

## **“Website scrapping using request library and beautiful Soup for extracting data of malicious website”**

**Ms. Shreya Thakur**

MSc. Cyber Forensics

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore, Madhya Pradesh, India

**Ms. Shilpa Jossy**

Assistant Professor, Department of Forensic Science

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore, Madhya Pradesh, India

### **Abstract**

The main objective of this research is to leverage Python modules like Requests and BeautifulSoup for web scraping, which allows data extraction from malicious websites. The Requests package makes use of HTTP requests to make it easier to retrieve web pages, while BeautifulSoup is used to parse and browse HTML content so that important information can be extracted quickly. The process entails actions like locating potentially harmful URLs, examining webpage components, and extracting pertinent information including URLs, IP addresses, and possibly malicious scripts. As part of the study, the extracted data will also be stored in an organized format for further examination in the field of digital forensics.

The project shows how online scraping may be used for cybersecurity, giving analysts and researchers important information about malware distribution channels, phishing URLs, and possible dangers. The outcomes highlight how useful these tools are for automating data collecting, which can improve threat intelligence and help identify cyberthreats early on. The ethical and legal aspects of online scraping are emphasized as being dependent on legal concerns, particularly in delicate situations.

To sum up, integrating BeautifulSoup with the Requests library provides a useful method for obtaining useful information from harmful websites, which can help cybersecurity experts reduce risks. To strengthen cybersecurity defenses even further, future study may examine vulnerability scanning of the gathered data using programs like Nessus.

### **Keywords**

Web Scraping, Malicious Websites, Request Library, BeautifulSoup, Cybersecurity, Digital Forensics, Data Extraction, Threat Intelligence, HTML Parsing, Vulnerability Scanning.

### **Introduction**

Web scraping is a technique that is used to extract data from website. In today's digital world, huge amounts of information is available on internet, ranging from product details, financial data to articles and social media posts. Web scraping allows us to access and collect this data in structured form, which can then be analyse, processed, and utilized for different purposes.

At its core, web scraping involves retrieving web pages, parsing their content, and extracting relevant information. This process can be performed manually but is often automated using programming languages and specialized tools. Python, has a unique and rich ecosystem of libraries like BeautifulSoup and Requests, that is famous choice for web scraping due to its simplicity and flexibility.

Web scraping finds applications across numerous industries and domains. It enables businesses to gather competitive intelligence, researchers to collect data for academic studies, marketers to monitor social media trends, and much more. By automating the data collection process, web scraping streamlines workflows, saves

time, and provides access to valuable insights.

Also it is important to conduct web scraping responsibly and ethically. Websites also have terms of services that prohibit or restrict scraping activities, scraping large amounts of data can put strain on server resources and impact website performance. Hence, it is essential to be aware of legal and ethical considerations.

In this experimental study, we will explore the fundamentals and depth knowledge of web scraping, including common techniques, best practices, and practical examples.

### **Types of Website Scraping:**

Website scraping is a technique that is used to extract data from websites. There are various methods and approaches to perform web scraping, depending on the complexity and the structure of the website being scraped. Some common types of web scraping:

**HTML(Hyper Text Markup Language) Parsing:** This technique is used for parsing the HTML structure of a webpage to extract desired data using libraries like BeautifulSoup in Python.

**API Scraping:** Some websites offer APIs (Application Programming Interfaces) which allows developers to access data in structured form. API scraping involves querying these APIs to retrieve data. **Dynamic Content Scraping:** Websites that load content using JavaScript require a specialized

technique such as using headless browsers (e.g., Selenium) to render the page and then scrape the content.

**XPath Scraping:** a query language that is selecting nodes from an (XML) document. It can also be used to extract specific elements or data from HTML documents.

**Regular Expression (Regex) Scraping:** Regex is used to find patterns in text data, for extracting specific information from web pages.

**RSS Scraping:** Some websites provide RSS feeds that contain regularly updated content. RSS scraping involves parsing these feeds to extract data.

**Scraping with Web Scraping Frameworks:** There are several web scraping frameworks and libraries, such as Scrapy in Python, which provide high-level abstractions for building web scrapers.

**Data Extraction from Tables:** Many websites present data in tabular format. Scraping data from tables involves identifying and extracting relevant information from these tables.

**Image Scraping:** This involves downloading images from websites, either directly from image URLs or by extracting image links from HTML.

**Social Media Scraping:** Scraping data from social media platforms involves accessing APIs or using web scraping techniques to extract information such as posts, comments, and user profiles.

**Structured Data Scraping:** Some websites markup their content with structured data formats like Microdata, RDFa, or Schema.org. Scraping structured data involves parsing these markup formats to extract information.

**Web Archive Scraping:** Web Archive store snapshots of web pages. Scraping web archives can be useful for historical data analysis.

There are few examples, such as the website's structure, the type of data being extracted, legal considerations and copyright. It is important to review the policies and terms of a website before scraping, to ensure compliance with legal and ethical standards.

### **Uses of Website Scraping: -**

Website scraping serves a variety of purposes across different industries and domains. Here are some common use cases:

**Market Research and Competitive Analysis:** Companies can work on web scraping to gather data about their competitors, product, pricing information, customer reviews, and marketing strategies. This information helps organizations to stay competitive in the market.

**Lead Generation:** Web scraping is also used to extract contact information, such as contact numbers and Email address, from websites. This data can then be used for lead generation and targeted marketing campaigns.

**Content Aggregation:** News aggregators and content websites use web scraping to gather data such as articles and blog posts from various sources. This allows to provide users with a centralized platform for accessing relevant information.

**Financial Data Analysis:** Finance professionals use web scraping to collect financial data from websites such as stock, financial news portals, and investment forums. This data can be used for market analysis, investment research, and decision-making.

**Real Estate Analysis:** Web scraping can be used to gather data about properties listed for sale or rent on real estate websites.

**Job Market Analysis:** Job portals and recruitment agencies use web scraping to gather data about job listings, such as job titles, descriptions, salaries, and company information. This data helps job seekers find relevant opportunities and recruiters identify trends in the job market.

**Academic Research:** Researchers use website scraping to gather data for academic studies and research projects. This includes gathering data from scholarly articles, research publications, and online databases to analyze trends, conduct literature reviews, and generate insights.

**Weather Data Collection:** Meteorologists and weather forecasting agencies use web scraping to gather weather data from government websites, weather stations, and meteorological services. This data is essential for forecasting weather patterns and predicting natural disasters.

**Healthcare Analytics:** Healthcare professionals and researchers use web scraping to collect data from medical websites, research publications, and healthcare databases. This data can be used for epidemiological studies, disease surveillance, and healthcare analytics.

These are few examples of how website scraping is working across different industries and disciplines. Its versatility and scalability makes it a valuable tool.

### **Python: -**

Python is a versatile programming language known for its simplicity and flexibility. This language is invented by Guido van Rossum in 1991, Python has grown into one of the most popular high-level programming language.

It is a powerful language that can be used for web scraping due to its ecosystem of libraries and tools. Here are some popular Python libraries and frameworks used for web scraping:

**Beautiful Soup:** This Python library is used for parsing HTML and XML documents. It also provides a simple interface for navigating and manipulating the parse tree, making it ideal for extracting data from website pages.

### **Example:**

```
from bs4 import BeautifulSoup
```

```
#first make a :
```

```
import requests
```

```
#Add the URL of the website url = 'https://example.com' response = requests.get(URL)
```

```
# then Parse the HTML content
```

```
soup = BeautifulSoup(response.Text, 'html.Parser') # Find elements by tag, class, id,
```

```
title = soup.find('title').text
```

**Requests**: this is widely used for making (HTTP requests) in Python, which is often the first step in web scraping.

Example:

```
import requests
```

```
# Make a GET request to the webpage and then enter the URL of the website url = 'https://svvv.in'
```

```
response = requests.get (url)
```

```
# Access the response content and print the response print(response.text)
```

```
#After this the HTTP response code of the site will be generated
```

### **REASONS WHY PYTHON IS WELL SUITED FOR WEB SCRAPING:**

**Simplified syntax**: this is a python clean and readable syntax that makes it easy to write and understand web scraping. With no need for semicolons or curly braces, Python code is more concise and expressive, enhancing the development experience.

**Rich Library Ecosystem**: Python boasts a vast collection of libraries like NumPy, Pandas, and Matplotlib, which not only facilitate web scraping but also enable further manipulation and analysis of the extracted data. This extensive library ecosystem simplifies various aspects of the scraping process and enhances its capabilities.

