

```
root@DESKTOP-37QUITS:/home/asif/python_project# ./python_project.py_modi_nov18
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
Number of downloaded links are:40. The download path is /mnt/d/python/project/Wed, 27 Nov 2019.txt
Given word is: promo@promo.global.newegg.com The number of counts of given word are: 40
root@DESKTOP-37QUITS:/home/asif/python_project#
```

```
root@DESKTOP-37QUITS:/home/asif/python_project# ./web_crawl.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
Number of downloaded links are:40. The download path is /mnt/d/python/project/Tue, 17 Dec 2019.txt
Given word is: promo@promo.global.newegg.com The number of counts of given word are: 40
root@DESKTOP-37QUITS:/home/asif/python_project#
```

Step 2.6 Modify program based on website 'last-modified' date

Here, the program was Revisited and modified and put condition to check whether given website has 'last-modified' tag or not , if have whether download fresh page again or not , otherwise download fresh page again. **It is need to be mention that given website 'https://www.newegg.com/global/no-en/p/pl?d=processor" not have 'last-modified' tag**, as showing below screenshot ,so every time it will download fresh page of that date

Screenshot showing that 'https://www.newegg.com/global/no-en/p/pl?d=processor" has no 'last-modified' tag , during check by command **curl –I https://www.newegg.com/global/no-en/p/pl?d=processor**

```
root@DESKTOP-37QUITS:/home/asif/python_project# curl -I 'https://www.newegg.com/global/no-en/p/pl?d=processor' | grep -i '
last-modified'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
root@DESKTOP-37QUITS:/home/asif/python_project#
```

```
root@DESKTOP-37QUITS:/home/asif/python_project# curl -I 'https://www.newegg.com/global/no-en/p/pl?d=processor'
HTTP/2 200
content-type: text/html; charset=utf-8
nel: {"report_to": "default", "max_age": 2592000}`);
report-to: {"group":"default","max-age":10886400,"include_subdomains":"true","endpoints":[{"url":"https://pf.newegg.com/csp"}]};
pragma: no-cache
ga: 0
link: <https://assets.adobedtm.com>;rel=preconnect,<https://c1.neweggimages.com>;rel=preconnect,<https://promotions.newegg.com>;rel=preconnect,<https://www2.newegg.com>;rel=preconnect,<https://promotions.newegg.com>;rel=preconnect,<https://pf.newegg.com>;rel=preconnect,<https://ec-apis.newegg.com>;rel=preconnect,<https://sealserver.trustwave.com>;rel=preconnect,<https://help.newegg.com>;rel=preconnect,<https://states.newegg.com>;rel=preconnect,<https://fonts.gstatic.com>;rel=preconnect,<https://secure.newegg.com>;rel=preconnect,<https://tags.tiqcdn.com>;rel=preconnect,<https://images10.newegg.com>;rel=preconnect
x-server-id: 211
x-frame-options: SAMEORIGIN
x-ver: 12091901
x-newegg-flow: MISS
x-newegg-index: 0
content-security-policy: upgrade-insecure-requests
cache-control: no-cache
expires: Tue, 17 Dec 2019 17:16:48 GMT
date: Tue, 17 Dec 2019 17:16:48 GMT
set-cookie: NVTC=248326808.0001.6yznoex0x.1576603007.1576603007.1576603007.1; domain=.newegg.com; expires=Fri, 16-Dec-22 17:16:47 GMT; path=/;secure
set-cookie: NID=9D6I72344M1j6I726I; domain=.newegg.com; expires=Fri, 16-Dec-22 17:16:47 GMT; path=/;secure
set-cookie: NV_NVTCTIMESTAMP=1576603008; domain=.newegg.com; expires=Fri, 16-Dec-22 17:16:47 GMT; path=/;secure
set-cookie: NSC_mc-xxx.ofxfhh.dpn-vsmibti=30dfa3db25aec80e16f76639e55a56ccf9395fd0f6eca2787b8db5dd62231fa403b167d1;expires=Tue, 17-Dec-2019 17:44:58 GMT;path=/;secure
strict-transport-security: max-age=31536000

root@DESKTOP-37QUITS:/home/asif/python_project#
```

