Prediction_proj.py (Zain):

This file takes the sample_data csv, and scrapes the costs from the purchases. It adds them up, divides this new number by the number of days in the period to get an average. This average is referred to as the "average amount spent". This is done by the "average" function. The next part of this method is the "determine" function. The average is taken from the first function, but then a value is added to it will be compared against a maximum/minimum expenditure value to determine if something is over or under budget. If the maximum budget is 100, and the average from the average function was 79, then an item worth 21 or more would be deemed over budget. A simple idea, but one that is very practical in actual usage.

Sample_data.csv:

This csv file contains all of the data that we use for our program. It has the dates on which an item was purchased, what the item type is, what specific item it is, and how much the item costs. The csv file will be read in and converted into a dataframe. Also, it will be updated through user input.

Spare_change.ipynb (Dennis):

I take advantage of the csv file made by the user and the data frame created from Ajay's method. This method calculates the spare change of someone's expenses and then tells the user how much they could potentially save if they rounded up. The way this method works is it uses an input from the user asking what their opinion of an expensive item cost. So if they were to enter 100 then any expense in their sheet that is 100$ or greater would be considered an expensive item. We then round to the nearest dollar on regular items and then for the expensive items we round to the nearest ten dollars. At the end we take the summation of all these rounded items and then print a message to the user on how much they could save up if they rounded.

Spendings.py (Ajay and Ellis):

This file contains the code for the line graph and pie chart (created by Ellis). It also contains the code for the user input that updates the .CSV file (created by Ajay). The code is split up into two classes: class Graph (Ellis) and class Spending (Ajay). Everything that was coded under "class Graph" belongs to Ellis and everything coded under "class Spending" belongs to

Ajay. The purpose of this program is to allow the user to update, track, and analyze their spending.

The general flow of the program follows a simple user text based interface. The user will be prompted to enter purchases that they made, followed by committing these purchases to the dataframe. After displaying the user's entered data, the program will output the completed data frame along with a couple of graphs relevant to the user's spending information. One of these graphs is a line chart over time showing the money that the user spent and the dates that they spent it on. The second graph is a pie chart showing the average amount that a user spent on the different categories of items in their purchases. These graphs will display whether the user adds purchases or not.

- ***Instructions:***
    - In the command line, just type "python3 spendings.py" and it should run
    - You can type "0" to exit the program at any time during the input menus

Annotated Bibliography

Ellis:

1. “Pandas.dataframe.plot.” *Pandas.DataFrame.plot - Pandas 0.23.1 Documentation*, https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html.

The purpose of using this source was to develop the general code that was needed for a time series graph. Specifically, the layout of the .plot() function, as I had not worked with this function before and had no idea how to use it. The .plot() function was used in both the graph and avg_pie methods. The reason I used this specific source is because I was trying to plot data from a dataframe. The modifications that I had made were changing the x and y values, along with adding in a marker and a title to the graph to make it more visually appealing. This graph displayed how much was spent on each day.

2. “Basic Pie Chart.” *Basic Pie Chart - Matplotlib 3.5.1 Documentation*, https://matplotlib.org/stable/gallery/pie_and_polar_charts/pie_features.html.

I had used information from the first source to help develop the structure for the pie graph. Since I had the general structure from the graph method, all I had to do was change the kind = pie. However, for the other parameters such as autopct and explode, I had to use the source above to set those up. As I had no knowledge of how to work with pie charts in python. The reason I wanted those two parameters in the pie chart is because I wanted to display the percentages on each slice and create a more visually appealing pie chart with the explode parameter. This pie chart displayed how much was spent on average for each specific item type.