Gerrymandering

This assignment focuses on reading input files. Turn in a file named `gerrymandering.py`.

Program Description:

In representative electoral systems, **gerrymandering** is the political manipulation of electoral district boundaries to advantage a party, group, or socioeconomic class within the constituency.

Your program should give an introduction and then ask the user for a state name. It should then read the districts file looking for that state in a case-insensitive manner (ie it should find the state no matter how the user typed the name, case-sensitive).

- If the status is in the file, the program should print:
- The Democrats' total wasted votes.
- The total of wasted Republican votes.
- The total number of voters in the state.
- Who has gained an advantage from gerrymandering in the state, if there is gerrymandering

```
This program allows you to search through
data about congresssional voting districts
and determine whether a particular state is
gerrymandered.

Which state do you want to look up? Arizona
Total Wasted Democratic votes: 327852
Total Wasted Republican votes: 369697
4738332 elgible voters
```

Input Data and files:

Your program must read data from two files.

1) `districts.txt`: *Contains data for congressional districts.*

.

Each line `districts.txt` contains the name of a state followed by information in groups of tree.

- The first element is the name of the district.
- The Democrats´ votes in this district .
- The Republican' s votes in this district.

```
Arkansas,1,63555,124139,2,103477,123073,3,0,151630,4,87742,110789
Alaska,AL,114596,142566
Rhode Island,1,87060,58877,2,105716,63844
```

You can split a string called `line` on commas with the following call to `split`: `line.split(",")`.

Note that the district names will sometimes be numbers and sometimes strings. The vote counts will always be integers.

Once the user types a state name, search each line of `districts.txt` to see if it contains data for that state. If the state name is found, output its data line to the console. **Your code should not assume that the file is sorted alphabetically**.

If the state name is not found, output a "not found" message and don't show any data. No `DrawingPanel` should appear.

```
This program allows you to search through
data about congresssional voting districts
and determine whether a particular state is
gerrymandered.

Which state do you want to look up? mErLin
"mErLin" not found.
```

The data displayed above has a different number of districts for each state. Your program should work properly with any number of districts of data greater than 0. Since there is a limit to the size of the `DrawingPanel`, you may not be able to see all decades worth of data at the default height, but your code should process as many districts of data as it finds in the line.

---

1. `eligible_voters.txt`: *the number of eligible voters in each state*

If the state name is found in `districts.txt`, you should also read `eligible_voters.txt` to find its total number of eligible voters. Every state name in `districts.txt` is also in `eligible_voters.txt`, so you do not need to worry about a state having district data but no eligible voter data.

Each line of `eligible_voters.txt` contains a state name, followed by the eligible voter count for that state. For example:

```
Alabama,3606103
Alaska,519501
Arizona,4738332
Arkansas,2148441
California,25278803
```

Though the two input files contain different data, the task of searching for a state name in `districts.txt` is very similar to the task of searching for a state name in `eligible_voters.txt`. Your code should take advantage of this fact and should avoid redundancy. You should write your code in such a way that you stop searching a file once you find a line that has the name you're searching for.

You may not assume the input files will have 50 lines. Your program must work on any length input file.

Determining Gerrymandering:

You can determine gerrymandering by counting up and comparing the wasted votes cast for each party. We will define a wasted vote as any vote not needed to win the election. That means all votes for the party that loses the district seat are wasted as well as all votes for the winning party other than the half + 1 they need to win the majority.

For example, imagine that there was a state with the following districts:

	Dem	GOP	Wasted Dem	Wasted GOP
District 1:	2	7	2	2
District 2:	4	5	4	0
District 3:	10	7	1	7
Total:	16	19	7	9

Having calculated this data, we can sum up the wasted votes for each district. We find that the democrats wasted 7 votes and the republicans wasted 9. It is impossible to make voting districts exactly fair and so we shouldn't expect the wasted vote counts to be equal. However, researchers have discovered that it is almost impossible for the disadvantaged party to recover if the difference in wasted votes is greater or equal to 7 percent. Therefore, the researchers, as well as us for the purposes of this assignment, will consider a state gerrymandered when there is a 7% or greater difference in the wasted votes.

Implementation Guidelines:

We suggest you begin with the text output and file processing, then any "fixed" graphical output, and then the bars.

Your program should work correctly regardless of the capitalization the user uses to type the state name. If the user types `"AlAbAmA"` or `"alabama"`, you should find it even though the input files have it as `"Alabama"`.

Stylistic Guidelines:

Your functions should be well-structured and avoid redundancy, and your `main` function should be a concise summary of the overall program. Avoid "chaining," which is when many functions call each other without ever returning to `main`.

Follow past stylistic guidelines about indentation, line lengths, identifier names, and localizing variables, and commenting at the beginning of your program, at the start of each function, and on complex sections of code. You may not have global variables or nest functions in one another. You may not use the list `count` or `sum` functions.