“An Introduction to Web Scraping:

Collecting Data from 3rd Party Websites”

Project: Raider Data

Luke Harris

Table of Contents

1. Project Description
2. Web Scraping
3. Scraping Methods
   1. Copy-and-Paste
   2. Spreadsheets
   3. Web Browser Extensions
   4. Scraping Scripts
   5. Web Crawlers
4. HTML Basics
   1. HTML Basic Components
   2. Sample HTML Code
   3. Example Scraping Script
5. References
6. Project Description

Building a TTU mobile application on the Android platform is the goal of project Raider Data. Although the university already has a campus application it does not include features popular among students and suffers from a lack of proper implementation. As a result it has low ratings and is seldom used. Initially, we planned to build an app that would implement several new features including Hours of Operation, TechAnnounce, Bookstore purchases, and Student Organization Directory and improve some of the existing ones (Bus Routes, User Interface).

Based on feedback from Dr. Lim and our classmates we have decided to build the app and focus on implementing the TechAnnounce feature. One way the university disseminates information to faculty and students is through TechAnnounce announcements. These announcements are sent via email and posted on the university website twice a day. They include information on a wide variety of topics such as campus events, fundraising, student organizations, student employment opportunities, and university research. The goal of the TechAnnounce feature is to have announcements pushed to the mobile app in real time. This will require design and implementation of the app, connecting the app to a server, and creating a system to retrieve announcements from the TTU website and store them on the server.

The interviews we conducted with current TTU students and the feedback we have received from our classmates lead us to believe this is a feature that students would like to have and is therefore an improvement worth pursuing.

II. Web Scraping

The term web scraping has been defined in different ways based on the context of the publication and the motive of the author. For example, Webopedia.com defines it as “...an application that processes the [HTML](http://www.webopedia.com/TERM/H/HTML.html) of a w[eb page](http://www.webopedia.com/TERM/W/web_page.html) to extract data for manipulation.” WebHarvey.com offers a similar definition, namely that “[Web Scraping](http://en.wikipedia.org/wiki/Web_scraping)... is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database…” DistilNetworks.com, an advocate for the prohibition of scraping defines it in this way, “[Web scraping](http://www.distilnetworks.com/web-scraping/) is the act of taking content from a website with the intent of using it for purposes outside the direct control of the site owner.”

In general, web scraping is the collection of data from websites. For our purposes, data is anything published on a website including text, graphics, video, photos, etc...Scraping methods include everything from basic copy-and-paste duplication to using automated programs that simulate the browsing of a website by a human.

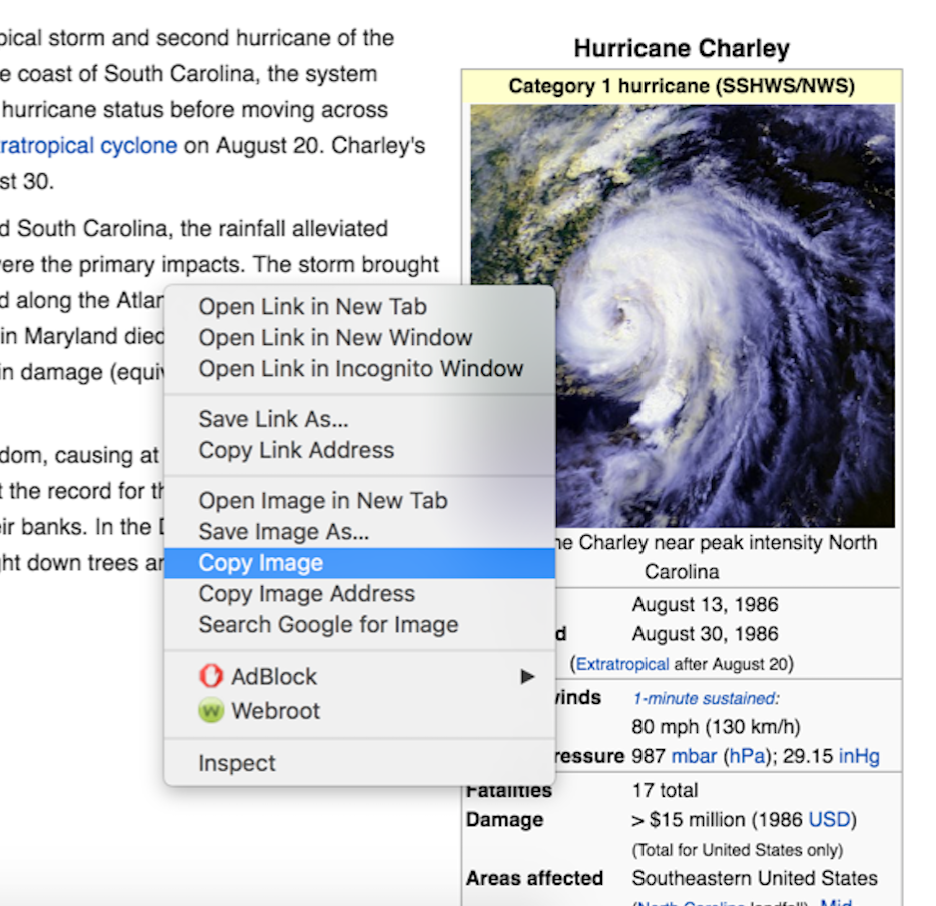
III. Scraping Methods

1. Copy-and-Paste

One of the easiest ways to scrape data from a website is the copy-and-paste method. The vast majority of websites publish information that can be duplicated in this way. Although this method is quick and simple it is also very limited in the type and amount of data it can capture; it can only be used to collect very small amounts of specific kinds of data at one time. People who use this method are typically looking to copy images, graphics, tables, charts, or small amounts of text (3).

The copy-and-paste method is very easy to use. All a person has to do is right-click on an image or highlighted section of text and choose the appropriate function. As an example, suppose someone wants to include a photo of Hurricane Charley in a research paper. As they are reading about hurricanes on the Wikipedia website they see a photo they would like to use. After right-clicking on the photo they should choose either ‘Copy Image’ or ‘Save Image As’ to duplicate the photo and use it as they need. This is illustrated in the picture below.

