**Week 3 algorithm**

Expense Tracker Program

This Python program allows users to track their daily expenses. It includes functionality for user input, data storage,expense categories, data analysis, a basic user interface, and error handling.

1. Function Descriptions:

- load\_data():

- Loads existing expense data from the 'expenses.json' file.

- Handles potential errors like FileNotFoundError and JSONDecodeError.

- Returns a dictionary containing the loaded data.

- save\_data(data):

- Saves the current expense data to the 'expenses.json' file.

- Takes a dictionary 'data' as input and writes it to the file in JSON format with indentation.

- add\_expense(data, date, category, amount):

- Adds a new expense entry to the expense data.

- Parameters:

- data: The dictionary containing existing expense data.

- date: The date of the expense in the format 'YYYY-MM-DD'.

- category: The expense category (e.g., 'Groceries', 'Utilities').

- amount: The expense amount, a floating-point number.

- Updates the data dictionary and calls save\_data to persist the changes.

- show\_summary(data):

- Displays a summary of expenses, organized by date and category.

- Parameters:

- data: The dictionary containing expense data.

- Prints a formatted summary with dates, categories, and corresponding amounts.

- main():

- The main entry point of the program

- Initializes the expense data by calling load\_data().

- Runs a loop for the user interface, allowing users to add expenses, view summaries, or exit the program.

- Error handling is implemented for invalid user inputs (e.g., non-numeric amounts).

2. Program Structure:

- The program starts by loading existing data from 'expenses.json'.

- Inside the main loop, the user is presented with options to add expenses, view summaries, or exit the program.

- If the user chooses to add an expense, the program prompts for date, category, and amount, with error handling.

- The add\_expense function is then called to update the data.

- Choosing to view a summary invokes the show\_summary function to display the expenses.

- The program continues running until the user chooses to exit.

3. Usage:

- The user interacts with the program through the command-line interface.

- The program provides clear prompts and messages to guide the user.

- The expense data is stored in a JSON file for persistence between program executions.

4. Note:

- The program assumes a simple structure where each expense has a date, category, and amount.

- The expense data is stored in a JSON file for ease of use and data persistence.