Census Bureau Web Scrap

Christopher Flowers

College of Information Technology, Western Governors University

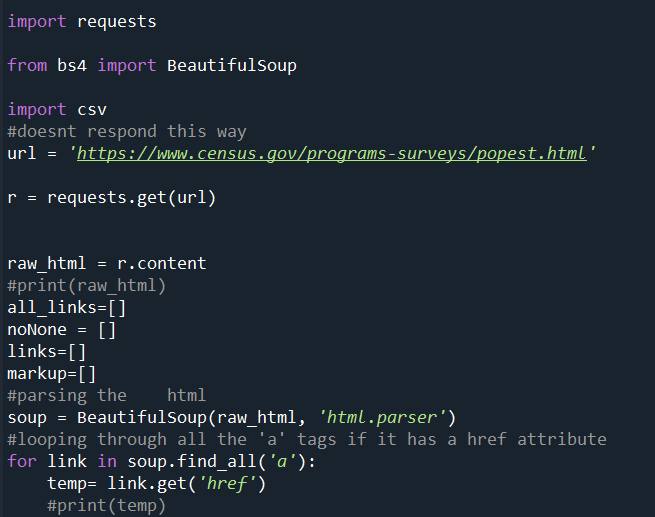
C996 Programming in Python

Dr. Sewell

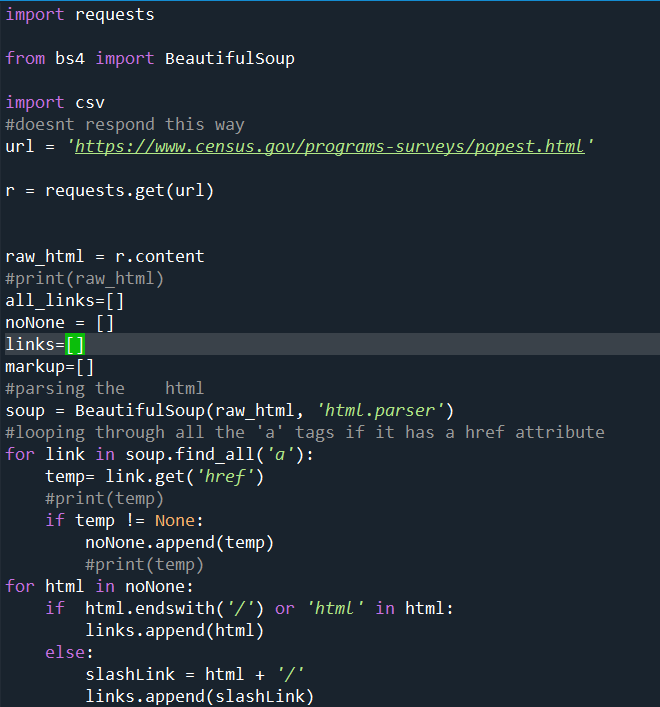
December 15, 2020

This course is programming in python and its objective is to get familiar with the programming language python. Python is different than most other programming languages as it doesn’t use curly braces for functions, conditionals and more. Instead, python uses colons, but it is also strict on its use of whitespace. Getting the whitespace right can be one of the most frustrating things in working with python. One of the main uses of python is for web scraping, which is scrapping a web site for specific things the user wants. The project for this course is to come up with a program which web scrap from the U.S. Census Bureau’s Current Estimates for all html web links.

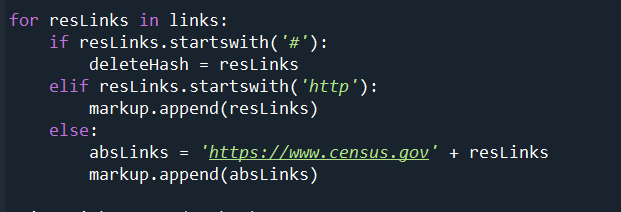
So, let’s start discussing how to write a web scrap program. Assuming you already have python downloaded, installed, libraries installed and the link to scrape. First step to scraping a website is to import the libraries installed, those are libraries to import are requests, Beautiful Soup and csv. Not all of the imported libraries are going to be used at this time but will be used later. Now we create a variable called URL and set it equal to the absolute web link we want to scrape in single quotes. For this project we use the following web link: ‘<https://www.census.gov/programs-surveys/popest.html>’. Now we want to request the URLs contents by creating a variable r equals requests.get(url) and the outcome of this statement is <response 200>. With a successful request of the URL, we need to get the html contents of the URL. This will be done using the r variable and its content section of the response, this creates a new variable. The new variable in the previous statement is raw\_html with its value being r.content, which is the html contents of the requests. Next create at list to hold the links, then create a variable called soup. Set soup to the imported function Beautiful Soup with the contents of the function being the variable raw\_html and in single quotes html.parser the attribute to parse the html. Using a for loop we will loop through link to find all instances of ‘a’ tags here href is true in the variable soup. The syntax for this for link in soup.find\_all(‘a’):. The first the thing inside this loop is where we will extract just the links, this is done by creating a variable called temp with it equal to link.get(‘href’). What this does is get all ‘href’ contents inside link and we have now scraped the weblink for all links. The code for this is lines 7 through 28 in the python script and can be seen the figure below.



