Python Voice Assistant

Period covered: Nov'20 – Jan'21

Project partner(s): RITIKA DEY, RAHUL KUMAR

Introduction ~

A personal voice assistant is a software which perform tasks of an individual according to the voice commands. This assistant is able to interpret human voices and respond via synthesized voice and text. It uses Google speech recognition API and many python libraries for completing the tasks assigned by user.

Functionalities ~

The implemented assistant can do many tasks such as play music, open YouTube, Google, StackOverflow and many sites of users choice, take screenshots, predict current time, search Wikipedia and Google, open system apps, open google map of a particular location and many more.

Packages used ~

- Speech recognition
- Pyttsx3
- DateTime
- Wikipedia
- WebBrowser
- Os
- Random
- PyAutoGui
- gTTS (Google text to speech)

Modules description ~

- **Speech recognition**: Speech recognition is an important feature of artificial intelligence. It allow us to manage the unwanted surrounding noise. We can control the noise by adjusting the energy threshold of recording.
- **Pyttsx3**: pyttxs3 is a python library used for conversion of speech to text. The main function of this module is to covert the voice command and to text. Pyttsx3 supports three different synthesizers :
    1. SAPI5 for Windows
    2. NSSS for Mac OS
    3. eSpeak for Ubuntu
- **Wikipedia:** This library is used here for getting search results from Wikipedia. We can easily fetch data using this python library. Wikipedia library have many methods such as ***search, summary, suggest*** used for searching a query, getting search suggestions and plain texts as summary of the page for the given query respectively. There are many more methods and classes which can be used.
- **DateTime:** Datetime is a in-built library in python it provides the information about the date object. In our application we used this library for getting current time. We used ***now*** method which creates a datetime object containing current local date and time. In this module we also used ***strftime*** method by which we can get output time in desired format.
- **WebBrowser**: It is also an in-built library in python. It provides high-level interface which allows to show web-based documents to users. We used the ***open*** and ***open_new_tab*** method to load the requested pages in current window and in a new window respectively. It also provides a ***get*** method by which we can show the results in browser of our choice rather than the default browser of the system.
- **Os :** It is a standard library in python. It allows us to use operating system dependent functionalities. OS module provides us lots of methods which allow us to read and write files, manipulate paths, creation of temporary files and directories and high level file handling etc.

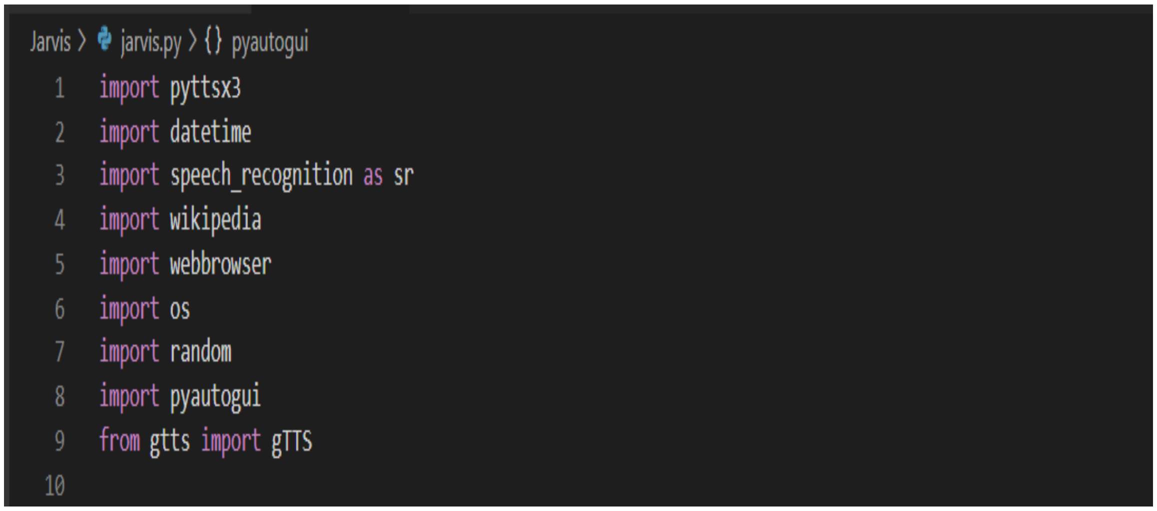
- **Random:** Random is also an in-built library in python. It is used for random selection of elements from a given range. In this application it is used for random selection of song from the songs array.
- **PyAutoGui:** It is a cross platform GUI automation module used for programmatically mouse and keyboard control. PyAutoGui is used for screenshot functions, displaying message box and programmatically mouse and keyboard control. PyAutoGui runs on Windows, macOS and linux.
- **gTTS:** Google text-to-speech is a python library and command line tool to interface with Google Translate's text-to-speech API.

