

# Web Scraping with Beautifulsoup

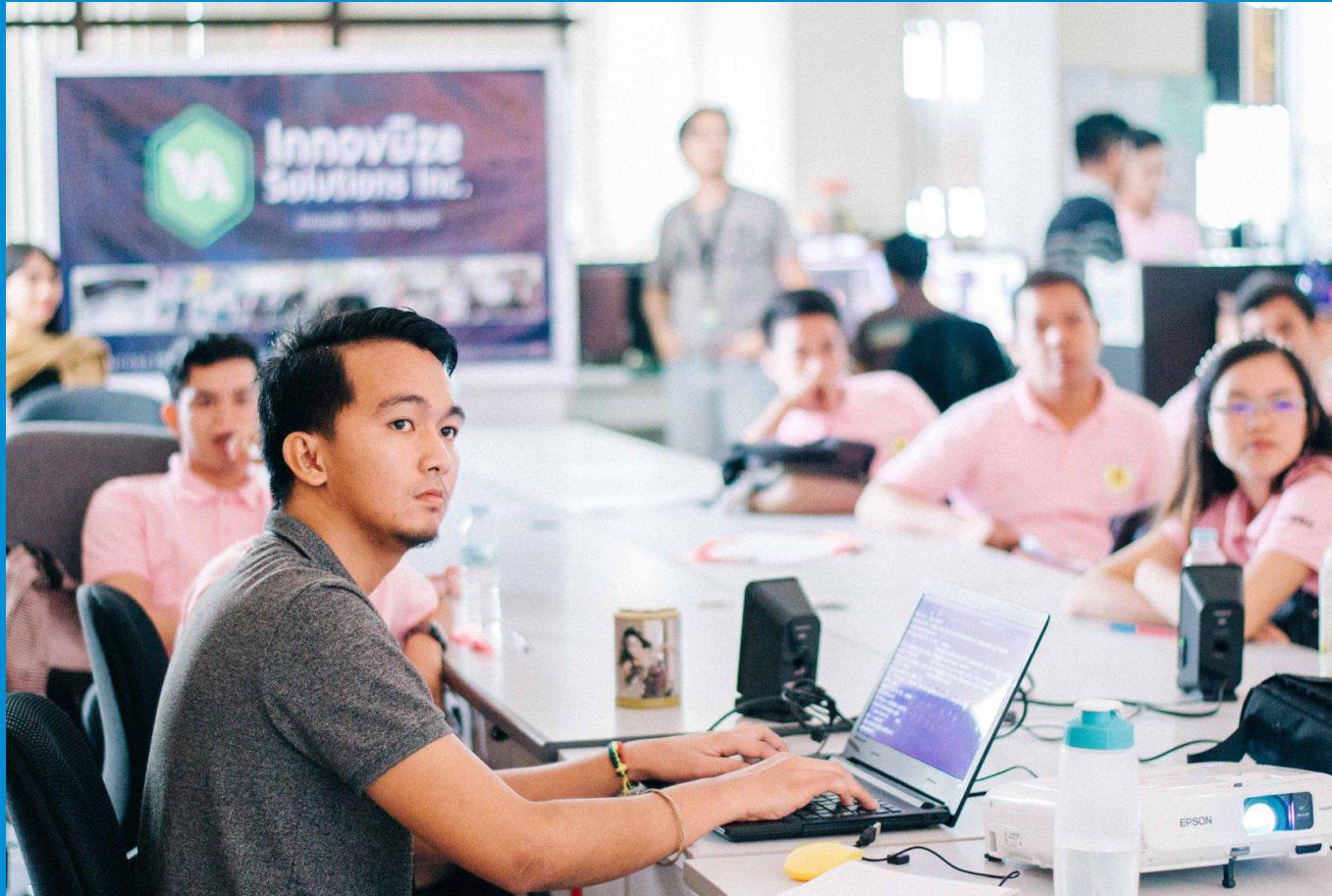
@harriebird

# whoami && groups

- Paul Harriet Asiñero / </harriebird>
- Web Developer at Innovuze Solutions Inc.
- PineapplePy, Hackm3 member
- Open-source software loverboy



# Previously...



# Previously...



# Previously...



# Previously...



# #ATM



# Agenda

- Introduce Web scraping
- Introduce BeautifulSoup
- Demo Time



# Needs

- Python 3
- beautifulsoup4
- background in HTML (tags, ids, classes)

# Web Scraping Overview

# What is Web Scraping?

- it is the practice of gathering data through any means other than a program interacting with an API



# What is Web Scraping?

- Commonly accomplished by writing an automated program that queries the web server, requests data, and then parses that data to extract needed information

# Why Scrape the Web?

- valuable data is coming from a website
- the website does not provide any API
- the website has an API but with limitations

# Web Scraping Workflow

- Retrieve HTML data from a web site
- Parse the data for target information
- Store the target information



# BeautifulSoup

- is a Python library for pulling out data out of HTML and XML files
- simplifies web scraping



# Using BeautifulSoup

# Initializing BeautifulSoup

- `from bs4 import BeautifulSoup`
- `soup = BeautifulSoup(html,'html.parser')`

# Navigating the parse tree

- **find()**
  - searches the soup and returns one result
- **findAll()**
  - searches the soup and returns all the results matching to the criteria
- **select()**
  - same with findAll() but uses CSS selectors in searching

# Navigating the parse tree

- `soup.body.p`
  - tag names can be used to navigate
- `getText()`
  - removes the tags enclosing the text and returns it
- `div.a['href']`
  - returns the value of the attribute specified

# Request data across the web

- **request from urllib**
  - can be helpful in making web requests
- **urlopen(url)**
  - opens the URL/Request object passed and returns the response contents
- **Request()**
  - used to create an advanced request to website like adding headers

# Saving data to a CSV file

- can be achieved by using the csv library
- `csv.writer(fileobj)`
  - specifies where the csv data will be saved
- `writerow(row)`
  - write the row to a file (method from csv module)

**Before we dive into the  
Demo**

# Read the Terms and Conditions of the website



# Check the robots.txt on the website



A screenshot of a web browser window displaying the robots.txt file for the website <https://www.facebook.com>. The browser's address bar shows the URL. The page content is a plain text document with the following content:

```
# Notice: Crawling Facebook is prohibited unless you have express written  
# permission. See: http://www.facebook.com/apps/site\_scraping\_tos\_terms.php  
  
User-agent: Applebot  
Disallow: /ajax/  
Disallow: /album.php  
Disallow: /checkpoint/  
Disallow: /contact_importer/  
Disallow: /feeds/  
Disallow: /file_download.php  
Disallow: /hashtag/  
Disallow: /l.php  
Disallow: /live/  
Disallow: /moments_app/  
Disallow: /p.php  
Disallow: /photo.php  
Disallow: /photos.php  
Disallow: /sharer/
```



**IPAGLABAN  
MO**

# Some websites block scrapers.



- modification of headers could help in this case

# Demo Time

# PineapplePy



- [facebook.com/groups/pineapplepy](https://facebook.com/groups/pineapplepy)

# Hackm3



• [facebook.com/hackm3.ph](https://facebook.com/hackm3.ph)