Republic of the Philippines
CAMARINES SUR POLYTECHNIC COLLEGES
Nabua, Camarines Sur

COLLEGE *of* COMPUTER STUDIES

PROJECT DOCUMENTATION

CSEC 311- Defensive Programming

Submitted by:

GONZALES, Mark Angelo L.
PALOMARES, Juan Paolo C.

BSCS-3A

Project Overview

1.1 Description of the Original Project

The "All-in-One Converter" application is a Python-based GUI application created using Tkinter. The application integrates multiple types of converters (such as currency, temperature, weight, length, and area) in a single, user-friendly interface. It allows users to easily perform conversions by entering a value and selecting the units for conversion.

1.2 Rationale for Choosing This Project

This project was chosen to demonstrate skills in Python, GUI programming with Tkinter, and modular application design. The purpose is to build a practical, everyday tool that simplifies conversions, which can be useful for both educational and personal purposes. Additionally, the project offers the potential for further expansion by adding more types of conversions and enhancing the user interface.

1.3 Summary of Improvements and Enhancements

The following improvements were made in the project:

- **Integrated multiple conversion types** into a single, easy-to-navigate GUI.
- **Added hover effects** on labels for enhanced user interaction.
- **Modularized code** to make each conversion type isolated, allowing easy maintenance and scalability.
- **Improved user interface** with intuitive placement of buttons and input fields for better usability.

---

2. Installation and Setup Guide

2.1 Steps to Set Up the Development Environment

1. **Install Python**: Download and install Python 3.x from the official website: https://www.python.org/downloads/.

   Install Tkinter: Tkinter is the standard GUI library for Python and is usually pre-installed with Python. If not, install it using:

2.2 Dependencies and Requirements

- Python 3.6 or higher.
- Tkinter (for the GUI).
- External library for real-time currency conversion (e.g., `API from fixer.io`).

---

3. User Manual

3.1 Instructions on How to Use the Application

1. **Launch the Application**: Execute the Python script to open the application window.
2. **Choose a Converter**: Select one of the conversion types by clicking the corresponding buttons (e.g., Currency, Temperature, Length, Weight, Area).
3. **Enter Input Value**: Input a numerical value in the designated input field.
4. **Select Units for Conversion**: Choose the appropriate units for both the "from" and "to" fields.
5. **Perform Conversion**: Click the "Convert" button to see the result.

3.2 Description of New Features and Functionalities

- **Hover Effects**: Labels on the interface change color when hovered, making the application more interactive.
- **Modular Conversion Functions**: Each type of conversion (currency, temperature, weight, etc.) has its own function, which allows for future scalability.
- **User-Friendly Interface**: Buttons and input fields are well-organized for easy navigation and use.

---

4. Technical Documentation

4.1 Architectural Changes

The architecture follows a modular design pattern. Each converter (Currency, Temperature, Length, Weight, and Area) is implemented as an individual function. This structure ensures that the code is easy to maintain and extend, as new converters can be added without affecting the existing ones.

4.2 Description of New Classes, Methods, and Functions

- **CurrencyConverter()**: Converts between different currencies using real-time exchange rates fetched from an API.
- **TemperatureConverter()**: Converts values between Celsius, and Fahrenheit
- **LengthConverter()**: Converts between various units of length (e.g., meters, kilometers, miles).
- **AreaConverter()**: Converts between different units of area (e.g., square meters, acres).
- **WeightConverter()**: Converts between various units of weight (e.g., kilograms, pounds).

4.3 Explanation of Defensive Programming Techniques Implemented

- **Error Handling**: The program uses `try-except` blocks to catch any errors, such as invalid input or issues with fetching data from external sources (like the currency API).
- **Input Validation**: Only numerical inputs are allowed for conversion, and non-numeric values prompt the user to enter valid numbers.

---

5. Code Review and Refactoring

5.1 Discussion of Major Code Changes

- **Refactoring the Currency Converter**: The original currency conversion function was refactored to utilize the `forex-python` library for real-time exchange rates.
- **Optimizing the Layout**: The widgets were rearranged to improve readability and the flow of the application.
- **Modularization**: The conversion functions were separated into their respective modules, improving the maintainability of the code.

5.2 Justification for Refactoring Decisions

- **Improved Code Readability**: Breaking down the code into smaller, dedicated functions makes it easier to understand and debug.
- **Maintainability**: With each conversion module isolated, adding new converters or updating existing ones becomes much simpler.
- **Efficiency**: Refactoring made the overall structure more efficient and responsive, especially when handling multiple types of conversions.

---

6. Testing Documentation

6.1 Overview of Testing Strategy

Testing was performed manually by inputting different values and verifying the correctness of the output for each converter. Functional testing was used to ensure the converters performed as expected.

6.2 Description of Test Cases and Their Results

1. **Currency Converter Test**:
    - **Input**: 100 USD, Convert to EUR.
    - **Expected Output**: Conversion result based on real-time exchange rates.
    - **Result**: Passed successfully.
2. **Temperature Converter Test**:
    - **Input**: 25°C, Convert to Fahrenheit.
    - **Expected Output**: 77°F.
    - **Result**: Passed successfully.
3. **Weight Converter Test**:
    - **Input**: 50 kilograms, Convert to pounds.
    - **Expected Output**: 110.23 pounds.
    - **Result**: Passed successfully.

---

7. Challenges and Solutions

7.1 Discussion of Difficulties Encountered During the Project

- **Handling Multiple Conversions**: Initially, the application struggled with managing multiple conversion types in a single interface.
- **Currency Conversion Accuracy**: Fetching real-time exchange rates from an external API introduced challenges related to API limits and error handling.

7.2 How These Challenges Were Addressed

- **Modularized Code**: Code refactoring helped isolate each conversion type, making it easier to manage and scale.
- **Error Handling for External API**: Implemented error handling to manage situations where the currency converter could not retrieve data from the API.

---

8. Future Improvements

- **Adding More Conversion Types**: Future versions of the application could support additional conversions, such as volume, speed, and energy.
- **Unit Testing**: Implement automated unit tests to validate the functionality of each converter.
- **UI Enhancements**: Enhance the user interface with additional features, such as real-time updates and graphical representations of conversion data.

9. Sources:

URL: Prajwal10031999/All-in-one-converter-using-python: Converter for currency, area, temperature, weights and also an app to check current sensex, gold rates and silver too...!!!