**Name:** Jaeger Liebster

**Date:** 09/28/2025

**Program Description:** This program provides a tool for users to track their monthly expenses. It interactively prompts the user to enter the type of expense (e.g., "Rent", "Groceries") and the corresponding monetary amount. Once the user has input their expenses, the program analyzes the data using Python's reduce method to calculate and display a summary, which includes the total monthly expense, the single highest expense, and the single lowest expense.

**Functions used in the Program (list in order as they are called):**

1. **Function Name:** get\_expenses()
   * **Description:** This function handles the user interaction for collecting expense data. It runs a loop that repeatedly asks the user for an expense type and its amount, creating a list of dictionary items. The loop terminates when the user provides empty input for the expense type.
   * **Parameters:** None
   * **Variables:**
     1. expenses (list): An empty list that is populated with dictionaries, where each dictionary represents a single expense ({'type': str, 'amount': float}).
     2. expense\_type (str): Stores the user's input for the name of the expense.
     3. amount\_str (str): Stores the user's raw input for the expense amount before conversion to a float.
     4. amount (float): The validated and converted numerical value of the expense.
   * **Logical Steps:**
     1. Initialize an empty list called expenses.
     2. Start an infinite while loop to continuously prompt the user.
     3. Ask the user for the "Type of expense".
     4. If the input is empty, break the loop.
     5. Start an inner while loop for amount validation.
     6. Ask for the "Amount" for the specified type.
     7. Use a try-except block to ensure the input is a valid positive number. If not, print an error and re-prompt.
     8. If the amount is valid, append a new dictionary {'type': expense\_type, 'amount': amount} to the expenses list.
   * **Returns:** expenses (list): The list of all collected expense dictionaries.
2. **Function Name:** analyze\_expenses(expenses)
   * **Description:** This function takes the list of expenses and performs all the calculations. It uses the reduce function from the functools module to determine the total, highest, and lowest expenses without using traditional loops.
   * **Parameters:** expenses (list): The list of expense dictionaries generated by get\_expenses().
   * **Variables:**
     1. total\_expense (float): The sum of all expense amounts, calculated by reduce.
     2. highest\_expense (dict): The dictionary from the list that has the largest 'amount' value, found by reduce.
     3. lowest\_expense (dict): The dictionary from the list that has the smallest 'amount' value, found by reduce.
   * **Logical Steps:**
     1. Check if the expenses list is empty. If so, print a message and return.
     2. Calculate total\_expense using reduce with a lambda function that accumulates the amounts (lambda total, expense: total + expense['amount']).
     3. Find highest\_expense using reduce with a lambda that compares the 'amount' of two expenses and returns the greater one.
     4. Find lowest\_expense using reduce with a lambda that compares the 'amount' of two expenses and returns the lesser one.
     5. Print the formatted results in a clear summary table.
   * **Returns:** None
3. **Function Name:** main()
   * **Description:** This is the main entry point of the program. It controls the overall program flow by calling the other functions in the correct sequence.
   * **Parameters:** None
   * **Variables:** user\_expenses (list): Stores the list of expenses returned by the get\_expenses function.
   * **Logical Steps:**
     1. Print a welcome message to the user.
     2. Call the get\_expenses() function and store its return value in user\_expenses.
     3. Call the analyze\_expenses() function, passing user\_expenses to it as an argument.
   * **Returns:** None

**Logical Steps:** The program execution starts when the script is run, which triggers the if \_\_name\_\_ == "\_\_main\_\_": block.

1. The main() function is called.
2. Inside main(), the get\_expenses() function is called to collect all expense data from the user.
3. Once the user is finished, get\_expenses() returns the list of expenses to main().
4. main() then calls analyze\_expenses() and passes the collected list to it.
5. analyze\_expenses() performs the calculations and prints the final report to the console.

**Link to your repository:**

<https://github.com/RoarinThundah/COP2373_Jaeger/upload/main>

**Output Screenshot:**

