ACIT4420 Problem solving with scripting

Project Report

Candidate Number 58

WEB CRAWLER PROJECT

What is a Web Crawler?

Web crawler sometimes known as spider or spiderbot is a process of crawling or scraping through the web site which systematically browses and extracts the content from the website.

It is a program which navigates through the World Wide Web in a systematic, methodical and automated manner. At the beginning, the web crawler visits the web site. It extracts the web page content and then visits the other pages by following the web links. This process of crawling goes on recursively until all the web pages have been visited and read. This process is known as Web Crawling or Web Spidering. Several sites in particular search engines use crawling as a method to provide the latest up-to-date data.

Web Crawlers are primarily used to create a copy of all the visited web pages through navigating for later processing by the search engine. This would index visited and downloaded web pages which would help to obtain quicker searches.

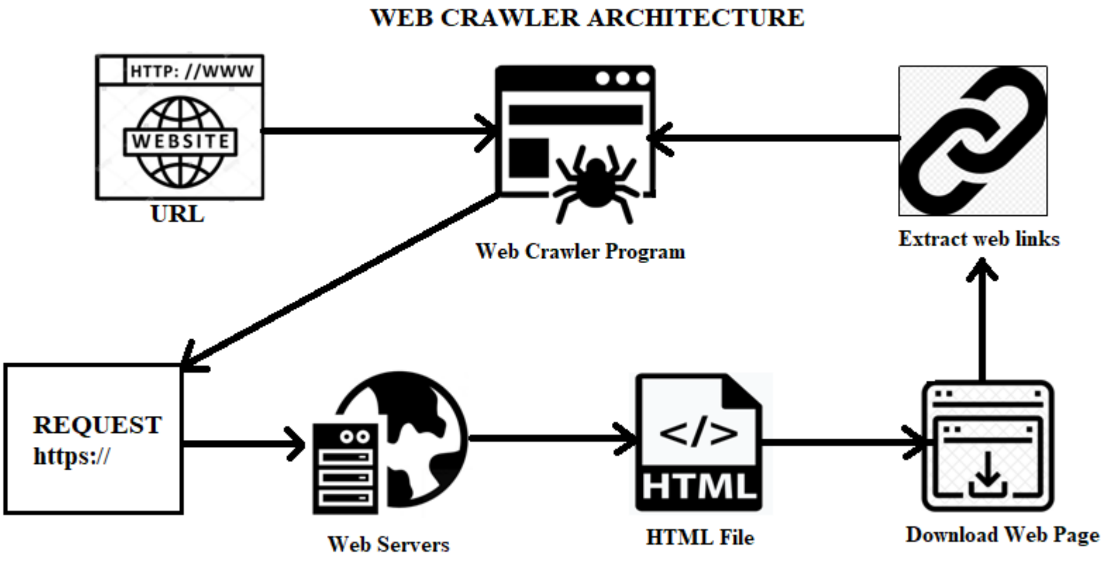
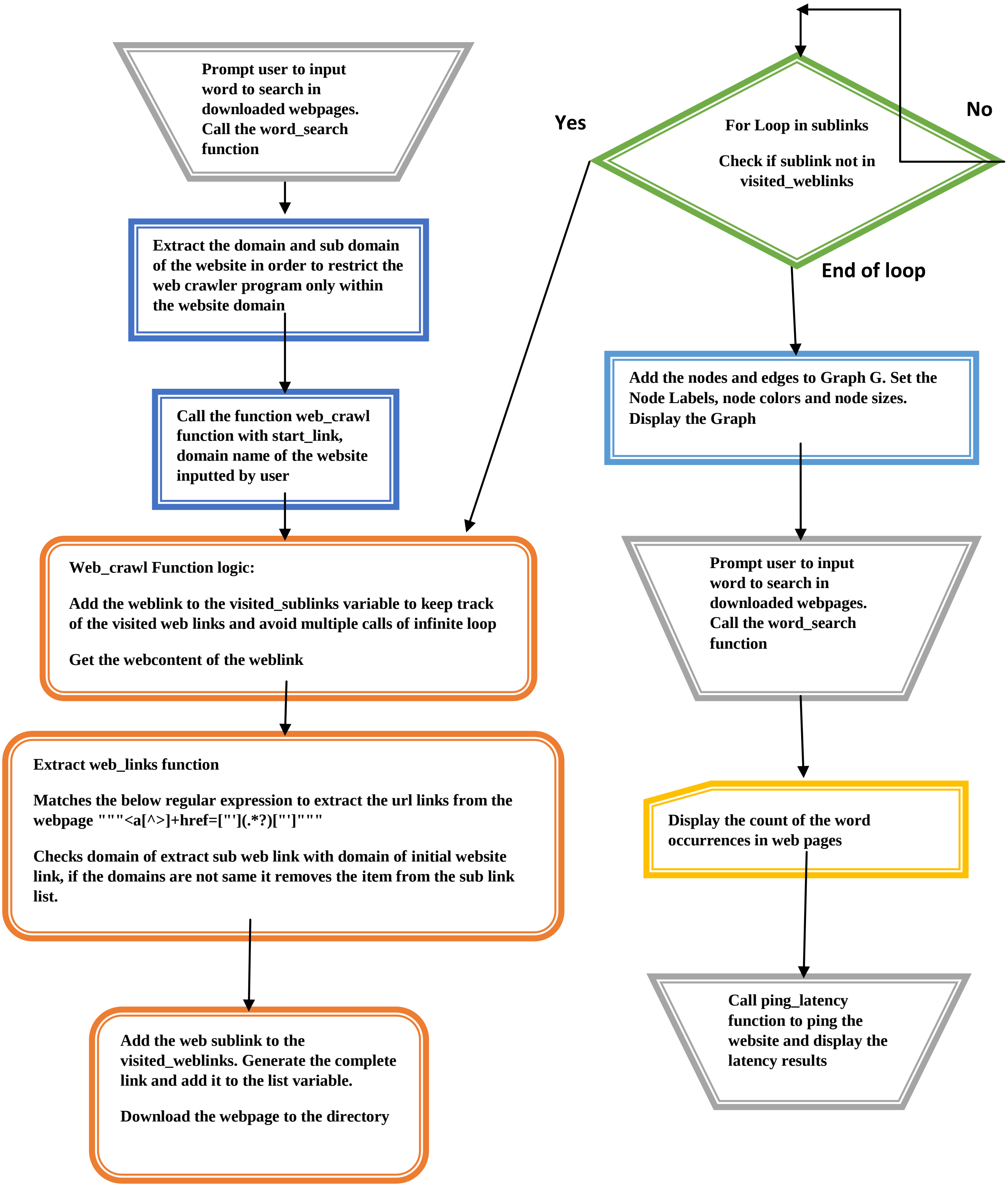


Figure 1 Architecture Diagram of a Web Crawler, Ref [2]

The Fig 1 above illustrates a typical architecture of a Web Crawler. A URL link is given as an input to the Web Crawler program. It then sends a http request to the web servers to obtain the HTML file content of the web page. The web page content is indexed and downloaded in a system. The sub links contained in the web page are extracted. The web crawler program executes again with each of the sub links iteratively until all the web links are crawled.

Technical Design:

Logical Flowchart of Technical Design for WebCrawler Program.


Prompt user to input word to search in downloaded webpages. Call the word_search function
Extract the domain and sub domain of the website in order to restrict the web crawler program only within the website domain
Call the function web_crawl function with start_link, domain name of the website inputted by user
Web_crawl Function logic:
Add the weblink to the visited_sublinks variable to keep track of the visited web links and avoid multiple calls of infinite loop
Get the webcontent of the weblink
Extract web_links function
Matches the below regular expression to extract the url links from the webpage """<a[^>]+href=["'](.*?)["']"""
Checks domain of extract sub web link with domain of initial website link, if the domains are not same it removes the item from the sub link list.
Add the web sublink to the visited_weblinks. Generate the complete link and add it to the list variable.
Download the webpage to the directory
Yes
No
For Loop in sublinks
Check if sublink not in visited_weblinks
End of loop
Add the nodes and edges to Graph G. Set the Node Labels, node colors and node sizes. Display the Graph
Prompt user to input word to search in downloaded webpages. Call the word_search function
Display the count of the word occurrences in web pages
Call ping_latency function to ping the website and display the latency results

Implementation:

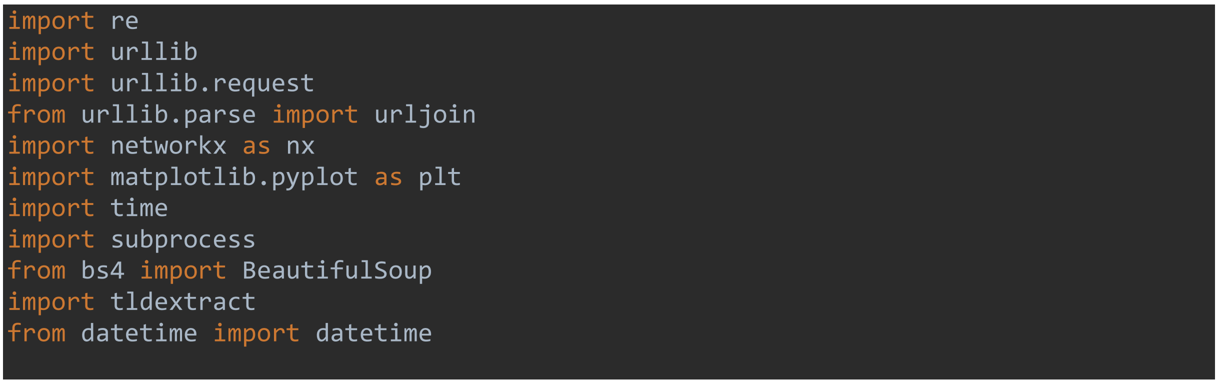
Name of the Program: Web_Crawler_Program.py

Code Structure: The program for Web Crawler is subdivided into the below parts:

- List of imported Packages
- Initialization section for initializing the global variables to be used in the code
- Set of functions which perform the different functionalities for web crawling
- Main section of the code to receive various input parameters from the user during execution

List of Imported Packages:

The list of packages or modules that are imported at the start of the program are as in the below image.



```
import re
import urllib
import urllib.request
from urllib.parse import urljoin
import networkx as nx
import matplotlib.pyplot as plt
import time
import subprocess
from bs4 import BeautifulSoup
import tldextract
from datetime import datetime
```

Details and usage of the imported packages in the Web Crawler program

re – It provides regular expression matching operations similar to the ones which are available in Perl. This module is used to generate the regular expression. The regular expression would try to search all the URL links in the web page content and would extract the web links from each webpage

urllib – This package is used to read the webpage content of an html page.

urllib.request- This Python module is for fetching the URL

urljoin from urllib.parse - This is used for joining the URL paths

networkx - It is for creating and manipulating graphs and networks

matplotlib.pyplot - This is used for creating figures and plotting for network graphs.

time – This module would be used in the Web crawler program for waiting during the code execution. To monitor and get the ping latency results after every 10 minutes for the website, the time module would be used to wait for 10 minutes during code execution.

subprocess – This package is used to run the shell execution commands with the input and output pipes by creating processes.

BeautifulSoup from bs4 - This library is used for pulling out the data from the HTML files. This would be used to perform the count of the given word based on user input in the paragraphs of the downloaded web pages.

tldextract – This would be used for extracting the subdomain, domain and suffix from a URL. This package would be useful to implement the Web Crawler which would crawl the web links which are in the domain of the original website.

from datetime import datetime – This modeule is used to get the current system date and time. This would be used to display the date and time when the ping latency results for the website are displayed.

Initialization Section

```
absolute_links = []             #List of all the complete links to crawl
visited_sublinks = set()        #List of all the visited web links
visited_edges = set()           #List of web links mapping pair
file_save_location = 'C:\\Users\\KCP\\Desktop\\Web_Crawler_Download\\'
#File Directory Path to save the web page
downloaded_files = set()        #Complete File Path of the downloaded files
G = nx.Graph()                  #Initialize Graph
```

In this section of the code, few of the variables would be defined and initialized.

absolute_links is initialized as empty list. This list variable would store the absolute web links obtained when the programs crawls through several web pages recursively.

visited_sublinks is initialized as empty set. This set variable would contain all the unique absolute web links which the web crawler program has visited.

visited_edges is initialized as empty set. This set variable would contain the edges between the web links.

file_save_location is a string variable which is initialized with the directory folder path where the Web Page would be downloaded and saved. *This variable should be changed if the program is executed on a different system in order to save the webpages.*

downloaded_files set variable would store the entire file path of the downloaded webpages.

With **G=nx.Graph()** , a graph variable G is defined which would generate the network graph of the web crawler program.

Set of Functions in the Web Crawler Program:

*Function **def extract_weblinks(web_content):***

```
def extract_weblinks(web_content):
    weblink_regex = re.compile("""<a[^>]+href=["'](.*?)["']""", re.IGNORECASE)
```

```
#Regular Expression to extract all the web links on the web page
    web_sublinks = re.findall(weblink_regex, web_content)
#Find all the web links in the web page
    return web_sublinks
```

This function would identify all the links in the web page and return them as a list to the calling function.

