CURRENCY CONVERTER

Group = 03

Prateek Tripathi (11908181)

Sahyogvir Singh (11904949)

Aditya Kumar (11914375)

Date – 03/11/2020

Course Title – Python Programming (INT 213)

Faculty's Name – Gagandeep Kaur

CONTENTS

INTRODUCTION TO PYTHON

•Python is an interpreted, high-level and general-purpose programming language.
•It was created by Guido Van Rossum. It was first released in 1991.
•It is a very easy language and help programmers to write clear and easy logical code for any program.
•Its extensions are .py, .pyi, .pyc, .pyd, .pyo (before version 3.5), .pyw, .pyz(after version 3.5).
•Its latest version is 3.9.0 which was released on 5 October 2020.
•It supports multiple programming paradigms like procedural, object oriented and functional programming.
•It uses dynamic typing and a combination of reference counting and a cycle detecting garbage collector for memory management.
•Unlike other languages Python don't use semicolon to end every line of code and curly braces to delimit blocks.

PROS OF PYTHON

- Python is a **high-level programming language**. This makes it easier to read, write and understand the code.
- Python is an interpreted language which means that Python **directly executes the code** line by line.
- Python doesn’t know the type of variable until we run the code. It automatically assigns the data type during execution. The programmer doesn’t need to worry about declaring variables and their data types.
- Python comes under the **OSI approved** open-source license. This makes it **free to use**. You can download the source code, modify it and even distribute your version of Python.
- The standard library of Python is huge, you can find almost all the functions needed for your task. It has a **Python package manager (pip) which** makes things easier to import other great packages from the **Python package index (PyPi)**.
- In other languages like C/C++, you need to change your code to run the program on different platforms. But in Python you write code once and run it anywhere.

CONS OF PYTHON

- Python is an interpreted language and dynamically-typed language. The line by line execution of code and dynamic nature of Python is responsible for the **slow speed.**
- Python programming language uses a **large amount of memory** which tells that python is not memory efficient.
- Python is not memory efficient and has slow speed so it is not used on the client-side or mobile applications.
- Programming in Python is easy and stress-free. But Python's database access layer is primitive and underdeveloped in comparison to the popular technologies like **JDBC** and **ODBC**.
- As we know Python is a dynamically typed language so the data type of a variable can change anytime.
- A variable containing integer number may hold a string in the future, which can lead to **Runtime Errors**.
- Therefore, Python programmers need to perform testing of the applications.

ABSTRACT IDEA – INTRODUCTION TO PROJECT

Different countries use different currency, and there is daily variation in these currencies relative to one another. Those who transfer money from one country to another (one currency to another) must be updated with the latest currency exchange rates in the market.

This Currency converter project is built keeping this thing in mind. It is simply a calculator-like window developed using the features of Python, Json, API Keys. In this calculator, there is regular update about currency of 52 enlisted countries by which it displays present currency market value and conversion rate.

Such application can be used by any user, but it is mainly useful for business, shares, and finance related areas where money transfer and currency exchange take place on a daily basis.

In this currency converter project, users are provided with an option to select the type of conversion, i.e. from “this” currency to “that” currency. This simple feature allows users to enter amount to be converted (say currency in Dollars), and display the converted amount (say currency in Euro).

MODULES & WIDGETS USED

- ¤REQUESTS
- ¤TKINTER
- ¤PYAUTOGUI
- ¤PYGAME

DISCRIPTION

- **REQUESTS :-** The requests library is used for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application. When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response. There's no need to manually add query strings to your URLs, or to form-encode your POST data.

  We can install it by entering the command terminal of your OS and entering 'pip install requests' and running this.
  Its syntax is 'request.methodname(parameters)'

There are various methods to make requests in python. They are :-

- GET- It is used to retrieve information from the given server using a given URL.
- POST- It requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it.
- PUT- It requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified and if the URL does not point to an existing resource, then the server can create the resource with that URL.
- DELETE- It deletes the specified resource.
- HEAD- It asks for a response identical to that of a GET request, but without the response body.
- PATCH- It is used for modify capabilities. The patch request only needs to contain the changes to the resource, not the complete resource

- **TKINTER** :- Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task.

