

# Web Scraping & Web Services



# Hello!

I am Eslam Ahmed

I am a software engineer.

You can find me at [jeksogsa@gmail.com](mailto:jeksogsa@gmail.com)



# Hello!

I am Eman Ehab

I am a ML research engineer.

You can find me at  
[emanehab.ieee@gmail.com](mailto:emanehab.ieee@gmail.com)



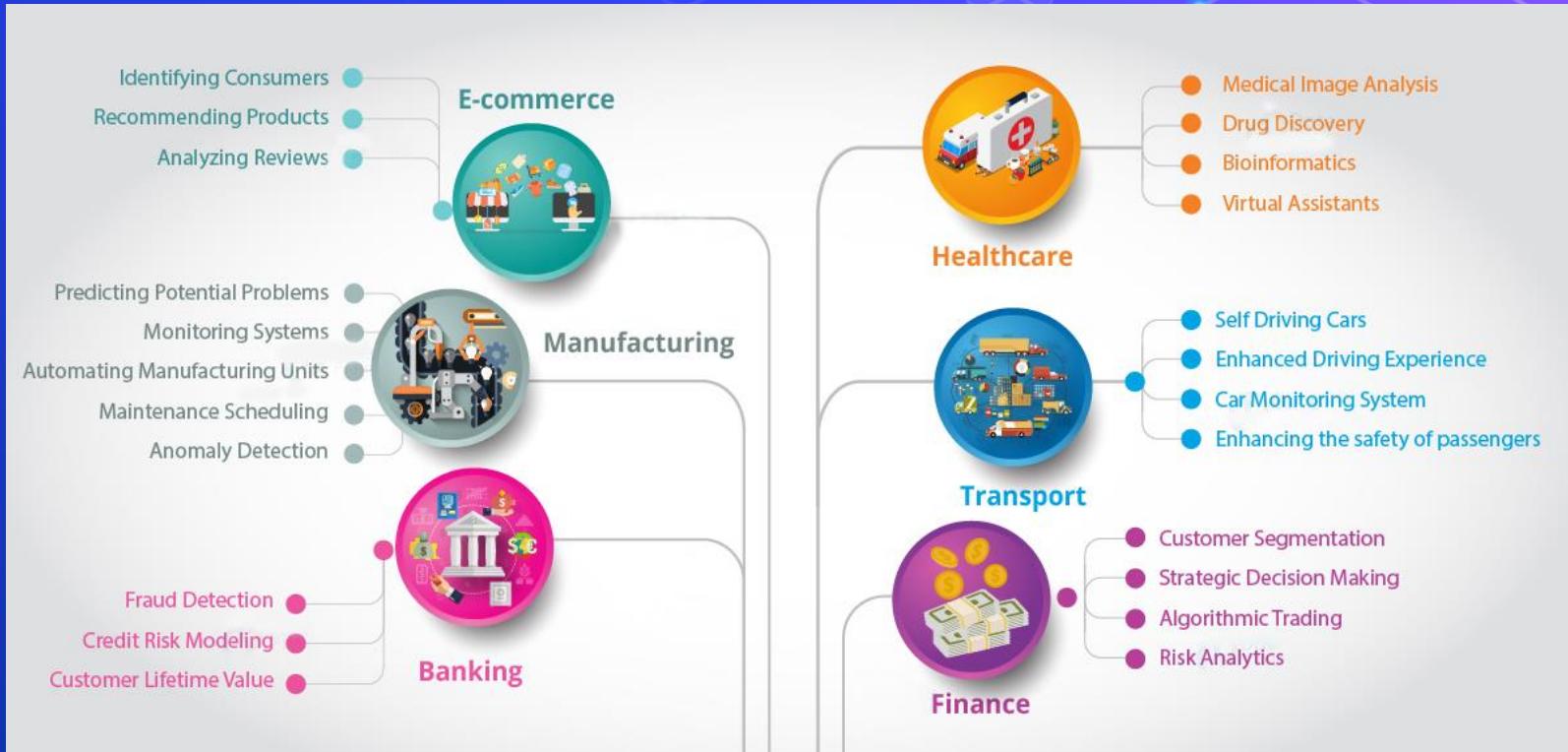
# What is Data Science

Data science is a concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods" in order to "understand and analyze actual phenomena with data.

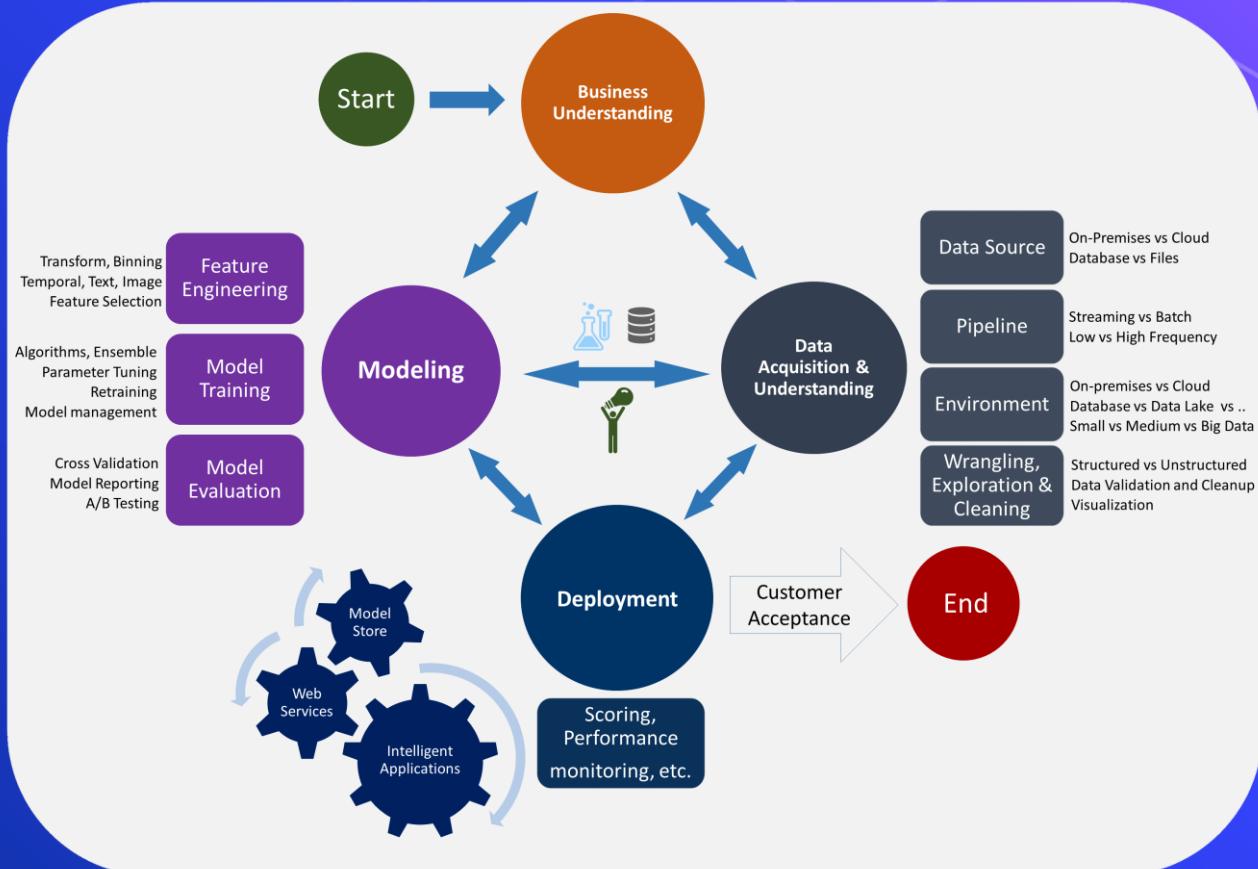
It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge and information science.