The Regular Expression used to extract the web url links from the wage pages is as follows:

`"""<a[^>]+href=["'](.*?)["']"""`

This would first match the characters `"""<a` case sensitive, `[^>]+` would not match a single character >, it would then match `href=` literally case sensitive, `["']"""` would match characters in square bracket followed by `""". (.*?)`This would be the capturing group matching between zero and unlimited times.

The **re.compile()** method in the re package library compiles the regular expression give with ignoring the case to form a pattern for regular expression.

The **re.findall()** method takes two arguments, first one as the regular expression and the second argument as the web content and returns the list of all the matched results of the regular expression found in the web content

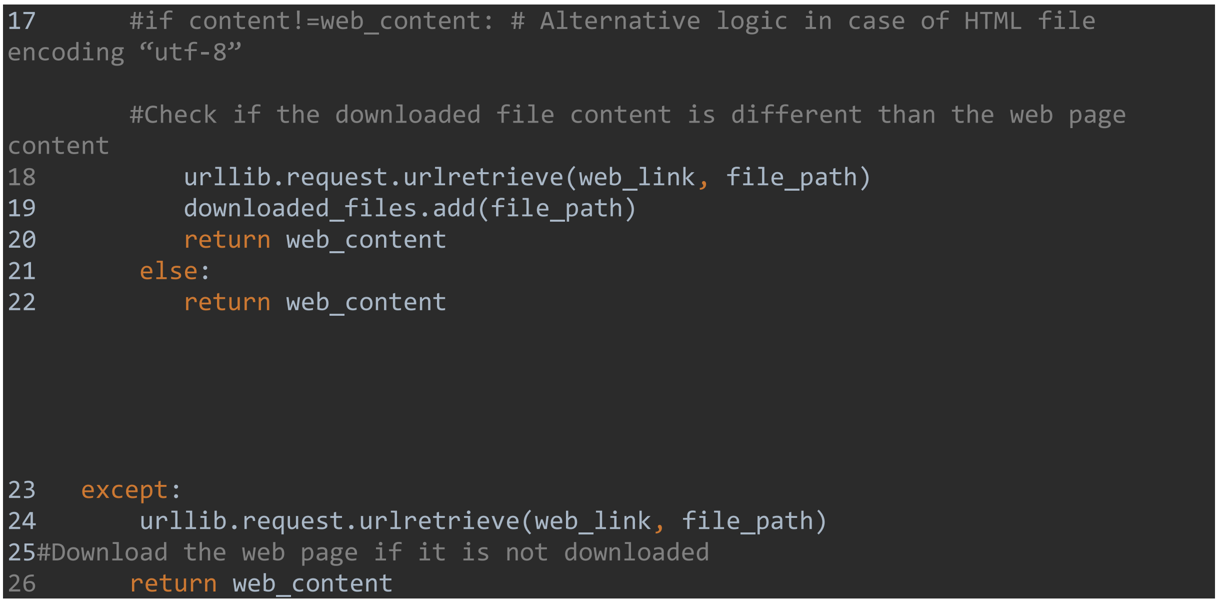
*Function **def get_save_webcontent(web_link)***

```
def get_save_webcontent(web_link):
1    response = urllib.request.urlopen(web_link)
2    webpage_source = response.read()
3    web_content = webpage_source.decode()
4    mod_link = web_link.replace('http://', '')
5#To generate the name of the webpage file to be downloaded
6    mod_link = mod_link.replace('https://', '')
7    mod_link = mod_link.replace('/', '.')
8    if mod_link.endswith('.'):
9        file_path = file_save_location + mod_link + 'html'
10#Generate the whole directory location path for the web page to be saved
11   else:
12       file_path = file_save_location + mod_link + '.html'
13   try:
14       file = open(file_path, 'r')
15       file_content = file.read()

        #Alternative Logic for file read in case of HTML file encoding “utf-8”
        #file = codecs.open(location, 'r',encoding="utf-8")
       #file_content = file.readlines()
        #content=’’
        #for item in file_content:
            content=content + item



16       if file_content!= web_content:
```



```
17      #if content!=web_content: # Alternative logic in case of HTML file
encoding “utf-8”

        #Check if the downloaded file content is different than the web page
content
18          urllib.request.urlretrieve(web_link, file_path)
19          downloaded_files.add(file_path)
20          return web_content
21       else:
22          return web_content




23   except:
24       urllib.request.urlretrieve(web_link, file_path)
25#Download the web page if it is not downloaded
26      return web_content
```

This function receives a web URL link as input. It gets the response from the web page link with the below command in line 1 as per above screenshot

response = urllib.request.urlopen(web_link)

It obtains the web page source with the below command on line 2

webpage_source = response.read()

In command on line 3, it decodes the web page content and obtains the web contents in a string format.

web_content = webpage_source.decode()

The statements from line 4 to line 7 is to generate the name of the html file to be downloaded. As the name of the file cannot contain certain characters and http:// string, it would be replaced with the spaces or ‘.’ Symbol to generate a valid file name. modlink variable is used for storing the modified link with multiple replaces.

modlink=weblink.replace(‘http://’,’ ‘)

modlink=modlink.replace(‘http://,’ ‘)

modlink=modlink.replace(‘/’,’. ‘)

In the next statements from line 8 to 12, a logic is written to check if the modlink ends with ‘.’

- If the modlink end with a ‘.’ Symbol , then the absolute file path is generated as the combination of the directory path to be saved + modified link + .html extension
File_save_location + modlink + “.html”
- Else , the absolute file path is generated as the combination of the directory path + modified link + ‘html’ extension
File_save_location + modlink + “html

In the try-except catch block from line 13 to line 26,

A file is opened with the above generated file path and in the read mode with the below command

file = open (file_path, ‘r’)

The below command reads the file contents

file_content= file.read()

Logic for checking if the website content has been changed since last downloaded in the directory folder is as follows:

If the web _content is not the same as the file content:

> Then the website is downloaded in the directory path on the system with the generated file name with the below command.
>
> Urllib.request.urlretrieve(web_link, file_path)
>
> It also adds the absolute path of the downloaded webpage in the variable downloaded_files and returns the webcontent to the calling function

Else

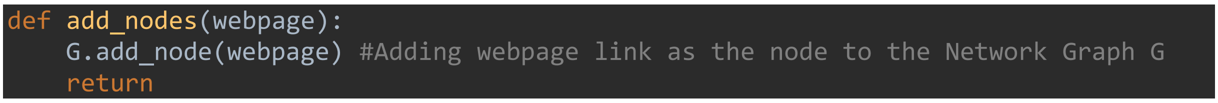
> It returns the webcontent to the calling function

This logic sometimes doesn’t work for the HTML saved files which are encoded in various formats such as “utf-8” or “latin-1”. For these cases, an alternate implementation logic is as follows:

Open the file in the encoding as “utf-8”. Use readlines() method to read the file contents in the form of list of strings. Through a for loop, merge all the strings into a single string named “content”. Compare content with web contents of the web page, and rest of the logical flow remains the same.

Another possible implementation logic for downloading the webpages only if the website content has changed since the last download is: Compare the size of the downloaded file contents with the size of the webpage contents, if the sizes are different then the website has been changed.

*Function **def add_nodes(web_page):***



```
def add_nodes(webpage):
    G.add_node(webpage) #Adding webpage link as the node to the Network Graph G
    return
```

This function would add the webpage link which is input parameter as a node to the Network Graph G and returns back to the calling function.

The method G.add_node(webpage) for the Graph G adds the node as webpage to the network graph G

Function ***def add_edges(node1, node 2) :***

```
def add_edges(node1, node2):
    G.add_edge(node1, node2) #Adding weblink mapping as edges to Network Graph G
    return
```

This function would add the edges between the two webpage links connected that are passed as input parameters to the Network Graph G and returns back to the calling function.

The method G.add_node(webpage) for the Graph G adds the node as webpage to the network graph G

Function ***def ping_latency(web_link):***

```
def ping_latency(web_link):
1    p = subprocess.Popen('ping ' + web_link, stdout=subprocess.PIPE)
2    while(True):
3        print(p.communicate()[0])
4        time.sleep(600)
5#Wait for 10 minutes before pinging again
6        p = subprocess.Popen('ping ' + web_link, stdout=subprocess.PIPE)
7        if p.poll() == 0:
8#If Web site is not up
9            break
10    return
```

The ping_latency(web_link) function takes the website link (starting with www) as an input. The line statement 1 subprocess.Popen() function executes the shell command "**ping**" with the parameters as the website link and writing output to the PIPE.

The line statement 3 prints the ping latency output immediately if the response from the website was received on ping.

The line statement 4 time.sleep(600) , waits in the exeution process for 600 seconds (10 minnutes) before executing the program further.

After 10 minutes are elapsed, the line number 6 again pings the website through subprocess.Popen and writes the ping latency output to the output PIPE if response received successfully.

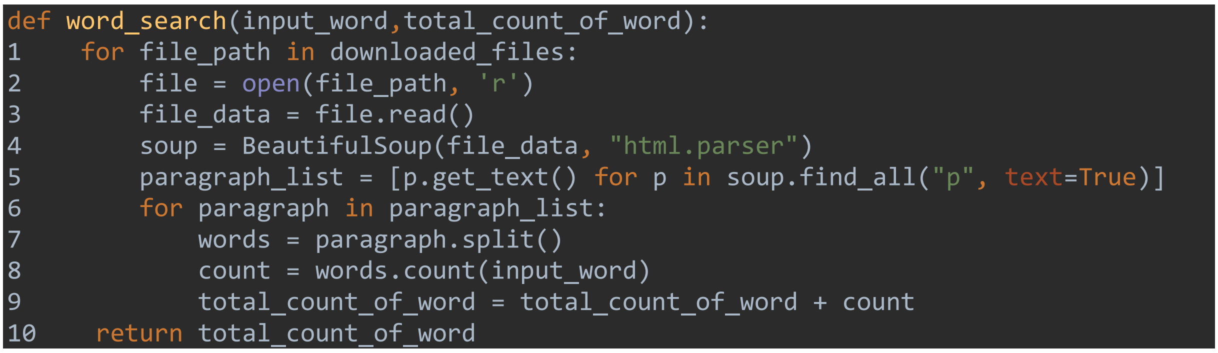
This loop executes continuously unless in either of the below conditions

- The ping latency response was not received from the website

- The website is down

- User terminated the session forcefully.

Function ***def word_search(input_word, total_count_of_word):***



```
def word_search(input_word,total_count_of_word):
1    for file_path in downloaded_files:
2        file = open(file_path, 'r')
3        file_data = file.read()
4        soup = BeautifulSoup(file_data, "html.parser")
5        paragraph_list = [p.get_text() for p in soup.find_all("p", text=True)]
6        for paragraph in paragraph_list:
7            words = paragraph.split()
8            count = words.count(input_word)
9            total_count_of_word = total_count_of_word + count
10    return total_count_of_word
```

This function takes input word to be searched in the paragraphs of the downloaded web pages as argument and initial value of 0 in the total_count_of_word

In the line statement 1 as above, it iterates in all the list of downloaded filepaths of the webpages, and opens the file in the line statement 2 in read mode.

In line statement 3, it reads the file content with file.read()

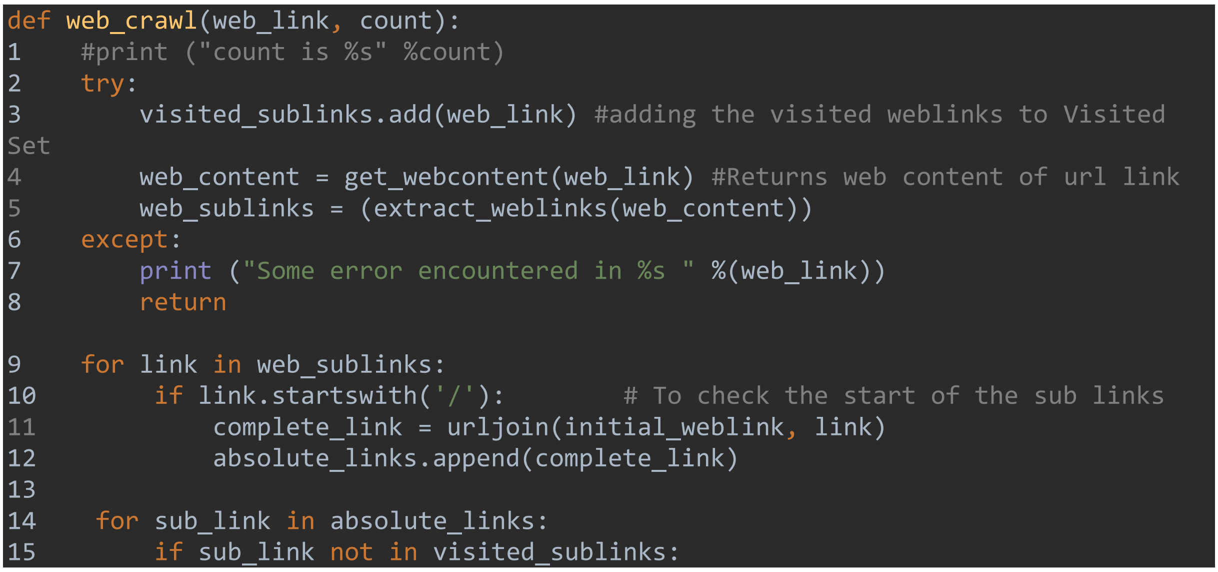
In line statement 4, BeautifulSoup packages parses the file in htmlparser format and returns the output to soup variable.