**Dynamic Typing**: Python's typing system eliminates the need to explicitly declare variables. This flexibility streamlines the development process and allows developers to focus more on solving problems rather than managing data types.

**Human-Readable Syntax**: Python's syntax is designed to resemble natural language, making it intuitive and easy to learn for both beginners and experienced developers. This readability enhances collaboration and understanding among team members working on web scraping projects.

**Concise Code**: Python's simplicity allows developers to achieve complex tasks with minimal code. This feature enables the creation of efficient and concise web scraping scripts, saving time and effort in both development and maintenance.

**Active Community Support**: Python has one of the largest and most vibrant developer communities. This provides resources, making easy for developers to find out solutions to their web scraping.

These factors collectively contribute to Python's popularity and effectiveness in web scraping, making it the

preferred choice for many developers and organizations seeking to extract data from the websites.

### **Importance of Web Scraping in cyber forensics: -**

Web scraping plays a significant role in, the process of gathering, analyzing the digital evidence for investigative purposes. Here are some key ways in which web scraping is important in cyber forensics:

**Data Collection:** Web scraping enables cyber forensic investigators to collect digital evidence from various sources, like websites, social media, forums, and online marketplaces. This extracted data includes text, images, videos, metadata, and other digital artifacts relevant to an investigation.

**Evidence Preservation:** Web scraping allows investigators to capture web content and preserve it as evidence in its original state. This is crucial for maintaining the integrity and authenticity of digital evidence, as websites and online content can change or be deleted over time.

**Investigative Leads:** Web scraping can uncover valuable leads and clues related to cybercrime investigations. By gathering information from online sources, investigators can identify suspects, uncover criminal activities, and track digital footprints left behind by perpetrators.

**Monitoring Illegal Activities:** Web scraping can be used to monitor illicit activities on the dark web, such as the sale of stolen data, illegal goods, and services, or discussions related to cyber-attacks and hacking techniques. This proactive approach allows law enforcement agencies and cybersecurity professionals to identify and respond to cyber threats more effectively.

**Digital Forensic Analysis:** Web scraping provides forensic analysts with access to a wealth of digital evidence that can be analyzed to reconstruct digital crime scenes, identify perpetrators, and establish a timeline of events. This analysis may involve examining website content, extracting metadata from digital files, and correlating information from multiple sources to build a comprehensive picture of a cyber incident.

**Evidence Presentation:** Web scraping tools and techniques enable investigators to organize and present digital evidence in a structured and comprehensible manner. This includes generating reports, visualizing data, and presenting findings in court proceedings or other legal settings.

**Proactive Cybersecurity:** Beyond reactive investigations, web scraping is used for proactive cybersecurity measures. By monitoring online forums, social media platforms, and hacker communities, organizations can identify potential threats, vulnerabilities, and emerging attack trends, allowing them to strengthen their defenses and mitigate risks proactively.

In summary, web scraper is a tool in cyber forensics, providing investigators with the aim to collect, analyze, preserve, and present digital evidence from online sources. With the scalability of web scraping, cyber forensic professionals would enhance their investigative capabilities and contribute to the detection and prevention of cybercrime.

## **REVIEW OF LITERATURE**

### **1. A Comprehensive and comparative study on website scrapping**

Authors: SCM de S Siri Suriya Year of publication:2015

The World Wide Web hosts a vast array of information across diverse domains such as social, financial, security, and academic sectors. Accessing this information for educational purposes often proves challenging due to its varied formats and interfaces. Web scraping emerges as a solution to this challenge, aiming to



transform unstructured web data into structured formats suitable for storage and analysis in local databases or spreadsheets. Various techniques exist, including methods like copy-paste, text grabbing, and expression matching, as well as more advanced approaches such as (HTTP) programming, (HTML) parsing, (DOM) parsing, and the utilization of the scraping software.

Among these techniques, web scraping software stands out as the most accessible option, particularly for those lacking technical expertise. An abundance of web scraping software tools is available today, with many developed using Java, Python, and Ruby. These tools range from open-source solutions to commercial products, with examples including Yahoo Pipes, Google Web Scrapers, and Outwit Firefox extensions, which are particularly well-suited for beginners.

This study's main aim is to provide a comprehensive and comparative analysis of various web scraping techniques and popular web scraping software. Through this comparison, we elucidate the strengths and weaknesses of particular approaches, offering insights into their applicability for extracting data from educational websites. Ultimately, this review serves as a valuable resource for individuals seeking to navigate the complexities of web scraping for educational purposes.

## **2. WebSelf {A website scraping framework}**

Authors: Jakob Thomsen, Michael Claus Brabrand and Erik Ernst Year of Publication: 2015

This study introduces Web Self, a pioneering framework devised for website scraping, which determines the process by decomposing it into independent and reusable constituents. Through implementation, the validity and adaptability of the framework are demonstrated, showcasing its ability to encapsulate prior methodologies prevalent in the realm of web scraping. To assess the efficacy of this approach, an extensive experiment was conducted, spanning daily iterations of 17 websites and encompassing approximately 11,000 HTML pages over a period exceeding one year.

The experiment involved a comparative analysis of various web scraping constituents, including both established techniques and innovative combinations thereof. By addressing three prominent challenges inherent in contemporary web scraping practices, the framework's potential for enhancing accuracy, precision, and specificity was evaluated. The experimental findings underscored the efficacy of integrating previous methodologies with novel techniques, highlighting the synergistic benefits of compositional approaches in optimizing web scraping processes. This study contributes to the discourse on web scraping methodologies, offering insights into the advancements and challenges within this domain.

## **3. Comparison of different Python libraries for website data extraction**

Authors: Edenic Uzum, Tarik Yerlikaya and Oguz Kirat Year of publication: 2018

Within the Python ecosystem, several libraries are dedicated to extracting valuable data from web pages. This research explores a comparative assessment of three prominent extraction libraries: BeautifulSoup, lxml, and regex, aiming to provide insights into their respective performance and challenges.

Experimental findings reveal regex as the top-performing library, exhibiting an impressive average processing time of 0.071 ms. However, a notable challenge arises when generating accurate extraction rules for regex, particularly in scenarios where the number of inner elements is unknown. Empirical evidence indicates that only 43.5% of extraction rules are deemed suitable for such tasks in our experiments.

In situations where regex falls short, DOM-based libraries like BeautifulSoup and lxml come into play. Among these, lxml emerges as the most efficient option, demonstrating an average processing time of 9.074 ms.

This study contributes to the existing body of knowledge by offering insights into the comparative performance and challenges associated with web data extraction libraries in the Python ecosystem. By synthesizing empirical findings, this research aims to inform practitioners and researchers about the strengths and limitations of each approach, facilitating informed decision-making in the selection of extraction techniques.

#### **4. Web scraping extraction on different websites**

Authors: Vojtech Draxl Year of publication: 2018

Web extraction or harvesting, is a technique used to extract data from WWW. The data is extracted utilizing (HTTP) or web browser.

Web scraping process may involve query the source, retrieving results and parsing the page to gain appropriate results.

This paper covers all the techniques and tools used in the history of website scraping including various tools like programming libraries, cloud software, Browser Extensions, and desktop software.

#### **5. An overview on web scraping techniques and tools** Authors: Anand V. Saurkar, Kedar G.

Pathare, Shweta. A Gode Year of publication: 2018

The evolution of the World Wide Web has precipitated significant transformations in internet usage patterns and data exchange modalities.

Consequently, a surge in internet users has ensued, resulting in an exponential proliferation of available data. Across diverse domains, including business, academia, and research, the internet serves as a primary platform for expeditious dissemination of advertisements and information. In response to this challenge, researchers have introduced an indispensable technique known as Web Scraping. A spectrum of techniques has been deployed for web scraping, encompassing conventional methods such as copy-and-paste, text grabbing, and regular expression matching, as well as more sophisticated approaches like HTTP programming, HTML parsing, DOM parsing, and utilization of web scraping software.

This study represents array of methodologies employed in web scraping and highlights recent advancements and tools utilized in this domain. By elucidating the evolution and progression of web scraping techniques.

#### **6. Comparison of web scraping techniques**

Authors: Rohmat Gunawan, Alam Rahmatulloh, Irfan Darmawan, Firman Firdaus. Year of publication: 2018

The data extraction from the sites on internet is commonly known as website scrapping.

This research focus on the comparison of three different methods of web scrapping that is REGEX regular expression, DOM, HTML, and XPATH on the parameters as data consumption, process time and memory usage. Where the outcome of comparison shows that all the three methods have different parameters for example regex is the smallest method used in memory usage, HTML and DOM consumes less time than regex and XPath method and theses methods are used to process web scrapping by searching (HTML) code from a targeted website.

