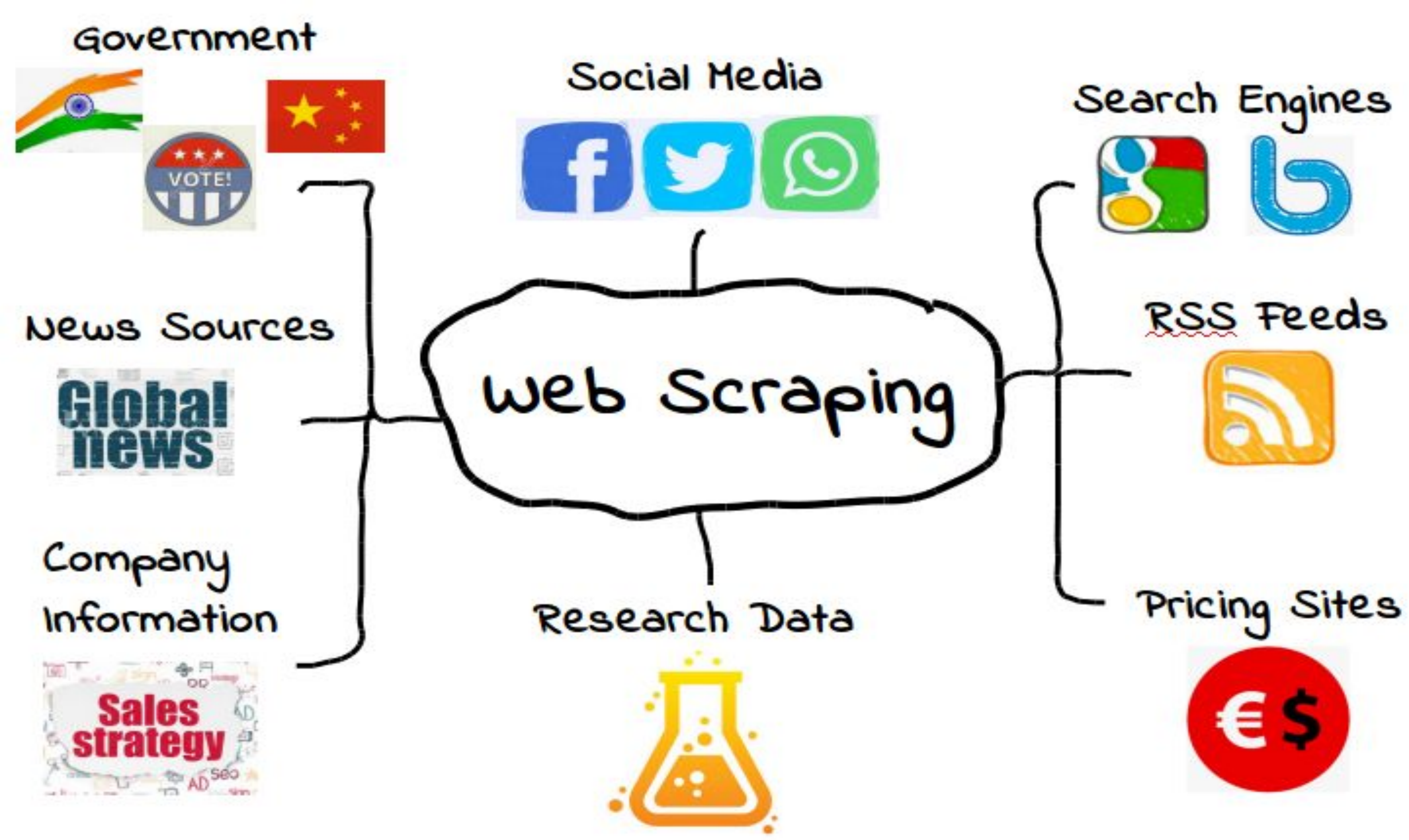
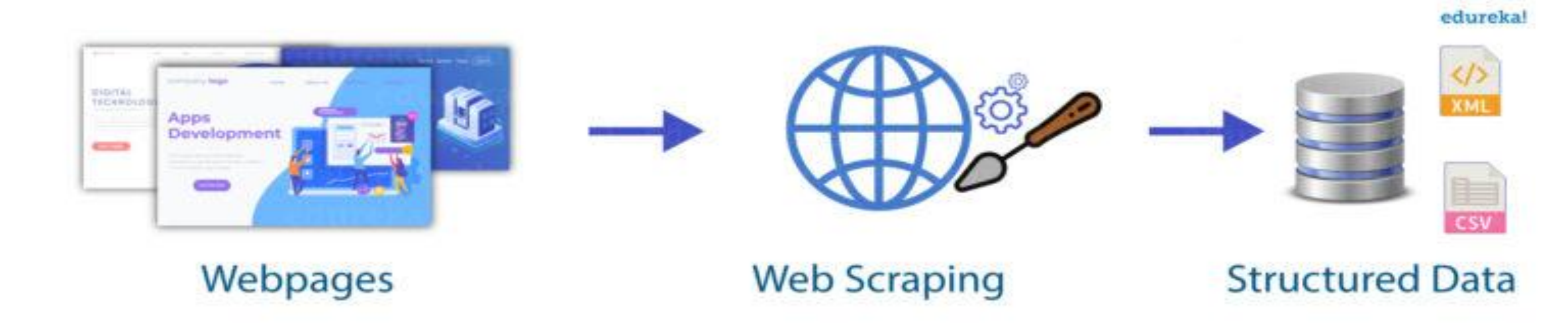
# Introduction to Web Scraping

