CURRENCY CONVERTER

A PYTHON-BUILT SOFTWARE

This report and project are written, developed and designed by-

NAME: ABHISHEK SAHU

REG. NO.: 11915545

ROLL NO.: 63

NAME: HRITHIK MAURYA

REG. NO.: 11915604

ROLL NO.: 52

Introduction

It seems easy when we calculate our accounts, savings or anything related to money. We just have to use a calculator and we are free. However, we do not live in a combined world, this world we live in is divided in many continents, countries etc. Also, we do not use same currency in every country or this does not mean in context of name only, the exchange rates change everyday and it's hard to keep record for everything, this is not the only job we have to do in a day. So, why remember all of this when we can build and use our own currency converter!

This report features a currency converter model built absolutely in python with the use of its inbuilt modules and functions. This report tells us about the detailed production and design of the function coded inside the currency converter software. It tells us how we can convert our desired amount into 33 different currencies and also obtain exchange rate graphs.

INDEX

Inbuilt Modules

```
import json
import requests
from pandas import *
from tkinter import *
from tkinter import ttk
import matplotlib.pyplot as plt
from matplotlib.figure import Figure
from currency_converter import CurrencyConverter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,NavigationToolbar2Tk)
```

This is the real advantage in python. We don't have to write every single detail for our use, we can import inbuilt modules and modify them accordingly. This software also consists of modules like these shown in above image. Functionality of the modules is written below:

1. tkinter : tkinter is our most important module in this software. Our main focus is to built a user friendly software and as we all know not everybody is a coder and specially who uses currency converter is definitely not. Our targeted audience is not friendly with coding in majority. So, our main focus is to make it easy, so every single person with zero experience in coding can use it. Tkinter helps us build GUI for user's convenience. Tkinter can be downloaded by using pip install tk-tools.

2. Currency_converter: our second focus is this module. This is our main goal right, GUI is just making it friendly but the main work is on this module and this is the beauty of this module. Just so easy to work with and it gives us our 34 countries with which we can convert our amount. This can be downloaded by using pip install currencyconverter.

3. JSON: JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. JSON represents objects as name/value pairs, just like a Python dictionary. This module is already installed in python.

4. Requests: The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data. This module can be downloaded by using pip install requests.

5. Pandas: Well, I assure you we did not use a real panda in our software and this is just a name. In real, pandas focus on data structures and operations for manipulating numerical tables and time series.

6. Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter.

Software functionalities

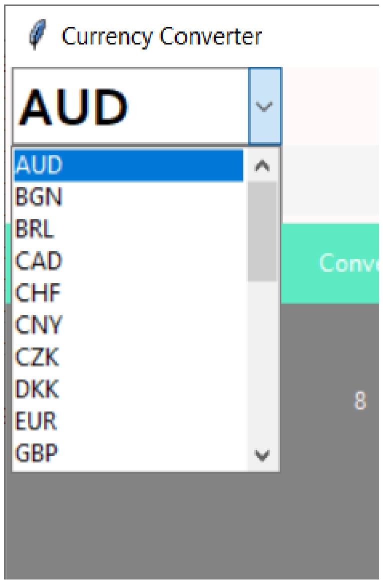
Main_frame: we used a main frame which will consists of all the buttons combobox and other functions in our tkinter window.

```
#Main part. All the global declaration is done below this.
window=Tk()#main window
window.title('Currency Converter')
window.configure(bg='#FFFAFA')
main_frame=Frame(window)#main frame which consists all the child frames.
main_frame.pack(side=LEFT)
main_frame.configure(bg='#fffafa')
```

Background color: ice white(#FFFAFA)

Window name: Currency Converter

From_country: country from which we convert our amount(can be considered as our base country). We used a combobox method in our software to show the list of the countries from which we can choose our base country and convert our amount. This from_country combobox consists of 34 countries from which we can convert our amount to another currency. This looks like the image shown below.



