Project Report: Python Currency Converter

Submitted by: [Ishant Agrawal] **Course:** B.Tech (Computer Science) - [2025]

1. Abstract

This project aims to develop a simple, efficient, and user-friendly command-line tool for currency conversion. In a globalized world, the need to convert values between different monetary units is frequent. This application uses Python to perform accurate conversions between a wide variety of international currencies using the United States Dollar (USD) as a base reference point.

2. Introduction

2.1 Problem Statement

Travelers, students, and international businesses often need to estimate costs in different currencies. Manual calculation is error-prone, especially when dealing with cross-rates (e.g., converting Indian Rupees directly to Euros without knowing the direct rate).

2.2 Objective

The primary objective of this project is to:

- Create a program that accepts a monetary amount from the user.
- Accept a source currency code and a target currency code.
- Compute the converted amount using stored exchange rates.
- Handle errors, such as invalid currency codes or non-numeric inputs.

3. Methodology and Approach

3.1 The "Base Currency" Approach

Instead of storing a conversion rate for every possible pair of currencies (which would require hundreds of combinations like INR-to-EUR, INR-to-GBP, EUR-to-JPY, etc.), this project uses a **Pivot Currency approach**.

- **Base Currency:** United States Dollar (USD).
- **Logic:** Every currency has a defined rate against 1 USD.
- **Conversion Path:** To convert from Currency A to Currency B, the system first converts A to USD, and then USD to B.

3.2 Algorithm

1. **Input:** Get Amount (X), Source Currency (S), and Target Currency (T).
2. **Step 1 (Normalize):** Convert S to USD.

$$\text{USD Value} = \frac{X}{\text{Rate of } S}$$

3. **Step 2 (Target):** Convert USD to T.

$$\text{Final Value} = \text{USD Value} \times \text{Rate of } T$$

4. **Output:** Display Final Value.

4. Implementation Details

4.1 Technical Specifications

- **Language:** Python 3
- **Interface:** Command Line Interface (CLI)
- **Data Structure:** Python Dictionary (`dict`) was used to map currency codes (Keys) to their exchange rates (Values). This provides $O(1)$ average time complexity for lookups.

4.2 Key Code Components

A. The Data Source A dictionary named `exchange_rates` stores the hardcoded values.

- *Example:* `"INR": 89.56` means 1 USD = 89.56 Indian Rupees.
- *Scope:* Covers major currencies (EUR, GBP), Asian markets (JPY, INR), and others (African, South American).

B. The Conversion Function The core logic is encapsulated in a function `convert_currency` :

```
def convert_currency(amount, from_currency, to_currency):
    # 1. Convert Input to USD
    amount_in_usd = amount / exchange_rates[from_currency]

    # 2. Convert USD to Target
    converted_amount = amount_in_usd * exchange_rates[to_currency]

    return converted_amount
```

C. Error Handling

- `try-except` **block:** Used to catch `ValueError` if the user inputs text instead of numbers for the amount.
- **Dictionary Checks:** The code verifies if the entered currency codes exist in the database before processing.

5. Results

The project was tested with various test cases to ensure accuracy.

Test Case 1: Standard Conversion

- **Input:** Amount: `100` , From: `USD` , To: `EUR`
- **Logic:** 100×0.869
- **Output:** `86.9`

Test Case 2: Cross Conversion (No USD involved in input)

- **Input:** Amount: `5000` , From: `INR` , To: `GBP`
- **Process:**
  1. INR to USD: $5000 / 89.56 \approx 55.82$ USD
  2. USD to GBP: $55.82 \times 0.763 \approx 42.59$ GBP
- **Output:** `42.5906...`

Test Case 3: Invalid Input

- **Input:** Amount: `"Hello"`
- **Output:** "Invalid amount. Please enter a number." (Program prevents crash and asks again).

6. Analysis and Limitations

6.1 Strengths

- **Simplicity:** The code is lightweight and runs instantly without installation.
- **Extensibility:** Adding a new currency is as simple as adding one line to the dictionary.
- **Robustness:** Handles capitalization automatically (e.g., inputs "inr" and "INR" both work).

6.2 Limitations

- **Static Rates:** The exchange rates are hardcoded. In the real world, rates change every second. This program reflects the rates at the time of coding.
- **Interface:** It lacks a Graphical User Interface (GUI), which might be less appealing to non-technical users.

7. Future Scope

To improve this project in the future, the following features could be added:

1. **API Integration:** Connect to a live API (like *OpenExchangeRates* or *Fixer.io*) to fetch real-time data instead of hardcoded values.
2. **GUI:** Build a frontend using `Tkinter` or `PyQt` for a window-based experience.
3. **History:** Save a log of recent conversions to a text file.

8. Conclusion

This project successfully demonstrates the application of Python data structures and basic algorithms to solve a real-world problem. It reinforces the importance of modular programming (using functions) and proper error handling in software development.