**Requirements**

**Correctly Reads in the CSV (10 points)**

* Reads in the CSVs for both PyBank and PyPoll using Python (5 points)
* Successfully stores the header row (5 points)

**Results Printed out to correctly to terminal (40 points)**

* Results correctly display for PyBank:
  + Total Months (5 points)
  + Total (5 points)
  + Average Change (5 points)
  + Greatest Increase (5 points)
  + Greatest Decrease (5 points)
* # Dependencies
* import csv
* import os
* # Files to load and output (update with correct file paths)
* file\_to\_load = os.path.join("Resources", "budget\_data.csv")  # Input file path
* file\_to\_output = os.path.join("analysis", "budget\_analysis.txt")  # Output file path
* #file\_to\_output = os.path.join("Resources", "budget\_analysis.txt")  # Output file path
* # Initialize variables
* changes = []
* previous\_value = None
* total\_months = 0
* total\_net = 0
* Avg\_change = 0
* max\_increase = float('-inf')  # Start with the smallest possible value
* max\_decrease = float('inf')    # Start with the largest possible value
* max\_increase\_month = ""
* max\_decrease\_month = ""
* # Function to print appended values
* def print\_appended\_values(values):
* print("Changes in Profits/Losses:")
* for value in values:
* print(value)
* # Open and read the CSV
* with open(file\_to\_load) as financial\_data:
* reader = csv.reader(financial\_data, delimiter=",")
* # Skip the header row
* header = next(reader)
* print(f"header: {header}")
* # Process each row of data
* for row in reader:
* total\_months += 1
* total\_net += float(row[1])
* current\_value = float(row[1])
* if previous\_value is not None:
* change = current\_value - previous\_value
* changes.append(change)
* # Check for max increase and decrease
* if change > max\_increase:
* max\_increase = change
* max\_increase\_month = row[0]  # Assuming the first column is the month
* if change < max\_decrease:
* max\_decrease = change
* max\_decrease\_month = row[0]  # Assuming the first column is the month
* previous\_value = current\_value  # Update the previous value
* # Calculate the average change
* if changes:
* Avg\_change = sum(changes) / len(changes)
* # Print the appended values
* #print\_appended\_values(changes)
* # Track the total
* print("Financial Analysis")
* print("-----------------------------")
* print(f"Total Months = {total\_months}")
* print("-----------------------------")
* print(f"Total: {total\_net}")
* print(f"Average Change = {Avg\_change:.2f}")
* print("-----------------------------")
* print(f"Greatest Increase in Profits: {max\_increase} in {max\_increase\_month}")
* print(f"Greatest Decrease in Profits: {max\_decrease} in {max\_decrease\_month}")
* # Prepare output text
* output = (
* f"Financial Analysis\n"
* f"----------------------------\n"
* f"Total Months: {total\_months}\n"
* f"----------------------------\n"
* f"Total: {total\_net}\n"
* f"Average Change: {Avg\_change:.2f}\n"
* f"----------------------------\n"
* f"Greatest Increase in Profits: {max\_increase} in {max\_increase\_month}\n"
* f"Greatest Decrease in Profits: {max\_decrease} in {max\_decrease\_month}\n"
* )
* # Write output to a text file
* with open(file\_to\_output, 'w') as txt\_file:
* txt\_file.write(output)
* #print("Financial analysis written to text file.")
* print("Values updated to budget\_analysis file, Thank you !!")
* Results correctly display for PyPoll:
  + Total Votes (5 points)
  + Each candidate’s total votes and percent of votes (5 points)
  + Winner (5 points)
* # -\*- coding: UTF-8 -\*-
* """PyPoll Homework Starter File."""
* # Import necessary modules
* import csv
* import os
* # Files to load and output (update with correct file paths)
* file\_to\_load = os.path.join("Resources", "election\_data.csv")  # Input file path
* file\_to\_output = os.path.join("analysis", "election\_analysis.txt")  # Output file path
* # Initialize variables to track the election data
* total\_votes = 0  # Track the total number of votes cast
* List\_candidates = []
* candidate\_votes = {}  # Dictionary to track candidate names and vote counts
* # Open the CSV file and process it
* with open(file\_to\_load) as election\_data:
* reader = csv.reader(election\_data)
* # Skip the header row
* header = next(reader)
* # Loop through each row of the dataset and process it
* for row in reader:
* # Increment the total vote count for each row
* total\_votes += 1
* candidate\_name = str(row[2])
* # If the candidate is not already in the candidate list, add them
* if candidate\_name not in List\_candidates:
* List\_candidates.append(candidate\_name)
* candidate\_votes[candidate\_name] = 0  # Initialize vote count for the new candidate
* # Add a vote to the candidate's count
* candidate\_votes[candidate\_name] += 1
* # Open a text file to save the output
* with open(file\_to\_output, "w") as txt\_file:
* # Print the total vote count (to terminal)
* print("Election Results")
* print("-----------------------------")
* print(f"Total votes: {total\_votes}")
* print("-----------------------------")
* print(f"Candidates list: {List\_candidates}")
* print("-----------------------------")
* # Write the total vote count to the text file
* txt\_file.write("Election Results\n")
* txt\_file.write("-----------------------------\n")
* txt\_file.write(f"Total votes: {total\_votes}\n")
* txt\_file.write("-----------------------------\n")
* txt\_file.write(f"Candidates list: {List\_candidates}\n")
* txt\_file.write("-----------------------------\n")
* # Initialize variables for the winning candidate
* winning\_candidate = ""
* winning\_count = 0
* # Loop through the candidates to calculate their vote counts and percentages
* for candidate, votes in candidate\_votes.items():
* # Calculate the vote percentage
* vote\_percentage = (votes / total\_votes) \* 100
* # Print and save each candidate's vote count and percentage
* print(f"{candidate}: {votes} votes ({vote\_percentage:.2f}%)")
* txt\_file.write(f"{candidate}: {votes} votes ({vote\_percentage:.2f}%)\n")
* # Update the winning candidate if this one has more votes
* if votes > winning\_count:
* winning\_count = votes
* winning\_candidate = candidate
* # Print and save the winning candidate summary
* print("-----------------------------")
* print(f"Winner: {winning\_candidate}")
* print(f"Winning Vote Count: {winning\_count}")
* print("-----------------------------")
* # Save the winning candidate summary to the text file
* txt\_file.write("-----------------------------\n")
* txt\_file.write(f"Winner: {winning\_candidate}\n")
* txt\_file.write(f"Winning Vote Count: {winning\_count}\n")
* txt\_file.write("-----------------------------\n")

