**Challenge 3 – Brain Dump/Notes**

*More notes on physical notebook*

**Help adding to GitHub – original empty files**

To add a folder to your local Git repository, you can follow these steps using Git Bash:

1. **Navigate to Your Local Repository**: Open Git Bash and navigate to the directory of your local Git repository where you want to add the new folder. You can use the cd command to change directories.
2. **Create a New Folder**: Use the mkdir command to create a new folder. For example, to create a folder named NewFolder, you can run:
3. mkdir NewFolder
4. **Check the Status**: Use the git status command to see the status of your repository. It will show you the untracked new folder.
5. **Add the New Folder**: To stage the new folder for the commit, use the git add command. For example, to add the NewFolder, run:
6. git add NewFolder
7. **Commit the Changes**: After adding the new folder, commit the changes to your local repository with a descriptive message using the git commit command. For example:
8. git commit -m "Add NewFolder to the repository"
9. **Verify the Changes**: You can use git status again to verify that the new folder has been added and committed.

By following these steps, you can add a new folder to your local Git repository using Git Bash.

**Help remembering how to loop through column to find unique months**

Here's an example of how you can loop through a column in a CSV file containing dates in the format "10-Jan" to find the number of unique months:

import csv

# Open the CSV file

with open('data.csv', 'r') as file:

reader = csv.DictReader(file)

# Create a set to store unique months

unique\_months = set()

# Loop through the rows in the CSV file

for row in reader:

date = row['date\_column']

month = date.split("-")[1]

unique\_months.add(month)

# Count the number of unique months

num\_unique\_months = len(unique\_months)

print("Number of unique months:", num\_unique\_months)

In this code snippet, we use the csv module to read the CSV file, loop through the rows, extract the month from the specified column, and store unique months in a set. Finally, we calculate the number of unique months in the column. You can run this code in VSCode to get the desired output without using Pandas.

***Had issues with path***

1. **Verify the File Structure**: Ensure that the folder structure of your project matches the path you are trying to create. Check if the budget\_data.csv file is located in the correct directory relative to your Python script.

***Code for total results in 0. I did something wrong…***

**It seems like the issue in your code is related to how you are reading the CSV file and looping through it. After looping through the file once to count the number of unique months, the file pointer is at the end of the file. When you try to loop through the file again to calculate the total profit/loss, there are no more rows left to read, resulting in a total of 0.**

To address this issue, you can either read the file again from the beginning or store the data you need in memory after the first loop. Here's an updated version of your code that reads the file once and calculates both the total number of months and the net total profit/loss correctly:

import os

import csv

bankpath = os.path.join('..', 'Resources', 'Budget\_data.csv')

# Initialize variables

unique\_months = set()

total\_profit\_losses = 0

# Read the CSV file

with open(bankpath, encoding="utf-8") as bankfile:

bankreader = csv.reader(bankfile)

next(bankreader) # Skip the header row

for row in bankreader:

date = row[0]

month = date.split("-")[0] # Assuming the date format is "MMM-YY"

unique\_months.add(month)

profit\_loss = int(row[1]) # Convert profit/loss to integer

total\_profit\_losses += profit\_loss

# Calculate total number of months

num\_months = len(unique\_months)

print("Total Months:", num\_months)

# Calculate net total Profit/Loss

print("Total: $", total\_profit\_losses)

In this updated code:

* I combined the loops to read the file only once.
* I converted the profit/loss values to integers before summing them up to get the total profit/loss.
* I used the correct format for splitting the date to extract the month.

***Tired to consolidate code to not have issues when trying to do total. Having some difficulties merging above recommendations with my original code idea for the sum***

I see you're trying to calculate the total profit/losses from a CSV file. There are a few issues in your code that need to be addressed:

1. When you read the CSV file, you should skip the header row.
2. The index for the Profit/Losses column is usually 1, not 0.
3. You are trying to convert the profit/loss value to an integer and then split it, which is not necessary for this calculation.

Here is the corrected code snippet for calculating the total profit/losses:

import os

import csv

bankpath = os.path.join('..', 'Resources', 'Budget\_data.csv')

unique\_months = set()

TOTAL\_PROFIT\_LOSSES = 0

with open(bankpath, encoding="utf-8") as bankfile:

bankreader = csv.reader(bankfile)

next(bankreader) # Skip the header row

for row in bankreader:

date = row[0] # Assuming the date is in the first column

month = date.split("-")[0]

unique\_months.add(month)

profit\_loss = int(row[1]) # Assuming profit/loss is in the second column

TOTAL\_PROFIT\_LOSSES += profit\_loss

NUM\_MONTHS = len(unique\_months)

print("Total Months:", NUM\_MONTHS)

print("Total: $", TOTAL\_PROFIT\_LOSSES)

Please replace row[0] and row[1] with the correct indices based on the structure of your CSV file. Additionally, ensure that the profit/loss values in the CSV file are in a format that can be directly converted to an integer.

***Get error when try to use above recc. Was because forgot to skip headers again (had done so before a bunch of changes). Fixed and now working!!***

# get file and using modules for it

import os

import csv

bankpath = os.path.join('..','Resources','Budget\_data.csv')

#initialize starters

unique\_months = set()

TOTAL\_PROFIT\_LOSSES = 0

# Reading using csv module. Open CSV file

with open(bankpath, encoding="utf-8") as bankfile:

    bankreader = csv.reader(bankfile)

    next(bankreader)  # Skip the header row

    for row in bankreader:

        date = row[0]  # the date is in the first column

        month = date.split(",")[0]

        unique\_months.add(month)

        profit\_loss = int(row[1])  # Assuming profit/loss is in the second column

        TOTAL\_PROFIT\_LOSSES += profit\_loss

# Calculate total number of months

NUM\_MONTHS = len(unique\_months)

print("Total Months:", NUM\_MONTHS)

# Calculate net total Profit/Loss

print("Total: $", TOTAL\_PROFIT\_LOSSES)

***difficulty understanding how to track changes in profit/loss. XpertLearning gave following suggestion***

* We create a list changes\_in\_profit\_losses to store the changes in profit/losses over the entire period.
* We iterate through the profit/loss values and calculate the change between the current profit/loss and the previous profit/loss.
* We calculate the average change by summing all changes and dividing by the total number of changes.
* Finally, we print the changes in profit/losses and the average change rounded to two decimal places.

