

PROJECT REPORT

Currency Converter – A Python-Based Exchange Rate Calculator

1. Introduction

The Currency Converter project is a Python implementation of a command-line application that enables users to convert monetary amounts between different international currencies. The application uses a dictionary-based system to store exchange rates relative to the US Dollar (USD) and provides instant conversion results with accurate exchange rate information.

The program is designed to be simple, efficient, and user-friendly, allowing anyone to quickly convert between 10 major world currencies without requiring internet connectivity or external dependencies.

2. Objectives

The primary objectives of this project are:

- To implement currency conversion logic using Python programming
 - To apply concepts of functions, dictionaries, input validation, and error handling
 - To practice mathematical calculations for multi-currency conversions
 - To create a modular and maintainable code structure
 - To develop an interactive command-line application with clear user feedback
 - To demonstrate proper data organization using Python dictionaries
 - To enhance problem-solving skills in real-world financial applications
-

3. Tools and Technologies Used

Programming Language: Python 3

Modules Used:

- Built-in functions (no external libraries required)
- Dictionary data structure for exchange rate storage
- Exception handling (try-except blocks)

- String formatting for output display

Development Environment: Visual Studio Code / PyCharm / Terminal-based Python environment

Core Python Concepts:

- Functions and modular programming
 - Dictionaries and key-value pairs
 - User input handling
 - Type conversion and validation
 - Mathematical operations
 - Formatted string output
-

4. Problem Description

In international finance and travel, currency conversion is a fundamental requirement. People need to:

- Convert money when traveling abroad
- Calculate costs in different currencies for online shopping
- Understand exchange rates for international transactions
- Compare prices across different countries

This application solves these needs by:

- Providing instant currency conversion between 10 major currencies
- Displaying accurate exchange rates
- Validating user inputs to prevent errors
- Offering a simple command-line interface
- Working completely offline with pre-loaded exchange rates

Supported Currencies:

- USD (US Dollar)
- EUR (Euro)
- GBP (British Pound)
- INR (Indian Rupee)

- JPY (Japanese Yen)
 - AUD (Australian Dollar)
 - CAD (Canadian Dollar)
 - CHF (Swiss Franc)
 - CNY (Chinese Yuan)
 - AED (UAE Dirham)
-