Photo 1. Scraping a photo using copy-and-paste



The technology behind copy-and-paste is straightforward. The operating system allocates a portion of memory as a data buffer. When information is copied it is put onto a utility called the clipboard and held there to be used as needed (21).

B. Spreadsheets

Another fairly simple way to scrape a website is to use the import statements built in to spreadsheet software such as Google Sheets. This method allows much more data to be scraped at one time than copy-and-paste, allows users to manipulate the extracted information, and can also be automated. There are several statements that allow different kinds of data to be scraped. A good example of this is using the ImportHTML function to scrape a large table of data and import it into a spreadsheet. The following example shows how to do this from a Google Drive (8, 12).

The FDIC publishes a list of banks that have failed since 2000. This information can easily be scraped into a Google Sheets spreadsheet using the correct import statement and the URL of the site where the data is published. The syntax of the import statement is =IMPORTHTML(URL, dataType, index). The URL gives the location of the data and must be enclosed in quotation marks and must include the protocol. The data type is either “table” or “list”. The index specifies which table or list in the HTML source should be returned. The import statement we use for this example becomes =IMPORTHTML("https://www.fdic. gov/bank/individual/failed/banklist.html", "table", 0). Enter this into the A1 space of the spreadsheet and hit Enter. The data is scraped from the website and imported into the document as seen below (12, 13, 22).

Photo 2. Scraping a table using spreadsheet import statements

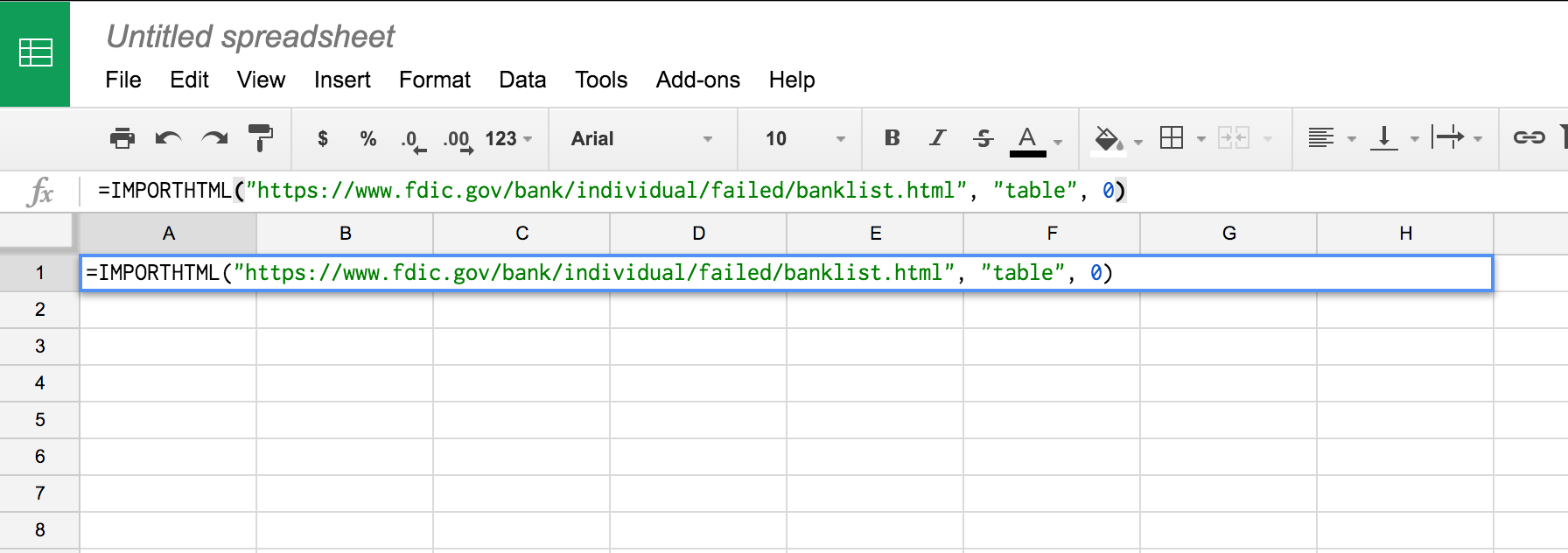
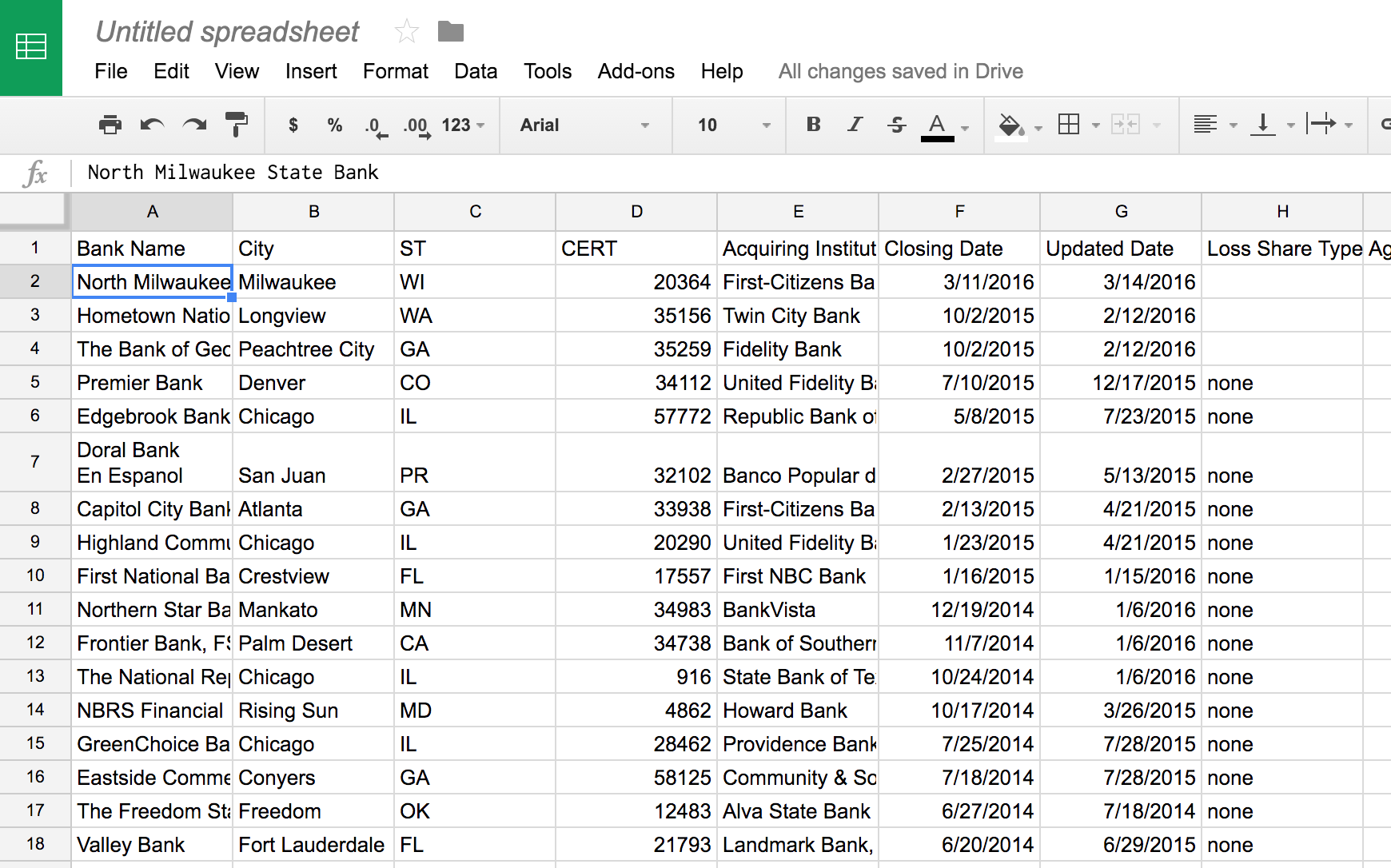


