**IS452: Final Project – Programming Project**

**Sangkyu Yi**

**4CR**

Instruction:

You should expect to submit: • your project’s github page link should be somewhere in your deliverable materials • a long narrative about your project, it isn’t substantive if you won’t have at least 1,000 words to say about it. • substantial code I can run (exceptions can be made) • a readme file describing how to run the code and an overview of how it works • a manifest file describing the different files and folder structure within your project • and have significant code comments to help me understand it • I need to be able to see your original and exported data, if that’s a part of your project.

1. **Github link**: <https://github.com/slyi2/IS452>
2. **Narrative**:

(Disclaimer: After multiple days of working on this project, I couldn’t get past few hurdles in making the code work, so my code is relatively short, and the result may not be as comprehensive and reflect the hours I’ve put into the project.

Therefore, I am including an extensive narrative with additional queries.

For some of the HTML data that I could not extract and include , I am including XPath queries to show that I’ve succeeded in locating the text I wanted to extract as written in my initial proposal, but failed miserably in converting that query into a code in BeautifulSoup…)

As a Compensation Analyst, one of the major job duties we do is job analysis. Having a comprehensive and updated job description is crucial in comparing internal jobs to external jobs in the market (even with same title) and take them into consideration in determining an appropriate estimate range of the base pay rate. While there are many resources out there, many refer to O\*NET to get a comprehensive overview of what the job is about. In order to efficiently extract data out of the website in a time-efficient way, I intended to create a python code that extracts key data from this website and export them to a csv file.

In order to extract the data, I used a library called BeautifulSoup, which is useful in extracting data in HTML format. Since we are familiar with XPath, I thought I could expand that idea and experiment with another way to extract data from the websites.

I first imported a module called requests so I could request to a web page and print the text. Then, I imported another library called bs4 within module BeautifulSoup to parse the data loaded from requests. Once the Modules were loaded, I set the variable response to requests.get() function which grabs the data from the url that I put in, and I also set another variable that parse that data loaded from request so I can properly extract the data.

The key data I wanted to extract was located in Job Zone Table of the website, which includes various key information about the job, such as education level, related work experience, job training, similar job examples, and SVP range. In order to understand the structure of the HTML, I investigated the tags and locations of this data using XPath initially.

It was in a table format, which made it easier to extract and export the data into the csv file. The data was under the tag ‘table’, but since there were multiple tables, I also had to define the attribute, which was summary=”Job Zone information for this occupation”. I assigned table variable which a soup.find() code to locate that data.

After grabbing the HTML codes that contains all the table data, I created an empty list called data, then created a loop to place all the data in the right location in the empty list. Under the table tag, the column titles and information pertaining to the column title were located under tr/td tag.

To get all the td tag data, I created another variable that locates all the td tag values and save it as a list using list(tr.find\_all(‘td’)) function. After that, I found out that the list was ordered in column title, then column info. Since the data I want were the contents of these tags, I located these texts and saved them into variables describing column title (column\_name) and column info (info) with tds[0].text and tds[1].text.

After locating all the data, I appended these items into the empty data list that I created earlier. I printed the data variable and made sure that the list was in the right format to be exported into a csv file.

In order to export that list into a csv file, I imported a csv module. As we have often done in class, I used outfile = open(‘filename.csv’,’w’) to define the outfilename (Job\_Summary.csv in this case). Then, I wrote out the csv\_out = csv.writer(outfile) so I can further define the row names. For the row name, I used csv\_out.writerow() function to define the row name as [‘Infotype’,’information’]. Then, once the row names were defined, I called in the data variable containing all the info I want in a list format. Finally, I closed the outfile by using outfile.close().

Initially, I also intended to extract more information under tasks and wage rates as a part of extraction, but there were challenges that prevented me from achieving this. One of the most troubling error that I constantly faced was this error:

TypeError: 'ResultSet' object is not callable

TypeError: unexpected EOF

My soup.find() function worked, but whenever I printed the results to check if I was extracting the right info, it kept giving me this error. I believe that it was not an issue with my code, since this code was running well until I started working on it again few days later with exactly the same code. Also, my codes were working on another person’s computer, but just not on my computer. I tried working on it on pycharm, and jupyter notebook, but I still got the same error.

Since I could not get the HTML extraction done, I have included how I would’ve extracted the data by showing XPath queries to traverse and extract data.

The list of additional data that I will include can be located by this XPath query:

*//table[@summary="Wages & Employment Trends information for this occupation"]//td[@class="report2b"]*

Which shows these 5 results:

Median wages (2019)

Employment (2018)

Projected growth (2018-2028)

Projected job openings (2018-2028)

Top industries (2018)

After I save those into one variable as a list,

I will locate the info for those columns with this query:

*//table[@summary="Wages & Employment Trends information for this occupation"]//td[@class="report2"]*

Which shows these 5 results as well:

$29.77 hourly, $61,920 annual

625,700 employees

Average (4% to 6%)

67,700

Administrative and Support Services, Professional, Scientific, and Technical Services

After saving them into another variable as a list

I would append them into an empty string like the previous table, and include them under the csv\_out.writerows().

1. **Readme**: In the github link
2. **Manifest file description**: in the github link