Megan Williams

IS452 – Final Narrative

My intent in developing a large programming project was to create something that could be used in my current job to help better manage the selection of an award and remove some of the more tedious aspects of collecting survey results. A brief overview of the process is to:

1. Start with an already started nomination list
2. Add names from committee members to it to create a longer list
3. Use that longer list to have committee members choose their top five nominees to get a smaller list
4. Use the smaller list to have committee members rate the nominees
5. Return the highest ranked nominee

Start with a list

When creating my nomination list, I decided that I'd use real, but unrelated authors for my testing so that it was familiar to me enough to relate to the work, but didn't cross over into the real world so much that I was working with sensitive information. But in creating that initial document, I didn't know if the authors should be there as a string or a list, so I created both to be able to test out a couple methods to see what would be not easiest, but would lend itself to being able to do the most things to it without destroying the data or recreating it. I wanted the "data" to be carried from document to document. I quickly found out that I needed a string versus a list. That document is "Start_authors_string.txt".

Add names to that list and create a new list

In the first python file (1-Adding nominees.py), the script starts out by reading in the starting authors list as a string with .read. I chose .read for the reasons in the first paragraph of needing a string to start. It worked best to keep it as a string. Then I created an out file for the first list and the new nominees to be written to. I tried many times to use 'w', but I kept getting the starting list and the one nominee added. It would overwrite everything that I did each time I ran it. I did a little research on reading and writing files and found 'a' for append. I was then able to write the starting list to a new document and add names to it – appended_authors_string.txt.

But here is a problem that I couldn't solve. Because I needed to copy the first list to the new list and then add names to it, it wouldn't "append" the way I wanted it to. When I would run it again it would just add the starting list in again and then add the next nominee. To try and solve this problem so that I could move on to the next step, I added a for loop that asked the number of nominees that needed to be added and then had the user input the nominees. One thing that I did nest inside of the for loop as a if, elif and else statement, that would let the user know if their nominee had been entered or not with print statements. I thought this was important to not only prevent duplicate entries, but also assure the user that something was happening when they were entering names. One thing that I discovered is that the nominees had to entered exactly the same way for it to work. I tried to demonstrate that in the input statement, but even I kept messing up and using the wrong quote or reversing first and last names when testing.

Narrow down the list

So this next python file 2– survey_choose_5.py – takes the 2-appended_authors_string.txt and uses that to ask the user to pick 5 to move to the next round so that you can have a smaller list to rate and rank the authors. The first thing that needs to happen is that the script reads in the string of names from the appended list with .read. Then I printed those out so that the user can see all of the nominees and to be able to "select" their top five. Due to getting tired of typing in five names over and over again, I changed it to 3 nominees for the testing period and then changed it back to 5 when I got it to do what I wanted.

I created an out file that would append strings to it just like I did in the first python file. I then created a for loop that had a range of five for each of the user's picks they get to make. The user then will be prompted 5 times to enter a name. Once again, I needed to specify the format that they needed to be written in. The format doesn't affect this stage, but will be good to keep for the next stage when they rank the authors. I didn't address duplicates in this script because I wanted to see what I could do in the next python file that didn't just repeat an if statement. These nominees get written to the next file starting with a new line so that each new entry appears on the next line and not right next to each other. This file is called 3-narrowed_list.txt.

Ranking the authors

This next and final script is 3-ranking_authors.py. It first reads in the 3-narrowed_list.txt created in the last part with .read. Here is where I turn that string into a list finally because I realized I'm going to need to work with a dictionary. If figured making my way to a dictionary was my best option because I just couldn't even begin to think of how to create a pivot table in Python. But, a dictionary has keys and values so despite wanting to go there, I went with dictionary and kept getting confused on what was a key and what was a value.

So with the list, I split the string on the new line character. That split then created an empty string in the first position. Instead of figuring out how to just remove the first position, I decided to remove all elements that were an empty list just in case another shows up in there some how. So now I have a list but there are duplicates in there. I did a bunch of googling and looking in the book on how to remove duplicates from a list. I found some help that basically created function that made a list and a list and a set. I tested it out and it worked. I had to look up more about sets to see how they work and the unordered, unique elements was exactly what I needed, but I needed a list not a set. So it is like this function made a list from a set. And now I have a list that I can turn into a dictionary.

So I did replicated this process twice, but I'll only cover it once for brevity. I started with an empty dictionary, first to create a dictionary that would collect the scores of rating the author's grammar skills. I created an input statement that would be needed in a nested loop asking for the number of committee members. To be able to iterate through all the authors who made it to this round and ask the user the rating of the author's grammar for the number of committee members there were I knew that I needed a nested loop. My top loop would iterate over the list of authors. My nested loop is a range loop that asks for the scores. Those scores are added to a list that serves as the value for the key that the author now has become in the dictionary. My biggest challenge in getting this to work was getting all the inputted scores to make the list in its proper form. I realized that I could get the right scores to connect with the right author in the dictionary by moving the location of the empty list that the scores would go into. After that I manipulated some code that would find the key in the dictionary with the highest value. I was able to use this to find who had the highest grammar score. I replicated this exactly for the rating of an author's prose to create another dictionary, but I still used the original input for the number of committee members.

Next to find the total score that the authors were given, I needed to add together the two dictionaries on their keys and add their values. Within the Python documentation I found a simple example of exactly what I needed. I imported a counter, applied it to each of the dictionaries, added those objects and printed the new dictionary. I just needed to tweak that max score code one last time to get the author with the highest cumulative score.

I didn't achieve all that I wanted with this project. I need a lot more skills to make this web-based and I probably would have needed a database as well. However, I did get it to do "the thing". I was able to use strings, lists, dictionaries, for loops, if statements and a bunch of other little things. Even if I didn't know how to do it exactly, I had the skills to speak the language for looking for a potential path and then being able to apply it to my specific project. Looking up "combine python dictionaries with same keys but add values" was much more helpful than "python add names with scores". It definitely made me realize that I captured more of the language than I thought.