Though website has no 'last-modified' tag but below condition has put by considering if website had 'last-modified' tag:

curl –I given website and grep 'last-modified' and take the output as standard output and put the value in a variable

```
command="curl -I 'https://www.newegg.com/global/no-en/p/pl?d=processor' | grep -i last-modified"
p = subprocess.Popen(command,shell=True,stdout=subprocess.PIPE)
line = p.stdout.readline().rstrip()
```

#if output has no line then it will for a function web_scrap () , the command explained in chapter 2.1 to chapter 2.4 are defined in fuction web_scrap ().

```
if not line:
  web_scrap ()
```

#else,it will chdir to local directory and listdir , if no file found then call function to web_scrap() to download fresh page, this steps is very be useful if the script run for the first time, else it will iterate over the files and take index from 0 to 16 to record only the date from saved local directory

filename and compare it with 'last-modified' date output index 15 to 31 ,i.e 'a=line [15:31]' , if file date not match with website last-modfified date then it call function web_scrap() to download fresh page , finally if date match with 'last-modified' date that means webpage didn't modify since last visit , so it just print like 'last visited date match . No Need to download' instead of downloading a new page.

```
else:
    a=line [15:31]
    chdir ("/mnt/d/python/project/")
    files=listdir('.')
    if not files:
      web_scrap()
    else:
      for i in files:
       if i[0:16] not in a:
          web_scrap()
       else:
          print "last vistited date match"+i[0:16]+" . No Need download"
```

After revisit and modify complete code is given below:

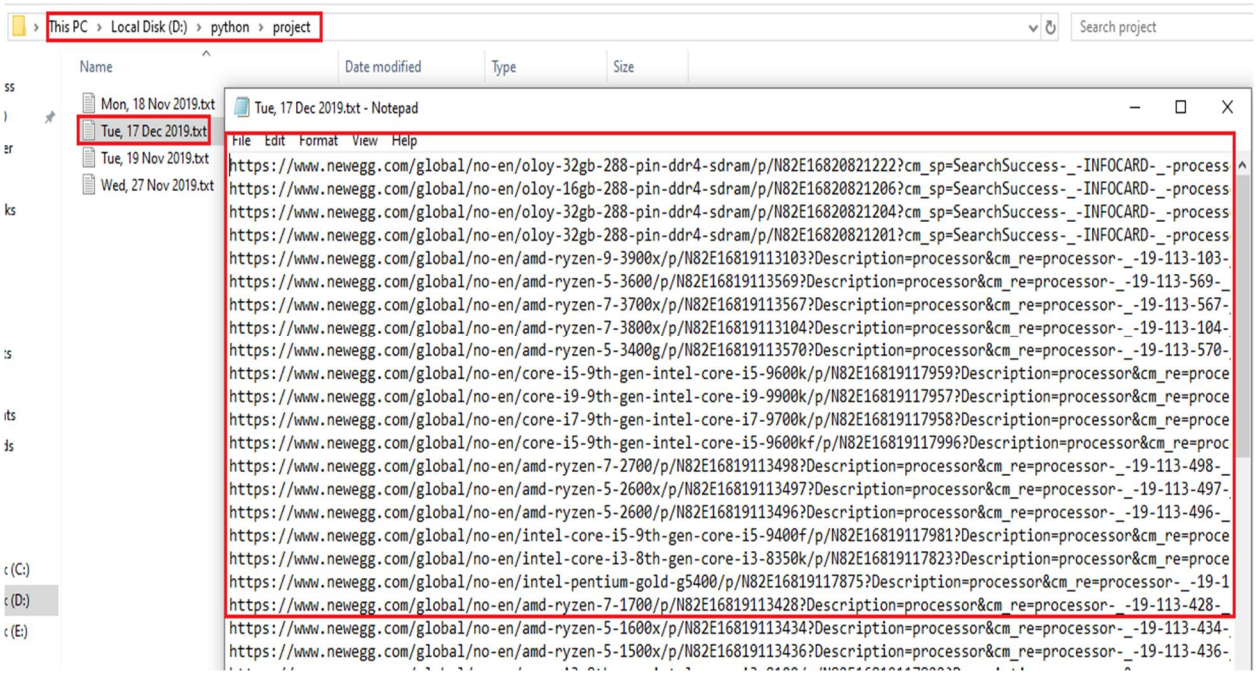
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from urllib2 import urlopen as Urequest
from bs4 import BeautifulSoup as soup
import datetime
import time
import subprocess
from os import getcwd
from os import chdir
from os import listdir