#### **7. News articles scraping and analysis** Authors: Shashwat Pande, Noor Shaikh, Bindu Garg Year of

Publication: 2020

This study determines its aim by performing sentiment analysis done on scraped news article.

The scraping process is done by using set of modules and libraries that will be unique to python language. The process starts by inspecting a news article website to analysis the tags and attributes after which parsing will be done ending with the extraction as the last step to gather

the results. This research mainly focused on the subjectivity that will determine how much terms are related to the data scrapped from the links and can help analysis the sentiments of the news article. Whereas this study creates a future aspect to define data and its new trends.

## **8. Development of web scraper**

Author: Pontus Andersson Year of Publication: 2021

This study examines the concept of scraping the website by developing the website scraper by using programming language like python to collect unstructured data and saved it in the form of structured data. The main goal of this study was to develop a python-based web scraper which can collect the extracted text from the websites (ex-time edit) and can convert unstructured data into structured form where users can get the data in fast and efficient manner.

This research paper provides sufficient ideas and methodology to use web scraper based on python language with all the step-by-step procedure and also shows the potential upside of web scrapping.

## **9. Web scraping techniques and application**

Authors: Chaimaa Lofti, Swetha Srinivasam, Myriam Ertz , Imen Latrous Year of publication: 2021

Web Scraping represents a fundamental approach in extracting the large amount of data from various sources like social media, online portals, websites etc.

This paper aims to describe the most advanced web scraping techniques to better equip scholars with helpful knowledge on how to mine online data. It includes all the detailed methods to start the basic design of web scraper and applications of web scrapping by using various tools.

## **10. Website scraping approaches and performance on modern websites** Authors: Ajay

Sudhir Bale, Rohith S , Rohith R,S Kamlesh, Naveen Gorpade and Rohan B Year of publication:2022

The study delves into various studies and research findings concerning web scraping.

It begins with the Bot Activity report, which underscores the substantial presence of bots in internet traffic. This paper serves as a backdrop, emphasizing the significance of investigating web scraping methodologies and their implications for website security.

The main objective is to encompass efficacy of diverse web scraping methods, discerning websites responses to bot interactions, and gauging the level of resilience against web Scraping endeavours. Also, it includes categorization of websites into different sectors for testing and scrutinizing the ramifications of web scraping across various domains.

## **11. Comparative analysis of dynamic web scraping techniques**



Authors: Kaajal Sharma, Gautam M Borkar Year of publication: 2022

This technical paper delves into the intricacies of web scraping, with the focus on handling dynamic web pages. It underscores the critical role of library selection in ensuring efficient data extraction.

The paper basically evolves the performance of prominent libraries as beautiful soup, LXML and Selenium .It also focuses on insights of effective web scraping solutions that can be in future become an indispensable asset for both researchers and practitioners in the field.

## **12. Benchmarking of Web Scraping data on various type of Industries.**

Authors: Shobha Rani and Deepa Bharadwaj Year of publication: 2022

The internet landscape is characterized by an abundance of data, spanning structured and unstructured formats, generated daily by a myriad of users, entities, and applications. This proliferation of data is a direct consequence of the expansion of the WWW, fundamentally altering the dynamics of data sharing, acquisition, and dissemination.

This study aims to offer a comprehensive examination of the website scraping process, a comparative analysis of various scraping tools and an exploration of industry data. Employing tools such as Octoparse, ParseHub, Fminer, and Mozenda, industry-specific data is extracted from prominent websites such as Justdial and sulekha.com. The extracted data encompasses a range of parameters, including industry names, contact information, geographical localization, categorical classifications (e.g., manufacturers, dealers), sub-categorical distinctions (e.g., cashew manufacturers, chemical manufacturers, steel manufacturers), and associated website links.

Following the extraction phase, rigorous preprocessing and data cleaning protocols ensures the integrity and reliability of the data. Subsequently, industry data procured from each web scraping tool undergoes detailed analysis, culminating in the generation of illustrative reports facilitating comparative assessments between the different web scraping methodologies employed.

## **13. Scraping Website Data for marketing insights**

Authors: Johannes Boegershausen, Andrew T. Stephen ,Hannes Datta and Abhishek Borah Year of publication: 2022

In this research study the researchers focused on scholars who are using website scraping and Application Programming Interfaces (APIs) to accumulate data from the internet.

Authors involve a novel methodological framework with different stages of accumulating web data: single out data sources, designing the data , and extracting the data. The study concludes directions for future research to identify website data sources and describe evolving marketplace.

## **14. Value of website data scraping**

Authors: Gianluca Barbera, Silvia Fernandes, and Luiz Araujo Year Of Publication: 2023

Social networks are necessary to assets products, or ideas to companies and to build loyalty with customers. With the increasing use of social sites or media today Social Media Analytics are more relevant in market's dynamics which requires effective monitoring and understanding.

In this study the use of Web Data Scraper (WDS) tool is optimized to extract the big published data on social media. The approach in this project to use WDS is to gather data from Trip Advisor's website pages. To obtain consultancy, from identification of regions, to help tourism organizations to magnify business and its model.

The WDS helps to detect the unstructured data very quickly without programming and it can be an object of future research to leverage consumer predictions.

## **15. Prevent Web Scrapping**

Authors: Yash Shelar, Ajay Gupta, Asst. Prof. Sonali Patil Year of publication: 2024

This study shows that how cybercriminals use web scrapping for malicious activities such as data selling, price undercutting, reselling of the content etc and tells that how and why the attackers use bots and other methods to gain unauthorized information using web scrapping attacks.

The article also defines content scrapping to make duplicate copies of the targeted sites. And concludes that Preventing web scrapping is a multifaceted challenge that involves a combination of technological, legal, and ethical considerations. The effective measures to prevent web scrapping require the protection of data, user experience, and adherence to legal and ethical standards.

### **RATIONALE**

This study aims at extracting out the data of various malicious websites to safeguard the contents of that website. This work is done on website scraping using the Requests library and BeautifulSoup for extracting data from malicious websites offers a compelling avenue for enhancing cybersecurity defence's, threat intelligence capabilities, and research efforts. By leveraging these techniques, organizations can bolster their resilience against evolving cyber threats and better protect their assets, data, and stakeholders .

### **OBJECTIVE**

To extract out data of malicious website using Python libraries.

### **METHODOLOGY**

- Main tools used in web scrapping

#### **Requests:**

Requests library is a tool used in python for making( HTTP) requests and website API. It is commonly used for sending GET and POST requests, handling cookies, and managing sessions.

#### **Beautiful Soup:**

Beautiful Soup is used for website scraping , primarily parsing HTML . It is used mainly for extracting the data from web pages in a readable and convenient manner.

- Performed with PyCharm community.
- Performed on malicious website.

#### **Steps to be followed while scrapping data from a malicious website**

For extracting the data from a malicious website from python we will follow these basics steps as mentioned :

STEP1: Find the malicious website URL you need to scrap. STEP2: Analyse the page.

STEP3: Find the data you need to extract out from the page.

STEP4: Open the PyCharm community install all the required packages. STEP5: Create a new project.

STEP6: write the program.

STEP7: Run the program and extract the data.

STEP8: Stored the extracted data in the needed format.

### **Example:**

We are going to scrap data from the malicious websites present on internet, for extracting the data .

To scrap the data, we need some prerequisites:

- Python 3.11 (PyCharm Community version)
- Google Chrome Browser
- Python 3.11 installed libraries such as Requests, BeautifulSoup, lxml etc
- Windows Operating System

Step 1: Find the malicious website URL from Google Chrome Browser

Step2: Analyse the page

To analyse the page, click the right on the element and click “Inspect”