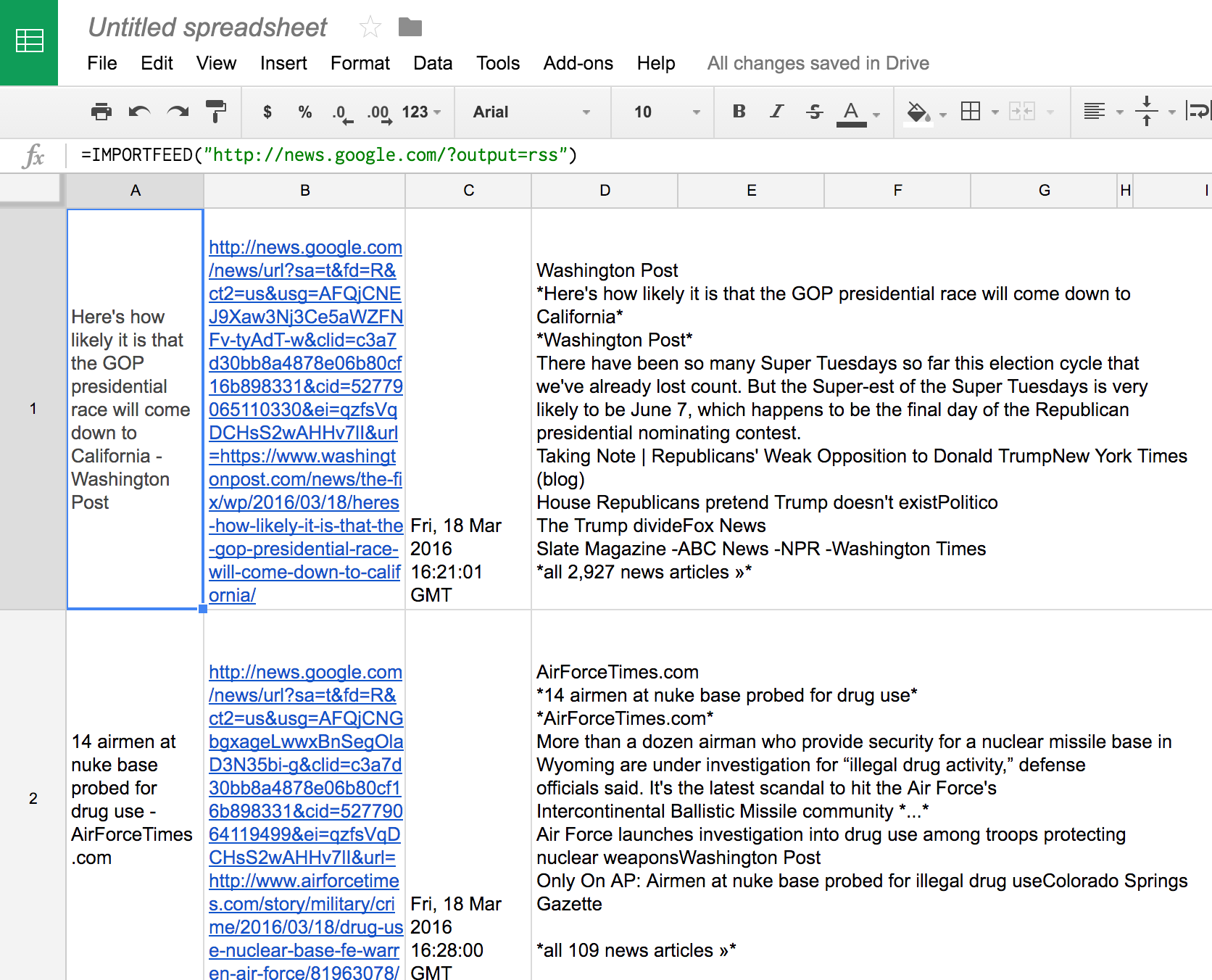
Photo 3. Data retrieved by the spreadsheet