Frame: from_frame

Text variable: from_country_name

Font: corbel

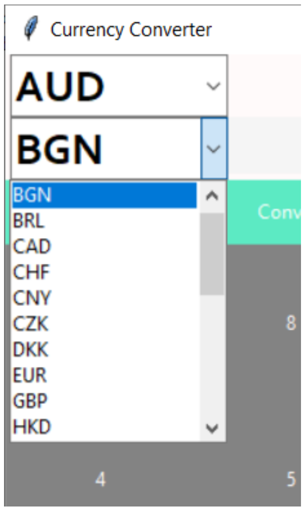
Current value: AUD

We use get() command to get the value selected by user and use it in our convert command which is invoked after we press the convert button. Till then the value is stored in text variable of the combobox.

Code:

```
from_frame=Frame(main_frame)

from_frame.configure(bg='#FFFAFA')

from_frame.pack()

from_country_name=StringVar()#holds the value of selected country.

from_countries=ttk.Combobox(from_frame,width=8,state='readonly',font=('corbel',20,'bold'),textv
ariable=from_country_name)

from_countries['values']=sorted(('EUR','CAD', 'HKD', 'ISK', 'PHP', 'DKK', 'HUF', 'CZK', 'AUD', 'RON',
'SEK', 'IDR', 'INR', 'BRL', 'RUB', 'HRK', 'JPY', 'THB', 'CHF', 'SGD', 'PLN', 'BGN', 'TRY', 'CNY', 'NOK',
'NZD', 'ZAR', 'USD', 'MXN', 'ILS', 'GBP', 'KRW', 'MYR'))

from_countries.current(0)#sets the default value.

from_countries.pack(anchor=W,side=LEFT)
```

To_country: country to which our amount will be converted after pressing the convert button.). We used a combobox method in our software to show the list of the countries to which we can choose our desired country and convert our amount. This to_country combobox consists of 34 countries to which we can convert our amount from our base country. This looks like the image shown below.



Frame: to_frame

Text variable: to_country_name

Font: corbel

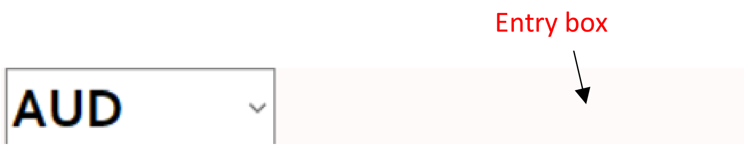
Current value: BGN

We use get() command to get the value selected by user and use it in our convert command which is invoked after we press the convert button. Till then the value is stored in text variable of the combobox.

Code:

```
#to_frame part which consists a combobox with the list of 33 countries to which we want our amount to be converted and the value of converted amount.

to_frame=Frame(main_frame)

to_frame.configure(bg='#FFFAFA')

to_frame.pack()

to_country_name=StringVar()

to_countries=ttk.Combobox(to_frame,width=8,state='readonly',font=('corbel',20,'bold'),textvariabl
e=to_country_name)

to_countries['values']=sorted(('EUR','CAD', 'HKD', 'ISK', 'PHP', 'DKK', 'HUF', 'CZK', 'AUD', 'RON',
'SEK', 'IDR', 'INR', 'BRL', 'RUB', 'HRK', 'JPY', 'THB', 'CHF', 'SGD', 'PLN', 'BGN', 'TRY', 'CNY', 'NOK',
'NZD', 'ZAR', 'USD', 'MXN', 'ILS', 'GBP', 'KRW', 'MYR'))

to_countries.current(1)

to_countries.pack(side=LEFT,anchor=W)
```

Amount: this is an entry box which will consists of the user input which is a valid float or int amount to convert into another currency to our base currency.



Frame: from_frame

Font: Candara

Background color: ice white(#FFFAFA)

Vairiable name: amount

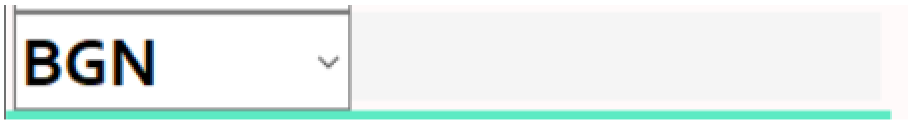
Justify: right

We use get() command to get the value entered by user and convert it in our convert command which is invoked after we press the convert button.

Code:

```
#amount which we want to convert.

amount=Entry(from_frame,width=15,bg='#FFFAFA',font=('candara',20,'bold'),justify=RIGHT,bd=0)

amount.pack(side=LEFT)
```

converted_amount: this is a label which will print the converted value processed in convert command and it will also show an error in case of an error is encountered.



Frame: to_frame

Font: Candara

Background color: light grey(#F5F5F5)

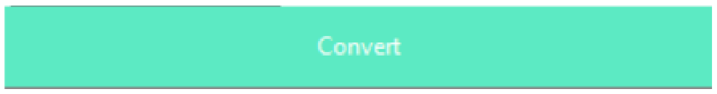
Variable name: converted_amount

Justify: right

Code:

```
converted_amount=Label(to_frame,text='',anchor=E,bg='#F5F5F5',width=14,font=('candara',20,'bo
ld'),bd=0)

converted_amount.pack()
```

convert_button: this is a button designed to use the main functionality of our software and after pressing it invokes convert command.



Frame: main_frame

Color: sea green(#5CEAC3)

Font: calibri

Command: convert

Foreground color: white

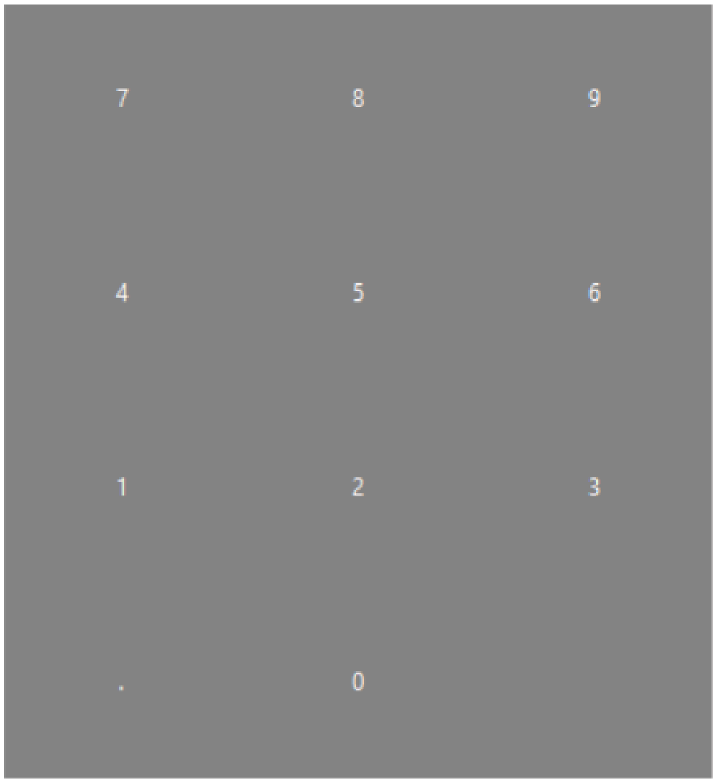
Code:

#convert button which invoked the convert function in order to convert the amount from chosen country to desired country.

convert_button=Button(main_frame,width=48,bd=0,fg='white',bg='#5CEAC3',padx=7,pady=10,text=' Convert',command=convert)

convert_button.pack()

numpad: for user convenience we also designed a numpad which enters the amount according to the user click on the button.



Font: calibri

Frame: numpad_frame

Foreground color: white

Background color: grey(#838383)

Active background color: sea green (#5CEAC3)

Code:

#numpad_frame is to create the numpad with which we can type into the box.

numpad_frame=Frame(main_frame)

```
numpad_frame.configure(background='#838383')

numpad_frame.pack()

create_numpad(numpad_frame,amount)
```

As shown in the code this invokes a function create_numpad to form the structure of the numpad.

```
#'create_numpad' function is used to create the numpad used in tkinter window.

def create_numpad(numpad_frame,amount):

    r=3

    c=0

    button_list=['7','8','9','4','5','6','1','2','3','.','0']

    for button in button_list:#creates the button and assigns the value from the button_list.

        cmd=lambda button=button:amount.insert(END,str(button))

button=Button(numpad_frame,text=button,width=16,command=cmd,bd=0,height=6,bg='#838383',fg='#FFFAFA',activebackground='#5CEAC3',activeforeground='#FFFAFA').grid(row=r,column=c)

        c+=1

        if c>2:

            c=0

            r+=1
```

every button is of same size and we used grid system to make them in proper alignment.

Convert: convert command is used to convert the amount and print it in converted_amount label.

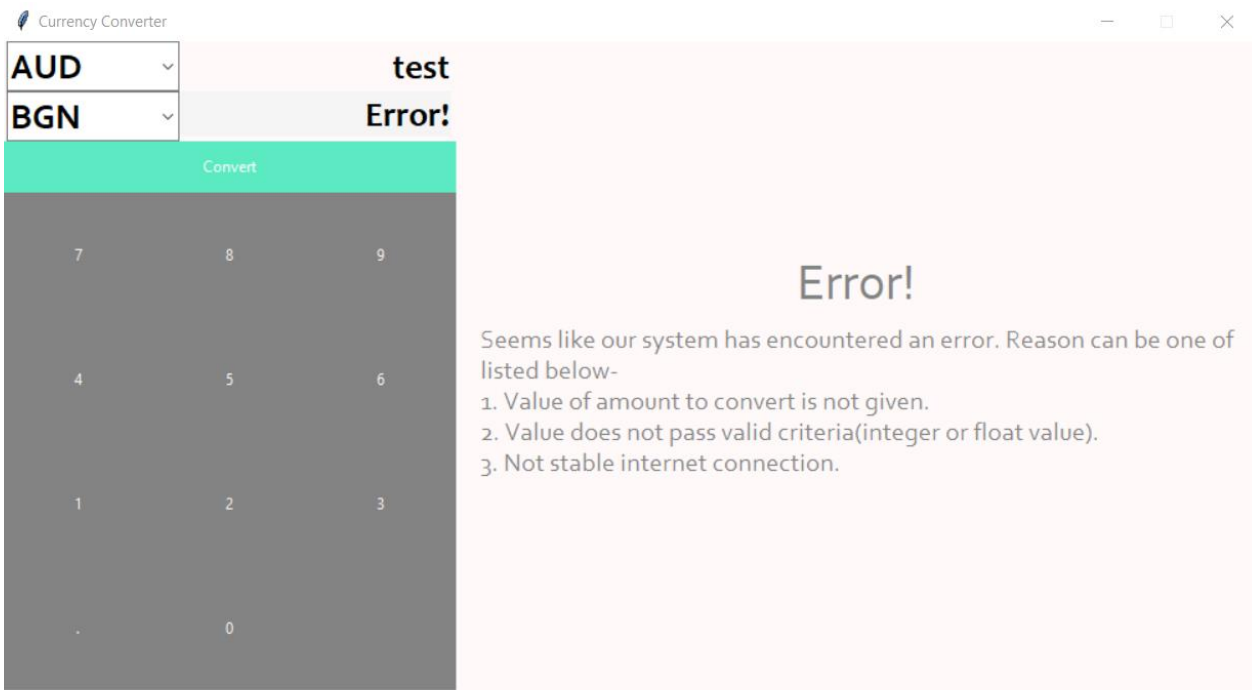
Code:

```
# 'convert' function is used to convert the amount from one currency to another and it also calls graph function.

def convert():

    for widget in graph_frame.winfo_children():#destroying previous graphs.

        widget.destroy()
    from_currency=from_country_name.get()

    to_currency=to_country_name.get()

    currency_conversion =
CurrencyConverter(fallback_on_missing_rate=True)#currency_converter module.

    try:#to identify if there is any input type error.

        amount_to_convert=float(amount.get())

        converted=currency_conversion.convert(amount_to_convert,from_currency,to_currency)

        converted_amount.configure(text=str(round(converted,3)))

        graph(from_currency,to_currency)#calls graph function with 2 arguments.

    except:#displaying error message in case of invalid input.

        converted_amount.configure(text='Error!')


error_title=Label(graph_frame,text='Error!',bg='#FFFAFA',fg='#838383',font=('corbel',30),anchor=
N)

        Error_message='Seems like our system has encountered an error. Reason can be one of
listed below-\n1. Value of amount to convert is not given.\n2. Value does not pass valid
criteria(integer or float value).\n3. Not stable internet connection.'


error_message=Message(graph_frame,text=Error_message,fg='#838383',bg='#FFFAFA',font=('c
orbel',15),padx=18,anchor=W,width=620)

        error_title.pack()

        error_message.pack()
```

Libraries used: currency_converter, tkinter

As a start the convert function clears all the previous data(if any) so that we can write and print new data and plot a new graph without any interruption. Then it gets the base and desired country values from their text variables and also creates a currency_conversion variable which helps us to convert the amount. Then in try module we get the amount from entry widget and convert it and print it in label section. And also invoke the graph function.

As a exception handling mechanism, if we encounter any error during this process we throw the exception and print ‘Error!’ in label and an error message in graph_frame like the image shown below.



Error!

Seems like our system has encountered an error. Reason can be one of listed below-
1. Value of amount to convert is not given.
2. Value does not pass valid criteria(integer or float value).
3. Not stable internet connection.

Graph: graph function is invoked by convert function after converting the amount to print the graph between base and desired countries historical exchange values.

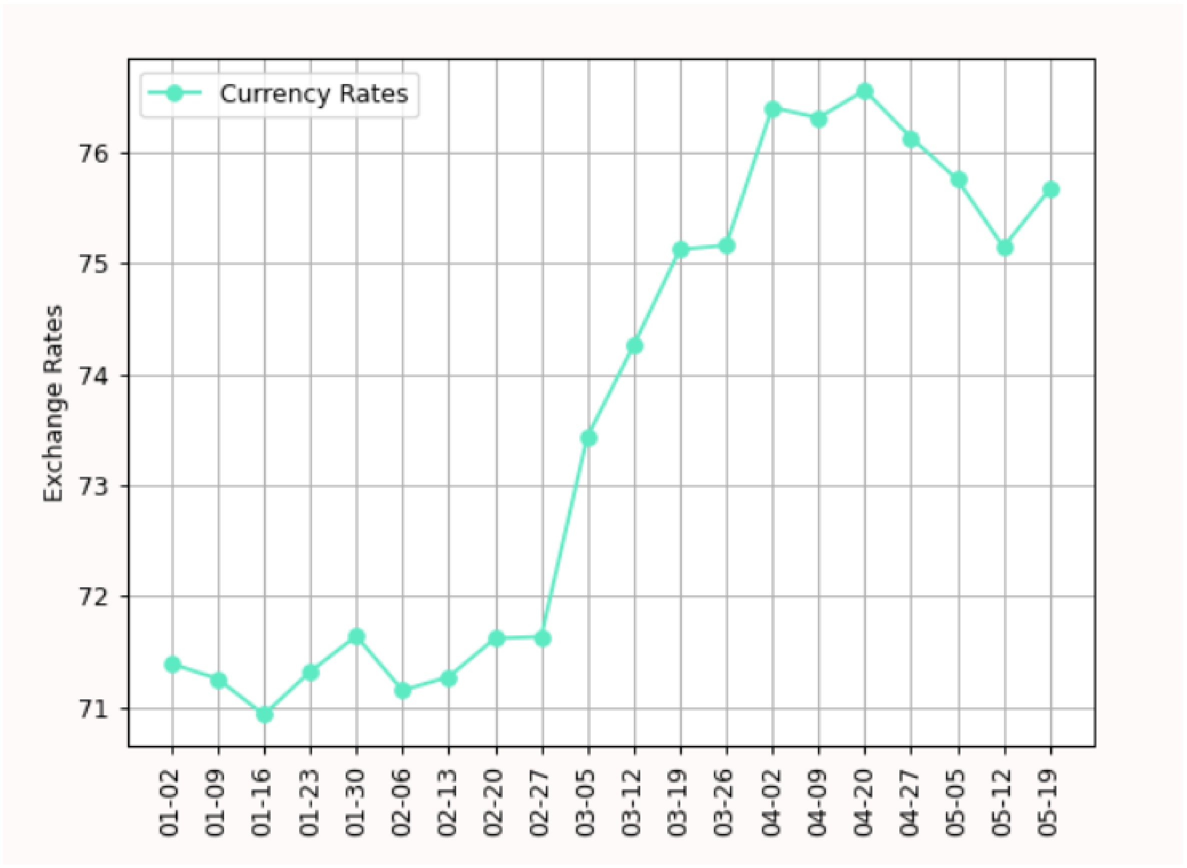
Code:

```
# 'graph' function consists of inbuilt functions from matplotlib, pandas, requests to form a graph for historical exchange rates of the currency chosen by the user.

def graph(from_currency,to_currency):

