

Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 3





مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY

Basics of Network Programming – Part II

Lecture 3

Prepared by
Dr. Hamada I. AbdulWakel

www.midocean.ae

Lecture

03

RECAP: LECTURE 2

- Service Oriented Architecture (SOA).
- Application Program Interface (API).
- API modules in Python.

Lecture

03

LECTURE MAIN POINTS

- Application Program Interface (API).
- BeautifulSoup and web scraping.
- Socket Programming in Python.

GOOGLE MAPS API

- **Geocoding**: the process of converting human-readable addresses into geographic coordinates (i.e., latitude and longitude), which you can use to place markers on a map.
- **Reverse geocoding**: the process of converting geographic coordinates into a human-readable address.
- Map APIs allow you to create location-aware infographics, mind maps and visualization aids.

Lecture

03

GOOGLE MAPS API

Cont'd

Enter a query to geocode, or click on the map to reverse geocode.

[Show options](#)

The request did not encounter any errors (OK).

Found 1 results. ([view all in map](#)) [[JSON](#)] [[XML](#)] [[Collapse](#)]

Address: Mecca Saudi Arabia

Location: 21.389082,39.857912 (type: APPROXIMATE)

Types: locality, political

Address components:

locality	Mecca	political
	Makkah	
administrative_area_level_2	Makkah Principality	political
administrative_area_level_1	Makkah Province	political
country	Saudi Arabia	political
	SA	

Bounds: 21.279028,39.690235 – 21.593003,40.002858

Viewport: 21.279028,39.690235 – 21.593003,40.002858

Place ID: [ChIJdYeB7UwbwhURzslwz2kkq5g](#)

