CURRENCY CONVERTER

ITW I Group Project

AUTHORS

Undertaken by- BT19CSE002-Jatin Chhangani, BT19CSE013-Yashashree Ganthale, BT19CSE046-Prabhu Satyam, BT19CSE053-Nandini Kapoor

Course Title

IT WORKSHOP I

Index:

Introduction:

What is a currency converter?

The Currency Converter enables the user to convert an amount of money in the currency of one's choice and to its corresponding value in another desired currency.

Instructions?

The 'ITW Project' window enables the user to enter amount in the 'Amount' field, select a currency from a drop-down menu under the field 'From Currency' and from under 'To Currency'. On pressing the button 'Convert' the desired value is shown in the field 'Converted Amount'. To clear all values and enter new ones, the 'Clear' button must be used.

Modules and methods introduction.

1. *tkinter:*

Tkinter is the standard GUI library for Python. Python with Tkinter is the fastest and easiest way to create GUI applications. Tkinter provides an object-oriented interface to the Tk GUI toolkit that is built-in to Python. Tkinter is a set of wrappers that implement the Tk widgets as Python classes. Tkinter makes it simple to create a GUI which handles user input and output. A GUI uses a form of object-oriented programming called "event-driven." This means that the program responds to events, which are actions that a user takes.

To import Tkinter module :

import tkinter

However, to import every exposed object in Tkinter into your current namespace:

from tkinter import *

To initialize tkinter, we have to create a Tk root widget, which is a window with a title bar and other decorations provided by the window manager. The root widget has to be created before any other widgets and there can only be one root widget.

root = Tk()

Widget in tkinter-

tkinter provides us with a variety of common GUI elements called widgets which we can use to build out interfaces – such as buttons, menus, and various kinds of entry fields and display areas. Each separate widget is a Python object, instances of classes that represent buttons, frames, and so on. When creating a widget, you must pass its parent as a parameter to the widget creation function(except the root widget).

The following widgets are used in code-

- **Initializing a variable using the variable construtor:**

General syntax: var_name= Tkinter_variable_type(master, value = any_value)

Example: variable1 = StringVar(root)

(To intialize a string variable in the root with the name variable1)

- **Setting the value to a variable:**
  General syntax: var_name.set(value)

  Example: variable1.set("currency")

  (To set the value of the string variable as “currency”)

- **Entry widget with get method:**
  The Entry Widget is used to Enter or display a single line of text and with the get method, it returns the entry’s current text as a string from the text entry box.

  General syntax: entry_name=Entry(parent,options)

  Example: amount = float(Amount1_field.get())
  (To set the value of the amount to the string returned from the Amount1_field text entry field as a float value)

- **Entry widget with insert method:**
  The Entry Widget is used to Enter or display a single line of text and with the insert method, it inserts a given string before the character at the given index.

  General syntax: entry_name=Entry(parent,options)

  Example: myentry.insert(0, "a default value")
  (To insert the string "a default value” at the 0th index of the entry “myentry”)

- **Entry widget with delete method:**
  The Entry Widget is used to Enter or display a single line of text and with the insert method, it delete the data currently within the Entry box from one index to another index.

  General syntax: entry_name=Entry(parent,options)

  Example: myentry.delete(1,3)
  (To delete the character from index 1 to index 3 of the entry “myentry”)

- **Configuring the widget:**
  configure() is used to access and change an object's attributes (widget’s options) after its initialization.

  General syntax: widget_name.cofigure(option=value)

  Example: root.configure(background = 'orange')

  (to change the value of the background option to orange of the root widget)

- **Geometry method of the widget:**
  This method is used to set the dimensions of the tkinter window and is used to set the position of the main window on the user's desktop.

  General syntax: widget_name.geometry(dimensions)

  Example: root.geometry("450x250")

  (to set the dimension of root widget to 450x250 pixels)

- **Title method of the widget:**
  This method is used to set the title of the tkinter window

  General syntax: widget_name.title("title")

  Example: root.title("IT Workshop 1 Group Project")

  (to set the title of root widget to "IT Workshop 1 Group Project")

- **Using the label widget with options text, fg and bg:**
  Tkinter Label is a widget that is used to implement display boxes where you can place text or images. Option text to display the required text, option bg is used to set the normal background color displayed behind the label and indicator ,and option fg to specify the color of the text.

