CURRENCY CONVERTER USING PYTHON

BY

KRITHEESHWAR.S

1. Aim of the project

This project outlines the development of a reliable and user-friendly tool for converting currency amounts accurately and efficiently. By leveraging Python's built-in functionality and without relying on external packages, the program aims to simplify the process of currency conversion for individuals and businesses engaged in international transactions. Through its intuitive interface, predefined exchange rates, and robust error handling mechanisms, the Currency Converter Program seeks to enhance the user experience and facilitate seamless currency conversions, thereby contributing to improved financial management and decision-making in a globalized world.

2. Problem Statement

Currency conversion is essential for individuals and businesses engaged in international transactions. However, manually calculating conversions can be time-consuming and prone to errors. Hence, the need for an automated currency conversion tool arises. The objective of this project is to develop a user-friendly program that can convert currencies quickly and accurately, thereby simplifying the process for users. By developing the Currency Converter Program, we aim to provide individuals and businesses with a reliable tool to facilitate currency conversions, thereby enabling better financial management

3. Project Objectives

The objective of the Currency Converter Project is to develop a robust and user-friendly Python application that enables users to convert currency amounts accurately and efficiently. By leveraging Python's built-in functionality and without relying on external packages, the project aims to provide a lightweight and intuitive tool for performing currency conversions in a globalized context.

Key objectives of the project include:

Implementing a mechanism to store and manage exchange rates between various currencies.

Designing an algorithm to perform currency conversions based on user input, ensuring accuracy and reliability.

Incorporating robust error handling mechanisms to handle invalid inputs and unexpected errors gracefully, enhancing the user experience.

Providing a simple and intuitive user interface for seamless interaction, catering to users with varying levels of technical proficiency.

Conducting thorough testing to ensure the accuracy, reliability, and usability of the application under different scenarios and edge cases.

4. Functionality

The program follows these steps:

User Input: The program prompts the user to enter the following information:
oAmount: The value in INR to be converted (e.g., 1000 INR)
oTarget Currency: The desired currency to convert to (e.g., USD, EUR, etc.) The program will include a list of supported currencies.

Conversion Logic:
The program utilizes a pre-defined function to store exchange rates relative to INR. (e.g., exchange_rates = {"USD": 0.014, "EUR": 0.012, ...})
Based on user input, the program retrieves the corresponding exchange rate from the dictionary.
The conversion calculation is performed: converted_amount = amount * exchange_rate[target_currency].
Output: The program displays the converted amount in the target currency with appropriate formatting (e.g., "1000 INR is equivalent to 14 USD").

5. Implementation

The program code will consist of the following elements:

Input Functions: Python's input() function will be used to capture user entries for amount and target currency.

Conditional Statements: The program might employ an if-else statement to handle conversions for unsupported currencies.

Output Formatting: String formatting techniques will be used to display the converted amount in a user-friendly manner.

Currency converting program :-

```
exchange_rates = {                                  #Assigning values
    'INR': 1.0,
    'USD': 0.013,                                  # 1 INR = 0.013 USD
    'EUR': 0.011,                                  # 1 INR = 0.011 EUR
    'GBP': 0.0097,                                 # 1 INR = 0.0097 GBP
    'JPY': 1.48,                                   # 1 INR = 1.48 JPY
    'CAD': 0.017,                                  # 1 INR = 0.017 CAD
    'CNY': 0.083,                                  # 1 INR = 0.083 CNY
    'SEK': 0.12,                                   # 1 INR = 0.12 SEK
}

def convert_currency(amount, from_currency, to_currency):
    if from_currency not in exchange_rates or to_currency not in exchange_rates:
        raise ValueError("Invalid currency code :- \n")


# Convert from 'from_currency' to INR, then from INR to 'to_currency'

    amount_in_inr = amount / exchange_rates[from_currency]
    converted_amount = amount_in_inr * exchange_rates[to_currency]

    return converted_amount



while True:
    try:
        amount = float(input("Enter the amount to be converted: "))
```

```
        from_currency = input("Enter the currency code you want to convert from (e.g.,
INR, USD, EUR, GBP, JPY): ").upper()
        to_currency = input("Enter the currency code you want to convert to (e.g., INR,
USD, EUR, GBP, JPY): ").upper()

        converted_amount = convert_currency(amount, from_currency, to_currency)
        print(f"{amount} {from_currency} is equal to {converted_amount:.2f}
{to_currency}")

        break
    except ValueError as e:
        print(e)
    except Exception as e:
        print("An error occurred:", e)
```

Output:

```
            Enter the amount to be converted:  1
Enter the currency code you want to convert from (e.g., INR, USD, EUR, GBP, JPY):
  ABC
Enter the currency code you want to convert to (e.g., INR, USD, EUR, GBP, JPY):  I
NR
Invalid currency code :-

Enter the amount to be converted:  165
Enter the currency code you want to convert from (e.g., INR, USD, EUR, GBP, JPY):
  INR
Enter the currency code you want to convert to (e.g., INR, USD, EUR, GBP, JPY):  U
SD
165.0 INR is equal to 1.98 USD
```

6.Error Handling and Exceptional Handling

Error handling is the process of responding to and managing errors that occur during the execution of a program. It ensures that the program can handle unexpected situations without crashing and provides meaningful feedback to the user. In the context of this currency conversion program, error handling is implemented using try-except blocks to manage potential errors that could arise during user input and currency conversion.

Error Handling in the Program

Invalid Currency Code:

1.Potential Error: The user might input a currency code that does not exist in the exchange_rates dictionary.

2.Handling: The convert_currency function checks if the provided currency codes (from_currency and to_currency) exist in the exchange_rates dictionary. If either code is invalid, it raises a ValueError with a specific error message indicating the invalid currency code.

Invalid Numeric Input:

1.Potential Error: The user might input a non-numeric value for the amount to be converted.

2.Handling: The program attempts to convert the user input for the amount to a float. If the input is not a valid number, a Value Error is raised, and the program catches this error to prompt the user again.

General Exception Handling:

1.Potential Error: Unexpected errors might occur during the execution of the program.

2.Handling: The program includes a general except block to catch any other exceptions that are not specifically handled. This prevents the program from crashing and provides a generic error message.

7. Limitations and Benefits

Static Exchange Rates: The program relies on pre-defined exchange rates, resulting in non-real-time conversions.

Limited Currency Support: The program can only handle a finite set of currencies included in the exchange rate dictionary.

Benefits

Learning Tool: This project serves as a practical introduction to Python concepts like variables, functions, dictionaries, and user input.

Self-Contained: The program demonstrates building functionality without external dependencies, promoting a foundational understanding of Python programming.

8. Future Enhancements

Dynamic Exchange Rate Updates: Integrate a mechanism to periodically update exchange rates from a reliable online source (e.g., web scraping).

Error Handling: Implement robust error handling to address invalid user input, unsupported currencies, or potential calculation errors.

Graphical User Interface (GUI): Consider developing a GUI using libraries like Tkinter to provide a more interactive user experience.

9. Conclusion

This currency converter project offers valuable insights into core Python programming principles. By avoiding external libraries, it emphasizes the fundamentals of building programs from scratch. The project serves as a stepping stone for creating more sophisticated currency converter applications with features like real-time exchange rates, a wider range of supported currencies, and error handling capabilities.