



K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

NAME – DRISHTI

ROLL NO. – 2501730232

SUBJECT – PROGRAMMING FOR PROBLEM SOLVING
USING PYTHON

SUBMITTED TO – MR. SAMEER FAROOQ

PROJECT REPORT 1

Project Name Daily Calorie Tracker

Goal To create a simple, command-line utility for logging daily meals, calculating total and average calorie intake, and providing immediate feedback against a user-defined daily limit.

Language Python

Key Concepts Used Lists, Loops (for), Conditional Logic (if/else), Data Type Casting, Built-in Functions (sum, len), File I/O.

Script Architecture and Data Flow

The program follows a sequential, six-task structure, as outlined below:

Task 1: Setup & Introduction: Displays a welcome message and explains the tool's purpose.

Task 2: Input & Data Collection:

Initial input determines num_meals.

A for loop collects meal names and calorie amounts.

Data is stored in two linked lists: meals (string) and calories (float).

Task 3: Calorie Calculations:

Calculates total_calories using sum(calories).

Calculates average_calories using total_calories / len(calories).

Prompts the user for the daily_limit (float).

Task 4: Exceed Limit Warning System:

Uses an if/else statement to compare total_calories against daily_limit.

Sets a status_message (e.g., "🟢 Great job!" or "⚠️ Warning") for the final report.

Task 5: Neatly Formatted Output: Generates the structured terminal report.

Task 6: Save Session Log to File (Bonus): Writes the report to calorie_log.txt.

Conclusion and Next Steps

The Daily Calorie Tracker is a complete and functional command-line application. It successfully abstracts complex data management into a simple, interactive process for the user.

Strengths:

Clarity: Clear input prompts and a well-formatted, easy-to-read report.

Utility: Provides immediate, actionable feedback (the limit warning).

Extensibility: The data structure (parallel lists) is simple and easily adaptable for future features (e.g., adding macronutrient tracking).

Potential Improvements:

Error Handling: Implement try-except blocks to handle non-numeric input (e.g., if a user types "ten" instead of 10 for calories).

Data Structure: Switch to a list of dictionaries (e.g., [{'name': 'Breakfast', 'cal': 350.0}]) for more robust linking of meal attributes.

Cumulative Tracking: Modify the file saving (Task 6) to append to the log ("a") instead of overwriting ("w") to maintain a history of daily reports.

