Nick Mitchell

3-15-2020

CPS3320/TECH4981

Professor Domanski

Project 1 Write-up

There are some situations when polls are needed for example The ACM club will need a new E-board in May. Most votes can be tallied by hand. If this is the case, this would take a while for a person to tally up the votes and with that, they may run into counting mistakes. A Python program handled these issues and made the counting more precise.

The data was the candidates or options that are to be voted for. To make things simpler, this data came from simple a text file. The only bias of this data could be that the program processing the data won't accept write-ins or options that are not in the text file.

Analysis steps for the program started with the text file itself and how it's formatted. The file "options.txt" was formatted to have a different candidate or option on each line. This data was incorporated into a list. The idea of incorporating the name of the option with id e.g. 1 bananas, 2 watermelons, 3 strawberries allowed the user to interact easier and faster with the program and prevented typos or spelling mistakes. Associating ids with options was made possible by merging a list of numbers with the options. In addition, a vote tally was appended to the end of

each row of the now 2d list (default value 0). Then it called a function called “printCandidates()”, which allowed the program to display all of the options for the user. Once the user chose the id for the option, the vote tally increments by 1 each vote. The program interacted with the user to ask if another vote will be done this was done through the variable “keepRunning”. If the user specified “y”, the voting process will repeat. If “n”, the voting process will be terminated, and the program would sort and display the results in descending order with the votes. The output

Prediction:

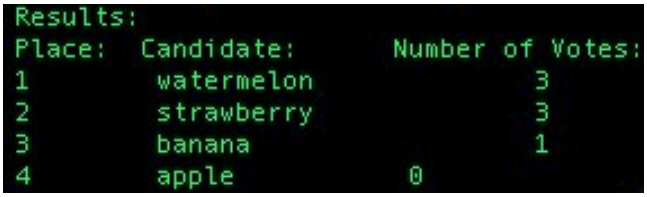
Results:

watermelons 56 votes

bananas 23 votes

strawberries 15 votes

Actual Output:



There were a few roadblocks with this project. One was finding a way of sorting the results. With some research, the sort was made possible by itemgetter in the python operator library. This however printed the lowest result first. The function reverse allowed the results to print the results in descending order.

Another roadblock was formatting the outputs. Although the output is readable, it could be a little more in a straight line. With a little more research, this issue should be fixed.

Comparing the hypothesis to the results, the program “vote.py” works as it is supposed to. it has a readable output and has an effective method of collecting the votes, and displaying the results. vote.py could be expanded by adding the tkinter library to make a user-friendly GUI.

Overall, this project was a bit challenging to make but enjoyable to work on. From the project originally proposed there were no changes made. One thing learned was implementing exception handling to the program. In the instance when the program would ask for the candidate id, if the user put in a letter instead of a number, the whole program would crash. The exception handler put into the program catches the input mismatch error and tells the user that they have not entered a number. The program would then go back to asking the user again for that number. If this project could be done over, more functions would be implemented to minimize lines and repeated code. In addition to this, working with tkinter as mentioned before would be another part to implement.