In line 5, a list comprehension is used to return all the list of paragraphs found within <p> and </p> in all the webpages.

From line statements 6 to 9, logic is written to iterate in all the paragraphs, split each paragraphs into words and count the occurrences of the input word in these list of words from the paragraphs.

In line 10, the function returns the variable total_count_of_word with the total count of matches found to the calling function.

Function ***def web_crawl(web_link, count):***



```
def web_crawl(web_link, count):
1    #print ("count is %s" %count)
2    try:
3        visited_sublinks.add(web_link) #adding the visited weblinks to Visited
Set
4        web_content = get_webcontent(web_link) #Returns web content of url link
5        web_sublinks = (extract_weblinks(web_content))
6    except:
7        print ("Some error encountered in %s " %(web_link))
8        return

9    for link in web_sublinks:
10        if link.startswith('/'):        # To check the start of the sub links
11            complete_link = urljoin(initial_weblink, link)
12            absolute_links.append(complete_link)
13
14    for sub_link in absolute_links:
15        if sub_link not in visited_sublinks:
```

```
16            visited_edges.append((web_link,sub_link,count))
17            print (web_link,sub_link,count)
18
19    for sub_link in absolute_links:
20        if sub_link not in visited_sublinks:
21            #if count>=100:    #For restricting the number of iterations
22            #    break
23            #else:
24            #    count+=1
25            web_crawl(sub_link, count)
```

The function is invoked by the main program with the initial web link for crawling the website. If some problem occurs in opening the webpage, an error message is displayed and the function returns.

In the function logic, in line 2 to 8, it adds the weblink to the visited_sublinks set variable. And gets the webcontent by calling the *get_webconten()t* function in the web_content variable.

The sublinks are extracted from the web page content by calling the *extract_weblinks()* function

From line 9 to 12, it checks the url web link, if it starts with / then it generates the complete link by appending initial start_link with the sub link. Also, the complete link is added to the absolute_links list variable.

From lines 14 to 17, it adds the edges between the web link and sublinks to the visited_edges variable.

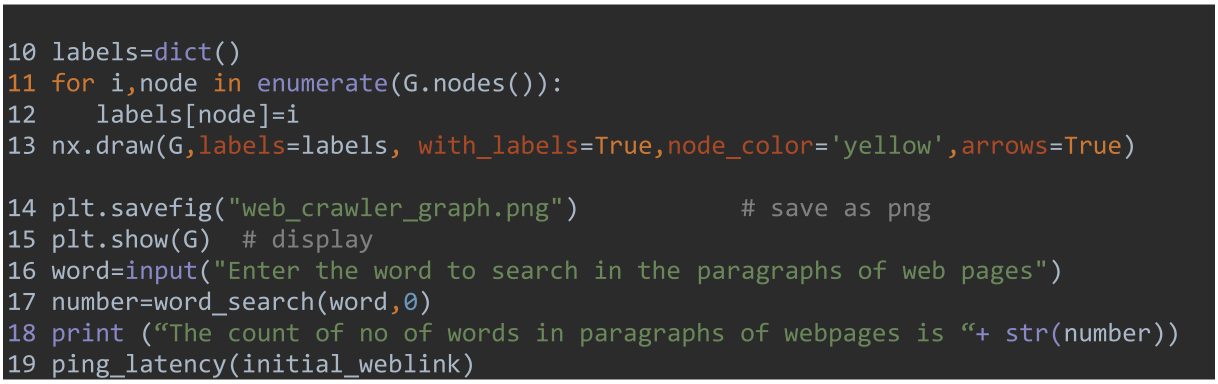
From line 19 to 25, the recursive logic is written which iterates each sublink in the absolute_link variable, and checks if it is in the visited_sublinks.

If it is not found in the visited sublinks then web_crawl() function is called with this sublink value, else it does nothing. This enables the Web Crawler program to recursively call for each of the web sub links without going into an infinite loop

Function ***main()***

```
1 user_weblink=input("Enter the website link to crawl")
  default_weblink=’http://www.oslomet.no/alumni’
2 if user_weblink in not null:
        web_crawl(user_weblink, 0)
  else:
        web_crawl(default_weblink,0)
3 node_number=1
4 for weblink in visited_sublinks:
5    add_nodes(weblink,node_number)
6    node_number+=1
7    print(weblink)

8  for entry in visited_edges:
9    add_edges(entry[0], entry[1], entry[2])
```



```
10 labels=dict()
11 for i,node in enumerate(G.nodes()):
12    labels[node]=i
13 nx.draw(G,labels=labels, with_labels=True,node_color='yellow',arrows=True)

14 plt.savefig("web_crawler_graph.png")           # save as png
15 plt.show(G)  # display
16 word=input("Enter the word to search in the paragraphs of web pages")
17 number=word_search(word,0)
18 print (“The count of no of words in paragraphs of webpages is “+ str(number))
19 ping_latency(initial_weblink)
```

The main function is the starting point of the Web Crawler program. It asks for the user input to enter the website link.

If the user enters the website link, the web_crawl function is called with the user input, else it is called with the default_weblink.

From lines 4 to 7, above it adds the nodes to the Graph G. in lines 8 and 9, it adds the edges to the Graph from the visited_edges list.

In lines 12 and 13, it labels the nodes based on the sequential numbers 0,1,2,… and node color as yellow.

In line 14, 15 , it shows the figure and saves the fig as png file.

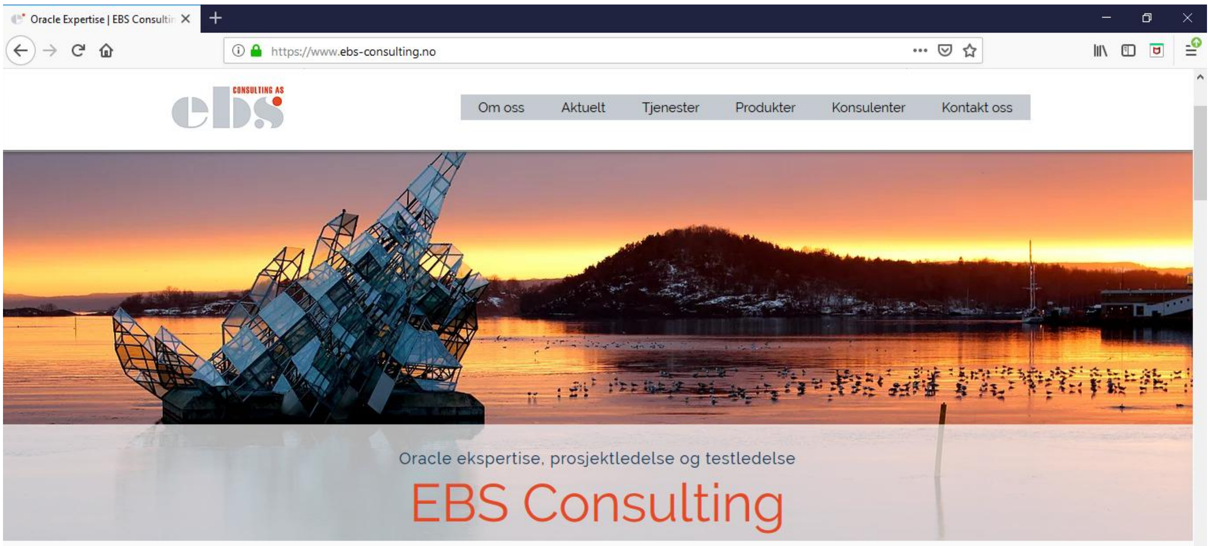
In line number 16, it asks for user input to enter the word to search in the Paragraphs of webpages.

In line 17,18, it calls the function word_search with word and 0 as parameters and prints the count of the words occurrences on the output screen.

In line 19, the program calls the ping_latency function to monitor the response time of ping to the website after every 10 minutes.

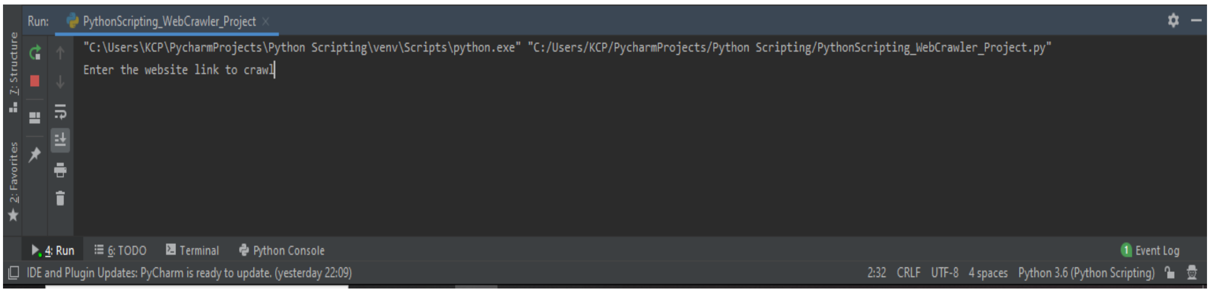
Test Execution Results:

Test Case 1: Execute the Web Crawler Program with the website link http://www.ebs-consulting.no/ for the first time.


Oracle Expertise | EBS Consulti
https://www.ebs-consulting.no
Om oss
Aktuelt
Tjenester
Produkter
Konsulenter
Kontakt oss
Oracle ekspertise, prosjektledelse og testledelse
EBS Consulting

The Web Crawler Program is executed.

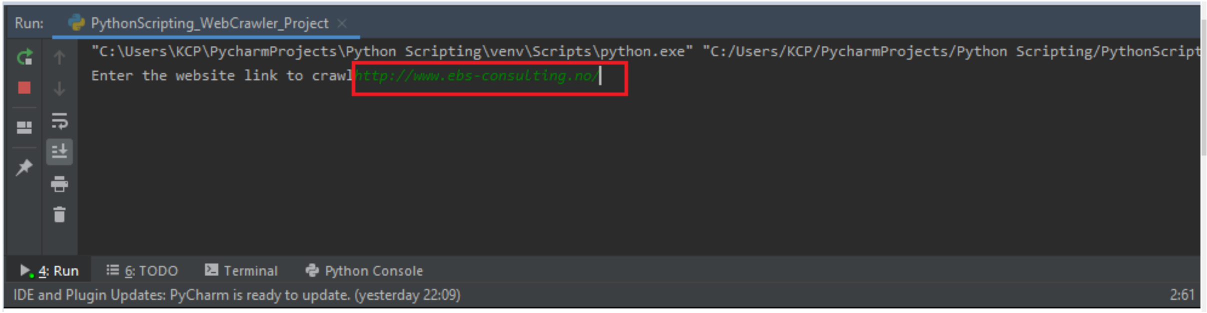
The program asks for user to input the Website link to crawl.



```
"C:\Users\KCP\PycharmProjects\Python Scripting\venv\Scripts\python.exe" "C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py"
Enter the website link to crawl
```

```
IDE and Plugin Updates: PyCharm is ready to update. (yesterday 22:09)    2:32  CRLF  UTF-8  4 spaces  Python 3.6 (Python Scripting)
```

