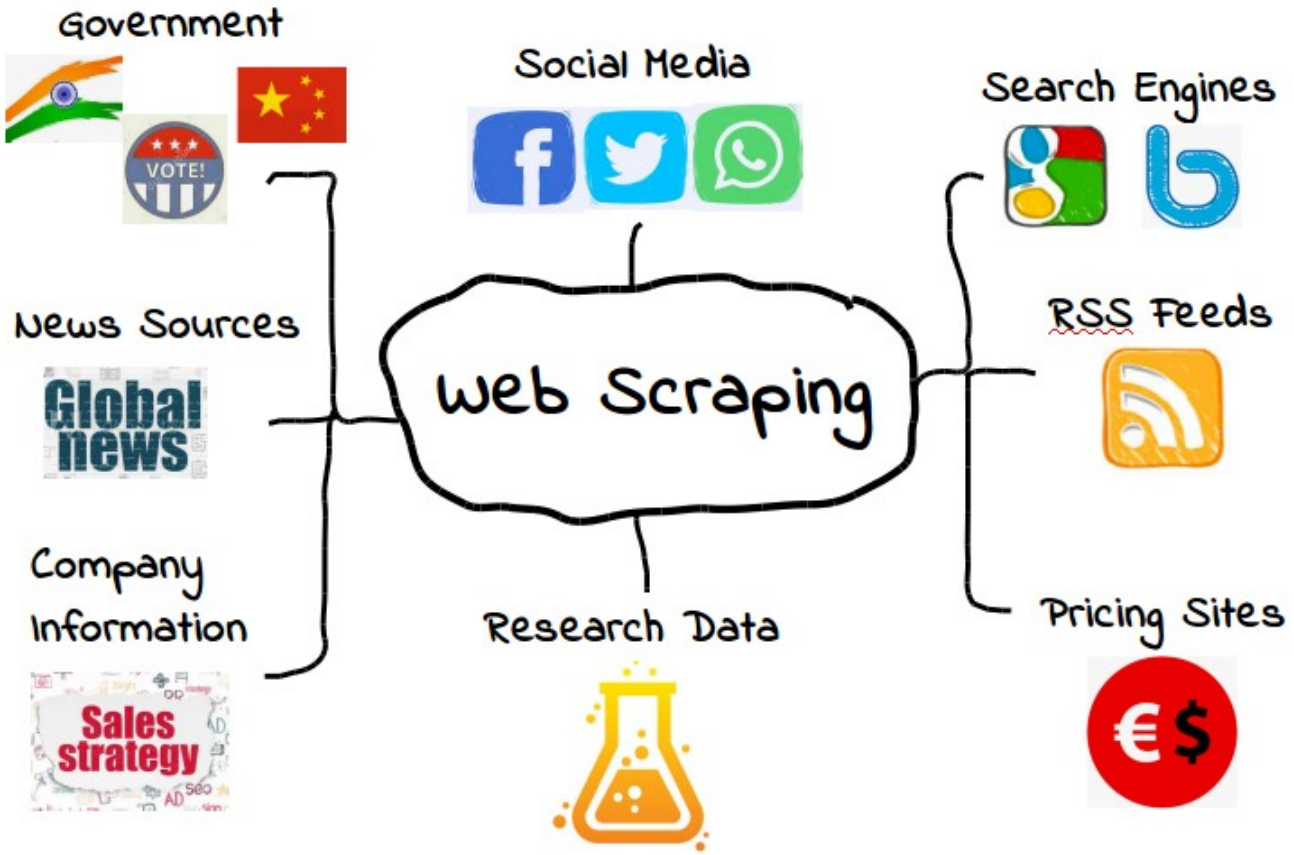
PYTHON WEB SCRAPING

Pertemuan 13 – 14

INTRODUCTION OF WEB SCRAPING

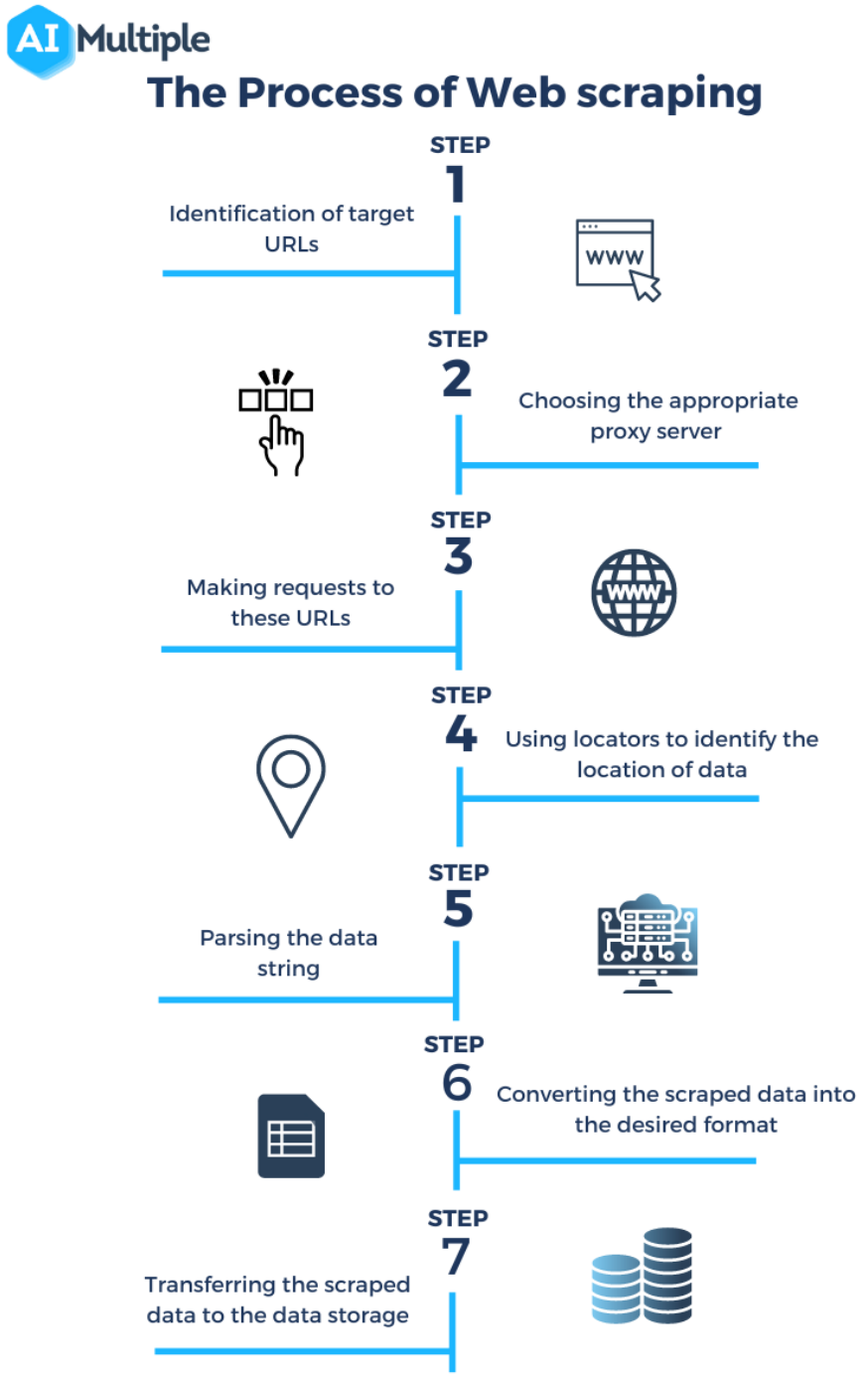
- Web Scraping adalah salah satu Teknik yang digunakan untuk mengambil data dari sebuah situs website.
- Data yang diambil adalah data yang tidak terstruktur dalam format HTML yang kemudian diubah menjadi data terstruktur dalam spreadsheet atau database sehingga dapat digunakan dalam berbagai aplikasi.

INTRODUCTION OF WEB SCRAPING



- Ada banyak cara untuk melakukan web scraping pada sebuah website, seperti :
  - Menggunakan online services (API)
  - Membuat code from scratch untuk web scraping

- Website – website besar seperti Google, Twitter, Facebook, StackOverflow, dll sudah memiliki API yang dapat membantu kita untuk mengakses data mereka dalam format yang terstruktur contoh nya json, csv, database, dll



HOW WEB SCRAPERS WORK ?

