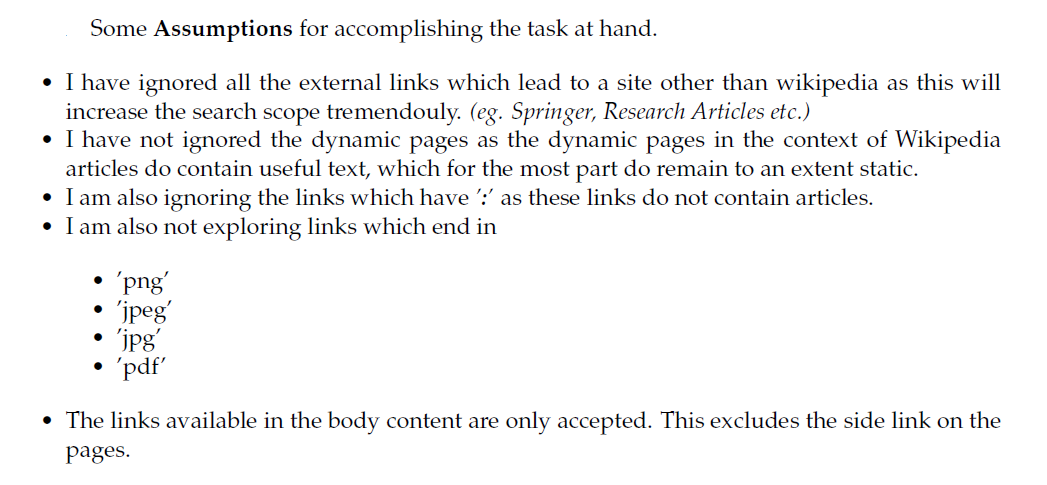
Experiment 3

**Basic Web Crawling**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | **:** | **Abhishek N N** | **Register No** | **:** | **20BCE1025** |
| Faculty | : | Dr. Alok Chauhan | Slot | : | L51-L52 AB1-605B |
| Course | : | Web Mining Lab | Code | : | CSE3024 |
| Programme | : | B.Tech CSE Core | Semester | : | Win – 22 - 23 |





**Ex.1**

Given a root URL, e.g., "https://en.wikipedia.org/wiki/Web\_mining", Design a simple crawler to return all pages that contains a string “crawler" from this site.

The keyword **crawler** was hard to find because <https://en.wikipedia.org/wiki/Web_mining> automatically redirects to <https://en.wikipedia.org/wiki/Data_mining>

So I used **mining** instead

Code:

from urllib.request import urlopen

from bs4 import BeautifulSoup

import re

pages = set()

pageUrl="https://en.wikipedia.org/wiki/Web\_mining"

html = urlopen(pageUrl.format(pageUrl))

bs = BeautifulSoup(html, 'html.parser')

s=str(bs).lower()

i=s.find('mining')

if i!=-1:

print(pageUrl)

print(s[max(0,i-100):min(len(s)-1,i+100)])

print('-'\*20)

for link in bs.find\_all('a', href=re.compile('^(/wiki/)')):

if 'href' in link.attrs:

if link.attrs['href'] not in pages:

#We have encountered a new page

newPage = link.attrs['href']

newPage="https://en.wikipedia.org/"+newPage

if newPage in pages:

continue

pages.add(newPage)

html = urlopen(newPage.format(newPage))

bs = BeautifulSoup(html, 'html.parser')

s=str(bs).lower()

i=s.find('mining')

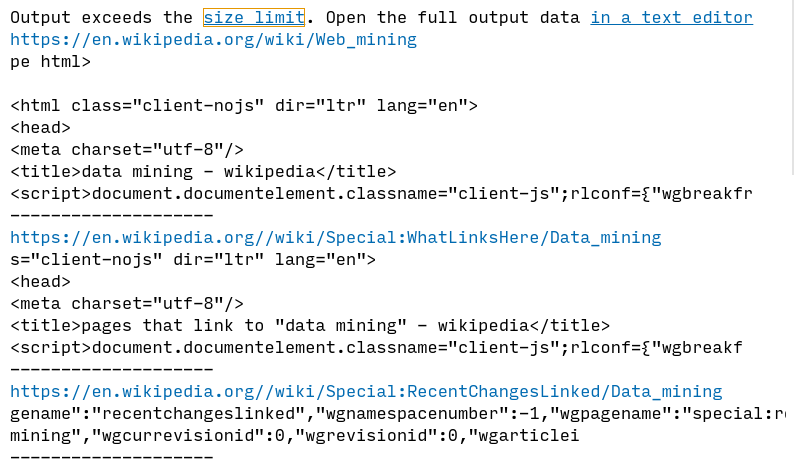
if i!=-1:

print(newPage)

print(s[max(0,i-100):min(len(s)-1,i+100)])

print('-'\*20)

output:





**Ex.2**

Write a web crawler program which takes as input a url, a search word and maximum number of pages to be searched and returns as output all the web pages it searched till it found the search word on a web page or return failure.

**Ex.3**

Implement breadth-first-search and depth-first-search crawlers for Ex. 2.