# Applications of Data Science



# Data Science life cycle



# Data Collecting



Public  
Datasets



Datasets  
for sale



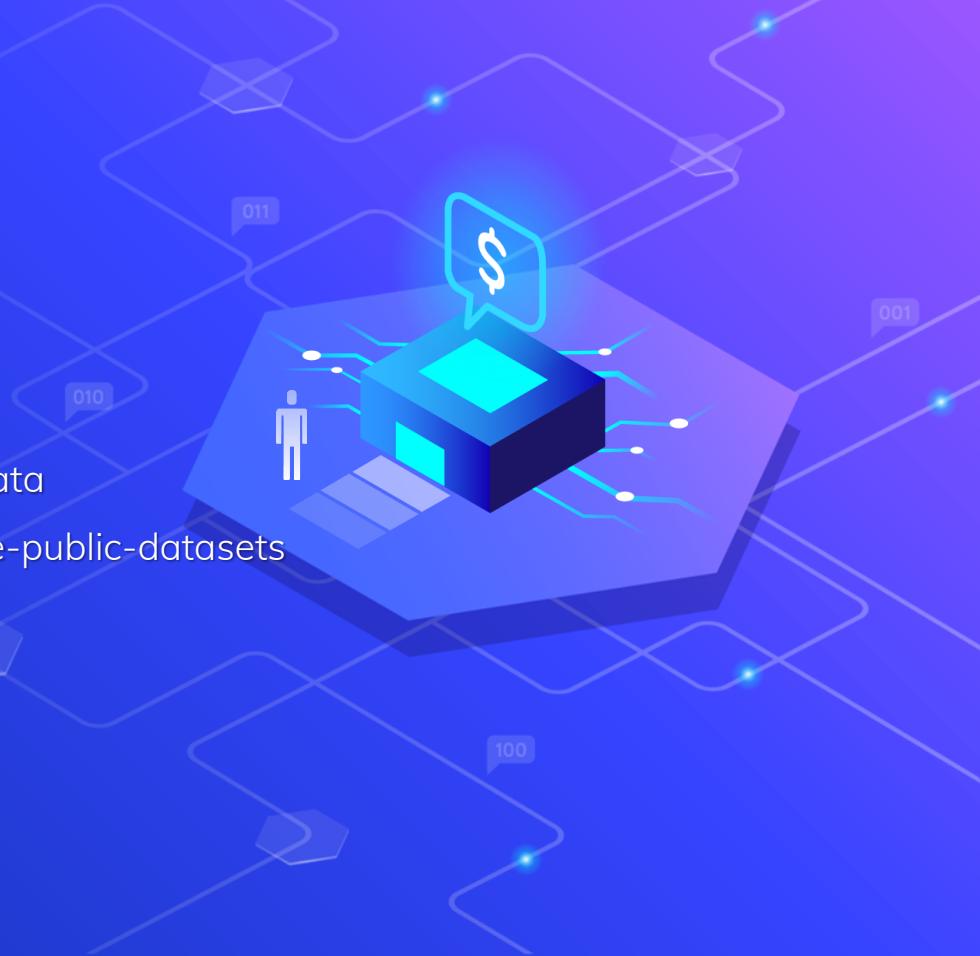
APIs



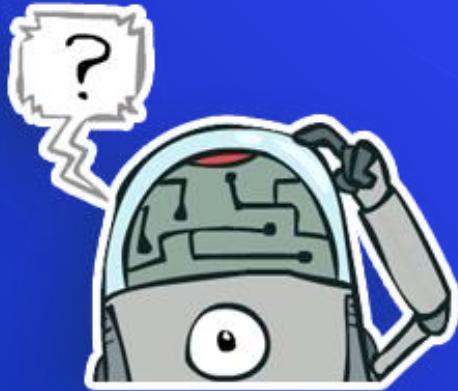
Scraping

# Public Datasets

- <https://www.kaggle.com/datasets>
- <http://archive.ics.uci.edu/ml/index.php>
- <https://data.fivethirtyeight.com>
- <https://github.com/BuzzFeedNews>
- <https://opendata.socrata.com>
- <https://cloud.google.com/bigquery/public-data>
- <https://github.com/awesomedata/awesome-public-datasets>
- <https://www.data.gov>
- <http://academictorrents.com/browse.php>
- <https://www.quandl.com/search>
- <https://tinyletter.com/data-is-plural>



# Data Collecting



So What we do if there is  
no datasets available ?!



We will learn about Web Scraping and Rest APIs web services

# Agenda

- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap** Library
- REST API Web Services
- Work with JSON



# Agenda

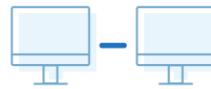
- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- BeautifulSoup Library
- REST API Web Services
- Work with JSON



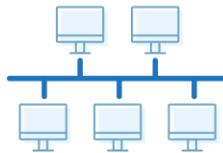
# What is Network Topologies

## Network Topology Types

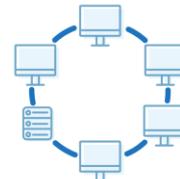
1 Point to point



2 Bus



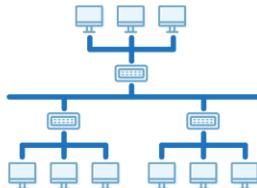
3 Ring



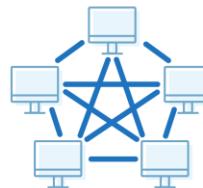
4 Star



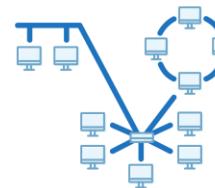
5 Tree



6 Mesh



7 Hybrid

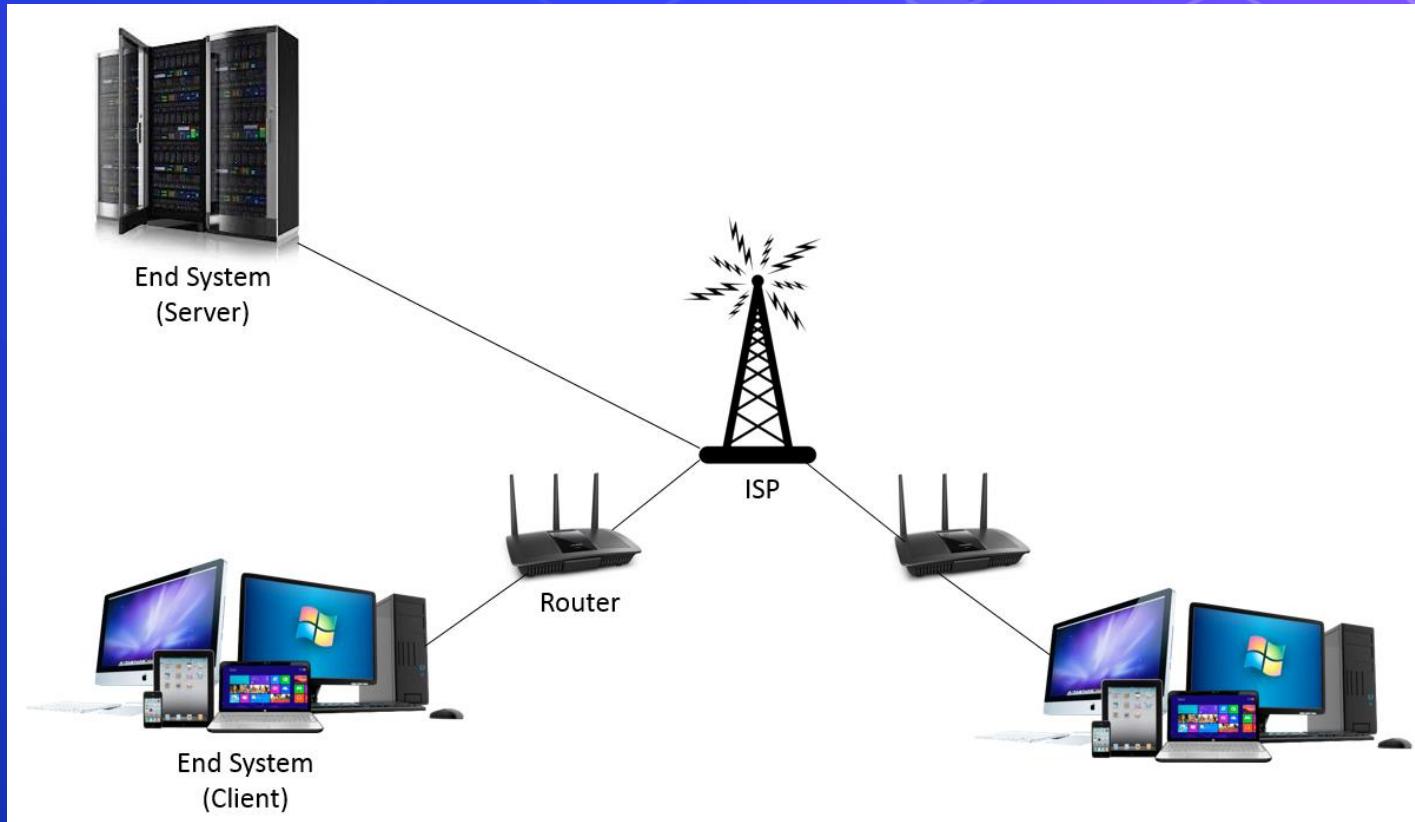


# Agenda

- What is Network Topologies
- **What is Internet and Web Servers**
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap Library**
- REST API Web Services
- Work with JSON