    url = 'https://api.exchangeratesapi.io/history?start_at=2020-01-01&end_at=2020-05-30&base='+from_currency+'&symbols='+to_currency#url to access the historical rates for desired currency.

    rates = json.loads(requests.get(url).text)

    rates_by_date = rates['rates']

    data = []

    for key, value in rates_by_date.items():#loop to arrange the data gathered from url in date/exchange_rate sequence.

        hist_dict = {'date': key[5:], 'exchange_rate': value[to_currency]}

        data.append(hist_dict)

    data.sort(key = lambda x:x['date'])#sorting the data according to the dates.

    dataframe = DataFrame(data[:100:5])#using pandas to arrange it in data and columns
```

```
x_axis = dataframe['date']

y_axis = dataframe['exchange_rate']

#plotting the graph using matplotlib.

rate_graph = plt.figure(figsize = (7,5), dpi = 89)

rate_graph.patch.set_facecolor('#FFFAFA')

plt.xticks(rotation=90)

plt.xlabel('Date')

plt.ylabel('Exchange Rates')

plt.plot(x_axis,y_axis,marker='o',label='Currency Rates',color='#5CEAC3')

plt.legend()

plt.grid()

#drawing the graph into the canvas in tkinter window.

canvas = FigureCanvasTkAgg(rate_graph, master = graph_frame)

canvas.draw()

get_widz = canvas.get_tk_widget()

get_widz.pack()
```

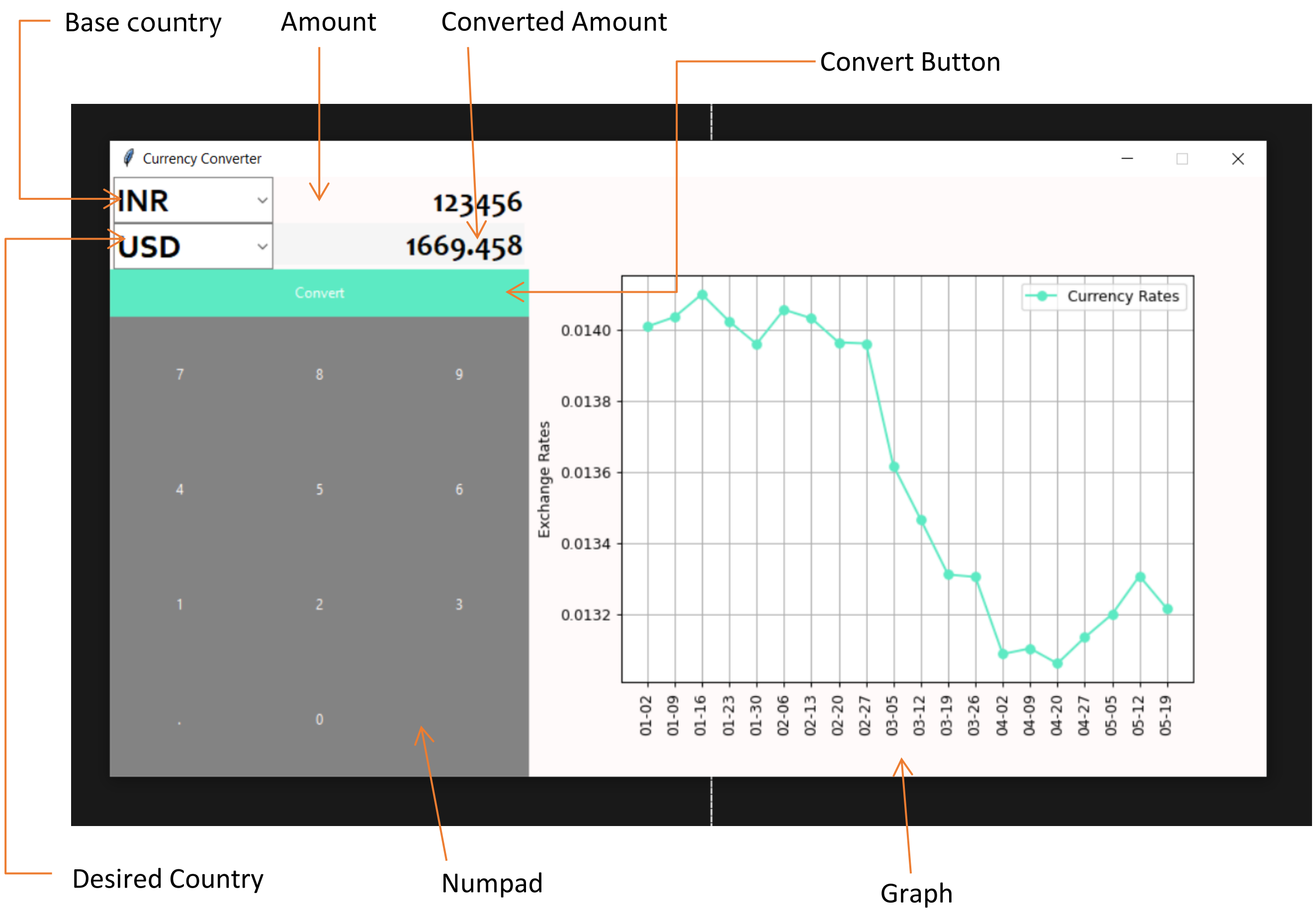
for the starters, the graph modules retrieves data from exchangerateapi url with the help of requests and then converts it into a dictionary with the help of JSON. Then we arrange these by date and sort them so that it is easy to plot a graph.

Then with the help of pandas dataframe we divide them in row and columns. We label our x axis as dates and y axis as rates, designate the size of the graph and color of the background(ice water). The x-axis values are also rotated at the angle of 90. We plot the graph and the line has the sea green (#5CEAC3) color. Then after plotting the graph we use FigureCanvasTkAgg to draw this graph onto the canvas inside the tkinter window and then pack it inside the graph_ frame. The graph looks like the image shown below.



At the end we just disable the resizable functionality of the window and write the mainloop() functions of tkinter to run the tkinter event loop.

Final Appearance



Elements:

1. Base currency
2. Desired currency
3. Amount
4. Converted amount
5. Convert button
6. Numpad
7. Graph

Links

GitHub:

https://github.com/1289-abhi/Currency_converter

Drive:

https://drive.google.com/drive/folders/12lkwoMZtWXZmg2ebc5yQjFVbnRBXd0xZ?usp=sharing