INTRODUCTION TO
SECURE WEB SCRAPING
WITH PYTHON

Lecturer: Jason Townes French

FOR THE WOMEN
FOUNDATION

COURSE OVERVIEW

- A hands-on overview to writing scripts to extract data from websites
- An exploration of challenges and concerns regarding web scraping

WHAT YOU'LL NEED

- A basic understanding of Python
- A computer running Python 2.7
- A basic understanding of Git and GitHub.com
- Chrome
- MySQL Workbench

WHO AM I?

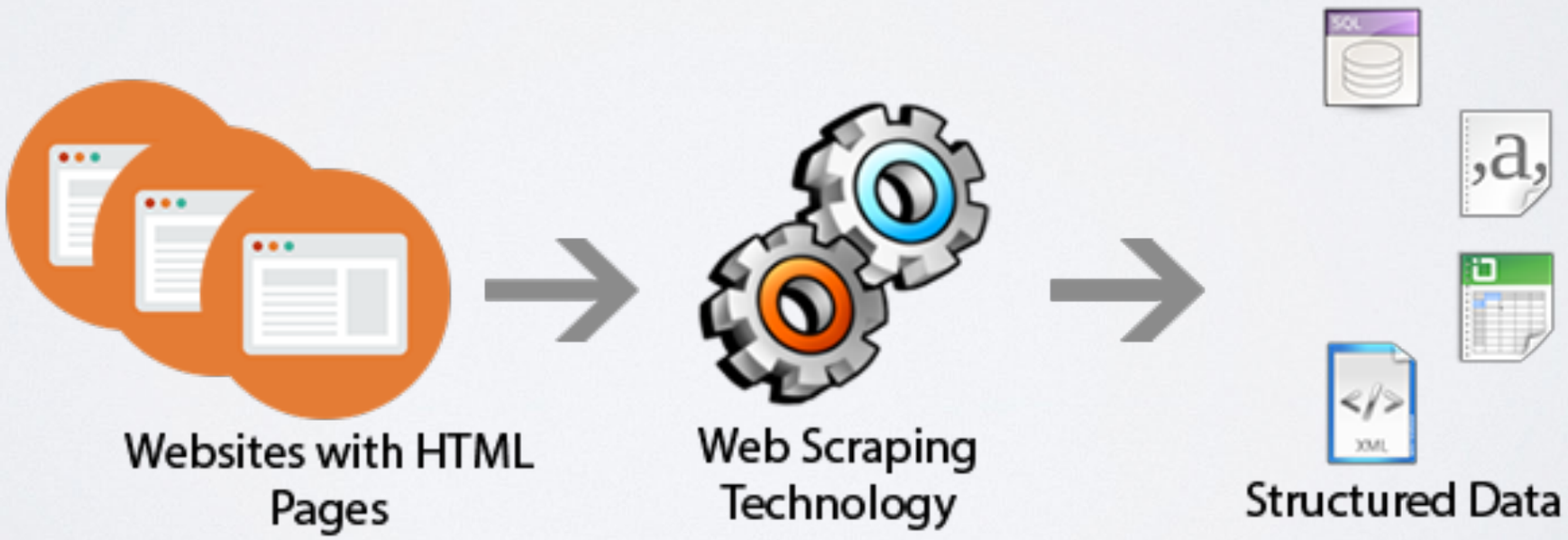
- Jason Townes French
- Brown University, Computer Science
- ex-Apple Software Engineer
- Entrepreneur from Silicon Valley
- Game developer

Raytheon
Space and Airborne Systems

WHAT IS “SCRAPING”?

WHAT IS WEB SCRAPING?

- Extracting data from a website
- Storing extracted data in structured format for further processing

