# CONVERT SPEECH TO TEXT AND TEXT TO SPEECH PYTHONGEEKS

#### **About speech to text and text to speech convertor in python project:**

There are times when we require an application to read out the text such as phones or even during transcription and when we need audio to be converted to text for usage such as note taking etc. In this article, we will see a simple implementation of the project using two libraries: Speechrecognition and GTTS

#### **Speech to text and text to speech convertor in python project:**

To implement the project, we need to install two libraries and an additional library. This project is for beginners, hence no prerequisite knowledge is required

#### **Project Prerequisites:**

We use tkinter to build the GUI of the application which is a built-in library in python. Import it to test its installation and refer to the following in case an error pops up: [Tkinter Installation](https://tkdocs.com/tutorial/install.html).

The other modules to install using pip are:

* pip install SpeechRecognition
* pip install gTTS
* pip install PyAudio

#### **Download code:**

You can download the source code in the given link: <add link>

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

Below is the flow of the project.

1. Importing necessary libraries
2. Declaring functions to view languages and their codes, convert text to speech and speech to text
3. Creating a user interface

##### **Importing necessary libraries:**

#Convert Speech to text and text to Speech: PythonGeeks

#import packages

from gtts import gTTS, lang

import os

from tkinter import \*

from tkinter import messagebox

import speech\_recognition as sr

**Code Explanation:**

* **from gtts import gTTS, lang:** To convert text to speech, we use the library gtts, Google Text to Speech.
* **import os:** To control the system, we use this library. Using this library we can create, switch, view or delete directories and so on.
* **from tkinter import \*:** Create the user interface using Tkinter which contains widgets and support to receive user input.
* **from tkinter import messagebox:** Prompt the user using dialog boxes using messagebox in tkinter
* **import speech\_recognition as sr:** Import the module to convert speech to text.

##### **Declaring functions to view languages and their codes, convert text to speech and speech to text:**

#define functions

#text to speech conversion

def text\_to\_speech():

#read inputs given by user

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

language = accent\_entry.get()

#Check if the user submitted inputs

if (len(text)<=1) | (len(language)<=0):

messagebox.showerror(message="Enter required details")

return

#Using the inputs, convert the text to speech

speech = gTTS(text = text, lang = language, slow = False)

#save the speech to an MP3 file

speech.save("text.mp3")

#Play the file using mpg123 in linux and start in windows

os.system("mpg123 "+"text.mp3")

**Code explanation:**

* **def text\_to\_speech():** Declare the function text\_to\_speech to initialise text to speech conversion.
* **text = text\_entry.get("1.0","end-1c"):** Obtain the contents of the text box using get. Since it is a Text widget, we specify the index of the string in get() to retrieve it. “ 1.0 ” indicates the start index and ”end-1c” is the last index
* **language = accent\_entry.get():** To get the accent, we use get() on the entry box, accent\_entry. Since it is an entry widget, we don't specify any index here
* **If….return:** Check if the user filled both fields and raise a prompt if anything is missing, using showerror() function of messagebox module. Message contains the message to tell to the user
* **speech = gTTS():** Use the gTTS() module to convert text to speech with the parameters: text, language accent and slow set to False. Slow indicates the speed of the audio.
* **speech.save():** Save the speech audio file using save. Specify the file name with the extension
* **os.system():** To play the audio, we use the system function of the OS library. This allows it to control the mp3 player of the laptop. We use mpg123 followed by the file name to play the file.

If mpg123 is not on your system, install it using sudo apt install mpg123

#List the supported languages and their keys

def list\_languages():

#access languages and access codes using lang.tts\_langs()

messagebox.showinfo(message=list(lang.tts\_langs().items()))

**Code explanation:**

* **def list\_languages():** Declare the function list\_languages() to display the supported languages..
* **messagebox.showinfo():** Display the languages in a prompt to the user. This is analogous to showerror() with the difference being the symbol in the prompt box. Since lang.tts\_langs() is a dictionary, we access the items using items and display it as a list

#speech to text conversion

def speech\_to\_text():

#Initialise the recognizer class

recorder = sr.Recognizer()

try:

duration =int(duration\_entry.get())

except:

messagebox.showerror(message="Enter the duration")

return

#use the microphone

messagebox.showinfo(message="Speak into the microphone and wait after finishing the recording")

with sr.Microphone() as mic:

#Prompt the user to record

#Record audio from the user

recorder.adjust\_for\_ambient\_noise(mic)

audio\_input = recorder.listen(mic, duration=duration)

try: #Convert to text

text\_output =recorder.recognize\_google(audio\_input)

#Display the output

messagebox.showinfo(message="You said:\n "+text\_output)

except:

messagebox.showerror(message="Couldn't process the audio input.")

**Code explanation:**

* **def speech\_to\_text():** Declare the function speech\_to\_text() to initialise speech to text conversion.
* **recorder = sr.Recognizer():** Initialise the recogniser class to the recorder.
* **duration:** Read the duration from the duration\_entry widget using get() and typecast it to int()
* **with sr.Microphone() as mic:** Activate the microphone and use it to record the audio
* **recorder.adjust\_for\_ambient\_noise(mic, duration=0.2):** Adjust the recorder for ambient noise using the parameters mic and set the duration
* **audio\_input = recorder.listen(mic):** Listen to audio using the listen function of the recorder
* **text\_output:** Send the recorded audio to recognize\_google and display the text in a prompt

##### **Creating a user interface:**

#Invoke call to class to view a window

window = Tk()

#Set dimensions of window and title

window.geometry("500x300")

window.title("Convert Speech to text and text to Speech: PythonGeeks")

title\_label = Label(window, text="Convert Speech to text and text to Speech: PythonGeeks").pack()