  General syntax: label_name = Label(master, text = " ", fg = ' ', bg = ' ')

  Example: mylabel = Label(root, text = "Amount :", fg = 'black', bg = 'dark green')

  (To sets the values of options in label "mylabel", option text to "Amount :" ,option fg to 'black' ,option bg to 'dark green')

- **Using a grid widget:**
  The grid geometry manager puts the widgets in a 2-dimensional table. The master widget is split into a number of rows and columns, and each "cell" in the resulting table can hold a widget and the **grid** method to tell the manager in which row and column to place them.

  General syntax: widget_name.grid(option=value)

  Example: mylabel.grid(row = 2, column = 1)

  (to place the "mylabel" widget into the 2nd row and 1st column of the grid)

- **Entry widget:**
  The entry widget is a standard Tkinter widget used to enter or display a single line of text.

  General syntax: entry_name = Entry(master, option=value)

  Example: Amount1_field = Entry(root)

(to create an entry widget with the name “Amount1_field” in the root master window)

- **Using a grid widget with ipadx,ipady options:**
  ipady defines how many pixels to pad widget vertically inside the widget's borders. While ipadx defines how many pixels to pad widget horizontally inside the widget's borders.

  General syntax: widget_name.grid(row , column , ipadx , ipady)

  Example: FromCurrency_option.grid(row = 2, column = 1, ipadx = 10, ipady=20)

  (to place the “FromCurrency_option” widget into the 2nd row and 1st column of the grid with 10 pixels of padding horizontally and 20 pixels of padding vertically)

- **Optionmenu widget:**
  The **optionmenu** class is a helper class that creates a popup menu, and a button to display it.

  General syntax: menu_name = OptionMenu(master, variable, option=value)

  Example: myoptionmenu = OptionMenu(master, myvar, "one", "two", "three")

  (to create a popup menu for variable “myvar” with options "one", "two", "three")

- **Button widget with text, bg, command, and fg options:**
  The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button. Options text to display the required text, option bg is used to set the normal background color displayed behind the label and indicator, option fg to specify the color of the text, and option command to execute a function or method to be called when the button is clicked.

  General syntax: button_name = Button(master, option=value)

  Example: mybutton = Button(root, text = "Clear", bg = "black", fg = "white", command = clear_all)

  (To sets the values of options in button “mybutton” in root master window, option text to "Clear" , option bg to 'black' , option fg to 'dark green', and option command to call the function “clear_all”)

2. *requests:*

Requests is a Python module that you can use to send all kinds of HTTP requests. It is an easy-to-use library with a lot of features ranging from passing parameters in URLs to sending custom headers and SSL Verification. Requests allows to send HTTP/1.1 requests. Headers, form data, multi-part files, and parameters with simple Python dictionaries, and access the response data in the same way.

To import Request module:

import requests

Methods used:

1. get(): The GET method is used to retrieve information from the given server using a given URI. The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the '?' character. Python's requests module provides in-built method called **get()** for making a GET request to a specified URI.

   Syntax: requests.get(url, params={key: value}, args)
   Parameters:
   url: Required. The url of the request.
   params: If the construction of the URL by hand, thebn the data would be given as key/value pairs in the URL after a question mark
   Return Value: The HTTP request returns a Response Object with all the response data.

2. json(): r**esponse.json()** returns a JSON object of the result (if the result was written in JSON format, if not it raises an error). Python requests are generally used to fetch the content from a particular resource URI. Whenever we make a request to a specified URI through Python, it returns a response object. Now, this response object would be used to access certain features such as content, headers, etc.
   In case the JSON decoding fails, r.json() raises an exception. For example, if the response gets a 204 (No Content), or if the response contains invalid JSON, attempting r.json() raises ValueError: No JSON object could be decoded.

3. *Pandas:*

   Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics. Pandas deals with the following three data structures –Series, Dataframe, Panel.

   To import pandas:

   1. import pandas as pd.
   2. Used Pandas class to convert the json object in DataFrame.

   DataFrame: **Pandas DataFrame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the **data**, **rows**, and **columns**.

   Syntax: pd.DataFrame(data, index, columns, dtype, copy).

   data: data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.
   index: For the row labels, the Index to be used for the resulting frame is Optional Default np.arange(n) if no index is passed.
   columns**:** For column labels, the optional default syntax is - np.arange(n). This is only true if no index is passed.
   dtype: Data type of each column.
   copy**:** This command (or whatever it is) is used for copying of data, if the default is False.

Working of the Code:

Link for google drive folder containing code – notebook and python file both uploaded.

NOTE- Terminology for convenience:

1. currency in which the amount is given by the user-initial currency.
2. Currency to which the user desires to convert given amount-final currency.

Importing tkinter library as object tk, and pandas as pd. Requests has limited use thus its methods are accessed directly using the module name.

```
import requests

import tkinter as tk

import pandas as pd
```

The class ‘Converter’ is defined also enabling a URL to be passed to the constructor in the variable url in the form of a string. The instances of this class would be containing a data frame ‘data’ (json object is converted to df). The method ‘requests’ returns a response object, contains the server’s response to the request and using .json() we convert this into a json object (the python equivalent of json object is dictionary). Then we use DataFrame method in pandas to convert this dictionary to a data frame.

Below is an image of what the data frame looks like being obtained from the json file:

	base	date	time_last_updated	rates
AED	USD	2020-11-19	1605744244	3.672028
ARS	USD	2020-11-19	1605744244	79.984359
AUD	USD	2020-11-19	1605744244	1.367604
BGN	USD	2020-11-19	1605744244	1.647006
BRL	USD	2020-11-19	1605744244	5.316588
BSD	USD	2020-11-19	1605744244	1.000000
CAD	USD	2020-11-19	1605744244	1.307610
CHF	USD	2020-11-19	1605744244	0.910835
CLP	USD	2020-11-19	1605744244	765.022095
CNY	USD	2020-11-19	1605744244	6.556348
COP	USD	2020-11-19	1605744244	3652.619048
CZK	USD	2020-11-19	1605744244	22.259844
DKK	USD	2020-11-19	1605744244	6.275367
DOP	USD	2020-11-19	1605744244	57.199851
EGP	USD	2020-11-19	1605744244	15.580947
EUR	USD	2020-11-19	1605744244	0.842705
FJD	USD	2020-11-19	1605744244	2.097828
GBP	USD	2020-11-19	1605744244	0.753288


```
class Converter:

    def __init__(self, url):

        self.data = pd.DataFrame(requests.get(url).json())

        print('object: ', self.data['rates'])

        self.currencies = self.data['rates']

        self.indices = self.data.index.tolist()
```

‘data’ contains the market exchange rates under the column label ‘rate’. We call for the column corresponding to ‘rate’ and store it in the class data-member ‘currencies’ which now contains the series with same indices as the row labels of ‘data’ and containing the rates of market exchange of all currencies with respect to USD.

Below is an image of the series ‘currencies’ printed along with its object type i.e. pandas series.

```
<class 'pandas.core.series.Series'>
AED       3.672028
ARS      79.984359
AUD       1.367604
BGN       1.647006
BRL       5.316588
BSD       1.000000
CAD       1.307610
CHF       0.910835
CLP     765.022095
CNY       6.556348
COP    3652.619048
CZK      22.259844
DKK       6.275367
DOP      57.199851
EGP      15.580947
EUR       0.842705
FJD       2.097828
GBP       0.753288
GTQ       7.770743
HKD       7.752421
HRK       6.371827
```

The acronyms are made into a list of strings using the .index method on data frame ‘data’ to access its row labels, and tolist method to convert this returned object to list. This converted list is finally stored in data member ‘indices’ of the class.