  It contains of various types of widgets. Some of them are:-

  - ❖ Button- To add a button in your application, this widget is used.
  - ❖ Canvas- It is used to draw pictures and other complex layout like graphics, text and widgets.
  - ❖ CheckButton- To select any number of options by displaying a number of options to a user as toggle buttons.
  - ❖ Entry- It is used to input the single line text entry from the user. For multi-line text input, Text widget is used.

- Frame- It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.
- Label- It refers to the display box where you can put any text or image which can be updated any time as per the code.
- Listbox- It offers a list to the user from which the user can accept any number of options.
- MenuButton- It is a part of top-down menu which stays on the window all the time.
- Menu- It is used to create all kinds of menus used by the application.
- MessageBox- It refers to the multi-line and non-editable text. It works same as that of Label.
- RadioButton- It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.
- Scale- It is used to provide a graphical slider that allows to select any value from that scale.
- Scrollbar- It refers to the slide controller which will be used to implement listed widgets.
- Text- To edit a multi-line text and format the way it has to be displayed.

- TopLevel- This widget is directly controlled by the window manager. It don’t need any parent window to work on.

These all widgets we can use only when the program has tkinter module in it.

- **PY AUTO GUI** :- The PYAUTOGUI library provides cross-platform support for managing mouse and keyboard operations through code to enable automation of tasks.

  We can install it by entering the command terminal of your OS and entering 'pip install pyautogui' and running this.

  It has various functions which we can perform related to mouse, keyboard and some general.

  Mouse Related Functions:

  - moveTo() Function- It moves the mouse cursor on the screen based on the coordinates we provide as parameters.
  - moveRel() Function- It is same as moveTo() function but it moves the mouse position relative to the current mouse position.
  - click() Function- The click() function is used to imitate mouse click operations.
  - scroll() Function- It has two options i.e. scroll up and scroll down. In this we can scroll from a particular point to anywhere we want to scroll.

- position() Function- It is used to know the current position of the mouse on the screen.

Keyboard Related Functions:

- typewrite() Function- It is used to type any text in the text field.
- hotkey() Function- It is used when we want to click two or more keys simultaneously like for copying an item i.e. ctrl+c.
- screenshot() Function- It is used when we want to take screenshot of the screen. It can also save the screenshot on spot in the computer memory.

General Functions:

i. onscreen() Function- It tells whether the point with coordinates x and y exists on the screen.
ii. size() Function- It is used to find the height and width i.e. resolution of the screen.
iii. confirm() Function- It is used to display some information and gives us two options i.e. ok and cancel.
iv. alert() Function- It is used to display some information and to acknowledge that you have read it displays one button i.e. ok.

v. prompt() Function- It is used to request some information from the user and after entering information user has to click ok.

PYGAME :- Game programming is very rewarding nowadays and it can also be used in advertising and as a teaching tool too. Game development includes mathematics, logic, physics, AI, and much more. In python, game programming is done in pygame.

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries also designed to be used with the Python programming language.

We can install it by entering the command terminal of your OS and entering 'pip install pygame' and running this.

Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives.

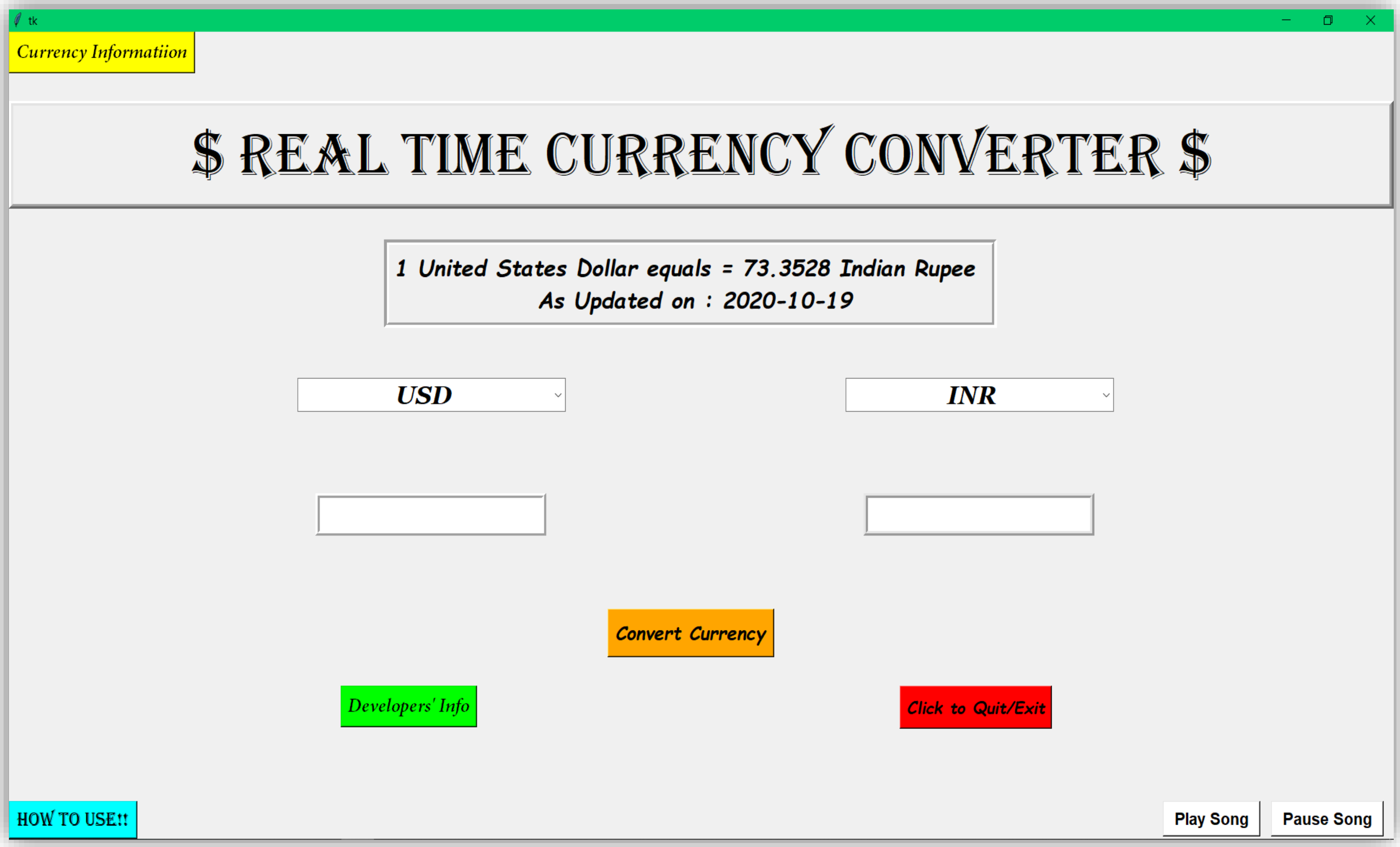
This is based on the assumption that the
most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.
It is initialized in our program using the command pygame.init().

PROJECT CODE

```
# # Python Project on Currency Converter

import requests
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox as mbox
import pyautogui
import pygame
pygame.mixer.init()

#============================================================================================
=============================================================================================
=============================================================================================

class RealTimeCurrencyConverter():
    def __init__(self,url):
            self.data = requests.get(url).json()
            self.currencies = self.data['rates']

#============================================================================================
=============================================================================================
=============================================================================================


    def convert(self, from_currency, to_currency, amount):
        #initial_amount = amount
        if from_currency != 'USD' :
            amount = amount / self.currencies[from_currency]

        # limiting the precision to 4 decimal places
        amount = round(amount * self.currencies[to_currency], 4)
        return amount

#============================================================================================
=============================================================================================
=============================================================================================
class App(tk.Tk):

    def __init__(self, converter):

        tk.Tk.__init__(self)
        self.title = 'Currency Converter'
        self.currency_converter = converter
        self.initUI()
        self.Currency_Full_Form()
        self.Usage_Info()
        self.play_music()
        self.Quit()

#======================================# Converter Label #===================================
=============================================================================================
=============================================================================================

        #self.configure(background = 'blue')
        self.geometry("1920x1080")

        # Label
        self.intro_label = Label(self, text = '$ Real Time Currency Converter $',  fg = 'blac
k', relief = tk.RAISED, borderwidth = 6)
        self.intro_label.config(font = ('algerian',48))#1ST HEADER
        self.intro_label.place(x = 000 , y = 90 , width = 1920 , height = 140) #LOCATION OF 1
ST HEADER

#======================================# Updated Currency #==================================
=============================================================================================
=============================================================================================

        #self.date_label = Label(self, text = f"1 Indian Rupee equals = {self.currency_conver
ter.convert('INR','USD',1)} USD \n Date : {self.currency_converter.data['date']}", relief = t
k.GROOVE, borderwidth = 7)
        self.date_label = Label(self, text = f"1 United States Dollar equals = {self.currency
_converter.convert('USD','INR',1)} Indian Rupee \n As Updated on : {self.currency_converter.d
ata['date']}", relief = tk.GROOVE, borderwidth = 7)
        self.date_label.config(font = ('comic sans ms',18,'bold italic'))
        self.date_label.place(x = 520, y= 270 , width = 850 , height = 115) #LOCATION OF DATE

#==========================================# Entry box #=====================================
=============================================================================================
=============================================================================================

        valid = (self.register(self.restrictNumberOnly), '%d', '%P')
        self.amount_field = Entry(self,bd = 3, relief = tk.RIDGE, justify = tk.CENTER,validat
e='key', validatecommand=valid, borderwidth = 6)
        self.converted_amount_field_label = Label(self, text = '', fg = 'blue', bg = 'white',
 relief = tk.RIDGE, justify = tk.CENTER, width = 17, borderwidth = 6)

#=========================================# dropdown #=========================================
============================================================================================
============================================================================================

        self.from_currency_variable = StringVar(self)
        self.from_currency_variable.set("USD") # default value
        self.to_currency_variable = StringVar(self)
        self.to_currency_variable.set("INR") # default value

#=====================================# Conversion Dropdowns #=================================
============================================================================================
============================================================================================

        font = ("georgia",20, "bold italic")
        self.option_add('*TCombobox*Listbox.font', font)

#==============================================# From #========================================
============================================================================================
============================================================================================

        self.from_currency_dropdown = ttk.Combobox(self, textvariable=self.from_currency_vari
able,values=list(self.currency_converter.currencies.keys()),
         font = font, state = 'readonly', width = 15 , justify = tk.CENTER)
        self.from_currency_dropdown.place(x = 400, y= 450)

#=======================================# From Input Box #=====================================
============================================================================================
============================================================================================

        self.amount_field.place(x = 425, y = 600 , width = 320 , height = 55 )
        self.amount_field.config(font = ('adobe hebrew' , 18 ))

#==============================================# To #==========================================
============================================================================================
============================================================================================

        self.to_currency_dropdown = ttk.Combobox(self, textvariable=self.to_currency_variable
,values=list(self.currency_converter.currencies.keys()),
        font = font, state = 'readonly', width = 15, justify = tk.CENTER)

        self.to_currency_dropdown.place(x = 1160, y= 450)
```

```
#========================================# To Input Box #======================================
===============================================================================================
===============================================================================================

        self.converted_amount_field_label.place(x = 1185, y = 600 , width = 320 , height = 55
 )
        self.converted_amount_field_label.config(font = ('adobe hebrew' , 18 , 'bold' ))

#========================================# Convert button #====================================
===============================================================================================
===============================================================================================

        self.convert_button = Button(self, text = "Convert Currency", fg = "black",bg = 'oran
ge' ,  command = self.perform)
        self.convert_button.config(font=('comic sans ms', 15, 'bold italic'))
        self.convert_button.place(x = 830, y = 750)

#=========================================# Quit Window #======================================
===============================================================================================
===============================================================================================

    def Quit(self):
        quit = Button(self, text="Click to Quit/Exit", fg="black", bg = 'red' , command = sel
f.iExit)
        quit.config(font=('comic sans ms', 13, 'bold italic'))
        quit.place(x = 1235 , y = 850)
        #self.quit.pack(side="bottom")


    def iExit(self):
        pyautogui.moveTo(920 , 650 )
        iexit = mbox.askyesno("Currency Converter","Confirm if you want to Exit")
        if iexit > 0:
            self.destroy()

#=====================================# Background Sound System #=============================
===============================================================================================
===============================================================================================

    def play_music(self):
        pygame.mixer.music.load("ThemeMusic.mp3")

        #Play Button
        play = Button(self , text = " Play Song " , fg = "black" , bg = "white" , command = s
elf.PlaySong )
        play.config(font = ('arial', 13 , 'bold'))
        play.place(x = 1600 , y = 1000)
        #Pause Button
        pause = Button(self , text = " Pause Song " , fg = "black" , bg = "white" , command =
 self.PauseSong )
        pause.config(font = ('arial', 13 , 'bold'))
        pause.place(x = 1750 , y = 1000)
        pygame.mixer.music.play()

    def PauseSong(self):
        pygame.mixer.music.pause()

    def PlaySong(self):
        pygame.mixer.music.unpause()

#=================================# How to USE the Calculator #==============================
============================================================================================
============================================================================================

    def Usage_Info(self):
        usage = Button(self , text = "How To Use!!" , bg = 'cyan' , command = self.UInfo)
        usage.config(font = ('algerian' , 15  ))
        usage.place(x = 0000 , y = 1000)


    def UInfo(self):
        pyautogui.moveTo(1140,820 )
        mbox.showinfo("How To Use!!" , "Usage : \n 1. Always input the amount to be converted
 in the left box. \n\n 2. You are not supposed to feed any data in the right input box , as i
t will not accept. \n Do not waste time on that. \n\n 3. To switch into other currencies ther
e are two dropdowns above each input box. \n  You can use that for the purpose. \n\n 4. The c
urrency is updated every day. \n\n 5. The Currency is updated from the API Key link,  DO NOT
TRY to ALTER that in the SOURCE CODE. \n\n 6. The Calculator works on internet connectivity.
