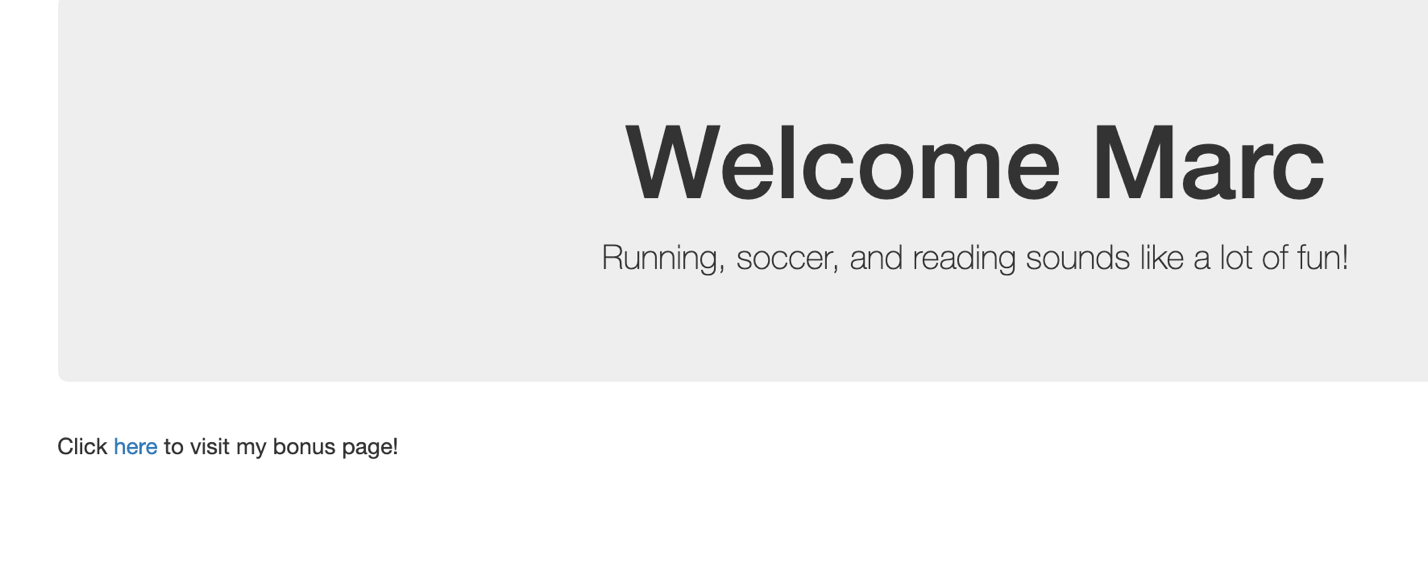
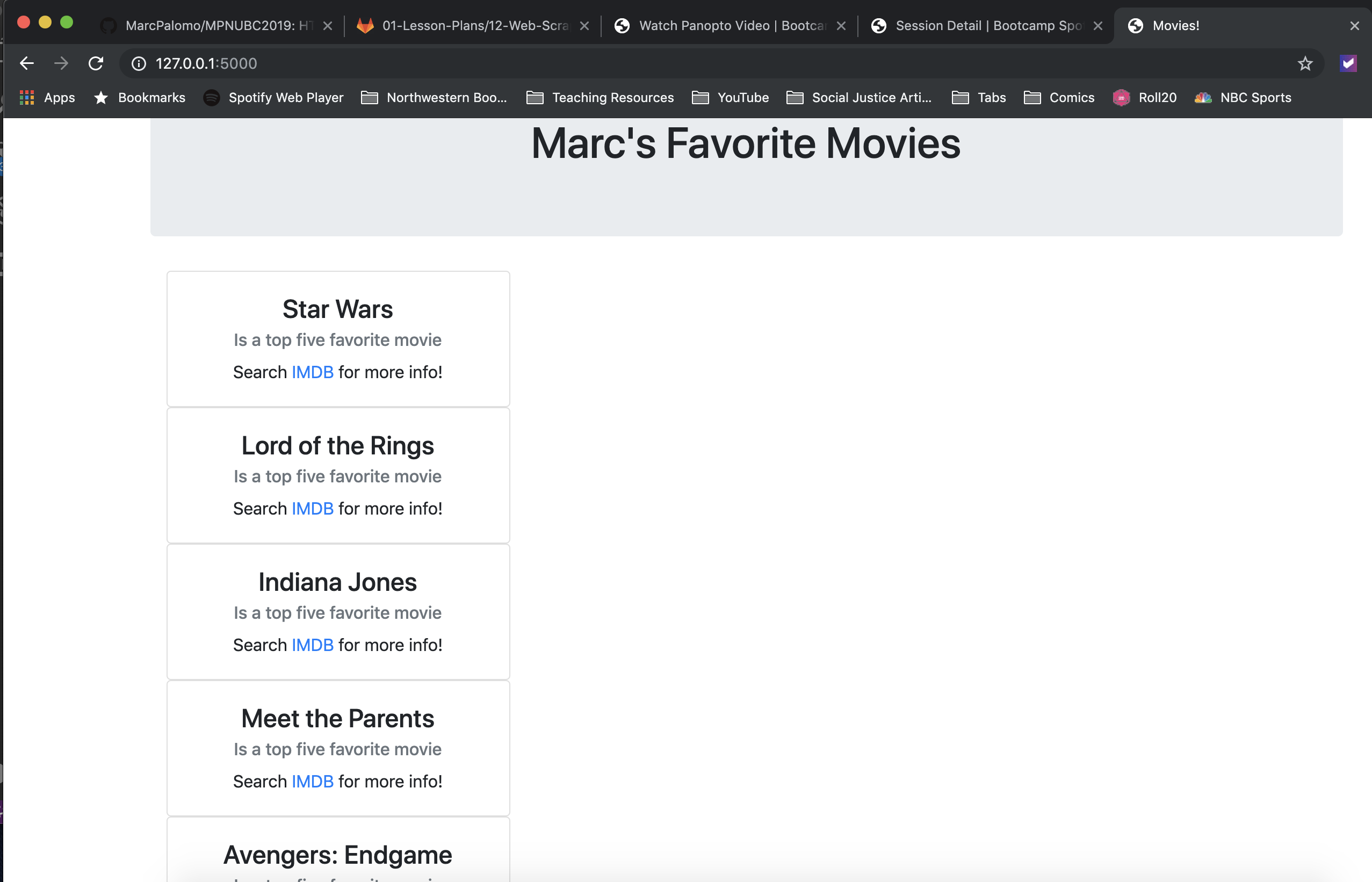
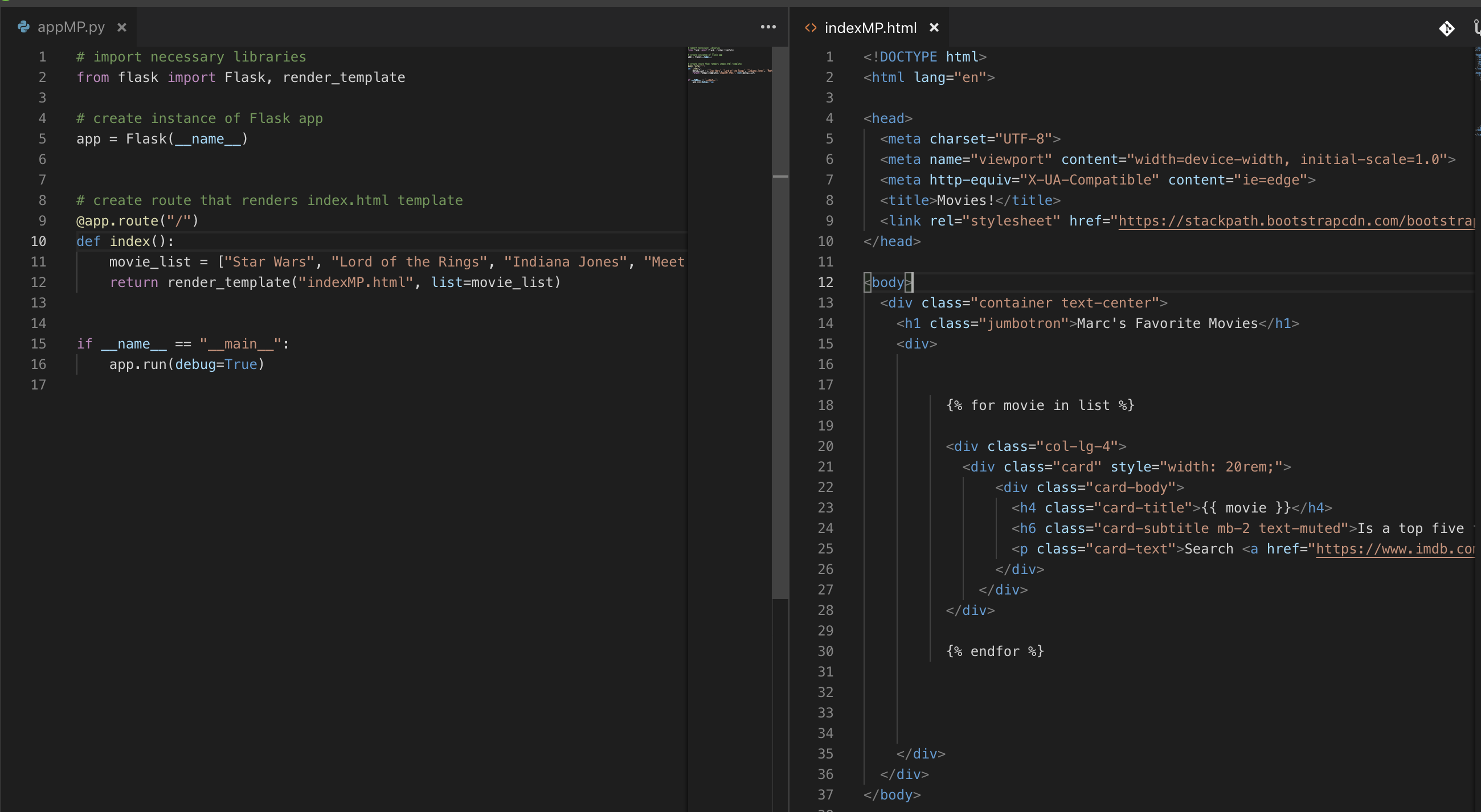
The first activity was new material and the student activity was skipped, but I went ahead and completed it because as he went into the second instructor lead activity, I