Another useful import statement is the ImportFEED which can be used to capture data from an RSS or ATOM feed. The syntax is =IMPORTFEED(url, query, headers, num\_items). url is the URL of the RSS or ATOM feed, including the protocol. The value for url must either be enclosed in quotation marks or be a reference to a cell containing the appropriate text. Query (optional) is "items" by default and specifies what data to fetch from the URL. The "feed" query returns a single row containing feed information including title, description, and url. The "feed <type>" query returns a particular attribute of the feed, where <type> is title, description, author, or url. The "items" query returns a full table containing items from the feed. If num\_items is not specified, all the items currently published on the feed are returned. "items <type>" returns a particular attribute of the requested items, where <type> is title, summary, url, or created (the post date associated with the item). Headers (optional) tells the function whether to include column headers as an extra row on top of the returned value. num\_items (optional) is used for queries of items, and returns the number of items, starting from the most recent. If num\_items is not specified, all items currently published on the feed are returned (12, 13).

A news feed can easily be imported into a spreadsheet. Putting =IMPORTFEED(“http://www.news.google.com”) into a Google Sheets spreadsheet gives the following.

Photo 4. Scraping a news feed



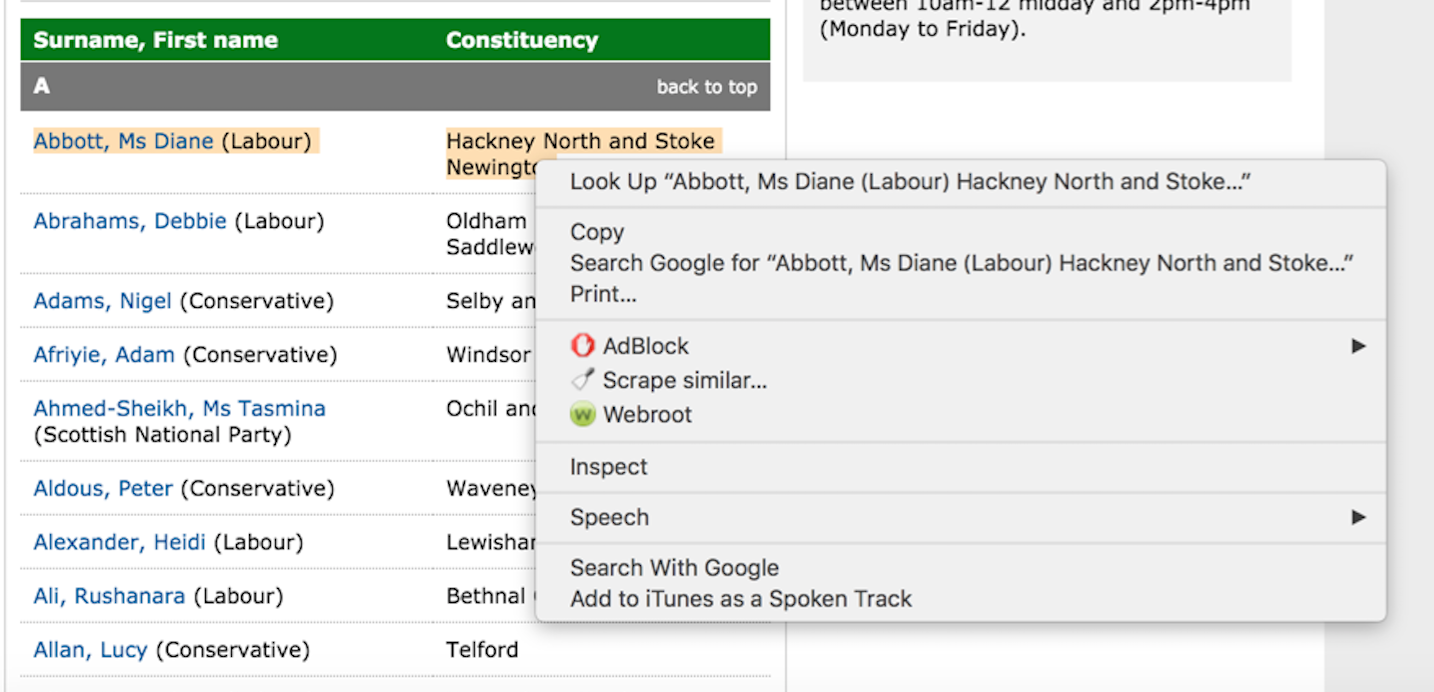
The output includes the article headline, its URL, a date/time stamp, and a summary.

C. Web Browser Extensions

Most web browsers offer scraper extensions that make data collection more efficient and much less frustrating. Google Chrome has an extension called Scraper that works similarly to Google Sheets but is slightly more powerful. The scraper extension is available for free from the Chrome store. The following example shows how to use the extension to scrape data from a website and place it in a spreadsheet for analyzing and/or manipulating (23).

