HW 1 - Web Science Intro

Prashant Tomar

1/29/2023

Q1

Draw the resulting directed graph (either sketch on paper or use another tool) showing how the nodes are connected to each other and include an image in your report. This does not need to fit into the bow-tie type diagram, but should look more similar to the graph on slide 24 from Module-01 Web-Science-Architecture.

Answer

Figure 1 represents the plotted direct graph from the question.

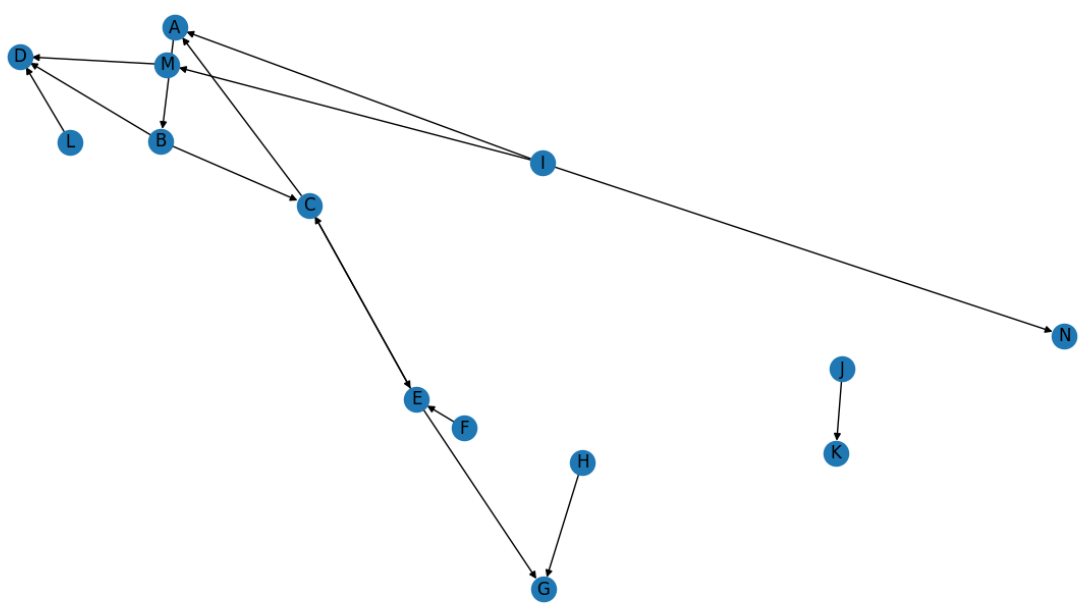

Figure 1: Directed Graph

Here is the python code to generate the directed graph.

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

# Create an empty directed graph
```

```python
G = nx.DiGraph()

# Add edges to the graph
G.add_edge("A", "B")
G.add_edge("B", "C")
G.add_edge("B", "D")
G.add_edge("C", "A")
G.add_edge("C", "E")
G.add_edge("E", "C")
G.add_edge("E", "G")
G.add_edge("F", "E")
G.add_edge("H", "G")
G.add_edge("I", "A")
G.add_edge("I", "M")
G.add_edge("I", "N")
G.add_edge("J", "K")
G.add_edge("L", "D")
G.add_edge("M", "D")