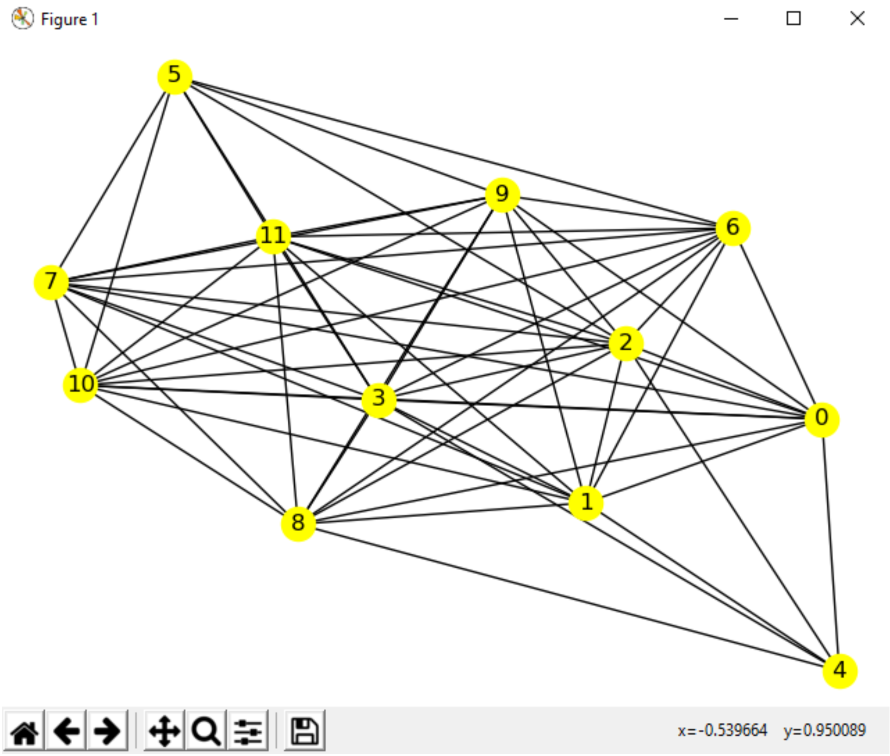
The website link http://www.ebs-consulting.no/ is entered as below:



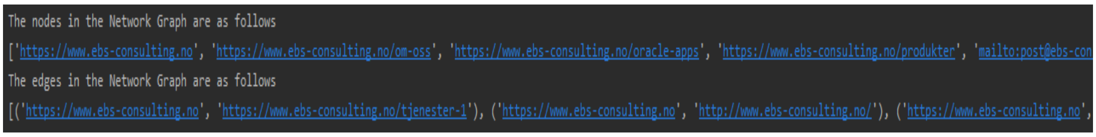
```
"C:\Users\KCP\PycharmProjects\Python Scripting\venv\Scripts\python.exe" "C:/Users/KCP/PycharmProjects/Python Scripting/PythonScript
Enter the website link to crawlhttp://www.ebs-consulting.no/
```

```
IDE and Plugin Updates: PyCharm is ready to update. (yesterday 22:09)    2:61
```

A network Graph is generated in the Figure image as below for the Website which was crawled by the Web Crawler program. The nodes in the graph are the web links, and the edges in the graphs are mapping between the web link and its corresponding sub links.

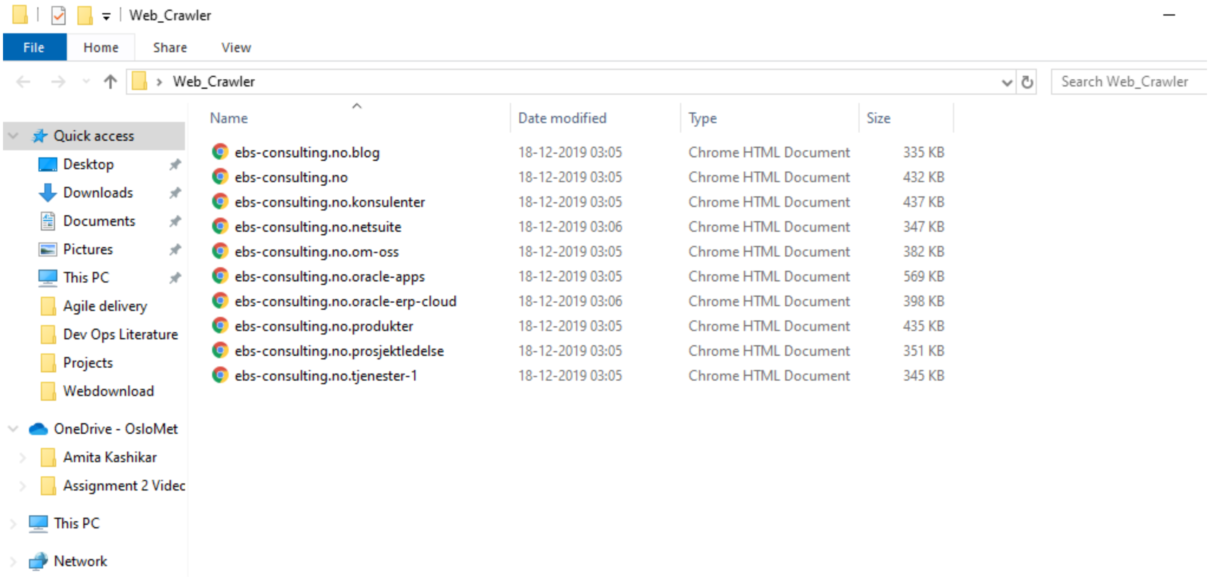


The nodes in the Graph and the Edges in the graph are displayed in the output of the program



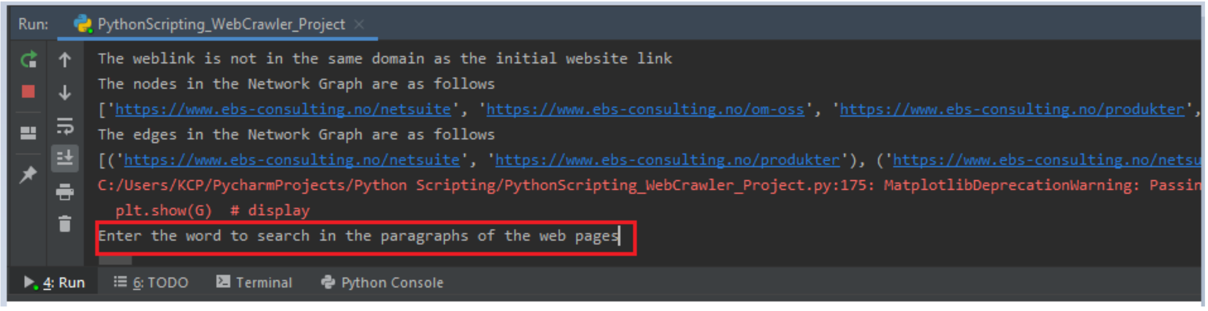
```
The nodes in the Network Graph are as follows
['https://www.ebs-consulting.no', 'https://www.ebs-consulting.no/om-oss', 'https://www.ebs-consulting.no/oracle-apps', 'https://www.ebs-consulting.no/produkter', 'mailto:post@ebs-con
The edges in the Network Graph are as follows
[('https://www.ebs-consulting.no', 'https://www.ebs-consulting.no/tjenester-1'), ('https://www.ebs-consulting.no', 'http://www.ebs-consulting.no/'), ('https://www.ebs-consulting.no',
```

The Web Pages are successfully downloaded with the dates in the directory folder as specified in the program



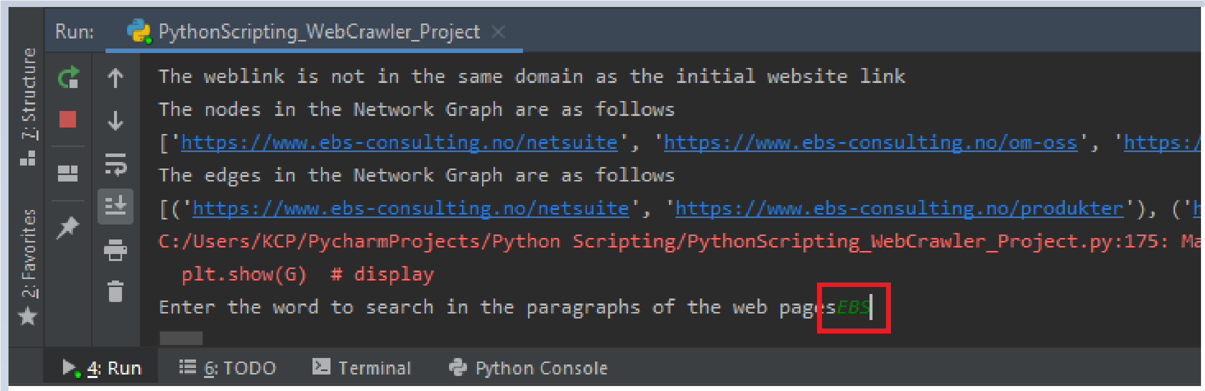
Name	Date modified	Type	Size
ebs-consulting.no.blog	18-12-2019 03:05	Chrome HTML Document	335 KB
ebs-consulting.no	18-12-2019 03:05	Chrome HTML Document	432 KB
ebs-consulting.no.konsulenter	18-12-2019 03:05	Chrome HTML Document	437 KB
ebs-consulting.no.netsuite	18-12-2019 03:06	Chrome HTML Document	347 KB
ebs-consulting.no.om-oss	18-12-2019 03:05	Chrome HTML Document	382 KB
ebs-consulting.no.oracle-apps	18-12-2019 03:05	Chrome HTML Document	569 KB
ebs-consulting.no.oracle-erp-cloud	18-12-2019 03:06	Chrome HTML Document	398 KB
ebs-consulting.no.produkter	18-12-2019 03:05	Chrome HTML Document	435 KB
ebs-consulting.no.prosjektledelse	18-12-2019 03:05	Chrome HTML Document	351 KB
ebs-consulting.no.tjenester-1	18-12-2019 03:05	Chrome HTML Document	345 KB

Once the Figure 1 displayed for Network Graph is closed. The program prompts for user input to search for any word in the paragraphs of the downloaded web pages for the inputted website links.



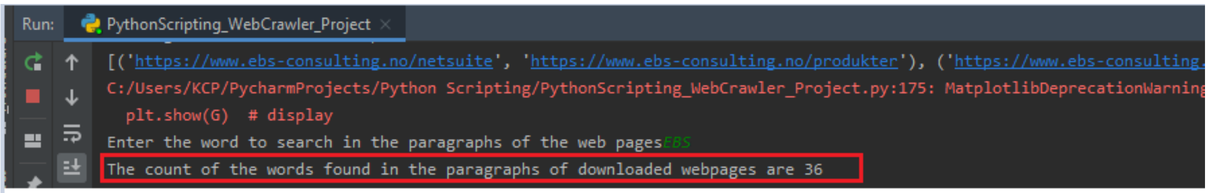
```
The weblink is not in the same domain as the initial website link
The nodes in the Network Graph are as follows
['https://www.ebs-consulting.no/netsuite', 'https://www.ebs-consulting.no/om-oss', 'https://www.ebs-consulting.no/produkter',
The edges in the Network Graph are as follows
[('https://www.ebs-consulting.no/netsuite', 'https://www.ebs-consulting.no/produkter'), ('https://www.ebs-consulting.no/netsu
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:175: MatplotlibDeprecationWarning: Passin
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pages
```

A word to search input is provided : “EBS” as in the below screenshot and Enter button is clicked.



```
The weblink is not in the same domain as the initial website link
The nodes in the Network Graph are as follows
['https://www.ebs-consulting.no/netsuite', 'https://www.ebs-consulting.no/om-oss', 'https:/
The edges in the Network Graph are as follows
[('https://www.ebs-consulting.no/netsuite', 'https://www.ebs-consulting.no/produkter'), ('h
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:175: Ma
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
```

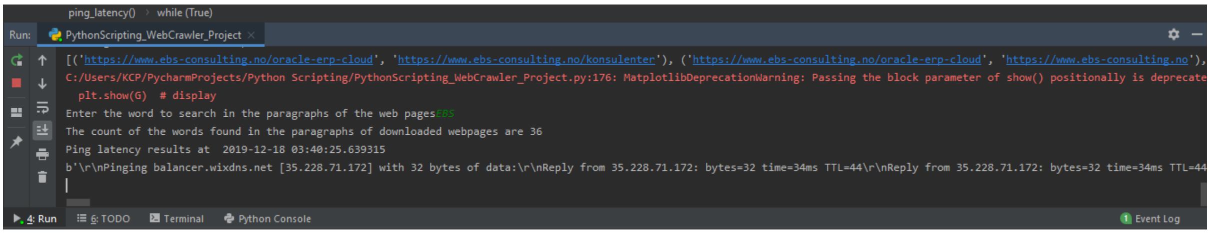
The output is displayed with the count of words found in the Paragraphs of the downloaded web pages as below.



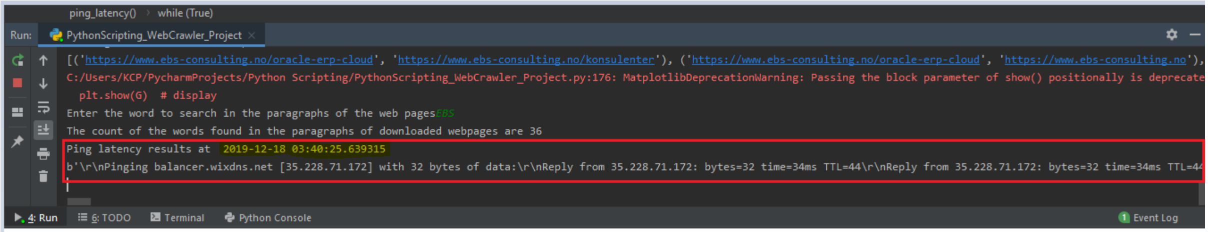
```
[('https://www.ebs-consulting.no/netsuite', 'https://www.ebs-consulting.no/produkter'), ('https://www.ebs-consulting.
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:175: MatplotlibDeprecationWarning
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
The count of the words found in the paragraphs of downloaded webpages are 36
```

The response time of the website using ping is monitored. The website is pinged every 10 minutes to check the response time of the website ping latency and the results are displayed on the output screen.

