**Programming Exercise#3**

**Name:** Chris McCathern

**Date Created:** September 23, 2025

**1. Function Name: collect\_expenses**

* **Description: Collects expense type and amount from the user until they type "done".**
* **Parameters: None**
* **Variables:**
  + **expenses - list to store tuples of (expense\_type, amount).**
  + **expense\_type → string entered by user.**
  + **amount - float entered by user.**
* **Logical Steps:**
  + **Prompt the user for expense type.**
  + **Stop if the input is "done".**
  + **Prompt for amount and validate numeric input.**
  + **Append (expense\_type, amount) to the expenses list.**
  + **Repeat until finished.**
* **Returns: List of tuples (expense\_type, amount).**

**2. Function Name: calculate\_total**

* **Description: Uses reduce to sum all expense amounts.**
* **Parameters: expenses (list of tuples).**
* **Variables:**
  + **acc - accumulator that keeps running total.**
  + **x - current tuple being processed.**
* **Logical Steps:**
  + **Start with 0 as initial value.**
  + **For each tuple, add x[1] (the amount) to the total.**
* **Returns: Float representing the total of all expenses.**

**3. Function Name: calculate\_highest**

* **Description: Finds the highest expense (tuple).**
* **Parameters: expenses (list of tuples).**
* **Variables:**
  + **a, b - two tuples being compared.**
* **Logical Steps:**
  + **Compare amounts of two tuples.**
  + **Keep the one with the higher value.**
  + **Repeat until all compared.**
* **Returns: A tuple (expense\_type, amount) with the highest expense.**

**4. Function Name: calculate\_lowest**

* **Description: Finds the lowest expense (tuple).**
* **Parameters: expenses (list of tuples).**
* **Variables:**
  + **a, b - two tuples being compared.**
* **Logical Steps:**
  + **Compare amounts of two tuples.**
  + **Keep the one with the lower value.**
  + **Repeat until all compared.**
* **Returns: A tuple (expense\_type, amount) with the lowest expense.**

**5. Function Name: main**

* **Description: Orchestrates the program flow.**
* **Parameters: None**
* **Variables:**
  + **expenses - list of all expense tuples.**
  + **total, highest, lowest -results from calculations.**
* **Logical Steps:**
  + **Call collect\_expenses().**
  + **If no expenses, display message and stop.**
  + **Call calculate\_total(), calculate\_highest(), and calculate\_lowest().**
  + **Print results with clear labels.**
* **Returns: None (prints results).**

**Logical Steps in Program Execution (Order of Calls)**

1. **Program starts at main().**
2. **collect\_expenses() is called to gather user input.**
3. **calculate\_total(expenses) is called.**
4. **calculate\_highest(expenses) is called.**
5. **calculate\_lowest(expenses) is called.**
6. **Results are displayed to the user.**

**Link to your repository:** [cmac2371/McCathern\_ProgrammingExercise\_3](https://github.com/cmac2371/McCathern_ProgrammingExercise_3)

A screen shot of a computer

AI-generated content may be incorrect.