- Aiswarya Ramachandran

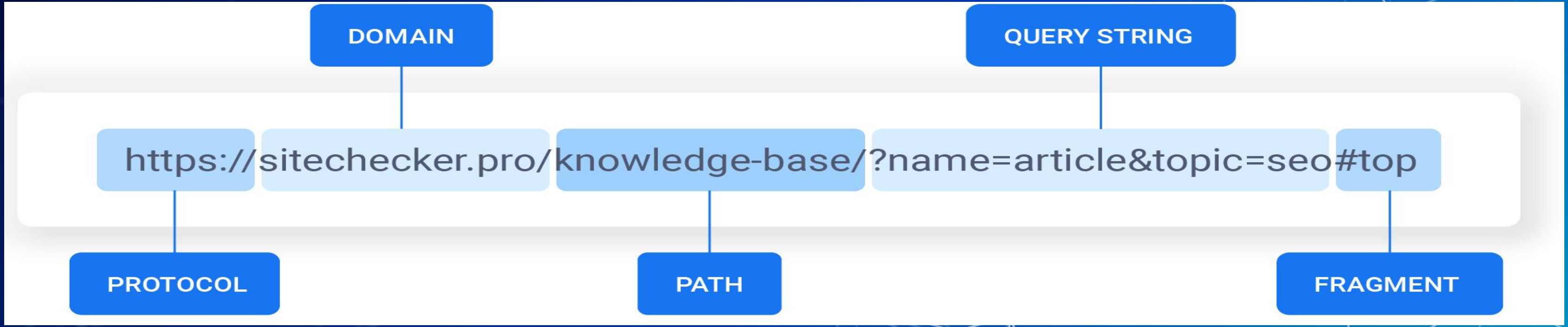

Government
VOTE!
Social Media
Search Engines
News Sources
Global news
Web Scraping
RSS Feeds
Company Information
Sales strategy
Research Data
Pricing Sites
€$



- **Automated way** to extract large amounts of data from the Web
- Collect the **Unstructured Data and Convert it into Structured Format**
- Check **robots.txt** to see if scraping is allowed
  - To check if amazon allows Web Scraping , go to :

    [https://www.amazon.in/robots.txt](https://www.amazon.in/robots.txt)