**In Exp 2 either I have to implement breadth first search or depth first search so I will merge both both Ex into one**

**Depth First search**

Code:

**from urllib.request import urlopen**

**from bs4 import BeautifulSoup**

**import re**

def depthFirstSearch(url):

global pages, searchWord, totalPages

if url in pages:

return

pages.add(url)

totalPages-=1

print(url)

html = urlopen(url.format(url))

bs = BeautifulSoup(html, 'html.parser')

s=str(bs).lower()

i=s.find(searchWord.lower())

if i!=-1:

print("found",searchWord,"in",url)

print(s[max(0,i-100):min(len(s)-1,i+100)])

print('-'\*20)

totalPages=-1

return

for link in bs.find\_all('a', href=re.compile('^(/wiki/)')):

if totalPages==-1:

return

if totalPages==0:

print("searching terminated due to max pages reached")

totalPages-=1

return

if 'href' in link.attrs:

if link.attrs['href'] not in pages:

newPage = link.attrs['href']

newPage="https://en.wikipedia.org/"+newPage

depthFirstSearch(newPage)

pages = set()

searchWord=input("Enter the word to search: ")

inputUrl=input("Enter the url to start with: ")

totalPages=int(input("Enter the number of pages to search: "))

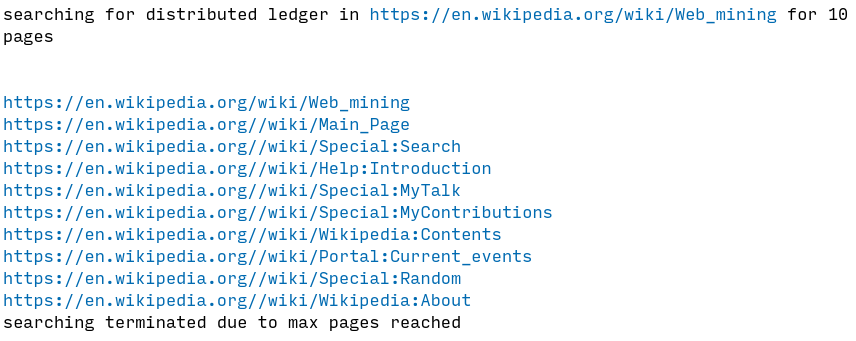
print("searching for",searchWord,"in",inputUrl,"for",totalPages,"pages\n\n")

depthFirstSearch(inputUrl)

output:



**but depth first search may get lost in deep web going out of context**

****

**Breadth First search**

**Code:**

from urllib.request import urlopen

from bs4 import BeautifulSoup

import re

from collections import deque

q = deque()

pages = set()

searchWord=input("Enter the word to search: ")

inputUrl=input("Enter the url to start with: ")

totalPages=int(input("Enter the number of pages to search: "))

print("searching for",searchWord,"in",inputUrl,"for",totalPages,"pages\n\n")

q.append(inputUrl)

while totalPages>0:

totalPages-=1

url = q.popleft()

if url in pages:

continue

pages.add(url)

html = urlopen(url.format(url))

bs = BeautifulSoup(html, 'html.parser')

s=str(bs).lower()

i=s.find(searchWord.lower())

if i!=-1:

print("found",searchWord,"in",url)

print(s[max(0,i-100):min(len(s)-1,i+100)])

print('-'\*20)

break

for link in bs.find\_all('a', href=re.compile('^(/wiki/)')):

if 'href' in link.attrs:

if link.attrs['href'] not in pages:

newPage = link.attrs['href']

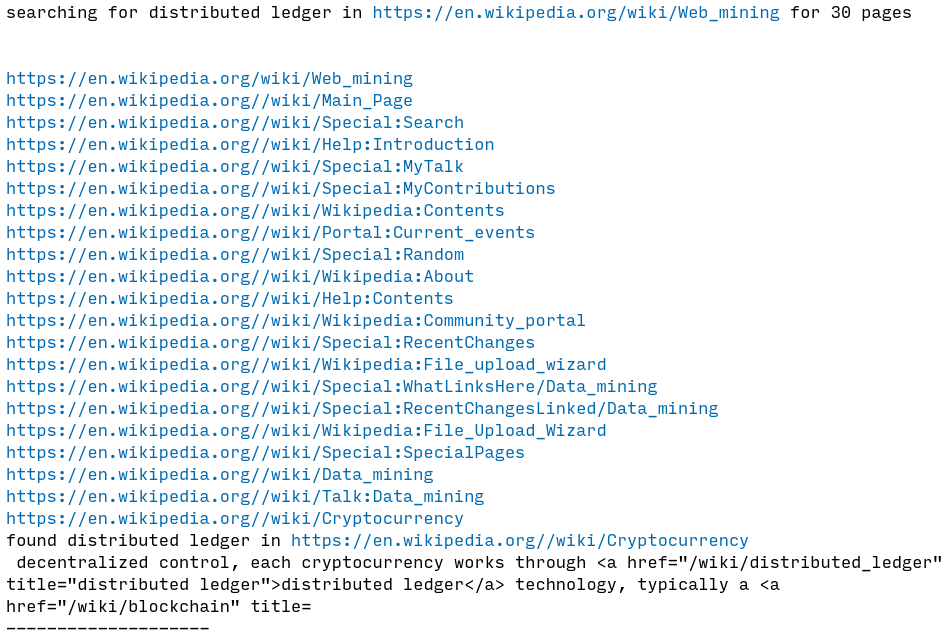
newPage="https://en.wikipedia.org/"+newPage

q.append(newPage)

if totalPages==0:

print("searching terminated due to max pages reached")

**output:**

****