Voice\_assistant\_app:

I was inspired by following things(voice assistants like Alexa, Siri,Google,Jarvis etc.) and it made me to search more about it and make a one for me.   I made my Voice Assistant(@diG48)using Python. The query for the assistant can be manipulated as per the user’s need.

Speech recognition is the process of converting audio into text. Python provides an API called **Speech\_Recognition** to allow us to convert audio into text for further processing. In this article, we will look at converting large or long audio files into text using the SpeechRecognition API in python.To make the voice\_assistant\_app more interactive and do more functions,I used many built in modules such as

pyttsx3, datetime, speech\_recognition, pywhatkit, Wikipedia, smtplib, pyjokes.

I explained about every madule and whole code line by line as follows….

To get the entire code browse my git\_hub profile:

github.com/adiG48/voice\_assistant\_app

**\*IMPORTING ALL NEEDED MODULES:**

I used pip to install all the commands in the terminal of pycharm. So we can use all the modules whenever needed without any errors, the command to install the modules are listed below.

Import all the following modules:

**from tkinter import \*  
 import pyttsx3  
 import datetime  
 import speech\_recognition as sr**

**import pywhatkit  
 import wikipedia  
 import smtplib  
 import pyjokes  
 from email.message import EmailMessage**

\*pip install tkinter

\*pip install pyttsx3

\*pip install speech\_recognition

\*pip install pywhatkit

\*pip install wikipedia

\*pip install smtplib

\*pip install emails

\*pip install pyjokes

**\*FRONT\_END\_PART:**

I had made the frontend part for my voice\_assistant\_app using Tkinter in python. I used this because it is easy to make window and buttons and we can display the content whenever needed easily and can update the screen while running easyly.

\*pip install tkinter

**Code:**

**from tkinter import \*** #importing everything from the module.

**window = Tk()** #cretingthetkinter window. **global var** # declaring 2 global variables to access it any where. **global var1  
var = StringVar()** # stringVar() is a tkinter module that holds the string values, by default it holds “” empty string.  
**var1 = StringVar()** # to hold the values that user says we are assigning it the variable which can be updated while running.

**label2 = Label(window, textvariable=var1, bg='#FAB60C'**)# labling the window using label(), ‘text variable’ is the value that we want to display so I’m assigning it to the variable ‘var1’ so it can be updated when the value changes, finally passing the third argument as background color, we can use any color code as a background.  
**label2.config(font=("Courier", 20))** # configuring our text such as font and size using ‘.config()’.  
**var1.set('User Said:')** # initially setting the value that to be displayed which is an optional one **label2.pack()** # it is used to update the widjet we made with minimal space and block of allocation.  
 **label1 = Label(window, textvariable=var, bg='#ADD8E6')** # same as above setting another variable  
**label1.config(font=("Courier", 20))  
var.set('Welcome')  
label1.pack()  
  
frames = [PhotoImage(file='Assistant3.gif', format='gif -index %i' % i) for i in range(100)]** #Tkinter Photoimage is one of the built-in methods which has been used to add the user-defined images in the application**.** Iterating it over a for loop since to animate the gif.  
**window.title('@diG48')** # giving title to our window using ‘.title()’  
 **label = Label(window, width=1000, height=500)** # labling the window dimensions  
**label.pack()  
window.after(0, update, 0)**# This method calls the function FuncName after the given delay in milisecond **btn1 = Button(text='START', width=23, command=play, bg='#5C85FB')** # creating the buttons on the window using button(),which takes arguments such as text and font size, if the buuton is pressed taking the command varible to operate some functions  
**btn1.config(font=("Courier", 12))** # configuring the buttons  
btn1.pack()  
**btn2 = Button(text='EXIT', width=25, command=window.destroy, bg='#12cee3')** # adding another button named exitand using a built in module to destroythe window created. **btn2.config(font=("Courier", 12))  
btn2.pack()  
  
window.mainloop() #** window.mainloop() tells Python to run the Tkinter **event loop**. This method listens for events, such as button clicks or keypresses, and **blocks** any code that comes after it from running until the window it’s called on is closed.