# Draw the graph
nx.draw(G, with_labels=True)
plt.show()
```

Listing 1: Python sample code loaded from file

Table 1 shows a nodes that are listed in the following categories.

Table 1: Simple Table

S.No.	Categories	Nodes
1	SCC:	A,B,C,E
2	IN:	D,G
3	OUT:	F,H,I,L
4	Tendrils:	N
5	Tubes:	M
6	Disconnected:	J,K

Q2

Demonstrate that you know how to use curl and are familiar with the available options

Answer

PART (a) First, load the URI directly in your browser and take a screenshot. The resulting webpage should show the ”User-Agent” HTTP request header that your web browser sends to the web server. This shows the directed web page in a browser.

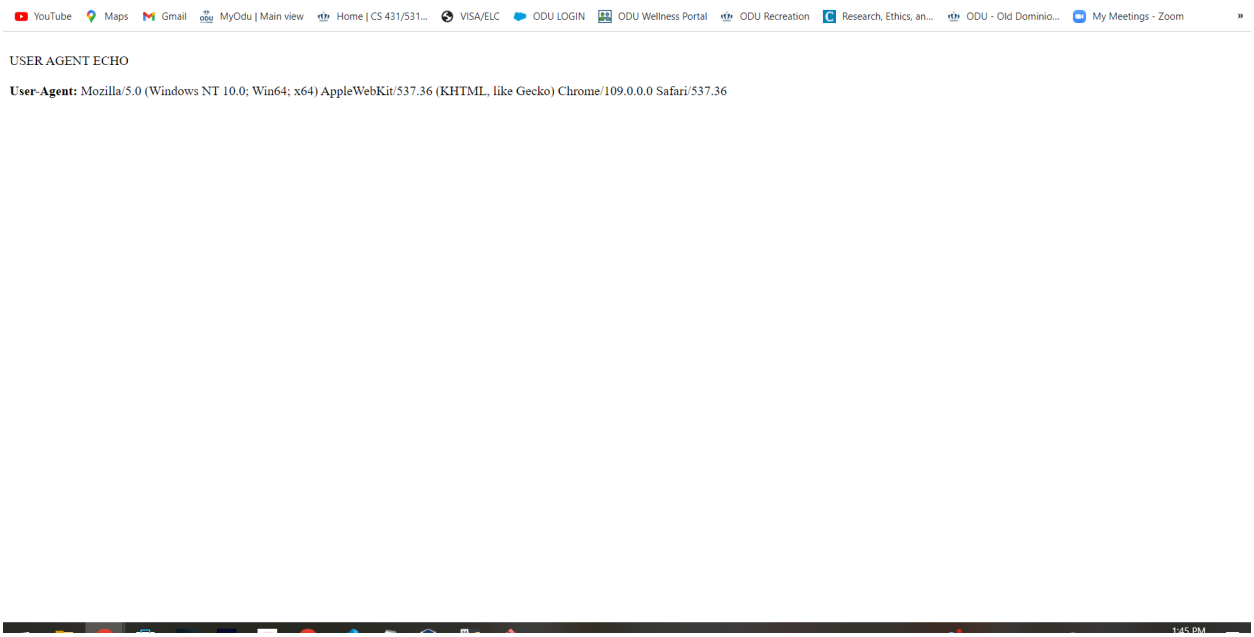

Figure 2: Webpage shows the ”User-Agent” HTTP request header that your web browser sends to the web server

This is the URI loaded in the browser.

PART (b) In a single curl command, request the URI, shows the HTTP response headers, following any redirects, and change the User-Agent HTTP request field to ”CS432/532”. Shown on command line.

```
> curl -I -L -H "User-Agent: CS432/532" http://www.cs.odu.edu/\~mweigle
  /courses/cs532/ua_echo.php
```

Listing 2: Python example copied into the LaTeX

In this CURL command -I: Show only the HTTP response headers, -L: Follow any redirects, -H: Set an HTTP request header

PART (c) In this single curl command, request the URI, follows any redirects, the User-Agent HTTP request field has been changed to ”CS432/532”, and then the HTML output been displayed below.

