PROGRAMMING IN PYTHON — C996

Carl Andrew Perkins

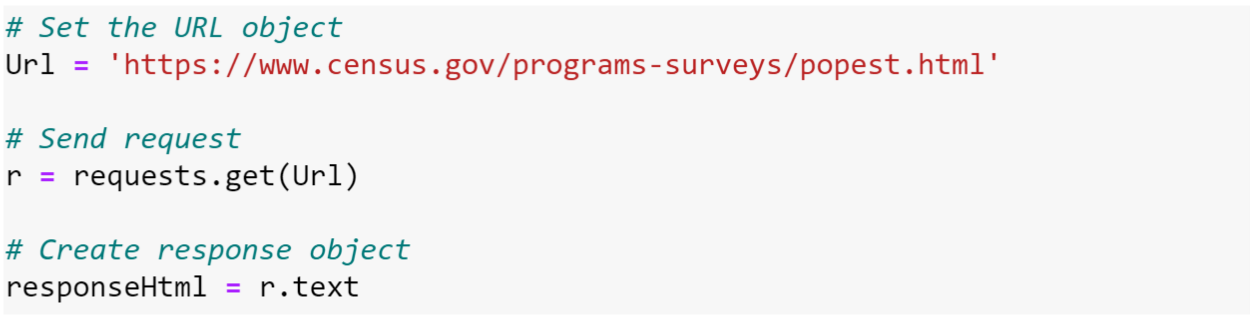
Western Governors University

PROGRAMMING IN PYTHON — C996

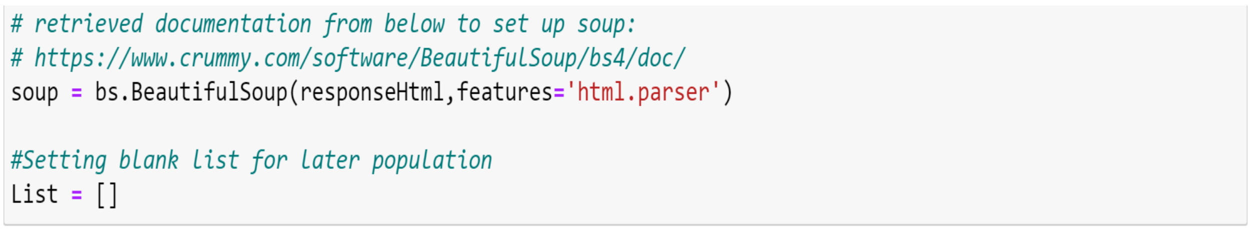
Below is the process and deliverables by which I scraped and summarized the data according to the task's request.

A. Explain how the Python program extracts the web links from the HTML code of the "Current Estimates," found in web links section.

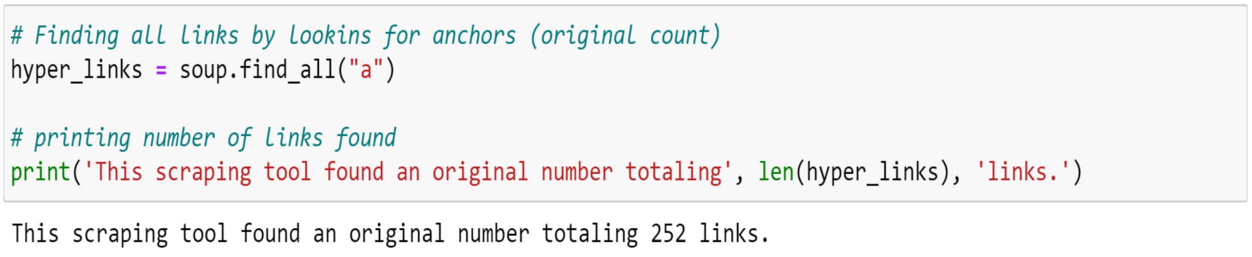
As with any typical web scraping the first thing to do is to define the URL. Afterwards, I used this location to request a call to the URL using the Request package. I was able to save this raw object and parse this html using beautiful soup (Richardson, 2019) and looking for anchor tags. Anchor tags help identify links which makes finding them relatively easy in conjunction with the beautiful soup package. Below are those same steps I took.



```
# Set the URL object
Url = 'https://www.census.gov/programs-surveys/popest.html'

# Send request
r = requests.get(Url)

# Create response object
responseHtml = r.text
```



```
# retrieved documentation from below to set up soup:
# https://www.crummy.com/software/BeautifulSoup/bs4/doc/
soup = bs.BeautifulSoup(responseHtml,features='html.parser')

#Setting blank list for later population
List = []
```

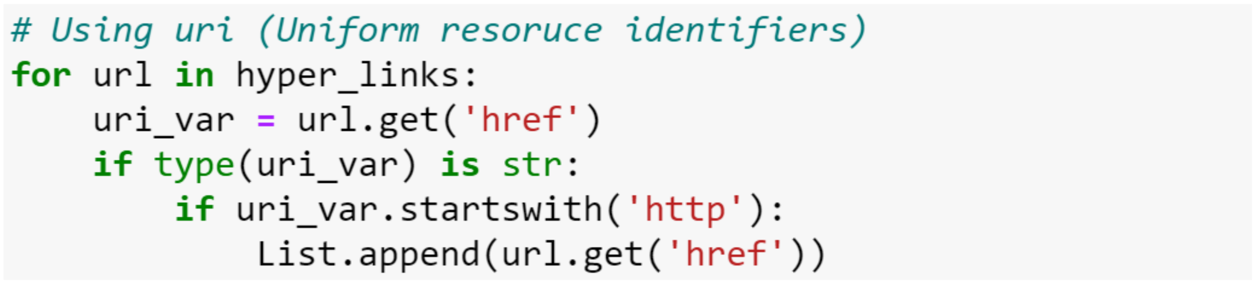


```
# Finding all links by lookins for anchors (original count)
hyper_links = soup.find_all("a")

# printing number of links found
print('This scraping tool found an original number totaling', len(hyper_links), 'links.')
```

```
This scraping tool found an original number totaling 252 links.
```

B. Explain the criteria you used to determine if a link is a locator to another HTML page. Identify the code segment that executes this action as part of your explanation.