# What is Internet and Web Servers

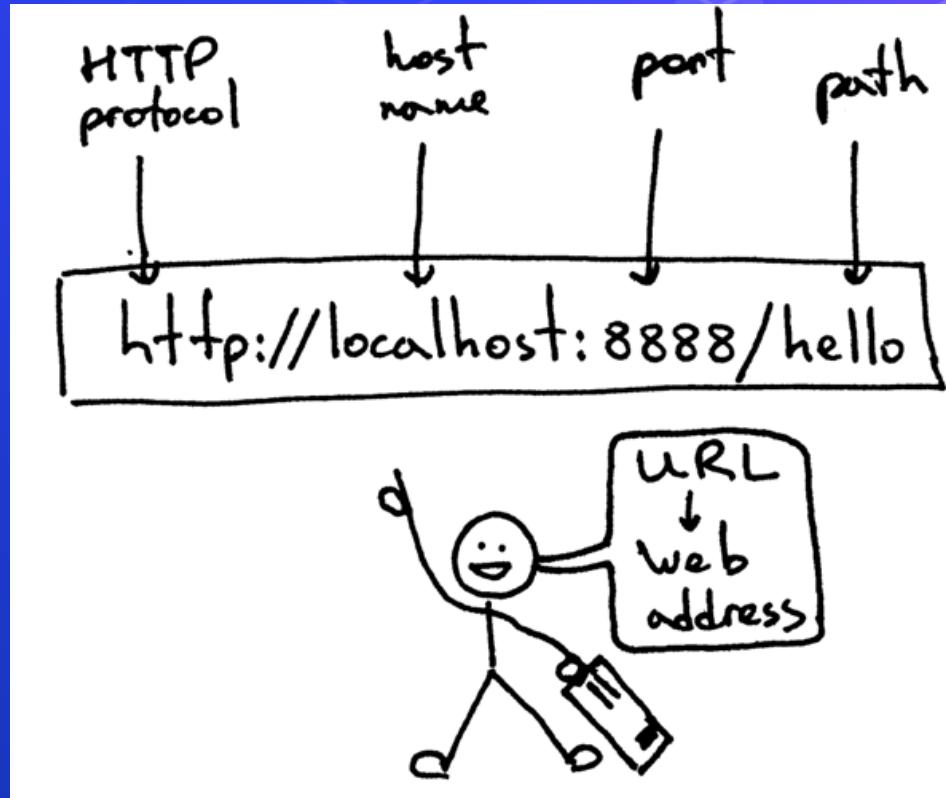


# Agenda

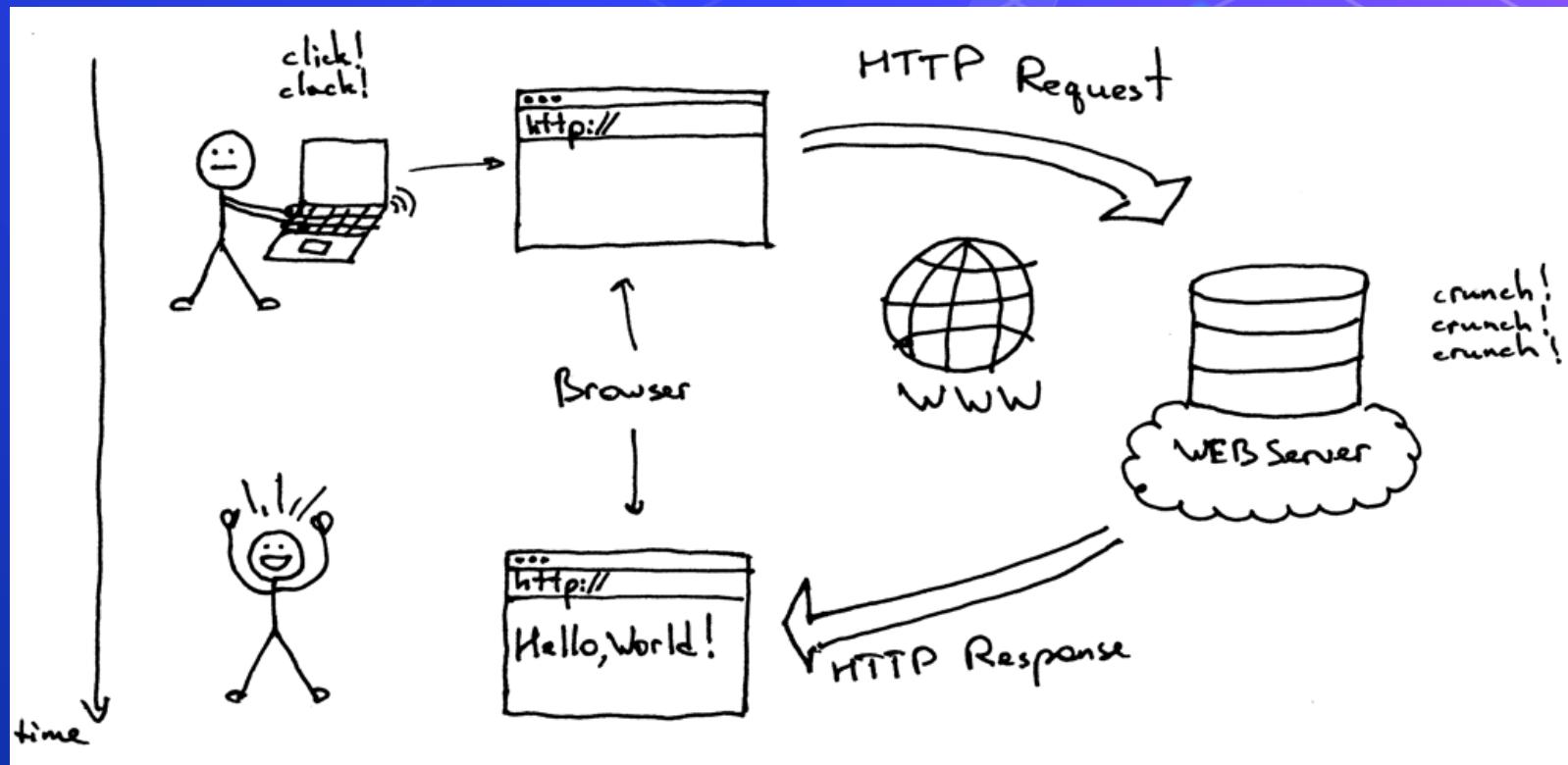
- What is Network Topologies
- What is Internet and Web Servers
- **HTTP Request/Response Cycle**
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap Library**
- REST API Web Services
- Work with JSON



# HTTP Request/Response Cycle

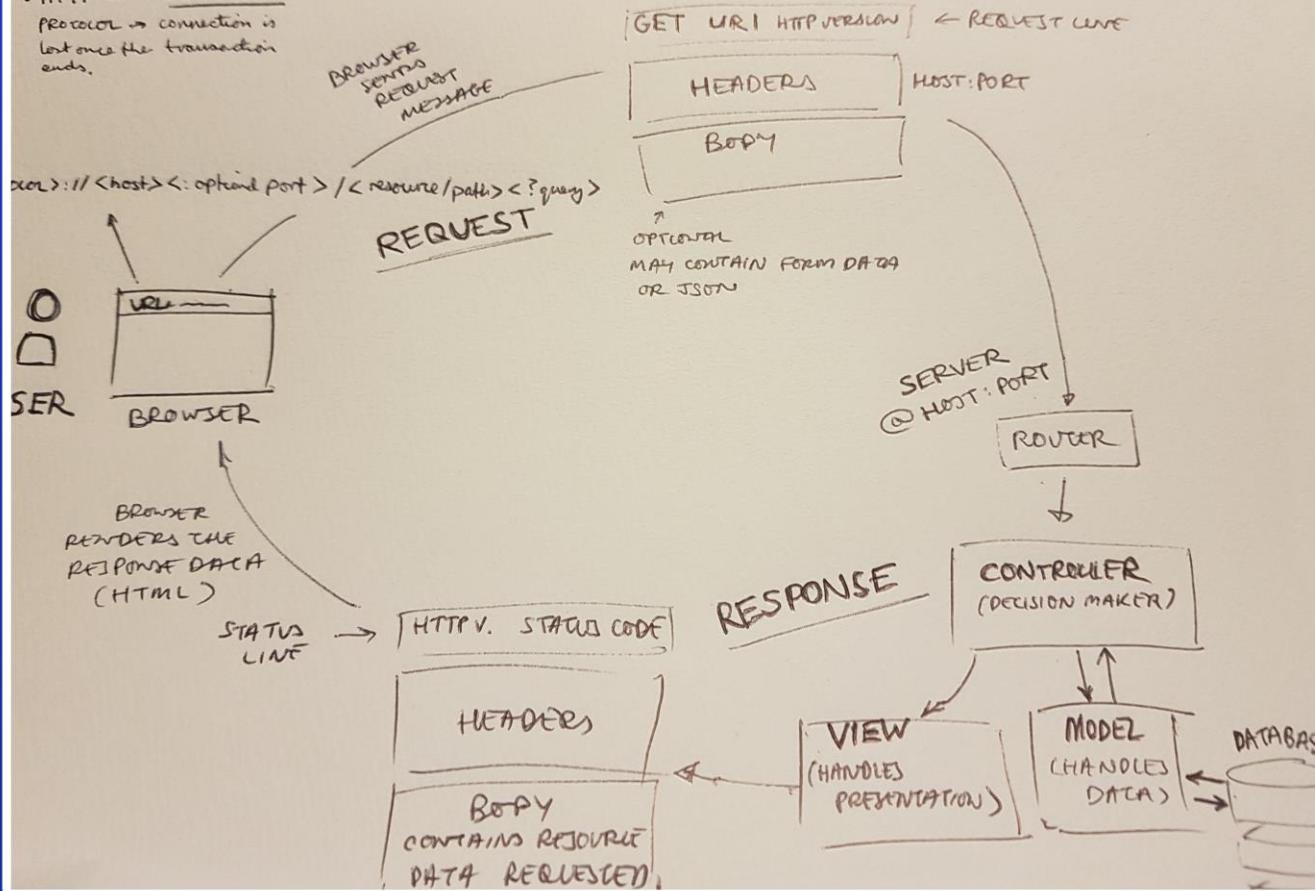


# HTTP Request/Response Cycle



# HTTP REQUEST-RESPONSE CYCLE

- HTTP is a STATELESS protocol → connection is lost once the transaction ends.