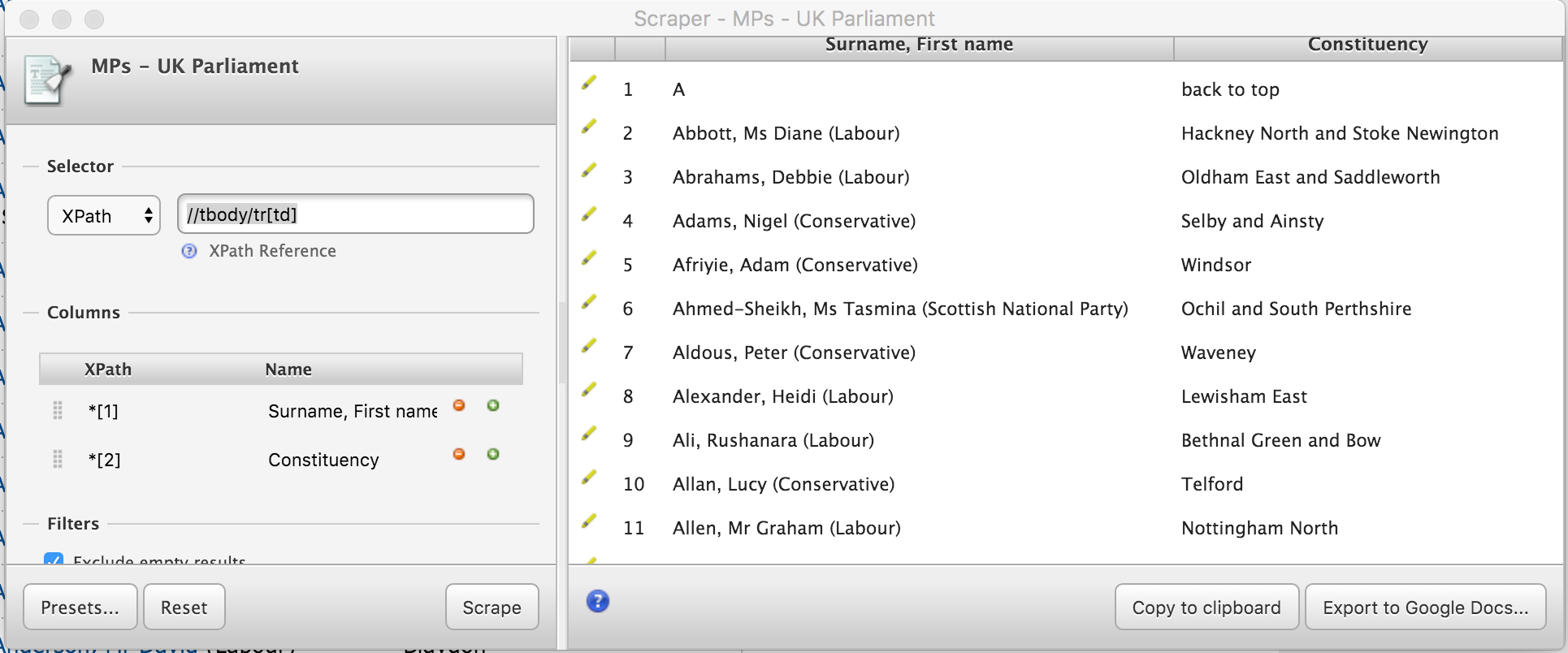
A list of the elected Members of Parliament for the United Kingdom is available at http://www.parliament.uk/mps-lords-and-offices/mps/. Gathering the data and adding it to a spreadsheet is simple enough. First, highlight the name and constituency of one of the MPs. Right click on the info and an option ‘Scrape Similar’ will be listed with the others as seen below.

Photo 5. Preparing to scrape the List of UK Members of Parliament with Chrome extension



After clicking on ‘Scrape similar…’ a new window will open. This is the scraper console and it contains the scraped data.

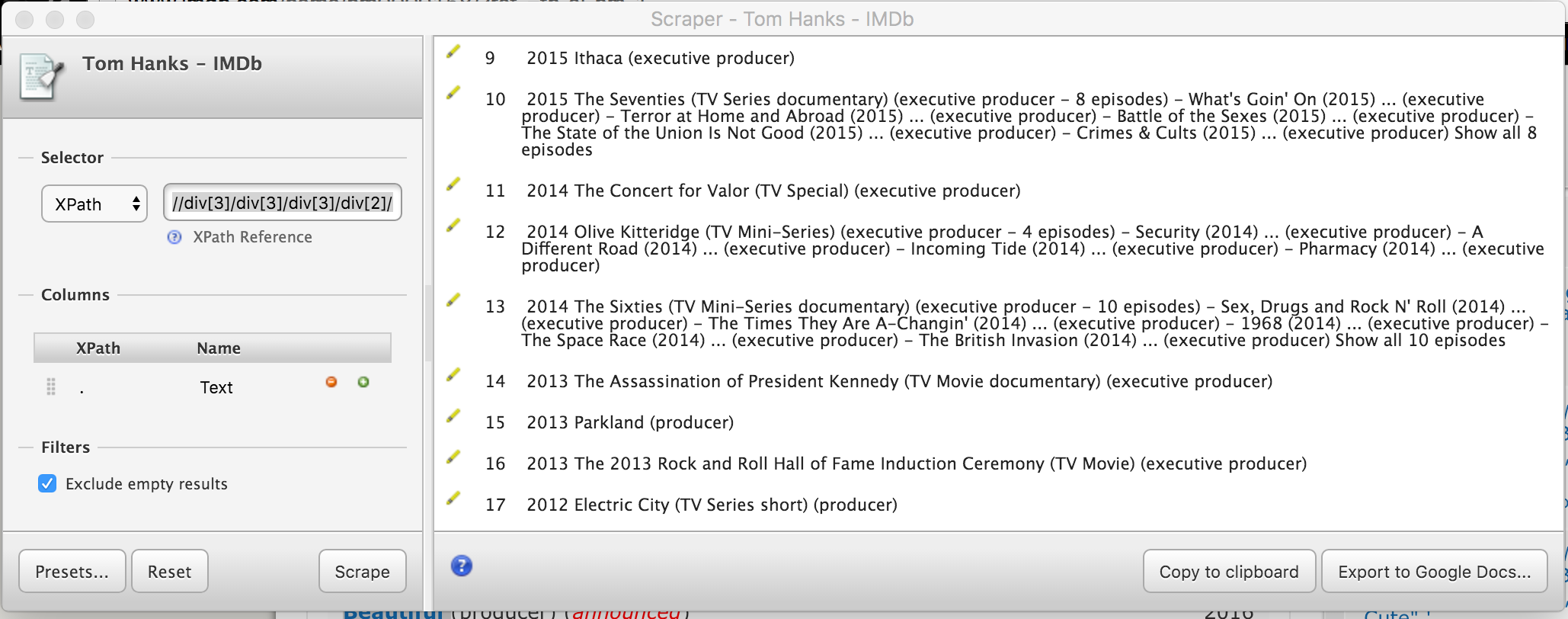
Photo 6. The scraper console displaying the extracted information



In the bottom, right corner of the console there is a button ‘Export to Google Docs’. Pushing it imports the scraped data into a Google Sheets spreadsheet.

