# Add our dependencies that we will be using to achieve our goal of tally count.

import csv

import os

# The os.path module implements some useful functions on pathnames.

# Add a variable to load a file from a path.

file\_to\_load = os.path.join("resources", "election\_results.csv")

# Add 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 = {}

# 1: Create a county list and county votes dictionary.

county\_list = []

county\_votes = {}

# Track the winning candidate, vote count and percentage

winning\_candidate = ""

winning\_count = 0

winning\_percentage = 0

# 2: Track the largest county and county voter turnout.

largest\_county = ""

largest\_county\_count = 0

largest\_county\_percentage = 0

# Read the csv and convert it into a list of dictionaries

with open(file\_to\_load) as election\_data:

    # Read the file object with the reader function.

    reader = csv.reader(election\_data)

    # Read the header

    header = next(reader)

    # For each row in the CSV file.

    for row in reader:

        # Add to the total vote count

        total\_votes += 1

        # Get the candidate name from each row.

        candidate\_name = row[2]

        # 3: Extract the county name from each row.

        county\_name = row[1]

        # If the candidate does not match any existing candidate add it to

        # 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

        # 4a: Write a decision statement that checks that the

        # county does not match any existing county in the county list.

        if county\_name not in county\_list:

            # 4b: Add the existing county to the list of counties.

            county\_list.append(county\_name)

            # 4c: Begin tracking the county's vote count.

            county\_votes[county\_name] = 0

        # 5: Add a vote to that county's vote count.

        county\_votes[county\_name] += 1

# Save the results to our text file.

with open(file\_to\_save, "w") as txt\_file:

    # Print the final vote count (to terminal)

    election\_results = (

        f"\nElection Results\n"

        f"-------------------------\n"

        f"Total Votes: {total\_votes:,}\n"

        f"-------------------------\n\n"

        f"County Votes:\n"

    )

    print(election\_results, end="")

    # Save the final vote count to the text file.

    txt\_file.write(election\_results)

    # 6a: Write a repetition statement to get the county from the county dictionary.

    for county in county\_list:

        # 6b: Initialize a variable to hold the county’s votes as they are retrieved from the county votes dictionary.

        county\_vote = county\_votes.get(county)

        # 6c: Calculate the percent of total votes for the county.

        county\_vote\_percentage = float(county\_vote) / float(total\_votes) \* 100

        # 6d: Print the county results to the terminal.

        county\_results = f"{county}: {county\_vote\_percentage:.1f}% ({county\_vote:,})\n"

        # Print the counties to test.

        print(county\_results)

        # 6e: Save the county votes to a text file.

        txt\_file.write(county\_results)

        # 6f: Write a decision statement to determine the winning county and get its vote count.

        if (county\_vote > largest\_county\_count) and (

            county\_vote\_percentage > largest\_county\_percentage

        ):

            # True

            largest\_county\_count = county\_vote

            largest\_county\_percentage = county\_vote\_percentage

            largest\_county = county

    # 7: Print the county with the largest turnout to the terminal.

    winning\_county\_print = (

        f"-------------------------\n"

        f"Largest County Turnout: {largest\_county}\n"

        f"-------------------------\n"

    )

    print(winning\_county\_print)

    # 8: Save the county with the largest turnout to a text file.

    txt\_file.write(winning\_county\_print)

    # Save the final candidate vote count to the text file.

    for candidate\_name in candidate\_votes:

        # Retrieve vote count and percentage

        votes = candidate\_votes.get(candidate\_name)

        vote\_percentage = float(votes) / float(total\_votes) \* 100

        candidate\_results = f"{candidate\_name}: {vote\_percentage:.1f}% ({votes:,})\n"

        # Print each candidate's voter count and percentage to the

        # terminal.

        print(candidate\_results)

        #  Save the candidate results to our text file.

        txt\_file.write(candidate\_results)

        # 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 candidate (to 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)

    # Save the winning candidate's name to the text file

    txt\_file.write(winning\_candidate\_summary)