1. Identifikasi website atau url target scraping
2. Pilih proxy server
3. Kirim request ke website/url tersebut
4. Identifikasi lokasi data
5. Parsing data
6. Konversi data hasil scraping ke dalam format yang diinginkan
7. Simpan data hasil scraping data ke dalam storage

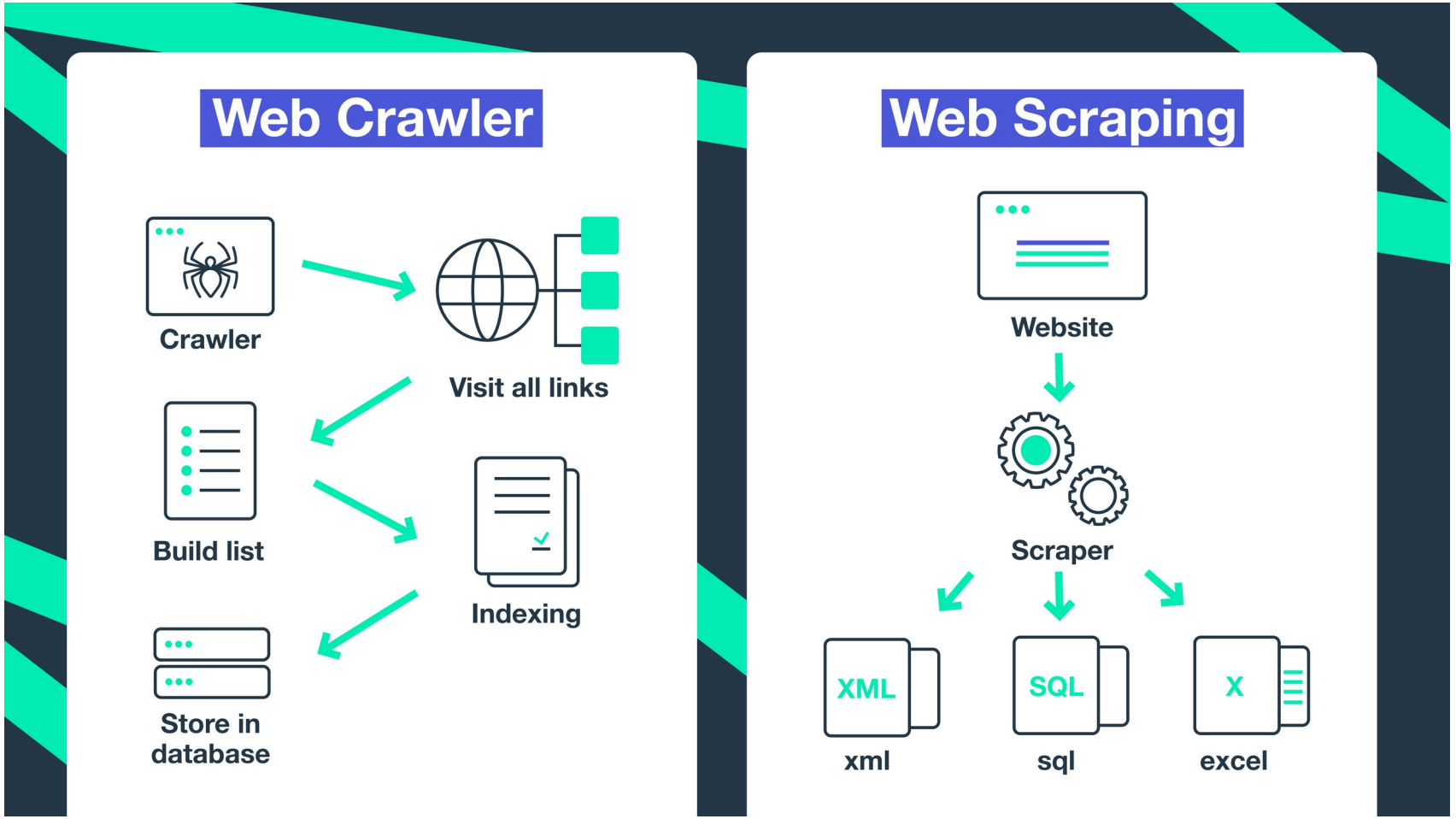
SCRAPING VS CRAWLING

AI Multiple