Map Satellite

Mecca مكة المكرمة

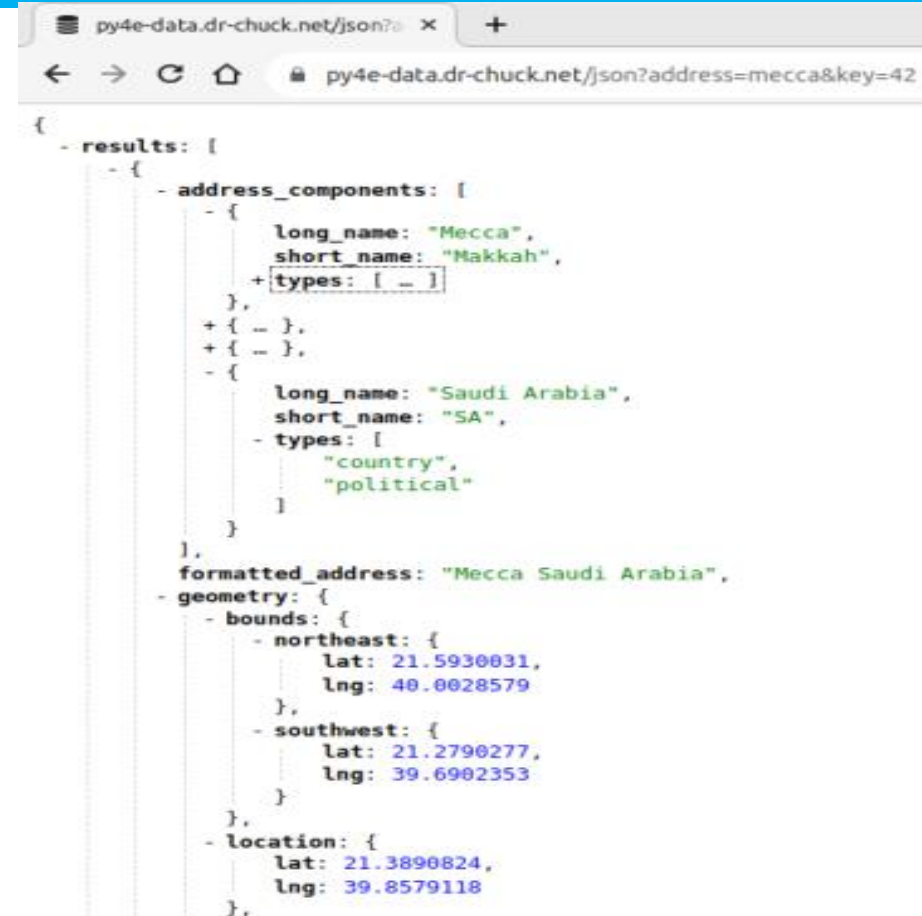
Google

Keyboard shortcuts Map data ©2023 Google 10 km Terms

GOOGLE MAPS API

Cont'd

- Google map API simulation.
 - URL:
<https://py4e-data.dr-chuck.net/json?address=mecca&key=42>



```
{
  - results: [
    - {
      - address_components: [
        - {
          long_name: "Mecca",
          short_name: "Makkah",
          + types: [ - ]
        },
        + { - },
        + { - },
        - {
          long_name: "Saudi Arabia",
          short_name: "SA",
          - types: [
            "country",
            "political"
          ]
        }
      ],
      formatted_address: "Mecca Saudi Arabia",
      - geometry: {
        - bounds: {
          - northeast: {
            lat: 21.5930031,
            lng: 40.0028579
          },
          - southwest: {
            lat: 21.2790277,
            lng: 39.6902353
          }
        },
        - location: {
          lat: 21.3890824,
          lng: 39.8579118
        }
      }
    }
  ],
}
```

GOOGLE MAPS API

Cont'd

- Code:

```
import urllib.error, urllib.parse, urllib.request
import json
api_key = False
if api_key is False:
    api_key = 42
    serviceurl = 'http://py4e-data.dr-chuck.net/json?'
else:
    serviceurl='https://maps.googleapis.com/maps/api/geocode/json?'
```

GOOGLE MAPS API

Cont'd

- Code:

```
while True:
    address = input("Enter location: ")
    if len(address) < 1:
        break
    parms = dict()
    parms['address'] = address
    if api_key is not False:
        parms['key'] = api_key
```

GOOGLE MAPS API

Cont'd

- Code (inside while loop):

```
url = serviceurl + urllib.parse.urlencode(parms)
print('Retrieving', url)
urllib_handler = urllib.request.urlopen(url)
data = urllib_handler.read().decode()
print('Retrieved', len(data), 'characters')
try:
    js = json.loads(data)
except:
    js = None
```

GOOGLE MAPS API

Cont'd

- Code (inside while loop):

```
if 'status' not in js or js['status'] != 'OK':  
    print('Fail to retrieve data')  
    print(data)  
    continue  
print(json.dumps(js, indent=4))  
lat = js['results'][0]['geometry']['location']['lat']  
lng = js['results'][0]['geometry']['location']['lng']  
location = js['results'][0]['formatted_address']  
print('lat', lat, 'lng', lng, "location", location)
```

Lecture

03

WEB SCRAPING

- When a program pretends to be a web browser and retrieves web pages, extracts information, and then looks at more pages.
- Uses:
 - Monitor a site for new information.
 - Make a database for a search engine.
 - Get your data from a system that has no export feature.
 - Pull social data.