Below is the result of printing the returned object and its type of ‘self.data.index’ and then ‘self.indices’.

```
before converting to list <class 'pandas.core.indexes.base.Index'>
Index(['AED', 'ARS', 'AUD', 'BGN', 'BRL', 'BSD', 'CAD', 'CHF', 'CLP', 'CNY',
       'COP', 'CZK', 'DKK', 'DOP', 'EGP', 'EUR', 'FJD', 'GBP', 'GTQ', 'HKD',
       'HRK', 'HUF', 'IDR', 'ILS', 'INR', 'ISK', 'JPY', 'KRW', 'KZT', 'MVR',
       'MXN', 'MYR', 'NOK', 'NZD', 'PAB', 'PEN', 'PHP', 'PKR', 'PLN', 'PYG',
       'RON', 'RUB', 'SAR', 'SEK', 'SGD', 'THB', 'TRY', 'TWD', 'UAH', 'USD',
       'UYU', 'ZAR'],
      dtype='object')
after converting to list:  <class 'list'>
['AED', 'ARS', 'AUD', 'BGN', 'BRL', 'BSD', 'CAD', 'CHF', 'CLP', 'CNY', 'COP', 'CZK', 'DKK', 'DOP', 'EGP', 'EUR', 'FJD', 'GBP',
'GTQ', 'HKD', 'HRK', 'HUF', 'IDR', 'ILS', 'INR', 'ISK', 'JPY', 'KRW', 'KZT', 'MVR', 'MXN', 'MYR', 'NOK', 'NZD', 'PAB', 'PEN',
'PHP', 'PKR', 'PLN', 'PYG', 'RON', 'RUB', 'SAR', 'SEK', 'SGD', 'THB', 'TRY', 'TWD', 'UAH', 'USD', 'UYU', 'ZAR']
```

The member function ‘Convert’ of this class contains parameters ‘fromC’-holding the initial currency, ‘toC’-holding the final currency and ‘amount’-holding the amount to be converted. It returns the final converted amount.

```
def Convert(self, fromC, toC, amount):
    intialamount = amount
    if(fromC != 'USD'):
        intialamount = intialamount/self.currencies[fromC]
    amount = round(intialamount*self.currencies[toC], 2)
    return amount
```

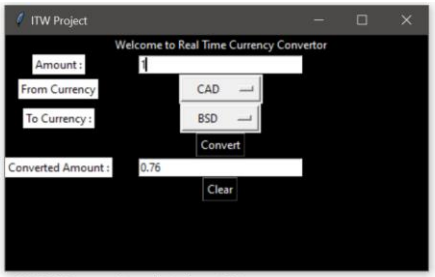
The rates are given with respect to USD and thus in case the initial is USD, it needn’t be converted. Whereas if the amount is in any currency apart from USD, it first needs to be converted to USD. Once we have the amount in USD, the available series ‘currencies’ containing rates of exchange can be used for the final conversion. As we know the final currency, we can access its rate of exchange with respect to USD by accessing the label i.e. acronym of currency corresponding to the final currency.

```
root = tk.Tk()
converter = Converter('https://api.exchangerate-api.com/v4/latest/USD')
variable1 = StringVar(root, "currency")
variable2 = StringVar(root, "currency")
```

An object ‘converter’ of type ‘Converter’ is created and thus we now obtain the series containing rates of exchange in the variable ‘converter.currencies’.

Set the strings of 2 tkinter StringVar objects to “currency”, these variables will further be used to provide default text to the currency selection buttons. Note that StringVar object is not the same as string, although using get method for an object of type StringVar returns a string. The below images depicting this.

```
PHP       48.245346
PKR      158.154639
PLN        3.768991
PYG     6973.181818
RON        4.104115
RUB       75.924033
SAR        3.750239
SEK        8.602934
SGD        1.341255
THB       30.286307
TRY        7.707329
TWD       28.513582
UAH       28.100035
USD        1.000000
UYU       42.590228
ZAR       15.406122
Name: rates, dtype: float64
<class 'tkinter.StringVar'> : PY_VAR94 ,  <class 'str'> : CAD
```



These type and value pairs in the image to the left correspond to the highlighted line in the below image of part of the code.

The highlighted line in the image to the left corresponds to the output in the above image.

```
def RealTimeCurrencyConversion():
    #from_currency = variable1.get()
    from_currency = variable1.get()
    print(type(variable1), ":", variable1, ", ",type(from_currency), ":" , from_currency)
    to_currency = variable2.get()
    amount = float(Amount1_field.get())
    new_amount = converter.Convert(from_currency, to_currency, amount)
    #print('You converted from ', from_currency)
    #print('to ', to_currency)
    Amount2_field.insert(0, str(new_amount))
```

In the function ‘RealTimeCurrencyConversion’, assign string values taken from user with objects variable1 and variable2 through get method of StringVar, to the variables ‘from_currency’ and ‘to_currency’.

