Python Web Scraping Tutorial Report: COMP390

Alyanna McGrath
mcgratha@oxy.edu
Occidental College

1 Introduction

I am building a more efficient and neatly organized webpage that presents the weekly menu at Occidental College's main dining hall, the Marketplace, often referred to as the "MP". This task requires scraping data from the current static page to populate the new webpage. A critical component of my project focuses on dietary restriction filtering, which enables students to filter the menu based on specific preferences such as vegetarian, gluten-free, lactose, shellfish, and more. A well-organized scraper capable of handling these specific requests is necessary to achieve this. While I am familiar with Python, implementing it for web scraping and utilizing built-in libraries is new to me. Therefore, starting the learning process early and focusing on it in this tutorial report [2] makes sense, especially since this task is a significant part of my project.

1.1 Tutorial Decision

The tutorial [3] I chose is Python Web Scraping Tutorial: Step-By-Step by Oxylabs. My selection process was twofold: first, I had to ensure that the tutorial was created more recently and by a reputable source that focused on the version of Python I had. Secondly, I began reviewing various tutorials I prioritized clarity, organization, and the inclusion of visual aids in my selection. Furthermore, I sought a tutorial that was accessible without needing paid tools or libraries, which was a crucial consideration for me as a student. The tutorial's section on parsing HTML will be particularly useful for extracting detailed dietary information, a crucial step for implementing the dietary restriction filters on the marketplace webpage.

The successful outcome of this tutorial is not merely teaching the ability to build a Python web scraper but also a comprehensive understanding of how to efficiently extract data using Python. By mastering web scraping, I will be able to automate the collection of menu data, ensuring the new webpage can dynamically update with accurate, dietary-specific options daily.

2 Methods

My process following the tutorial unfolded in three distinct phases to comprehend and implement the tutorial's guidance fully.

- **1.** First, I engaged with the material without actively coding, aiming to understand the necessary resources, files, and URLs required for preparation.
- **2.** During my second read-through, I identified potential deviations I would have to take from the tutorial due to the complex nature of the MP website's raw HTML and the specific intricacies of the web scraping I intended to undertake.
- **3.** The final phase of reading involved real-time coding alongside the tutorial.

As previously mentioned, I researched tutorials before making my final selection; during my skim, I noticed some creators used JupyterNotebook as their selected Interactive Development Environments (IDEs). Despite PyCharm being the recommended IDE to use, I opted for Jupyter Notebook. This choice was motivated by its capability for real-time feedback on code execution and the ease of visualizing and monitoring successes and failures, facilitating a systematic approach toward my goals. This deviation from the suggested IDE did not alter the programming foundation, as the core language remained Python. Choosing Jupyter Notebook over PyCharm and other IDEs aligns with best practices for interactive development, particularly for projects requiring iterative testing and immediate feedback, as emphasized in web development literature [1].

Continuing with the tutorial, I acquired foundational knowledge on web scraping and its applications and an introduction to crucial libraries such as Requests, Beautiful Soup, lxml, and Selenium. The tutorial featured a visual comparative analysis of these libraries, assessing them in a table based on ease of use, CPU and memory usage, among other criteria. I then integrated Requests and Beautiful Soup into my Jupyter Notebook setup. After establishing the scraper's architecture and initializing the soup variable—which pulled the raw HTML into my notebook—I chose not to use a WebDriver, which were the next steps in the tutorial process. I decided this because the project aimed to maintain simplicity and ease of use, ensuring the

tool could be easily modified and maintained. Incorporating a WebDriver would have added complexity, both in coding and in the understanding of browser interactions. By simplifying the technology stack, the scraper stays accessible for future adaptations.

Moving further along in the tutorial, looking further into the Developer Tools or right-click inspect element, I noticed a significant difference in the URL webpage they used and the marketplace menu's HTML, which was structured more convolutedly and less conventionally organized than the tutorial's example. I explored alternative data extraction strategies. Unlike the structured and easily identifiable HTML elements in the URL the tutorial used, the MP's menu data was enmeshed within a large, unorganized list of 'ul' tags, presenting a challenge for straightforward selection based on class or ID attributes, which are common in HTML. My approach evolved from attempting to select 'p' tags with enclosed bold elements to iterating over 'p' tags indexed by weekdays. Eventually, I developed an algorithm capable of parsing each line for a specified day in string format, extracting the entire block of text corresponding to that day's menu. This breakthrough led to me to create three distinct functions within the web scraper:

- `extract-menu-for-day` - The function utilizes a pattern-matching algorithm to scan through paragraph ¡p¿ tags within the HTML, searching for text that matches the specified day of the week and a date format. Once identified, using selective extraction, it flags the beginning of menu data collection for that day, aggregating relevant items until the end of the day's menu is detected or the next day's heading appears.
- `format-menu-by-category` - This function does not scrape the original URL; instead, it takes the raw menu items extracted for a specific day and organizes them into distinct sections based on predefined categories (e.g., "Grill Station", "Salad Bar"). It parses each menu item to identify meal periods (Breakfast, Lunch, Dinner) and categorizes listed dishes accordingly, enhancing readability by clearly delineating menu sections.
- `filter-menu-items` - This function, like `extract-menu-for-day`, refines the data extraction process to include only items that match specified dietary restrictions (e.g., vegetarian, gluten-free, lactose, nuts, etc). It filters the menu based on user-defined labels, associating each item with its day of the week. The output appears as the day of the week on the first line, and then the menu item with the specified filtration appears below, slightly indented.