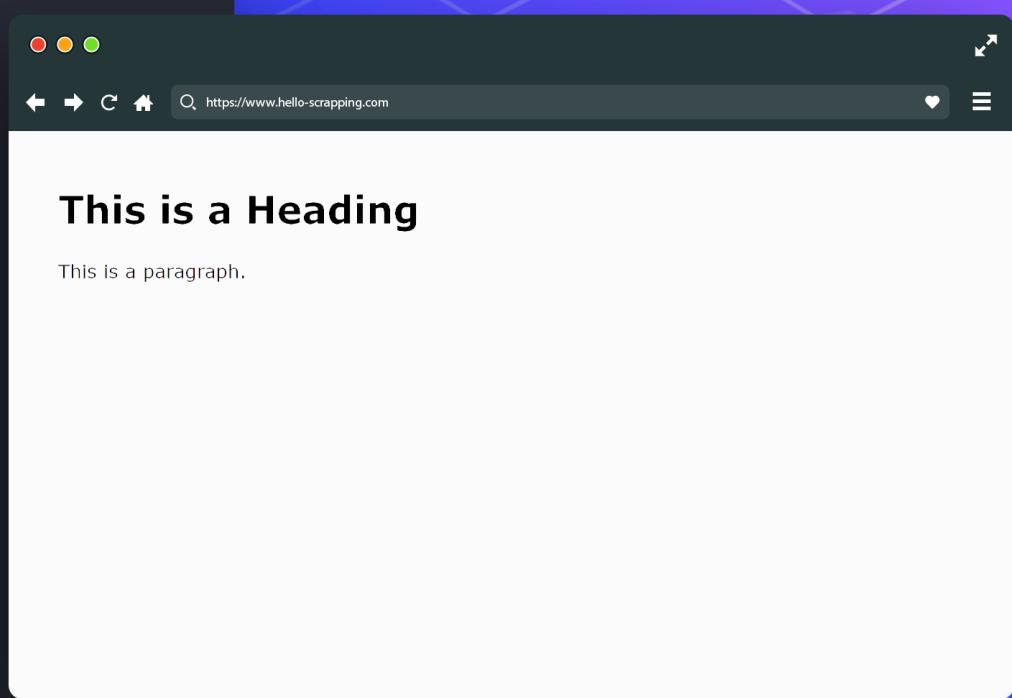
# Agenda

- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap** Library
- REST API Web Services
- Work with JSON



# HTML (base snippet code)

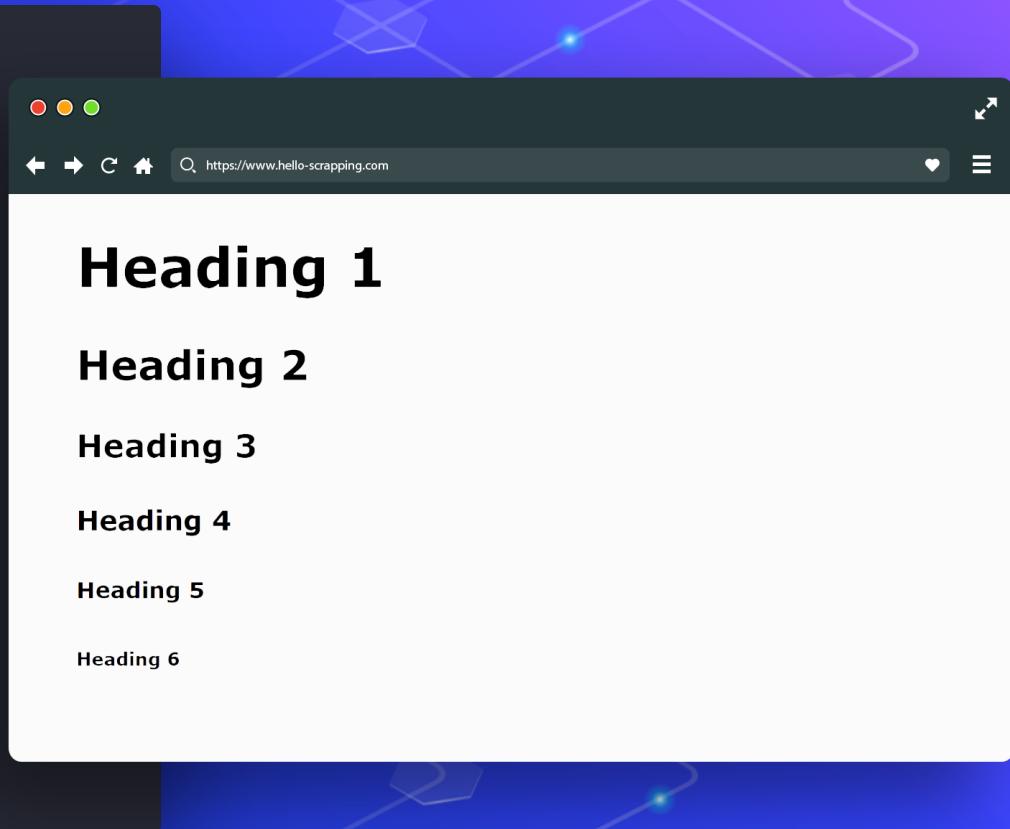
```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <title>Page Title</title>
6   </head>
7
8
9   <body>
10
11     <h1>This is a Heading</h1>
12     <p>This is a paragraph.</p>
13
14   </body>
15 </html>
16
```



# HTML (header)

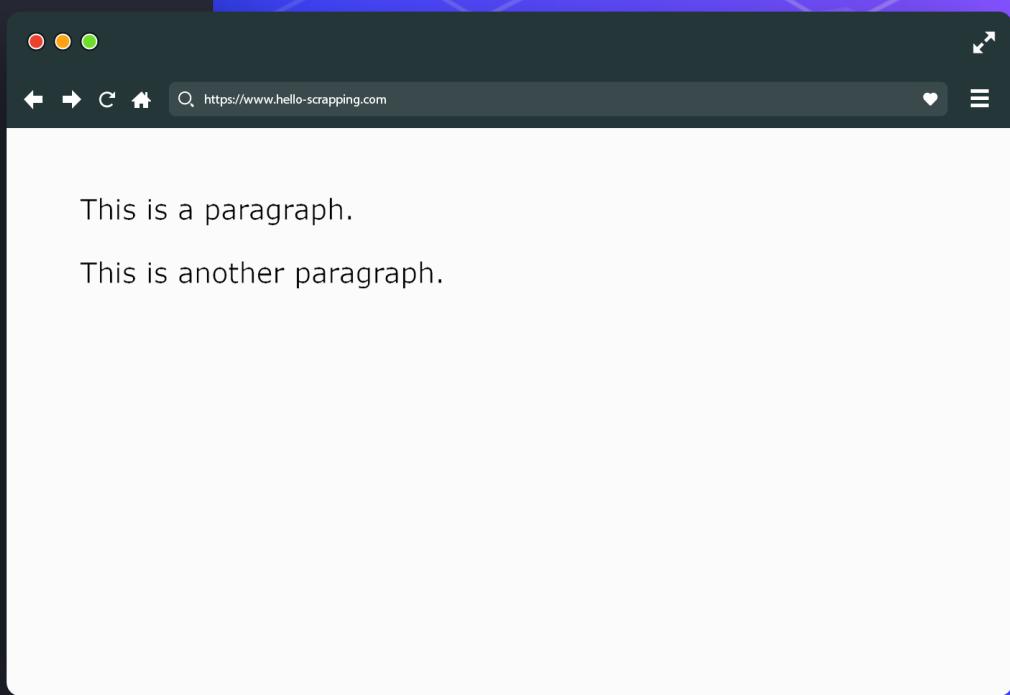


```
1 <!DOCTYPE html>
2 <html>
3
4     <head>
5         <title>Page Title</title>
6     </head>
7
8
9     <body>
10
11         <h1>Heading 1</h1>
12         <h2>Heading 2</h2>
13         <h3>Heading 3</h3>
14         <h4>Heading 4</h4>
15         <h5>Heading 5</h5>
16         <h6>Heading 6</h6>
17
18     </body>
19 </html>
20
```



