Web Scraping with Python

Rhett Sessions, Anthony Ayala, Chris Shen

Agenda

1. Overview of Web Scraping?
2. Three Practical Uses of Web Scraping
3. Impact of Web Scraping
4. Regulation & Compliance

What is Web Scraping?

- Automatic method to extract content and/or data from a website by extracting HTML code and replicating it elsewhere in a more structured format
- Use cases include:
  - Search engine bots ranking a website after scraping and analyzing its content
  - Auto-fetching prices and product descriptions to compare prices
  - Market research companies pulling data from forums, social media for sentiment analysis
- Python the most popular language for web scraping

Use 1: Web Scraping Weather Data

- Resource we used: Web Scraping Weather Data with Python by John Watson Rooney
- **Requests** are HTTP calls made to a web server to retrieve specific content from a website.
- **Sessions** maintain stateful information between multiple HTTP requests sent to the same server, useful for tasks like maintaining authentication across requests in web scraping.
- The big idea here is that you are just navigating/parsing HTML and trying to find the location of the data you want to collect and request it.

```
temp = (r.html.find('span#wob_tm', first=True).text)
unit = (r.html.find('div.vk_bk.wob-unit span.wob_t', first=True).text)
description = (r.html.find('div.VQF4g', first=True).find('span#wob_dc', first=True).text)
print(f"The Weather right now in {query} is at {temp}{unit} and it is {description}")
```

```
The Weather right now in Barcelona is at 57°F and it is Mostly sunny
```

Use 1: Example

Use 2: Controlling your browser experience and logging in with code

- Resource we used: Python Selenium Tutorial #1 - Web Scraping, Bots & Testing

- **Selenium** is a popular web scraping package that allows you to web scrape more effectively than HTML by not limiting you to parse HTML and allows you to control your browser experience with code.

Opening a Web Page with Selenium using the Microsoft Edge

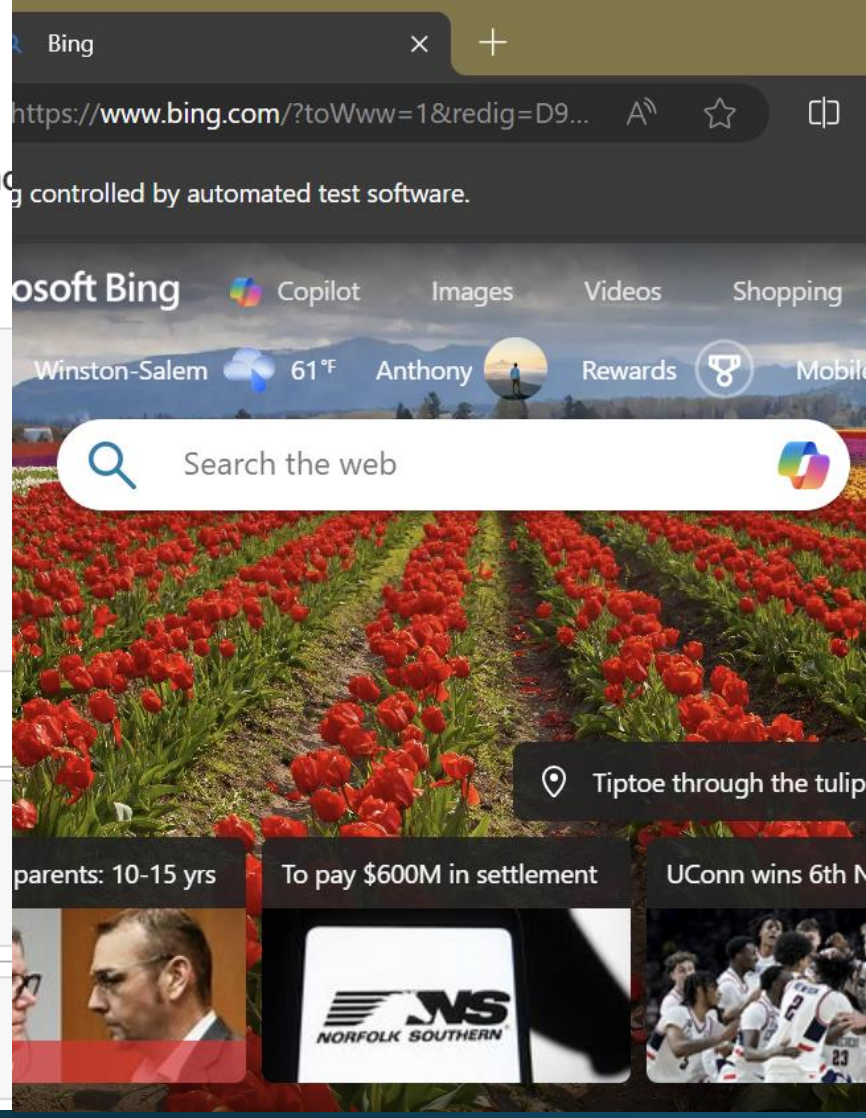
- Video Guide Link: INSERT TECH WITH TIM VIDEO PYTHON SELENIUM TUTORIAL #1
- Step one is to open a web page and then close the web page with the driver.close (closes a tab) and
- Step two is now to start locating elements from HTML

```
# More simple version of the code
driver = webdriver.Edge()

driver.get('https://bing.com')
print(driver.title)

time.sleep(5)
```

```
Bing
```

```
# Maximize the window and refresh the page
driver.refresh()
driver.fullscreen_window()
```

