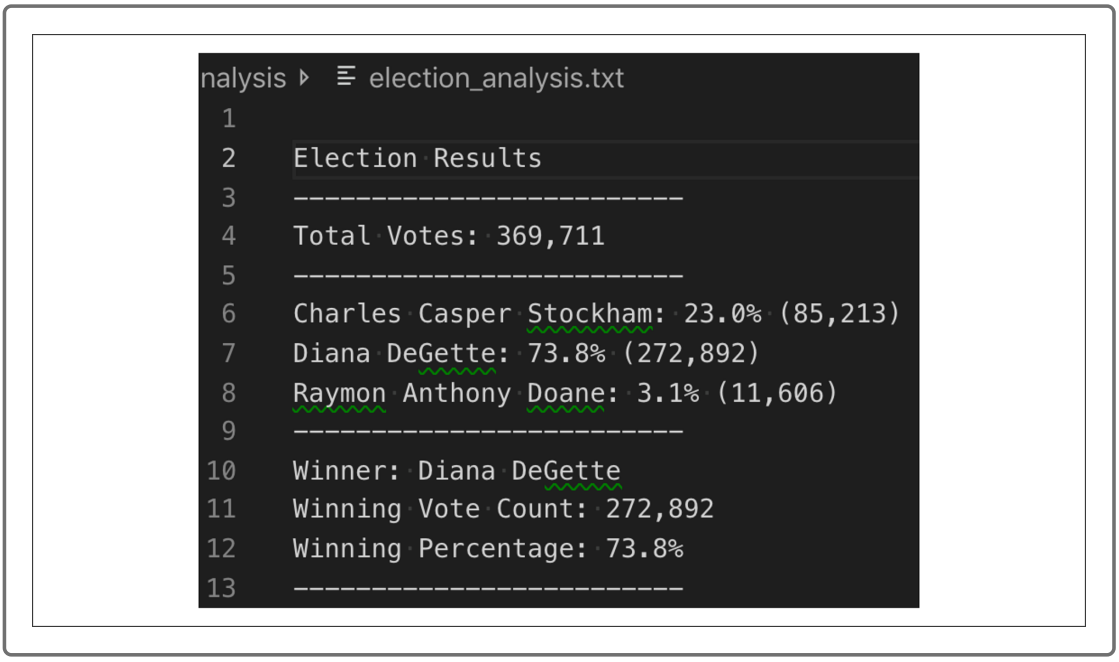
3.6.1 Write the Election Results to a Text File

Seth and Tom have looked over your election audit and are pleased with what you have accomplished. Now, Seth needs you to save your election audit results to a text file so he can send it to the election commission. Don't worry if you don't remember how to write data to file—Tom is going to walk you through the process to refresh your memory.

We need to save the election results to a text file and then print the file to the screen to make sure that the results are in the correct format.

After the election analysis is written to the text file, the file should look like this:



Now code should look like this:

```
# Add our dependencies.
import csv
import os
# Assign a variable to load a file from a path.
file_to_load = os.path.join("Resources", "election_results.csv")
# Assign a variable to save the file to a path.
file_to_save = os.path.join("analysis", "election_analysis.txt")
# Initialize a total vote counter.
total_votes = 0
# Candidate options and candidate votes
candidate_options = []
candidate_votes = {}
# Track the winning candidate, vote count, and percentage.
winning_candidate = ""
winning_count = 0
winning_percentage = 0
# Open the election results and read the file.
with open(file_to_load) as election_data:
    file_reader = csv.reader(election_data)
    # Read the header row.
    headers = next(file_reader)
    # Print each row in the CSV file.
    for row in file_reader:
        # Add to the total vote count.
```

```
        total_votes += 1
        # Get the candidate name from each row.
        candidate_name = row[2]
        # If the candidate does not match any existing candidate add it the
        # the candidate list.
        if candidate_name not in candidate_options:
            # Add the candidate name to the candidate list.
            candidate_options.append(candidate_name)
            # And begin tracking that candidate's voter count.
            candidate_votes[candidate_name] = 0
        # Add a vote to that candidate's count
        candidate_votes[candidate_name] += 1

for candidate_name in candidate_votes:
    # Retrieve vote count and percentage.
    votes = candidate_votes[candidate_name]
    vote_percentage = float(votes) / float(total_votes) * 100
    # Print each candidate, their voter count, and percentage to the
    # terminal.
    print(f"{candidate_name}: {vote_percentage:.1f}% ({votes:,})\n")

    # Determine winning vote count, winning percentage, and candidate.
    if (votes > winning_count) and (vote_percentage > winning_percentage):
        winning_count = votes
        winning_candidate = candidate_name
        winning_percentage = vote_percentage
# Print the winning candidates' results to the terminal.
winning_candidate_summary = (
    f"-------------------------\n"
    f"Winner: {winning_candidate}\n"
    f"Winning Vote Count: {winning_count:,}\n"
    f"Winning Percentage: {winning_percentage:.1f}%\n"
    f"-------------------------\n")

print(winning_candidate_summary)
```

Now we are going to modify our code so we can write the `election_results` to a text file.