Installations ~

- pip install speechRecognition
- pip install pyttsx3
- pip install Wikipedia
- pip install webbrowser
- pip install pyautogui
- pip install gTTS

Implementation ~

Imports: Following libraries are imported.



Jarvis > jarvis.py > {} pyautogui

```
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import random
import pyautogui
from gtts import gTTS
```

Initialization of speech engine:

Firstly pyttsx3.init() function is invoked to get an instance of pyttsx3.Engine object and stored in a variable. The ***init*** method takes two parameters first is the driver name which is a string value and the second is debug which is Boolean value, driver name depends on the system we are using. Here we used ***Sapi5*** driver which is a Microsoft developed speech recognition API.

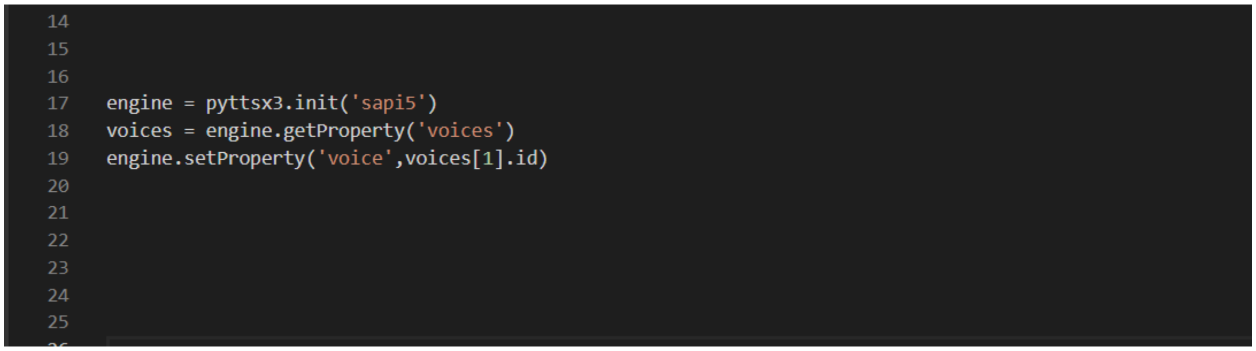
engine = pyttsx3.init(‘sapi5’)

After that ***getProperty*** method is used to get the current value of engine property. It takes the property name as parameter and returns value of property at the time of invocation. In this application we used \`voices\` property which is a list of pyttsx3.voice.Voice descriptor objects.

voices = engine.setProperty(\`voices\`)

Then setProperty method is used to queue a command to engine property. It takes two parameters first one is \`voice\` which is a string identifier of current voice and the second one is voice id set to either 0 or 1 , where 0 is for male voice and 1 is for female voice.

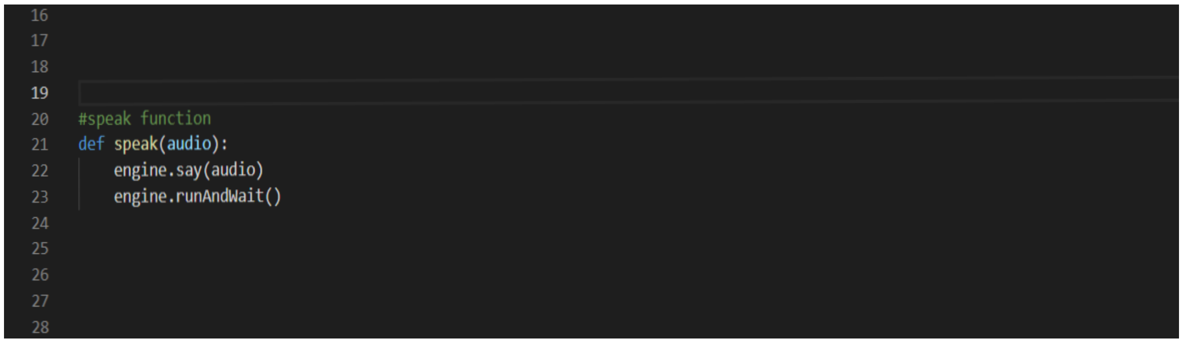
engine.setProperty(\`voice\`,voices[1].id)



```
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice',voices[1].id)
```