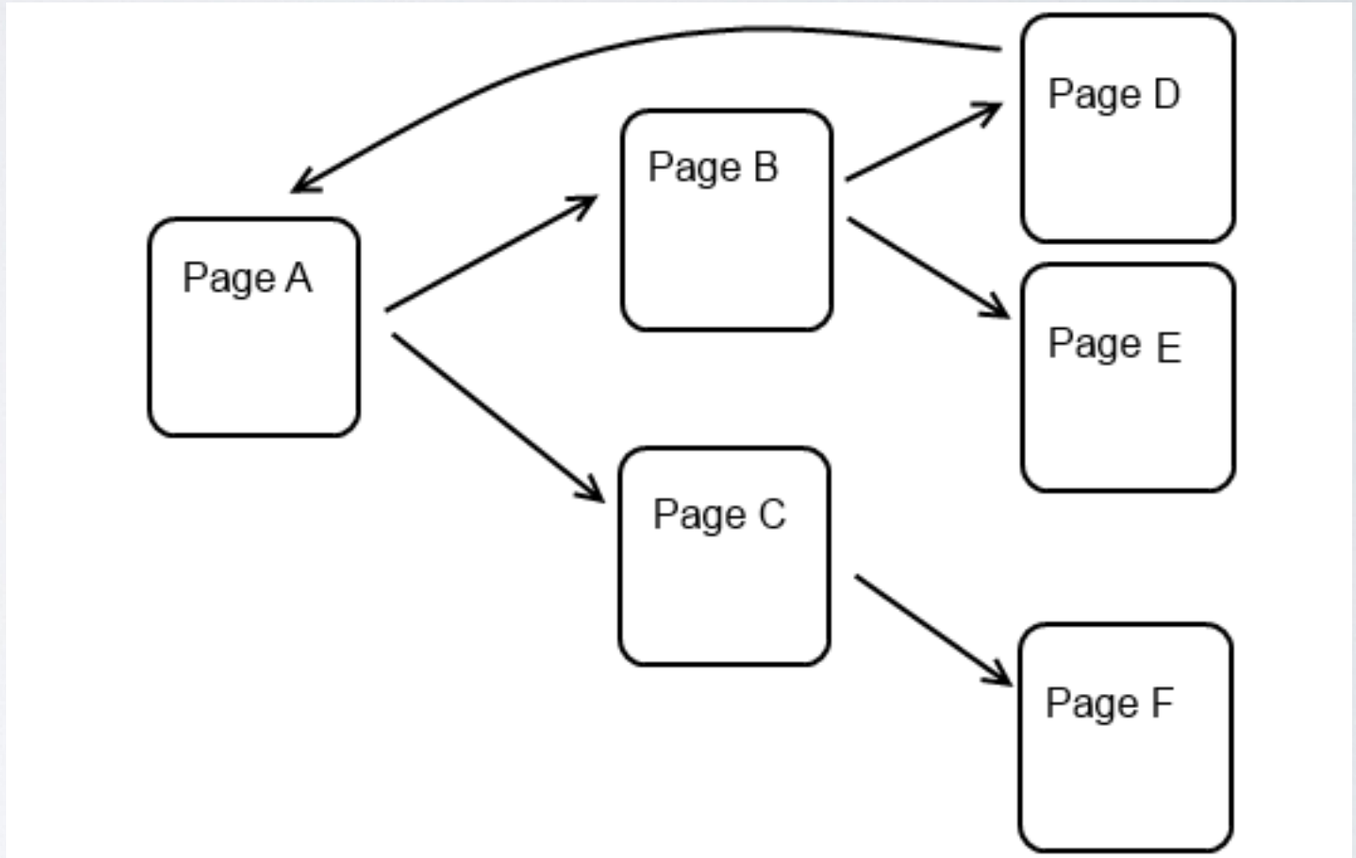


SCRAPING VS. CRAWLING

SCRAPING VS. CRAWLING

- Crawlers start at a page, visiting & indexing each link on the page, repeating until there are no more links to traverse

- Scrapers visit a specific page, extracting specific data for further processing



SCRAPING VS. CRAWLING

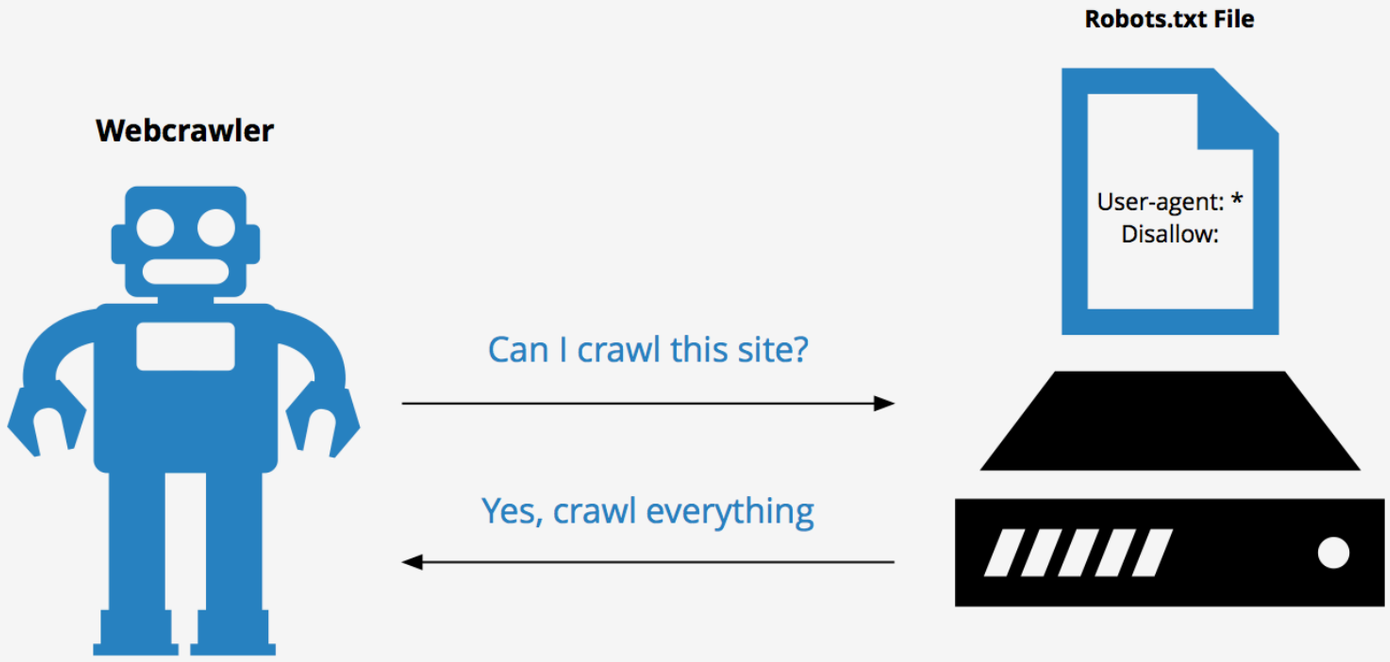
Google = CRAWLER

COMMON SCRAPING PROBLEMS

- Terms of Service
- robots.txt
- Interaction Throttling
- Tracking

COMMON SCRAPING PROBLEMS

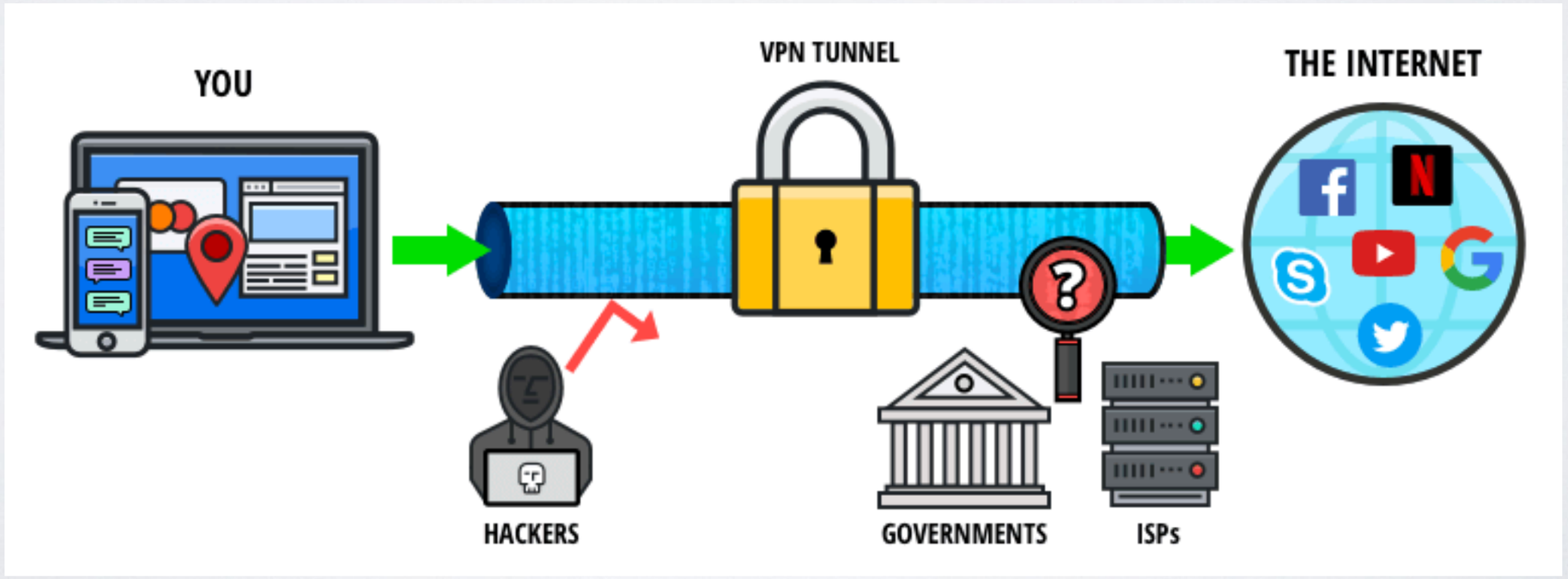
COMMON SCRAPING PROBLEMS



What Is a Robots.txt File

COMMON SCRAPING PROBLEMS

COMMON SCRAPING PROBLEMS



SECTION OVERVIEW: WHAT IS SCRAPING?

- Extracting structured data from websites
- Different from crawling
- Watch out for Terms of Service, robots.txt, and speed throttling
- Best to use a VPN

