Python Project On

“Currency Converter “

Submitted by: Pankaj Kumar Singh

UID: 24MCA20067

Section/Group: 2/A

Submitted To: Ms.Pawandeep Sharma

Course: MCA

Date Of Submission: 25/10/2024

July 2024 – November 2024

University Institute Of Computing

Chandigarh University

Mohali, Punjab

Project: Currency Converter Application in Python using Tkinter

Objective:

The goal of this project is to create a simple currency converter application that allows users to convert between different currencies based on exchange rates provided in INR (Indian Rupees). The application will have a graphical user interface (GUI) developed using Tkinter, where users can input the amount, select the "from" and "to" currencies, and perform the conversion.

Steps to Complete the Project:

1. Setting up the Environment:

- Install Python (if not installed already) on your system.
- Tkinter is included in Python by default, but if it's not installed, you can install it using `pip install tk`.
- Create a Python file (e.g., `currency_converter.py`) to write your code.

2. Understanding the Components of the Application:

The application contains several components:

- **Exchange Rate Data**: A dictionary (`exchange_rates_to_inr`) that holds the currency codes as keys and their corresponding value in INR.
- **Countries Data**: A dictionary (`countries`) that maps country names to their currency codes.
- **Conversion Logic**: The function `convert_currency()` that handles the logic for converting currencies.
- **User Interface (UI)**: Tkinter widgets (like labels, entry fields, comboboxes, and buttons) to create an intuitive interface for the user.

3. Code Breakdown and Explanation:

Step 1: Import the required modules.

```
import tkinter as tk from tkinter

import ttk, messagebox
```

- tkinter is used to create the GUI.
- ttk (Themed Tkinter Widgets) provides modern-looking widgets.
- messagebox is used to show warning and error messages.

Step 2: Define the exchange rates and country data.

```
exchange_rates_to_inr = { ... } countries

= { ... }
```

- These dictionaries provide the core data needed for conversion. The exchange rates dictionary contains currency codes and their values in INR.

Step 3: Create the convert_currency() function.

- This function takes an amount, a "from" currency, and a "to" currency.
- First, the amount is converted to INR (Indian Rupees).
- Then, the INR value is converted to the desired target currency using the provided exchange rates.
- If the conversion is successful, it returns the converted value; otherwise, it returns None.

```
def convert_currency(amount, from_currency, to_currency):

  if from_currency in exchange_rates_to_inr:
```

```
        amount_in_inr = amount * exchange_rates_to_inr[from_currency]

    else:

        return None

    if to_currency in exchange_rates_to_inr:

        converted_amount   =   amount_in_inr   /   exchange_rates_to_inr[to_currency]                    return
converted_amount

    else:

 return None
```

Step 4: Create the perform_conversion() function to handle user input.

```
def perform_conversion():

    try:

        amount = float(amount_entry.get())

        from_currency = from_currency_combobox.get().upper()  # Normalize to uppercase
to_currency = to_currency_combobox.get().upper()      # Normalize to uppercase



        if from_currency and to_currency and amount > 0:

            converted_amount = convert_currency(amount, from_currency, to_currency)            if
converted_amount is not None:
                result_label.config(text=f"{amount:.2f} {from_currency} =
{converted_amount:.4f} {to_currency}")

            else:

                messagebox.showerror("Error", "Currency conversion failed. Please check your input.")

        else:

            messagebox.showwarning("Warning", "Please enter a valid amount and select currencies.")
except ValueError:

        messagebox.showerror("Error", "Please enter a valid numeric amount.")
```

- This function retrieves user input (amount, from currency, and to currency).
- It validates the input (ensures that the amount is numeric and greater than zero).
- If the input is valid, it performs the conversion by calling convert_currency().
- The result is displayed in a label, or an error message is shown if the input is invalid.

Step 5: Create the User Interface (UI) using Tkinter.

```
root = tk.Tk() root.title("Currency
Converter") root.geometry("400x400")
root.configure(bg="#e9ecef")
```

- This initializes the main window of the application with a title and dimensions.
- The root.configure(bg="#e9ecef") sets the background color of the window.

Step 6: Add widgets for user interaction.

- **Title Label:**

```
title_label = tk.Label(root, text="Currency Converter", font=("Helvetica", 18, "bold"), bg="#e9ecef",
fg="#007BFF") title_label.pack(pady=20)
```

- Displays the application title in a styled font and color.

- **Amount Input Field:**

```
amount_label = tk.Label(root, text="Amount to Convert:", bg="#e9ecef", font=("Helvetica", 12)) amount_label.pack(pady=5) amount_entry = tk.Entry(root, font=("Helvetica", 12), bd=2, relief=tk.SUNKEN) amount_entry.pack(pady=5, padx=20, fill=tk.X)
```

- A label and an input field where users can type the amount they want to convert.

- **From and To Currency Dropdowns:**

```
from_currency_combobox = ttk.Combobox(root, values=list(countries.values()), font=("Helvetica", 12))

to_currency_combobox = ttk.Combobox(root, values=list(countries.values()), font=("Helvetica", 12))
```

- These dropdowns allow the user to select the "from" and "to" currencies.

- **Convert Button and Result Display:**

```
convert_button = tk.Button(root, text="Convert", command=perform_conversion, bg="#007BFF", fg="white", font=("Helvetica", 12)) convert_button.pack(pady=20) result_label = tk.Label(root, text="", bg="#e9ecef", font=("Helvetica", 12)) result_label.pack(pady=5)
```

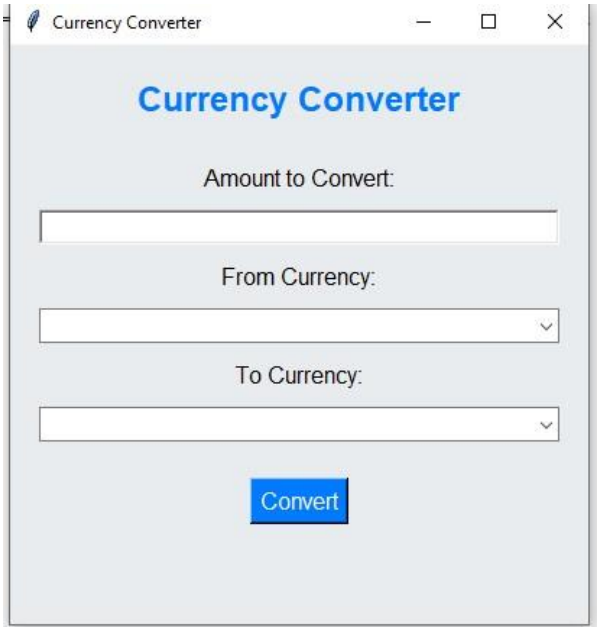
- The convert_button triggers the currency conversion when clicked.
- The result_label displays the conversion result.

Step 7: Run the Tkinter main loop.

```
root.mainloop()
```

- This command starts the application’s event loop, waiting for user interaction.

OUTPUT:



Conclusion:

In this project, we built a functional currency converter using Python's Tkinter for the GUI. The application demonstrates the ability to work with dictionary data for currency rates, input validation, and performing calculations to convert currency values.