The website URL of the page is <https://financereports.co/>

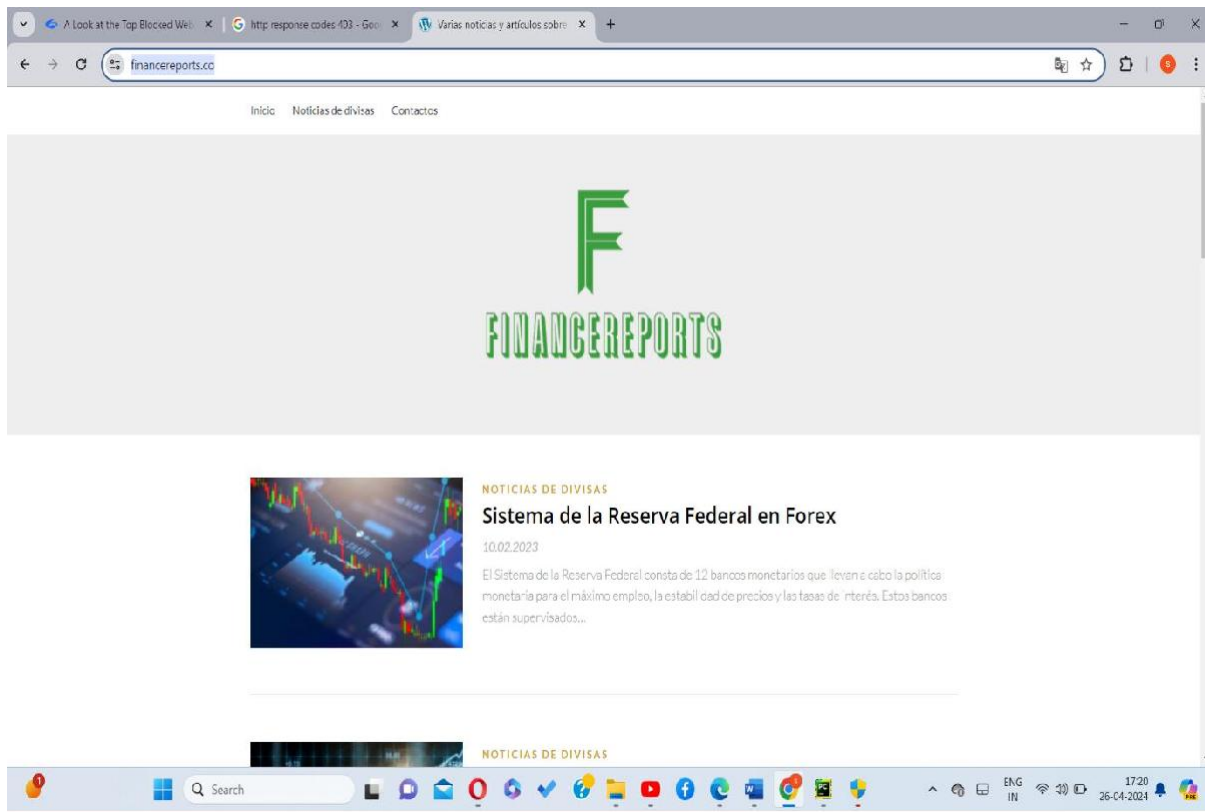


Figure 1: selecting the malicious website

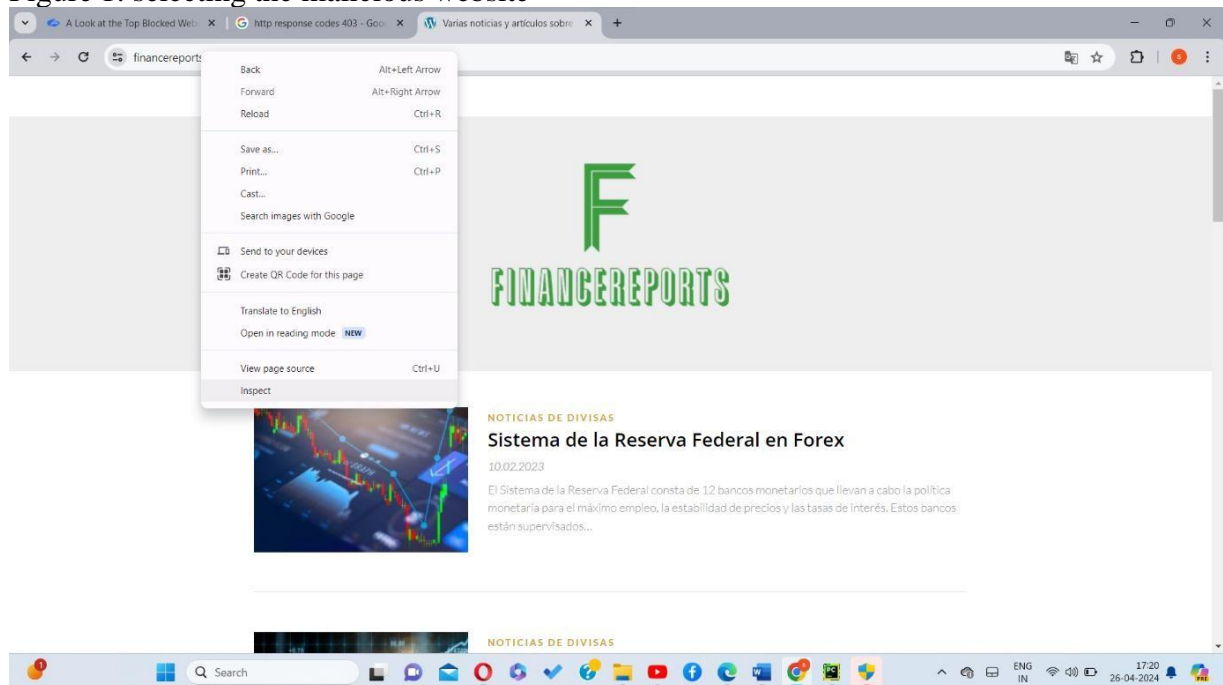


Figure2: Inspecting the page

When you click the inspect tab , a browser Inspector Box will be seen at the right side .



Figure3: Browser Inspect Box

Step 3: Extract the information from the <div> tag from the html code present Step4: Open PyCharm and create new project and install all the required packages.

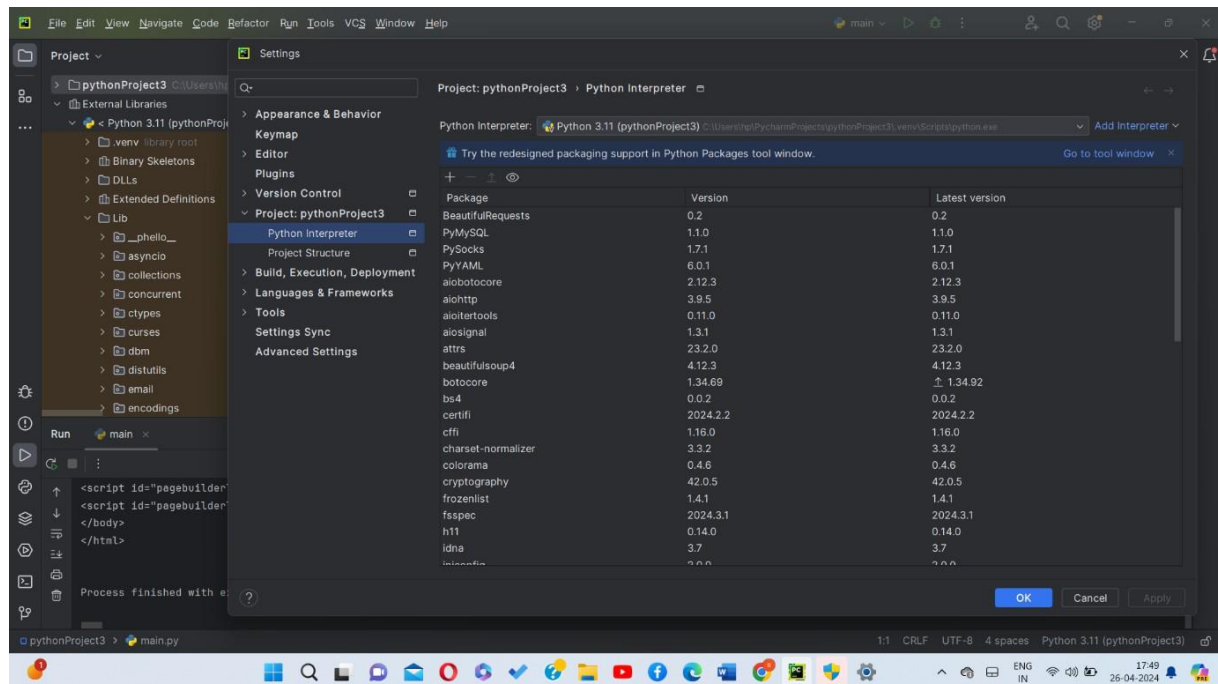
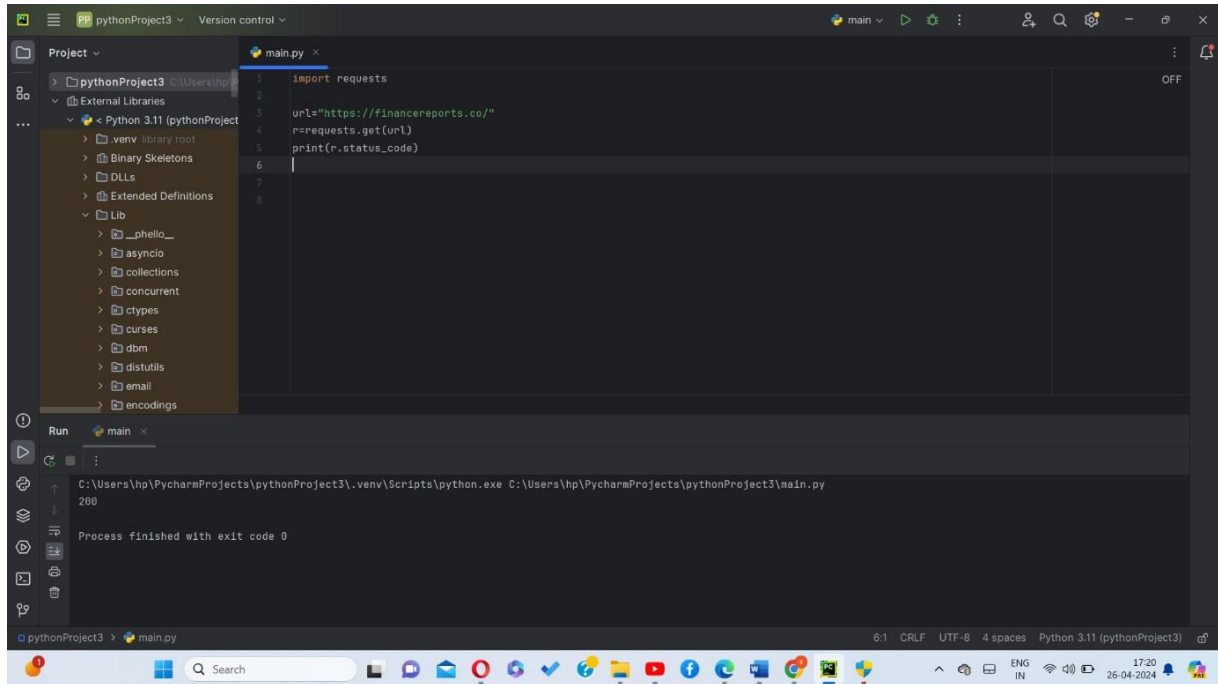