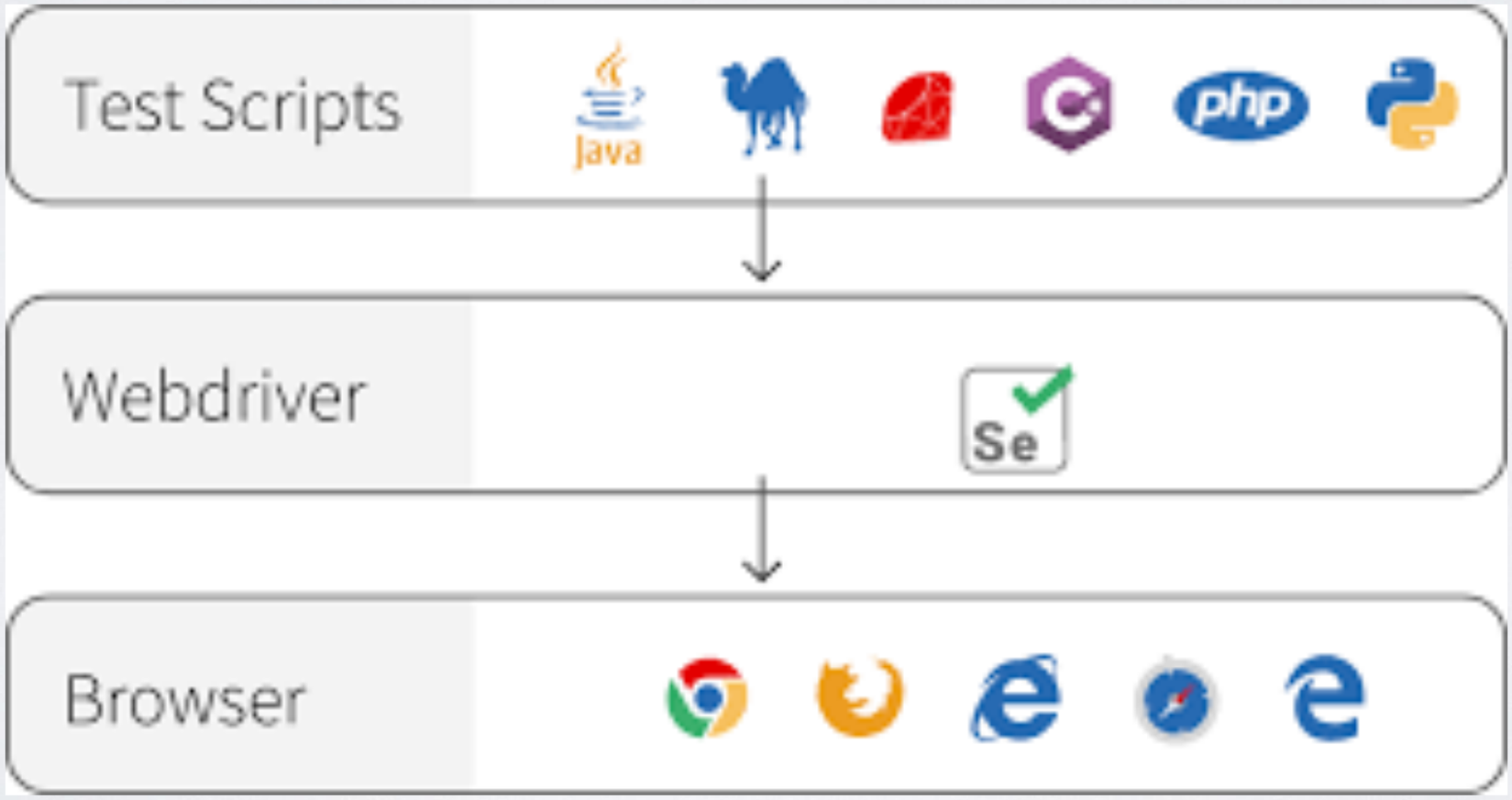
INTRODUCTION TO SELENIUM

WHAT IS SELENIUM?

- A tool for automating interaction with a web browser
- Useful for many types of testing but also scraping

HOW IT WORKS

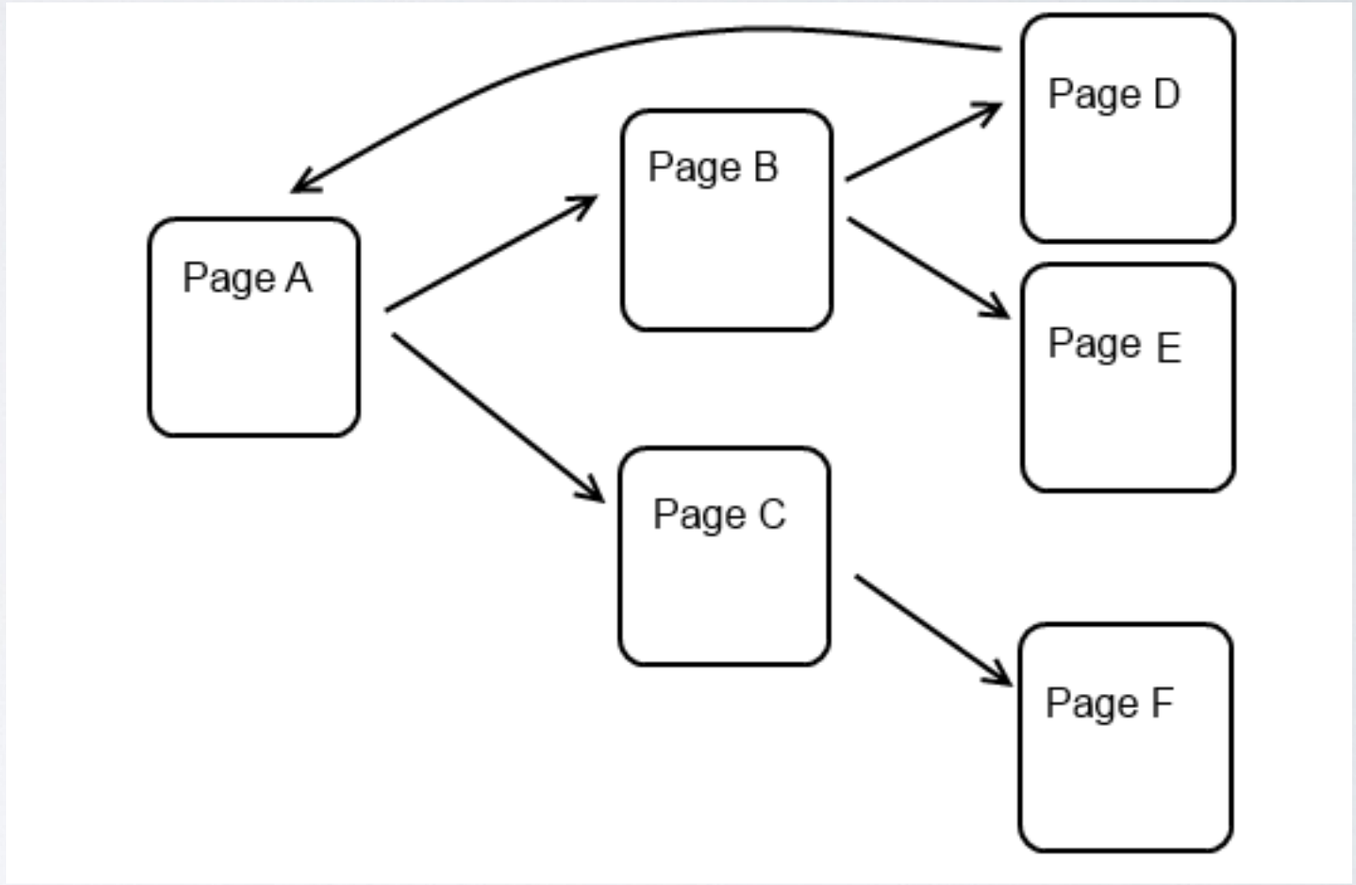
- You write scripts which talk to the Web Driver
- The Web Driver talks to the Browser
- The Browser does what your script said