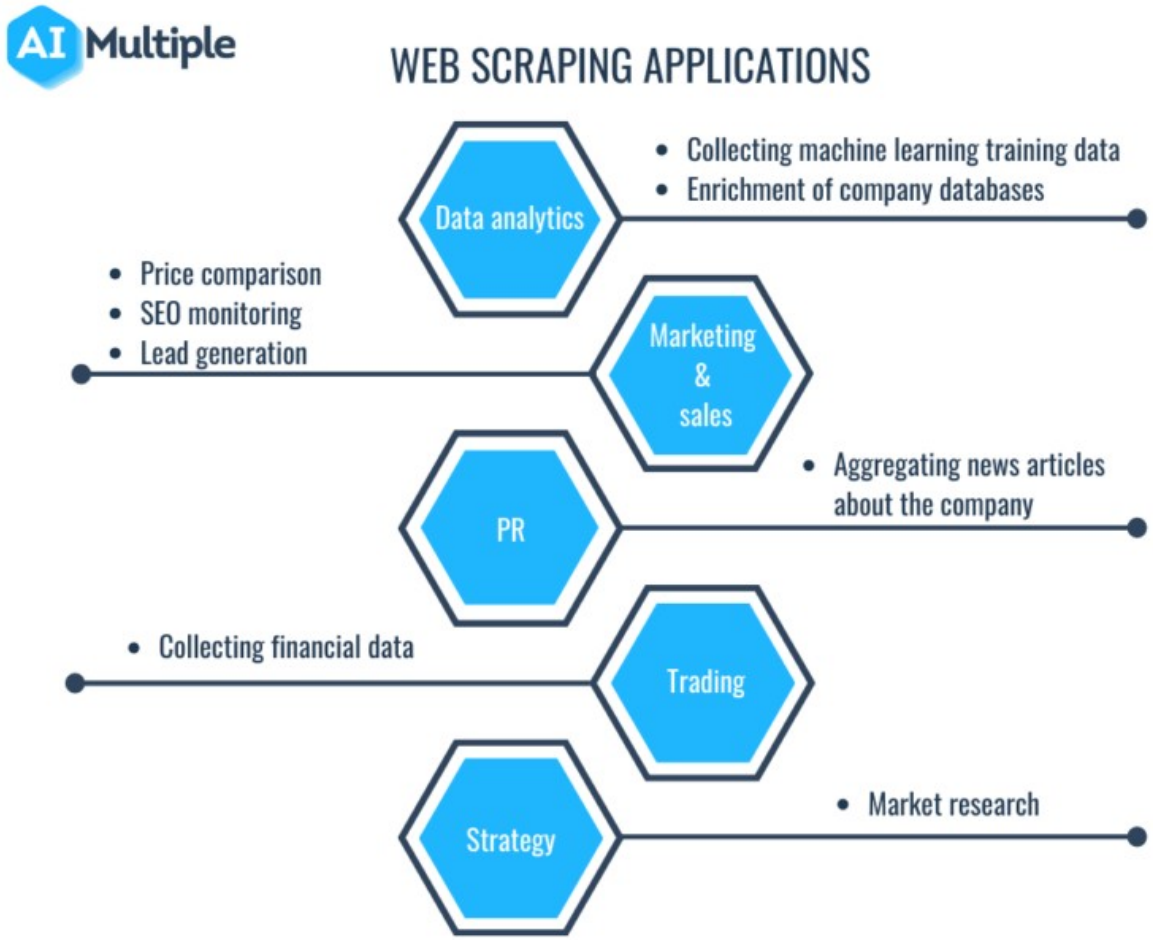
WEB CRAWLING VS WEB SCRAPING	
Web scraping	Web crawling
• Process of extracting data from multiple websites. • Using scraper. • Done on both small and large scale.	• Process of crawling and indexing all of the information on a web page. • Using crawler or spider. • Mostly used on a large scale.

