Bradford Stallworth

Professor Jeyaraj PhD

MIS 3810

March 10, 2020

Extra Credit Summary

The extra credit activity was a fun and unique challenge. I coded in multiple python files, however, the codes I knew were correct I used in a final python file named “**50\_STATES\_EXTRACT\_ATTRACTIONS”**. Throughout the coding processed I learned a lot of skills and techniques. I believe the hardest part about the extra credit was the many ways to approach the extraction strategy and figuring out which method is the most efficient. Another challenge was extracted data from text without tags i.e. the address from every attraction. I do not believe the code that I have created is perfect, however, it was a great learning experience. I was able to successfully extract state names and attraction names. The address I could not figure out. Towards the end of the assignment I went back to my orginal **“OHIO\_HTM\_SCRAP.PY”** file after I created the CSV file from the combined 50 states and started to experiment with different outcomes. That is the reason the code inside the OHIO\_HTM\_SCRAPE.PY may be a little messy and not consistent with the final product.

Approach:

The first step in my thought process was to identify the difference between the url and the different states. The dedicated pages all have similar patterns. Here is an example of Ohio vs Indiana links.

**Ohio:** https://www.attractionsofamerica.com/attractions/ohio.php

**Indiana:** <https://www.attractionsofamerica.com/attractions/indiana.php>

Now that I have spotted the differences, I needed to use the libraries in python to extract and properly save them into a txt.file. The main Libraries that I used were:

1. Pandas
2. CSV
3. BeautifulSoup

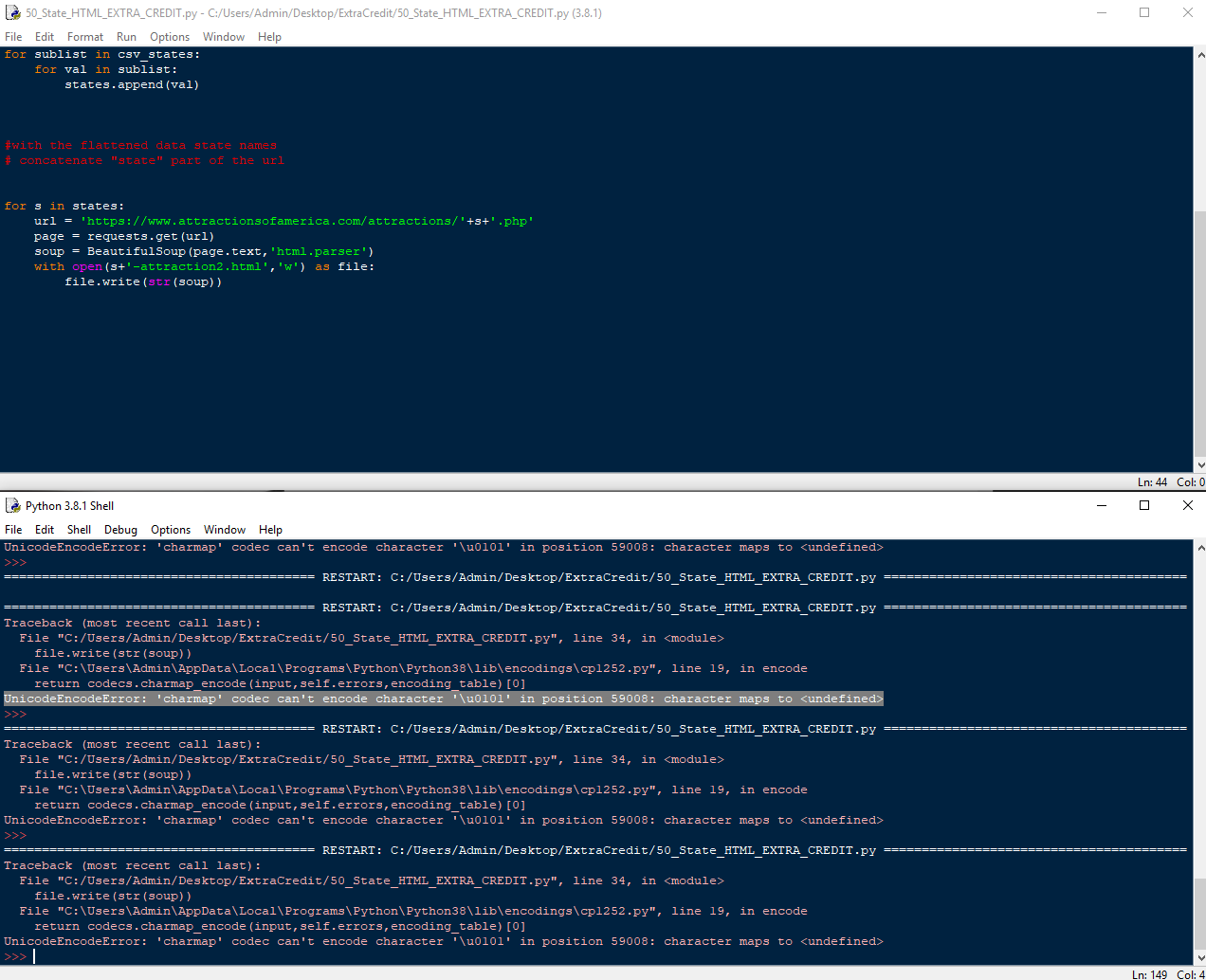
Once I extracted the correct text from the Ohio link. The next step is to put that function into a for loop and use string concatenation to insert the different states. Lastly, combine the extracted data and place them in a data frame using panda and convert the data into a txt file.

Major Challenges:

|  |  |
| --- | --- |
| Problem | Solution |
| Hard coding States is not allowed | Saved all 50 states in lower case in a csv file. And stored into list |
| Importing the csv file and creating a list caused a minor issue.  Incorrect result:  states = [ [‘ohio’], [‘indiana’], ‘[florida]’ ] | Remove the brackets inside the list by nesting list comprehension  states = []  for sublist in csv\_states:  for val in sublist:  states.append(val)  **correct result:**  **states = [‘ ohio’, ‘indiana’, ‘ florida’]** |
| couldn’t open CSV file | Import CSV function |
| Line Error from extracting multiple states | Encoding = ‘utf – 8’ in the open function fixed this issue. **See figure 1.1** |
| Extract the text of the webpage | **Using soup.findAll()** function. Find the tag of where the text you want to extract is located.  EXAMPLE: <h1> ohio top 20 attractions<h1/>  #example: ohio top 20 Attractions  title = soup.findAll('h1') |
| Ohio had 20 attractions while other states had 10. | Created a variable ‘repeat\_title’ and multiplied the results of the soup.FindAll(title)\*10   I did this so that the data is consistent. I was running into errors during the zip list process. I believe the data in each index have to be consistent in terms of items. |
| Drop the tags | Used lambda function |
| Could not extract the address text from the html webpage | I believe it is because it was a navigable string. I was able to PRINT the address but extracting it and appending to a list was a challenge I failed to solve. .nextsibling seem find the text within the <strong> tags |
|  |  |

Line errors from extracting multiple sites

Figure 1.1



Solution.

with open(s+'-attraction2.html','w', encoding = 'utf-8') as file:

CODE: FINAL CODE

1. #parse all 50 state html files
2. #By: Bradford Stallworth
3. #MIS 3810 EXTRA CREDIT
5. **from** bs4 **import** BeautifulSoup
6. **import** pandas as pd
7. **import** csv
9. #open csv file that contains lower cased state names
10. with open('50\_us\_states\_lowercased.csv', newline= '') as f:
11. reader = csv.reader(f)
12. csv\_states = list(reader)
14. #importing the csv file and creating a list caused a minor issue
15. # example [[ohio],[florida],[chicago]]
16. #figured out a way to remove the brackets inside the list by nesting list comprehension
18. states = []
19. **for** sublist **in** csv\_states:
20. **for** val **in** sublist:
21. states.append(val)

24. alldata = []
25. **for** s **in** states:
26. # open HTML page for all 50 states
27. with open(s+'-attraction2.html', 'r',encoding = 'utf-8') as file:
28. page = file.read()
29. soup = BeautifulSoup(page,'html.parser')
30. soup.prettify()
31. **print**(soup) #Why is it i take this away and it only prints one line?
33. #example: ohio top 20 Attractions
34. title = soup.findAll('h1')
35. **print**(title)
37. #repeat string
38. repeat\_title = (title)\*10


42. #Paragraphs
43. attraction = soup.findAll('h2')
44. **print**(attraction)


48. #address
49. results = []
50. **for** strong\_tag **in** soup.findAll('strong'):
51. results.append(strong\_tag.next\_sibling)
53. #drop tags
54. repeat\_title = list(map(**lambda** x: x.text, repeat\_title))
55. attraction = list(map(**lambda** x: x.text, attraction))
57. #combine data into a list
58. x = list(zip(repeat\_title, attraction, results))
59. alldata.append(x)
61. #flatten data
62. alldata = sum(alldata,[])
64. # convert into a dataframe and save to file
65. rows = pd.DataFrame(alldata,columns=['State','Attraction', 'Address' ])
66. **print**(rows)
67. rows.to\_csv('all-attractions.txt',sep='\t',index=False)

