AAM1 TASK 1: WEB SCRAPING

Josh King

Introduction

This project results in the creation of a Python web scraping script capable of extracting all unique url's on a US Census Bureau page that point to an html file and storing each link in a newly created csv file.

This project submission contains the following files:

- "Josh King Web Scraping Documentation.pdf" – This file, explaining the process.
- "Population and Housing Unit Estimates.html" – The downloaded US Census Bureau page's html file from which the links were extracted (for reference).
- "Web Scraping – Josh King – WGU.py" – The actual Python code used to extract the links.
- "Web Scraping – Josh King – WGU.html" – The web scraping code was written in a jupyter notebook environment. As such, the notebook was copied as an html file for easier viewing.
- "Josh Webscrape WGU.csv" – The resultant csv file into which the unique links were dumped via execution of the Python code.

The Python code itself utilizes the below libraries and modules for execution. The documentation for each library is provided in the "References" section at the end of this document. The general utility of each library or module is described immediately, but the specific use cases will be addressed as needed in the task prompt explanations.

- BeautifulSoup – This library allows for the parsing of html files for easy navigation, searching, and manipulation of their elements.
- Requests – This library also allows for parsing of html files but here is used as an efficient means of making a call to online pages for data.
- Regular Expressions (re) – This module provides a powerful means of matching string expressions.
- CSV File Reading and Writing (csv) – This module provides options for its namesake, the reading and writing of data to csv files.

Program Tasks

A) Web Links Extraction:

After first importing the needed libraries and modules for the project, I utilized the Requests library to make a call to the census.gov url and stored that data as a request object. The text content of this object was then parsed by the BeautifulSoup library and stored in the object "soup" for manipulation as shown below:

```
[ ]:
1 from bs4 import BeautifulSoup
2 import requests
3 import re
4 import csv
```

```
[ ]:
1 url = 'http://www.census.gov/programs-surveys/popest.html'
2 response = requests.get(url)
3 soup = BeautifulSoup(response.text, 'html.parser')
```

To extract the web links from the soup object (and thus the census website), I used BeautifulSoup's "find_all" method which can extract specified elements from an html file. Here I chose to select all "<a… >" html tags having a "href" attribute, each such tag representing a link on the webpage. This was done in a loop in order to later evaluate each link according to the requirements explained in the following sections. To get the actual link from each html tag, I utilized BeautifulSoup's tag.get() method to retrieve the value of each "href" attribute as seen here:

```
3 for tag in soup.find_all('a', href=True):
4
5     link = tag.get('href').rstrip('/')
6
```

B) Selecting of Only HTML Links:

The prompt requested that we only extract links pointing to html pages (as opposed to that of pdf, csv, or other filetypes for instance). One temptation would be to only extract links ending with ".htm" or ".html" file extensions via a ".endswith()" string selection. This however would miss the significant case of index files. For instance, "www.someaddress.com" and "www.abcdefg.com/directory", while not ending in ".htm" or ".html", often automatically direct to or through an implicit "index.html" file. While it is correct that these index files are sometimes other types (such as an "index.php" file), for the scope of this project, I assumed all indexes routed to or resulted in an html file. In order to accomplish the objective while not losing these links, I chose to use the regular expressions module to discriminate against links ending with the pattern "*/(some characters).(some three-letter file extension)*". The relevant code section follows:

```
7     if bool(re.search('[^\/]\/[^\/].*\....$', link)) and not link.endswith('.htm'):
8         continue
```

The generalized logic is as follows: *If the link ends with a three-letter file extension, and that file extension is not .htm (an html file), then skip this link within our loop*. Most internet filetypes are three letters, so this provides a reasonable method while allowing the index files to be included and not discriminating against the more common ".html" extension.

The regular expression used can be broken down into the following segments for clarification:

- "[^\/]\/[^\/]" – This begins our expression by looking for a forward slash (\/), but not immediately touching another forward slash ([^\/]). We want to avoid the double forward slash as otherwise the expression might match cases such as "https://website.com", treating the ".com" as a file extension.
- ".*\." – After seeing the forward slash, we allow for any number of any characters (.*) up until we see the period (\.).
- "…$" – After the period, we expect three more characters (…) before the end of the string ($).

