Code Logic:

1) I have placed the sample ‘transactions.csv’ file and the Python script ‘main/py’ in the same directory.

2) I have created 2 user-defined functions and 1 custom Exception class.

3) The 1st function is ‘convert_file_to_dataframe(file_name):’
   - In this function I pass the name of the file in the .csv format and it returns a Python data frame after reading the file and using the read_csv() method available in pandas module.

4) The 2nd function is ‘calculate_points(input_dataframe, input_points):’
   - In this function, I pass the data frame created by the 1st function and the points to be spent as the inputs.
   - I have sorted the data frame in ascending order on the basis of the timestamp values. To ensure, that the row numbers do not get re-indexed after sorting, I have set the ‘ignore_index’ parameter to True.
   - The idea here is to spend the old points first and keep on subtracting the points spent from the input_points.
   - So, I have used a ‘while’ loop with condition that the input_points > 0 and also the iterator variable < the length of the input_dataframe.
   - Both these conditions need to be True or else we may get exceptions.
   - If the value of points at the intersection of ith row and points column > input_points, I will subtract the input_points from the value at the cell and that will be the remaining points for that user. Basically, if the points to spend are less than the points for a specific payer, then the difference will be the points remaining for that payer. Then we exit from the loop because we have 0 points available to spend.
   - If the value of points at the intersection of ith row and points column < input_points, then I will firstly subtract the value at the cell from the input_points. In addition to this, I will make the points for the specific payer as 0. This means, if the points to spend are more than the points for a specific payer, then the difference will be the points available for further spending and that payer will have 0 points left with them.
   - Once the loop ends completely, I have grouped the points by the payer and the grouping is done with the sum() aggregation to calculate the total of points remaining as balance for each payer.
   - This will give us the final data frame which will contain the name of the payers and their total balance points.

5) The NegativePointsException(Exception) class is defined by me to check whether the points entered by user in command line arguments > 0. This is because, we want to spend points and this number cannot be negative. In case, the user provides a negative number, it will throw an Exception and say that negative values are not allowed and the program will terminate.

6) The program execution starts from the main function and we accept the points as input from the command line.

7) We have explicitly specified the file name ‘transactions.csv’ in the code. If the file is not resent in the directory, it will raise an Exception saying the file is not found and the program will terminate.

8) The ‘convert_file_to_dataframe(file_name)’ function is called and the file name is passed to it which returns the data frame.

9) Then the ‘‘calculate_points(input_dataframe, input_points)’ is called and the data frame and the command line points are passed. This return the final data frame which has the solution.