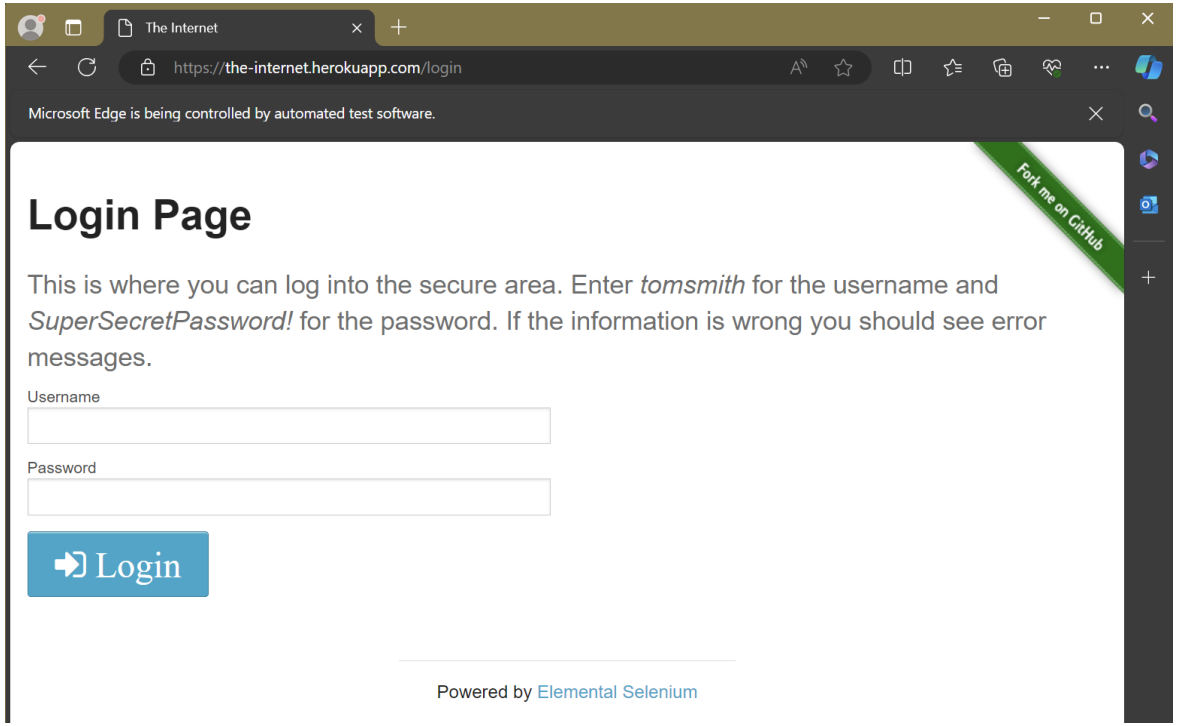
```
# Close the entire window
driver.quit()
```



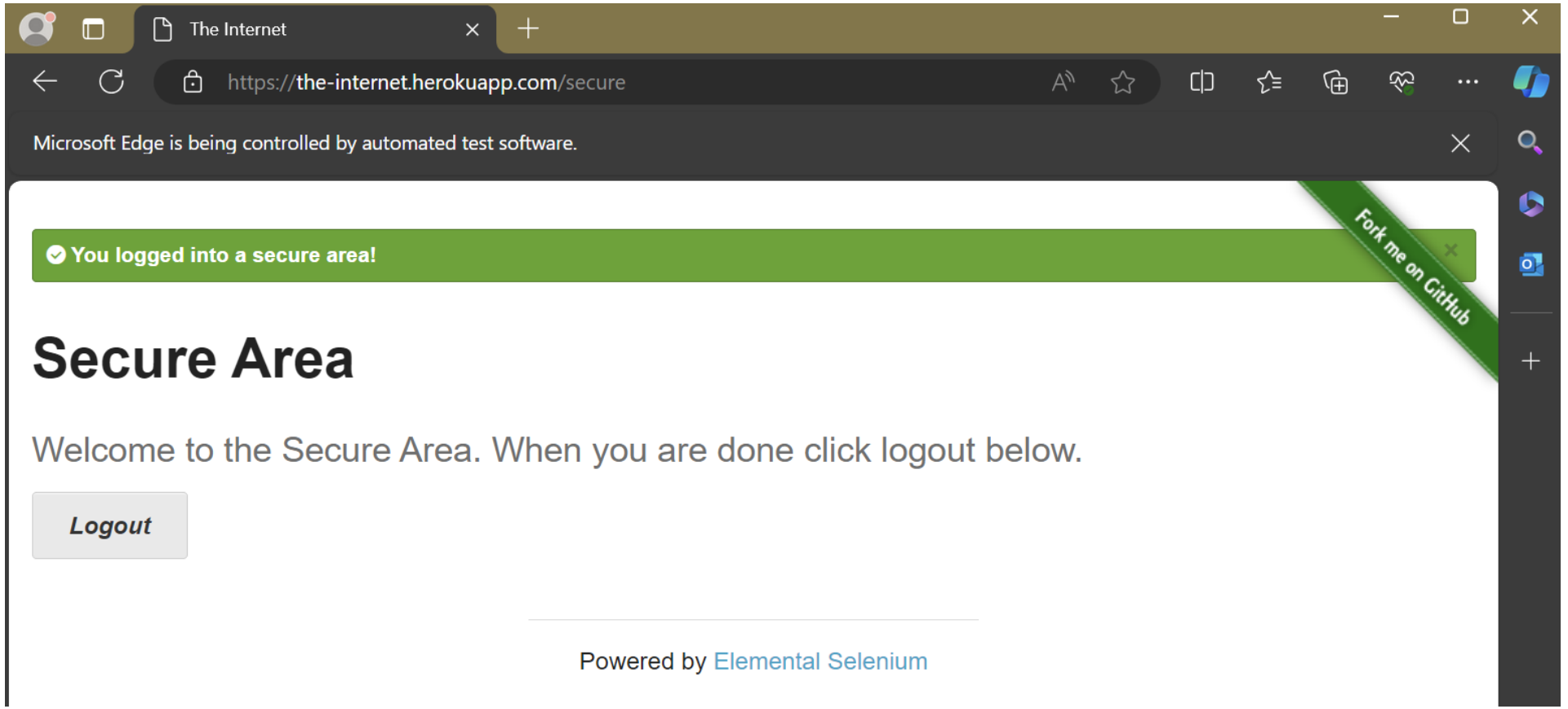
Use 2: Example

Use 2: Logging In

```
# The page asks for a username and a password and we must copy the XPath of the username slot and password slot
# This identifies where username and password should be entered, which will be done by the code here
# Then we need to click on the login in button so that our credentials pass and we can further proceed

# Maximize the window and refresh the page;
driver.refresh()

# Username XPath: //*[@id="username"]
# Password XPath: //*[@id="password"]
# Login Button XPath: //*[@id="login"]/button
# Edge's driver function works as driver.find_element(s)(By.XPATH,.....)
driver.find_element(By.XPATH,'//*[@id="username"]').send_keys('tomsmith')
driver.find_element(By.XPATH,'//*[@id="password"]').send_keys('SuperSecretPassword!')
driver.find_element(By.XPATH,'//*[@id="login"]/button').click()
```



Use 2: Successful Log In



Use 3: Data Extraction with Selenium and Pandas

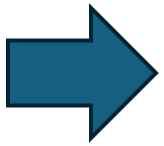
- Resource we used: How to SCRAPE DYNAMIC websites with Selenium

- **Selenium** is a popular web scraping package that allows you to web scrape more effectively than HTML by not limiting you to parse HTML and allows you to control your browser experience with code.

Use 3: YouTuber Data

```
# Beginning of for loop by Class
# First Video (Class Name): 'style-scope ytd-rich-item-renderer'
# Title (XPATH): '//*[@id="video-title-link"]'
# Views (XPATH): '//*[@id="metadata-line"]/span[1]'
# Upload date (XPATH): '//*[@id="metadata-line"]/span[2]'

videos = driver.find_elements(By.CLASS_NAME, 'style-scope ytd-rich-item-renderer')

for video in videos:
    title = video.find_element(By.XPATH, './/*[@id="video-title-link"]').text
    views = video.find_element(By.XPATH, './/*[@id="metadata-line"]/span[1]').text
    when = video.find_element(By.XPATH, './/*[@id="metadata-line"]/span[2]').text
    print(title, views, when)
```

```
Always Check for the Hidden API when Web Scraping 549K views 2 years ago
Scrapy for Beginners - A Complete How To Example Web Scraping Project 239K views 3 years ago
How to SCRAPE DYNAMIC websites with Selenium 150K views 3 years ago
Web Scraping with Python: Ecommerce Product Pages. In Depth including troubleshooting 133K views 3 years ago
Web Scrape Websites with a LOGIN - Python Basic Auth 110K views 3 years ago
How to Rotate Proxies with Python 109K views 3 years ago
Python CSV files - with PANDAS 87K views 3 years ago
Indeed Jobs Web Scraping Save to CSV 83K views 3 years ago
How I Scrape multiple pages on Amazon with Python, Requests & BeautifulSoup 83K views 3 years ago
Automate Excel Work with Python and Pandas 80K views 2 years ago
The Biggest Mistake Beginners Make When Web Scraping 77K views 1 year ago
Working With APIs in Python - Pagination and Data Extraction 76K views 2 years ago
Login and Scrape Data with Playwright and Python 76K views 2 years ago
How to Scrape and Download ALL images from a webpage with Python 75K views 3 years ago
How to Make 2500 HTTP Requests in 2 Seconds with Async & Await 71K views 1 year ago
I Don't Waste Time Parsing HTML (So I do THIS) 70K views 1 year ago
Web Scraping: HTML Tables with Python 70K views 4 years ago
Beautifulsoup vs Selenium vs Scrapy - Which Tool for Web Scraping? 69K views 2 years ago
How I use SELENIUM to AUTOMATE the Web with PYTHON. Pt1 66K views 3 years ago
How I Scrape JAVASCRIPT websites with Python 62K views 3 years ago
Automate your job with Python 61K views 10 months ago
Argparse Basics - How I run my scripts via the Command Line 58K views 2 years ago
Render Dynamic Pages - Web Scraping Product Links with Python 53K views 3 years ago
Web Scraping NEWS Articles with Python 52K views 3 years ago
How to scrape SPORTS STATS websites with Python 52K views 3 years ago
API Endpoints? Get data from the web easily with PYTHON 51K views 3 years ago
Web Scraping Weather Data with Python 49K views 2 years ago
How I save my Scraped Data to a Database with Python! Beginners sqlite3 tutorial 48K views 3 years ago
```



```
df.nlargest(5, 'views')
```

	title	views	posted
0	Always Check for the Hidden API when Web Scraping	549000	2 years ago
1	Scrapy for Beginners - A Complete How To Examp...	239000	3 years ago
2	How to SCRAPE DYNAMIC websites with Selenium	150000	3 years ago
3	Web Scraping with Python: Ecommerce Product Pa...	133000	3 years ago
4	Web Scrape Websites with a LOGIN - Python Basi...	110000	3 years ago

Regulation & Compliance

- **Terms of Service**: Always check a website's terms before scraping.
- **Robots.txt**: Respect the guidelines to avoid unwanted access.
- **Copyright and Intellectual Property**: Ensure you have the right to use the data you scrape.
- **Data Privacy**: Collect personal data responsibly and comply with privacy laws.
- **Rate Limiting and Throttling**: Respect server limitations to avoid being blocked.
- **Attribution and Crediting**: Provide credit when using scraped data.
- **Ethical Considerations**: Avoid scraping sensitive or illegal content.
- **Potential Liability**: Understand the legal risks of scraping and act accordingly.

There are many other packages out there and ways to web scrape

1. **BeautifulSoup**: Parses HTML and XML documents for data extraction.
2. **Scrapy**: A framework for large-scale web crawling and scraping projects.
3. **LXML**: Efficient library for processing XML and HTML documents.
4. **MechanicalSoup**: Automates interactions with websites, useful for form submissions and login tasks.
5. **PyQuery**: Provides jQuery-like syntax for selecting and manipulating HTML elements.
6. **Splash**: Headless browser for scraping dynamic JavaScript-heavy websites.

Thank you!
Any Questions?