The results of the ping latency of the website get displayed as below along with the Date and Time of ping command.

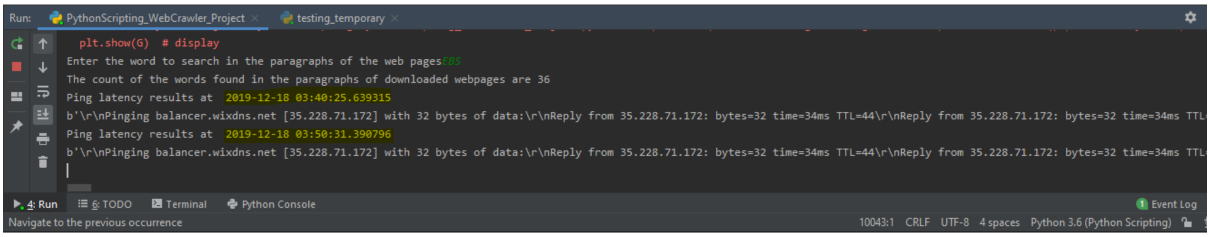

ping_latency() › while (True)
Run:
PythonScripting_WebCrawler_Project
4: Run
6: TODO
Terminal
Python Console
Event Log

```
[('https://www.ebs-consulting.no/oracle-erp-cloud', 'https://www.ebs-consulting.no/konsulenter'), ('https://www.ebs-consulting.no/oracle-erp-cloud', 'https://www.ebs-consulting.no'),
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:176: MatplotlibDeprecationWarning: Passing the block parameter of show() positionally is deprecate
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
The count of the words found in the paragraphs of downloaded webpages are 36
Ping latency results at  2019-12-18 03:40:25.639315
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44
```

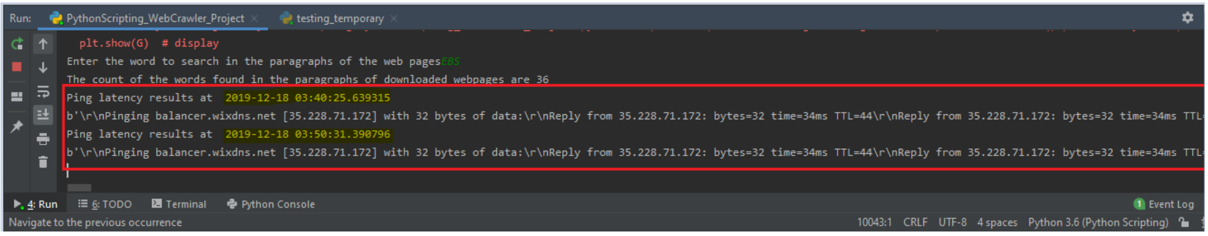

ping_latency() › while (True)
Run:
PythonScripting_WebCrawler_Project
4: Run
6: TODO
Terminal
Python Console
Event Log

```
[('https://www.ebs-consulting.no/oracle-erp-cloud', 'https://www.ebs-consulting.no/konsulenter'), ('https://www.ebs-consulting.no/oracle-erp-cloud', 'https://www.ebs-consulting.no'),
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:176: MatplotlibDeprecationWarning: Passing the block parameter of show() positionally is deprecate
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
The count of the words found in the paragraphs of downloaded webpages are 36
Ping latency results at  2019-12-18 03:40:25.639315
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44
```

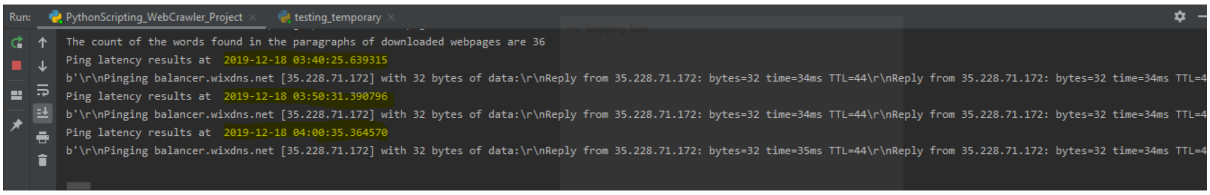
After 10 minutes are elapsed, the program pings the website again to monitor the response time and the ping latency results are displayed on the screen.


Run:
PythonScripting_WebCrawler_Project
testing_temporary
4: Run
6: TODO
Terminal
Python Console
Event Log
Navigate to the previous occurrence
10043:1 CRLF UTF-8 4 spaces Python 3.6 (Python Scripting)

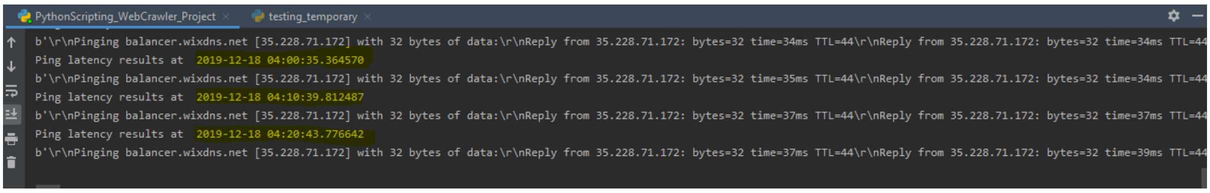
```
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
The count of the words found in the paragraphs of downloaded webpages are 36
Ping latency results at  2019-12-18 03:40:25.639315
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL
Ping latency results at  2019-12-18 03:50:31.390796
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL
```


Run:
PythonScripting_WebCrawler_Project
testing_temporary
4: Run
6: TODO
Terminal
Python Console
Event Log
Navigate to the previous occurrence
10043:1 CRLF UTF-8 4 spaces Python 3.6 (Python Scripting)

```
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesEBS
The count of the words found in the paragraphs of downloaded webpages are 36
Ping latency results at  2019-12-18 03:40:25.639315
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL
Ping latency results at  2019-12-18 03:50:31.390796
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL
```

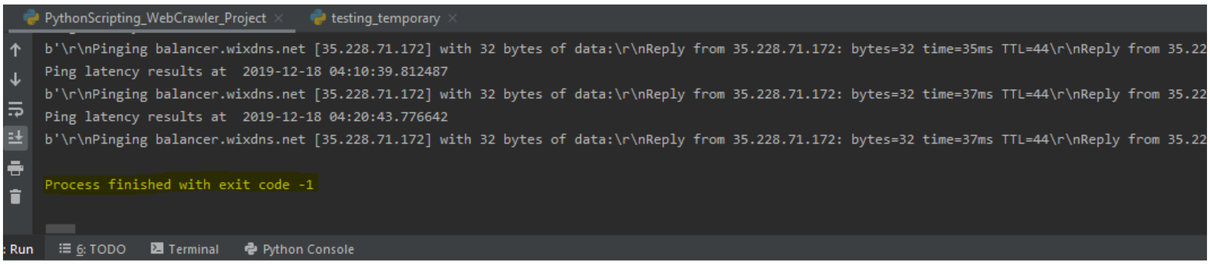


```
The count of the words found in the paragraphs of downloaded webpages are 36
Ping latency results at  2019-12-18 03:40:25.639315
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=4
Ping latency results at  2019-12-18 03:50:31.390796
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=4
Ping latency results at  2019-12-18 04:00:35.364570
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=35ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=4
```



```
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44
Ping latency results at  2019-12-18 04:00:35.364570
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=35ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=34ms TTL=44
Ping latency results at  2019-12-18 04:10:39.812487
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=37ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=37ms TTL=44
Ping latency results at  2019-12-18 04:20:43.776642
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=37ms TTL=44\r\nReply from 35.228.71.172: bytes=32 time=39ms TTL=44
```

Terminate the web crawler program manually.

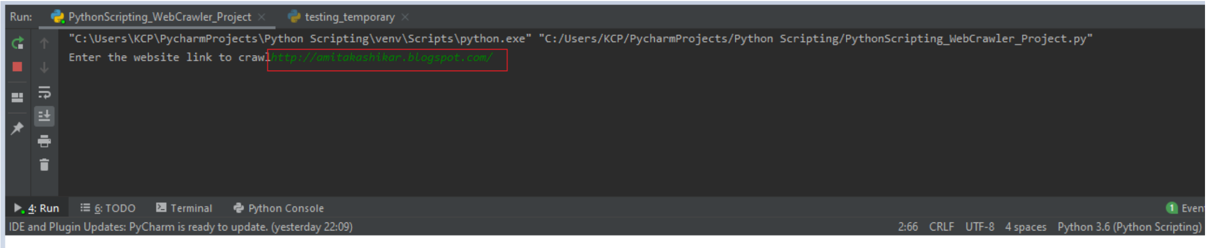


```
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=35ms TTL=44\r\nReply from 35.22
Ping latency results at  2019-12-18 04:10:39.812487
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=37ms TTL=44\r\nReply from 35.22
Ping latency results at  2019-12-18 04:20:43.776642
b'\r\nPinging balancer.wixdns.net [35.228.71.172] with 32 bytes of data:\r\nReply from 35.228.71.172: bytes=32 time=37ms TTL=44\r\nReply from 35.22

Process finished with exit code -1
```

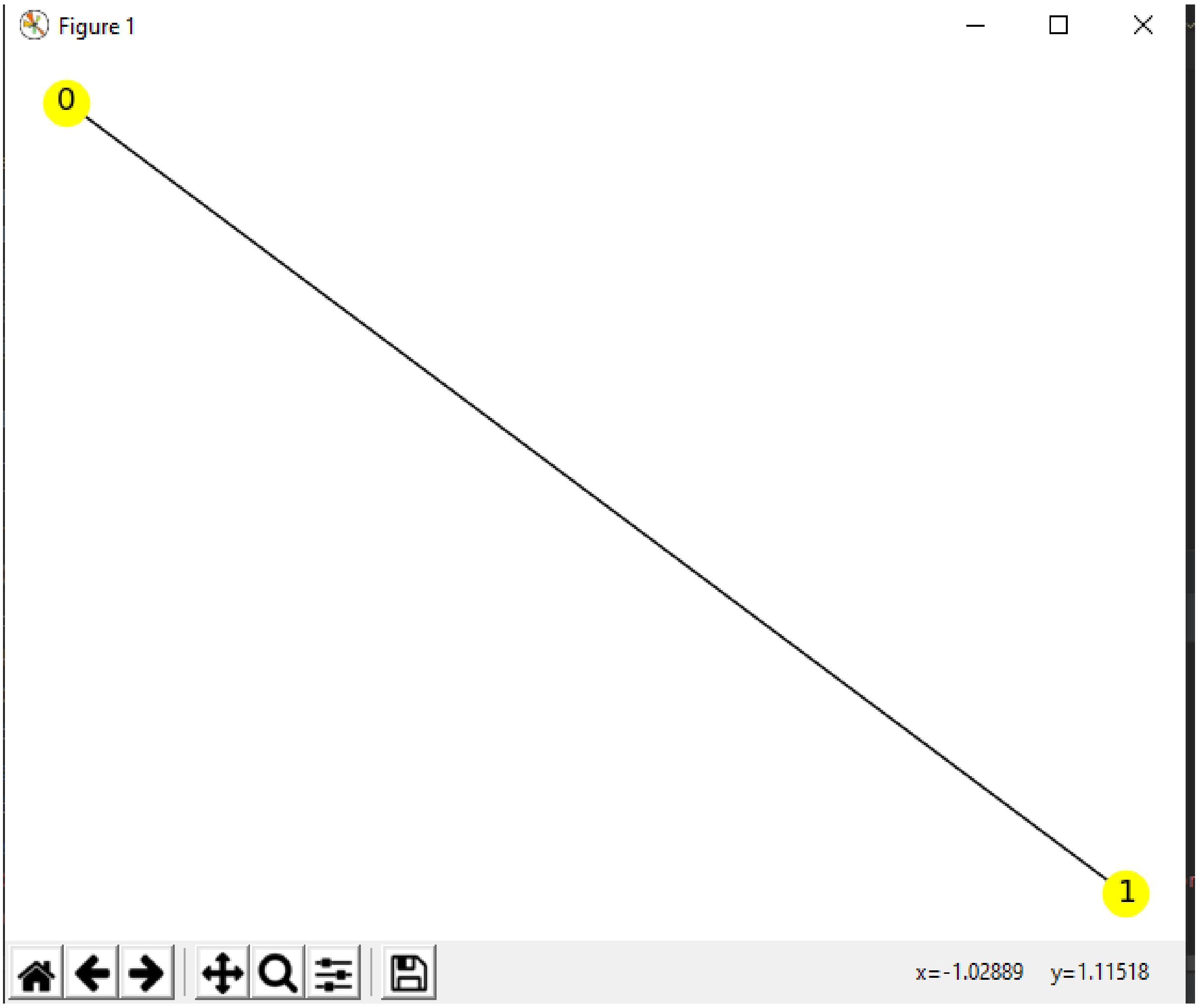
The web crawler program successfully crawls the website over the web links contained in the same domain and downloads all the web pages in the directory folder on the system. It always successfully searches the count of the word inputted in all the paragraphs of the webpages downloaded.It also displays the ping latency results on the output screen every 10 minutes until the program is unable to ping the website, or the website didn’t respond to the ping command or if the program is manually terminated by the user.

