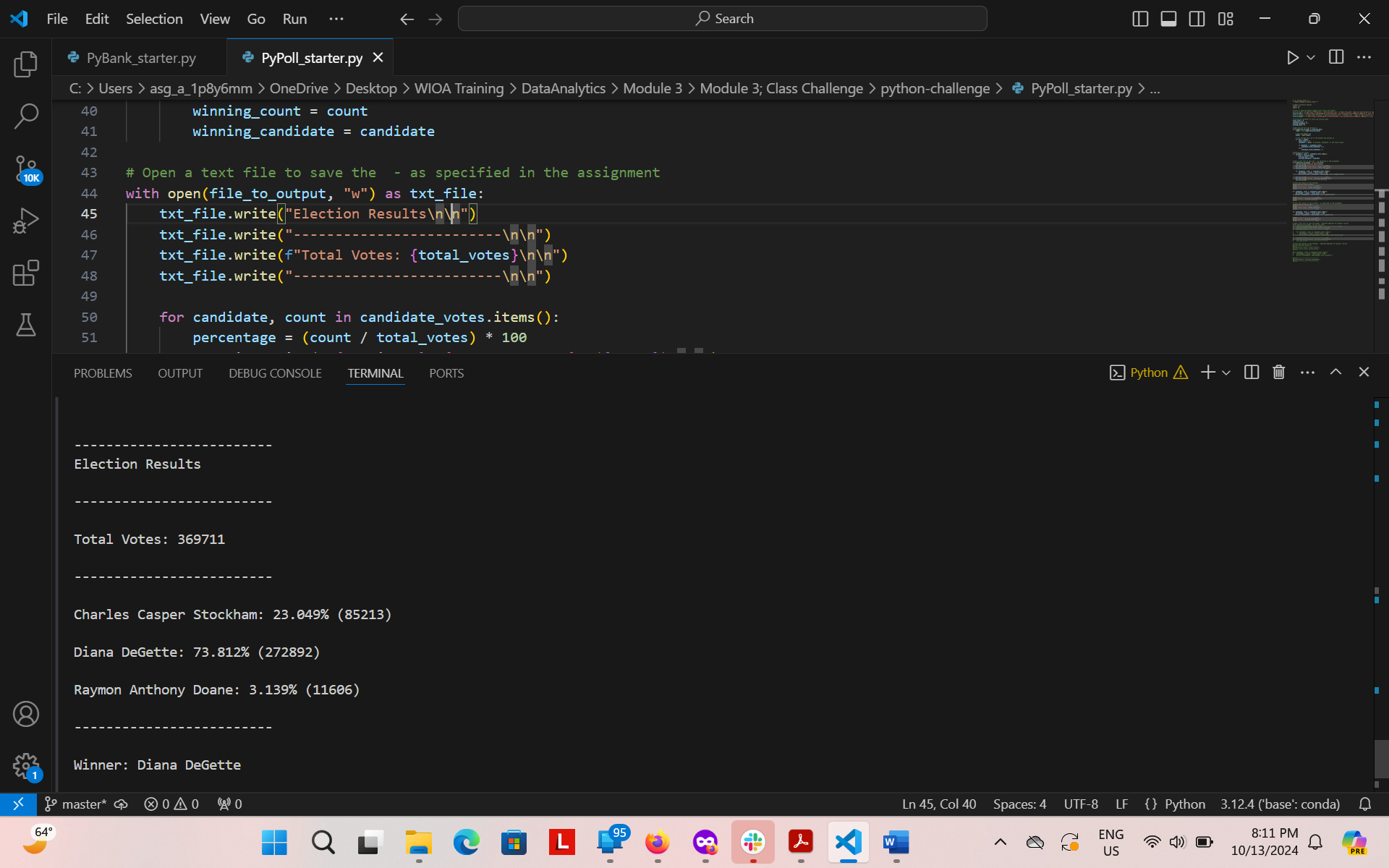
**PyPoll Starter**

**Explanation:**

1. **Import Modules:** Import csv for reading the CSV file and os for file path handling.
2. **File Paths:** Set the input and output file paths.
3. **Initialize Variables:**
   * total\_votes: Stores the total number of votes.
   * candidate\_votes: A dictionary to store candidate names as keys and their vote counts as values.
   * winning\_candidate: Stores the name of the winning candidate.
   * winning\_count: Stores the number of votes the winner received.
4. **Open and Read CSV:**
   * Open the CSV file using with open(file\_to\_load) as election\_data:.
   * Create a csv.reader object to read the data row by row.
   * Skip the header row using header = next(reader).
5. **Process Data:**
   * Loop through each row in the CSV using for row in reader:.
   * Increment total\_votes for each row.
   * Extract the candidate's name from the third column (row[2]).
   * Update the candidate\_votes dictionary:
     + If the candidate is already in the dictionary, increment their vote count.
     + If the candidate is new, add them to the dictionary with a vote count of 1.
6. **Determine the Winner:**
   * Loop through the candidate\_votes dictionary.
   * For each candidate, compare their vote count to winning\_count.
   * If a candidate has more votes than the current winning\_count, update winning\_candidate and winning\_count.
7. **Write Output to File:**
   * Open the output text file using with open(file\_to\_output, "w") as txt\_file:.
   * Write the election results to the file, including:
     + Total votes
     + Each candidate's name, vote count, and percentage
     + The winner
8. **Print Output to Terminal:**
   * Print the election results to the terminal using print() statements.
   * Output according to assignment specifications:



* Output modified for greater output clarity