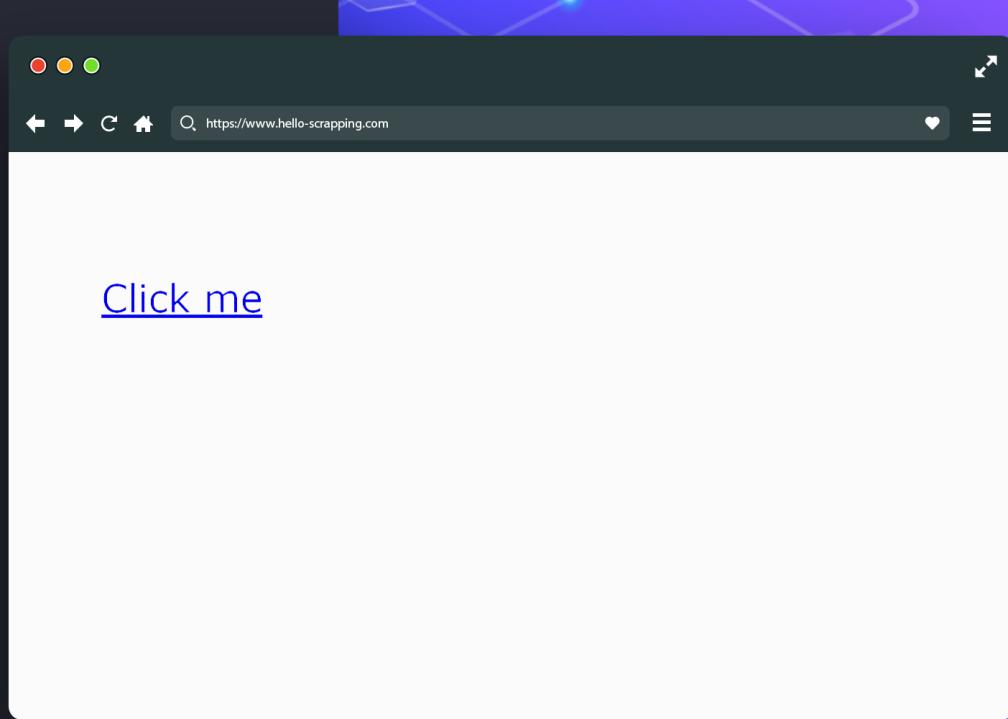
# HTML (paragraph)

```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <title>Page Title</title>
6   </head>
7
8
9   <body>
10
11     <p>This is a paragraph.</p>
12     <p>This is another paragraph.</p>
13
14   </body>
15 </html>
16
```



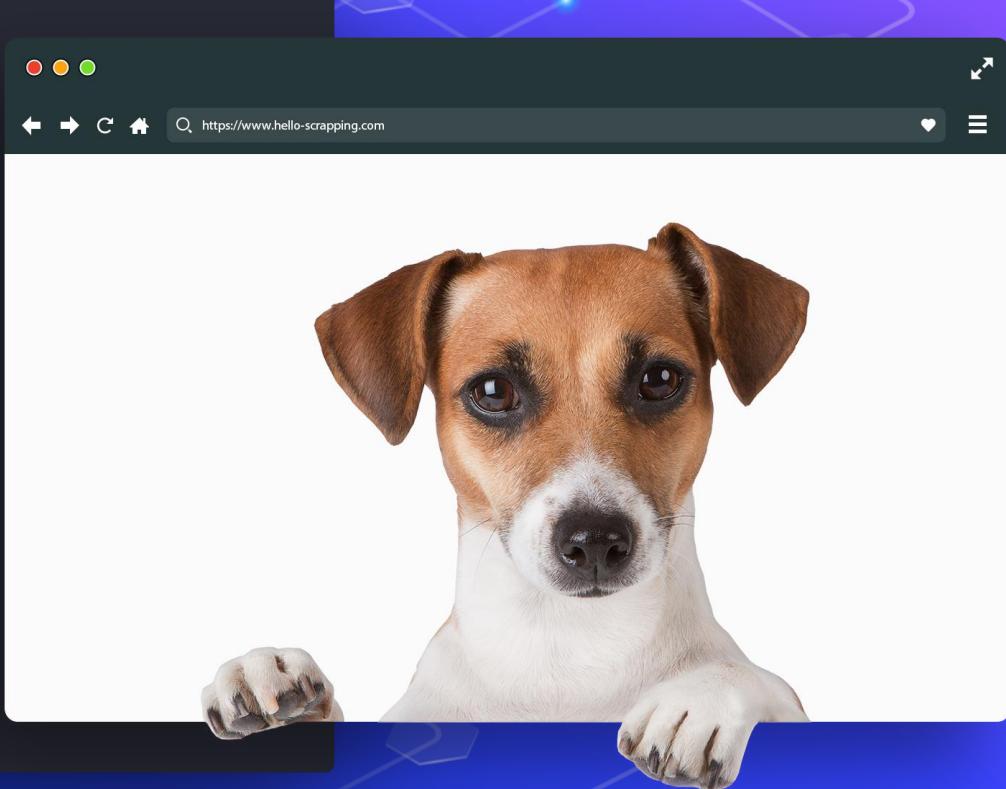
# HTML (link)

```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <title>Page Title</title>
6   </head>
7
8
9   <body>
10
11     <a href="url">Click me</a>
12
13   </body>
14 </html>
15
```



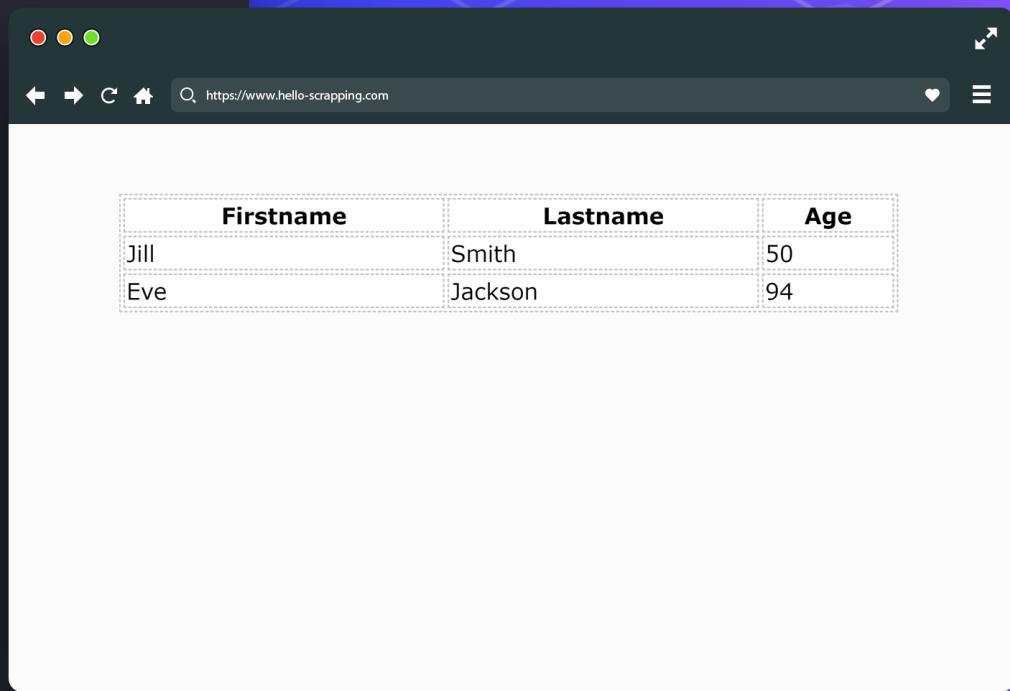
# HTML (image)

```
1 <!DOCTYPE html>
2 <html>
3
4     <head>
5         <title>Page Title</title>
6     </head>
7
8
9     <body>
10
11         
12
13     </body>
14 </html>
15
```



# HTML (table)

```
1 <table style="width:100%">
2   <tr>
3     <th>Firstname</th>
4     <th>Lastname</th>
5     <th>Age</th>
6   </tr>
7   <tr>
8     <td>Jill</td>
9     <td>Smith</td>
10    <td>50</td>
11  </tr>
12  <tr>
13    <td>Eve</td>
14    <td>Jackson</td>
15    <td>94</td>
16  </tr>
17 </table>
18
```



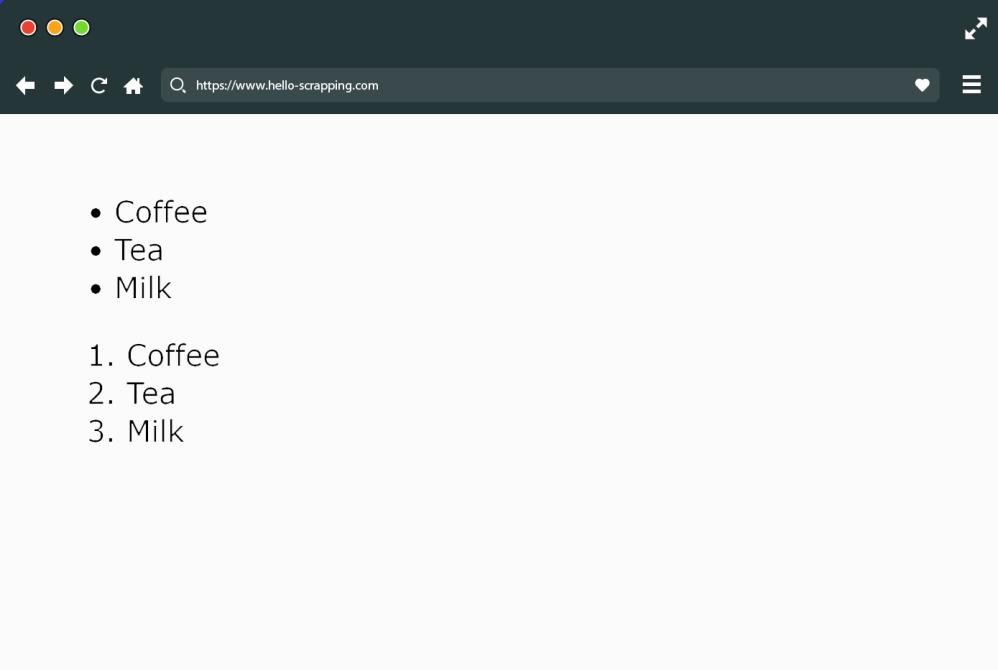
A screenshot of a web browser window displaying a simple HTML table. The browser interface includes a header with three dots (red, yellow, green), a back/forward button, a refresh/circular arrow button, a home icon, and a search bar containing the URL <https://www.hello-scraping.com>. To the right of the search bar are icons for a heart, a list, and a menu.