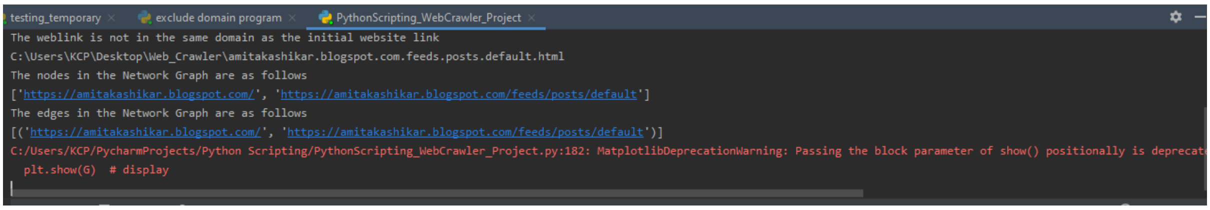
Test Case 2: Execute the Web Crawler Program with the website link http://amitakashikar.blogspot.com/ for the first time.



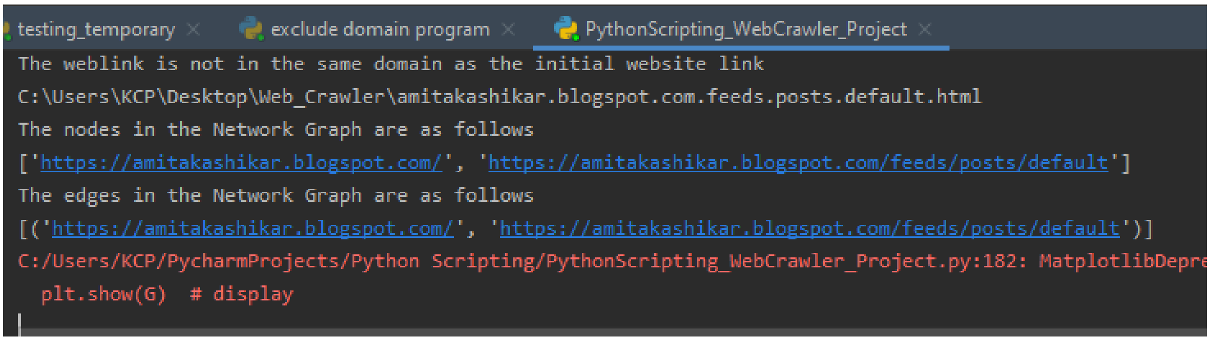
```
"C:\Users\KCP\PycharmProjects\Python Scripting\venv\Scripts\python.exe" "C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py"
Enter the website link to crawlhttp://amitakashikar.blogspot.com/
```

IDE and Plugin Updates: PyCharm is ready to update. (yesterday 22:09)   2:66 CRLF UTF-8 4 spaces Python 3.6 (Python Scripting)





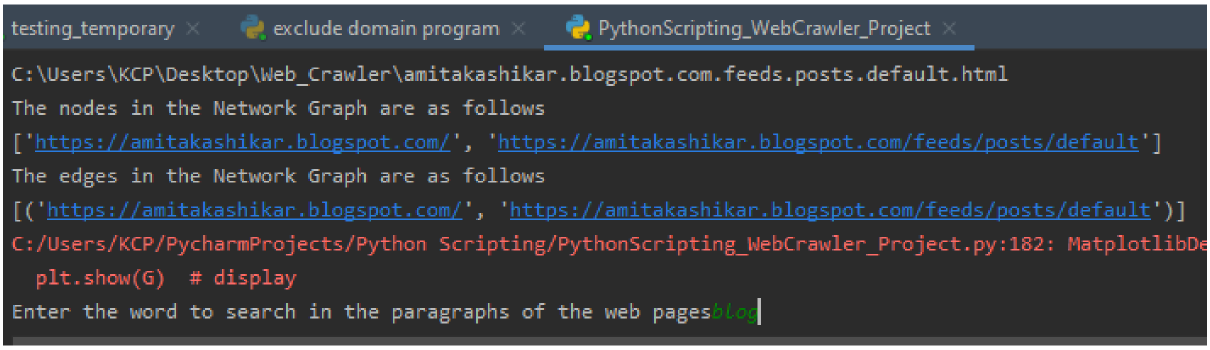
```
The weblink is not in the same domain as the initial website link
C:\Users\KCP\Desktop\Web_Crawler\amitakashikar.blogspot.com.feeds.posts.default.html
The nodes in the Network Graph are as follows
['https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default']
The edges in the Network Graph are as follows
[('https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default')]
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:182: MatplotlibDeprecationWarning: Passing the block parameter of show() positionally is deprecat
  plt.show(G)  # display
```



```
The weblink is not in the same domain as the initial website link
C:\Users\KCP\Desktop\Web_Crawler\amitakashikar.blogspot.com.feeds.posts.default.html
The nodes in the Network Graph are as follows
['https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default']
The edges in the Network Graph are as follows
[('https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default')]
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:182: MatplotlibDepre
  plt.show(G)  # display
```

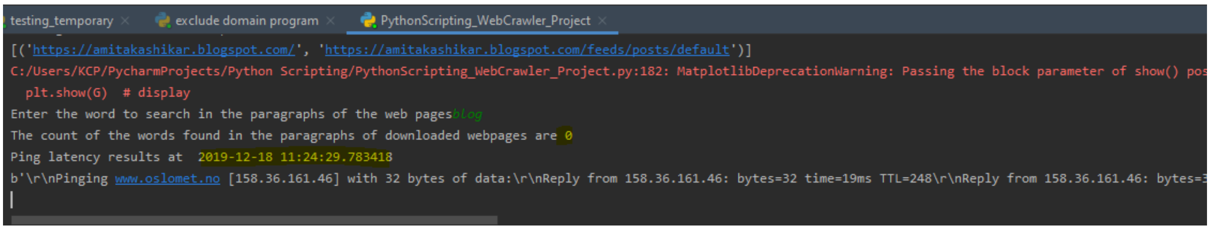
Most of the weblinks in the website are not in the same domain as of the main website therefore, the number of nodes which got displayed are very few.

Enter the word to search



```
C:\Users\KCP\Desktop\Web_Crawler\amitakashikar.blogspot.com.feeds.posts.default.html
The nodes in the Network Graph are as follows
['https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default']
The edges in the Network Graph are as follows
[('https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default')]
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:182: MatplotlibDe
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesblog
```

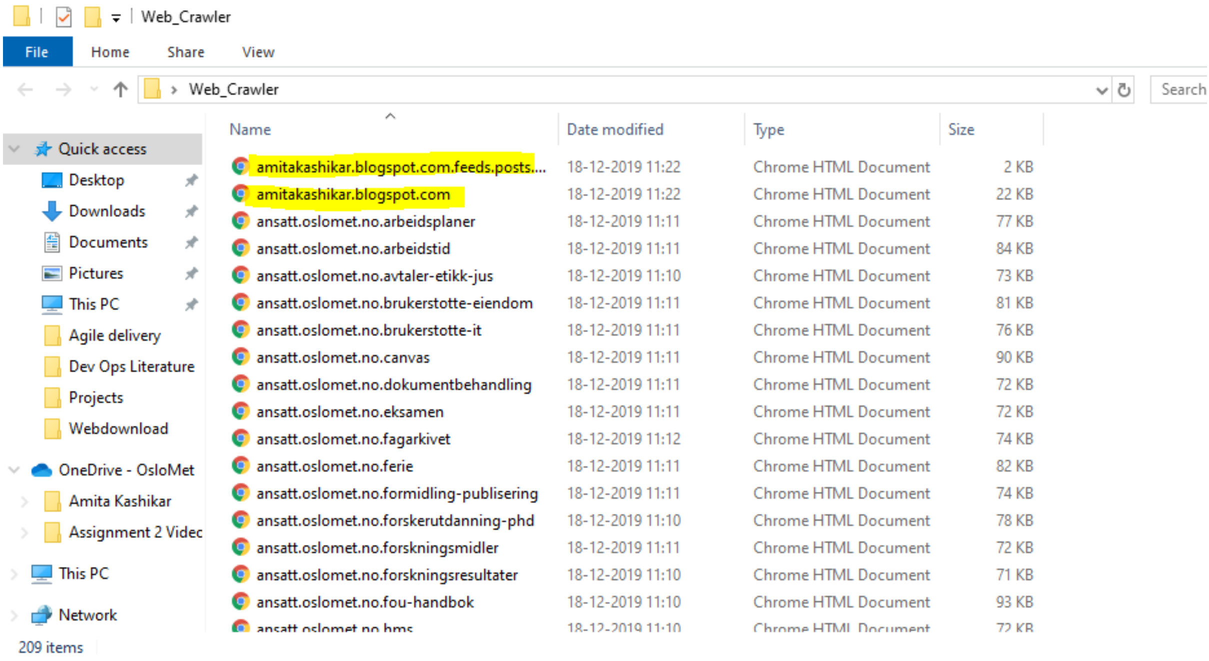
The number of words is zero.



```
[('https://amitakashikar.blogspot.com/', 'https://amitakashikar.blogspot.com/feeds/posts/default')]
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:182: MatplotlibDeprecationWarning: Passing the block parameter of show() pos
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesblog
The count of the words found in the paragraphs of downloaded webpages are 0
Ping latency results at  2019-12-18 11:24:29.783418
b'\r\nPinging www.oslomet.no [158.36.161.46] with 32 bytes of data:\r\nReply from 158.36.161.46: bytes=32 time=19ms TTL=248\r\nReply from 158.36.161.46: bytes=3
```

