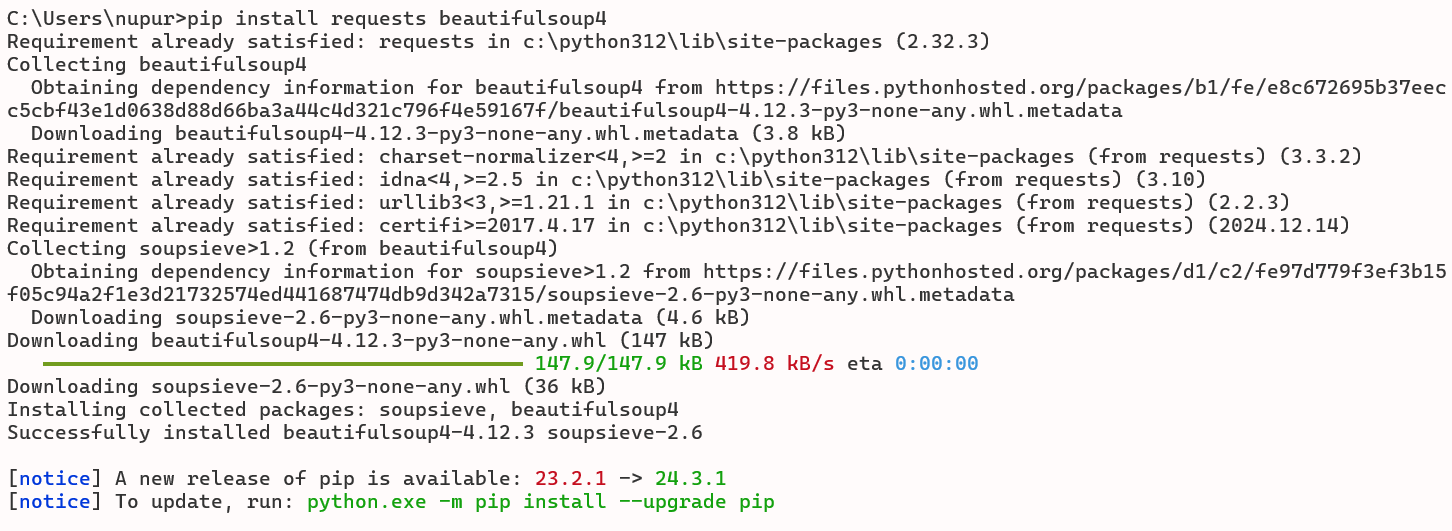
**PRACTICAL NO 7**

**AIM :** Web Crawling and Indexing

* Develop a web crawler to fetch and index web pages.
* Handle challenges such as robots.txt, dynamic content, and crawling delays.

**SOLUTION:**

Install the following modules using pip: pip install requests beautifulsoap4



**INPUT:**

import requests

from bs4 import BeautifulSoup

import time

from urllib.parse import urljoin, urlparse

from urllib.robotparser import RobotFileParser

def get\_html(url):

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

try:

response=requests.get(url,headers=headers)

response.raise\_for\_status()

return response.text

except requests.exceptions.HTTPError as errh:

print(f"HTTP Error:{errh}")

except requests.exceptions.RequestException as err:

print(f"Request Error:{err}")

return None

def save\_robots\_txt(url):

try:

robots\_url=urljoin(url,'/robots.txt')

robots\_content= get\_html(robots\_url)

if robots\_content:

with open('robots.txt','wb') as file:

file.write(robots\_content.encode('utf-8-sig'))

except Exception as e:

print(f"Error saving robots.txt: {e}")

def load\_robots\_txt():

try:

with open('robots.txt', 'rb') as file:

return file.read().decode('utf-8-sig')

except FileNotFoundError:

return None

def extract\_links(html,base\_url):

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

links=[]

for link in soup.find\_all('a',href=True):

absolute\_url=urljoin(base\_url,link['href'])

links.append(absolute\_url)

return links

def is\_allowed\_by\_robots(url,robots\_content):

parser=RobotFileParser()

parser.parse(robots\_content.split('\n'))

return parser.can\_fetch('\*',url)

def crawl(start\_url,max\_depth=3, delay=1):

print("Hello")

visited\_urls=set()

def recursive\_crawl(url,depth, robots\_content):

if depth>max\_depth or url in visited\_urls or not is\_allowed\_by\_robots(url,robots\_content):

return

visited\_urls.add(url)

time.sleep(delay)

html=get\_html(url)

if html:

print(f"Crawling {url}")

links=extract\_links(html,url)

for link in links:

recursive\_crawl(link,depth+1,robots\_content)

save\_robots\_txt(start\_url)

robots\_content=load\_robots\_txt()

if not robots\_content:

print("Unable to retrieve robots.txt. Crawling without restrictions.")

else:

print("Using robots.txt for crawling restrictions.")

recursive\_crawl(start\_url,1,robots\_content)

print("Performed by Nupur Karpe")

crawl('https://', max\_depth=2, delay=2)

**OUTPUT:**



Robot.txt file

