Roll Call

Prompt and Format automation

1. Project Description:
Roll Call is a tailor made roll call and formatting program that I created for my wife to use in her professional work. The program iterates through a CSV file and prompts the user to assign a status of attending, apologising, chairing, or taking minutes, to names in the CSV file and then prints them in the desired format. This program has 2 different modes, 'Rapid Fire', and 'Search'. 'Rapid Fire' iterates through the CSV file names one by one and prompts the user to assign a status to each one. The 'Search' mode prompts the user to type in names for each status type, and if the names match a name in the CSV file, the name will be appended to the selected status list. Both programs have the ability to restart, exit or print at any time.

The purpose of this project was to gain hands on experience using python in a self-guided environment within the context of meeting user requirements.

2. Problem Statement:
As this was my first computer science project, the goal was to solve my lack of experience coding in a self-guided environment. At the time my wife was confronted with a large amount of work, this provided an opportunity to gain hands on experience programming in a self-guided environment to meet user specifications.

3. Solution Implementation:
The consultation was casual and came naturally as I had newly acquired Python skills and my wifes workload was building up. After hearing her user problems and needs, I asked further questions to gather information that would guide my approach, and I applied the solution as follows:

For Both Modes:

1. Imported Re(REGEX), CSV and Time libraries
2. Then I created an empty string assigned to a variable called 'intro'
3. Started a while loop about 'intro' to continuously prompt the user with the input() function until a valid mode, 'Rapid Fire' or 'Search', is selected, and assigned the response to the variable 'intro'
4. Created an 'if' statement about 'intro', so when the user enters 'Rapid Fire' or 'Search', the appropriate mode is selected
5. Included the lower() function to remove capitalisation, and the strip() function to remove leading and trailing white spaces, to account for every variation of 'Rapid Fire' or 'Search' the user might enter
6. Created 4 lists that represent each status of attendance (Attended, Chair, Minutes, Apologies) that names in the CVS can be assigned to
7. Using the print statement, I instructed the user on how to assign names to a list and print the list upon completion
8. With open(), I opened the csv file as a variable called 'file'
9. Using csv.reader(), I read the file and assigned it to a variable called 'myfile'

For Rapid Fire Mode:

10. Then I created a for loop to iterate through every name in the file
11. Within that loop, I created a series of 'if' statements with the append function, to respond to the users inputs allowing them to assign names to a status, print the names in their given list, and to restart if an incorrect input is entered
12. The code above is encapsulated in an object called 'rapid()', which is called to restart the program upon false input

For Search Mode:

13. I Used input() to prompt the user who chaired and minutest the meeting

14. Created a nested for loop to iterate through the names in the file
15. Used the search() function from regex to match what the user inputs with the names in the file and assigned it to a variable called 'match' (the chair) and 'match2' (the minutest)
16. Embedded in that loop, I created an 'if' statement about the matches and used append() to append the matched name to the corresponding lists
17. Next I use a while loop to enable the user unlimited searches for the Attendees and Apologies
18. I used the same method as steps (14), (15) and (16) to search the file and append to the lists but assigned to different variables

For Both Modes Again:

19. Printing the lists in the users requested format presented challenging due to the data type and it being embedded in 'if', 'with' and 'while' loops. The solution was to create a for loop and iterate through the list and using the join() function to add a comma to each object in the list.
20. Called the time.sleep() function to give the user ample time with the printed results.
21. Finally I used auto-py-to-exe to turn the python script into an executable file anyone can use

4. Results and Impact:

In alignment with the purpose, I completed my first hands-on experience creating a Python program in a self-guided environment, and in addition met a user's requirements. Through this, I improved my problem-solving skills, deepened my understanding of Python syntax, and became proficient at developing functional programs without solely relying on pre-existing solutions.

The roll call program stands out for its efficiency, intuitiveness, and user-friendly design. Its implementation resulted in a significant time saving of 30 minutes per use for my wife, who utilized it approximately once a week. Overall, the project proved to be a success, effectively meeting the intended objectives.

5. Technical Skills Demonstrated:

Python, for loops, while loops, if and elif statements, formatted strings, objects, lists, Regex, CSV, Time, consulting, meeting user specifications

6. GitHub Repository:

https://github.com/Davidooj/Projects/blob/main/Roll_Call.py

7. Key Takeaways:

The Rubik's effect: Analogous to the Rubik's cube, coding often appears daunting when viewed in its whole, but becomes manageable when broken down into smaller, comprehensible steps, this parallels my journey from learning Python to completing this project.

Much like unravelling the steps of a Rubik's cube, coding reveals its complexities, unveiling a manageable puzzle rather than an insurmountable challenge. This underscores the significance of step-by-step learning and highlights the approachability of coding when approached systematically.

This project also reinforced the vital role of proper consultation and establishing a technical framework, to deduce the steps required to complete the project while meeting customer specifications.