It should be noted that the function names I created use Python standard naming conventions, with underscores instead of dashes. LaTex does not allow underscores unless used in mathematical formulas. The tutorial's final instructions are how to incorporate web-scraped data into csv and Excel files. While I followed these steps and successfully downloaded the data—specifically, the menu items labeled with dietary restrictions—onto my Mac, I saw no significant use for this feature, as the additional step of downloading and navigating data through static files like csv or Excel did not align with the primary objective of providing easily accessible and dynamically filtered menu information directly on the website. Having met my initial objectives, I sought to enhance the project with a user input feature for functions 2 and 3. By adding this I/O, I was able to test the scraper's accuracy specifically and get a first draft of its potential interactivity for the webpage. After completing the tutorial's final section, I had a functioning web scraper enriched with this personalized filtering capability.

3 Metrics and Results

Reflecting on the tutorial's effectiveness in teaching web scraping with Python and BeautifulSoup library, I found it particularly valuable. The tutorial not only introduced the basics of web scraping but also guided me through the practical implementation of a web scraper for my project. After completing the tutorial, I was equipped with a functional scraper capable of accurately gathering and filtering menu data from Occidental College's MP website, also focusing on dietary preferences. To evaluate the tutorial's success, I relied on three metrics: the clarity of instruction, user feedback, and the practical outcome of following the tutorial. The clarity of instruction was assessed by how well the tutorial communicated complex concepts, an evaluation grounded in my own learning experience and general best practices in instructional design. User feedback included my experience with the tutorial and its user functionality; being a CS student, I have a more detailed eye to be able to evaluate that, and the practical outcome was measured by the functionality of the web scraper I developed, specifically its ability to gather and filter data as intended accurately. I considering using alternative evaluative metrics such as the time required to complete the tutorial versus the actual time taken and the tutorial's stated prerequisites versus my prior knowledge. However, these were not primary metrics for my evaluation because the main goal was to complete the tutorial quickly and thoroughly understand and apply the web scraping techniques it taught. Based on these criteria, the tutorial has been both valuable and successful.

The article was concise, and although I occasionally had to revisit certain sections/scroll up to ensure accuracy in my coding, the instructions were clear and straightforward. The tutorial's author excelled in explaining the "how" and the "why" behind each step, aiding in a deeper understanding of

the concepts. More than the "how," the tutorial did a good job suggesting alternatives to steps, such as in the format "you can also do it this way." I also greatly appreciated the hyperlinked articles and links that took you to a more detailed description or instruction of the process of something mentioned that might have been unfamiliar. This approach proved invaluable for applying the learned techniques to my project, which presented unique challenges due to the complex structure of the target website's HTML.

User feedback on the tutorial, in this case, reflects my own experience following it and the logistics of how it was constructed. Adomas Sulcas, a Senior PR Manager at Oxylabs, authored the tutorial. Despite his advanced knowledge on the subject, I believe the tutorial was crafted to be accessible to anyone with any level of knowledge in computer science. The tutorial article is listed as a 16-minute read, and while the initial read-through, as discussed in the Methods section, might have been 16 minutes, following along with the tutorial in real time took almost twice as long. This was due to pausing at certain sections to code my parts, ensuring they worked before proceeding. The only constructive piece of feedback I have is the inclusion of more potential error scenarios. Instead of just listing instructions, the section of code, and how everything should look when done correctly, adding what might happen if something is done incorrectly or a section about frequently encountered problems could make the tutorial even better than it already is.

The practical knowledge gained has been instrumental in advancing my project. Not only did I develop a web scraper that met my project's specifications, but I also acquired a versatile skill set applicable to future projects. The ability to adapt the tutorial's principles to the specific requirements of my project highlights the universality and adaptability of the skills taught.

In conclusion, the tutorial's emphasis on clear instruction, positive user feedback, and successful application of its lessons underscore its effectiveness. Despite variations in project scope and complexity, my ability to transfer the learning effective solution that will remain my project attests to the tutorial's practical value and the universality of the coding skills it imparts.

4 Reflection

I feel I've moved closer to understanding my comps topic's intricate aspects. Initially, the topic's excitement and positive feedback from peers and professors fueled my enthusiasm. However, the thought of actually building it seemed daunting. The task felt overwhelming as time passed until I began taking concrete steps toward the overall goal. Finding a suitable tutorial for web scraping was challenging. I struggled to find solid documentation that suited my needs. Hence, my primary concern is the potential for changes to the college's website in the near future. I worry that if the webpage hosting the MP menu changes, my Python web scraper will need to be adjusted significantly. Through this project, I've grown confident in parsing data and scraping for specific nutritional preferences. My background is mostly in front-end development, focusing on creating engaging user interfaces with HTML, CSS, and JavaScript. I still have slight doubts about managing the back-end, particularly with functionalities like favoriting menu items and efficiently extracting menu data; however, I know that once started, I will problem-solve these issues with no impact on the overall success of the project. One last concern is ensuring the project's sustainability once I graduate, in case the format of the menu or the website changes. Adapting the web scraping application to accommodate such changes is essential for long-term viability.

In conclusion, the tutorial has helped me better understand not only my technological capabilities but also the impact of what a project like this might bring. My concerns about adaptability and sustainability loom with significance in creating effective solutions that will remain effective beyond my involvement. Nonetheless, I am even more passionate about my comps project, which is to create a better way to view Occidental's MP menu daily, emphasizing filtering for nutritional preferences.

References

[1] Clarke, Madeline. *Jupyter Notebook vs PyCharm: Software Comparison*. 2023. URL: https://www.techrepublic.com/article/jupyter-notebook-vs-pycharm/.

[2] McGrath, Alyanna. *Python Web Scraping Tutorial Report: COMP390*. 2024. URL: https://www.overleaf.com/read/ctqcnmhqkzjz#b37597.

[3] Sulcas, Adomas. *Python Web Scraping Tutorial: Step-By-Step*. 2024. URL: https://oxylabs.io/blog/python-web-scraping.