def web_scrap ():
    min_url="https://www.newegg.com/global/no-en/p/pl?d=processor"
    uclient = Urequest(min_url)
    page_html= uclient.read()
    uclient.close()
    page_soup=soup(page_html,"html.parser")
    div_class=page_soup.findAll("div",{"class":"item-container"})
    div_processor = div_class

    dtdate2=subprocess.Popen(["date | awk -F' ' '{print
$1,$3,$2,$6}'"],shell=True,stdout=subprocess.PIPE)
    line2 = dtdate2.stdout.readline().rstrip()
    f =open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','w')
    g =open ('/mnt/d/python/projectlinks/links_plot.txt','w')
```

```
    for container in div_processor:
      product_link=container.a["href"]
      webpages = product_link+'\n'
      for i in webpages:
         f.write(i)
         g.write(i)
    f.close()
    g.close()

    h = open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','r')
    word='promo@promo.global.newegg.com'
    word_list=[]
    word2='https'
    download_count=[]
    for i in h :
      min_text=i
      if word2 in min_text:
         download_count.append(word2)
         uclient = Urequest(min_text)
         page= uclient.read()
         uclient.close()
         page_text=soup(page,"html.parser")
         for i in page_text.findAll('p'):
           k = ''.join(i.findAll(text=True))
           if word in k:
            word_list.append(word)
    a=word_list.count(word)
    b=download_count.count(word2)
    print "Number of downloaded links are:"+str(b)+". The download path is
/mnt/d/python/project/"+line2[0:3]+','+line2[3:    ]+'.txt'
    print "Given word is: "+word+" The number of counts of given word are: " +str(a)
    h.close()

command="curl -I 'https://www.newegg.com/global/no-en/p/pl?d=processor' | grep -i
last-modified"
p = subprocess.Popen(command,shell=True,stdout=subprocess.PIPE)
line = p.stdout.readline().rstrip()
if not line:
web_scrap ()
else:
    a=line [15:31]
    chdir ("/mnt/d/python/project/")
    files=listdir('.')
    if not files:
      web_scrap()
    else:
      for i in files:
       if i[0:16] not in a:
          web_scrap()
       else:
          print "last visited date match “+i[0:16]+" . No Need download"
```

modified script run screenshot, where it first check website 'last-modified' date and based on that it decide and download page to local file and count given word accordingly.

```
root@DESKTOP-37QUITS:/home/asif/python_project# ./web_crawl.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
Number of downloaded links are:40. The download path is /mnt/d/python/project/Tue, 17 Dec 2019.txt
Given word is: promo@promo.global.newegg.com The number of counts of given word are: 40
```

screenshot of downloaded links in local directory filename Tue,17 Dec 2019.txt



Step 2.7 Monitor the response time of the given website in every 10minute and save the output in a file for display

To be started

---declare the source header at the start----

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#Import datetime provides various time-related functions
import datetime
```

#The subprocess module allows us to spawn processes, connect to their input/output/error pipes,and obtain their return codes.

```
import subprocess
```

Jump into the code ,

First, subprocess Popen run ping command in shell and take the output in standard output .

