Language Translator - PythonGeeks

#### **About Language translator:**

With so many languages spoken and used around the world, it is hard to learn all languages to survive in a new land. Thus a translator is handy where it converts the required information to the target language. This makes our work easy because we just need an app to support us without needing to know the target language.

#### **Python Language Translator:**

The objective of this project is to create a simple script to read user text and convert it to the desired language.

#### **Project Prerequisites:**

The project uses only two libraries, **googletrans and tkinter**. To install googletrans through pip, use the command: pip install googletrans==3.1.0a0.

The alpha version is recommended to avoid errors when translating.

Tkinter is a built-in GUI library in python, hence import to check if it is present

python

>>> import tkinter

This command should not display an error, if it is present. In case the module is missing, install it on linux systems using: sudo apt-get install python3-tkinter and for windows users, reinstall python.

#### **Download Language Translator code:**

You can download the source code for the language translator in the given link: [Language Translator Python Code](https://drive.google.com/file/d/1m4kN6-3_oG5YYjqSNb3DJovzaFiVWyfN/view?usp=sharing)

#### **Project File Structure:**

Let’s have a look at the steps to build the project:

1. Import necessary modules: googletrans and tkinter
2. Define translate\_function
3. Define a function to clear the text widgets
4. Define the GUI and invoke the Translator() class
5. Create the GUI components and read input

Let us look at the implementation in detail.

1. **Import necessary module**

#PythonGeeks GUIDE TO TRANSLATE LANGUAGES USING Google Translate API

import googletrans

from googletrans import Translator

from tkinter import \*

from tkinter import messagebox

**Code Explanation:**

* **Import googletrans:** This module contains the Translator class to read text and convert it to the desired language. Googletrans supports 106 languages.
* **from googletrans import Translator:** Importing the Translator() class to use the translate function.
* **from tkinter import messagebox:** Messagebox displays a prompt or a dialog box based on conditions set. Showerrors, askyesorno, showinfo are some messagebox prompts. We will use showerrors and showinfo in this project
* **From tkinter import \*:**  Tkinter is used to create the GUI of the timer. It contains widgets to take user input and buttons to perform a certain function. Tkinter also supports mouse movements and clicks.

1. **Define translate\_function:**

#Since default options are allowed, we check for

#explicitly given source and destination languages

def translate\_function():

#check if the source and target languages are empty

if (len(src\_entry.get("1.0","end-1c"))>1):

src\_v = src\_entry.get("1.0","end-1c").lower()

src\_v =src\_v.replace(" ","")

else:

src\_v = None

if (len(dest\_entry.get("1.0","end-1c"))>1):

dest\_v = dest\_entry.get("1.0","end-1c").lower()

dest\_v =dest\_v.replace(" ","")

else:

dest\_v = None

#Check if the text is empty. If so, prompt user to key it

if (len(text\_entry.get("1.0","end-1c"))<=1):

messagebox.showerror(message="Enter valid text")

else:

#Send the parameters based on user input provided

text\_v = text\_entry.get("1.0","end-1c")

if (not src\_v) & (not dest\_v):

translated\_text = translator\_object.translate(text\_v)

elif (not src\_v):

translated\_text = translator\_object.translate(text\_v,dest=dest\_v)

elif (not dest\_v):

translated\_text = translator\_object.translate(text\_v,src=src\_v)

else:

translated\_text = translator\_object.translate(text\_v,src=src\_v,dest=dest\_v)

#Display translated text on a prompt

messagebox.showinfo(message = "TRANSLATED TEXT: "+ translated\_text.text)

**Code explanation:**

* **def translate\_function():** Declare the function to translate the input text
* **(len(src\_entry.get("1.0","end-1c"))>1:**  To check if the user enters a text in the source language text box, we check the length. The length for an empty widget that contains no string is 1. Extract the contents of a textbox widget using get(), where ‘1.0’ denotes the first index ie. 0 of the string and end-1c denotes the end of string. If the length of the string is greater than 1, we read the string and convert it to lowercase using **string.lower() function** and remove spaces using **replace()** and set it to the variable src\_v. If the length is 1, src\_v variable is set to None. Similarly extend the same for the target language. Src\_v set to None autodetect the source language and dest\_v set to None translates to the default target language, English
* **if (len(text\_entry.get("1.0","end-1c"))<=1):** If the text box for text input is empty, we simply cannot translate the text because there is none. Thus we check the length and if it is less than or equal to 1, we raise a prompt asking the user to enter the text using **messagebox.showerror(message="Enter valid text")**
* **Test conditions:** The translator\_object.translate() functions take 1 mandatory parameter and 2 optional parameters. Text is the mandatory parameter and source and target languages (src and dest) are the optional parameters. If the src\_v and dest\_v variables are not None and contain a language, then they are passed accordingly to the translate function