**Code Runs Error Free (10 points)**

* Error Free (5 points)
* Producing consistent results (5 points)

**Exports results to text file (30 points)**

* The text file contains for PyBank:
  + Total Months (2.5 points)
  + Total (2.5 points)
  + Average Change (5 points)
  + Greatest Increase (5 points)
  + Greatest Decrease (5 points)

A screenshot of a computer

Description automatically generated

* The text file contains for Pypoll:
  + Total Votes (2.5 points)
  + Each candidate’s total votes and percent of votes (2.5 points)
  + Winner (5 points)

A screenshot of a computer

Description automatically generated

**Code is cleaned and commented (10 points)**

* Has additional tests and debugging removed (5 points)
* Commented (5 points)

**Grading**

This assignment will be evaluated against the requirements and assigned a grade according to the following table:

| **Grade** | **Points** |
| --- | --- |
| A (+/-) | 90+ |
| B (+/-) | 80–89 |
| C (+/-) | 70–79 |
| D (+/-) | 60–69 |
| F (+/-) | < 60 |

**Submission**

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

**note**

You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Advisor. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

**important**

**It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo**. This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a challenge assignment or any aspect of the academic curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.
2. AskBCS Learning Assistants exists in your class Slack application.
3. Office hours facilitated by your instructional staff before and after each class session.
4. [Tutoring Guidelines](https://docs.google.com/document/d/1hTldEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing) - schedule a tutor session in the Tutor Sessions section of Bootcampspot - Canvas
5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section of your BCS application.

**References**

Data for this dataset was generated by edX Boot Camps LLC, and is intended for educational purposes only.