Then current date time output put into a variable

```
command = "ping -c 5 www.newegg.com"
command=command.split()
process = subprocess.Popen(command,stdout=subprocess.PIPE)
time=datetime.datetime.now()
```

Put condition where if ping process still running then check real time output of command and put the value in a variable , and check whether word 'time=' is in the value or not , if have then open file "f" at local directory as append mode and write the file as "current date & time + space + index -12 to rest which cover only time delay part of the ping command output"

```
while process.poll() is None:
     line = process.stdout.readline().rstrip()
     a=line
     if 'time=' in a:
      f = open ('/mnt/d/python/projectlinks/timedelay.txt','a+')
      f.write(str(time)+' '+a[-12:]+'\n')
      f.close()
```

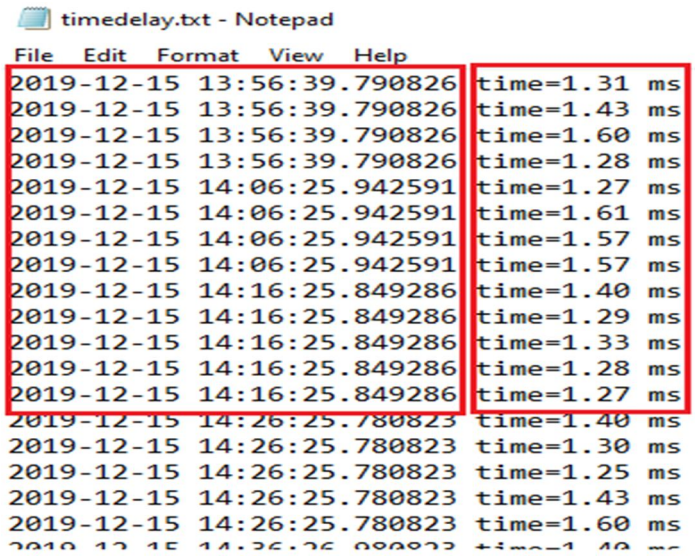
Finally, when run the script then the output in the file will be like below screenshot:

```
2019-12-15 13:56:39.790826 time=1.31 ms
2019-12-15 13:56:39.790826 time=1.43 ms
2019-12-15 13:56:39.790826 time=1.60 ms
2019-12-15 13:56:39.790826 time=1.28 ms
2019-12-15 14:06:25.942591 time=1.27 ms
2019-12-15 14:06:25.942591 time=1.61 ms
2019-12-15 14:06:25.942591 time=1.57 ms
2019-12-15 14:06:04.430121 time=1.33 ms
2019-12-15 14:06:04.430121 time=1.69 ms
```

Now , data is ready . So make the output available in every 10 minutes , add cronjob schedule as showing below screenshot which will run the command in every 10 minute and save the output in the file

```
root@DESKTOP-37QUITS:/home/asif/python_project# crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirecte
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cro
#
# m h  dom mon dow   command
10 * * * * /home/asif/python_project/ping_script.py
root@DESKTOP-37QUITS:/home/asif/python_project#
```

So, according to cronjon program append ping command time delay output at local directory file 'timedelay.txt' in every 10 minute and save it to display, showing below screenshot:

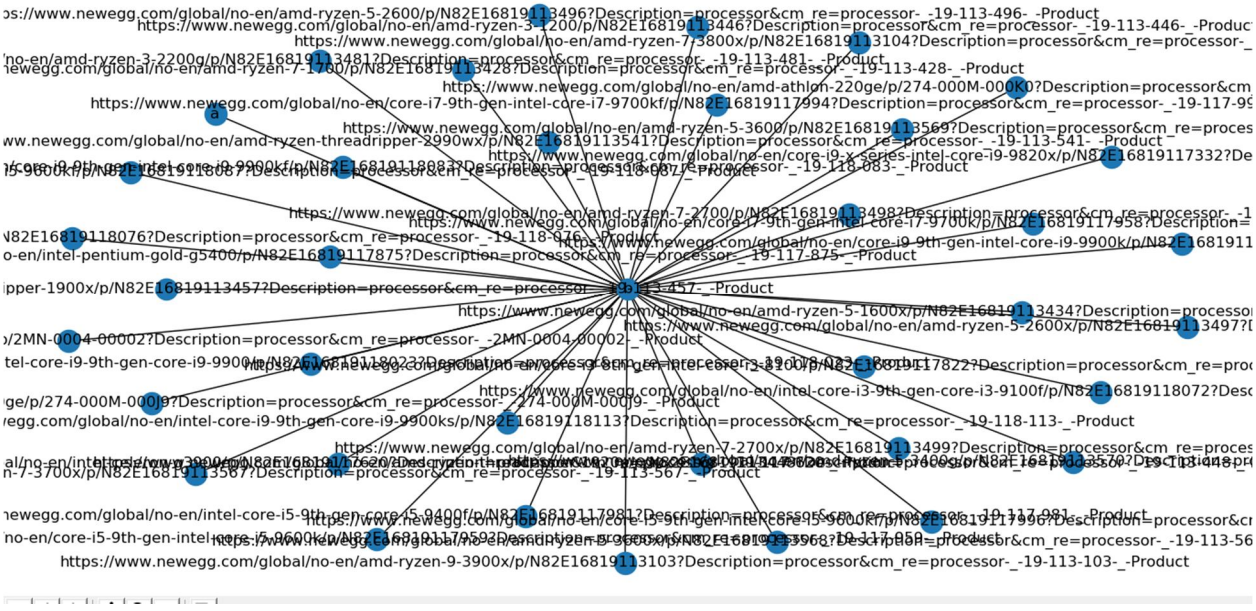


Step 2.8 Visulaize a network graph

#Import Networkx and matplotlib.pyplot for network graph .

```
import networkx as nx
import matplotlib.pyplot as plt

#Create a empty graph
h = nx.Graph()
#define node , which is in this case is the website address
b = 'https://www.newegg.com/global/no-en/p/pl?d=processor'

#define edgelist , which is in this case weblinks address crawled from website
h=nx.read_edgelist('D:\python\projectlinks\links_plot.txt',create_using=nx.Graph(),nodetype=str)

#print node and edge info
print(nx.info(h))

#draw the node and edges with lebel
nx.draw(h,with_labels=True)

#Finally visualize the graph
plt.show()
```

Visualize network graph like below screenshot. Where center point is the node(https://www.newegg.com/global/no-en/p/pl?d=processor) and all other weblinks connected as edges with node



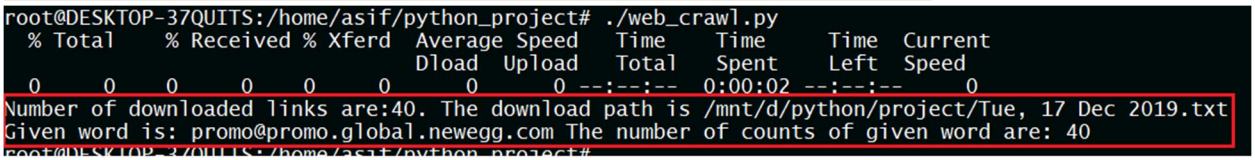
Section 3

Results:

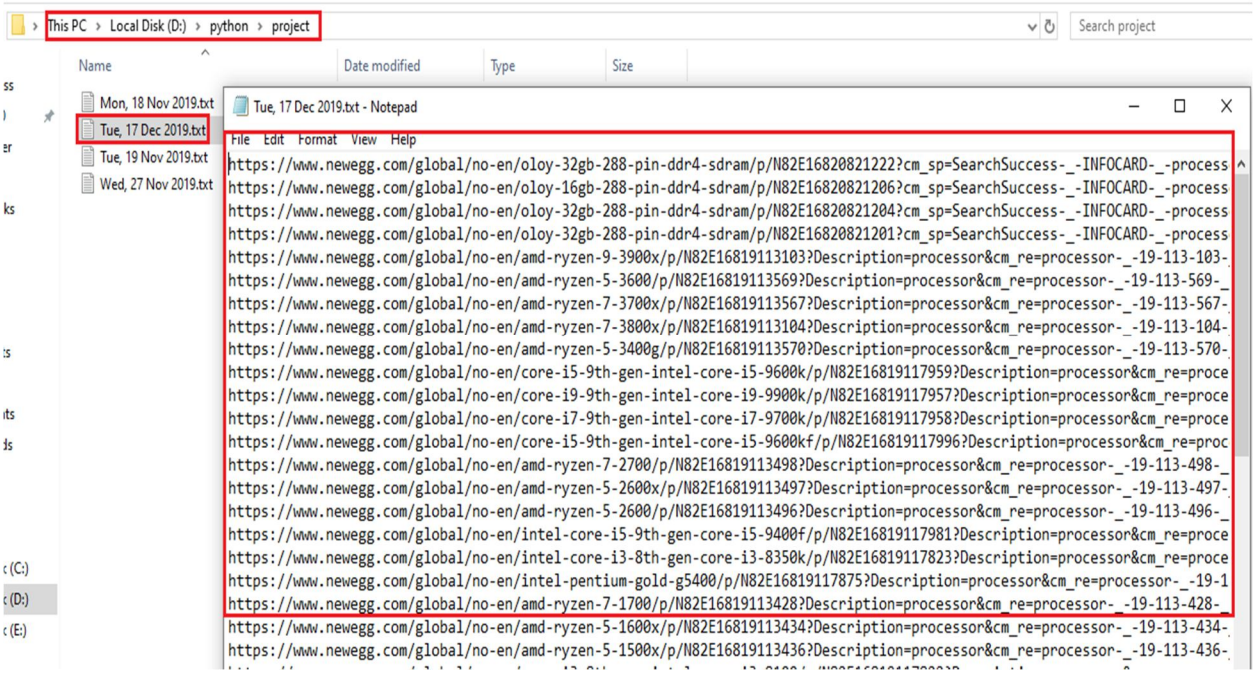
Based website 'https://www.newegg.com/global/no-en/p/pl?d=processor' last-modified tag check on create a web crawlaer to pull out desired weblinks from , storage them into local directory file and crawl over the gathered weblinks and search for a particular word 'promo@promo.global.newegg.com' and display the number of count of occurrences of that word. Besides, record ping time delay of the website in every 10 minutes and visualize a network graph.

Result of the work show as screenshot as below one by one:

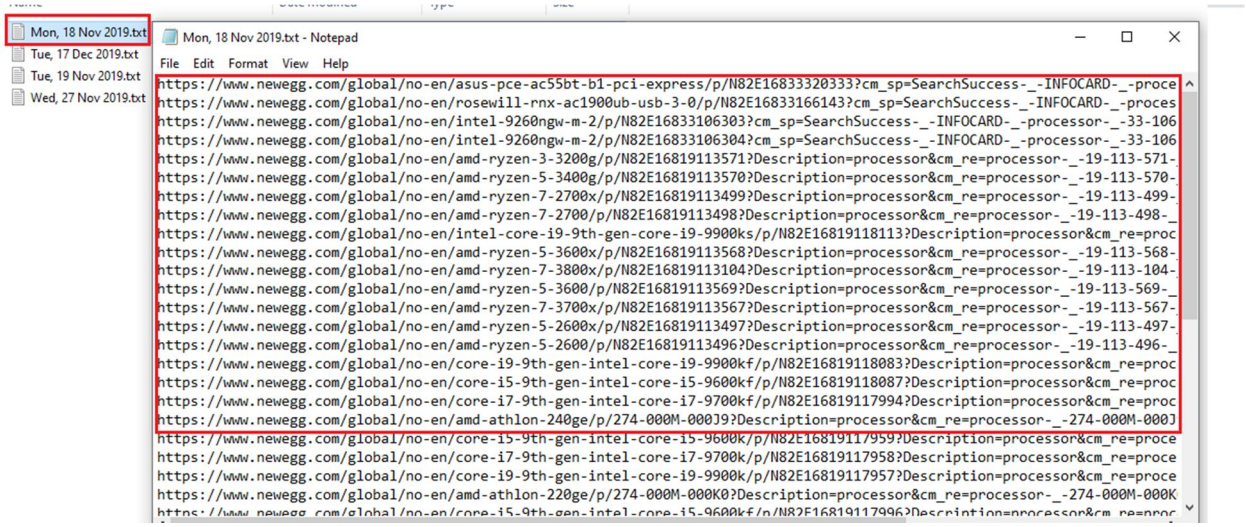
screenshot, where it first check website 'last-modified' date and based on that it decide and download page to local file and count given word accordingly.