CORE FUNCTIONALITY

- Crawlers start at a page, visiting & indexing each link on the page, repeating until there are no more links to traverse

- Scrapers visit a specific page, extracting specific data for further processing



EXERCISE 1: INSTALL ENVIRONMENT

- Checkout git repository:

```
git clone https://github.com/DigitalSolomon/ftw-secure-web-scraping.git
```

- Install Python Virtual Environment:

```
sudo pip install virtualenv
```

- Create virtual environment:

```
virtualenv env
```

- Activate virtual environment

```
source env/bin/activate
```

CYGWIN

https://www.cygwin.com

Apps | EDUKASYON Jira | Staging | Internshi... | Ot

Cygwin

Get that Linux feeling - on Windows

This is the home of the Cygwin project

What...

...is it?

Cygwin is:

- a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows.
- a DLL (cygwin1.dll) which provides substantial POSIX API functionality.

...isn't it?

Cygwin is not:

- a way to run native Linux apps on Windows. You must rebuild your application *from source* if you want it to run on Windows.
- a way to magically make native Windows apps aware of UNIX® functionality like signals, ptys, etc. Again, you need to build your apps *from source* if you want to take advantage of Cygwin functionality.

The Cygwin DLL currently works with all recent, commercially released x86 32 bit and 64 bit versions of Windows, starting with Windows Vista. For more information see the FAQ.

Cygwin version

The most recent version of the Cygwin DLL is **3.0.7**.

Installing Cygwin

Install Cygwin by running setup-x86_64.exe (64-bit installation) or setup-x86.exe (32-bit installation)

Use the setup program to perform a fresh install or to update an existing installation.

Keep in mind that individual packages in the distribution are updated separately from the DLL so the Cygwin DLL version is not useful as a general Cygwin distribution release number.

EXERCISE 2: INSTALL SELENIUM

- Install Selenium

```
pip install selenium
```

- Download Web Driver

```
https://sites.google.com/a/chromium.org/chromedriver/downloads
```

- Write first script
  - Fill out exercise2.py

EXERCISE2.PY

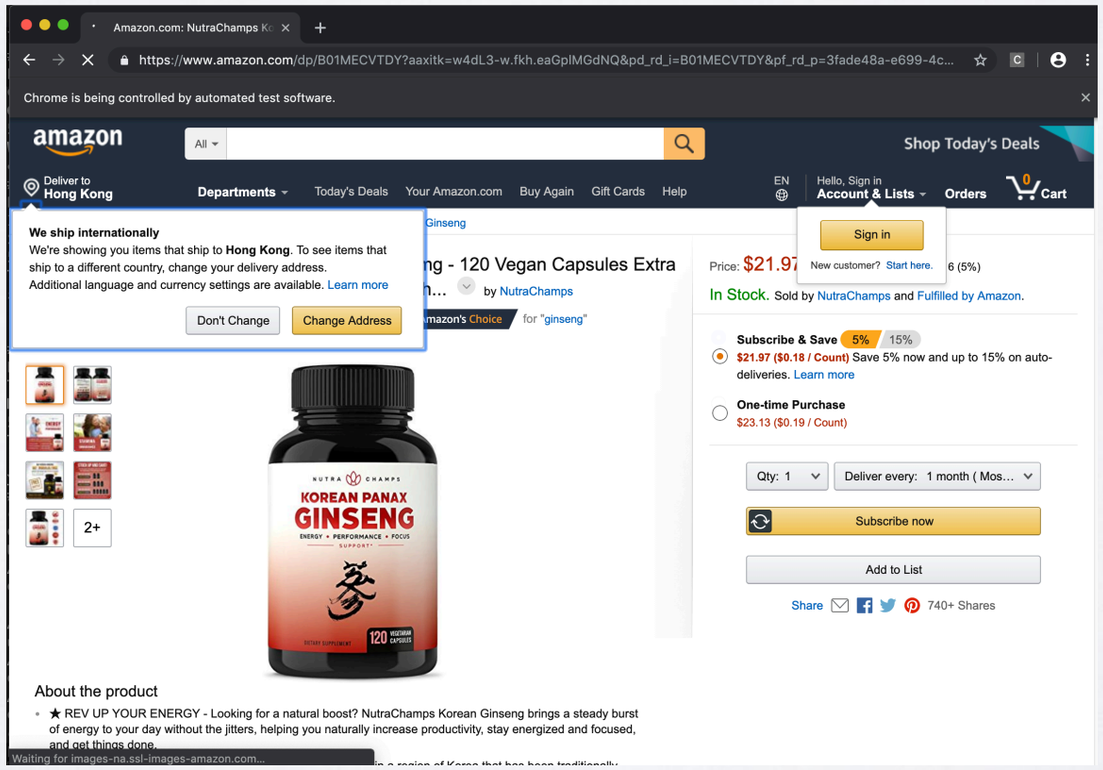
```
1   from selenium import webdriver
2   import os
3
4
5   dirpath = os.getcwd()
6   filepath = dirpath + '/chromedriver'
7   print('Path to Driver: ' + filepath)
8   browser = webdriver.Chrome(executable_path = filepath)
9   # browser.get('...
10
```