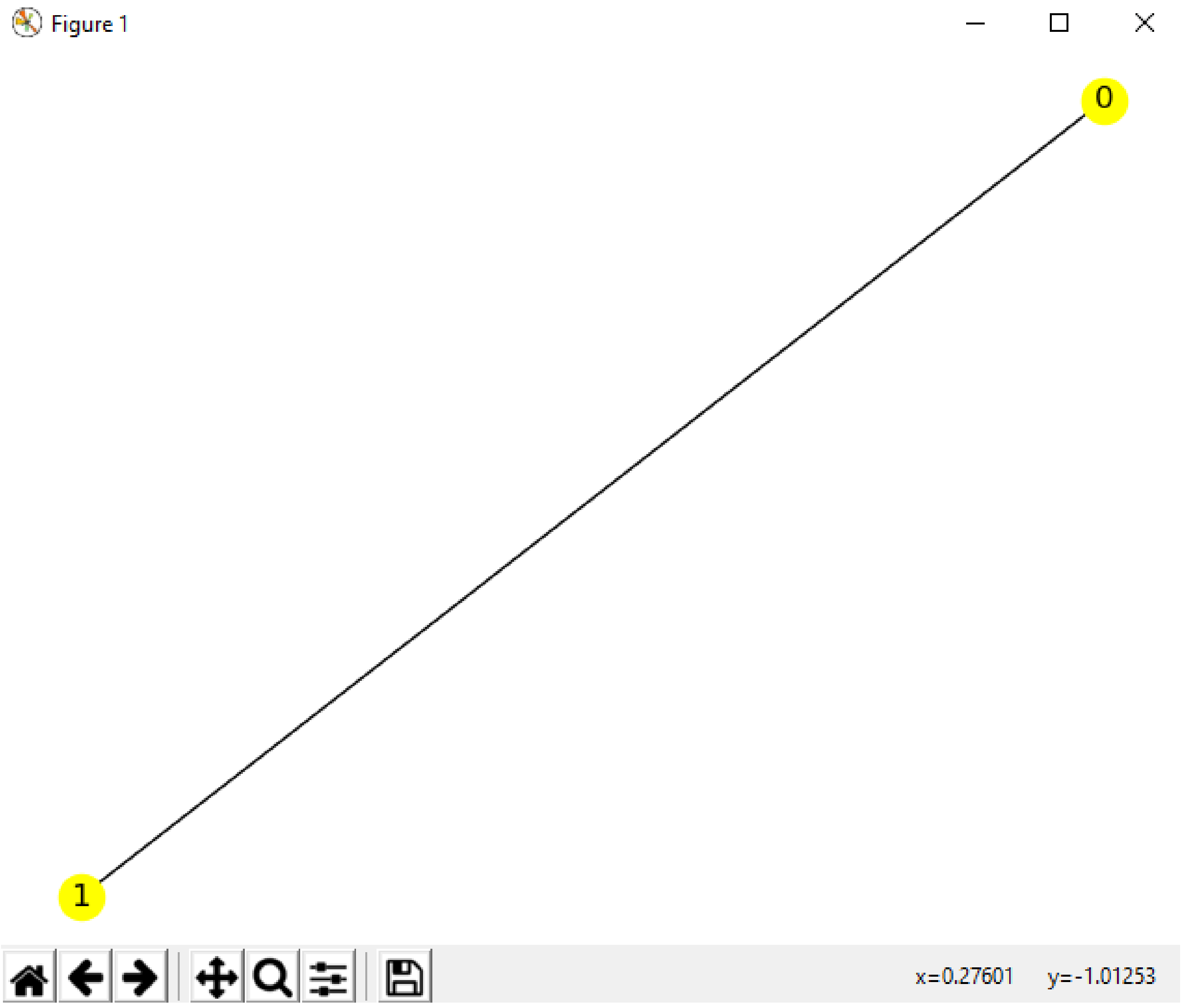
Ping Latency results are displayed

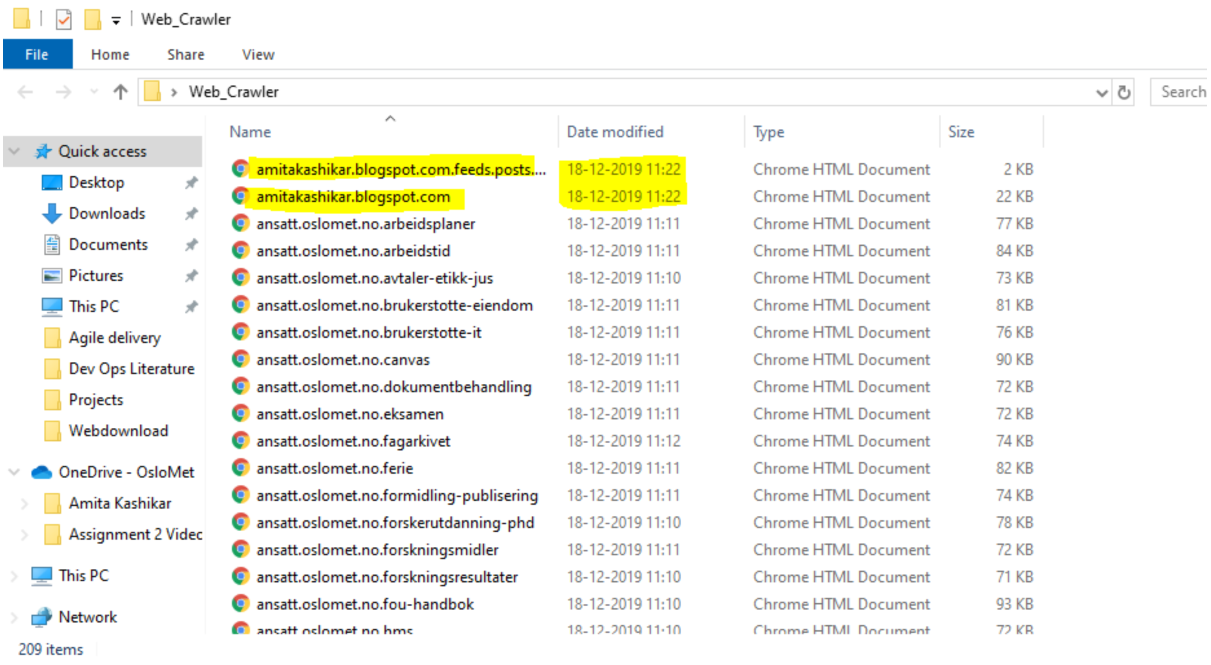
The web pages are saved to the desktop



Name	Date modified	Type	Size
amitakashikar.blogspot.com.feeds.posts....	18-12-2019 11:22	Chrome HTML Document	2 KB
amitakashikar.blogspot.com	18-12-2019 11:22	Chrome HTML Document	22 KB
ansatt.oslomet.no.arbeidsplaner	18-12-2019 11:11	Chrome HTML Document	77 KB
ansatt.oslomet.no.arbeidstid	18-12-2019 11:11	Chrome HTML Document	84 KB
ansatt.oslomet.no.avtaler-etikk-jus	18-12-2019 11:10	Chrome HTML Document	73 KB
ansatt.oslomet.no.brukerstotte-eiendom	18-12-2019 11:11	Chrome HTML Document	81 KB
ansatt.oslomet.no.brukerstotte-it	18-12-2019 11:11	Chrome HTML Document	76 KB
ansatt.oslomet.no.canvas	18-12-2019 11:11	Chrome HTML Document	90 KB
ansatt.oslomet.no.dokumentbehandling	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.eksamen	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.fagarkivet	18-12-2019 11:12	Chrome HTML Document	74 KB
ansatt.oslomet.no.ferie	18-12-2019 11:11	Chrome HTML Document	82 KB
ansatt.oslomet.no.formidling-publisering	18-12-2019 11:11	Chrome HTML Document	74 KB
ansatt.oslomet.no.forskerutdanning-phd	18-12-2019 11:10	Chrome HTML Document	78 KB
ansatt.oslomet.no.forskningsmidler	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.forskningsresultater	18-12-2019 11:10	Chrome HTML Document	71 KB
ansatt.oslomet.no.fou-handbok	18-12-2019 11:10	Chrome HTML Document	93 KB
ansatt.oslomet.no.hms	18-12-2019 11:10	Chrome HTML Document	72 KB

Test case 3:: Test the Web Crawler Program for a small website again by running at a different time http://amitakashikar.blogspot.com and view the results. Check that the webpages should not be downloaded again if the content has not changed.

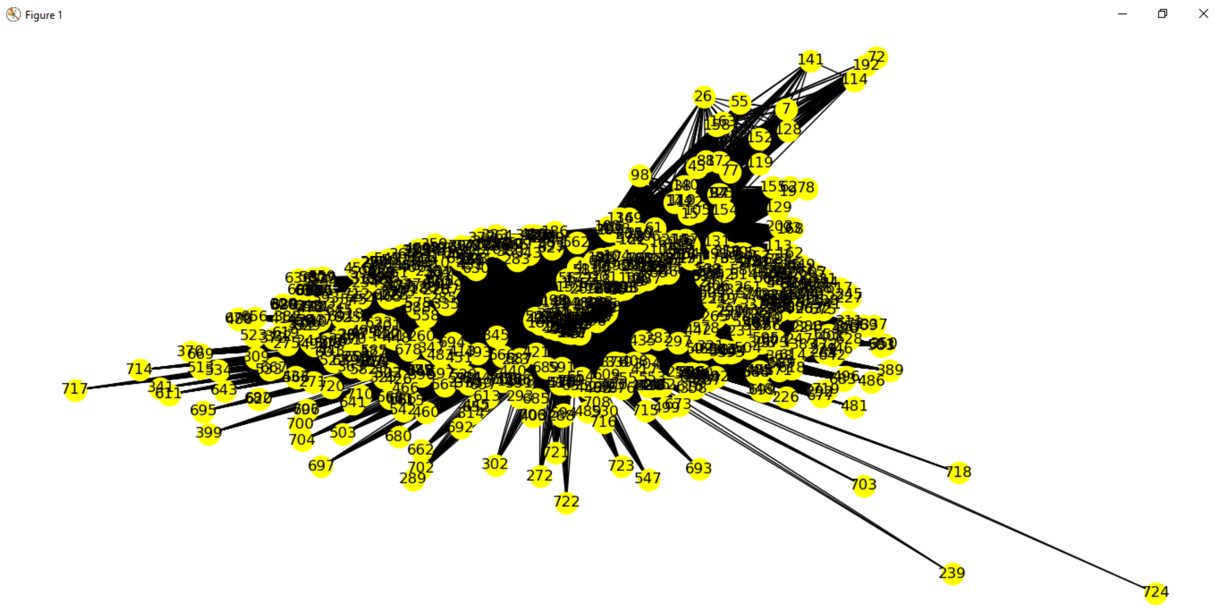




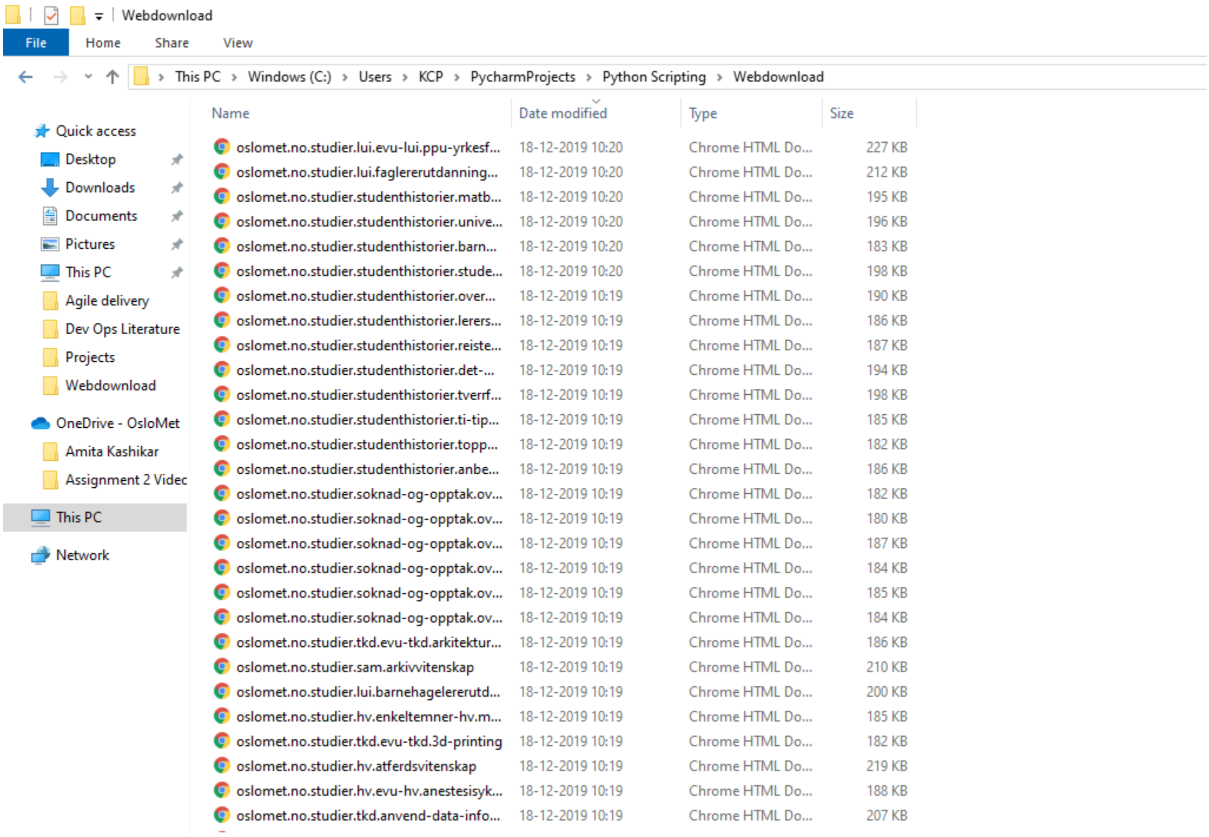
Name	Date modified	Type	Size
amitakashikar.blogspot.com.feeds.posts....	18-12-2019 11:22	Chrome HTML Document	2 KB
amitakashikar.blogspot.com	18-12-2019 11:22	Chrome HTML Document	22 KB
ansatt.oslomet.no.arbeidsplaner	18-12-2019 11:11	Chrome HTML Document	77 KB
ansatt.oslomet.no.arbeidstid	18-12-2019 11:11	Chrome HTML Document	84 KB
ansatt.oslomet.no.avtaler-etikk-jus	18-12-2019 11:10	Chrome HTML Document	73 KB
ansatt.oslomet.no.brukerstotte-eiendom	18-12-2019 11:11	Chrome HTML Document	81 KB
ansatt.oslomet.no.brukerstotte-it	18-12-2019 11:11	Chrome HTML Document	76 KB
ansatt.oslomet.no.canvas	18-12-2019 11:11	Chrome HTML Document	90 KB
ansatt.oslomet.no.dokumentbehandling	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.eksamen	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.fagarkivet	18-12-2019 11:12	Chrome HTML Document	74 KB
ansatt.oslomet.no.ferie	18-12-2019 11:11	Chrome HTML Document	82 KB
ansatt.oslomet.no.formidling-publisering	18-12-2019 11:11	Chrome HTML Document	74 KB
ansatt.oslomet.no.forskerutdanning-phd	18-12-2019 11:10	Chrome HTML Document	78 KB
ansatt.oslomet.no.forskningsmidler	18-12-2019 11:11	Chrome HTML Document	72 KB
ansatt.oslomet.no.forskningsresultater	18-12-2019 11:10	Chrome HTML Document	71 KB
ansatt.oslomet.no.fou-handbok	18-12-2019 11:10	Chrome HTML Document	93 KB
ansatt.oslomet.no.hms	18-12-2019 11:10	Chrome HTML Document	72 KB

Test Case 4: Execute the Web Crawler Program with the website link http://www.oslomet.no/alumni limiting the number of iterations as the website is too huge to crawl.