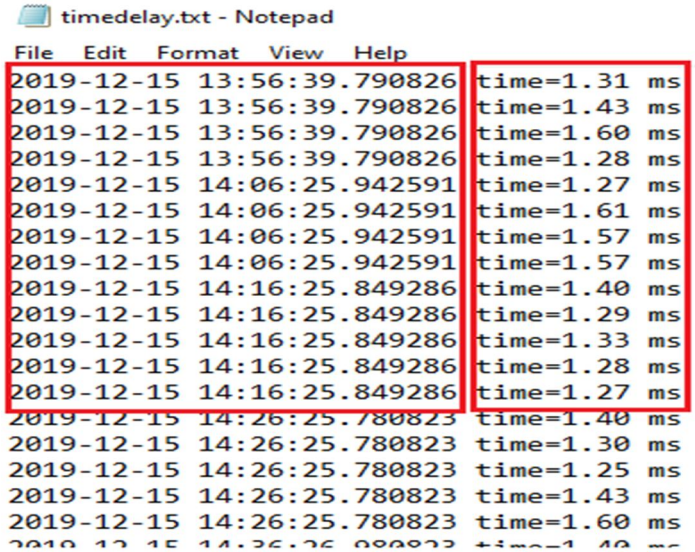
Screenshot of downloaded links in local directory filename Tue,17 Dec 2019.txt



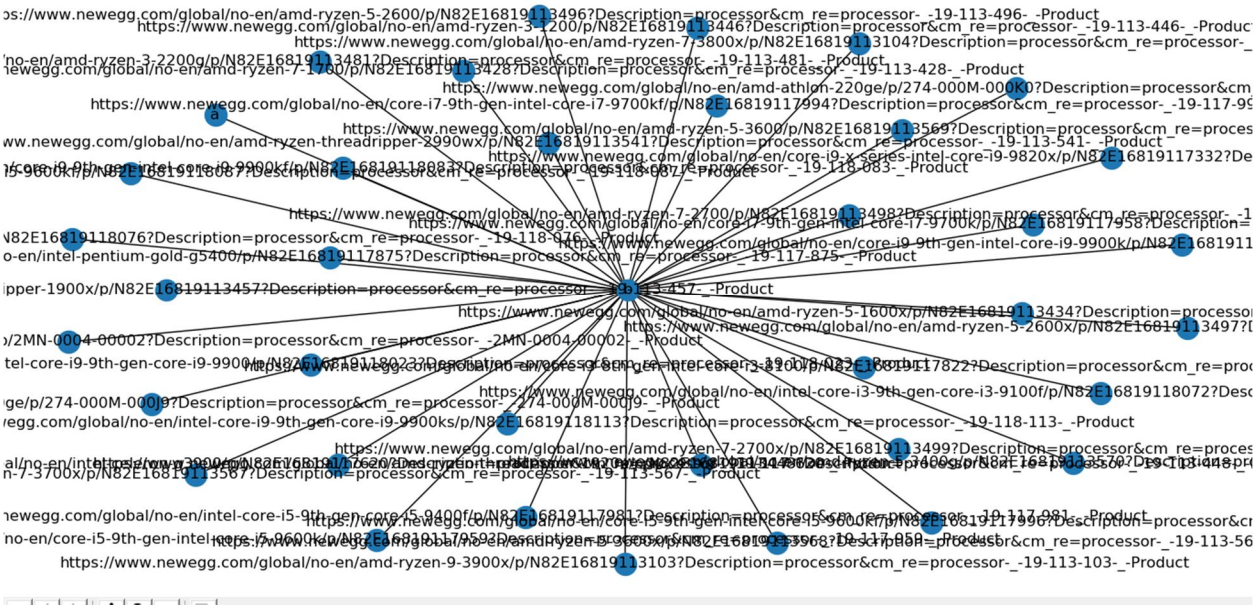
Screenshot of another date saved data



According to cronjob program append ping command time delay output at local directory file 'timedelay.txt' in every 10 minute and save it to display, showing below screenshot:



Visualize network graph like below screenshot. Where center point is the node(https://www.newegg.com/global/no-en/p/pl?d=processor) and all other weblinks connected as edges with node



Evaluation:

Below are evaluation points which can be done in future to optimize the work:

1. Create a statistical data based on crawled weblinks and send a PDF report to user.
2. Send notification to user via email or instant messaging platform if crawled weblinks quantity exceed or reduced to or from any certain level.
3. Make network plot graph creation script more automated.