1. **Define a function to clear the text widgets:**

#Function to clear the text boxes

def clear():

dest\_entry.delete("1.0","end-1c")

src\_entry.delete("1.0","end-1c")

text\_entry.delete("1.0","end-1c")

**Code explanation:**

* **dest\_entry.delete():** This function deletes the contents of the text widget within a given range. To completely clear the contents, provide the start and end index as ‘1.0’ and ‘end-1c’. Similarly extend for all text boxes

1. **Define the GUI and invoke the Translator() class:**

#Invoke call to class to view a window

window = Tk()

#Set dimensions of window and title

window.geometry("500x300")

window.title("PythonGeeks-Language Translator")

#Import the Translator class which will read the input and translate

#Default translation is done by detection of input and to english

translator\_object = Translator()

**Code explanation:**

* **Window = Tk():** The class Tk() contains all the widgets necessary to develop the user interface. Thus we call the class and assign it to the object window.
* **window.geometry():** Set the dimensions of the window. First parameter is the length and the second parameter is the breadth of the window
* **window.title():**  Set the title for the application.
* **translator\_object = Translator():** Invokes the translator class which contains functions to translate user text and view the supported languages.

1. **Create GUI components and read input:**

#Title of the app

title\_label = Label(window, text="PythonGeeks Language Translator Using Python",font=("Gayathri", 12)).pack()

#Read inputs

#Text input

text\_label = Label(window, text="Text to translate:").place(x=10,y=20)

text\_entry = Text(window, width=40, height=5,font=("Ubuntu Mono",12))

text\_entry.place(x=130,y=20)

#Source language input

src\_label = Label(window, text="Source language (empty: auto-detect):").place(x=10,y=120)

src\_entry = Text(window, width=20,height=1,font=("Ubuntu Mono",12))

src\_entry.place(x=275,y=120)

#Destination input

dest\_label = Label(window, text="Target language (empty: english-default):").place(x=10,y=150)

dest\_entry = Text(window, width=20,height=1,font=("Ubuntu Mono",12))

dest\_entry.place(x=300,y=150)

#Translate function and clear function activated through buttons

button1 = Button(window,text='Translate', bg = 'Turquoise',command=translate\_function).place(x=160,y=190)

button2 = Button(window,text='Clear', bg = 'Turquoise',command=clear).place(x=270,y=190)

#close the app

window.mainloop()

**Code explanation:**

* **Label():** View the title and other textual data on the window using Label() which takes 2 main arguments: window of the application and text to display. Font styling is optional. Pack sets the position to the center. The optional parameter pady centers it from y distance from the top margin. The content Label() displays is not editable on the app.
* **Text() widget:** Read user input using Text() widget. It takes 3 parameters: window of the application, width of the app and the height, optional font styling parameters using font().
* **place():** Position the widgets and labels using place(). This function takes x and y parameters, where x corresponds to distance from the left and y corresponds to the distance from the top.
* **Button():** Button widget activates the function mentioned in the command parameter. The parameters text is the text on the button, bg is the colour of the button and command activates the mentioned function

1. **Extras:**

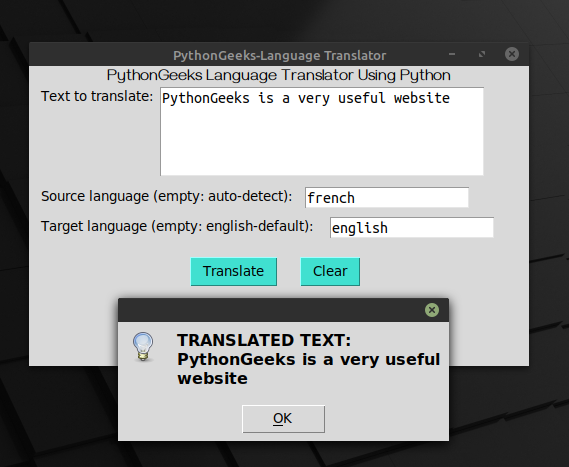
print(googletrans.LANGUAGES)

**Code explanation:**

* **googletrans.LANGUAGES:** You can view the list of languages and the corresponding codes supported by googletrans.

#### **Project output:**

Enter the inputs and view the output:



#### **Summary**

Thus using python, we created a simple language translator. Since Google translate API is a paid service with a limited free trial, we used the free library and executed the code. This project is also an introduction to tkinter and its widgets.