3.4.4 Read the Election Results

Tom tells you that you have been given the green light to read the election results. You know how to open the file, so you can use that code to start. Then, Tom will guide you in reading the data in the `election_results.csv` file.

Now it's time to read the `election_results.csv` file. As a reminder, the code in our `PyPoll.py` file should look similar to this code:

```
# 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")

# Open the election results and read the file.
with open(file_to_load) as election_data:

    # To do: read and analyze the data here.
```

Next, we'll use the `reader` function to read `election_data.csv`.

IMPORTANT

> Don't forget to add a colon after the `with` statement and indent on the next line. Forgetting to add a colon will result in a `SyntaxError`, and not

> indenting on the next line will result in an `IndentationError`. In both cases, your code will not run.

REWIND

Remember that we found the `reader` function within the `csv` module that will read the CSV file.

To `PyPoll.py`, add the following code where it says `# To do: read and analyze the data here`. This will allow us to read the CSV file using the `csv` module with the `reader` function.

```
    # Read the file object with the reader function.
    file_reader = csv.reader(election_data)
```

The variable, `file_reader`, is referencing the file object, which is stored in memory. To "pull" the data out of the file object, we can iterate through the `file_reader` variable and print each row, including the headers, or column names.

Add the following code to `PyPoll.py` and run the file in the VS Code terminal.

```
    # Print each row in the CSV file.
    for row in file_reader:
        print(row)
```

Wow—the output was printed quickly! If you blinked, then you might have missed it. So, run it again. If your output for the last 10 lines looks like this, then your code is correct.

```
['4762851', 'Arapahoe', 'Raymon Anthony Doane']
['4678093', 'Arapahoe', 'Raymon Anthony Doane']
['4196905', 'Arapahoe', 'Raymon Anthony Doane']
['4299985', 'Arapahoe', 'Raymon Anthony Doane']
['4620283', 'Arapahoe', 'Raymon Anthony Doane']
['4714953', 'Arapahoe', 'Raymon Anthony Doane']
['4497542', 'Arapahoe', 'Raymon Anthony Doane']
['4085849', 'Arapahoe', 'Raymon Anthony Doane']
['4592018', 'Arapahoe', 'Raymon Anthony Doane']
['4660518', 'Arapahoe', 'Raymon Anthony Doane']
```

While this output was being generated, two things happened:

1. We did not see the headers or columns printed because the output was generated very quickly.
2. Each row in the CSV file was printed out as a list.

In Python, we can retrieve just the headers from the CSV file using a specific method, which we will use a bit later. We can use list indexing to get each element, or ballot ID, county, and candidate, in each row or list.

Retake

For our analysis of the election data, we do not need the column headers. Therefore, when we retrieve the data from the CSV file, we will have to skip the first row—the header row.

To skip the header row of the CSV file, use the `next()` method. This method will skip the first row and return the next item in the list.

Inside the parentheses of the `next()` method, add the variable `file_reader` that is referencing the file object assigned to the CSV file: `next(file_reader)`.

Running this code will skip the first instance of what is being read. In our case, it is the first row, or header row, of the CSV file. Then we can start iterating through the data starting with the second row.

Just to make sure we are skipping the header row, let's print out the headers for the CSV file. Modify the code as follows and run the file.

```
    # Read the file object with the reader function.
    file_reader = csv.reader(election_data)

    # Print the header row.
    headers = next(file_reader)
    print(headers)
```

At this point, `PyPoll.py` 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")

# Open the election results and read the file.
with open(file_to_load) as election_data:
    file_reader = csv.reader(election_data)

    # Read and print the header row.
    headers = next(file_reader)
    print(headers)
```

When you run this code in your VS Code terminal, the output should only show the headers from the CSV file.

```
['Ballot ID', 'County', 'Candidate']
```

Now that we have confirmed that we skipped the header row, we can iterate through each row and gather data for our analysis.

NOTE

> For more information, see the **documentation on using the next() method with the csv module (https://docs.python.org/3.7/library/csv.html#reader-objects)** .