Network Graph of the web crawler for http://www.oslomet.no/alumni

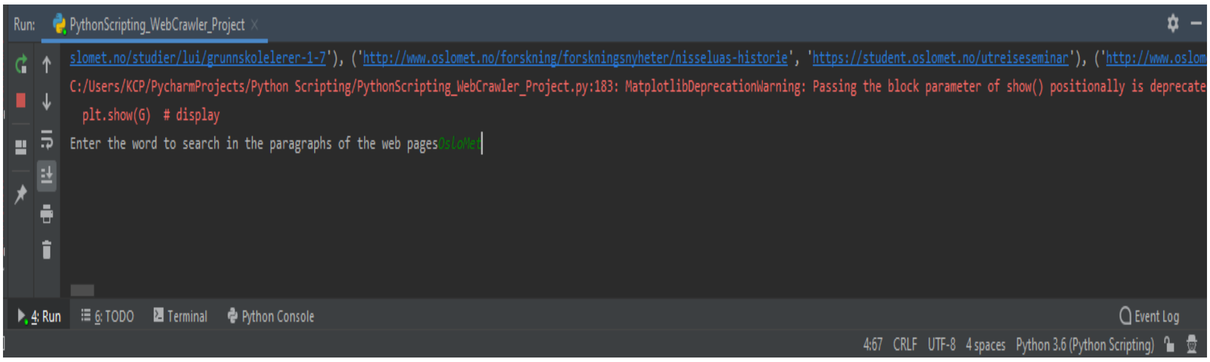


The contents are downloaded in the directory folder as below



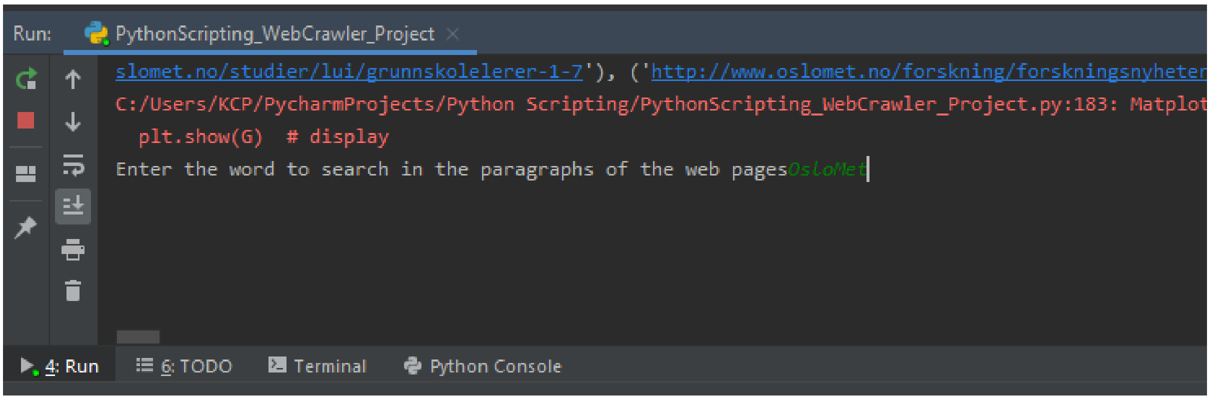
Name	Date modified	Type	Size
oslomet.no.studier.lui.evu-lui.ppu-yrkesf...	18-12-2019 10:20	Chrome HTML Do...	227 KB
oslomet.no.studier.lui.faglererutdanning...	18-12-2019 10:20	Chrome HTML Do...	212 KB
oslomet.no.studier.studenthistorier.matb...	18-12-2019 10:20	Chrome HTML Do...	195 KB
oslomet.no.studier.studenthistorier.unive...	18-12-2019 10:20	Chrome HTML Do...	196 KB
oslomet.no.studier.studenthistorier.barn...	18-12-2019 10:20	Chrome HTML Do...	183 KB
oslomet.no.studier.studenthistorier.stude...	18-12-2019 10:20	Chrome HTML Do...	198 KB
oslomet.no.studier.studenthistorier.over...	18-12-2019 10:19	Chrome HTML Do...	190 KB
oslomet.no.studier.studenthistorier.lerers...	18-12-2019 10:19	Chrome HTML Do...	186 KB
oslomet.no.studier.studenthistorier.reiste...	18-12-2019 10:19	Chrome HTML Do...	187 KB
oslomet.no.studier.studenthistorier.det-...	18-12-2019 10:19	Chrome HTML Do...	194 KB
oslomet.no.studier.studenthistorier.tverrf...	18-12-2019 10:19	Chrome HTML Do...	198 KB
oslomet.no.studier.studenthistorier.ti-tip...	18-12-2019 10:19	Chrome HTML Do...	185 KB
oslomet.no.studier.studenthistorier.topp...	18-12-2019 10:19	Chrome HTML Do...	182 KB
oslomet.no.studier.studenthistorier.anbe...	18-12-2019 10:19	Chrome HTML Do...	186 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	182 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	180 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	187 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	184 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	185 KB
oslomet.no.studier.soknad-og-opptak.ov...	18-12-2019 10:19	Chrome HTML Do...	184 KB
oslomet.no.studier.tkd.evu-tkd.arkitektur...	18-12-2019 10:19	Chrome HTML Do...	186 KB
oslomet.no.studier.sam.arkivvitenskap	18-12-2019 10:19	Chrome HTML Do...	210 KB
oslomet.no.studier.lui.barnehagelererutd...	18-12-2019 10:19	Chrome HTML Do...	200 KB
oslomet.no.studier.hv.enkeltemner-hv.m...	18-12-2019 10:19	Chrome HTML Do...	185 KB
oslomet.no.studier.tkd.evu-tkd.3d-printing	18-12-2019 10:19	Chrome HTML Do...	182 KB
oslomet.no.studier.hv.atferdsvitenskap	18-12-2019 10:19	Chrome HTML Do...	219 KB
oslomet.no.studier.hv.evu-hv.anestesisyk...	18-12-2019 10:19	Chrome HTML Do...	188 KB
oslomet.no.studier.tkd.anvend-data-info...	18-12-2019 10:19	Chrome HTML Do...	207 KB

Close the Fig in the program, and the program prompts to input a word to search.


Run:
PythonScripting_WebCrawler_Project
4: Run
6: TODO
Terminal
Python Console
Event Log
4:67 CRLF UTF-8 4 spaces Python 3.6 (Python Scripting)

```
slomet.no/studier/lui/grunnskolelerer-1-7'), ('http://www.oslomet.no/forskning/forskningsnyheter/nisseluas-historie', 'https://student.oslomet.no/utreiseseminar'), ('http://www.oslom
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:183: MatplotlibDeprecationWarning: Passing the block parameter of show() positionally is deprecate
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesOsloMet
```

Enter OsloMet

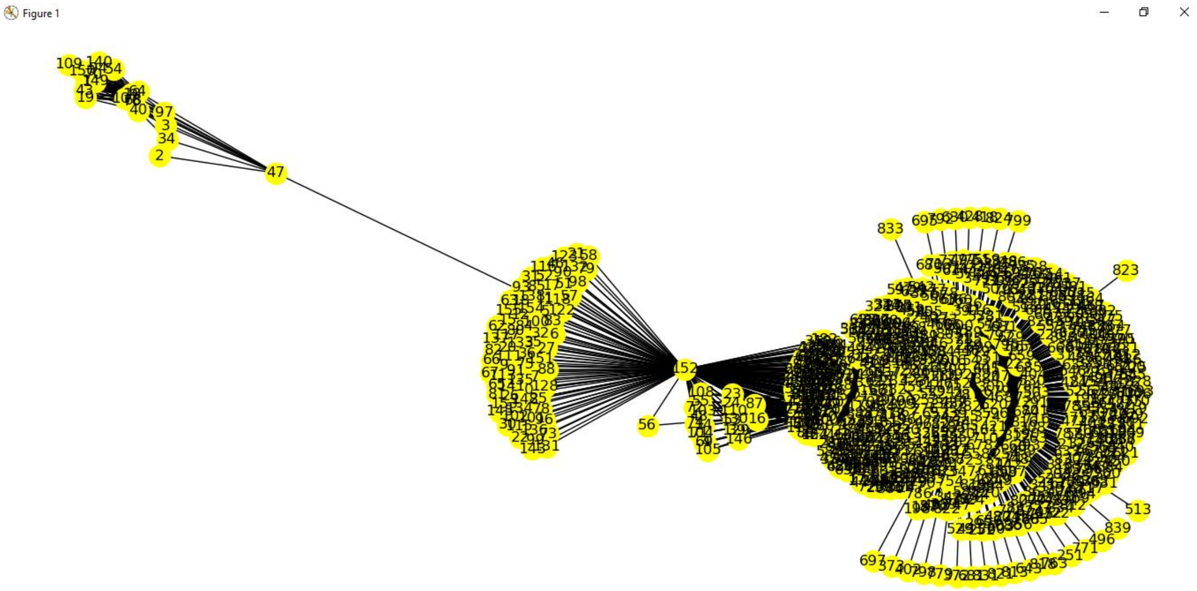


```
slomet.no/studier/lui/grunnskolelerer-1-7'), ('http://www.oslomet.no/forskning/forskningsnyheter
C:/Users/KCP/PycharmProjects/Python Scripting/PythonScripting_WebCrawler_Project.py:183: Matplot
  plt.show(G)  # display
Enter the word to search in the paragraphs of the web pagesOsloMet
```

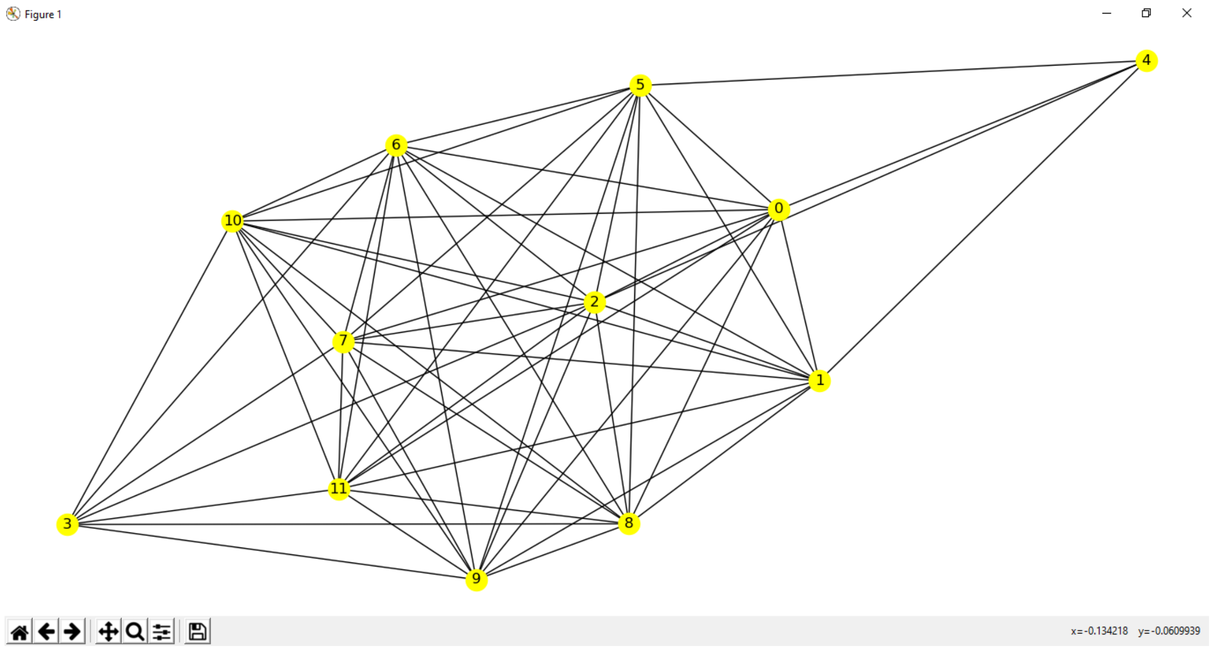
Displays the count of words and ping latency results.

Test Case 5:

For the website http://www.ebs-consulting/no, the Web Crawler program was executed before excluding the domains other than the original website link, and the network graph that was obtained was as below.



After the logic of excluding the domains outside the original website domain, a below network graph was obtained.



The Web Crawler Program is implemented successfully with all the functionalities in the Project description achieved.

References:

[1] ScienceDaily.com

[2] https://www.capitoltechsolutions.com/portfolio/website-scraping-script-optimizes-data-integration-saves-time-and-money/s