- Web Scraping terdiri dari dua bagian, yaitu crawler dan scraper
  1. Crawler : Merupakan algoritma kecerdasan buatan yang menjelajahi website untuk mencari data tertentu sesuai keperluan
  2. Scraper : Merupakan alat tertentu yang dibuat untuk ekstraksi data dari situs web

SCRAPING VS CRAWLING

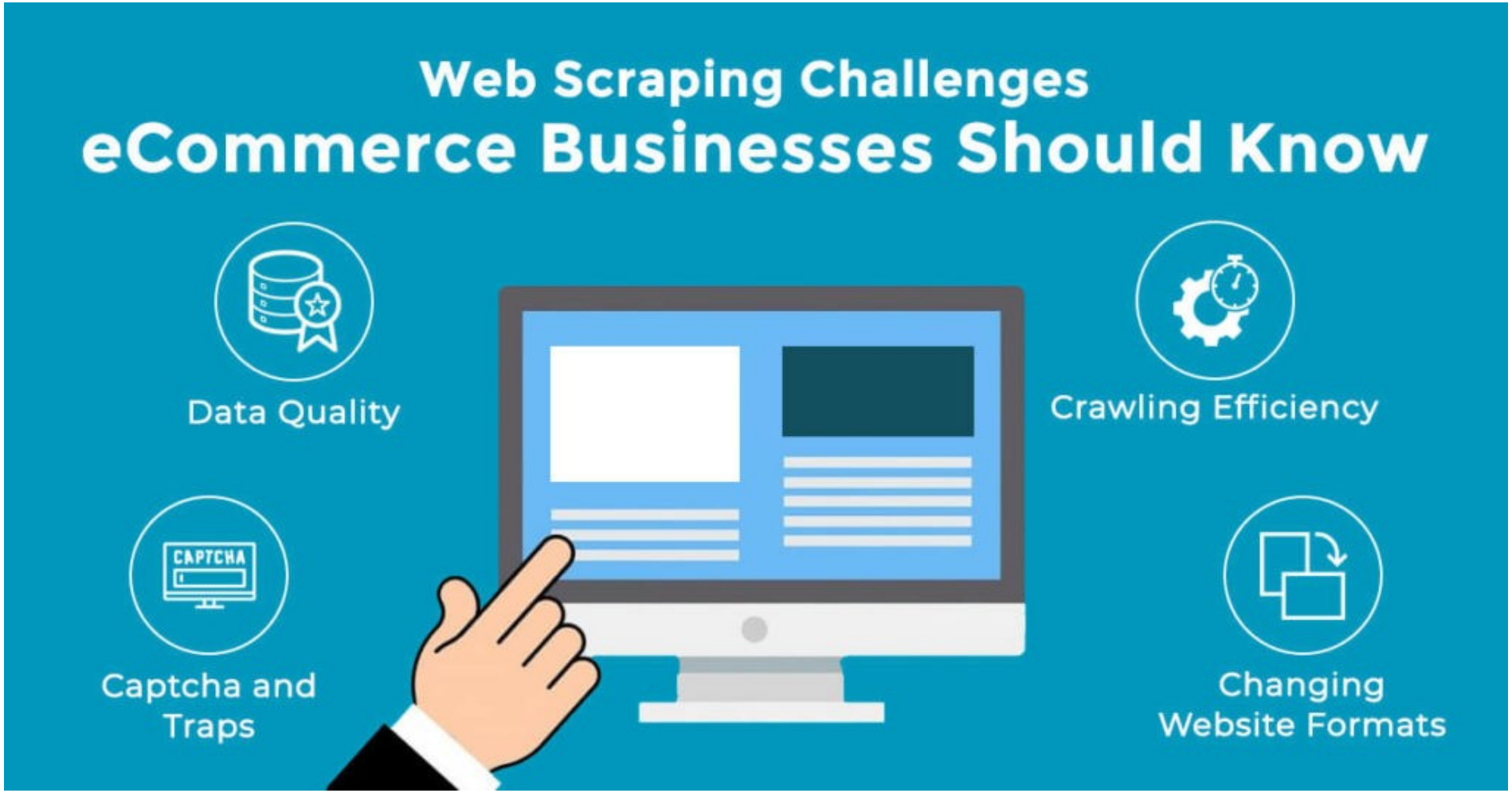


WHAT IS WEB SCRAPING USED FOR

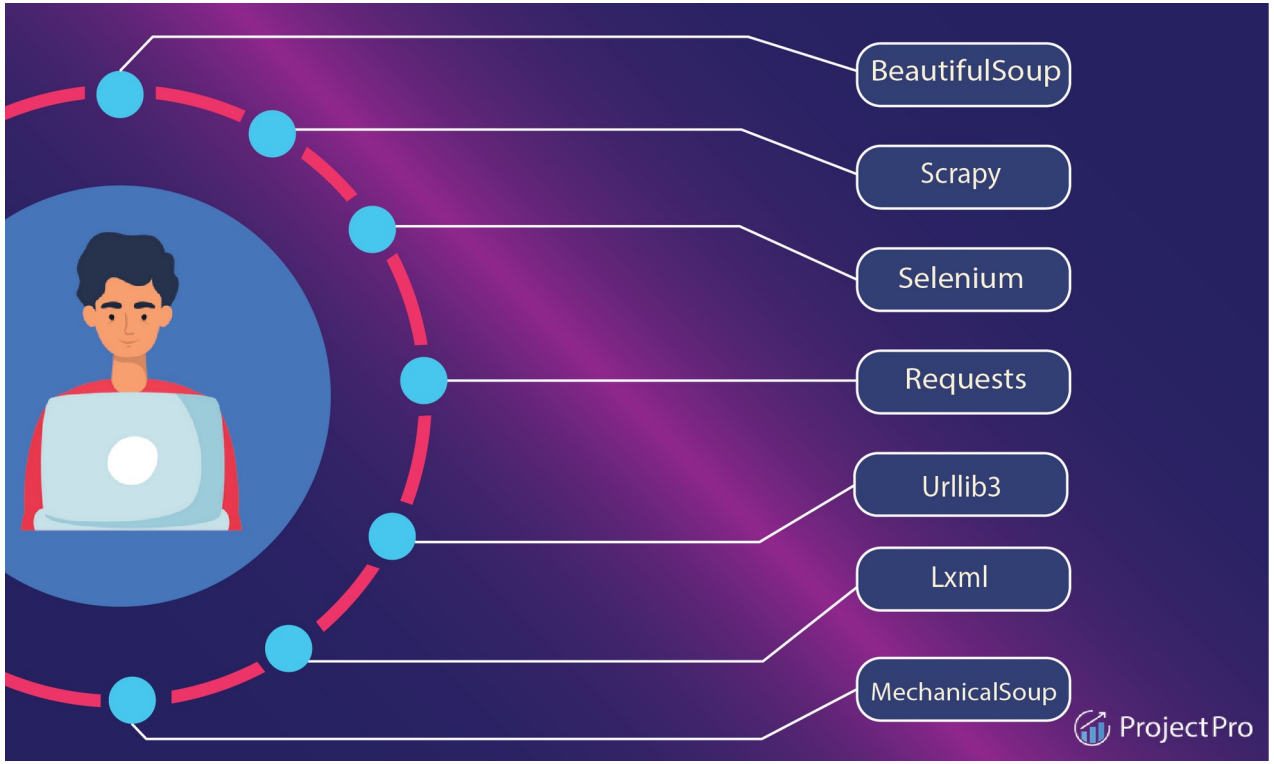


1. Data Analytics and Data Science
2. Marketing and Sales
3. Human Resources
4. Trading
5. Strategy

CHALLENGES OF WEB SCRAPING



WHY PYTHON ??



- Python memiliki banyak library yang dapat digunakan untuk membuat scraper ataupun crawler tools.
  1. Scrapy merupakan framework crawling website yang populer digunakan. Biasanya digunakan untuk mengesktrak data menggunakan API.
  2. BeautifulSoup merupakan library python yang digunakan untuk ekstraksi data dari website.

