Name:

Luke Seavey

Date Created:

09/28/25

Program Description:

This program scans an email's text to check the likelihood of it being spammail based on common words found in said emails.

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

1. Function Name: add\_expenses

Description:

Add the expenses together to find the total.

Parameters:

* Total - running total of expenses
* expense - name, amount tuple

Variables:

* No variables

Logical Steps:

1. Extracts expense[1]
2. Return total + amount

Returns:

The total of the expenses added together.

1. Function Name: higher\_expense

Description:

Calculates which expense imputed is the highest value.

Parameters:

* Current - current highest expense
* Expense - next expense to compare

Variables:

* No variables

Logical Steps:

1. Compare current[1] to expense[1]
2. Return the tuple with the highest amount

Returns:

* The highest expense among the ones imputed.

3. Function name: lower\_expense

Description:

Calculates which expense imputed has the lowest value.

Parameters:

* Current - current lowest expense
* expense - next expense to compare

Variables:

* No variables

Logical Steps:

1. Compare current[1] to expense[1]
2. Return tuple with the smaller amount

Returns:

* The lowest expense among the ones imputed.

1. Function Name: main

Description:

Main function that asks the user for a list of expense types and prices and handles errors and prints the total, lowest, and highest expenses with their labels.

Parameters:

* No parameters

Variables:

* Expenses - list of the expense type and the amount
* Expense\_type - is the label associated with the expense value
* Amount - is the amount entered for the expense
* Total\_expenses - is the total of all the expenses
* Highest\_expense - is the highest cost of the expenses
* Lowest\_expense - is the lowest cost of the expenses

Logical Steps:

1. Prompt user for each expense type and amount until done is called
2. Use reduce(add\_expense, expenses, 0) to get total\_expenses
3. Use reduce(higher\_expense, expenses, 0) to get highest\_expense
4. Use reduce(lower\_expense, expenses, 0) to get lowest\_expense
5. Print formatted results

Returns:

* The printed results for total expenses, highest expense, and lowest expense back to the user.

Logical steps:

1. Gathers a list of expenses via user input
2. Validate and store each expense
3. Compute total, highest and lowest using reduce with helper functions
4. Display the results with clear labels

List of functions in order:

1. Main
2. Add\_expenses
3. Higher\_expense
4. lower\_expense

Link to your repository:

<https://github.com/FennecAce/COP2373/tree/c65d582fd05b3d6d111e4712b224b17601c0383c/LukeSeavey_ProgrammingExercise_3>

Output Screenshot:

