Code Explanation for the Electoral Votes Simulator

Imports:

- csv: Used for exporting results to a CSV file.

- random: Used to simulate random voting outcomes.

- tkinter as tk: Standard Python library for creating graphical user interfaces (GUIs).

- messagebox, simpledialog, filedialog: Modules from tkinter used for showing dialogs and messages.

- logging: Used to log various events and errors in the application for debugging and tracking purposes.

Logging Configuration:

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

- Configures the logging to show informational messages and errors with timestamps, log levels, and messages. This is helpful for debugging and understanding the flow of the application.

Custom Dialogs:

CustomDialog Class:

- Inherits from simpledialog._QueryString.

- Used to create custom input dialogs with a larger entry box (width = 50) to make it easier for users to enter names or other information.

```
class CustomDialog(simpledialog._QueryString):
    ...
```

VoteDialog Class:

- Inherits from simpledialog.Dialog.

- Used to create a custom dialog for casting votes. The window is larger (400x350 pixels) to provide

a better user experience.

class VoteDialog(simpledialog.Dialog):

    ...

State Electoral Votes:

states = {

  'Alabama': 9,

  ...

  'Wyoming': 3,

}

- This dictionary holds all U.S. states and their respective electoral votes based on the 2020 Census. It's used to simulate the electoral college voting.

Core Functions:

cast_vote Function:

- Handles the logic of casting a vote. It increments the vote count for the respective candidate and shows a message confirming the vote. It also logs the current vote count.

def cast_vote(vote, candidate_1_votes, candidate_2_votes, total_votes, candidate_1_name, candidate_2_name):

    ...

export_results_to_csv Function:

- Exports the voting results, including vote counts, percentages, and electoral votes, to a CSV file. This allows for easy sharing and analysis of the results.

def export_results_to_csv(candidate_1_votes, candidate_2_votes, total_votes, candidate_1_name, candidate_2_name, candidate_1_electoral_votes, candidate_2_electoral_votes, filename):

...

simulate_electoral_college Function:

- Simulates the electoral college by randomly determining which candidate wins each state. The winner of each state receives the state's electoral votes. The function logs and returns the final electoral vote counts.

def simulate_electoral_college(candidate_1_name, candidate_2_name):

...

run_simulator Function:

- This is the main loop that runs the voting simulation. It gathers candidate names and allows users to cast votes. After voting, it simulates the electoral college and exports the results if there's no tie. The function also handles user interactions, such as restarting the voting process in case of ties.

def run_simulator():

...

display_results Function:

- Displays the final voting results to the user in a message box. This provides a clear summary of the voting outcome.

def display_results(candidate_1_votes, candidate_2_votes, total_votes, candidate_1_name, candidate_2_name):

...

Main Function:

def main():

...

- Sets up the main window for the GUI, adjusts its size, and centers it on the screen. It also provides

buttons for starting the simulation or exiting the application. This function initializes the user interface.

Entry Point:

```
if __name__ == '__main__':
    main()
```

- This ensures that the main() function is called when the script is run directly. It starts the GUI application.

This script allows users to simulate a U.S. presidential election by casting votes and seeing how the electoral college would likely turn out. It logs the process and exports the results to a CSV file for further analysis.