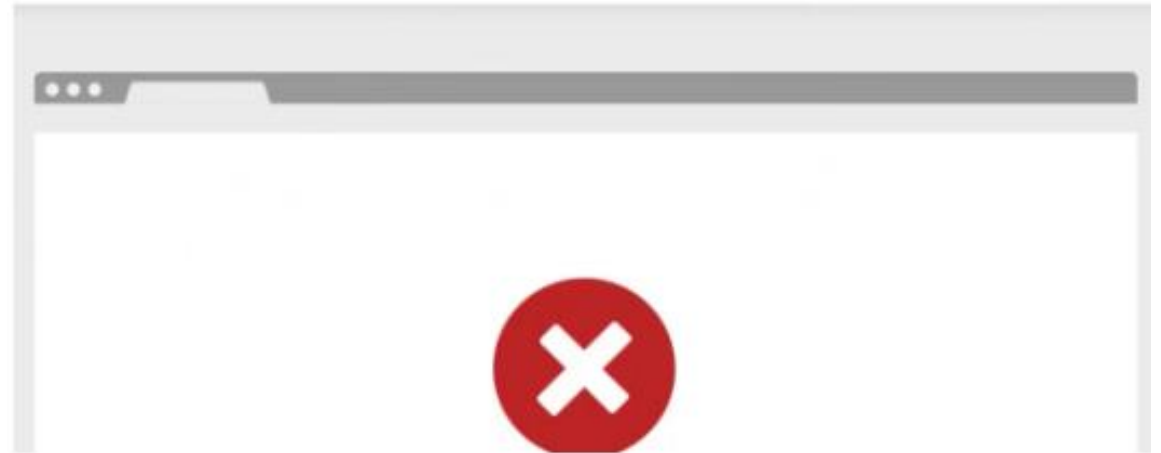
Lecture

03

BE CAREFUL WITH WEB SCRAPING

- There is some controversy about web scraping.
- Do not republishing copyrighted material.
- Do not violate terms of service.

Sorry, you have been blocked
You are unable to access youm7.com

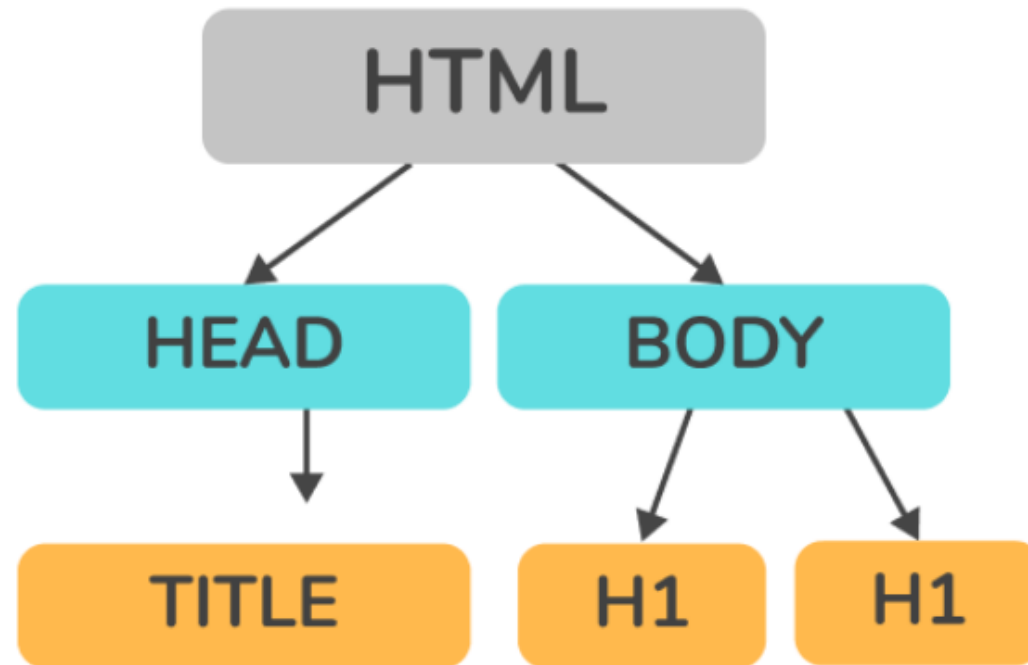


Lecture

03

BEAUTIFULSOUP MODULE

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
  </head>
  <body>
    <h1></h1>
    <h1></h1>
  </body>
</html>
```



BEAUTIFULSOUP MODULE

Cont'd

- **BeautifulSoup** is a Python module for pulling data out of HTML and XML files.
- One of the important pieces of HTML document are **tags**, which may contain other tags/strings (tag's children).
- BeautifulSoup provides different ways to navigate and iterate over these tags.
- BeautifulSoup is the object created when we try to **scrape a web resource**.

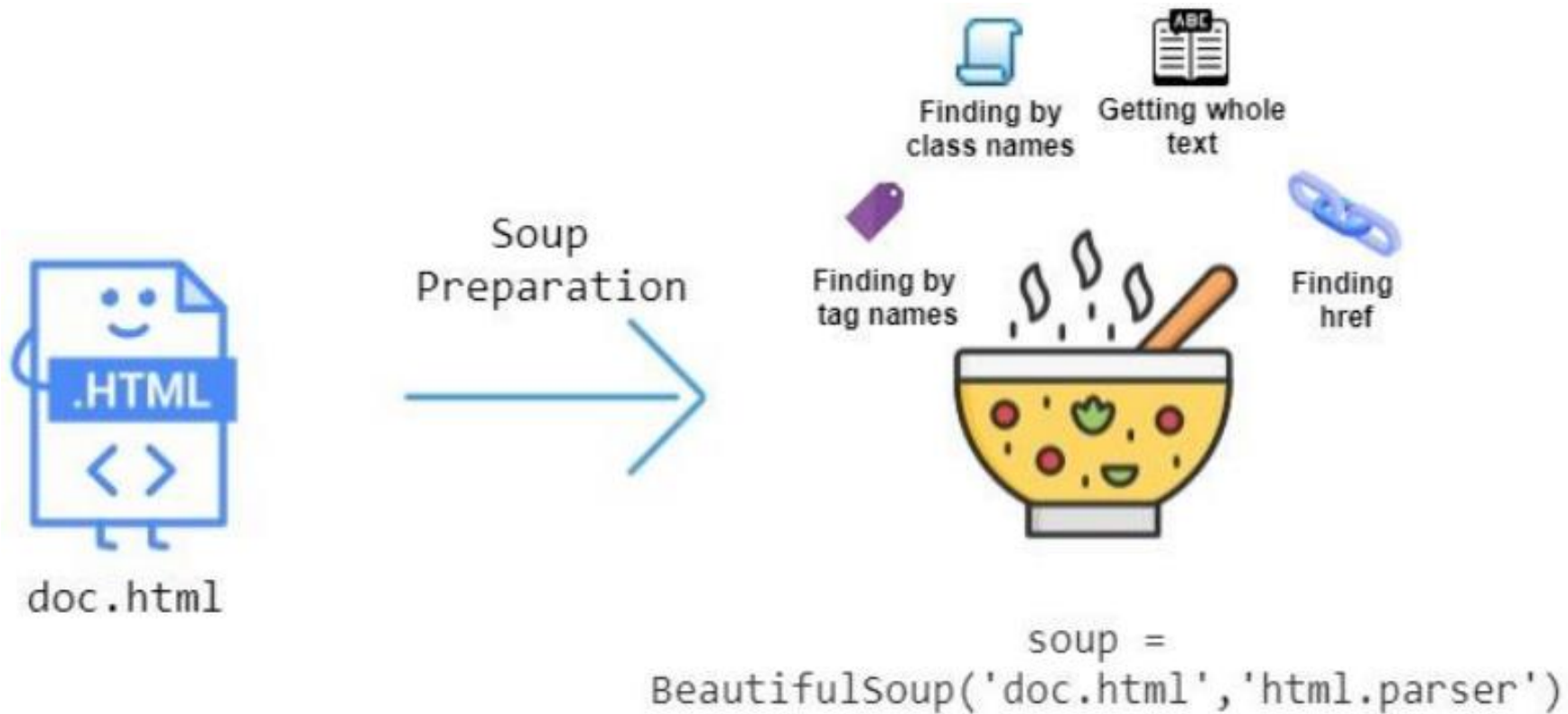
INSTALLATION OF BEAUTIFULSOUP MODULE

- Installation:
 - PyPi: `pip install beautifulsoup4`
- Beautiful Soup supports different HTML parser such as:
 - **Html.parser**: the default HTML parser included in Python's standard library.
 - lxml
 - `pip install lxml`
 - Html5lib
 - `pip install html5lib`

Lecture

03

WEB SCRAPING STEPS



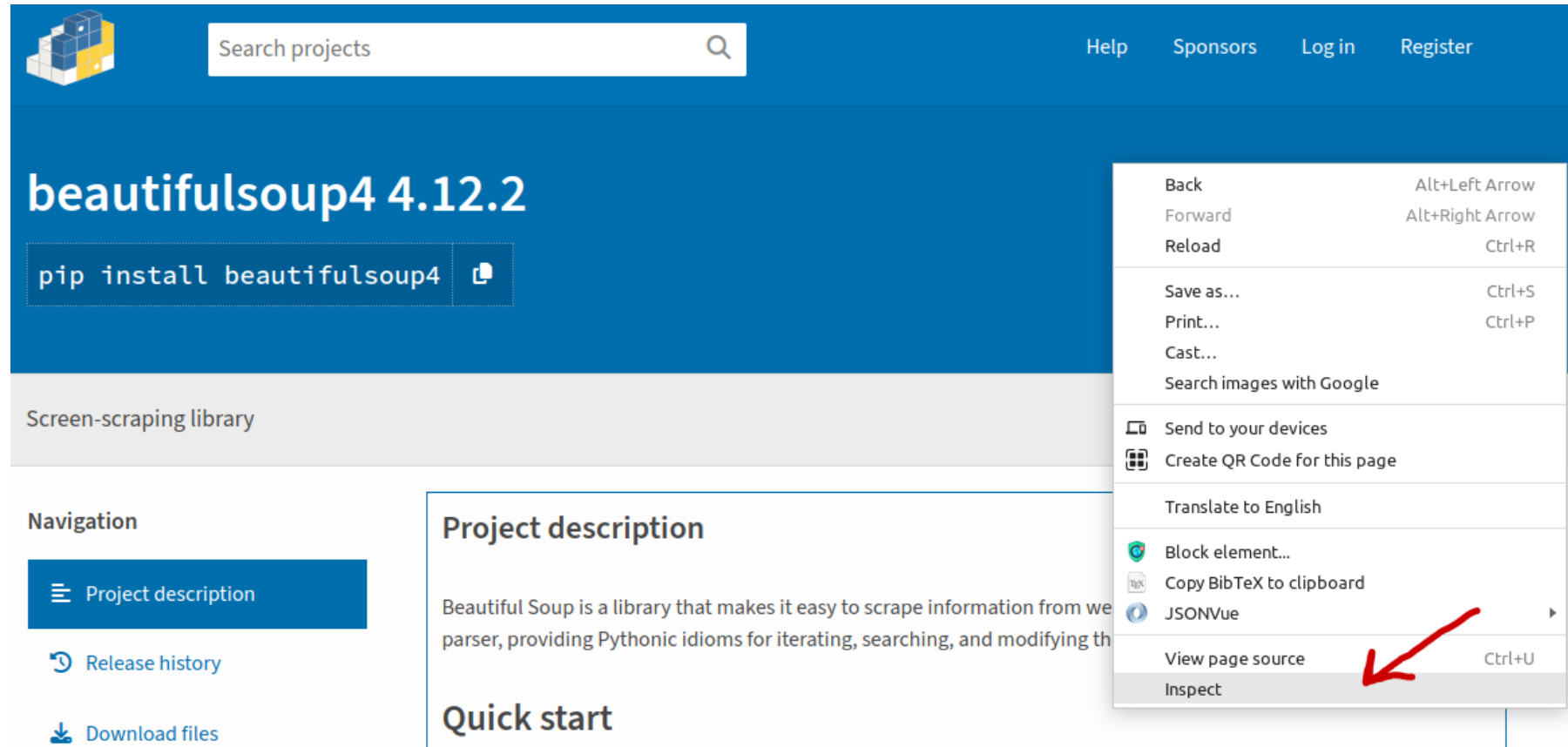
WEB SCRAPING STEPS

Cont'd

- **Step 1: Inspect your data source:**
 - A manual step.
 - Open the website you want to pull important content from it.
 - Open inspection.
 - On Mac:
`Cmd+Alt+I`
 - On Windows/Linux:
`Ctrl+Shift+I`
 - Understand the website structure to know where these important data.

WEB SCRAPING STEPS

Cont'd



The screenshot shows the PyPI page for BeautifulSoup4 4.12.2. The page has a blue header with a search bar and navigation links (Help, Sponsors, Log in, Register). The main content area is blue and features the package name and version, a code block for installation, and a 'Screen-scraping library' badge. A left sidebar contains a 'Navigation' section with links to 'Project description', 'Release history', and 'Download files'. The main content area also has a 'Project description' section and a 'Quick start' section. A browser context menu is open on the right side of the page, with a red arrow pointing to the 'Inspect' option at the bottom.

Search projects

Help Sponsors Log in Register

beautifulsoup4 4.12.2

```
pip install beautifulsoup4
```

Screen-scraping library

Navigation

- Project description
- Release history
- Download files

Project description

Beautiful Soup is a library that makes it easy to scrape information from web pages, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

Quick start

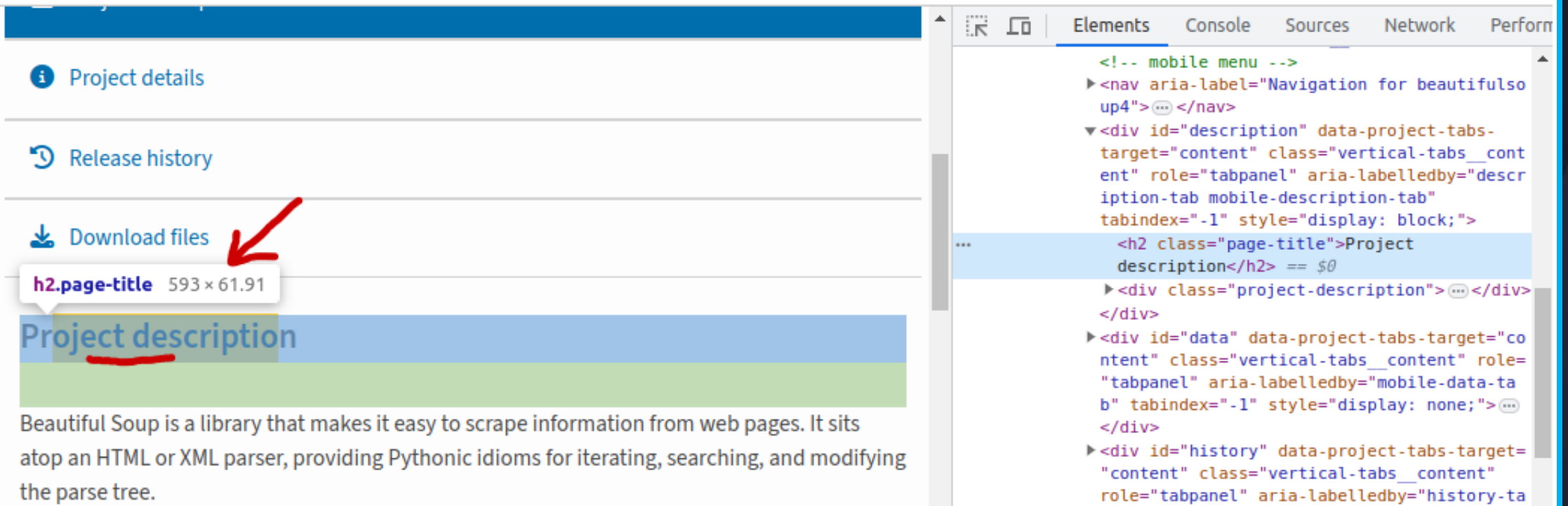
- Back (Alt+Left Arrow)
- Forward (Alt+Right Arrow)
- Reload (Ctrl+R)
- Save as... (Ctrl+S)
- Print... (Ctrl+P)
- Cast...
- Search images with Google
- Send to your devices
- Create QR Code for this page
- Translate to English
- Block element...
- Copy BibTeX to clipboard
- JSONVue
- View page source (Ctrl+U)
- Inspect

Lecture

03

WEB SCRAPING STEPS

Cont'd



Project details

Release history

Download files

h2.page-title 593 x 61.91

Project description

Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

```

<!-- mobile menu -->
<nav aria-label="Navigation for beautifulso
up4">...</nav>
<div id="description" data-project-tabs-
target="content" class="vertical-tabs__cont
ent" role="tabpanel" aria-labelledby="descr
iption-tab mobile-description-tab"
tabindex="-1" style="display: block;">
...
<h2 class="page-title">Project
description</h2> == $0
<div class="project-description">...</div>
</div>
<div id="data" data-project-tabs-target="co
ntent" class="vertical-tabs__content" role=
"tabpanel" aria-labelledby="mobile-data-ta
b" tabindex="-1" style="display: none;">...
</div>
<div id="history" data-project-tabs-target=
"content" class="vertical-tabs__content"
role="tabpanel" aria-labelledby="history-ta

```

WEB SCRAPING STEPS

Cont'd

- **Step 2: Scrape HTML content from a page:**
 - Create the soup object.
 - Read the web page content using this object.
- **Step 3: Parse HTML code with BeautifulSoup:**
 - Parse the web page content to make it more accessible.
 - Pick out the data from the required HTML tags.

BEAUTIFULSOUP MODULE METHODS

- **Beautifulsoup(content, parser):** creates a soup object.
- **soupObj.prettify():** gives the visual representation of the parsed URL. It arranges all the tags in a parse-tree manner with better readability.
- **soupObj.title:** gives the title tag with its content.
- **soupObj.title.string:** gives the title content only without the tag.

BEAUTIFULSOUP MODULE METHODS

Cont'd

- `soupObj.find("tagName")`: gives the first occurrence of the given tag name.
- `soupObj.find_all("tagName")`: gives a list of all occurrences of the given tag name with they contents.
 - `firstTag = soupObj.find_all("tagName")[0]`
 - If the firstTag contains other tags:
`firstTag.find_all("otherTagName")`

BEAUTIFULSOUP MODULE METHODS

Cont'd

- **soupObj.find_all("tagName", { css style })**: gives a list of all occurrences of the given tag name with their contents which have the same given css style.
- **tag.get("propertyName")**: gives the value of the given property for the selected tag.
- **tag.text**: gives the content of the selected tag.

Lecture

03

BEAUTIFULSOUP EXAMPLE

- Code:

```
import requests
from bs4 import BeautifulSoup
url = 'https://www.demoblaze.com/'
html = requests.get(url)
soup = BeautifulSoup(html.content, 'html.parser')
print(soup.prettify())
print(soup.title)
print(soup.title.string)
```

BEAUTIFULSOUP EXAMPLE

Cont'd

- Code:

```
print( soup.find("p") )  
tags = soup('a')           #get all anchor tags as a ResultSet  
tags = soup.find_all("a")  #the same as above  
for tag in tags:  
    print( tag.get('href') )
```

WEB SCRAPPING TO SEARCH FOR A JOB EXAMPLE

- Using Wuzzuf website:

```
from itertools import zip_longest
import requests
from bs4 import BeautifulSoup
import csv
```

```
result = requests.get('https://wuzzuf.net/search/jobs/?q=python&a=hpb')
soup = BeautifulSoup(result.content, 'html.parser')
job_titles = soup.find_all("h2", {"class":"css-m604qf"})
```

WEB SCRAPPING TO SEARCH FOR A JOB EXAMPLE

Cont'd

- Using Wuzzuf website:

```
company_names = soup.find_all("a", {"class":"css-17s97q8"})
locations = soup.find_all("span", {"class":"css-5wys0k"})
skills = soup.find_all("div", {"class":"css-y4udm8"})
print(len(job_titles))
jobs = []
comapnies = []
locs = []
skill = []
```

WEB SCRAPPING TO SEARCH FOR A JOB EXAMPLE

Cont'd

- Using Wuzzuf website:
 for i in range(len(job_titles)):
 jobs.append(job_titles[i].text)
 comapnies.append(company_names[i].text)
 locs.append(locations[i].text)
 skill.append(skills[i].text)

WEB SCRAPPING TO SEARCH FOR A JOB EXAMPLE

Cont'd

- Using Wuzzuf website:

```
fileLists = [jobs, comapnies, locs, skill]
```

```
exported = zip_longest(*fileLists)
```

```
with open("/path/to/create/jobsResults.csv", "w") as f:
```

```
    wr = csv.writer(f)
```

```
    wr.writerow(["job title", "company name", "location", "skills"])
```

```
    wr.writerows(exported)
```

SOCKETS

- **Sockets** are the endpoints of a bidirectional communications channel between connection devices.
- Sockets are the most popular form of inter-process communication especially for cross-platform communication.
- Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

THE SOCKET MODULE

- Python's standard library consists of various built-in modules that support inter-process communication and networking.
- The **socket** module defines how server and client machines can communicate at hardware level using socket endpoints on top of the operating system.
- The socket module provides various objects, constants, functions and related exceptions for building full-fledged network applications including client and server programs.

THE SOCKET MODULE

Cont'd

- To program a connection between two end-points, you need to configure both end-points sockets to know each other.
- The socket configuration includes:
 - Socket family: represents the IP version used.
 - **AF_INET** for IPv4
 - **AF_UNIX** for IPv6
 - Socket type: represents the type of connection.
 - **SOCK_STREAM** for connection-oriented type (TCP).
 - **SOCK_DGRAM** for connectionless type (UDP).

Lecture

03

THE SOCKET MODULE METHODS

- **socket():** creates sockets objects for the server and the client.
- **gethostname():** returns a string containing the hostname.
- **gethostbyname():** returns IP address.
- **bind():** binds/associates the address to the created socket object.

THE SOCKET MODULE METHODS

Cont'd

- **connect():** takes a two-item tuple object as argument. The two items are IP address and port number of the server.
- **listen():** starts the server and runs it into a listen loop looking for connection request from client.
- **accept():** when a connection request is intercepted by server, this method accepts it and identifies the client socket with its address.
- **close():** marks the socket as closed.

Lecture

03

THE SOCKET CREATION

- Code:

```
import socket
```

```
# create a TCP socket
```

```
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
print("Socket succesfully created!")
```

Lecture

03

THE SOCKET CREATION WITH EXCEPTION HANDLING

- Code:

```
import socket
target_host = input("Enter the target host name to connect: ")
target_port = input("Enter the target port: ")
try:
    sock.connect((target_host, int(target_port)))
    print("Socket Connected to", target_host, "on port:", target_port)
except socket.error as err:
    print("Failed to connect to", target_host, "on port", target_port)
    print("Reason:", str(err) )
```

Lecture

03

SIMPLE SERVER USING SOCKETS

- Code:

```
import socket
```

```
soc = socket.socket()  
print("socket created")
```

```
# the server will be in the local machine  
soc.bind((socket.gethostname(), 5485))  
soc.listen(3) #accepts three connection  
print("Waiting for connections .....")
```

QUESTIONS FOR ONLINE DISCUSSION

- Write a program and use this link <https://we-xpats.com/en/job/me/sa/search/?area=3411> to create a web scraping application to get the available jobs in Makkah.



مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY
BY EVERYONE'S SMART UNIVERSITY



Thank You