EXERCISE 3: CLICK REVIEWS

- exercise3.py
  - Access Amazon Product Listing

```
browser.get('AmazonListingURLHere')
```

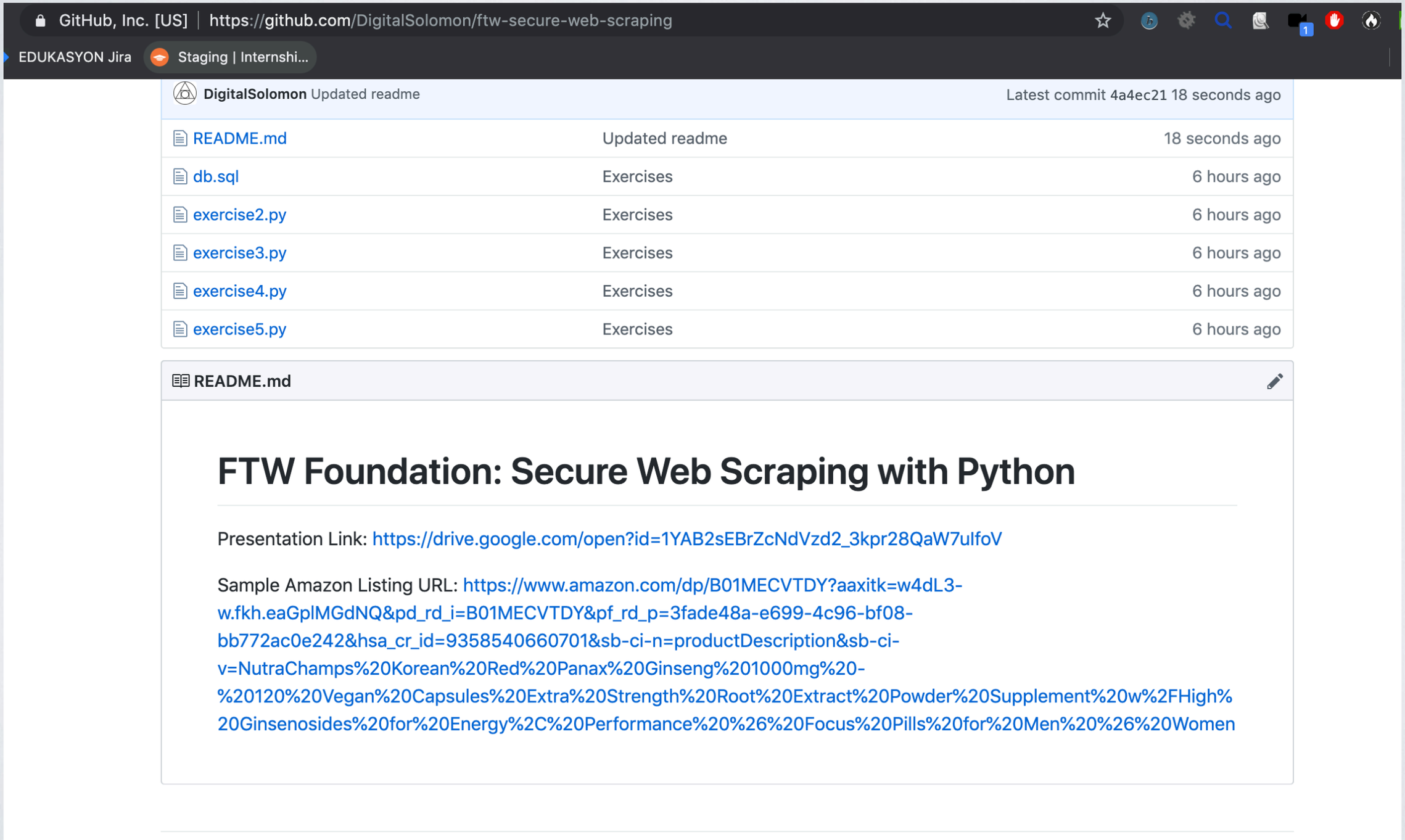
- Close Popup & Click on Reviews



EXERCISE3.PY

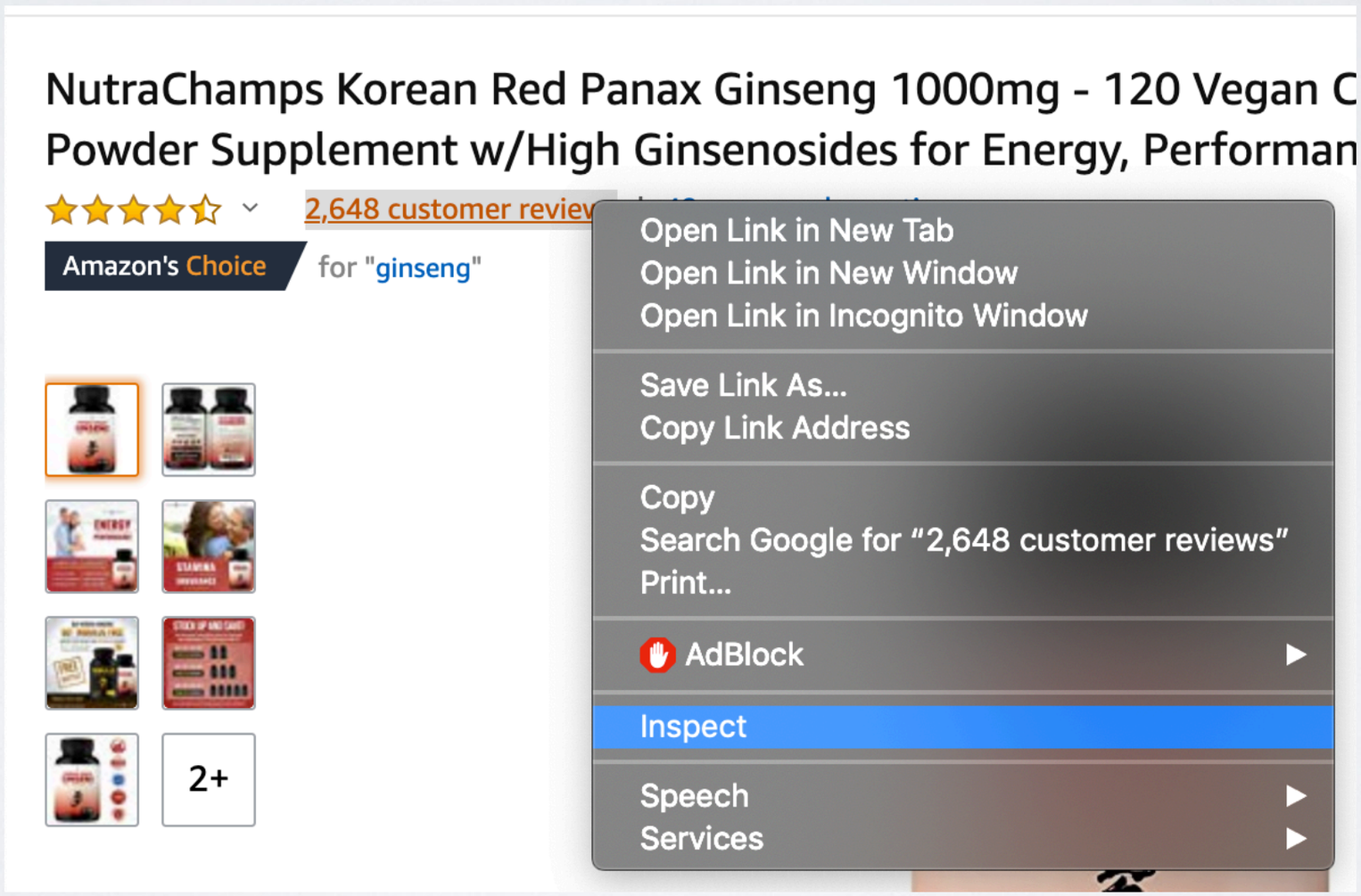
```python
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.common.exceptions import TimeoutException
import os

dirpath = os.getcwd()
filepath = dirpath + '/chromedriver'
print('Path to Driver: ' + filepath)
browser = webdriver.Chrome(executable_path = filepath)
browser.get('AmazonListingURLHere')

try:
    # Dismiss initial popup
    element = WebDriverWait(browser,5).until(
        EC.presence_of_element_located((By.XPATH, 'FillThisIn')))

    element.click()

    # Click on Reviews
    element = WebDriverWait(browser,5).until(
        EC.presence_of_element_located((By.XPATH, 'FillThisIn')))

    element.click()


except TimeoutException:
    print("Failed to load element")
finally:
    browser.quit()
```