The main content area shows a table with three columns: Firstname, Lastname, and Age. The table has four rows of data:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

# HTML (list)

```
1 <ul>
2     <li>Coffee</li>
3     <li>Tea</li>
4     <li>Milk</li>
5 </ul>
6
7 <ol>
8     <li>Coffee</li>
9     <li>Tea</li>
10    <li>Milk</li>
11 </ol>
```



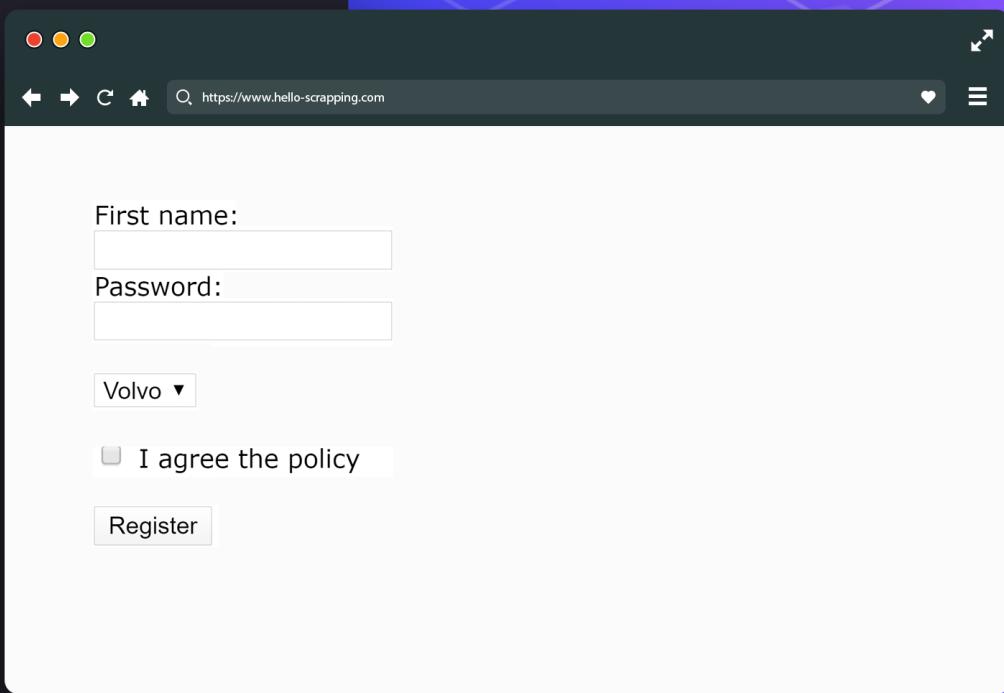
The screenshot shows a web browser window with a dark theme. The address bar displays the URL <https://www.hello-scrappling.com>. The main content area shows two lists:

- Coffee
- Tea
- Milk

1. Coffee
2. Tea
3. Milk

# HTML (form)

```
1 <form>
2   First name: <input type="text">
3   Password: <input type="password">
4
5   <select>
6     <option>Volvo</option>
7     <option>Saab</option>
8     <option>Fiat</option>
9     <option>Audi</option>
10
11   </select>
12
13
14   <input type="checkbox">
15   I agree the policy
16
17
18   <input type="submit" value="Register">
19
20 </form>
```



A screenshot of a web browser window displaying a registration form. The browser has a dark theme with a green header bar. The address bar shows the URL <https://www.hello-scraping.com>. The page content is as follows:

First name:

Password:

Volvo ▾

I agree the policy

# Other HTML Elements

- Div
- Span
- Video
- Audio
- Iframe
- Header
- Footer
- Canvas
- ...



# Agenda

- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap** Library
- REST API Web Services
- Work with JSON



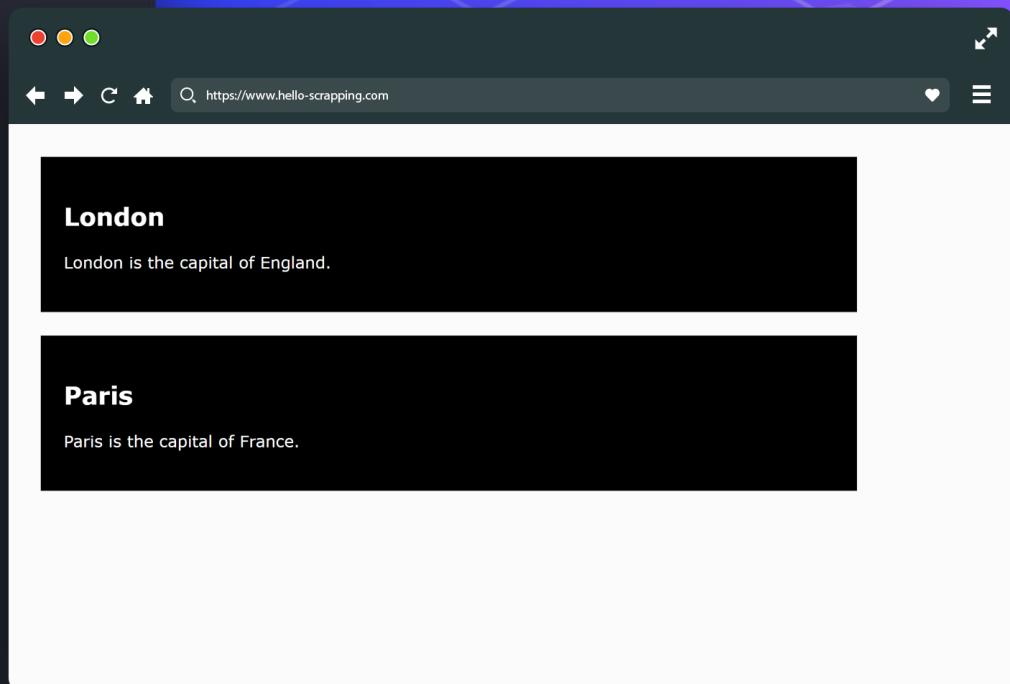
# CSS

- Text & Fonts
- Colors
- Backgrounds
- Borders
- Margin & Padding
- Width & Height
- Gradient
- Shadows
- ...