‘Amount1_field’ (as we will see in main) is an Entry widget thus on using get method, it would assign the line in its field to variable ‘amount’. Get method returns string, thus before assigning we typecast the input to float.

```
def RealTimeCurrencyConversion():
    from_currency = variable1.get()
    to_currency = variable2.get()
    amount = float(Amount1_field.get())
    new_amount = converter.Convert(from_currency, to_currency, amount)
    print('You converted from ', from_currency)
    print('to ', to_currency)
    Amount2_field.insert(0, str(new_amount))
```

‘Amount2_field’ (as we will see in main) is another Entry widget thus on using insert method would print the second parameter given to it, namely new_amount type-casted to string in the textbox before the character at index 0 in the text (i.e. text already present in the box).

This justifies the fact that the amount printed in the converted amount text box will give you output in the format of xxxx.yyyy.zzzz… (xxxx being the latest conversion) if the application is used multiple times without hitting clear.

```
def clear_all() :
```

```
    Amount1_field.delete(0, END)
    Amount2_field.delete(0, END)
```

‘clear_all’ function clears the textboxes in the window of the application making them ready for newer entries.

The main function contains initializations of the entry widgets, these are accessed in functions later without being passed as parameters, this is because main rather than being an actual function is identified with the help of an if statement thus enabling access to its variables in functions run after it.

```
if __name__ == "__main__":
    root.configure(background = 'black')
    root.geometry("450x250")
    root.title("ITW Project")
```

Tkinter objects attribute are not handled through python attribute mechanism. Instead, tkinter provide two ways to alter this attribute, them being- using objects as a dictionary and using configure method. In the code sample, configure has been used. Size of the window and title respectively, of the application, have been asserted using geometry and title methods respectively.

```
    headlabel = Label(root, text = 'Welcome to Real Time Currency Convertor', fg = 'white', bg = "black")
    label1 = Label(root, text = "Amount :", fg = 'black', bg = 'white')
    label2 = Label(root, text = "From Currency", fg = 'black', bg = 'white')
    label3 = Label(root, text = "To Currency :",fg = 'black', bg = 'white')
    label4 = Label(root, text = "Converted Amount :", fg = 'black', bg = 'white')
```

Widget ‘Label’ implements the display boxes.

```
    headlabel.grid(row = 0, column = 1)
    label1.grid(row = 1, column = 0)
    label2.grid(row = 2, column = 0)
    label3.grid(row = 3, column = 0)
    label4.grid(row = 5, column = 0)
```

Tabular positioning of the text boxes in the window is enabled by the use of grid method on label1, label2, label3, label4 widgets.

```
    Amount1_field = Entry(root)
    Amount2_field = Entry(root)


    Amount1_field.grid(row = 1, column = 1, ipadx ="25")
    Amount2_field.grid(row = 5, column = 1, ipadx ="25")
```

Amount1_field and Amount2_field are entry widgets which have been used for taking in value through textboxes, namely insert and delete methods as seen earlier in the. Ipadx and ipady options are used for padding inside the widget's borders. Here it provides a 25-pixel padding horizontally.

```
CurrenyCode_list = converter.indices

print()

print('List of currencies: ', CurrenyCode_list, '\n')
```

The list of labels i.e. currency acronyms earlier stored in indices data member of the class 'Converter' are stored in a list named 'CurrencyCode_list'.