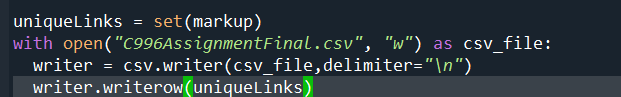
Next let’s discuss how to determine if a link links to another html page. The first thing to do is to is print the outcome of the temp variable to console, this allows you see what the data looks like so it can be cleaned. After looking at the data the next step is to get rid of the 'none’ values, using the temp variable from the paragraph above. An if statement will be used as a conditional statement to determine if temp is not equal to None. If the results are print the variable temp to the console to see if it’s the desired results, in this case we get the desired results. We add a variable called noNone with a value of an empty list above the declaration of the variable soup, now noNone will be used by appending the variable temp to it. Now we need to loop through the list noNone by using an iterating variable called html. Before entering the loop, we need to create a list variable to hold the links in the loop, this will be done just like noNone but will be called links. Inside the loop first check to see if the links are html pages or end with a slash, if it is true, we can append links to the variable to html. If false set a new variable slashLink equal to html plus a slash in single quotes, then append slashLinks to the list links. With this we have determined to if any of the links link to other html pages. This is lines 30 through 38 in the python script and the code is below.



Now let’s discuss absolute and relative URLs, absolute URLs are a full URL like ‘http://www.census.gov/data’ and relative URLs are just the file path for the URL like ‘/programs-surveys/popest/about.html’. We need to handle make sure all links are absolute URLs by making any relative links either into absolute URLs or get rid of the value. We will get create an empty list variable called markup and place it underneath the creation of the variable noNone. What is going to happen next is to loop through the links list this will be done by using a for loop and an iterator variable called reslinks. Once in the for loop we are going to test to see if the reslinks variable starts with a hashtag for each value in the links list. When this is true create a dummy variable called deleteHarsh to hold the values, but this variable will not be used because it is only holding values, we want to get rid of. If the first condition is not meet inside the for loop, we move to the next condition. If the variable reslinks starts with ‘http’ then we will append reslinks to the list markup and if the other two conditions are not meant, then we move to the else condition. Inside the else condition a variable will need to be created to hold the equation result called absLinks. The equation is the link ‘https://census.gov’ plus the resLinks iteration variable then we can append the variable resLinks to markup. We have now tested for if the link starts with http, hashtag or if it is a relative link and needs to be transformed into an absolute link. This will help sort through duplicates or error values in the list. This code is located on lines 41 through 48 of the code and a copy of the code is below.



So far, we have already scraped the web link to get our list of links, tested if the link is a link to another html file, removed the none values in the list, added slashes to the end of links if needed, removed links starting with a hashtag or http and changed relative links to absolute links. Now we need to remove any duplicated links inside the list. The list used will be the same one used above in saving our absolute URLs called markup. Python makes it easy to get rid of duplicated values in a list, it’s a single function. The function is called set and the variable passed to it will have any duplicated values removed from the list. So, use the set function with the markup list as the variable passed to the function this will be stored in a new function called uniqueLinks. To check all links left are duplicates we can print it to the console to a file. In this case it will be easier to print to a file, so we will write it to a csv file. We will use the with statement to open method with the file name and write as the parameters as the variable csv\_file. Now using the csv import mentioned earlier inside the with statement. We will use the writer function in the csv library with the csv\_file declared above with another parameter the delimiter attribute and delimiter are set to ‘/n’, which means it wants to place each value on a new line. Now we use the writerow event in the csv library, with the uniqueLinks variable as the parameter of writerow which will actually write the data to the csv file. The file in this case is “C996AssignmentFinal.csv” and when you open the file you have a total of 118 links. This is on lines 50 through 53 in the code and the code is also below.



In conclusion we were asked to write a program to scrape a web link of the Current Estimates section of the Census Bureau. In this paper we explained how to go about writing the program. It starts with importing the needed libraries then setting the link to be scraped set to the URL, the link is 'https://www.census.gov/programs-surveys/popest.html'. Then using the requests library get the response from the URL then grab its contents into a new variable called raw\_html. Using the Beautiful Soup library parse the html using raw\_html and ‘html.parser’ attribute as new variable called soup. Now we can loop through and find all instances of ‘a’ tags with a variable link, using get link variable to get just the ‘href’ attribute of the ‘a’ tag this is the first part of the web scraping. Then the list of links needs the value of none removed before going any further after running an if statement to filter out the none values into a new list. Next check to see if the links end with a slash or as an html page and if this is true append the iterator to a new list. When it is neither of the those two, we need to a slash to the end of the iterator. Now we need to make all the links to the absolute URL links by looping through all of the links list. Inside the loop if the iterator of resLinks starts with a hashtag when this is true, we need to set resLinks to a dummy variable. If it starts with ‘http’ then append the resLinks iterator to a new list called markup. When the iterator is not either use a new variable with it equal to ‘https://www.census.gov’ plus resLinks then append the new variable to the list markup. Now we want to remove all duplicates by using the set function in python with markup being passed into the function with this set to a new variable called uniqueLinks. Finally, we want to write uniqueLinks to a csv file this is done by using the with statement and opening a file called C996AssignemntFinal.csv with the write attribute being passed as a variable called csv\_file. Then using the csv import writer method with two parameters passed, csv\_file and the delimiter of “/n” which means, all this goes to a new variable writer. Using the writer variable with the writerow event having uniqueLinks as its parameter. This will write uniqueLinks values to csv file and will have 118 records.