\n \n After Reading \n \n Click OK to Close !!")

#================================# Currency Information #====================================
============================================================================================
============================================================================================

    def Currency_Full_Form(self):
            fullform = Button(self, text="Currency Informatiion",bg = 'yellow', command=self.
fullForm)
            fullform.config(font = ('adobe hebrew',15,'bold italic'))
            fullform.grid(row=1, column=1)
            #fullform.place(x = 100 , y = 100)


    def fullForm(self):
        pyautogui.moveTo(1085,900)

        mbox.showinfo("FullForm"," 1. USD = United States Dollar \n 2. INR = Indian Rupee \n
3. AED = United Arab Emirates Dirham \n 4. ARS = Argentine Peso \n 5. AUD = Australian Dollar
 \n 6. BGN = Bulgarian Lev \n 7. BRL = Brazilian Real \n 8. BSD = Bahamian Dollar \n 9. CAD =
 Canadian Dollar \n 10. CHF = Swiss Franc Currency \n 11. CLP = Chilean Peso \n 12. CNY = Chi
nese Yuan \n 13. COP = Colombian Peso \n 14. CZK = koruna Of Czech Republic \n 15. DKK = Dani
sh Krone , Denmark \n 16. DOP = Dominican Peso \n 17. EGP = Egyptian Pound \n 18. EUR = Euro
\n 19. FJD = Fiji Dollar \n 20. GBP = British Pound Sterling \n 21. GTQ = Guatemalan Quetzal
\n 22. HKD = Hong Kong Dollar \n 23. HRK = kuna , Croatia \n 24. HUF = Hungarian Forint \n 25
. IDR = Indonesian Rupiah \n 26. ILS = Israeli New Shekel \n \n \nClick OK")
        pyautogui.moveTo(1085,900)

        mbox.showinfo("FullForm", " 27. ISK = króna , Iceland \n 28. JPY = Japanese Yen \n 29
. KRW = Korean Republic Won \n 30. KZT = Tenge , Kzakhastan \n 31. MVR = Maldivian Rufiyaa \n
 32. MXN = Mexican Peso \n 33. MYR = Malaysian Ringgit \n 34. NOK = Norwegian Krone \n 35. NZ
D = New Zealand Dollar \n 36. PAB = Panamanian Balboa \n 37. PEN = Peruvian Sol , Peru \n 38.
 PHP = Philippine Peso \n 39. PKR = Pakistani Rupee \n 40. PLN = Polish Zloty \n 41. PYG = Pa
raguayan Guarani , Pragua \n 42. RON = Romanian Leu \n 43. RUB = Russian Ruble \n 44. SAR = S
audi Riyal \n 45. SEK = Swedish Krona \n 46. SGD = Singapore Dollar \n 47. THB = Thai Baht \n
 48. TRY = Turkish Lira \n 49. TWD = New Taiwan Dollar \n 50. UAH = Ukrainian Hryvnia \n 51.
UYU = Uruguayan Peso \n 52. ZAR = South African Rand \n \n \nClick OK To Exit !!")
        pyautogui.moveTo(1085,900)

#======================================# Developers Information #============================
=============================================================================================
=============================================================================================

    def initUI(self):
            inform = Button(self, text="Developers' Info",bg = 'lime', command=self.onInfo)
            inform.config(font = ('adobe hebrew',15,'bold italic'))
            #inform.grid(row=1, column=1)
            inform.place(x = 460 , y = 850)


    def onInfo(self):

        pyautogui.moveTo(1000,650)

        mbox.showinfo("Information","Project Group = 3 \n \nClick OK")
        pyautogui.moveTo(1020,695)

        mbox.showinfo("Information", "Name = Prateek Tripathi \n \nRegdn = 11908181 \n \nRoll
 No. = RK19RHB56 \n \nClick OK!!")
        pyautogui.moveTo(1020,695)
        mbox.showinfo("Information", "Name = Sahyogvir Singh \n \nRegdn = 11904949 \n \nRoll No. = RK19RHB45 \n \nClick OK!!")
        pyautogui.moveTo(1020,695)

        mbox.showinfo("Information", "Name = Aditya Yadav \n \nRegdn = 11914375 \n \nRoll No. = RK19RHB71 \n \nClick OK!!")
        pyautogui.moveTo(1020,695)

#=======================================# Calculations #=====================================
=============================================================================================
=============================================================================================

    def perform(self):
        amount = float(self.amount_field.get())
        from_curr = self.from_currency_variable.get()
        to_curr = self.to_currency_variable.get()

        converted_amount = self.currency_converter.convert(from_curr,to_curr,amount)
        converted_amount = round(converted_amount, 3)

        self.converted_amount_field_label.config(text = str(converted_amount))

    def restrictNumberOnly(self, action, string):
        regex = re.compile(r"[0-9,]*?(\.)?[0-9,]*$")
        result = regex.match(string)
        return (string == "" or (string.count('.') <= 1 and result is not None))

if __name__ == '__main__':
    url = 'https://api.exchangerate-api.com/v4/latest/USD'
    converter = RealTimeCurrencyConverter(url)

    App(converter)
    mainloop()
```

SNAP-SHOT OF PROJECT


tk
Currency Informatiion
$ REAL TIME CURRENCY CONVERTER $
1 United States Dollar equals = 73.3528 Indian Rupee
As Updated on : 2020-10-19
USD
INR
Convert Currency
Developers' Info
Click to Quit/Exit
HOW TO USE!!
Play Song
Pause Song

BIBLIOGRAPHY

- WIKIPEDIA
- GEEKS FOR GEEKS
- https://www.python.org/
- YOUTUBE
- Assigned Modules :-
  - Prateek:-
    - Tkinter
    - Requests
  - Sahyogvir :-
    - Pygame
  - Aditya :-
    - Pyautogui