***Xpert suggestion for code snippet***

import os

import csv

bankpath = os.path.join('..', 'Resources', 'Budget\_data.csv')

changes\_in\_profit\_losses = []

previous\_profit\_loss = None

with open(bankpath, encoding="utf-8") as bankfile:

bankreader = csv.reader(bankfile)

next(bankreader) # Skip the header row

for row in bankreader:

profit\_loss = int(row[1]) # Assuming profit/loss is in the second column

if previous\_profit\_loss is not None:

change = profit\_loss - previous\_profit\_loss

changes\_in\_profit\_losses.append(change)

previous\_profit\_loss = profit\_loss

# Calculate the average of changes in Profit/Losses

average\_change = sum(changes\_in\_profit\_losses) / len(changes\_in\_profit\_losses)

print("Changes in Profit/Losses over the entire period:")

print(changes\_in\_profit\_losses)

print("Average Change: $", round(average\_change, 2))

***I adjusted things and it works for the avg change!!***

# get file and using modules for it

import os

import csv

bankpath = os.path.join('..','Resources','Budget\_data.csv')

#initialize starters

unique\_months = set()

TOTAL\_PROFIT\_LOSSES = 0

changes\_in\_profit\_losses = []

previous\_profit\_loss = None

# Reading using csv module. Open CSV file

with open(bankpath, encoding="utf-8") as bankfile:

    bankreader = csv.reader(bankfile)

    next(bankreader)  # Skip the header row

    for row in bankreader:

        date = row[0]  # the date is in the first column

        month = date.split(",")[0]

        unique\_months.add(month)

        profit\_loss = int(row[1])  # Assuming profit/loss is in the second column

        TOTAL\_PROFIT\_LOSSES += profit\_loss

        #if conditional to see changes in current vs previous row

        if previous\_profit\_loss is not None: #[]

            change = profit\_loss - previous\_profit\_loss

            changes\_in\_profit\_losses.append(change)

        previous\_profit\_loss = profit\_loss #for next comparison, make current the "previous"

# Calculate total number of months

NUM\_MONTHS = len(unique\_months)

print("Total Months:", NUM\_MONTHS)

# Calculate net total Profit/Loss

print("Total: $", TOTAL\_PROFIT\_LOSSES)

# Calculate the average of changes in Profit/Losses

average\_change = sum(changes\_in\_profit\_losses) / len(changes\_in\_profit\_losses)

print("Average Change: $", round(average\_change, 2))

***added in calcs for greatest incr and decr and values correct BUT dates keep changing every time I run it***

# get file and using modules for it

import os

import csv

bankpath = os.path.join('..','Resources','Budget\_data.csv')

#initialize starters

unique\_months = set()

TOTAL\_PROFIT\_LOSSES = 0

changes\_in\_profit\_losses = []

previous\_profit\_loss = None

# Reading using csv module. Open CSV file

with open(bankpath, encoding="utf-8") as bankfile:

    bankreader = csv.reader(bankfile)

    next(bankreader)  # Skip the header row

    for row in bankreader:

        date = row[0]  # the date is in the first column

        month = date.split(",")[0]

        unique\_months.add(month)

        profit\_loss = int(row[1])  # Assuming profit/loss is in the second column

        TOTAL\_PROFIT\_LOSSES += profit\_loss

        #if conditional to see changes in current vs previous row

        if previous\_profit\_loss is not None: #[]

            change = profit\_loss - previous\_profit\_loss

            changes\_in\_profit\_losses.append(change) #add each change from each comparison to list with changes

        previous\_profit\_loss = profit\_loss #for next comparison, make current the "previous"

# Calculate total number of months

NUM\_MONTHS = len(unique\_months)

print("Total Months:", NUM\_MONTHS)

# Calculate net total Profit/Loss

print("Total: $", TOTAL\_PROFIT\_LOSSES)

# Calculate the average of changes in Profit/Losses

average\_change = sum(changes\_in\_profit\_losses) / len(changes\_in\_profit\_losses)

print("Average Change: $", round(average\_change, 2))

#now that have list of the changes over the period, can just do min and max for greatest incr and decr.placing here so after loop ends

greatest\_increase = max(changes\_in\_profit\_losses)

greatest\_decrease = min(changes\_in\_profit\_losses) #hopefully works keeping into account negative values

# Find the corresponding dates for the greatest increase and decrease. using index function

index\_greatest\_increase = changes\_in\_profit\_losses.index(greatest\_increase)

index\_greatest\_decrease = changes\_in\_profit\_losses.index(greatest\_decrease)

#make list by converting unique months set. maybe should have made it a list from the beginning?

dates = list(unique\_months)

date\_greatest\_increase = dates[index\_greatest\_increase+1]  # Add 1 to account for skipping the first row

date\_greatest\_decrease = dates[index\_greatest\_decrease+1]  # Add 1 to account for skipping the first row

print("Greatest Increase in Profits:", date\_greatest\_increase, "($", greatest\_increase, ")")

print("Greatest Decrease in Profits:", date\_greatest\_decrease, "($", greatest\_decrease, ")")

#dates are different when I run them? but min and mx number stays the same...

***[]***