was a bit lost on what was happening:



What I came to understand was the HTML file is in a way prioritizing the flask file that it is being run through terminal. Or I guess the flask file is forcing the HTML to run with the information populated from variables in the flask file. You can create variables that will imbed into the html document without actually changing the HTML at all.

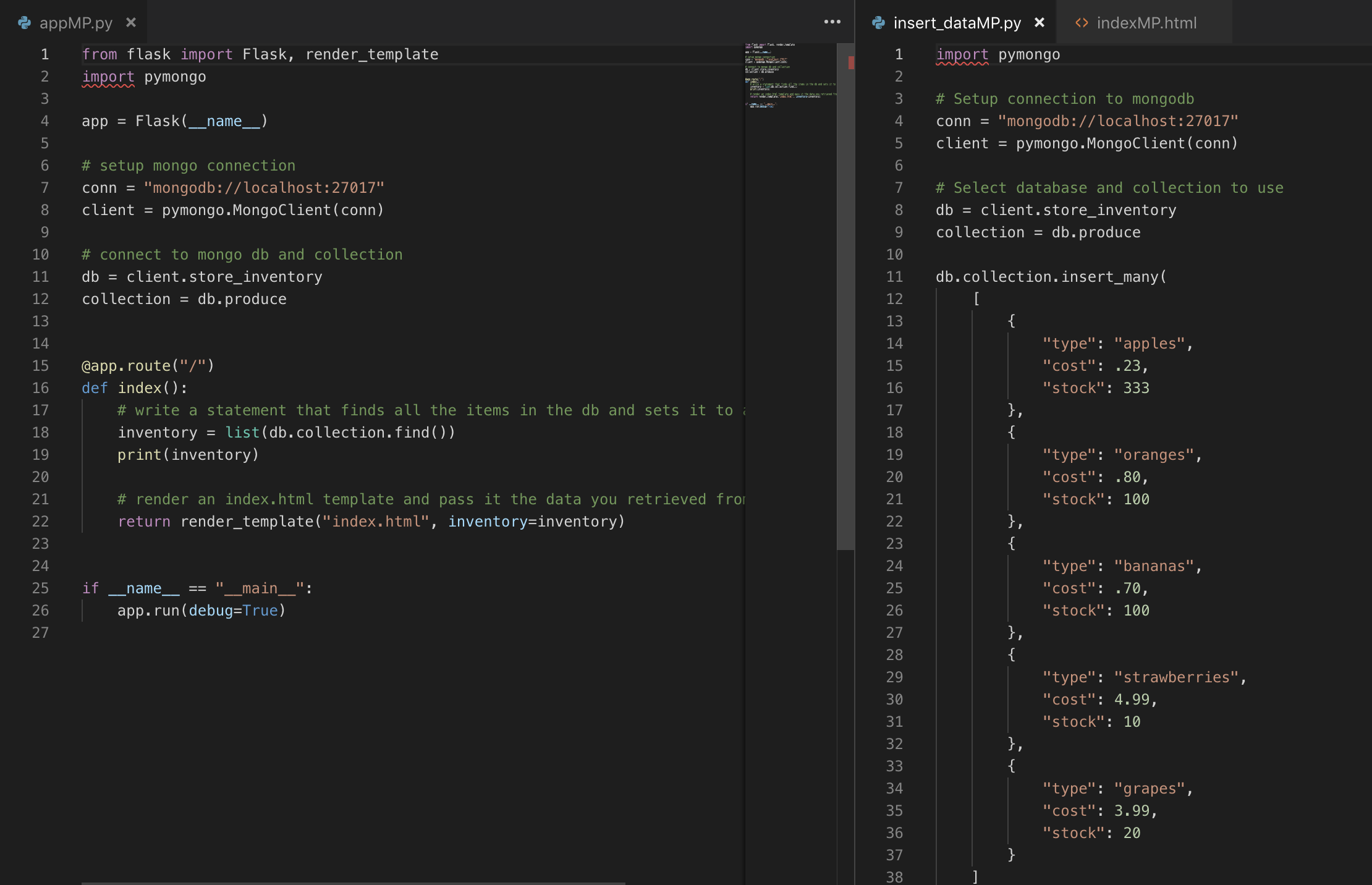
The actual first activity done was 04-Stu\_Render\_List where we employ this concept in a loop to get multiple variables in a list on the HTML. Once I got the idea, it was pretty straight forward to implement the code in the flask .py file. I took some of the answer code in the html for the bootstrap additions and the exact column sizing to match the answer. I also manually changed the tab title from “Hurricanes” to “Movies”.

