

Currency Converter Project Documentation

1. Introduction

The Currency Converter (15x15) is a Python-based software tool designed to simplify the process of converting one currency into another. It addresses the daily need for currency conversion for students, travelers, businesses, and individuals involved in global transactions. Unlike online converters that require internet access and external APIs, this project offers an offline, easy-to-use, accurate, and efficient solution capable of converting between 15 international currencies using predefined exchange rates. Built with Python for simplicity and compatibility, it demonstrates essential programming concepts such as dictionaries, conditionals, mathematical operations, and user interaction via the command-line interface.

2. Functional Requirements

- FR1: Currency Selection:** User can select a source currency (FROM) from a list of 15 currencies.
- FR2: Target Currency Selection:** User can select a destination currency (TO) from the same list of 15 currencies.
- FR3: Amount Input:** User can input the amount they wish to convert.
- FR4: Conversion Process:** The system must convert the amount using INR as the internal base (Source Currency → INR → Target Currency).
- FR5: Display Output:** The system must display the final converted amount clearly with two decimal precision.
- FR6: Error Handling:** The system must detect invalid inputs (e.g., numbers outside 1–15, non-numeric values, empty inputs).
- FR7: Offline Operation:** The system must work without an internet connection.

3. Non-Functional Requirements

- NFR1: Usability:** Easy navigation via simple menu-based input.
- NFR2: Performance:** Instantaneous conversion without noticeable delay.
- NFR3: Reliability:** Consistent and accurate performance for all supported currencies.
- NFR4: Portability:** Runs on any platform supporting Python (Windows, macOS, Linux).
- NFR5: Maintainability:** Exchange rates and currency lists are easy to update within the code.
- NFR6: Simplicity:** Uses simple logic for beginner understanding.

4. System Architecture

The system follows a three-layer model:

Input Layer: Accepts user choices for FROM currency, TO currency, and amount; validates input correctness.

Processing Layer: Handles core logic, including fetching exchange rates, converting to INR, converting INR to the target currency, and calculating the final output.

Output Layer: Displays the converted amount or messages for invalid input/errors.

Central Logic: INR acts as an internal base currency to simplify conversion among 15 currencies.

5. Design Decisions & Rationale

Decision 1: Using a Python dictionary *Rationale:* Dictionaries offer fast key-value access, making currency codes easy to store, modify, and retrieve.

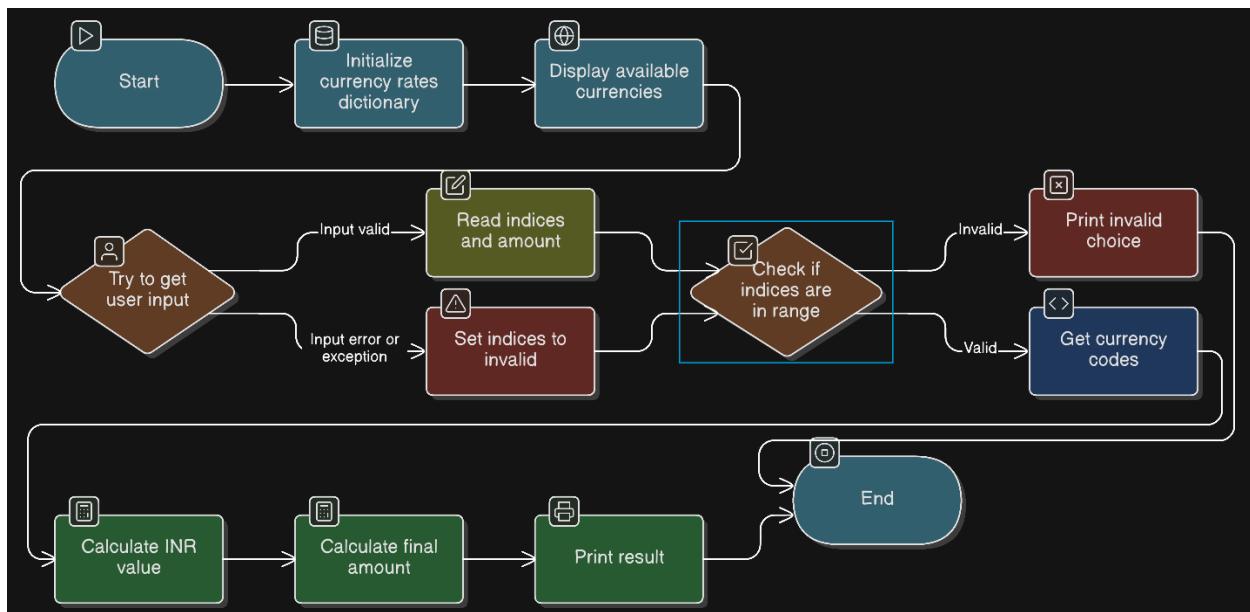
Decision 2: Using INR as internal base *Rationale:* Instead of calculating 15x15 combinations, conversion through INR simplifies logic and reduces complexity.