Figure 4: creating new project and installing all the required packages.

Step 5: Write the program to import requests from request library

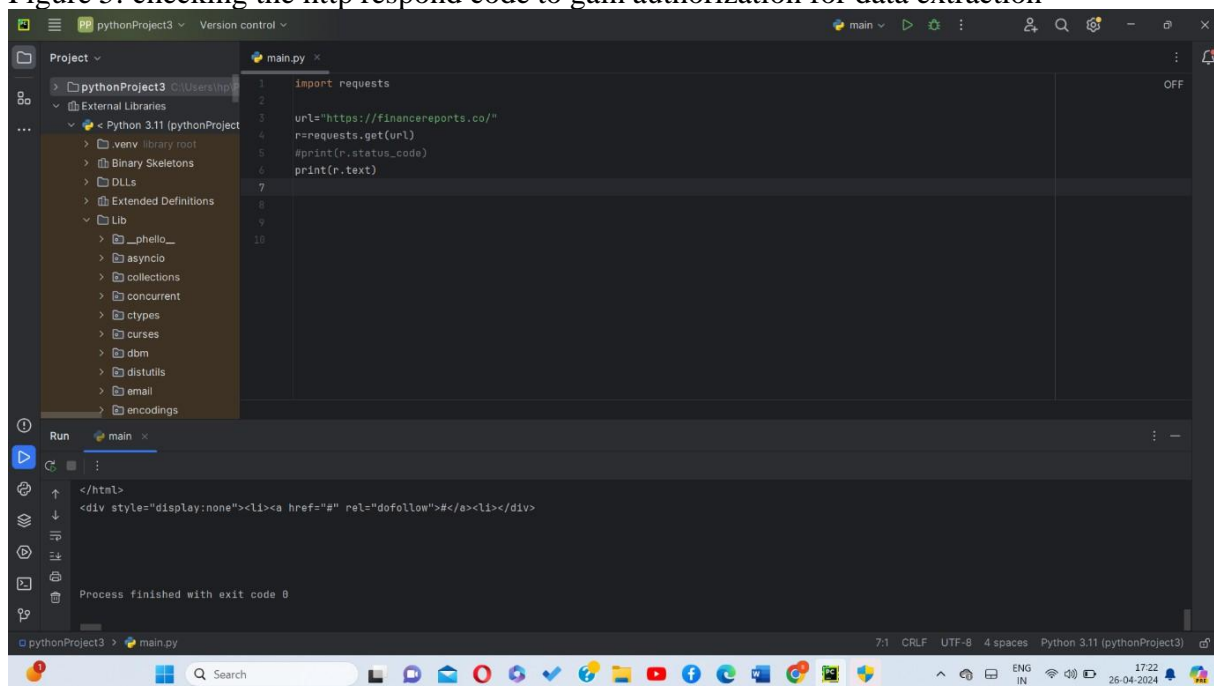


```
1 import requests
2
3 url='https://financereports.co/'
4 r=requests.get(url)
5 print(r.status_code)
6
7
8
```

Run console output: C:\Users\hp\PycharmProjects\pythonProject3\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\pythonProject3\main.py 200

Process finished with exit code 0

Figure 5: checking the http respond code to gain authorization for data extraction



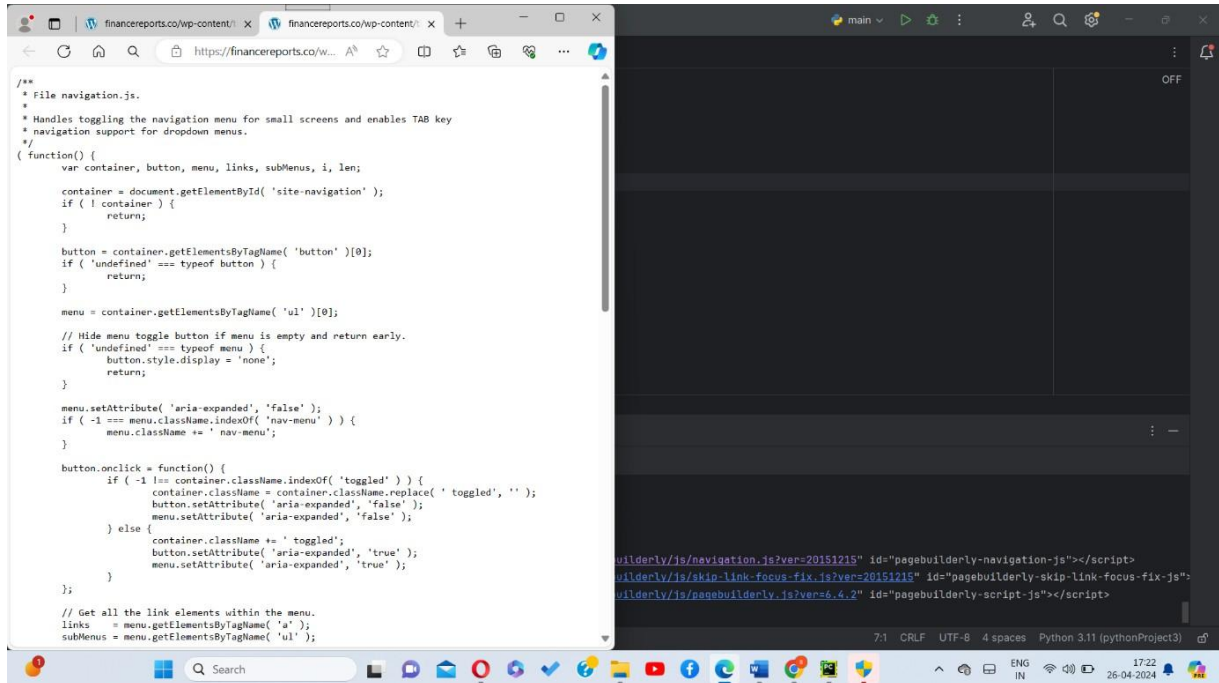
```
1 import requests
2
3 url='https://financereports.co/'
4 r=requests.get(url)
5 #print(r.status_code)
6 print(r.text)
7
8
9
10
```

Run console output: </html> <div style="display:none"><li><a href="#" rel="dofollow">#</a></li></div>