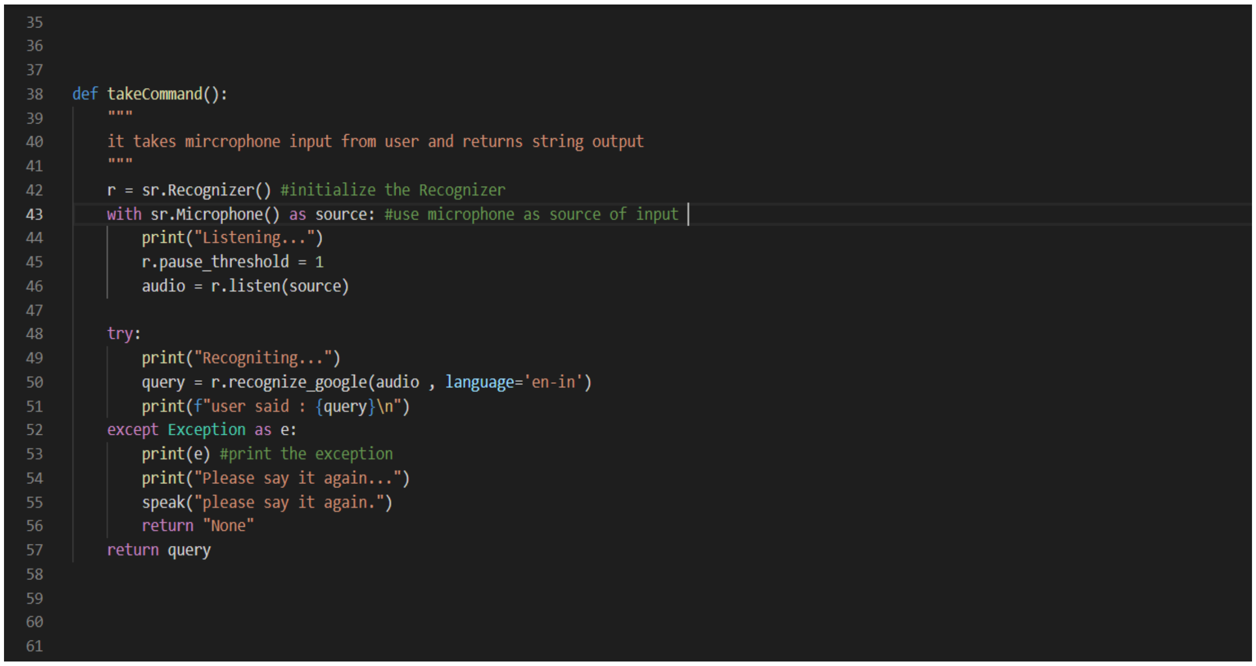
Speak function:

Speak function is used to convert text to speech. It takes text to speak as argument. ***Say*** method is used in the function which queue the command to speak an assertion. For blocking calls while all queued commands are possessing ***runAndWait*** method is used. It returns when all the queued commands before the current command are emptied from the queue.


```
#speak function
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
```

TakeCommand function:

This function is used to take command from the user. Microphone grabs the human voice command and recognizer recognizes the speech to give response. Here ***Recognizer*** function initializes the recognizer and ***Microphone*** function used for connecting microphone to take user input. ***Listen*** method listens the voice command and then this command is recognized by ***recognize_google*** method which is the Google Speech API. Exception handling is used for any type of runtime exception.


```
def takeCommand():
    """
    it takes mircrophone input from user and returns string output
    """
    r = sr.Recognizer() #initialize the Recognizer
    with sr.Microphone() as source: #use microphone as source of input
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recogniting...")
        query = r.recognize_google(audio , language='en-in')
        print(f"user said : {query}\n")
    except Exception as e:
        print(e) #print the exception
        print("Please say it again...")
        speak("please say it again.")
        return "None"
    return query
```

Main function:

It’s time to wrap up all the things we have done till now in the main function. In the main function we executed a while loop and call the takeCommand function to take the query from user. All the functionalities are called from main method and when the user command is “Stop” the loop ends. Before the while loop a Greet function is called in which application greets the user.

```
if __name__ == "__main__":
  greetMe()
 while True:
 query = takeCommand().lower()
#if statements for execution of  commands
if "good bye" in query or "ok bye" in query or "stop" in query:
    print('your personal assistant is shutting down. have a nice day,bye')
    speak('your personal assistant is shutting down. have a nice day,bye')
    break
```

Functionalities ~

1. **Greetings:** The voice assistant will greet you whenever you open the application. Greetings is done by a ***GreetMe*** method which evaluates the current time and predicts is it morning, noon, evening or night and then greet accordingly.

```
#greeting function

def greetMe():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        speak("good morning ma'am. please tell me how can i help you")
    elif hour>=12 and hour<18:
        speak("good Afternoon ma'am. please tell me how can i help you")
    else:
        speak("good night ma'am. please tell me how can i help you")
```

2. **Fetching data from Wikipedia:** Whenever user gives a command to search via Wikipedia then the application take the query and search for that particular keyword in Wikipedia then summarize the results.

```
   if 'wikipedia' in query:
           speak("searching wikipedia")
           query = query.replace("wikipedia","")
           result = wikipedia.summary(query , sentences=2)
           speak("According to wikipedia")
           print(result)
           speak(result)
```

3. **Open applications and websites:** Using this application allows you to open your favorite desktop applications and websites such as YouTube, Google, StackOverflow, Visual Studio Code, Chrome, Gmail by giving just one command .

```
       elif 'open youtube' in query:
           webbrowser.open('youtube.com')

       elif 'open google' in query:
           webbrowser.open('google.com')

       elif 'open stack overflow' in query:
           webbrowser.open('stackoverflow.com')

       elif 'open gmail' in query:
            webbrowser.open_new_tab("gmail.com")

       elif 'open code' in query or 'open visual studio' in query:
            os.startfile("C:\\Users\\hp\\AppData\\Local\\Programs\\Microsoft
VS Code\\Code.exe")

       elif 'open chrome' in query:
            os.startfile("C:\\Program Files (x86)\\Google\\Chrome\\Applicatio
n\\chrome.exe")
```

4. **Play music:** This application can play music randomly from music folder by using ***startfile*** method of ***OS*** module and random method to select random song from music directory.

```
    elif 'play music' in query:
           songs = os.listdir("path to music directory")
           print(songs)
           os.startfile(os.path.join("path to music
directory”, songs[random.randrange(0 , len(songs) - 1)]))
           #use random module for playing random song
```

5. **Fetch data from web:** Data can be fetched using ***open_new_tab*** method of ***webBrowser*** module. The results fetched are shown in a new tab in your default web browser. Same work is done for fetching data from YouTube, also user is redirected to YouTube when the results are fetched from YouTube.

```
 elif 'search' in query:
           query = query.replace("search","")
           webbrowser.open_new_tab(query)

       elif 'in youtube' in query:
           query = query.replace("in youtube" , "")
           webbrowser.open_new_tab('https://www.youtube.com/results?search_qu
ery=' + query)
```

6. **Taking ScreenShot:** This application is able to take screenshot of the current page displaying on the system. This is done by using PyAutoGUI library and some of it’s methods such as **screenshot** and **save** used for taking the scrrenshot and saving it in provided path repectively.

```
elif 'take a screenshot' in query:
           myss = pyautogui.screenshot()
           myss.save('img.png')
           speak("screenshot is saved , do you want to see results ?")
           ans = takeCommand().lower()
           if 'yes' in ans:
                os.startfile(“path for the directory where image is saved”)
```

7. **Resume Builder:** Most esteemed part of the whole application is this **The Resume Builder** by which user are able to build there resume by just giving there information as asked by the assistant and a standard Resume is ready in less than 10 minutes. This resume builder is created with the help of python Docx library and it’s functions such as heading, add_paragraph and many more.

Link for code of Resume builder: Not available now

References :

- https://towardsdatascience.com/how-to-build-your-own-ai-personal-assistant-using-python-f57247b4494b
- https://en.wikipedia.org/wiki/Virtual_assistant
- https://pypi.org/project/SpeechRecognition/
- https://pyttsx3.readthedocs.io/en/latest/index.htm
- https://wikipedia.readthedocs.io/en/latest/
- https://www.w3schools.com/python/python_datetime.asp
- https://docs.python.org/3/library/webbrowser.html
- https://docs.python.org/2/library/os.html
- https://pypi.org/project/PyAutoGUI/
- https://pyautogui.readthedocs.io/en/latest/screenshot.html
- https://pypi.org/project/gTTS/
- https://python-docx.readthedocs.io/en/latest/