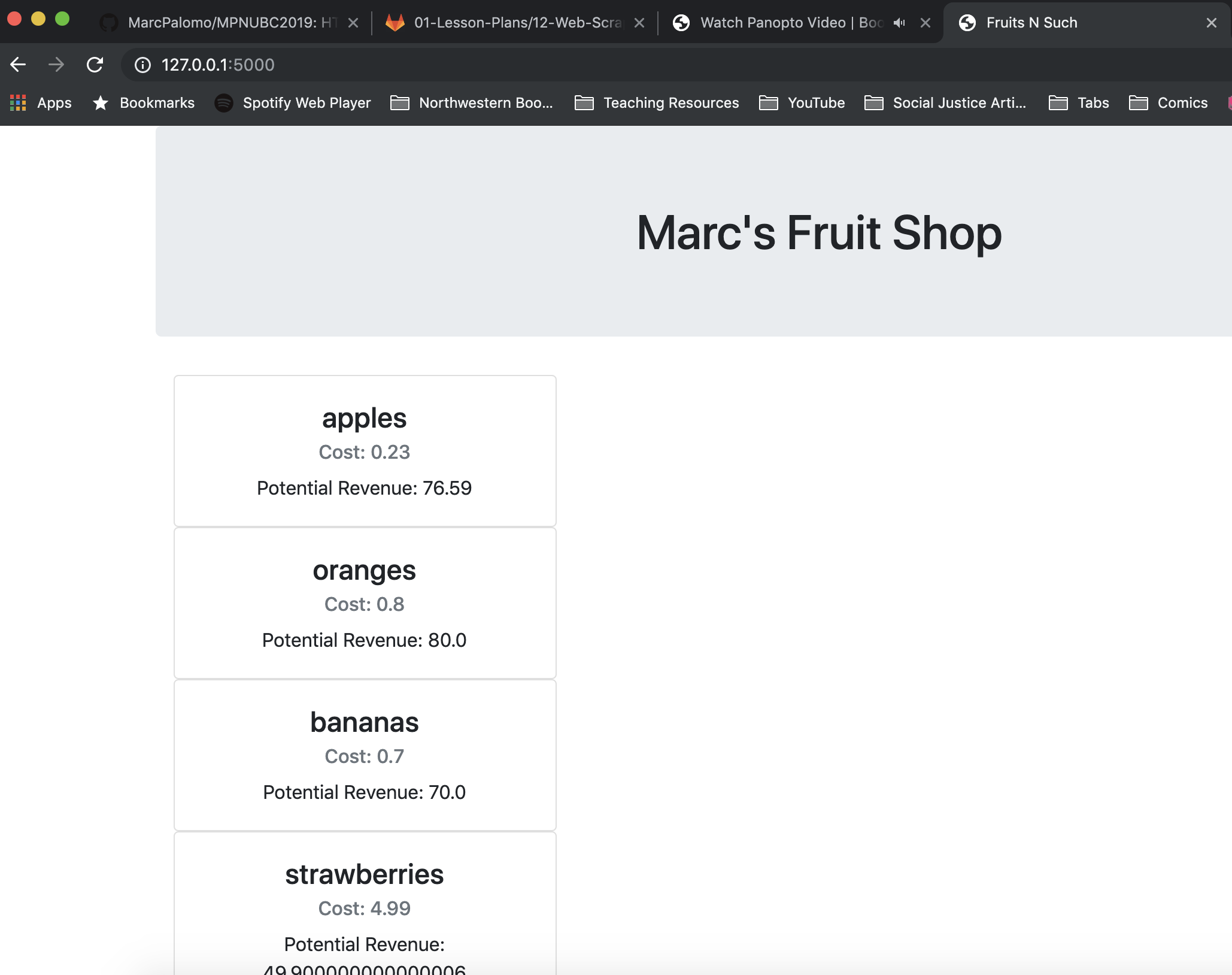




The next instructor activity and student activity was skipped because it repeats the same concept but runs through a list of dictionaries. The Daisy and Chris can be heard during the work time explaining they’re going to skip the activity in favor of giving more time for the remaining activities.

Mongodb will be used for the remainder of the activites. The concept continues to build by using databases to dynamically add information on a webpage. Chris explains this is how most of the information we see running from companies is populated and references Amazon as a *prime* example. The 08-Stu\_Render\_From\_Mongo activity was pretty straightforward too, although I did need some help with the bootstrap code still for the final list. I had to go to the answer to once again make the column width and format the list.





The next activity is going to use Craigslist to scrape data. I would also like to point out that finally there is a Ninja Turtles Splinter joke as Chris is explaining the “splinter import Browser” and “bs4 import Beautifulsoup” imports! In all seriousness I found this activity incredibly challenging. I had to rely on the answers to really dig into what was happening for the references in the HTML. I had a difficult time also referencing the index of the image. In the end I was able to complete the activity and follow my way along. I made some slight modifications in order to make the page my own. Below is a picture of the final result. I went ahead and did the bonus with the picture since the answer was provided, which actually helped me remember I need to index different tags.