C) Saving Links as Absolute URIs:

In order to save the links as absolute URIs, this meant that all links must be resolvable by a browser as-is. The code responsible for this task follows:

```
9       elif link.startswith('#'):
10          continue
11      elif link == "":
12          continue
13      elif link.startswith('/'):
14          link = "https://www.census.gov" + link
```

As seen above, multiple checks were taken during the loop across each link. Accordingly:

- The first "elif" statement causes the loop to skip over links that were simply linking to other anchor points on the same page (these would start with a "#").
- The second "elif" statement makes a simple error check so as to not include any blank links.
- The final "elif" statement checks to see if the link is an internal link to the same domain. Such links would not include the domain out of redundancy and begin at the corresponding directory. As such, the first character would be "/". These were resolved by prepending the domain back into the link to make it a URI.

D) Removal of Duplicate Links:

In order to ensure no duplicate links were included in the final list, the following code snippets were used in tandem:

Within the for loop collecting each link:

```
link = tag.get('href').rstrip('/')
```

After the for loop had completed and the list of links ("links_list") was compiled:

```
links_nodupes = list(set(links_list))
```

Of the above, the first code snippet strips the trailing "/" from each link before processing. This is useful both for our file extension check noted earlier and also to create an equivalency in cases where one reference to a link may not include the final slash (for instance "website.com" vs "website.com/").

The second code snippet operates on the compiled list of links by first converting it into a Python "set" object which automatically results in only the unique values contained in "links_list." This set object was then converted back to the list format for ease of use in later operations and stored as "links_nodupes".

E) Provided Code:

The code used can be found within the files "Web Scraping – Josh King – WGU.py" or "Web Scraping – Josh King – WGU.html".

F) HTML at Time of Web Scraping:

This HTML file can be found in the included "Population and Housing Unit Estimates.html".

G) Produced CSV File:

The produced csv file containing all processed extracted links can be found in the included "Josh Webscrape WGU.csv" file.

H) Verification of Results:

The below image is a screenshot showing both the code and print of the list where all links from the Census Bureau page were appended after processing. However, due to the length of the list, the full image does not display well here.