Process finished with exit code 0

Figure 6: Console the program

When the program is run we get the desired html code of the page ,where we can select the links, tags and attribute to gather the information of the page.



```

/**
 * File navigation.js.
 *
 * Handles toggling the navigation menu for small screens and enables TAB key
 * navigation support for dropdown menus.
 */
(function() {
  var container, button, menu, links, subMenu, i, len;

  container = document.getElementById( 'site-navigation' );
  if ( ! container ) {
    return;
  }

  button = container.getElementsByTagName( 'button' )[0];
  if ( 'undefined' === typeof button ) {
    return;
  }

  menu = container.getElementsByTagName( 'ul' )[0];

  // Hide menu toggle button if menu is empty and return early.
  if ( 'undefined' === typeof menu ) {
    button.style.display = 'none';
    return;
  }

  menu.setAttribute( 'aria-expanded', 'false' );
  if ( -1 === menu.className.indexOf( 'nav-menu' ) ) {
    menu.className += ' nav-menu';
  }

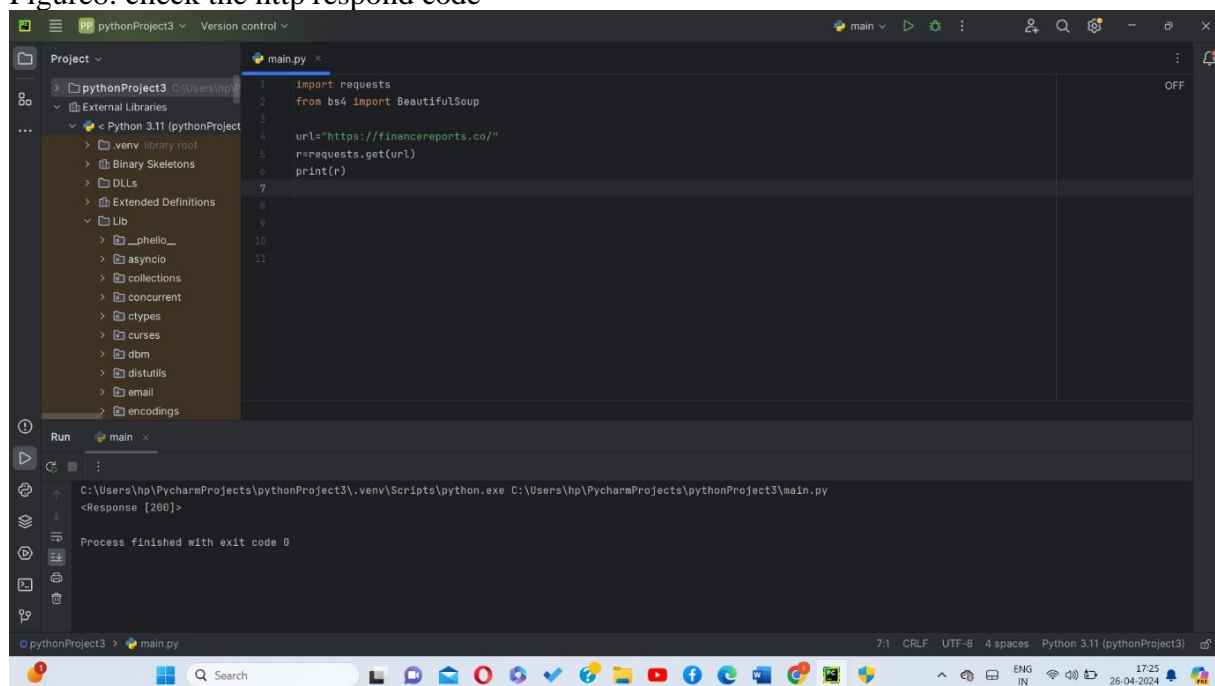
  button.onclick = function() {
    if ( -1 !== container.className.indexOf( 'toggled' ) ) {
      container.className = container.className.replace( ' toggled', '' );
      button.setAttribute( 'aria-expanded', 'false' );
      menu.setAttribute( 'aria-expanded', 'false' );
    } else {
      container.className += ' toggled';
      button.setAttribute( 'aria-expanded', 'true' );
      menu.setAttribute( 'aria-expanded', 'true' );
    }
  };

  // Get all the link elements within the menu.
  links = menu.getElementsByTagName( 'a' );
  subMenu = menu.getElementsByTagName( 'ul' );

```

Figure 7: html code generated through Request library Step 6: Write the program to extract the data from BeautifulSoup

Figure8: check the http respond code



```

1 import requests
2 from bs4 import BeautifulSoup
3
4 url="https://financereports.co/"
5 r=requests.get(url)
6 print(r)
7
8
9
10
11

```

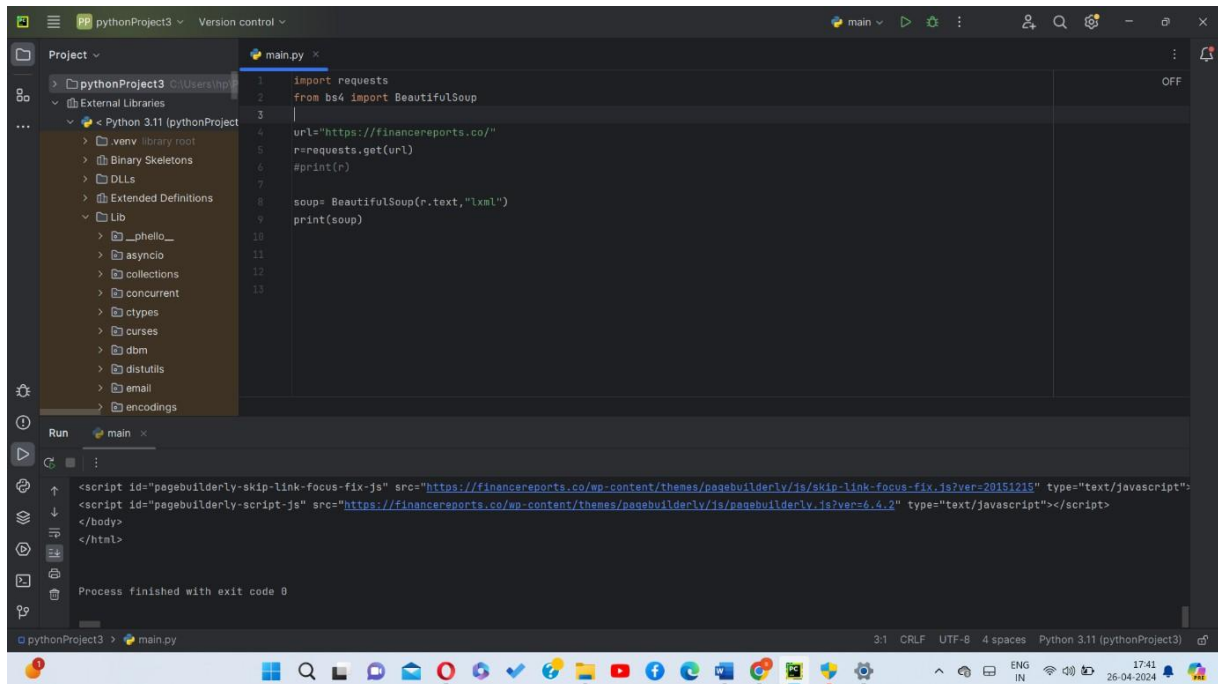
Run main

C:\Users\hp\PycharmProjects\pythonProject3\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\pythonProject3\main.py