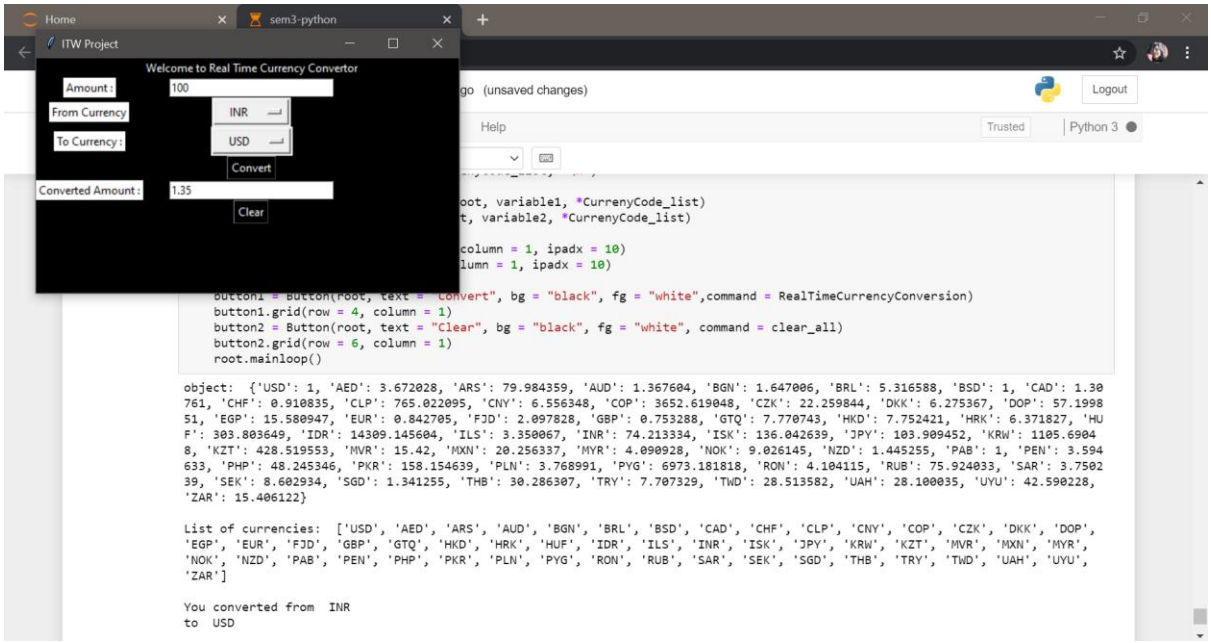
```
FromCurrency_option = OptionMenu(root, variable1, *CurrenyCode_list)

ToCurrency_option = OptionMenu(root, variable2, *CurrenyCode_list)



FromCurrency_option.grid(row = 2, column = 1, ipadx = 10)

ToCurrency_option.grid(row = 3, column = 1, ipadx = 10)



button1 = Button(root, text = "Convert", bg = "black", fg = "white", command =
RealTimeCurrencyConversion)



button1.grid(row = 4, column = 1)

button2 = Button(root, text = "Clear", bg = "black", fg = "white", command = clear_all)

button2.grid(row = 6, column = 1)



root.mainloop()
```

The OptionMenu helper class creates a pop-up menu of the elements of 'CurrencyCode_list' and an associated button to display it. The button will display the default value of the string corresponding to StringVar object passed, in this case variable1 and variable2, which were previously set to strings "currency".

The conversion and clear buttons are given the functions they need to execute on click as command parameter. Mainloop is a method on the main window, in this case the root, which is called so as to run the application finally.

Input / Output:

Here is what the window of the application would look like on running the code.



The amount to be converted given through text box, initial and final currencies given using dropdown menus.

The output i.e. the final converted amount is also displayed through a textbox.

User interactions include clearing the textboxes and telling the application when to convert.

Conclusion:

A currency converter was created utilizing a data frame obtained from json file (fetched through an exchange-rates API) and methods of tkinter library in python.

References:

https://api.exchangerate-api.com/v4/latest/USD

https://www.tutorialspoint.com/python/tk_button.htm

https://stackoverflow.com/questions/52249092/how-to-set-variable-in-tkinter-optionmenu

https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html

https://www.tutorialspoint.com/python/tk_grid.htm

https://stackoverflow.com/questions/8707039/difference-between-pack-and-configure-for-widgets-in-tkinter

https://www.tutorialspoint.com/python/tk_label.htm

https://realpython.com/python-main-function/

https://www.geeksforgeeks.org/python-tkinter-entry-widget/

https://stackoverflow.com/questions/32756394/python-access-main-program-variable-from-outside

https://www.geeksforgeeks.org/python-setting-and-retrieving-values-of-tkinter-variable/

https://www.geeksforgeeks.org/python-pandas-apply/

https://effbot.org/tkinterbook/optionmenu.htm

https://www.geeksforgeeks.org/response-json-python-requests/

https://requests.readthedocs.io/en/master/api/#requests.Response

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Index.html

https://www.geeksforgeeks.org/python-setting-and-retrieving-values-of-tkinter-variable/

https://www.edureka.co/blog/python-requests-tutorial/

https://requests.readthedocs.io/en/master/user/quickstart/#json-response-content

https://www.geeksforgeeks.org/response-json-python-requests/?ref=lbp

Thank you.