Appendix:

Attach are the source code of the work:

web crawl.py:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from urllib2 import urlopen as Urequest
from bs4 import BeautifulSoup as soup
import datetime
import time
import subprocess
from os import getcwd
from os import chdir
from os import listdir

def web_scrap ():
    min_url="https://www.newegg.com/global/no-en/p/pl?d=processor"
    uclient = Urequest(min_url)
    page_html= uclient.read()
```

```
    uclient.close()
    page_soup=soup(page_html,"html.parser")
    div_class=page_soup.findAll("div",{"class":"item-container"})
    div_processor = div_class

    dtdate2=subprocess.Popen(["date | awk -F' ' '{print
$1,$3,$2,$6}'"],shell=True,stdout=subprocess.PIPE)
    line2 = dtdate2.stdout.readline().rstrip()
    f =open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','w')
    g =open ('/mnt/d/python/projectlinks/links_plot.txt','w')
    for container in div_processor:
      product_link=container.a["href"]
      webpages = product_link+'\n'
      for i in webpages:
         f.write(i)
         g.write(i)
    f.close()
    g.close()

    h = open ('/mnt/d/python/project/'+line2[0:3]+','+line2[3:]+'.txt','r')
    word='promo@promo.global.newegg.com'
    word_list=[]
    word2='https'
    download_count=[]
    for i in h :
      min_text=i
      if word2 in min_text:
         download_count.append(word2)
         uclient = Urequest(min_text)
         page= uclient.read()
         uclient.close()
         page_text=soup(page,"html.parser")
         for i in page_text.findAll('p'):
           k = ''.join(i.findAll(text=True))
           if word in k:
            word_list.append(word)
    a=word_list.count(word)
    b=download_count.count(word2)
    print "Number of downloaded links are:"+str(b)+". The download path is
/mnt/d/python/project/"+line2[0:3]+','+line2[3:    ]+'.txt'
    print "Given word is: "+word+" The number of counts of given word are: " +str(a)
    h.close()

command="curl -I 'https://www.newegg.com/global/no-en/p/pl?d=processor' | grep -i
last-modified"
p = subprocess.Popen(command,shell=True,stdout=subprocess.PIPE)
line = p.stdout.readline().rstrip()
if not line:
web_scrap ()
else:
    a=line [15:31]
    chdir ("/mnt/d/python/project/")
    files=listdir('.')
    if not files:
      web_scrap()
```

```
    else:
      for i in files:
       if i[0:16] not in a:
          web_scrap()
       else:
          print "last vistited date match"+i[0:16]+" . No Need download"
```

ping script.py:

```
#! /usr/bin/python
import subprocess
import datetime

command = "ping -c 5 www.newegg.com"
command=command.split()
process = subprocess.Popen(command,stdout=subprocess.PIPE)
time=datetime.datetime.now()

while process.poll() is None:
     line = process.stdout.readline().rstrip()
     a=line
     if 'time=' in a:
      #print a
      #print a[-9:]
      f = open ('/mnt/d/python/projectlinks/timedelay.txt','a+')
      f.write(str(time)+' '+a[-12:]+'\n')
      f.close()
import networkx as nx
import matplotlib.pyplot as plt
```

network graph.py:

```
h = nx.Graph()
b = 'https://www.newegg.com/global/no-en/p/pl?d=processor'
h=nx.read_edgelist('D:\python\projectlinks\links.txt',create_using=nx.Graph(),nodetyp
e=str)
print(nx.info(h))
nx.draw(h,with_labels=True)
plt.show()
```