Pedro Vincenty

Web Analytics

Street Easy Navigator

** The program has difficulty running after the first few iterations. Clicking the search button has yet to work the first time. I haven't been able to find a solution in time for submission but will take this project forward outside of class to fix this dilemma.

*** Run program. Search won't hit first time. Run program again. Will get bot detection. Run the block of code that creates the web driver to waive bot detection. Run program again 3rd time for successful cycle.

One of the most useful tools the course provided me was an introduction to Python's Selenium Package. This package allows me to interact with my web browser so I may parse and collect data from the internet. This interactive package and the experience I gained through this course runs parallel to the full stack course I'm currently enrolled in on Codecademy. I'm independently learning about JavaScript and how it adds functionality and interactivity to web pages, and this course has helped me see other possibilities of interacting with the web. To demonstrate this capability, I created a program that parses and collects apartment listings on StreetEasy based on a user's preferences.

With Selenium I'm able to load, send, and click on data with google chrome. Quickly after beginning my project and establishing a connection with street easy, my page would verify that I wasn't a bot and ask for me to click and hold on the verification button. The program must be restarted every time this occurs. I've cited the article where I've implemented the code to avoid bot detection, but the dilemma persists. As long as we're avoiding bot detection, we may proceed with the program.

This program was great practice for getting familiar with selenium. As I'm navigating through Chrome's DevTools, I'm able to determine which XPath to create so I may interact with the website accordingly. There's something I discovered while trying to locate certain elements within the HTML structure of the page – there are eventListeners such as focusout and blur that keep certain elements hidden while trying to navigate towards them on Chrome's DevTools. If I

disabled them I'm able to access the elements that would hide from the Dev Environment, namely the drop-down values of 'max prices' and the list of neighborhoods that come up in the neighborhood search bar. When I first encountered this issue I didn't know what an eventListener was and was able to find my answer of disabling it while googling around for a solution. I later learned about eventListeners in my full stack course and how they are related to javascript's interactivity with HTML elements. This combined experience was a great learning opportunity and helped me understand details and behaviors in web development.

There were some instances where clicking on an element would give me an 'ElementClickInterceptedException' error and it seemed like I kept having to alter the xpaths to the button I was trying to click. Normally I use the syntax 'driver.find_element()' to find the element I'm looking to click on, but locating the parent's element would help me explicitly solve this issue. Once I was able to find the parent element of the element I was trying to click on I would then use the following syntax to click on the child element 'parent.find_element()'. This idea came to me as I was having difficulty sending the value key for neighborhoods into the search bar. For some reason the program wouldn't sent in neighborhoods or boroughs with one name (Tribeca, Manhattan, SoHo) so I had to think of another way of accessing the xpath of the neighborhood search bar and clicking its elements appropriately. This discovery was incredibly helpful and a great exercise to understand the different ways of accessing HTML elements with Selenium.

I want my program to be as user-friendly as possible, so there are a few practices I implemented to make the program as strong as possible. First, when the program prompts the user to enter their budget it rejects anything less than 500 USD as there is no listing option on StreetEasy below that value. Next, when the program prompts the user to select the number of bedrooms it will only accept an integer value less than 5 and the regular expression value 'studio'. The program also accepts a list of neighborhood's separated by ', ' inputted by the user. StreetEasy has it's preset intervals of budget, but what if a user's budget was in between these values? This program accepts the inputted value and rounds up to the next listed interval so that all listings fitting the user's budget will be captured. The program returns only the listings that are equal to or less than the user's budget. e.g. If a user's budget is $2300 the

program will set the search for listings at $2500. The returned data frame will only include listings at $2300 or lower.

This project has been a great opportunity to practice web parsing with Selenium. There is room for scalability as I could incorporate for added preferences such as amenities and pets. I'm grateful to now have selenium in my toolkit for my future academic and professional endeavors.