Sometimes the information we want is not formatted the way we would like. The scraper extension can also retrieve and format data fairly easily. Movie titles and their year of production starring a certain actor are available at IMDB.com. Entering an actor/actresses name in the search bar quickly yields the information to be scraped. However, using the above method to scrape the data yields a garbled table of info that it not very useful in its current state (10).

Photo 7. Improperly formatted data



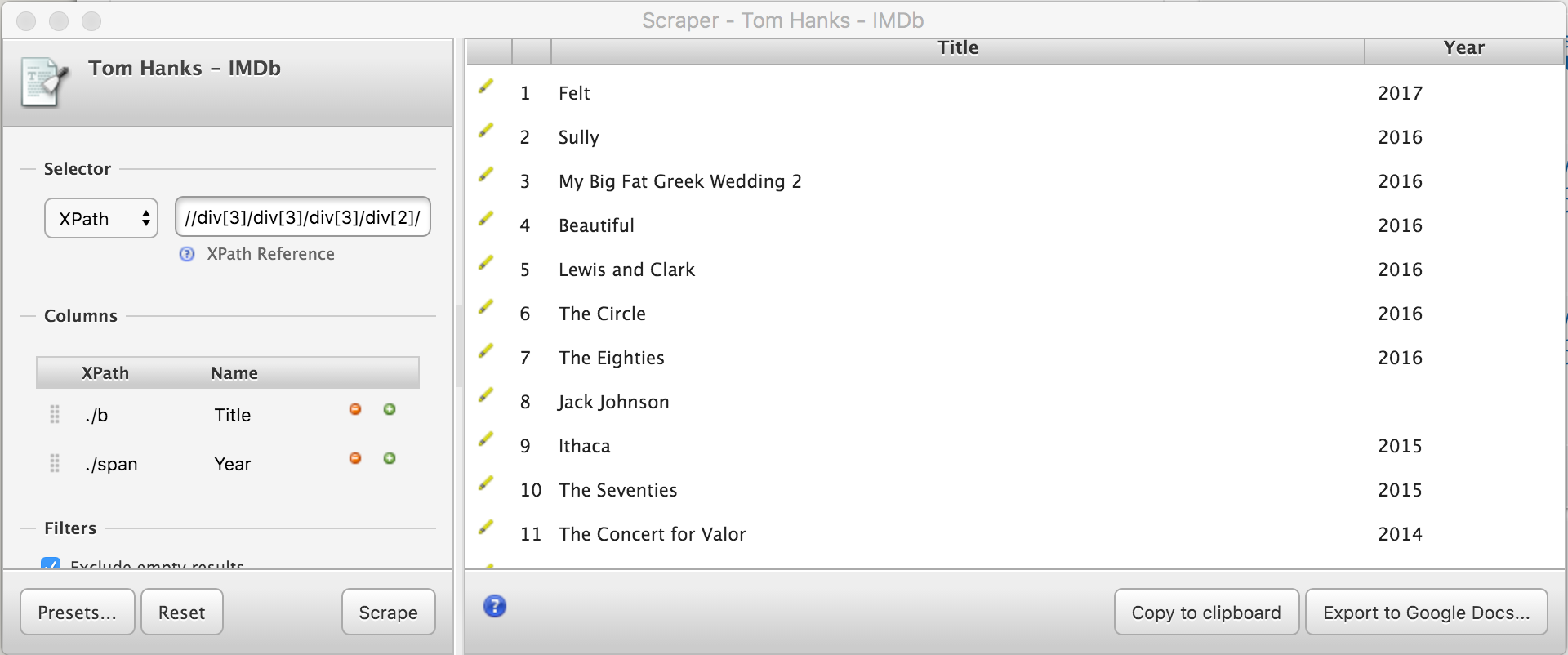
The scraper console includes the XPath of the data that we scraped. XPath is a syntax that describes a way to locate and process items in XML and HTML documents by using an addressing syntax based on a path through the document's logical structure or hierarchy. The scraper generates the XPath for the scraped data automatically and all the information contained in the path is displayed. The data can be formatted once it is located in the HTML code for the web page. Right clicking on a movie title and choosing Inspect produces the following screen (10, 24).

Photo 8. Inspector pane of a web browser



The production year of the movie is located between the <span> tags and the movie title between a pair of <b> tags. Using the columns feature of the scraper console is the next step. Add /b to the first XPath and change the name of the column to Title. Then add another XPath, making it ./span and change the name to Year. Clicking on ‘Scrape’ produces the info we are looking for (10).

Photo 9. Properly formatted scraped data



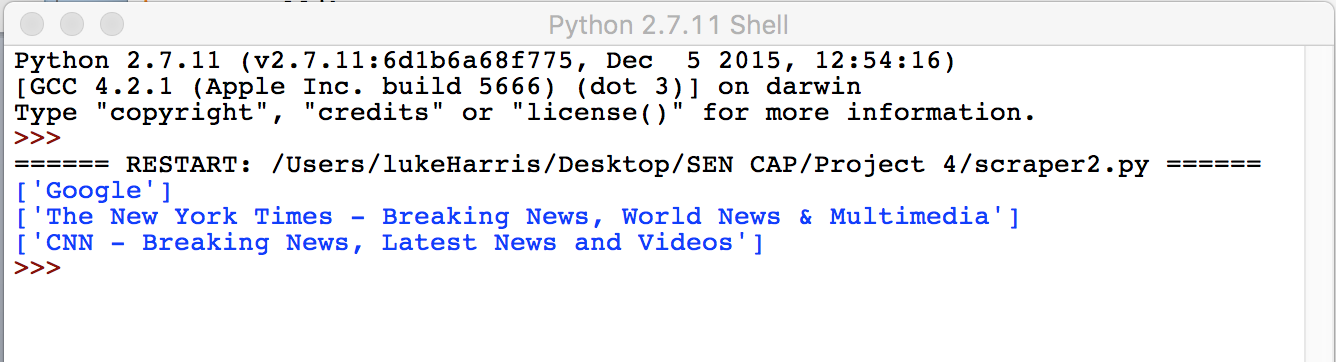
D. Scraping Scripts

Scripts are short programs written in high-level programming languages. They are the most versatile and powerful of the scraping methods described so far. In general they use URL and regex library function calls to retrieve data from websites (25).