**\* UPDATE MODULE:**

Used to update the gif on the screen after the specified amount of time.

**def update(ind):** # used to update the gif. **frame = frames[ind % 100]  
 ind += 2  
 label.configure(image=frame)  
 window.after(100, update, ind)** # updating the gif after specified time of intervals.

**SAMPLE\_IMAGE:** 

**\*PYTTSX3 MODULE**:

First to make the app interactive with the user I made use of pyttsx3(python text to speech module) which speaks the text what we have typed.

\*pip install pyttsx3

**Code**:

**import pyttsx3** *#importing the module*

**engine = pyttsx3.init()** *#initializing speech-text in var engine*  
**voices = engine.getProperty('voices')** *#from pyttsx3 getting property as voices and assigning it to voices*  
**engine.setProperty('voice', voices[1].id)** *#from built in voices we are setting it to a female voice using “voice[1]”, to use a male voice we can use “voice[0]”*

To speak what we have typed I defined a function as follows:

**Code:**

**def speak(audio):   
 engine.say(audio)** *# say is a module that speaks what we send as input.*  
 **engine.runAndWait()** *# to make it wait wait for the users response.*

When we call this function with an argument with strings it will speak it.

**\*SPEECH\_RECOGINITION MODULE:**

To get the voice from user and convert it into text this module is used.

\*pip install speech\_recoginition

I put everything inside a function named “take\_command():”

**Code:**

**import speech\_recognition as sr** # *importing the module as sr to call it with easy way.*

**def takecommand():** *# function to take command from user*  
 **r = sr.Recognizer()** #initializing r as recognizer of our voice.  
 **with sr.Microphone() as source:** # making our microphone as source to take voice.  
 **var.set("Listening*...")*** *#used in app to diplay* ***window.update()*** *#updating the window of the app to make the changes what we did.* **print("Listening...")  
 r.pause\_threshold = 1** *# making the pause threshold of our voice as 1* **r.energy\_threshold = 400** *# if our voice is over the threshold of 400 it will recognize it* **audio = r.listen(source)** *#listening the voice using listen() and assigning it to audio.*  
 **try:  
 var.set("Recognizing...")  
 window.update()  
 print("Recognizing")  
 query = r.recognize\_google(audio, language='en-in')** *# recognizing the audio with google with language as Indian English.*  
 **except Exception as e:** #*exception block to avoid errors if nothing is said.*  
 **return "None"  
 var1.set(query)  
 window.update()  
 return query** *# finally retuning the value of query which we use later in the app.*

**\*EMAIL\_SENDER\_MODULE:**

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides smtplib module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail :

**Code:**

**import smtplib**

**smtpObj = smtplib.SMTP( [host [, port [, local\_hostname]]] )**

**def send(msg):  
 with smtplib.SMTP\_SSL('smtp.gmail.com', 465) as smtp:** # logging into the smtp server.   
 **smtp.login('Enter the mail id', 'password’)** # login into your email id from where u want send messages. “# Make sure to give app access in your Google account” **to do so follow 🡪 go to the mail 🡪 settings 🡪 security 🡪 less secure app access 🡪 turn it on.  
 smtp.send\_message(msg)** # sending the message using the builtin sen\_message().

**try:** # Trying this block of code if user wants to send an email. **msg = EmailMessage()** #used to pass the values to a variable as in E-Mail format like from,to and subject. **c = {'name1': 'mail id', name2': mail id'}** # giving the name as key and assigning it with the respected mail id’s. **speak('To Whom You Want To Send Email')** # Speaks what is inside the quotes.  
 **b = takecommand()** # from user taking command as voice and assigning it to a variable. **d = c[b]** # checking wether the key is present what user command. **msg['To'] = d** # assigning TO message of the E-mail **speak('Tell Me The Subject In Your Email')  
 a = takecommand()** # Taking command from user for subject **msg['Subject'] = a** # assigning SUBJECT of the E-Mail **msg['From'] = 'adithyaaaibot@gmail.com'** # assigning FROM of the E-Mail.  
 **speak('Tell Me The Content Of The Email')  
 e = takecommand()  
 msg.set\_content(e)** # setting the content of the E-Mail. **send(msg)** # send() is used to send the message to the desired one and it is passed to the function that we have decleared earlier. **var.set('Your Email Has Been Sent Sir!!')** # to display on the screen  
 **speak('Your Email Has Been Sent Sir!!')** # confirmation message said by our voice\_assistant.  
 **except Exception as e:** # excepting the errors as e, to make exception hanling while running. **print(e)  
 var.set("Sorry Sir! I was not able to send this email")  
 window.update()  
 speak('Sorry Sir! I was not able to send this email')** # say’s this when an exception occurs or the E-mail is not sent.