# Classes and ID

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       .cities
6     {
7       background-color: black;
8       color: white;
9       margin: 20px;
10      padding: 20px;
11    }
12  </style>
13 </head>
14 <body>
15
16 <div class="cities">
17   <h2>London</h2>
18   <p>London is the capital of England.</p>
19 </div>
20
21 <div class="cities">
22   <h2>Paris</h2>
23   <p>Paris is the capital of France.</p>
24 </div>
25
26 </body>
27 </html>
```



# Check Also

- JavaScript
- React & Angular
- Ajax
- Web Sockets
- PHP: Laravel
- Python: Flask, Django
- JavaScript: Node.js
- ...



# Agenda

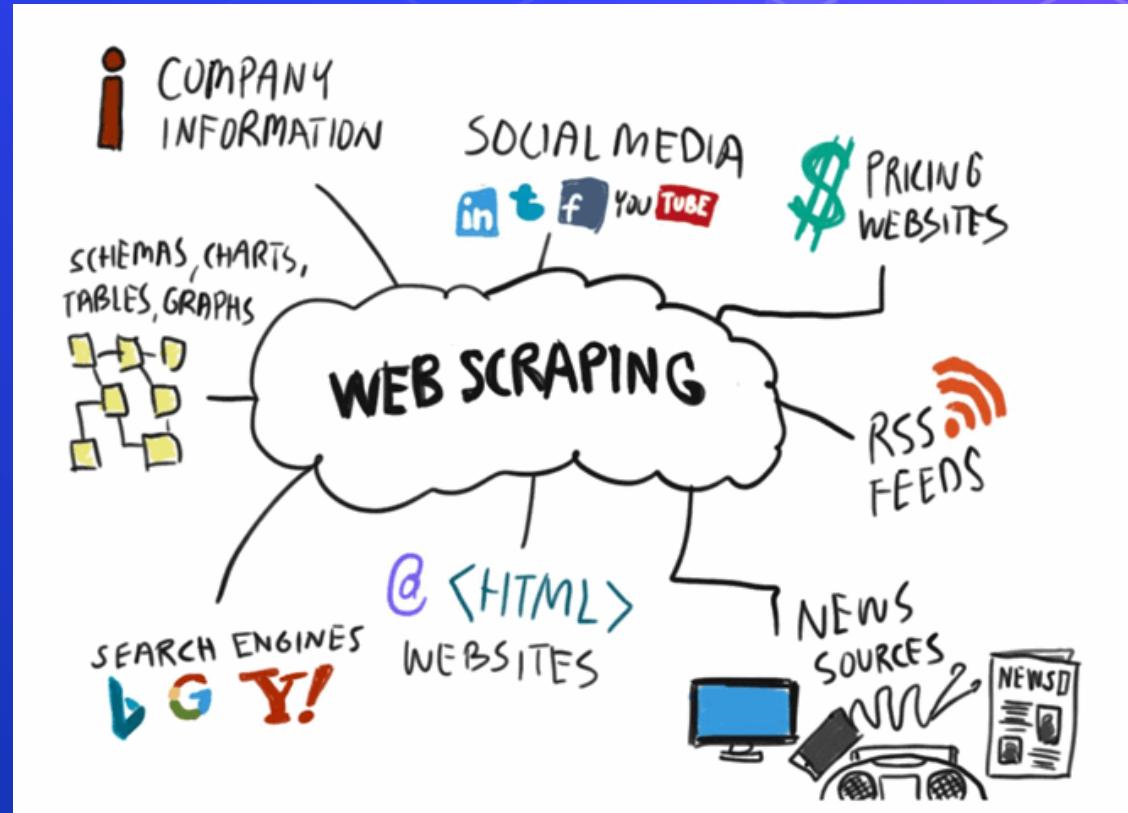
- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scraping Concept
- BeautifulSoup Library
- REST API Web Services
- Work with JSON



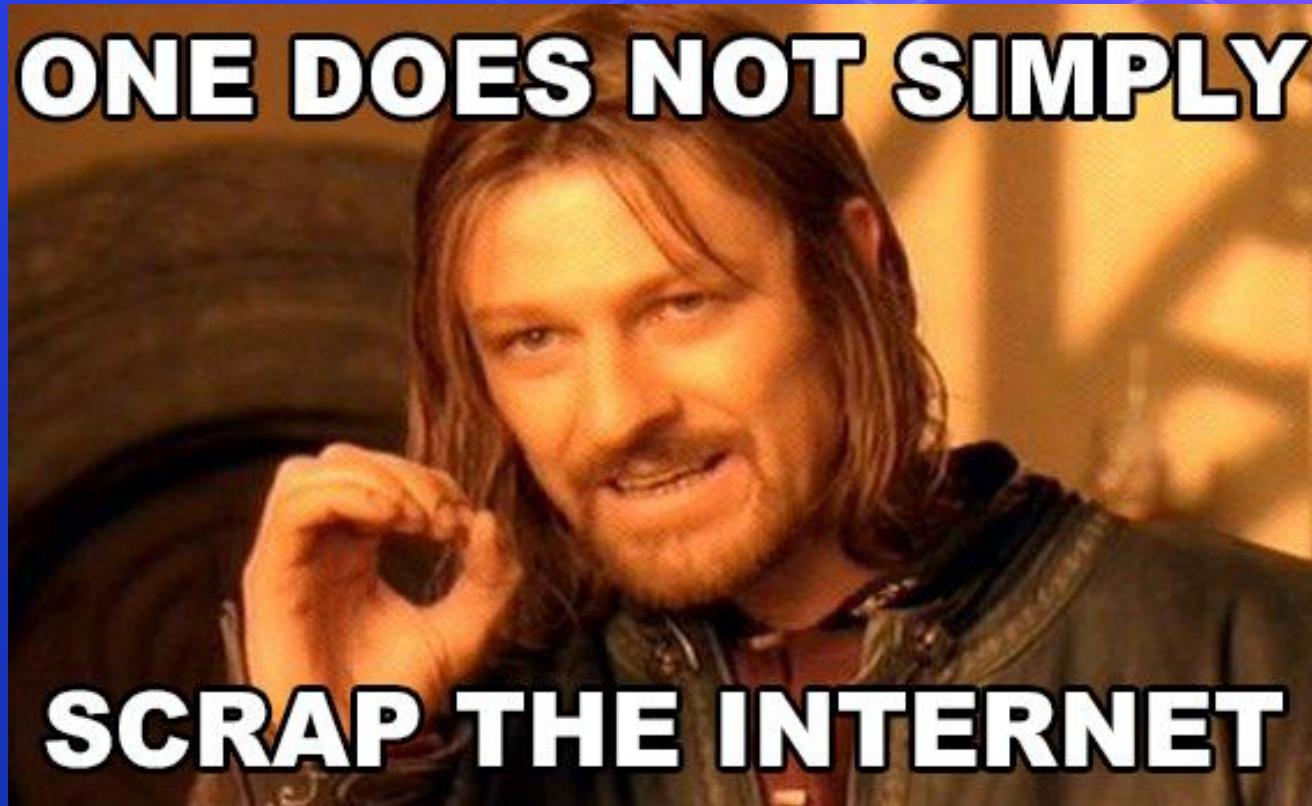
# Scraping Concept



# Scraping Concept



## Scraping Concept



# Agenda

- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- [BeautifulSoap Library](#)
- REST API Web Services
- Work with JSON



# BeautifulSoup Library

## Web Scraping



```
1 conda install beautifulsoup4
```

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = 'https://jekso.github.io/scrapping-example/index.html'
5 response = requests.get(url)
6
7 soup = BeautifulSoup(response.text, 'html.parser')
8
9 # -----
10
11 # find first element by type
12 link = soup.find('a')
13
14 # find first element by type and an attribute
15 link = soup.find('a', attrs={'class': 'social-link'})
16
17 # -----
18
19 # find all elements by type
20 all_links = soup.find_all('a')
21
22 # find all elements by type and an attribute
23 all_links = soup.find_all('a', attrs={'class': 'social-link'})
24
25 # -----
26
27 # check if element has attribute
28 link.has_attr('class')
29
30 # get attribute
31 link.get('href')
32
33 # -----
34
35 # get content
36 soup.find('a').get_text()
37
```

# Check Also

Other methods to scrap the internet.

- <https://scrapy.org>
- <https://selenium-python.readthedocs.io>
- ...