PYTHON WEB SCRAPING

- Install Request Module : “pip install requests”
- Modul Request memiliki beberapa metode bawaan untuk membuat HTTP Request ke URI tertentu menggunakan methode GET, POST, PUT, PATCH, atau HEAD.
- HTTP REQUEST berarti mengambil data dari URI tertentu atau untuk mengaupload data ke server
- Sebagai contoh, kita akan gunakan metode get, dimana kita akan mengambil informasi dari server berdasarkan URI yang dimasukkan.

```
import requests

# Making a GET request
r = requests.get('https://uns.ac.id/id/')

# check status code for response received
# success code - 200
print(r)

# print content of request
print(r.content)
```

PYTHON WEB SCRAPING

- Response Object
  - Ketika membuat permintaan ke URI, maka URI akan mengembalikan respons.
  - Respons adalaah objek yang dapat digunakan untuk membantu menormalkan
  - Misalnya response.status_code berarti mengembalikan status kode dari header itu sendiri, dan seseorang dapat memeriksa apakah permintaan itu berhasil diproses atau tidak.
  - Contoh :

```
import requests

# Making a GET request
r = requests.get('https://vokasi.uns.ac.id/')

# print request object
print(r.url)

# print status code
print(r.status_code)
```

```
https://vokasi.uns.ac.id/
200
```

BEAUTIFULSOUP

- BeautifulSoup digunakan untuk mengekstrak informasi dari file HTML dan XML.
- Instalasi :
  - pip install beautifulsoup4

BEAUTIFULSOUP

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://vokasi.uns.ac.id/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

# Getting the title tag
print(soup.title)

# Getting the name of the tag
print(soup.title.name)

# Getting the name of parent tag
print(soup.title.parent.name)

# use the child attribute to get
# the name of the child tag
```

```
<title>Beranda - Portal Vokasi UNS</title>
title
head
```

- Setelah mendapatkan HTML halaman, mari kita lihat cara mem-parsing kode HTML mentah ini menjadi beberapa informasi berguna. Pertama-tama, kita akan membuat objek BeautifulSoup dengan menentukan parser yang ingin kita gunakan.

EXTRACT DATA FROM HTML CONTENT

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://smartinotek.com/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

s = soup.find('div', class_='entry-content')
content = s.find_all('p')

print(content)
```

- Finding element by class
- Untuk mendapatkan data pada element tertentu, kita dapat menggunakan method find. Pada kasus ini, kita akan mengekstrak data pada website smartinotek.com pada kelas ‘entry-content.’
- Ekstraksi content pada class tertentu dapat dilakukan setelah kita mengetahui nama class dengan inspect element website yang akan kita ekstrak.

EXTRACT DATA FROM HTML CONTENT

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-programming-language/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

# Finding by id
s = soup.find('div', id= 'main')

# Getting the leftbar
leftbar = s.find('ul', class_='leftBarList')

# All the li under the above ul
content = leftbar.find_all('li')

print(content)
```

- Finding element by id
- Langkah pertama adalah identifikasi struktur website yang akan kita scraping.
- Kita akan mengesktrak element yang ada pada dengan id = “main”
- Element yang diekstrak terletak pada class leftBarList dengan tag ul
- Konten yang akan diekstrak berada pada tag “li”

EXTRACTING TEXT FROM TAGS

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-programming-language/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

s = soup.find('div', class_='entry-content')

lines = s.find_all('p')

for line in lines:
    print(line.text)
```

- Removing the tags from the content of the page

- Pada contoh ini, kita akan mengekstraksi teks yang berada di class “entry-content”
- Teks yang akan kita ekstrak merupakan bagian dari tag <p>

EXTRACTING TEXT FROM TAGS

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-programming-language/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

# Finding by id
s = soup.find('div', id= 'main')

# Getting the leftbar
leftbar = s.find('ul', class_='leftBarList')

# All the li under the above ul
lines = leftbar.find_all('li')

for line in lines:
    print(line.text)
```

- Removing the tags from the content of the leftbar

- Contoh berikutnya adalah kita akan mengekstrak teks dalam bentuk list

EXTRACTING TEXT FROM TAGS

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-programming-language/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

# find all the anchor tags with "href"
for link in soup.find_all('a'):
    print(link.get('href'))
```

- Python BeautifulSoup Extracting Links

EXTRACTING IMAGE INFORMATION

```
import requests
from bs4 import BeautifulSoup


# Making a GET request
r = requests.get('https://www.geeksforgeeks.org/python-programming-language/')

# Parsing the HTML
soup = BeautifulSoup(r.content, 'html.parser')

images_list = []

images = soup.select('img')
for image in images:
    src = image.get('src')
    alt = image.get('alt')
    images_list.append({"src": src, "alt": alt})

for image in images_list:
    print(image)
```

- Biasanya data dalam bentuk gambar terletak pada tag img.
- Sedangkan alamat gambar terletak pada atribut src.