```
> curl -I -L -H "User-Agent: CS432/532" http://www.cs.odu.edu/\~mweigle
  /courses/cs532/ua_echo.php --output index.html
```

Listing 3: Python example copied into the LaTeX

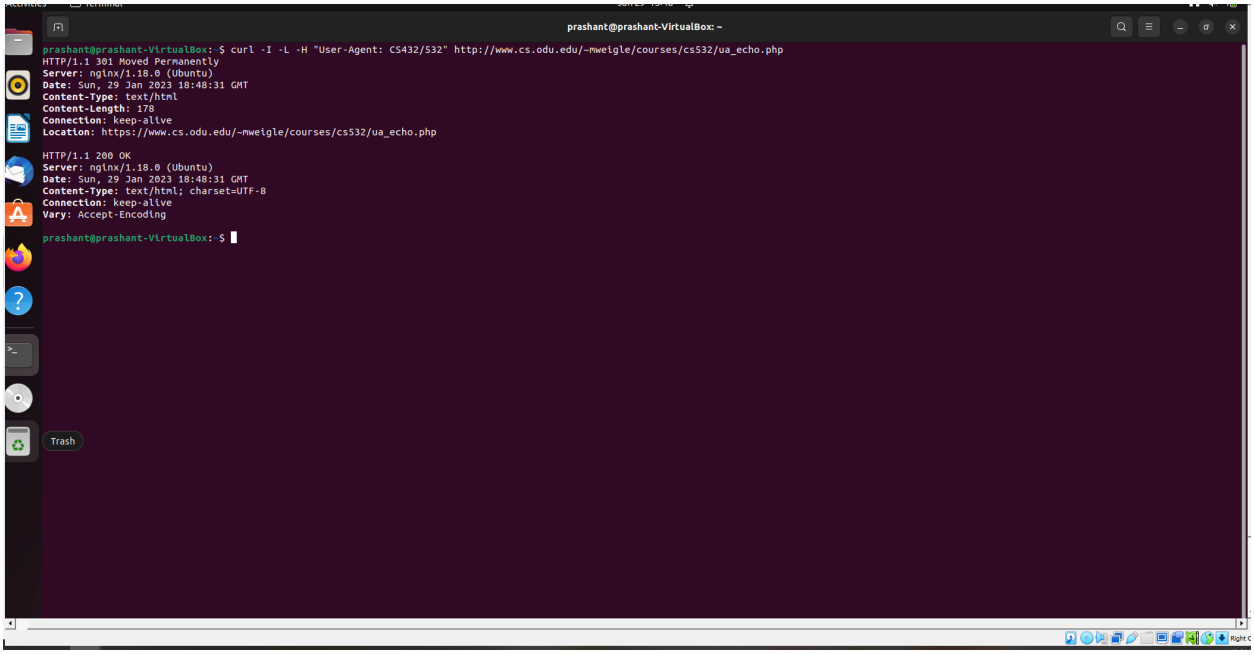

Figure 3: Requested the URI which shows the HTTP response headers, follow any redirects, and changed User-Agent HTTP request field to ”CS432/532”.

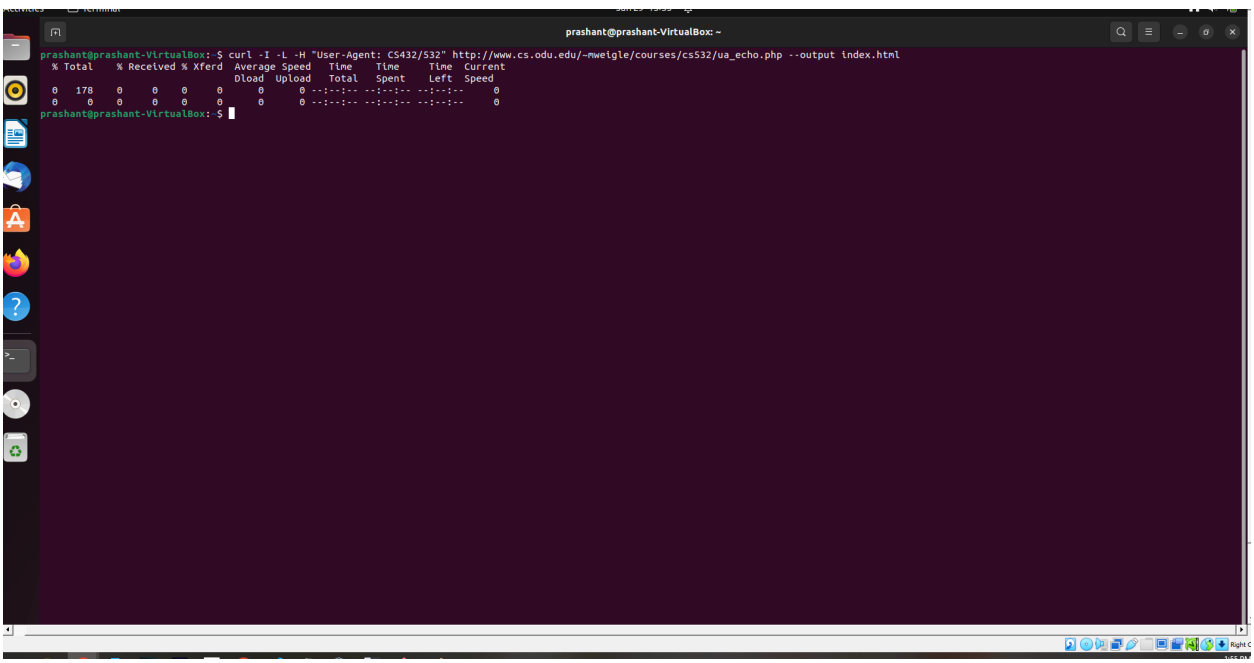

Figure 4: Giving path to html file