It is recommended to review the included “Web Scraping – Josh King – WGU.html” copy of the Jupyter notebook which includes this print instead.

```
links_nodupes = list(set(links_list))
print(links_nodupes)
```

```
['https://www.census.gov/library/publications/2010/demo/p25-1138.html', 'https://www.census.gov/newsroom/stories.html', 'http
s://www.census.gov/newsroom/blogs.html', 'https://www.census.gov/library/visualizations/2019/comm/15-fastest-growing-cities.htm
l', 'https://www.census.gov/library/video.html', 'https://www.census.gov/about/policies/privacy/privacy-policy.html#accessibili
ty', 'https://www.census.gov/library/photos.html', 'https://www.census.gov/topics/income-poverty.html', 'https://www.census.go
v/programs-surveys/popest/about.html', 'https://www.census.gov/data/training-workshops.html', 'https://www.commerce.gov', 'http
s://www.census.gov/internationalprograms', 'https://www.census.gov/programs-surveys/popproj.html', 'https://www.census.gov/dat
a.html', 'https://www.census.gov/topics/research.html', 'https://www.census.gov/about/faqs.html', 'https://www.census.gov/data/
tables/time-series/demo/popest/pre-1980-national.html', 'https://www.census.gov/programs-surveys/popest/guidance.html', 'http
s://www.census.gov/topics/population/race.html', 'https://www.census.gov/2020census', 'https://www.census.gov/programs-surveys/
are-you-in-a-survey.html', 'https://www.census.gov/quality', 'https://www.census.gov/programs-surveys/sis.html', 'https://www.c
ensus.gov/datalinkage', 'https://www.census.gov/topics/population/hispanic-origin.html', 'https://www.census.gov/topics/health.
html', 'https://twitter.com/uscensusbureau', 'https://www.census.gov/about/what.html', 'https://www.instagram.com/uscensusburea
u', 'https://www.census.gov/newsroom/press-releases/2019/estimates-characteristics.html', 'https://www.census.gov/data/related-
sites.html', 'https://www.census.gov/programs-surveys/popest/guidance-geographies.html', 'https://www.census.gov/about-us', 'ht
tps://www.census.gov/topics/education.html', 'https://www.census.gov/developers', 'https://www.census.gov/academy', 'https://ww
w.census.gov/library/publications.html', 'https://www.census.gov/library/visualizations/2019/comm/age-race-distribution.html',
'https://www.census.gov/programs-surveys/popest/data/tables.html', 'https://www.census.gov/programs-surveys/geography.html', 'h
ttps://www.census.gov/careers', 'https://www.census.gov/sipp', 'https://www.census.gov/partners', 'https://www.census.gov/progr
ams-surveys/popest/library.html', 'https://www.census.gov/data/data-tools.html', 'https://www.census.gov/programs-surveys/cps.h
tml', 'https://www.census.gov/newsroom/press-releases/2019/characteristics-embargo-advisory.html', 'https://www.census.gov/libr
ary/reference/code-lists/schedule/b.html', 'https://www.census.gov/programs-surveys.html', 'https://www.census.gov/programs-sur
veys/cbp.html', 'https://www.census.gov/fieldjobs', 'https://www.census.gov/topics/preparedness.html', 'https://www.census.gov/
programs-surveys/surveys-programs.html', 'https://www.census.gov/topics/families.html', 'https://www.census.gov/about/policies.
html', 'https://www.usa.gov', 'https://www.census.gov/about/who.html', 'https://www.census.gov/programs-surveys/sbo.html', 'htt
ps://www.facebook.com/uscensusbureau', 'https://www.census.gov/programs-surveys/popest/library/visualizations.html', 'https://w
ww.census.gov/programs-surveys/abs.html', 'https://www.census.gov/en.html', 'https://www.census.gov/about/index.html', 'http
s://www.census.gov/EconomicCensus', 'https://www.census.gov/library/reference/code-lists/naics.html', 'https://www.census.gov/p
rograms-surveys/popest/library/publications.html', 'https://www.census.gov/programs-surveys/acs', 'https://www.census.gov/progr
ams-surveys/cog.html', 'https://www.census.gov/programs-surveys/decennial-census/decade/2010.html', 'https://www.census.gov/abo
ut/business-opportunities.html', 'https://www.census.gov/AmericaCounts', 'https://www.census.gov/programs-surveys/metro-micro.h
tml', 'https://www.census.gov/programs-surveys/susb.html', 'https://www.census.gov/data/tables.html', 'https://www.census.gov/n
ewsroom/press-releases/2019/estimates-characteristics/estimates-characteristics-sp.html', 'https://www.census.gov/data/softwar
e.html', 'https://www.census.gov/library/audio.html', 'https://www.census.gov/about/contact-us/staff-finder.html', 'https://ww
w.census.gov/programs-surveys/popest/about/schedule.html', 'https://www.census.gov/topics/international-trade.html', 'https://w
ww.census.gov/about/what/admin-data.html', 'https://www.census.gov/data/developers/data-sets/Geocoding-services.html', 'http
s://www.census.gov/businessandeconomy', 'https://www.census.gov/programs-surveys/ahs.html', 'https://www.census.gov/library/pub
lications/2010/demo/p25-1139.html', 'https://www.census.gov/topics/population/population-estimates.html', 'https://www.census.g
```

References

Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation. (2019). Retrieved 16 August 2019, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

csv — CSV File Reading and Writing — Python 3.7.4 documentation. (2019). Retrieved 16 August 2019, from https://docs.Python.org/3/library/csv.html

re — Regular expression operations — Python 3.7.4 documentation. (2019). Retrieved 16 August 2019, from https://docs.Python.org/3/library/re.html

Requests-HTML: HTML Parsing for Humans (writing Python 3)! — requests-HTML v0.3.4 documentation. (2019). Retrieved 16 August 2019, from https://html.Python-requests.org/