**\*WISH\_MODULE:**

I made this function to greet the user whenever he presses the play button, my voice\_Assistant will greet the user according to the current time in their system. This function can later called when we are inquiring our command from the user.

**Code:**

**def wishme():  
 hour = int(datetime.datetime.now().hour)** # from datetime module we are getting the current time of the system in hours. **if (hour >= 0) and (hour <= 12):** # it will check the hour and greet us according to the time hour condition we mentioned **var.set("Good Morning Sir")  
 window.update()  
 speak("Good Morning Sir!")  
 elif (hour >= 12) and (hour <= 18):  
 var.set("Good Afternoon Sir!")  
 window.update()  
 speak("Good Afternoon Sir!")  
 else:  
 var.set("Good Evening Sir")  
 window.update()  
 speak("Good Evening Sir!")  
 speak("Myself @diG48! How may I help you sir")** # after greeting us it will say it’s name and ask us how it could help us.

**\*PLAY\_MODULE:**

I had put everything that is to be done when the play button is pressed. I used simple ‘if elif else’ condition statements to check wether the user said is present in there are not using ‘in’ statement. I put like playing musics, asking who is that person,greet, time, date, name of the bot, by whom it created, saying hello to everyone, sending E-Mail, sending whatsapp messages and giving some inbuilt jokes etc… we can make more tasks to be done if we want.

**Code:**

**def play():** # this module is called when the play button on the screen is pressed. **btn2['state'] = 'disabled'** # then I made the exit button disabled.  
 **btn1.configure(bg='orange')** # changing the color of the buttons after the play button is pressed.  
 **btn2.configure(bg='#5C85FB')** # changing the color of the buttons after the play button is pressed. **wishme()** # calling the wishme() module to greet the user. **while True:** # until the exit button is enabled it will execute repeatedly. **btn1.configure(bg='orange')** # changing the button color. **query = takecommand().lower()** # converting everything from takecommand() to lower case which is said by the user to execute further.  
 **if 'exit' in query:** # if the user asks or says exit in takecommand() this block will be executed. **var.set("Bye sir")   
 btn1.configure(bg='#5C85FB')** # changing the color of the buttons back to intial stage.  
 **btn2['state'] = 'normal'** # enabling the exit button to quit the program.  
 **btn2.configure(bg='#12cee3')  
 window.update()  
 speak("Bye sir")** # saying bye to the user **break** # breaking the looping statement. **elif 'play' in query:** # if user asks to play something it will be played in youtube.  
 **song = query.replace('play', '')** # removing the play statement said by the user and searching the other in YT.  
 **speak('Playing' + song)** # says palyin the song which u said.  
 **pywhatkit.playonyt(song)** # pywhatkit’s inbuilt module to paly anything on YT.  
 **window.update()** # updating the window **takecommand()  
 window.update()** # updating the window after exit from YT.  
 **elif 'who is' in query:** # if user asks who is this/that person is this block will be executed.  
 **person = query.replace('who is', '')** # replacing or removing who is and taking only the name of the person. **info = wikipedia.summary(person, 1)** # wikipedia’s inbuilt module to search about a person in google and taking only the 1 statement.

