Introduction

We decided to create web scraping project using the Python language and the Beautiful Soup library. Web scraping is a form of data scraping that is used to extract data from websites. Some uses of web scraping are monitoring customer sentiment, price monitoring, news tracking and so on.

Technical Approaches

Beautiful Soup a python library that extracts data from HTML and XML files. It uses the source code of the file to create a parse tree to extract the data in a hierarchical and readable fashion. In the case of our code, beautiful soup extracts the price, the rating, number of ratings, and availability of the product from the Amazon page. We then store that information in Python lists (equivalent to an array in C, C++, Java, etc.) in a for loop that iterates over the number of links in the list of links. Next, we store the price, rating, number of ratings, and availability of the product in dictionaries where the key is the title of the product in for loop that iterates over the number of products in the list of product names. We sort the dictionary of {title: price} from lowest to highest price and the key-value pairs in a new dictionary. Next, we create the message that will be sent as an email. We did this by creating function. The message is originally initialized to an empty string. A for loop that iterates over the dictionary of title:price key-value pairs sorted from lowest to highest. In this loop, we have five strings that make up the message for the information for each product. Each string is interpolated into the next string. Each string has one newline character at the end of the string for readability except for the fifth string. The fifth string has two newline characters in order to separate the product information of the next product. The function prints message sent to the console to signal that the message has been created. This was used for debugging purposes. Lastly, we had a while true loop. In the loop we stored the product with the lowest price. We call a function that calls the functions that get all the product information again. We store the most recently calculated lowest price in another variable. In a if statement, we compare the former lowest price product with the current lowest price product. If the current lowest price is different (higher or lower) than the previous lowest price, we create the message again and send the email again.

Flowchart

Code starts -> Get request is sent for the amazon link -> beautiful soup parses the HTML website -> Store product information in arrays and dictionaries -> sort products by lowest price -> create email message -> send email-> update message if lowest price has changed -> loop back to create email message -> send email

Could add functionality to see if the product with lowest price currently is the same as before and if it is not