# Smart Finance Tracker (Python CLI Application)

## 🎯 Project Overview

This is a simple, command-line interface (CLI) application developed in Python to help users track their personal income and expenses. It provides a quick way to record financial transactions, view a comprehensive summary, and analyze spending patterns by category.

The project is designed to demonstrate proficiency in fundamental Python programming concepts, particularly those covered in an introductory course like **CSE1021: Introduction to Problem Solving and Programming**.

## ✨ Features

The application provides the following core functionalities through an interactive menu:

1. **Add Income**: Record a new income transaction with amount, source, and description.
2. **Add Expense**: Record a new expense transaction with amount, category, and description.
3. **View Summary & Breakdown**:
   * Displays **Total Income**, **Total Expense**, and **Net Balance**.
   * Provides a **Categorical Breakdown** of expenses, including the percentage each category contributes to the total spending, offering a basic form of financial analysis.
4. **View All Transactions**: Lists all recorded transactions with clear, color-coded formatting for easy readability.
5. **Save & Exit**: Automatically saves all transaction data to a local finance\_data.json file before exiting, ensuring data persistence.

## 💻 How to Run

1. **Prerequisite**: Ensure you have Python installed on your system (Python 3.6+ is recommended).
2. **Save the File**: Save the provided code as finance\_tracker.py.
3. **Execute**: Open your terminal or command prompt, navigate to the directory where you saved the file, and run:  
   python finance\_tracker.py
4. **Interact**: Follow the on-screen menu instructions (e.g., enter 1 for Add Income, 3 for Summary, etc.).

## 🎓 Educational Concepts Demonstrated

This project successfully integrates several key programming concepts from the course curriculum:

| **Course Concept (Unit/Experiment)** | **Implementation in Tracker** |
| --- | --- |
| **Lists and Dictionaries** (Unit 5) | Used the TRANSACTIONS **List** to hold all records, and each record is stored as a **Dictionary**. Dictionaries are also used for category\_totals in the summary. |
| **Input and Output** (Exp. 2) | Used input() to receive transaction details and print() with ANSI color codes for enhanced, readable output. |
| **Functions** (Unit 2) | Modularized the code using functions like add\_transaction(), view\_summary(), load\_data(), and save\_data(). |
| **Control Flow (if/else, while)** (Unit 3, Exp. 4) | The main\_menu() uses a while True loop and if-elif-else statements to control the application flow based on user choice. |
| **Summation** (Unit 3) | Implemented summation logic within view\_summary() to calculate total\_income, total\_expense, and balance. |
| **Data Persistence** | Used the standard json library to save and load data, preventing data loss between sessions. |