This is the output of the index page file.

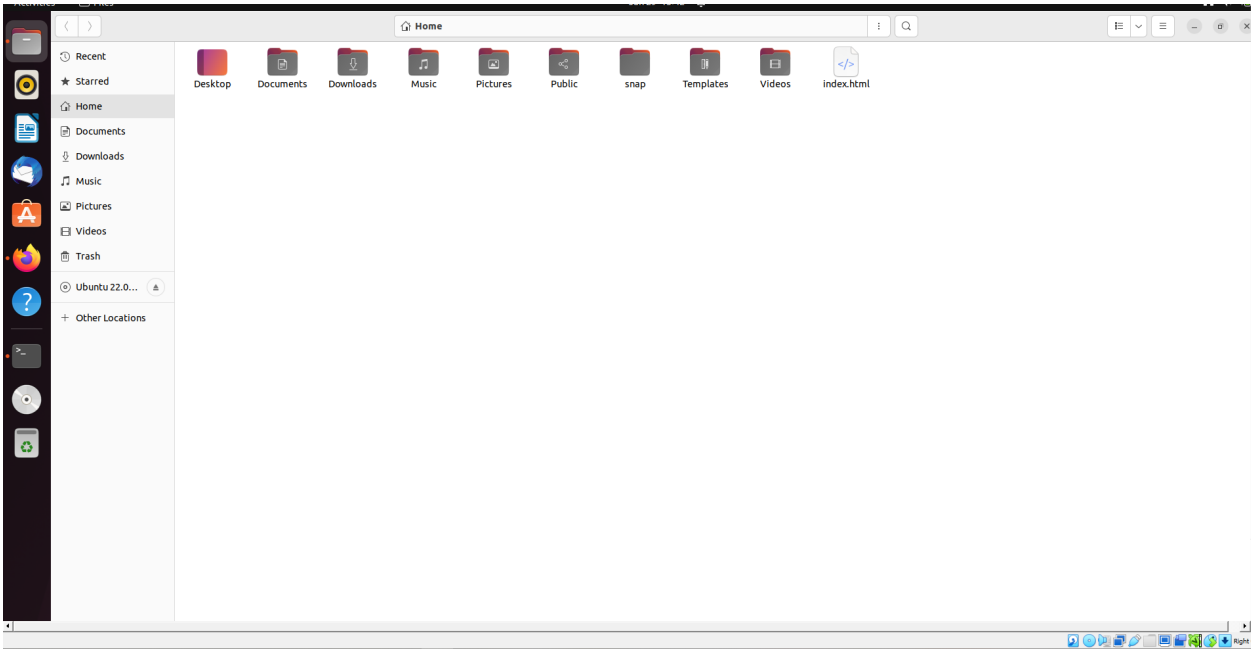

Figure 5: HTML file on local machine

Figure 6: User-Agent HTTP request field to ”CS432/532”

Q3

Write a Python program to find links to PDFs in a webpage

Answer

Here are some prints of the original URI (found in the source of the original HTML), the final URI (after any redirects), and the number of bytes in the PDF file.

Here is the python code to read the pdf from uri's.