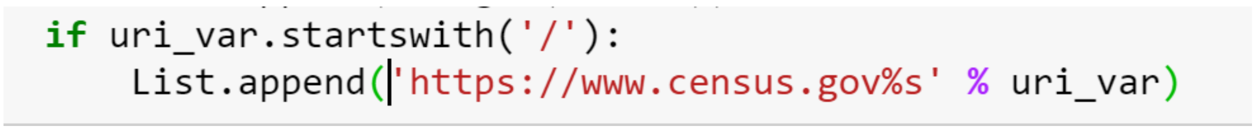
According to (Rajagopal, 2019) by iterating through the uniform resource identifiers and pulling all of the ‘Hrefs’, I was able to target anything that didn’t contain the .html or one that wasn’t specified at all.



```
# Using uri (Uniform resoruce identifiers)
for url in hyper_links:
    uri_var = url.get('href')
    if type(uri_var) is str:
        if uri_var.startswith('http'):
            List.append(url.get('href'))
```

C. Explain how the program ensures that relative links are saved as absolute URIs in the output file. Identify the code segment that executes this action as part of your explanation.

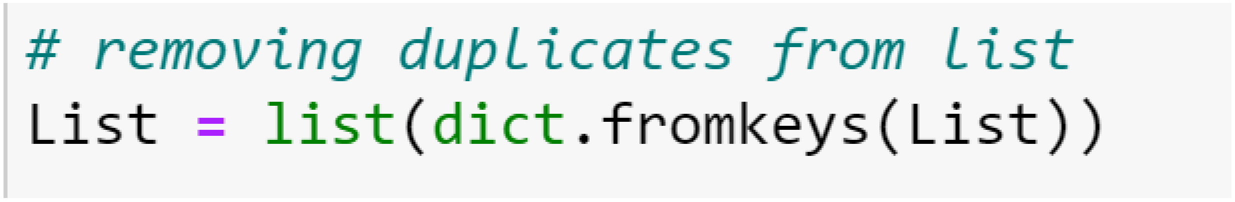
Links that are relative in nature in this file start with a ‘/’. There didn’t’ appear to be any that were filename only. Of course, any ‘hrefs’ that started with a ‘/’ were turned into an absolute URL by adding the census domain in front using simple concatenation. The code used to accomplish this is below.



```
if uri_var.startswith('/'):
    List.append('https://www.census.gov%s' % uri_var)
```

D. Explain how the program ensures that there are no duplicated links in the output file. Identify the code that executes this action as part of your explanation.

I have used a similar method to remove API duplicates in previous roles in data. Because Python doesn't really allow for duplicate keys with in dictionaries and easy conversation from the list to a dictionary was able to do the necessary deduplication process. Within the same line I was able to convert that same dictionary back to a list object, effectively not only removing the duplicates but keeping a consistent format.



```
# removing duplicates from list
List = list(dict.fromkeys(List))
```

E. Provide the Python code you wrote to extract all the unique web links from the HTML code of the "Current Estimates" (in the web links section), that point out to other HTML pages.

Please see Python_Scraping_C996.ipynb or Python_Scraping_C996.py files

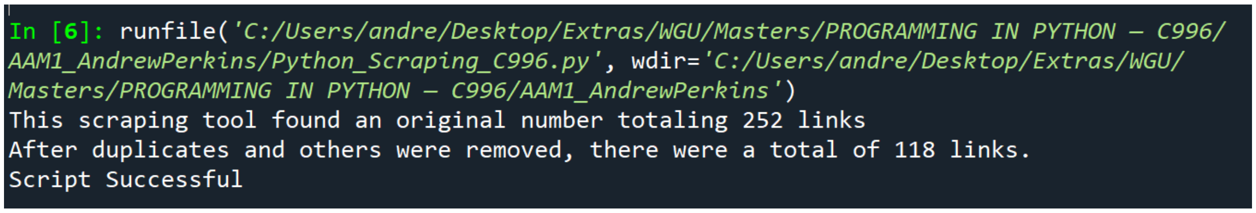
F. Provide the HTML code of the "Current Estimates" web page scrapped at the time when the scraper was run and the CSV file was generated.

Please see html_response.txt file

G. Provide the CSV file that your script created.

Please see uriList.csv

H. Run your script and provide a screenshot of the successfully executed results.



```
In [6]: runfile('C:/Users/andre/Desktop/Extras/WGU/Masters/PROGRAMMING IN PYTHON – C996/
AAM1_AndrewPerkins/Python_Scraping_C996.py', wdir='C:/Users/andre/Desktop/Extras/WGU/
Masters/PROGRAMMING IN PYTHON – C996/AAM1_AndrewPerkins')
This scraping tool found an original number totaling 252 links
After duplicates and others were removed, there were a total of 118 links.
Script Successful
```

References

Richardson, Leonard. (2019). Beautiful Soup. Retrieved from: https://www.crummy.com/software/BeautifulSoup/

Rajagopal, S. K. P. (2019, December 17). Web Scraping Using Python (2019). Retrieved from: https://dev.to/prsharankumar/web-scraping-using-python-2ip6#:~:text=get(%E2%80%9Chref%E2%80%9D)%20provides,URIs%20in%20the%20output%20file.