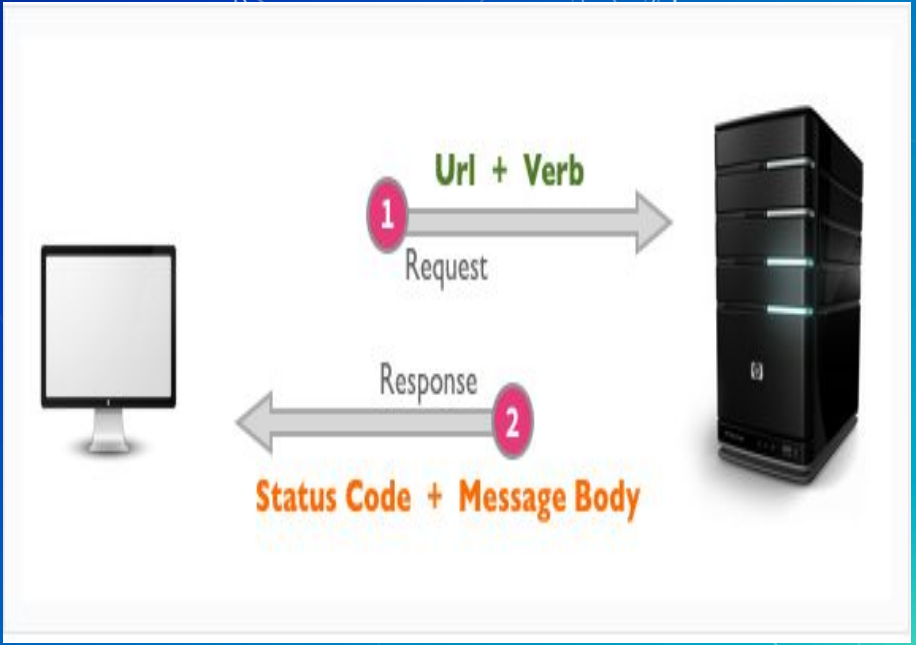
# Structure of An URL


DOMAIN
QUERY STRING
https://sitechecker.pro/knowledge-base/?name=article&topic=seo#top
PROTOCOL
PATH
FRAGMENT

- **Domain** is the target website
- **Path** specifies the server path to the data
- **Query String** is the data request action

# Request/Response Cycle

- **Request** : Message sent to an URL to fetch data
- **Response**:  The server fetches the web page content along with the status code
- **Status Code 200**: Request has succeeded
- **Status Code 403** : Accessing the web page is forbidden
- To get the content from a web page "**requests**" library is used in python



For a list **of status codes** and their meaning :
https://en.wikipedia.org/wiki/List_of_HTTP_status_

# Web Page Content

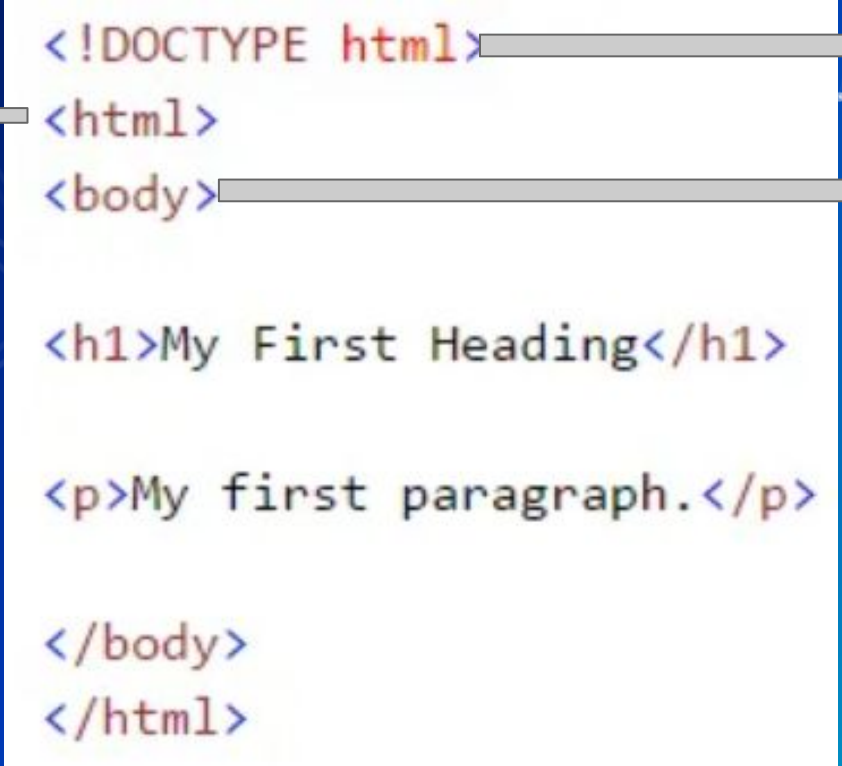
- Server Response contains 4 types of files
  - **HTML** : contains main content of the page
  - **CSS** : adding styles to page to make it beautiful
  - **Javascript**:  add interactivity to the page

# Web Page Content - HTML

- HTML - markup language that tells the browser how to layout content on the web
- Allows to do things similar to a word processor - like making text bold, starting a new paragraph
- Links are defined within the <a> tag