**speak(info)** # this will call the speak module and speak the information what it got from Wikipedia.  
 **elif 'greet' in query:** # if user says gteet it will wish what we said in the speak module and displays it as its value is assigned to a global variable ‘var’. **var.set('Hello Sir, have a nice day')  
 window.update()  
 speak("Hello Sir, have a nice day")  
  
 elif 'the time' in query:** # if user asks for time this block will be executed.  
 **strtime =datetime.datetime.now().strftime("%H:%M:%S")** #this statement will fetch the correct time in our system using the builtin module datetime() and give us the format in which we want. **var.set("Sir the time is %s" % strtime)  
 window.update()  
 speak("Sir the time is %s" % strtime)  
  
 elif 'the date' in query:** # if user asks for the date today it will fetch the current date in which format we specified. **strdate = datetime.datetime.today().strftime("%d %m %y")** # using datetime() method we can refer the date.  
 **var.set("Sir today's date is %s" % strdate)  
 window.update()  
 speak("Sir today's date is %s" % strdate)  
  
 elif 'thank you' in query:** # if user says thank you this block will be executed and speaks whatever given in speak(). **var.set("Welcome Sir")  
 window.update()  
 speak("Welcome Sir")  
  
 elif 'your name' in query:** # if user asks for the robots name it will say the following which is specified inside this block. **var.set("Hi, Im @diG48 the Robot. Speed 1 terahertz, memory 1 zigabyte ")  
 window.update()  
 speak('Hi, Im @diG48 the Robot. Speed 1 terahertz, memory 1 zigabyte ')  
  
 elif 'who created you' in query:** # if user asks for who created the robot the following statements will be executed. **var.set('I was created by AdithyaaG48')  
 window.update()  
 speak('I was created by AdithyaaG48')  
  
 elif 'say hello' in query:** # if user asks say hello the following statement will be executed. **var.set('Hello Everyone! My self @diG48')  
 window.update()  
 speak('Hello Everyone! My self @diG48')  
  
 elif 'email' in query:** # if user wants to send an email this E-Mail module which I mentioned above will be executed. **try:  
 msg = EmailMessage()  
 c = {'black': 'adithyaasankar2@gmail.com', 'Shiny mam': 'shiny.suresh@gmail.com'}  
 speak('To Whom You Want To Send Email')  
 b = takecommand()  
 d = c[b]  
 msg['To'] = d  
 speak('Tell Me The Subject In Your Email')  
 a = takecommand()  
 msg['Subject'] = a  
 msg['From'] = 'adithyaaaibot@gmail.com'  
 speak('Tell Me The Content Of The Email')  
 e = takecommand()  
 msg.set\_content(e)  
 send(msg)  
 var.set('Your Email Has Been Sent Sir!!')  
 speak('Your Email Has Been Sent Sir!!')  
  
 except Exception as e:  
 print(e)  
 var.set("Sorry Sir! I was not able to send this email")  
 window.update()  
 speak('Sorry Sir! I was not able to send this email')  
 elif 'joke' in query:** # if user asks for a joke this will be executed.  
 **speak(pyjokes.get\_joke())** # pyjokes module has builtin function to get some of the builtin jokes inside it’s module and gives us a random joke whenever executed.

To make the app more beautiful, I added more images and made them to display on the window when the respected block is executing. I added it by a simple way as follows

Created more frames and called it whenever needed

**Code:**

**frames0 = [PhotoImage(file='name.png')]** # initializing the frames with desired images using ‘PhotoImage() method’. **frames1 = [PhotoImage(file='creator.png')]  
frames2 = [PhotoImage(file='time.png')]  
frames3 = [PhotoImage(file='MAIL.png')]  
frames4 = [PhotoImage(file='date.png')]  
frames5 = [PhotoImage(file='WA.png')]  
frames6 = [PhotoImage(file='hello.png')]  
frames7 = [PhotoImage(file='joker.png')]  
frames8 = [PhotoImage(file='YT.png')]  
frames9 = [PhotoImage(file='creator.png')]  
frames10 = [PhotoImage(file='bye.png')]  
frames11 = [PhotoImage(file='who.png')]  
frames12 = [PhotoImage(file='welcome.png')]**

Calling the frames whenever needed as follows and updating the window then and there.

**Code:**

**label.configure(image=frames0)** # configuring the window with the respected image.  
**window.update()** # updating the window to view the updated image.

The following image shows what are the images I have used:



**To get the entire code browse my git\_hub profile:**

github.com/adiG48/voice\_assistant\_app