```python
import requests
from bs4 import BeautifulSoup

def get_pdf_links(url):
    res = requests.get(url)
    soup = BeautifulSoup(res.content, "html.parser")
    links = [link.get("href") for link in soup.find_all("a")]
    pdf_links = []
    for link in links:
        try:
            if link.endswith(".pdf"):
                response = requests.head(link, allow_redirects=True)
                content_type = response.headers.get("Content-Type", "").lower()
                if "pdf" in content_type:
                    pdf_links.append((link, response.url, response.headers.get("Content-Length", 0)))
        except:
            pass
    return pdf_links

if __name__ == "__main__":
    import sys
    if len(sys.argv) < 2:
        print("Usage: python findPDFs.py URL")
        sys.exit(1)
    url = sys.argv[1]
    pdf_links = get_pdf_links(url)
    for link, final_url, size in pdf_links:
        print("URI:", link)
        print("Final URI:", final_url)
        print("Content Length:", size, "bytes")
        print()
```

Listing 4: Python sample code loaded from file

Here are the screenshots of the webpages after executing the python commands),

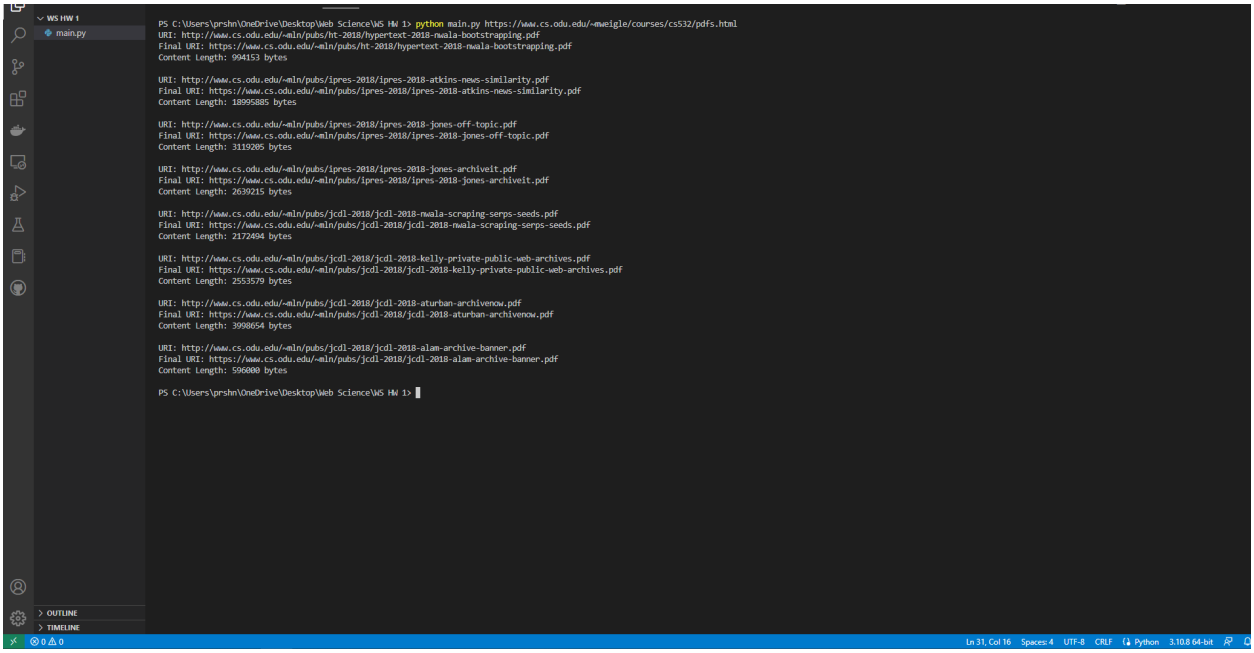

Figure 7: PDF:1

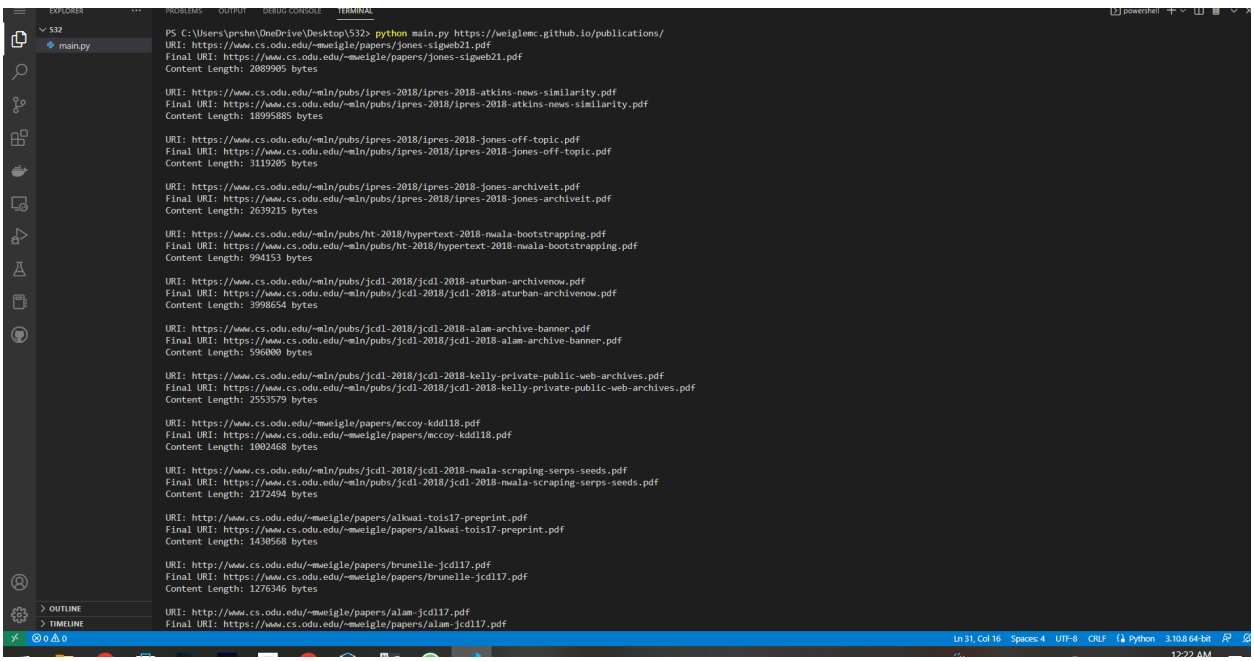

Figure 8: PDF:2

References

In this homework I have referred to python docs for installation of libraries and google developer doc for running the code.

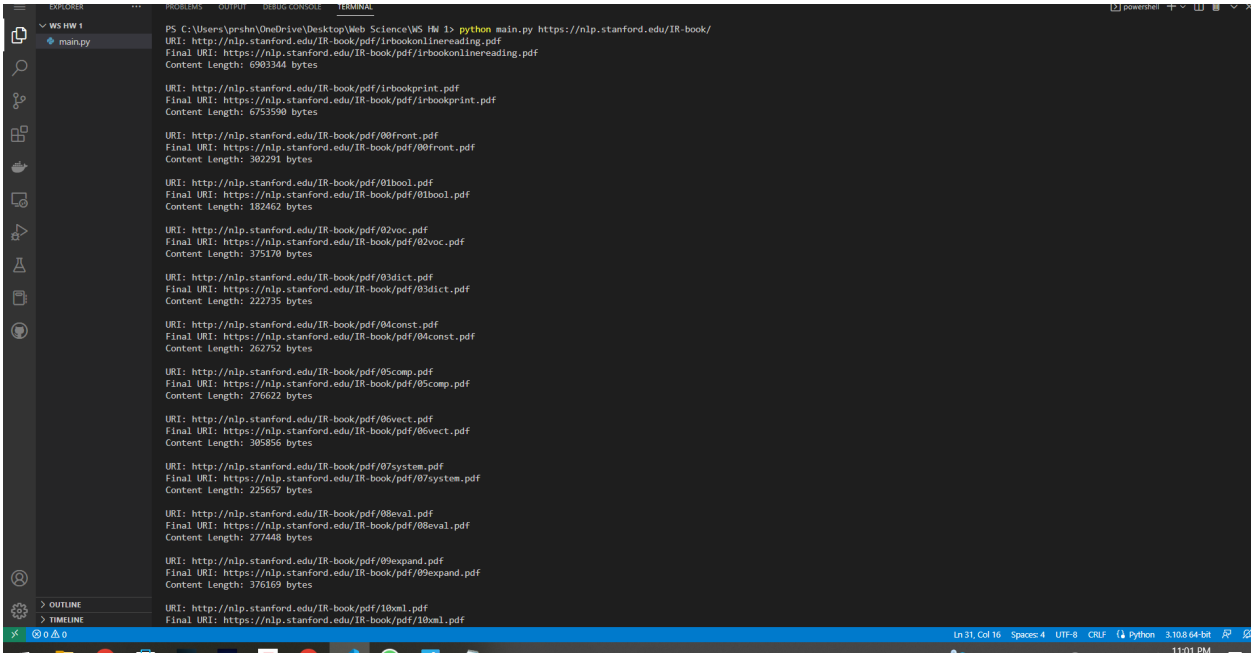

Figure 9: PDF:3

- Insert Reference 1, `https://docs.python-requests.org/en/v0.8.4/api/`
- Insert Reference 2, `https://developers.google.com/edu/python/regular-expressions`