A regular expression is a sequence of characters that define a search pattern. They are generally used in pattern matching for strings and have numerous useful functions including regex processors found in search engines, search and replace dialogs of word processors and text editors, and in the command lines of text processing utilities, such as grep, sed and AWK (19, 20). A short script that employs these technologies and its output is seen below.

Photo 10. Python script that retrieves the titles of given websites

Photo 11. Title scraper output



E. Web Crawlers

A Web crawler is a bot that systematically browses the Internet. Scrapers, search engines, and websites use them to update their content or indexes of others sites' web content. Web crawlers can copy all the pages they visit for later processing (26).

A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are visited according to a set of policies. The crawler typically maintains an archive of websites it has visited. The archives are usually stored in such a way they can be viewed, read and navigated as they were on the live web, but are preserved as ‘snapshots'.

A basic web crawler and its output are seen below. It uses Python’s BeautifulSoup and urlparse technologies. Beautiful Soup is a Python library for extracting data out of HTML and XML files. It works with parsers to provide idiomatic ways of navigating, searching, and modifying the parse tree. The urlparse module defines a standard interface to break URL strings up in components (addressing scheme, network location, path etc.), to combine the components back into a URL string, and to convert a relative URL to an absolute URL given a base URL (27, 28).

Photo 12. A web crawler used to visit and archive hyperlinks on a website

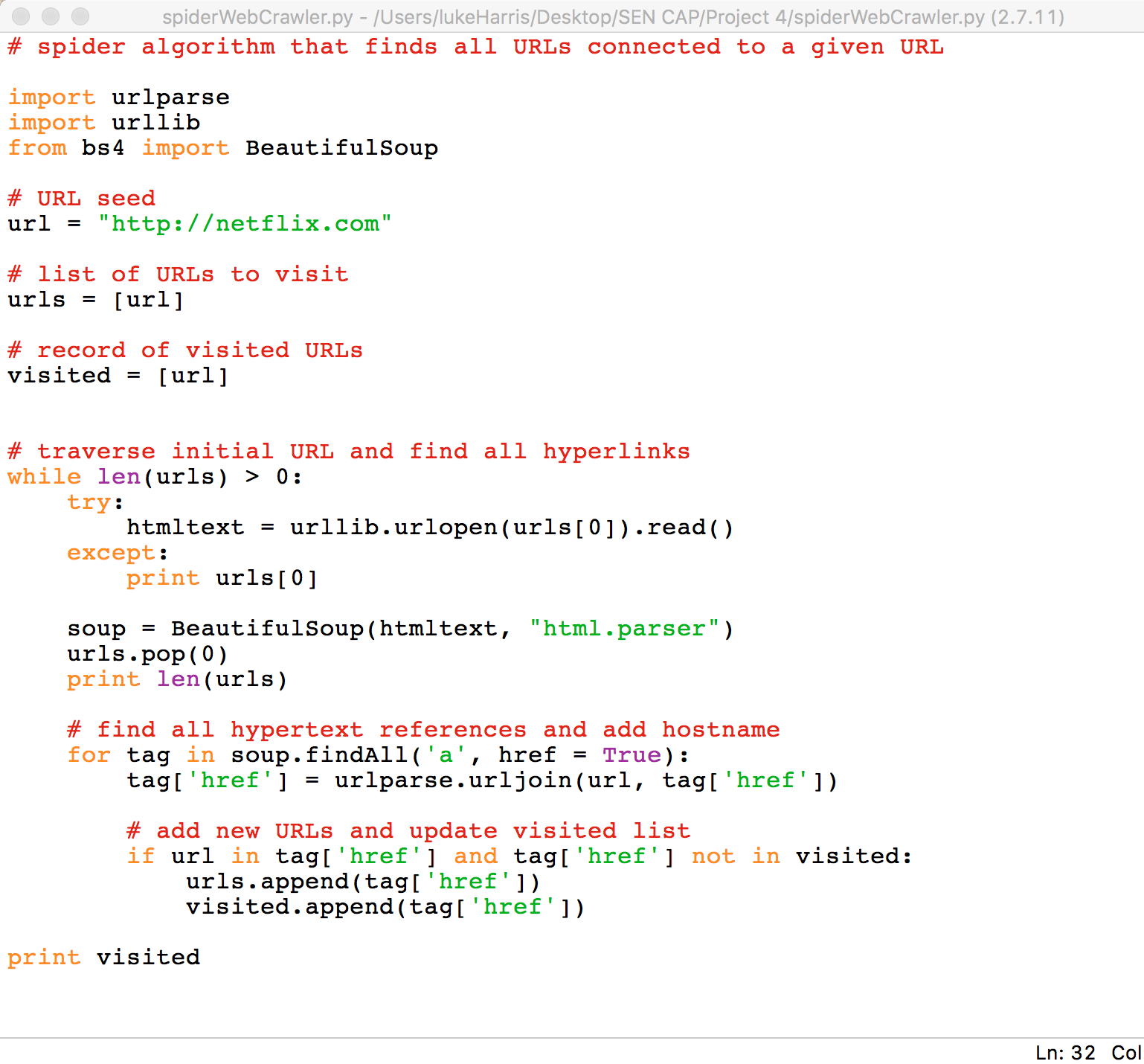
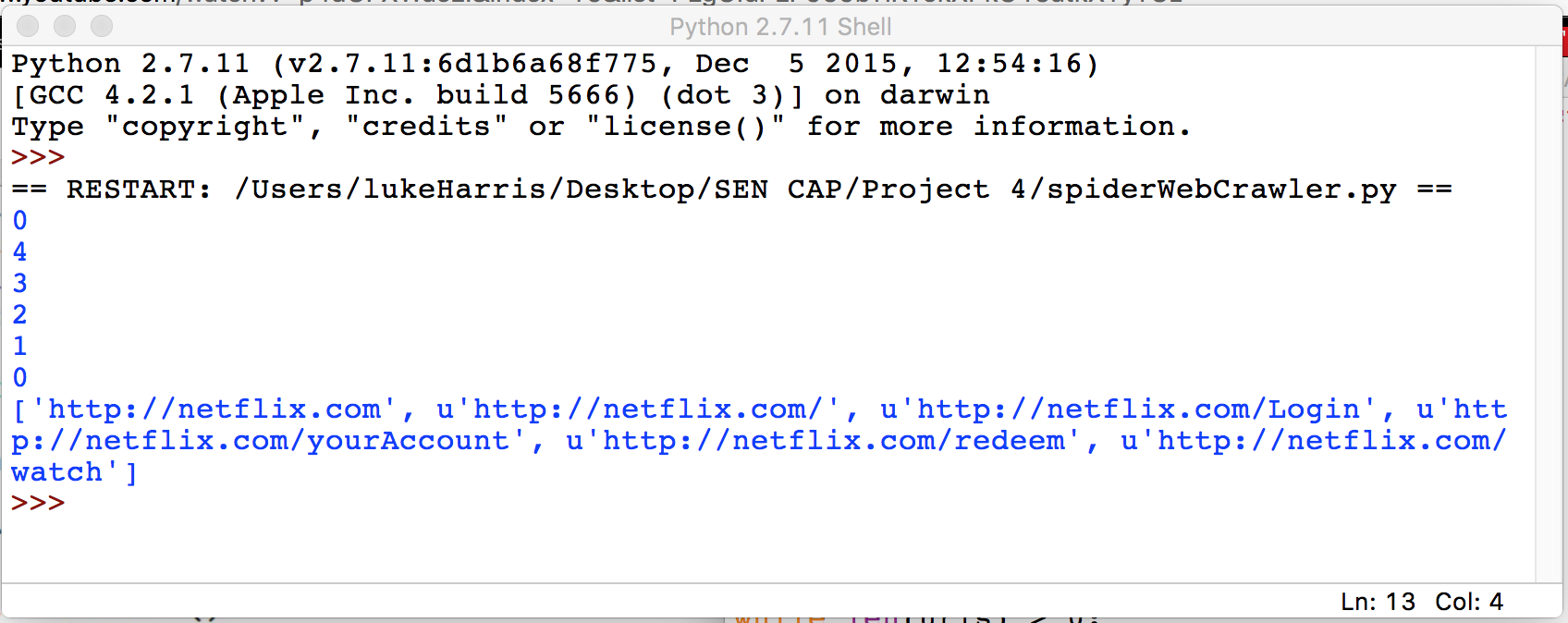