First, comment out `print(f"{candidate_name}: {vote_percentage:.1f}% ({votes:,})\n")` and `print(winning_candidate_summary)` by adding a `#` in front of both lines.

Next, insert `with open(file_to_save, "w") as txt_file:` after `candidate_votes[candidate_name] += 1`. Make sure that the filename `file_to_save` is in the `"w"` mode to write data to the file.

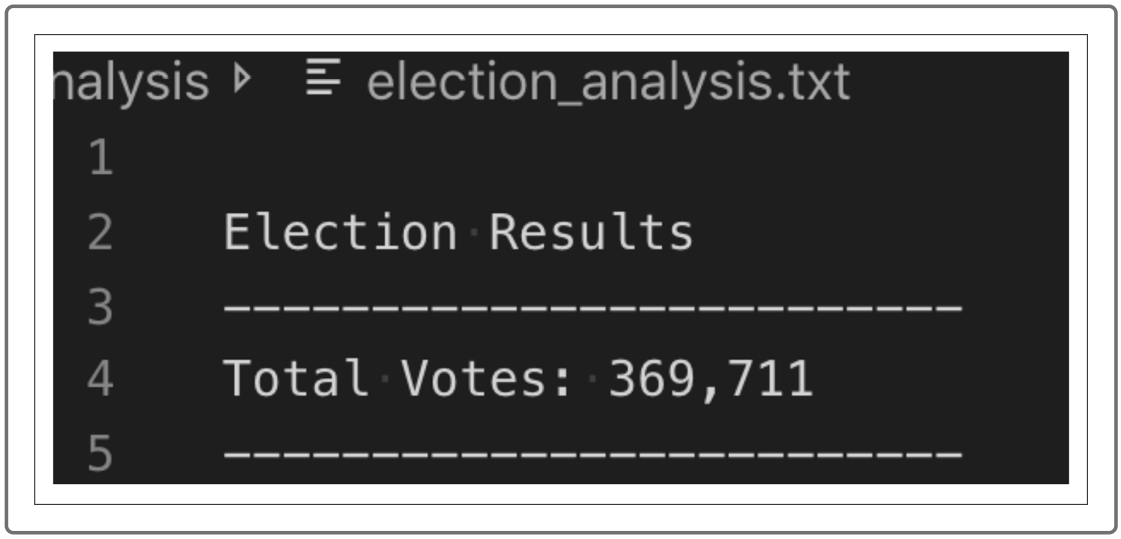
This section of the code should look like this:

```
# Add a vote to that candidate's count
        candidate_votes[candidate_name] += 1

# Save the results to our text file.
with open(file_to_save, "w") as txt_file:
```

Next, indent all the code below `with open(file_to_save, "w") as txt_file:` by four spaces. To do this, select all the code and comments below the `with open` section and press the Tab key once.

Next, below the `with open` statement, write the text shown on lines 2–5 in the following image as an f-string literal message, "election_results".



Add this code below the `with open` statement:

```
# Print the final vote count to the terminal.
    election_results = (
```

```
        f"\nElection Results\n"
        f"-------------------------\n"
        f"Total Votes: {total_votes:,}\n"
        f"-------------------------\n")
    print(election_results, end="")
    # Save the final vote count to the text file.
    txt_file.write(election_results)
```

Let's examine the code we are writing to the text file `file_to_save` and printing to the terminal.

- The variable, `election_results`, has four strings written to it.
- The first string, `Election Results`, has the newline character, `\n`, before and after it. When this line is saved to the text file or printed, it will be on the second line; then a newline is created.
- On the third line, 25 dashes will be printed, and then a newline line is created.
- On the fourth line, the string `Total Votes: {total_votes:,}` will be printed with the votes formatted with a thousands separator, and then a newline is created.
- On the fifth line, 25 dashes will be printed, and then a newline is created.
- Then, we print the `election_results`, with the parameter `end=""` equal to an empty string.
- Finally, we write `election_results` to the text file.

NOTE

> By default, the "end" parameter will print a newline, `\n`. You can add something between the double quotes, which will be printed after the 25 dashes are printed to the terminal. The "end" parameter is added to ensure that nothing will be printed on the last line when the `election_results`. Anything code that is printed after `print(election_results, end="")` will be printed on a newline.

When we execute `PyPoll.py`, the output to the terminal will look like this:

```
Election Results
-------------------------
Total Votes: 369,711
-------------------------
```

The `election_results.txt` file will look like this: