Transliteration based search for Indian languages

Transliteration is the process of transferring a word from the alphabet of one language to another. Transliteration helps people pronounce words and names in foreign languages. It is also useful for non-Latin script language speakers to use in their own language, as typing in Latin is more convenient.

Ex. **"नमस्ते"** transliterated to **"namaste"**

Google has now rolled out transliteration support in languages including Bangla, Gujrati, Hindi, Kannada, Malayalam, Marathi, Odia, Punjabi, Tamil, and Telugu. This new feature is aimed at people who have difficulty understanding English.

Syntax , parameters and Scripts

Syntax : transliterate(text, romanization_style, script)
Parameters :
test : The text totransliterated
romanization_style : The following romanization styles are available :
HK = ‘hk’
IAST = ‘iast’
ITRANS = ‘itrans’
OPTITRANS = ‘optitrans’
KOLKATA = ‘kolkata’
SLP1 = ‘slp1’
WX = ‘wx’

script : The script to be transliterated into. The following scripts are available :
Bengali
Devanagari
Gujarati
Kannada
Malayalam
Telugu
Gurmukhi/ Punjabi/ Panjabi
Returns : A string of the transliterated text.

Python – English to Hindi text convertor GUI using Thinter

Python offers multiple options for developing a GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create GUI applications. Now, it's up to the imagination or necessity of a developer, what he/she wants to develop using this toolkit.

To create a tkinter :
-Importing the module – tkinter
-Create the main window (container)
-Add any number of widgets to the main window.
-Apply the event Trigger on the widgets.

Below is the implementation in Python3:

```
# import sanscript class from the indic_transliteration module
from indic_transliteration import sanscript

# import transliterate method from sanscript
# class of the indic_transliteration module
from indic_transliteration.sanscript import transliterate

# import all functions from the tkinter
from tkinter import *
# Function to clear both the text areas
def clearAll() :
# whole content of text area is deleted
text1_field.delete(1.0, END)
text2_field.delete(1.0, END)
```

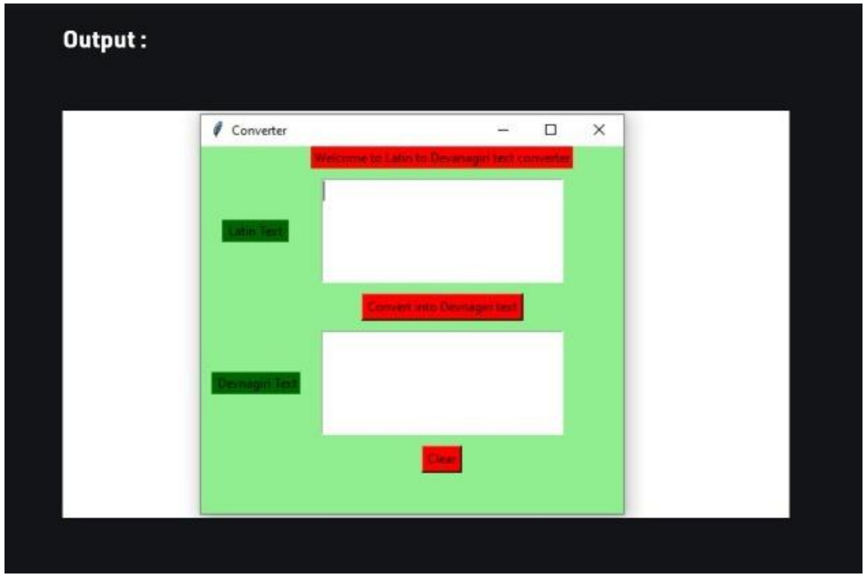
```
# Function to convert into Devanagari text
def convert() :

# get a whole input content from text box
# ignoring \n from the text box content
input_text = text1_field.get("1.0", "end")[:-1]

# converted into the given devanagari
# transliterated text
output_text = transliterate(input_text, sanscript.ITRANS,
sanscript.DEVANAGARI)

text2_field.insert('end -1 chars', output_text)

# Driver code
if _name_ == "_main_" :
# Create a GUI window
root = Tk()
# Set the background colour of GUI window
root.configure(background = 'light green')
# Set the configuration of GUI window (WidthxHeight)
root.geometry("400x350")
```

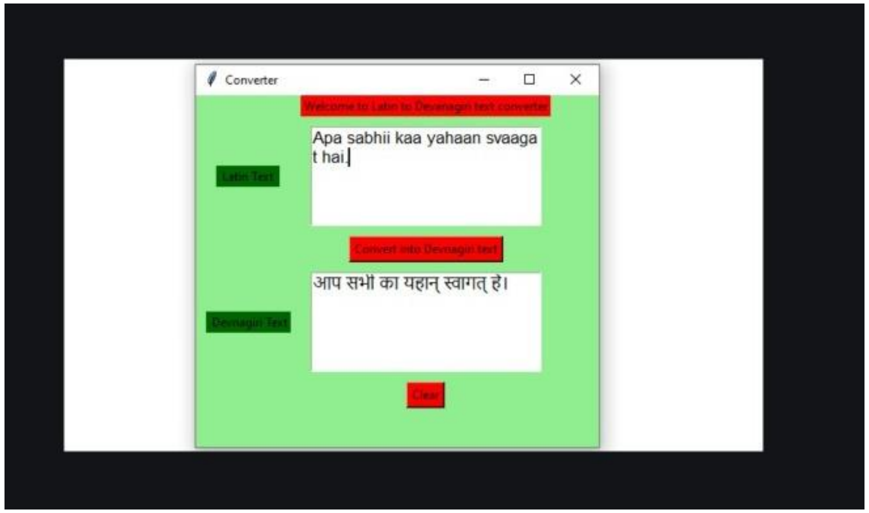
Python – English to Hindi text convertor GUI using Thinter

```
# set the name of tkinter GUI window
root.title("Converter")
# Create Welcome to Latin to Devanagiri text converter
headlabel = Label(root, text = 'Welcome to Latin to Devanagiri text converter',
fg = 'black', bg = "red")


# Create a " Latin Text " label
label1 = Label(root, text = " Latin Text ",
fg = 'black', bg = 'dark green')
# Create a " Devanagiri Text " label
label2 = Label(root, text = " Devnagiri Text",
fg = 'black', bg = 'dark green')


# grid method is used for placing
# the widgets at respective positions
# in table like structure .
headlabel.grid(row = 0, column = 1)


# padx keyword argument used to set padding along x-axis .
# pady keyword argument used to set padding along y-axis .
label1.grid(row = 1, column = 0, padx = 10, pady = 10)
label2.grid(row = 3, column = 0, padx = 10, pady = 10)
```

```
# Create a text area box
# for filling or typing the information.
text1_field = Text(root, height = 5, width = 25, font = "lucida 13")
text2_field = Text(root, height = 5, width = 25, font = "lucida 13")
# padx keyword argument used to set padding along x-axis .
# pady keyword argument used to set padding along y-axis .
text1_field.grid(row = 1, column = 1, padx = 10, pady = 10)
text2_field.grid(row = 3, column = 1, padx = 10, pady = 10)


# Create a Convert Button and attached
# with convert function
button1 = Button(root, text = "Convert into Devnagiri text",
bg = "red", fg = "black", command = convert)
button1.grid(row = 2, column = 1)


# Create a Clear Button and attached
# with clearAll function
button2 = Button(root, text = "Clear", bg = "red",
fg = "black", command = clearAll)
button2.grid(row = 4, column = 1)
# Start the GUI
root.mainloop()dd text
```

Python – English to Hindi text convertor GUI using Thinter


Output :
Converter
Welcome to Latin to Devanagiri text converter
Latin Text
Convert into Devnagiri text
Devnagiri Text
Clear


Converter
Welcome to Latin to Devanagiri text converter
Latin Text
Convert into Devnagiri text
Devnagiri Text
Clear

```
Apa sabhii kaa yahaan svaaga
t hai.
```

```
आप सभी का यहान् स्वागत् हे।
```

Available Models for transliteration

Transliteration tool used for Indian languages :

Unicode Transformation format is an encoding standard for characters, that gives a unique number to every single character in every single language.

WX-Notation: WX notation is a transliteration scheme to denote a script in Roman script. It defines a standard for the representation of Indian Languages in Roman script. These standards aim at providing a unique representation of Indian Languages in Roman alphabet.

ITRANS is a well known old transliteration software. It works with special Indic fonts which the user has to download before using it. ITRANS provides transliteration for Devanagari (Sanskrit/Hindi/Marathi), Tamil, Telugu, Kannada, Bengali, Gujarati and Gurmukhi.

JTRANS is a package similar to ITRANS , but has been written in javascript. So you can download it and use it offline.

References

Transliteration among Indian Languages using WX Notation

In this paper, we present an algorithm for the efficient transliteration between Indian Languages. We presented a brief overview of UTF and WX notations and then our algorithm that involved transition from UTF to WX of source language and then back to UTF for target language.