5. System Design

5.1 Workflow

1. Initialize Exchange Rates

- Load pre-defined exchange rates relative to USD

2. Display Welcome Screen

- Show program title
- List all available currencies

3. Accept User Input

- Source currency code
- Target currency code
- Amount to convert

4. Validate Inputs

- Check if currencies exist in the system
- Validate amount is a valid number

5. Perform Conversion

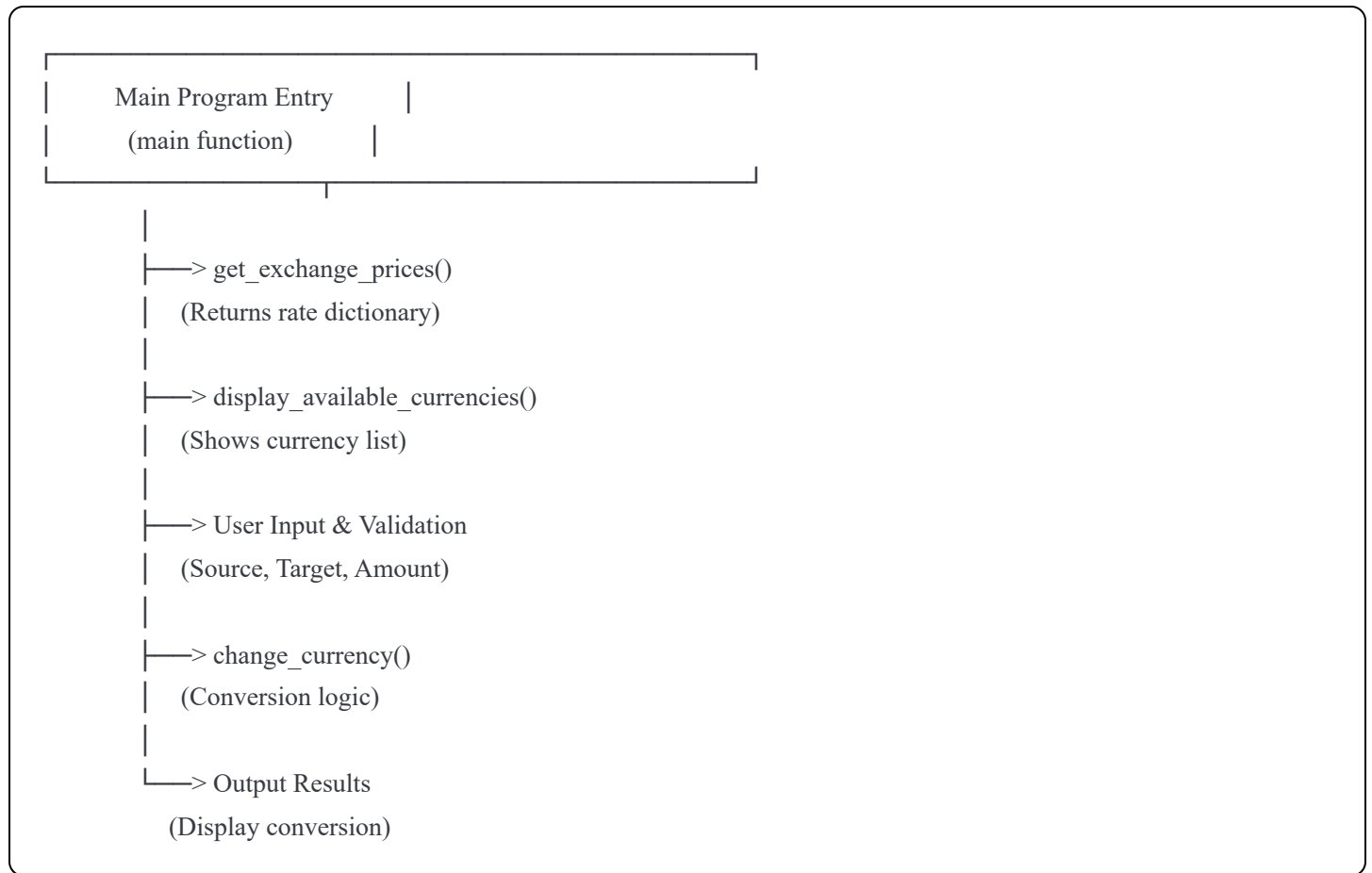
- Convert source currency to USD (base currency)
- Convert USD to target currency
- Calculate exchange rate between the two currencies

6. Display Results

- Show converted amount
- Display exchange rate

- Format output clearly

5.2 System Architecture



6. Code Explanation (Block-by-Block)

6.1 Exchange Rate Storage ((get_exchange_prices))

Purpose: Stores all exchange rates relative to USD (base currency)

Key Features:

- Dictionary structure for efficient lookup
- USD = 1.0 as the base reference
- Easy to update rates when needed
- Contains 10 major world currencies with their conversion rates

6.2 Currency Conversion Logic ((change_currency))

Purpose: Converts amount from source to target currency

Algorithm:

1. **Validation:** Check if both currencies exist in the system
2. **Step 1:** Convert source currency to USD
 - Formula: $\text{USD_amount} = \text{input_amount} / \text{source_rate}$
3. **Step 2:** Convert USD to target currency
 - Formula: $\text{target_amount} = \text{USD_amount} \times \text{target_rate}$
4. **Return:** Converted amount or None if error occurs

Example:

- Convert 100 INR to EUR
- Step 1: $100 / 83.12 = 1.203$ USD
- Step 2: $1.203 \times 0.92 = 1.107$ EUR

Result: 100 INR = 1.11 EUR

6.3 Display Function (`display_available_currencies`)

Purpose: Shows all supported currency codes to the user

Functionality:

- Iterates through the exchange rate dictionary
- Displays each currency code in a formatted list
- Provides clear visibility of available options

Benefits:

- User knows which currencies are available
 - Prevents invalid currency entries
 - Improves user experience
 - Reduces input errors
-

6.4 Main Program Flow (`main`)

Purpose: Orchestrates the entire currency conversion process

Process Flow:

1. Initialization

- Load exchange rates from the dictionary
- Display welcome header and program title

2. Display Available Currencies

- Show list of all supported currency codes
- Help users choose valid options

3. User Input Collection

- Accept source currency code
- Accept target currency code
- Convert inputs to uppercase for consistency

4. Currency Validation

- Check if source currency exists in the system
- Check if target currency exists in the system
- Display error message and exit if invalid

5. Amount Input with Error Handling

- Prompt user to enter amount to convert
- Use try-except block to catch invalid numeric input
- Convert string input to float for calculations
- Handle ValueError for non-numeric entries

6. Conversion Execution

- Call the `change_currency` function
- Pass amount, source currency, target currency, and rates
- Receive converted amount as result

7. Result Display

- Format and display converted amount (2 decimal places)
- Calculate direct exchange rate between the two currencies
- Display exchange rate (4 decimal places for precision)
- Show formatted output with clear separators

Key Features:

- **Input Validation:** Checks currency existence before processing
 - **Exception Handling:** Catches invalid numeric input gracefully
 - **Case Insensitivity:** Converts input to uppercase automatically
 - **Clear Output:** Formatted results with appropriate decimal places
 - **Exchange Rate Display:** Shows rate between selected currencies
 - **Error Messages:** Provides helpful feedback for user mistakes
-

7. Sample Output

Example 1: Successful Conversion

```
=====
CURRENCY CONVERTER
=====

Available currencies:
- USD
- EUR
- GBP
- INR
- JPY
- AUD
- CAD
- CHF
- CNY
- AED

Enter source currency: USD
Enter target currency: INR
Enter amount in USD: 100

=====
100.0 USD = 8312.00 INR
Exchange Rate: 1 USD = 83.1200 INR
=====
```

Example 2: Invalid Currency

Available currencies:

- USD
- EUR
- INR
- ...

Enter source currency: XYZ

Enter target currency: INR

Error: XYZ is not available!

Example 3: Invalid Amount

Enter source currency: USD

Enter target currency: EUR

Enter amount in USD: abc

Invalid amount! Please enter a number.

8. Features and Functionality

8.1 Core Features

✅ **Multi-Currency Support:** 10 major world currencies ✅ **Bi-directional Conversion:** Convert from any currency to any other ✅ **Accurate Calculations:** Uses mathematical conversion through USD base ✅ **Exchange Rate Display:** Shows the rate between selected currencies ✅ **Input Validation:** Prevents invalid currency codes and amounts ✅ **Error Handling:** Graceful handling of user input errors ✅ **Offline Operation:** No internet connection required ✅ **Zero Dependencies:** Uses only Python standard library

8.2 Technical Features

- **Modular Design:** Separate functions for each task
 - **Dictionary-Based Storage:** Efficient data structure
 - **Type Conversion:** Proper handling of string and numeric types
 - **Formatted Output:** Clean and professional display
 - **Case Insensitive Input:** Accepts both upper and lowercase
-

9. Applications

This project demonstrates understanding of:

Financial Applications:

- International money transfers
- Travel budgeting
- E-commerce price comparison
- Investment portfolio calculations

Programming Concepts:

- Function design and modularity
- Data structures (dictionaries)
- Input validation and error handling
- Mathematical operations
- String formatting
- User interaction design

Real-World Use Cases:

- Personal finance management
 - Business transactions
 - Educational tool for learning exchange rates
 - Foundation for more complex financial applications
-

10. Advantages and Limitations

Advantages

- ✓ Simple and easy to use
- ✓ Fast execution
- ✓ No internet required
- ✓ No external dependencies
- ✓ Lightweight application
- ✓ Easy to maintain and update
- ✓ Cross-platform compatible

Limitations

- ✗ Exchange rates are static (not live)
 - ✗ Limited to 10 currencies
 - ✗ Requires manual rate updates
 - ✗ Command-line interface only
 - ✗ No transaction history
 - ✗ No support for cryptocurrency
-

11. Future Enhancements

Potential improvements for this project:

1. **Live Exchange Rates:** Integrate API for real-time data
 2. **More Currencies:** Expand to 50+ currencies
 3. **Graphical Interface:** Create GUI using Tkinter or PyQt
 4. **Conversion History:** Save previous conversions
 5. **Rate Trends:** Show historical rate changes
 6. **Batch Conversion:** Convert multiple amounts at once
 7. **Currency Symbols:** Display proper symbols (₹, \$, €, £)
 8. **Reverse Calculator:** Calculate source amount from target
 9. **Favorite Pairs:** Save frequently used currency pairs
 10. **Export Results:** Save conversions to file
-

12. Conclusion

The Currency Converter project is a practical demonstration of applying fundamental Python programming concepts to solve a real-world problem. It showcases clean code structure, proper error handling, and user-friendly design principles.

This application serves as an excellent learning tool for understanding:

- Function-based programming
- Data structure usage
- User input validation

- Mathematical computations
- Program flow control

The project successfully achieves its objectives of creating a functional, efficient, and maintainable currency conversion tool that can be easily extended and improved for more advanced features.

13. References and Resources

- Python Official Documentation: <https://docs.python.org/3/>
 - Exchange Rate Data: Based on approximate market rates
 - Programming Best Practices: PEP 8 Style Guide
 - Financial Calculations: Standard currency conversion formulas
-

Project Metadata:

- **Language:** Python 3.x
 - **Type:** Command-Line Application
 - **Domain:** Financial Tools
 - **Complexity:** Beginner to Intermediate
 - **Lines of Code:** ~60
 - **Functions:** 4
-

End of Report