Authors: Peter Bauschlicher & Carl Frédéric Beckmann Date: 19.12.2025

CO2 Accounting Tool - Group Project

Course: Skills: Programming - Introduction Level (HSG)

1. Project Overview

This tool automates the process of assigning carbon footprint data to business invoices. It acts as a bridge between financial data and environmental impact analysis by reading invoice line items, matching them against a "Keyword Database" of emission factors, and generating a detailed report.

Key Features:

- Smart Matching Engine (Level 3): Uses a sophisticated 4-stage algorithm (Normalization, Mapping, Keyword Search, Word-by-Word Fuzzy Matching) to handle inconsistent naming, synonyms (e.g., Uber/Taxi), and typos in invoice data.
- Professional Engineering Standards: The codebase adheres to best practices including Type Hints (str, float), Constants for configuration, and Intent-Based Comments.
- Modular Architecture: Logic is strictly separated from data loading and reporting for maximum maintainability.
- Built-in Unit Testing: Includes a self-check mechanism to validate logic before processing data.

2. Requirements

Python 3.x
No external libraries are required (uses standard csv, difflib, os, sys).

3. Project Structure

The project is organized into modular files to separate concerns:

- main.py: The entry point. Handles the execution flow and defines configuration constants (e.g., filenames).
- data_loader.py: Handles robust file I/O, error checking, and CSV parsing.
- logic.py: The core "Engine". Contains the fuzzy matching algorithm, emissions calculations, and the Unit Test block.
- analytics.py: Handles the presentation layer, generating console summaries and exporting CSV results.
- invoices.csv: Input data containing raw invoice line items.
- emission_factors.csv: The knowledge base acting as a keyword map (includes specific items like "macbook" and aliases like “uber").

4. How to Run

- Ensure all .py and .csv files are in the same folder.
- Open your terminal or command prompt and navigate to the folder.
- To Run the Main Program:
- python3 main.py
- Output: A console summary and a new file named invoice_with_co2_results.csv.
- To Run the Logic Self-Check (Unit Test):

- python3 logic.py
- Output: Verifies that the matching engine is correctly identifying exact matches, substrings, and typos.

5. Implementation Details (The "Smart" Engine)

- In logic.py, we implemented a get_emission_factor function that goes beyond simple exact matching. It now follows a 4-step priority algorithm:
- Normalisation: Input strings are converted to lowercase and stripped of whitespace to ensure " Laptop " equals "laptop".
- Data Mapping: The database is loaded into a hash map for O(1) lookup speed.
- Keyword Matching: Checks if a known category exists as a substring (e.g., finding "coffee" inside "Premium Coffee Beans").
- Word-by-Word Fuzzy Matching: The description is split into individual words. Each word is checked against the database using difflib with a 70% similarity cutoff. This allows the system to catch specific typos (e.g., "Ppr" successfully matches "paper") without triggering false positives on the full sentence.