Photo 13. Web crawler output



IV. HTML Basics

A. Basic HTML Components

HTML code uses tags, which are enclosed in < > or </> when closing the section, to organize the code into sections. The tag <body> is used to contain all other tags within the code. All other tags must be inside the body. The tag <h#>, which can be h1or any number following sequentially for multiple headings, is used for headings and subheadings. These appear larger and above the main paragraph text. Finally, <p> is used for the main text that appears in paragraphs. This is where most of the text that appears on the webpage is written.

B. Sample HTML Code

Photo 14. Simple HTML Code



The code in photo 14 produces the webpage in photo 15 below.

Photo 15. Produced Webpage



The text that appears on the webpage is put into the table with the id datatable. The column titles in bold, A and B, and then put into the tags <TH>, so that they will be more prominent. The data in the tables, the numbers, are then put into the data tags, which are <TD>. This produces the table that is shown in photo 15.

C. Example Scraping Script

HTML scraping scripts are used to automated the process of getting data from HTML code. Below in photo 16 is an example of a scraping script.

Photo 16. HTML Scraping Script



This scraping script gets the data from the HTML code shown in photo 14. It returns the following output when run, shown in photo 17.

Photo 17. Generated Output



This makes it much easier to get data from websites, when a large amount of data is needed from the website.

V. References

1. http://www.webopedia.com/TERM/W/Web\_Scraping.html

2. https://www.webharvy.com/articles/what-is-web-scraping.html

3. https://en.wikipedia.org/wiki/Web\_scraping

4. http://docs.python-guide.org/en/latest/scenarios/scrape/

5. https://www.europeandataportal.eu/sites/default/files/library/201510\_web\_scraping\_applications\_and\_ tools.pdf

6. https://blog.hartleybrody.com/web-scraping/

7. http://www.distilnetworks.com/web-scraping/

8. https://eagereyes.org/data/scrape-tables-using-google-docs

9. https://en.wikipedia.org/wiki/Regular\_expression#Patterns\_for\_non-regular\_languages

10. https://datapatterns.readthedocs.org/en/latest/recipes/scraper-extension-for-chrome.html

11. http://searchsoa.techtarget.com/definition/XPath

12. https://support.google.com/docs/answer/3093335?hl=en

13. https://support.google.com/docs/answer/3093337

14. https://en.wikipedia.org/wiki/Regular\_expression

15. https://docs.python.org/2/library/urllib.html?highlight=urllib

16. https://docs.python.org/2/library/re.html?highlight=re.compile#

17. https://docs.python.org/2/library/re.html

18. http://www.regular-expressions.info/

19. https://en.wikipedia.org/wiki/Regular\_expression#Patterns\_for\_non-regular\_languages

20. https://en.wikipedia.org/wiki/Grep

21. https://en.wikipedia.org/wiki/Clipboard\_(computing)

22. https://www.fdic.gov/bank/individual/failed/banklist.html

23. https://www.notprovided.eu/7-tools-web-scraping-use-data-journalism-creating-insightful-content/

24. http://www.w3schools.com/xsl/xpath\_intro.asp

25. https://www.youtube.com/playlist?list=PLgGfaPLP959bTiKYokXPkO15atkXYyYOz

26. https://en.wikipedia.org/wiki/Web\_crawler

27. http://www.crummy.com/software/BeautifulSoup/bs4/doc/

28. <https://docs.python.org/2/library/urlparse.html>

29. http://www.simplehtmlguide.com/basics.php