Michael O’Hanlon & Aughdon Breslin

Learning Module #5 Python/Browser Automation

Initial Proposal

Michael and Aughdon want to learn Python because it is a highly impactful language to know and is applicable in many projects. If we are able to code in Python and do something even basic with it, we will have developed a strong foundation for a coding language that we will certainly use many times in the future. While we are unfamiliar with how expansive our knowledge of Python can become within this module, we’d like to automate browsers. Automating browsers essentially means going to given websites and performing actions without human interaction, such as automatically posting a tweet, sending an email, and more. Specifically, we would like to open a browser to Google, create a search query, and then open a page from the links, and potentially login to that website once we have reached it.

In order to do this, we will probably use the program Selenium to aid in performing the functions on a Chrome browser, and YouTube and other media sharing services to research tutorials and guides to get us started. Selenium is an open source tool which we can use to automate the tests carried out on web browsers, so working with it will be a large portion of our learning module. However, both Michael and Aughdon will research how to use Python and how to incorporate it with Selenium. We will both be involved in the scripting and troubleshooting of our project. Days 1 and 2 will largely be based around searching for guides and tutorials on Selenium. Days 3 and 4 will be about creating basic functions that Selenium can perform and familiarizing ourselves with the different syntax and formatting of Python. Days 5 and 6 will be creating a script that has multiple steps, such as opening the browser and searching something up and clicking on a result. Days 7 and 8 will be troubleshooting our script as well as trying to add to the script to login to a website.

1. Day 1 Log (11/18/19): We are going to use PyCharm as our IDE and have set up Python as the language we will use within it. We also found out how to implement the Selenium WebDriver for Chrome and import it into PyCharm. We will use PyCharm to create the code under Selenium that it will execute in our web browser. Within the ChromeDriver class we discovered the constructors and useful methods such as launching Chrome and getting a website.
   1. Automating Browser Intro
      1. <https://towardsdatascience.com/controlling-the-web-with-python-6fceb22c5f08>
   2. Selenium WebDriver
      1. <https://docs.seleniumhq.org/>
   3. Guide for things to do with Python:
      1. <https://intersog.com/blog/some-cool-things-you-can-do-with-python/>
   4. Tutorial on learning Python Selenium
      1. Selenium Python Beginner Tutorial - Learn Selenium Python in one video | Step by Step
   5. Download Python as WebDriver Language Binding
      1. <https://www.seleniumhq.org/download/>
   6. This video uses Python, PyCharm, and shows how to login to facebook
      1. <https://www.youtube.com/watch?v=oM-yAjUGO-E>
2. Day 2 Log (11/29/18): We started off class watching a basketball video. More importantly, it was about using enormous data sets (big data) to identify patterns in movement, such as plays and variations of plays, and using various factors to predict the likelihood of a successful shot. It also learned more than any one coach would know by processing all of this data and figuring out the probabilities and more successful strategies at given moments. For our learning module, we continued watching videos on ChromeDriver and Python, finishing the downloads for python 3.7.1.
3. Day 3 Log (12/3/18): We are struggling to get around the administrator’s restrictions as both command prompt and run are disabled. We need command prompt in order to be able to open the Chrome web browser from PyCharm, and to interact with the browser automatically. However, Mike found an ingenious solution of using a BAT file, which “consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file” (<https://en.wikipedia.org/wiki/Batch_file>), and typing in “pause” “dir” then “pause” again. It successfully executed a command in the command prompt (it opened up the directory and paused), however when we clicked again it still force closed. The same applied with variations such as “cmd,” except it will just state that its blocked and then close. Today was rough.
   1. (12/3/18) Aughdon looked into other things we can do with Python and browsers that may not be as extensive and would not require making the computer do the operations for us. I found quite a few articles on how to send emails with Python, articles that would still help us learn the language but that would not require as much access to the computer’s control panels and system settings (8:00-9:00). For this plan, these articles should help a lot.
      1. <https://docs.python.org/2/library/smtplib.html>
      2. <http://www.tutorialspoint.com/python/python_sending_email.htm>
      3. <http://www.pythonforbeginners.com/code-snippets-source-code/using-python-to-send-email>
      4. <http://www.blog.pythonlibrary.org/2013/06/26/python-102-how-to-send-an-email-using-smtplib-email/>
      5. <http://naelshiab.com/tutorial-send-email-python/>
4. Day 4 Log (12/4/18): We looked into what we could do without accessing the command prompt, and have run into more trouble importing the webdriver from Selenium. We spent a bit of time checking the placement of all our files to make sure the locations we are referencing are containing the file, but have yet to solve the problem. Near the end of class, however, we found an article that might be able to help.
   1. <https://intellij-support.jetbrains.com/hc/en-us/community/posts/360000436264-Importing-webdriver-from-selenium-cannot-find-reference>
5. Day 5 Log (12/5/18): Following the advice of the article, we started off uninstalling and reinstalling Selenium. This fixed the error we had, but replaced it with a different error, one that needed access to System Settings to be fixed. We decided to start from scratch using a new computer with no restrictions. We then began the installation process of Python and PyCharm all over again. After encountering some more errors, we managed to figure out the version of python we had. After some difficulties, we began to edit the environment variables.
   1. (12/6/2018) Michael watched YouTube videos pertaining to using Selenium with Python and Selenium, as well as videos on basic coding in python. (7:00-8:00pm).
6. Day 6 Log (12/7/2018): Aughdon was absent today as well as Mr. Detrick. After starting from scratch on the new computer, we arrived at the point in which we were stuck before. With no restrictions, I was able to access the system settings in order to change the environment variables. Because the tutorial is for Windows 10 and our laptop is Windows 7, the procedure is slightly different. After coding our sample program to open a website. I encountered a similar error that we had experienced on the other laptop. This error was a WebDriverException, claiming that our chromedriver is not in “PATH”. After googling this error, I encountered many articles and forums in which almost every person was claiming a different method worked for them. Overwhelmed, I tried some of the simple fixes people have claimed to work for them, such as restarting PyCharm, and restarting the laptop but neither of those worked.
   1. ADV TOPICS PC #1
   2. **Put chromedriver path in PATH system environment variable**
   3. (12/8/18) Aughdon looked up YouTube videos to figure out what the command prompt’s main purpose was for this project and what we could skip or do that wouldn’t require complete access to the computer (1:00-2:00pm).
   4. (12/8/18) Michael practiced using Selenium and Python at home by downloading them onto his home computer and testing different methods/commands. (7:00-8:00pm).
7. Day 7 Log (12/10/18): We began class with a well deserved ice cream party. After eating a lot of it, Michael got to work on the project because Aughdon was absent again. Unfortunately, the computer we were using was not charged when I walked into class so I had to charge it. While it was charging, I ate ice cream and did research on the error I had encountered in the previous class. I discovered multiple ways in which the error was fixed for other people, so at least we had multiple options to resolve our issue. After watching a tutorial on using Selenium and Chromedriver in python, I discovered that the person was using many different methods and imports than we were. After adjusting my code, I encountered error after error. All of the errors were different than our previous “chromedriver needs to be in path” error. After resolving 3 errors, I was hit with the error that had previously plagued us. Another day passed, and little success was had.
8. Day 8 Log (12/11/18): Aughdon was late to class today and we are still running into these errors. However we decided to go with the automatic sending of an email and were able to make some real progress. We achieved a successfully sent email with a subject header, message, and sender and receiver names. We are now working on automatically signing into Google Mail and opening up the email we receive. We are currently trying to enter our valid email and username into the browser.
   1. (12/12/18) Aughdon researched how to input text into a space on a webpage, such as typing in login info to access media sharing sites and emails (8-9:30).
      1. https://jszapp.com/how-to-create-a-shortcut-that-automatically-logs-in-to-any-website/
9. Day 9 Log (12/13/18): Today was amazing. We were able to identify the slot for a username with ease and the script would actually type in the email automatically. Seeing something typed by a computer is pretty crazy. Then we found that there was no id value for the password. We got stuck trying to figure out how to allow the computer to recognize where to type our password in. We inspected the password box and were given a bunch of variables we weren’t really familiar with, but no id to search for. Instead of finding it by id, Mike came up with an idea of searching through another variable, like by class. After thinking about viable variables to search for, we came across searching by name value, and that was the one we needed. We were able to go to the google login page, type in a username, then a password, and login to successfully view our email without doing anything but running the program.