<Response [200]>

Process finished with exit code 0

Step7: execute the program



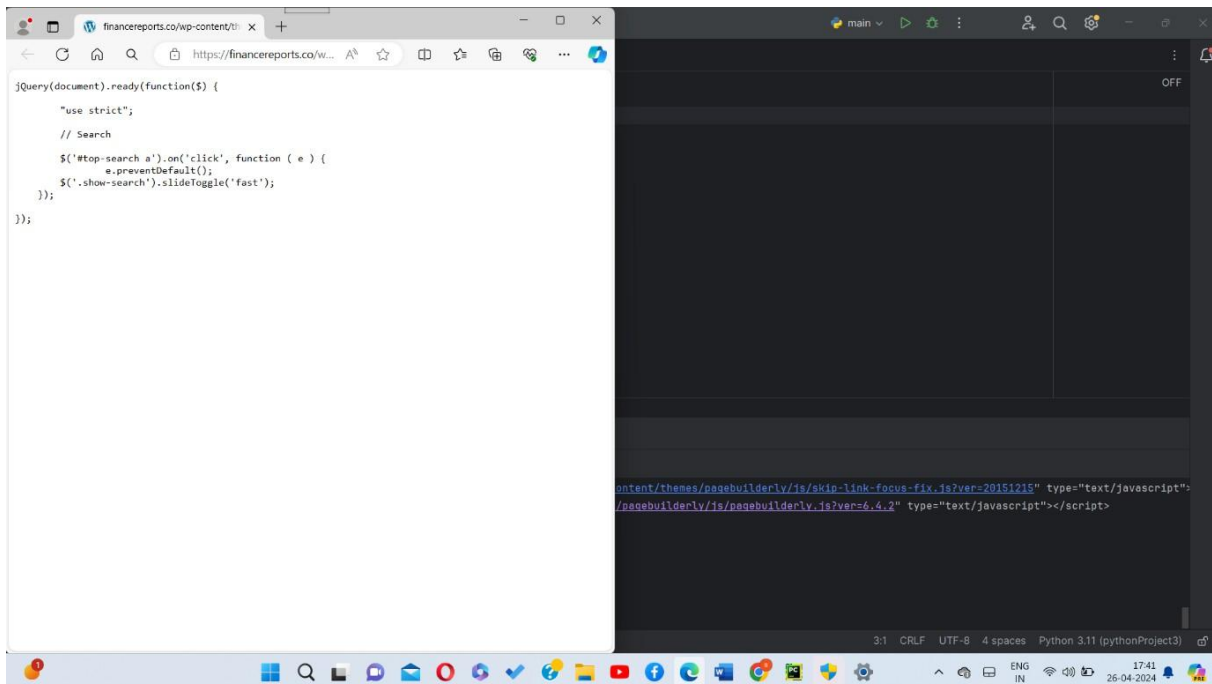
```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url="https://financereports.co/"
5 r=requests.get(url)
6 #print(r)
7
8 soup= BeautifulSoup(r.text,"lxml")
9 print(soup)
10
11
12
13
```

Run console output:

```
<script id="pagebuilderly-skip-link-focus-fix-js" src="https://financereports.co/wp-content/themes/pagebuilderly/is/skip-link-focus-fix.js?ver=20151215" type="text/javascript">
<script id="pagebuilderly-script-js" src="https://financereports.co/wp-content/themes/pagebuilderly/is/pagebuilderly-is?ver=6.4.2" type="text/javascript"></script>
</body>
</html>
```

Figure9:run the program

Step8: After executing the program the html code will be generated from where we can view the extracted data.



```
jQuery(document).ready(function($) {
    "use strict";
    // Search
    $('#top-search a').on('click', function ( e ) {
        e.preventDefault();
        $('#show-search').slideToggle('fast');
    });
});
```

Step 9: After the data is extracted, then is important to look for the html tags and attributes. Let us consider another malicious website <https://virusshare.com/> .

Now to extract the data and to find out data feed within a particular tag or attribute we need to follow these steps:

Inspect the website and select the particular tag or attribute to extract out the hidden data.



Figure11: inspecting the website

Write the code :

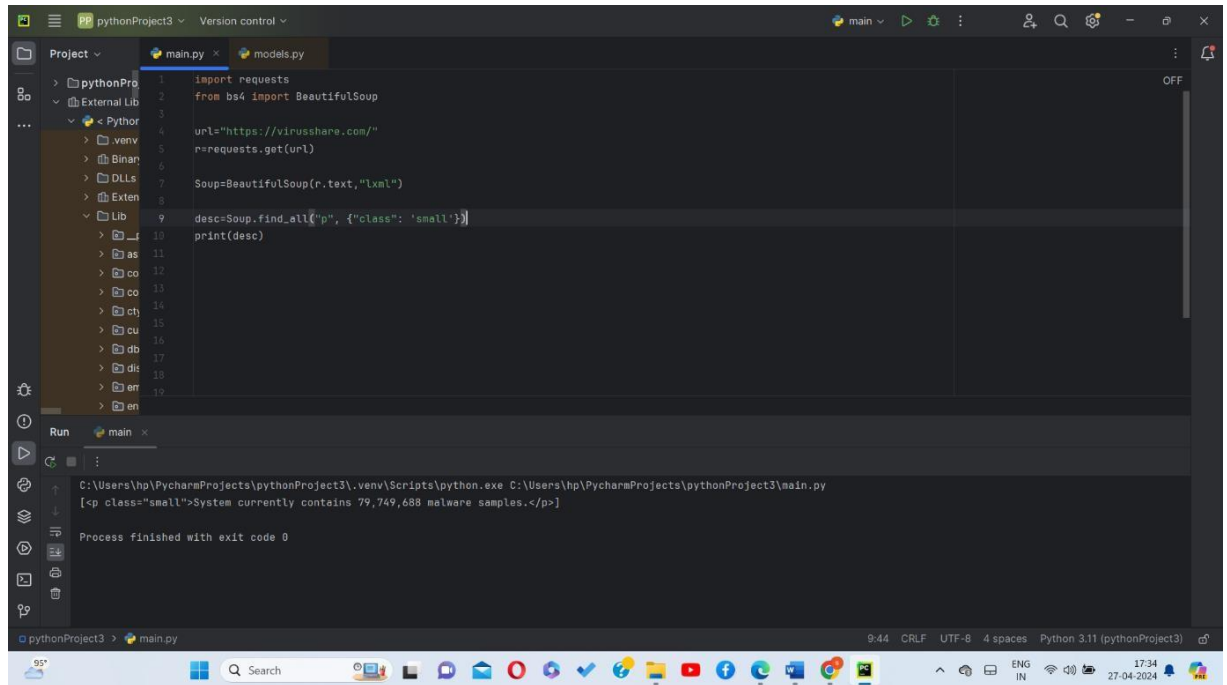


Figure:12 run the program with Soup.find\_all ()

Step10:After extracting out the needed data the final step is to save the extracted data into a required file.

Write the program:

```
import bs4 import requests
```

```
from bs4 import BeautifulSoup
```

```
URL= "https://virusshare.com/" r=requests.get(url)
```

```
soup = BeautifulSoup(r.content, "html. Parser") paragraphs=soup.find_all ('a')
```

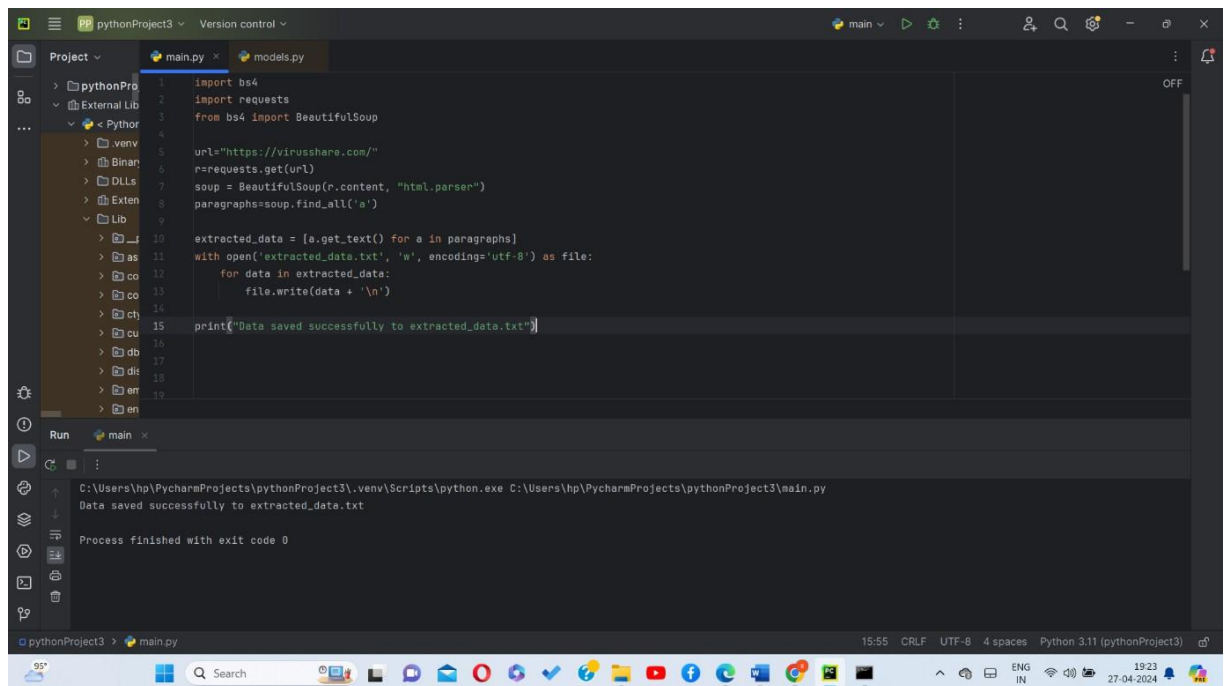
```
extracted data = [a.get_text( ) for a in paragraphs]
```

```
with open('extracted_data.txt', 'w', encoding='utf-8') as file: for data in extracted data:
```

```
file.write(data + '\n')
```

```
print("Data saved successfully to extracted_data.txt") and run it .
```





```
1 import bs4
2 import requests
3 from bs4 import BeautifulSoup
4
5 url="https://virusshare.com/"
6 r=requests.get(url)
7 soup = BeautifulSoup(r.content, "html.parser")
8 paragraphs=soup.find_all('p')
9
10 extracted_data = [a.get_text() for a in paragraphs]
11 with open('extracted_data.txt', 'w', encoding='utf-8') as file:
12     for data in extracted_data:
13         file.write(data + '\n')
14
15 print("Data saved successfully to extracted_data.txt")
```

Run main x

C:\Users\hp\PycharmProjects\pythonProject3\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\pythonProject3\main.py

Data saved successfully to extracted\_data.txt

Process finished with exit code 0

Figure:13 Saving the data into a particular file

As now the extracted data is saved ,it can be retrieved any time whenever needed.