SAMPLE AMAZON URL



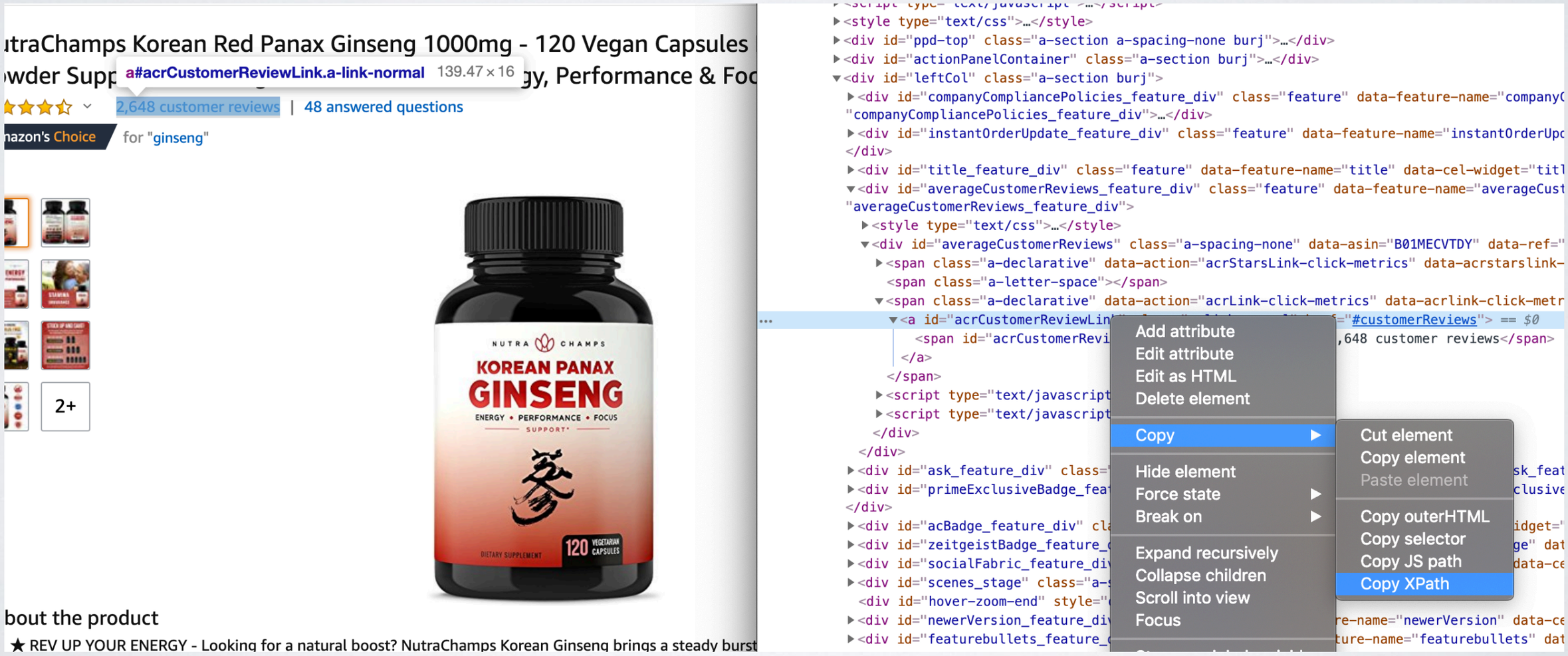
EXERCISE 3: CLICK REVIEWS

- Use "Inspect Element" to copy the XPath



EXERCISE 3: CLICK REVIEWS

- Use “Inspect Element” to copy the XPath



EXERCISE 4: PROCESS REVIEWS

- Open exercise4.py
- Find the element which contains all of the reviews
- Find the class name of each individual review
- Hint: Use the Inspector!

EXERCISE 5: STORE IN DATABASE

- Copy/paste db.sql into MySQL Workbench and execute it to create your database and tables

Limit to 1000 rows

```
1  CREATE DATABASE scraperdb;
2  USE scraperdb;
3  ALTER DATABASE scraperdb CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci;
4  CREATE TABLE reviews (id BIGINT(7) NOT NULL AUTO_INCREMENT,
5  reviewId VARCHAR(100), title VARCHAR(200), stars VARCHAR(100), reviewDate VARCHAR(50), content VARCHAR(10000),PRIMARY KEY(id));
6  ALTER TABLE reviews CONVERT TO CHARACTER SET utf8mb4 COLLATE
7  utf8mb4_unicode_ci;
8  ALTER TABLE reviews CHANGE title title VARCHAR(200) CHARACTER SET utf8mb4
9  COLLATE utf8mb4_unicode_ci;
10 ALTER TABLE reviews CHANGE content content VARCHAR(10000) CHARACTER SET utf8mb4
11 COLLATE utf8mb4_unicode_ci;
```

EXERCISE 5: STORE IN DATABASE

- Install PyMySQL

```
pip install PyMySQL
```

- Import PyMySQL in Python script

```
import pymysql
```

- Fill out Database Connection Details

```
pymysql.connect(host='127.0.0.1',user='root', passwd = 'YourPasswordHere', db = 'mysql',
charset = 'utf8')
```

- Complete Database Insert Function

```
cur.execute('INSERT INTO reviews (reviewId, title, stars, reviewDate, content)
VALUES ''("%s","%s","%s","%s","%s")', (id, title, stars, date, content))
```

- Call your Database Insert Function

```
store(reviewId, date, stars, title, reviewContent)
```