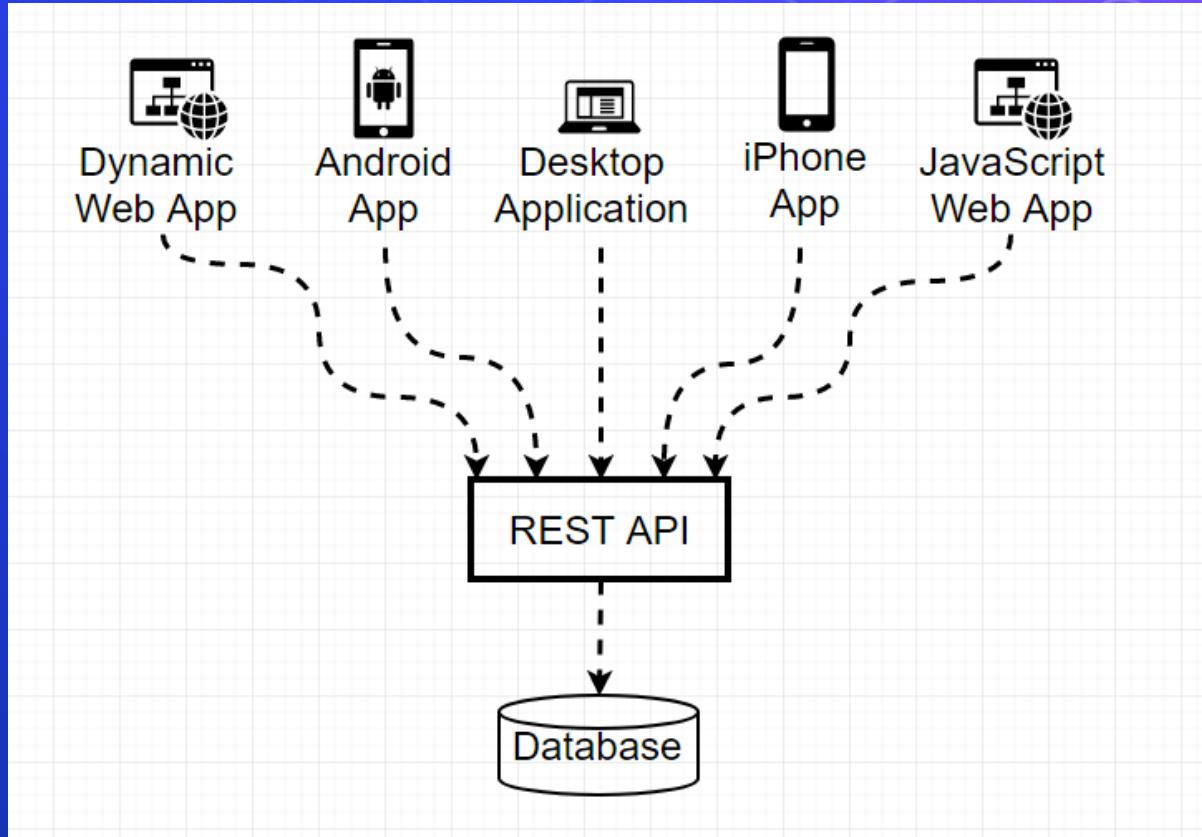


# Agenda

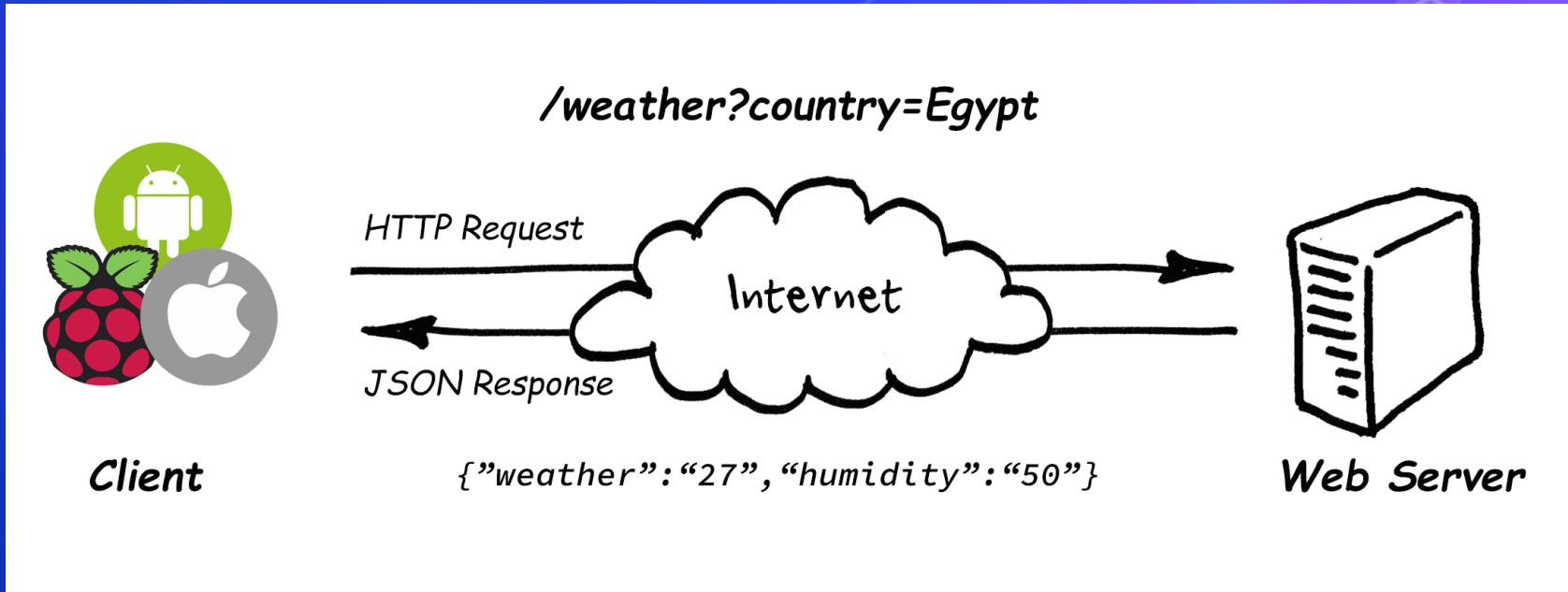
- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- BeautifulSoup Library
- REST API Web Services
- Work with JSON



# REST API Web Services

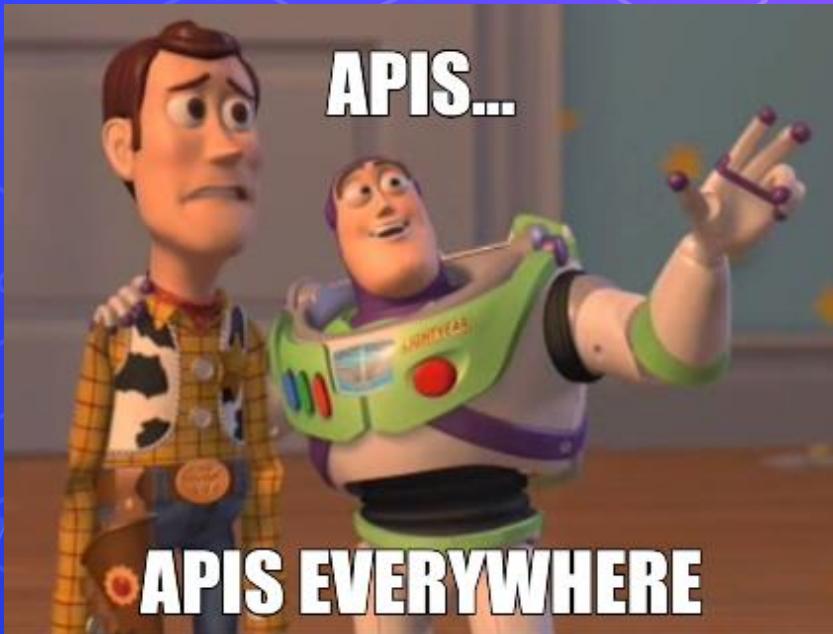


# REST API Web Services



# REST API Web Services

- <https://any-api.com/>
- <https://github.com/public-apis/public-apis>
- <https://cloud.google.com/products/ai>
- <https://www.twilio.com/>
- ...



# Agenda

- What is Network Topologies
- What is Internet and Web Servers
- HTTP Request/Response Cycle
- HTML
- CSS
- Scrapping Concept
- **BeautifulSoap Library**
- REST API Web Services
- Work with JSON



# Work with JSON

```
1 {
2     "employees": [
3         {
4             "id": "1",
5             "employee_name": "Ahmed",
6             "employee_salary": "320800",
7             "employee_age": "61"
8         },
9         {
10            "id": "2",
11            "employee_name": "Amr",
12            "employee_salary": "170750",
13            "employee_age": "63"
14        },
15        {
16            "id": "3",
17            "employee_name": "Sara",
18            "employee_salary": "86000",
19            "employee_age": "66"
20        }
21    ]
22 }
23 }
```



# Work with JSON

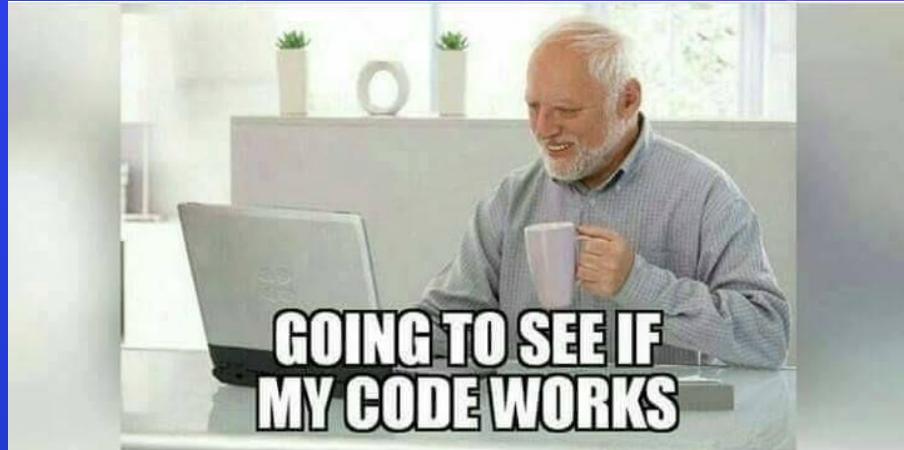
Install JSON Viewer extension  
For chrome or firefox

<https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpalibnhdgobh>

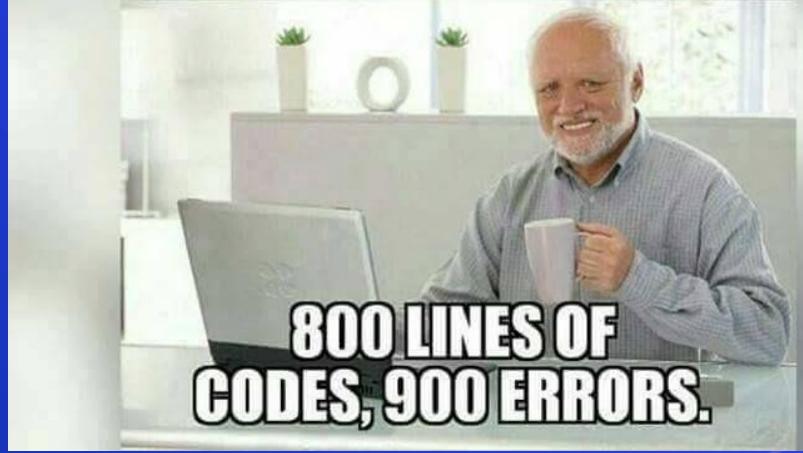
# Work with JSON

```
1 import json
2
3 # some JSON string:
4 json_text = '{ "name":"John", "age":30, "city":"New York"}'
5
6 # parse json_text:
7 user = json.loads(json_text)
8
9 # the result is a Python dictionary:
10 print(user["age"])
```





**GOING TO SEE IF  
MY CODE WORKS**



# Questions ?!



# Thanks!

>\_ Live long and prosper