Final Write-up

Michael O’Hanlon and Aughdon Breslin worked on using Python to develop a script to automate browsers. This means we write directions for a program to navigate the internet, filling in information and clicking on links/icons automatically. At first, we did not really have a specific goal in mind, so we looked up a lot of articles on beginner Python projects for newbies and scrolled through some generic topic ideas. We came across the automated browser idea and obviously it sounded really cool, but we had trouble understanding just how operating an automatic browser would even work. We searched on YouTube for answers, and came across a half hour long video that claimed to achieve all of this step-by-step, called “Selenium Python Beginner Tutorial - Learn Selenium Python in one video | Step by Step.” It was at least a good start, showing us what we needed to download to get started, but despite watching it about 30 times over, we never got past the first five minutes. It seemed Raghav Pal was using a command prompt, but we didn’t know how to open one, or when we did find out, we didn’t know how to get past it being blocked by the administrator and immediately force closing.

We were really frustrated because pretty much the entirety of three work days was spent trying to get around this, and we started thinking of some backup plans since nothing seems to ever flow smoothly. Aughdon found some great articles on sending emails with Python, and we were able to get something on the accomplishment list before the learning module was over. However, it didn’t end there. Despite it being the last day of the project, we were blessed with an extra day, and so we capitalized on it, and resorted to Java to see if we could do anything with automation whatsoever. To start, we watched tutorials on using Selenium with Java as opposed to Python. The basic commands were the same, so our brief experience of Selenium with Python helped us jump-start our success with Java. Getting start, we created a new Package for Selenium, and then created a new Class for our first program. In order to use all of the functions of Selenium, we had to update the build path of the package to use Selenium, which was fairly straight-forward. Once the initial setup was completed, we were able to begin coding our first automated browser program. Using methods such as get() and findElement() we got started by typing in a Google search. We then decided to try to do something more in depth by logging into a GMail and opening an email in the inbox. After struggling to inspect elements on Google Chrome and find the correct IDs of the elements we wanted to click on, we began implementing our newly found element IDs to create a path towards logging in. We encountered a problem however, not all elements we needed to manipulate had IDs attached to them. To counter this, we discovered that we could use the name and class of the element to find it and click on it. We had to use this in order to click on certain objects. After a lot of fails, we eventually were successful in creating a program that would open Google Chrome, login to GMail and open an email in the inbox. That email however would be an email that we sent from another email using our program in Python.

We learned a lot about using code in two languages to automate everyday tasks on the web, such as sending an email as well as logging in to a website. We learned basic coding and syntax in Python, as we were both not very familiar with the language going into this learning module. If we did this learning module again, we would try to not be so particular on using Python for the entirety of our module, as it delayed us heavily and took up a large majority of our time. For other groups wanting to do a similar learning module, we recommend you stay away from Python when trying to do browser automation, as you will encounter a hefty amount of errors, and some will be seemingly impossible to resolve due to lack of information or clarity. Additional things we could pursue from this learning module is to use voice automation to do the launch the browser automation code, the email sending code, or both. Also, we could look into using the code in coordination with each other to have once code call the other one.