SCRAPING MULTIPLE PAGES

- Terdapat dua cara untuk mengekstraksi data pada website yang memiliki lebih dari 1 halaman.
- Cara pertama adalah menulis dengan membuat kode yang ditujukan untuk ekstraksi alamat web yang sama (looping)
- Cara kedua adalah dengan kode yang ditujukan untuk ekstraksi pada URL situs di halaman lain

SCRAPING MULTIPLE PAGES

```
from bs4 import BeautifulSoup
import requests

start_url = "https://proxyway.com/guides"

def scrape_page(url):
    print ("URL: " + url)
    r = requests.get(url)
    soup = BeautifulSoup(r.content, "html.parser")
    get_data(soup)
    next_page_link = soup.find("a", class_="next")
    if next_page_link is not None:
        href = next_page_link.get("href")
        scrape_page(href)
    else:
        print ("Done")


def get_data(content):
    #we could do some scraping of web content here
    pass

def main():
    scrape_page(start_url)


if __name__ == "__main__":
    main()
```

- Kode disamping merupakan contoh penggunaan BeautifulSoup untuk scraping pada lebih dari 1 halaman website. Kode diatas me-retrieve semua URL yang ada pada web tersebut

SCRAPING MULTIPLE PAGES

```
import requests
from bs4 import BeautifulSoup as bs

URL = 'https://proxyway.com/guides/page/2'

req = requests.get(URL)
soup = bs(req.text, 'html.parser')

titles = soup.find_all('h3',attrs = {'class','archive-list__title'})

print(titles)

print(titles[1].text)
```

- Kode program diasamping merupakan bentuk implementasi BeautifulSoup untuk menambang data pada suatu page tertentu dalam suatu website

SCRAPING MULTIPLE PAGES

```
import requests
from bs4 import BeautifulSoup as bs

URL = 'https://proxyway.com/guides/page/'

for page in range(1, 9):
    print("\n")
    print("Sub Titles Page : ", page, "\n")

    req = requests.get(URL + str(page) + '/')
    soup = bs(req.text, 'html.parser')

    titles = soup.find_all('h3', attrs={'class', 'archive-list__title'})

    if page == 9 :
        for i in range(0, 10):
            print(f"{i+1} " + titles[i].text)
    else :
        for i in range(0, 12):
            print(f"{i+1} " + titles[i].text)
```

- Kode program diasamping merupakan bentuk implementasi BeautifulSoup untuk menambang data pada banyak halaman pada sebuah website

SAVE DATA TO CSV

```python
import requests
from bs4 import BeautifulSoup as bs
import csv

URL = 'https://proxyway.com/guides/page/'

titles_list = []
title_page9 = []

for page in range(1,10):
    req = requests.get(URL + str(page) + '/')
    soup = bs(req.text, 'html.parser')

    titles = soup.find_all('h3', attrs={'class', 'archive-list__title'})

    count = 1
    for title in titles:
        d = {}
        d['Page Number'] = f'Page {page}'
        d['Title Number'] = f'Title {count}'
        d['Title Name'] = title.text
        count += 1
        titles_list.append(d)
```

```python
    if page == 9 :
        count = 1
        for title in titles:
            e = {}
            e['Page Number'] = f'Page {page}'
            e['Title Number'] = f'Title {count}'
            e['Title Name'] = title.text
            count+=1
            title_page9.append(e)


titles_all = titles_list + title_page9

#print(title_page9)

filename = 'titles.csv'
with open(filename, 'w', newline='') as f:
    w = csv.DictWriter(f,['Page Number','Title Number','Title Name'])
    w.writeheader()
    
    w.writerows(titles_all)
```

TUGAS LATIHAN SCRAPING WEBSITE

- Silahkan scraping data di website https://proxyway.com/reviews atau https://proxyway.com/news
- Ambil data judul/title di setiap halaman
- Simpan data ke dalam format CSV