# HTML Basics

HTML Document is contained within this tag



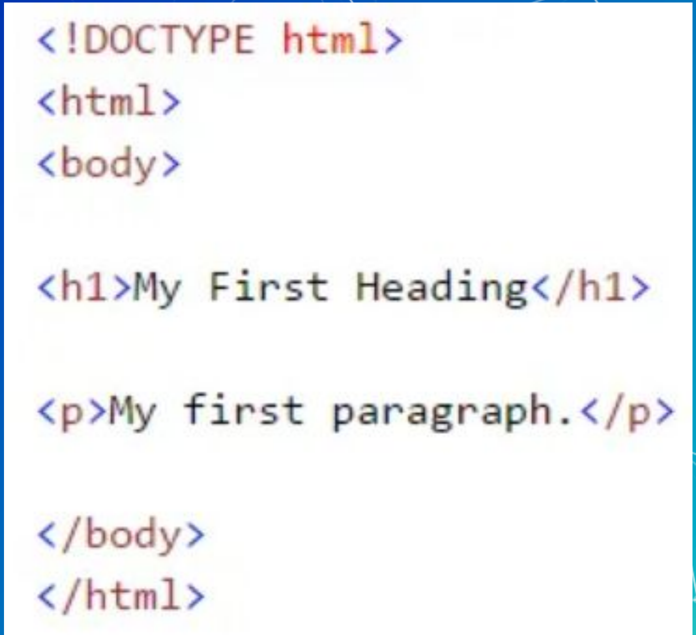
```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

HTML documents must start with a type declaration

Visible part of the web page content is contained within the <body> & </body> tag

# Tags in HTML

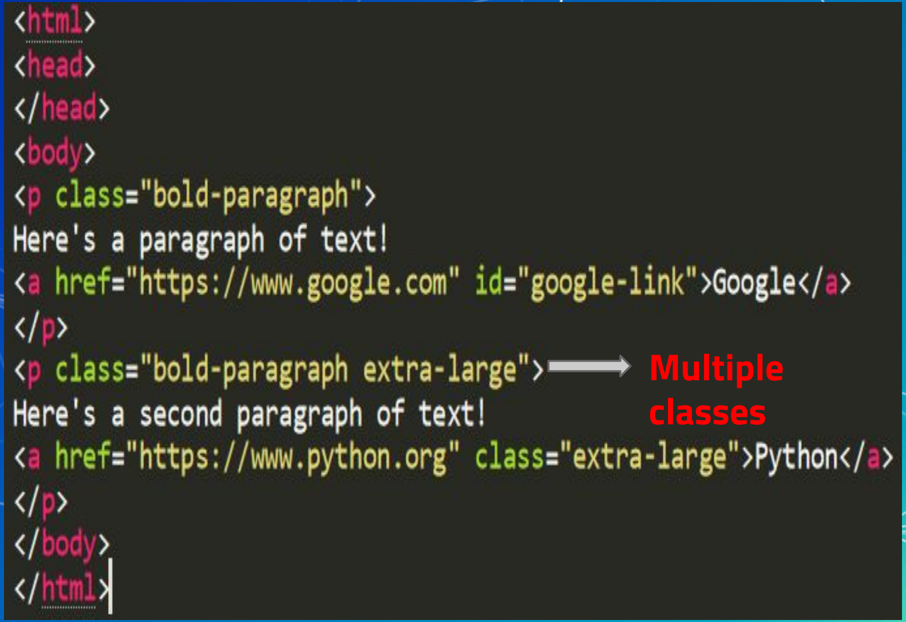
- **Child Tag** : Tag inside another tag
  - <h1> and <p> are child tags of <body>
- **Parent Tag**: <html> is the parent tag of <body>tag
- **Sibling Tag** : <h1> and <p> tags are sibling tags



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

# Properties in HTML Tags - Class and ID

- Class and ID are special properties that give HTML elements name
- An element can belong to multiple classes and a class can be shared among elements
- Each element can have only one id.
- An Id can be used only once on and page



```
<html>
<head>
</head>
<body>
<p class="bold-paragraph">
Here's a paragraph of text!
<a href="https://www.google.com" id="google-link">Google</a>
</p>
<p class="bold-paragraph extra-large">
Here's a second paragraph of text!
<a href="https://www.python.org" class="extra-large">Python</a>
</p>
</body>
</html>
```

Multiple classes

# Beautiful Soup Commands

- **BeautifulSoup(response.content)** - converts the content into a proper format
- **soup.find(tag, attrs)** - finds the first instance of a tag with the given attributes. Attrs is optional
- **soup.findAll(tag,attrs)**- returns all instances of a tag with the given attributes
- **get_text()** - extracts all text within a tag
- **soup.select(p a)** - selects all <a> tags within a <p> tag
- **soup.select(p.outer-text)** - selects all <p> tags with class outer-text

# STEPS TO SCRAP DATA

1. Get the URL of the page to be scrapped.
2. Inspect the elements of the Page and identify the tags required.
3. Access the URL
4. Get the element from the required tags

# WHEN BEAUTIFUL SOUP WILL FAIL?

- On Youtube for example where there is “infinite” scrolling.To overcome this, we can use Selenium - allows us to open browser and automate process of scrolling
- “Scrapy” is another library which can be used for Scraping - especially when you need to do a lot of parallel processing and collect data