## **RESULT AND DISCUSSION**

In this experimental work, we aimed at extracting out the malicious website data through website scraping using tools like Requests library and BeautifulSoup.This study also aims at preserving the data into its structured form which can be later analyzed and preserved for investigative purposes in digital forensics.

## **CONCLUSION**

In conclusion, leveraging the combination of Request library and parsing tools such as BeautifulSoup for website Scraping offers a potent solution for extracting data from malicious websites. By harnessing the power of these tools, researchers and cybersecurity professionals can efficiently gather critical information about potential threats, such as phishing URLs, malware distribution points, or malicious scripts.

The seamless integration of Requests for fetching HTML content and BeautifulSoup for parsing and navigating through the retrieved data streamlines the extraction process. This approach enables the extraction of various elements from malicious websites, including URLs, domain names, IP addresses, suspicious patterns, and potentially harmful code snippets.

Moreover, by automating the data extraction process through web scraping, analysts can enhance their threat intelligence capabilities, rapidly identifying emerging threats and proactively mitigating risks. This proactive stance is crucial in today's ever-evolving cybersecurity landscape, where early detection and response are paramount.

However, it is essential to emphasize the legal consideration associated with Website Scraping, especially when dealing with potentially sensitive or private data. Attach to the guidelines and website terms, and ensuring regulation that are imperative throughout the web scraping process.

In essence, the combination of request libraries and BeautifulSoup for Website Scraping provides a tool set for extraction of data which is actionable intelligence from the malicious web, empowering cybersecurity

professionals in their ongoing efforts to safeguard digital ecosystems against evolving threats.

The future scope of this experimental study will include the scanning of vulnerabilities of the scrapped data from the tools like Nessus to determine the vulnerabilities present in the websites.

## **REFERENCES**

1. de S Sirisuriya, S. (2015). A Comparative Study on Web Scraping. Retrieved from <https://api.semanticscholar.org/CorpusID:196204953>
2. Thomsen, J. G., Ernst, E., Brabrand, C., & Schwartzbach, M. (2015). WebSelF: A Web Scraping Framework. In Lecture Notes in Computer Science. Lecture Notes in Computer Science (pp. 347–361). doi:10.1007/978-3-642-31753-8\_28 <http://www.informatik.uni-trier.de/~ley/db/conf/icwe/>
3. Uzun, E., Yerlikaya, T., & Kirat, O. (2018). Comparison of Python Libraries used for Web Data Extraction. 24, 87. [https://www.researchgate.net/publication/346659877\\_Comparison\\_of\\_Python\\_Libraries\\_used\\_for\\_Web\\_Data\\_Extraction](https://www.researchgate.net/publication/346659877_Comparison_of_Python_Libraries_used_for_Web_Data_Extraction)
4. Vojtech Draxl.(2018).Web Scraping Data Extration from websites. [https://www.academia.edu/35901535/BACHELOR\\_PAPER\\_Web\\_Scraping\\_Data\\_Extraction\\_from\\_websites](https://www.academia.edu/35901535/BACHELOR_PAPER_Web_Scraping_Data_Extraction_from_websites)
5. Saurkar, A. V., Pathare, K. G., & Gode, S. A. (2018). An Overview On Web Scraping Techniques And Tools. Retrieved from <https://api.semanticscholar.org/CorpusID:198993824>
6. Gunawan, R., Rahmatulloh, A., Darmawan, I., & Firdaus, F. (2019). Comparison of Web Scraping Techniques : Regular Expression, HTML DOM and Xpath. Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering (IcoIESE 2018). Retrieved from <https://api.semanticscholar.org/CorpusID:132839834>
7. Shashwat pande.,Noor Shaikh., Bindu Garg.(2020).News Article And Analysis Using Python(e-ISSN: 2582-5208) IRJMETS [https://www.irjmets.com/uploadedfiles/paper/volume2/issue\\_9\\_september\\_2020/3383/1628083134.pdf](https://www.irjmets.com/uploadedfiles/paper/volume2/issue_9_september_2020/3383/1628083134.pdf)
8. P Andersson (2021). A study on the development of a web scraper for TimeEdit. <https://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-43140>
9. Lotfi, C., Srinivasan, S., Ertz, M., & Latrous, I. (2021). Web Scraping Techniques and Applications: A Literature Review. doi:10.52458/978-93-91842-08-6-38. [https://www.researchgate.net/publication/367719780\\_Web\\_Scraping\\_Techniques\\_and\\_Applications\\_A\\_Literature\\_Review](https://www.researchgate.net/publication/367719780_Web_Scraping_Techniques_and_Applications_A_Literature_Review)
10. Bale, A. S., Ghorpade, N., Rohith, Kamalesh, S., Rohith, & B, R. (2022, August 17). Web scraping approaches and their performance on modern websites. 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC). Presented at the 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India. doi:10.1109/icesc54411.2022.9885689. <https://ieeexplore.ieee.org/document/9885689>
11. Sharma, K., & Borkar, G. (2024). Comparative Analysis of Dynamic Web Scraping Strategies: Evaluating Techniques for Enhanced Data Acquisition. doi:10.56155/978- 81-955020-7-3-22. [https://www.researchgate.net/publication/378162501\\_Comparative\\_Analysis\\_of\\_Dynamic\\_Web\\_Scraping\\_Strategies\\_Evaluating\\_Techniques\\_for\\_Enhanced\\_Data\\_Acquisition](https://www.researchgate.net/publication/378162501_Comparative_Analysis_of_Dynamic_Web_Scraping_Strategies_Evaluating_Techniques_for_Enhanced_Data_Acquisition)
12. Bale, A., Ghorpade, N., Kamalesh, S., S, Rohith, R, Rohith, & S, Rohan. (2022). Web Scraping Approaches and their Performance on Modern Websites. doi:10.1109/ICESC54411.2022.9885689. [https://www.researchgate.net/publication/363669276\\_Web\\_Scraping\\_Approaches\\_and\\_their\\_Performance\\_on\\_Modern\\_Websites](https://www.researchgate.net/publication/363669276_Web_Scraping_Approaches_and_their_Performance_on_Modern_Websites)
13. Boegershausen, J., Datta, H., Borah, A., & Stephen, A. (2022). Fields of Gold: Scraping Web Data

for Marketing Insights. Journal of Marketing, 86, 1–20. doi:10.1177/00222429221100750.

[https://www.researchgate.net/publication/360324664\\_Fields\\_of\\_Gold\\_Scraping\\_Web](https://www.researchgate.net/publication/360324664_Fields_of_Gold_Scraping_Web)

[Data for Marketing Insights](#)

14. Barbera, G., Araujo, L., & Fernandes, S. (2023). The Value of Web Data Scraping: An Application to TripAdvisor. Big Data and Cognitive Computing, 7, 121. doi:10.3390/bdcc7030121

[https://www.researchgate.net/publication/371777832\\_The\\_Value\\_of\\_Web\\_Data\\_Scraping\\_An\\_Application\\_to\\_TripAdvisor](https://www.researchgate.net/publication/371777832_The_Value_of_Web_Data_Scraping_An_Application_to_TripAdvisor)

[ping\\_An\\_Application\\_to\\_TripAdvisor](#)

15. Yash Shelar, Ajay Gupta, Asst.Prof.Sonali patil.(2024).Prevent Web Scrapping.ISSN 2582-7421 www.Ijrpr.com.

<https://www.ijrpr.com/archive.php?volume=5&issue=1>