#Read inputs

#text\_to\_speech input

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

text\_entry = Text(window, width=30,height=5)

text\_entry.place(x=80,y=20)

#Accent input

accent\_label = Label(window, text="Accent:").place(x=10,y=110)

accent\_entry = Entry(window, width=26)

accent\_entry.place(x=80,y=110)

duration\_label = Label(window, text="Duration:").place(x=10,y=110)

duration\_entry = Entry(window, width=26)

duration\_entry.place(x=80,y=140)

#Perform the functions

button1 = Button(window,text='List languages', bg = 'Turquoise',fg='Red',command=list\_languages).place(x=10,y=190)

button2 = Button(window,text='Convert Text to Speech', bg = 'Turquoise',fg='Red',command=text\_to\_speech).place(x=130,y=190)

button3 = Button(window,text='Convert Speech to Text', bg = 'Turquoise',fg='Red',command=speech\_to\_text).place(x=305,y=190)

#close the app

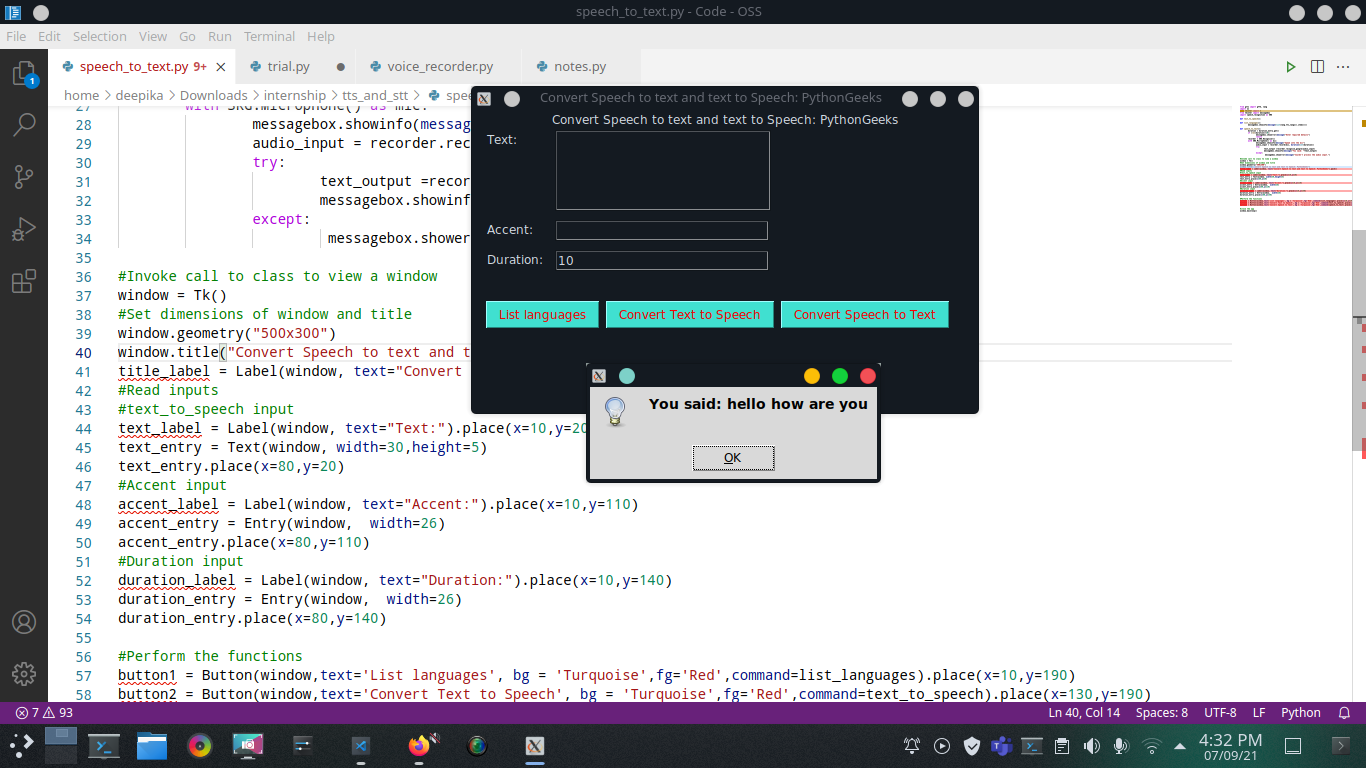
window.mainloop()

**Code explanation:**

* **window = Tk():** Initialise the window with tkinter constructor to use the objects and widgets
* **window.geometry("500x300"):**Set the dimensions of the window by specifying the width and the height of the window
* **window.title():** Give the application window a title
* **title\_label, text\_label, duration\_label, accent\_label:** Define a label with the parameters: window of the screen and the text to display. Labels cannot be copied or edited.
* **accent\_entry, duration\_entry:** Entry widget is an input field to obtain user input. Specify the width of the widget using the width parameter.
* **text\_entry:** Text widget is another input field to obtain user input. Use this widget to obtain long lines of text from the user. Specify the height and width of the text box.
* **place():** Place is another positioning element analogous to pack(). Here we specify the distance from the left margin and the top margin in the x and y coordinates respectively.
* **pack():** Pack() is an automatic formatting which positions the element in the center of a row.
* **button1, button2, button3:** Buttons perform a function when the user selects it. The parameters are window of the application, name of the button, background colour of the application, text colour using the foreground and the function call is invoked using command parameter.
* **window.mainloop():** When the user terminates the application, the control flows beyond this line thereby terminating the application. Widgets placed after this line will not be displayed

#### **Project output:**

Run the program and view your output



#### **Summary**

Thus we implemented a speech to text and text to speech program in python. This program is a great introduction to various libraries.