Decision 3: Using menu-driven CLI *Rationale:* A command-line interface is simple, universal, and ideal for beginners.

Decision 4: Avoiding external APIs *Rationale:* Ensures offline operation and independence from network conditions.

Decision 5: Error checking for user input *Rationale:* Improves user experience and prevents program crashes.

6. Implementation Details



Implemented in Python using simple built-in functions. Currency rates are stored in a `currency_rates` dictionary. Input is taken using the `input()` function. The conversion method is: Amount in source currency → INR → target currency. Outputs are formatted using `:.2f` for clarity. Basic exception handling (`try-except`) manages invalid input gracefully.

7. Results (Text Version)

The screenshot shows a terminal window with the following content:

```
File Edit Selection View Go Run Terminal Help
home > annant-pundir > Downloads > currency-converter-project38 (1) > currency_converter_project38.py > ...
10 except:
11     # Quick fallback, not elegant but works
12     print("Something went wrong with your input.")
13     from_idx, to_idx = -1, -1 # force invalid
14     amt = 0
15
16     # Quick bounds check
17     if 1 <= from_idx <= 15 and 1 <= to_idx <= 15:
18         # grabbing the currency codes
19         # maybe should've done better naming but whatever, works for now
20         from_curr = curr_list[from_idx - 1]
21         to_curr = curr_list[to_idx - 1]
22
23         # Converting FROM -> INR -> target currency
24         # Doing this in two steps so it's easier to read (though a single line would work)
25         inr_value = amt * currency_rates[from_curr]
26
27         # Slightly unnecessary variable but makes the next line clearer in my head
28         final_amount = inr_value / currency_rates[to_curr]
29
30         print(f"\n{amt} {from_curr} is approximately {final_amount:.2f} {to_curr}\n")
31
32     else:
33         print("\nInvalid choice! (I'll probably add better error messages later.)")
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
```

TERMINAL

```
annant-pundir@annant-pundir-VivoBook-ASUSLaptop-X515EA-X1500EA:~$ /bin/python3 ~/home/annant-pundir/Downloads/currency-converter-project38 (1)/currency_converter_project38.py
12) ZAR
13) AED
14) SAR
15) KRW
Currency to convert FROM (1-15): 5
Currency to convert TO (1-15): 6
Enter the amount you'd like to convert: 34345
34345.0 CAD is approximately 3888615.18 JPY
```

Ln 60, Col 1 Spaces: 4 UFT-8 LF {} Python ⌂ ⌂

Example output:

==== Currency Converter (15x15) ===

Available Currencies:

- 1) USD
- 2) EUR
- 3) GBP
- ...
- 15) KRW

Currency to convert FROM (1-15): 1

Currency to convert TO (1-15): 2

Enter the amount you'd like to convert: 100

100 USD is approximately 91.15 EUR

Plain TextCopy

8. Testing Approach

Test 1: Valid Inputs Conversions between all 15 currencies with typical values (1, 100, 1000).

Test 2: Invalid Inputs Entering letters instead of numbers, currency indices outside the range (0 or 16), or negative values.

Test 3: Edge Cases Converting the same currency (result should be the same), zero amount (output should be zero).

Test 4: Stability Testing Re-running the program multiple times to check consistency.

9. Challenges Faced

Ensuring accurate conversion without live rates.
Designing a clean user-flow for beginners.
Handling invalid input without program crashes.
Maintaining readability while supporting 15 currencies.

10. Learnings & Key Takeaways

Effective use of Python dictionaries.
Understanding menu-driven programming.
Improved input validation and error handling skills.
Gaining experience in creating program documentation.
Learning how offline systems manage conversion logic.

11. Future Enhancements

GUI version using Tkinter or PyQt.
Live currency rates using an API (e.g., forex API).
Adding support for 50+ currencies.
Saving conversion history to a file.
A mobile app version.
Adding voice input and output.