EMMA: THE PERSONAL VOICE ASSISTANT

PROBLEM STATEMENT:

We are all well aware of Cortana which is designed to help the users in Windows to do their task but the thing is this voice assistant can't operate on apps like Whatsapp i.e it doesn't have the feature of automated whatsapp message sending so I made my personal voice assistant which can send automated messages to my contacts. It can also open the apps that I install from external source, search Wikipedia, play music and many more features which I will list further in my documentation.

Lets see what are hardware and software requirements for this software.

REQUIREMENTS:

Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Installed RAM	8.00 GB (7.77 GB usable)
System type	64-bit operating system, x64-based processor
Edition	Windows 10 Home Single Language
Version	20H2
OS build	19042.1348
Software	Visual Studio Code
Python version	3.10.1

OTHER REQUIREMENTS:

- Proper internet connection
- Github Credentials
- Python 3.10.1
- Microphone(speech to text)

TECHNOLOGY USED:

LANGUAGE USED:	PYTHON
IDE USED :	VISUAL STUDIO CODE

SCOPE:

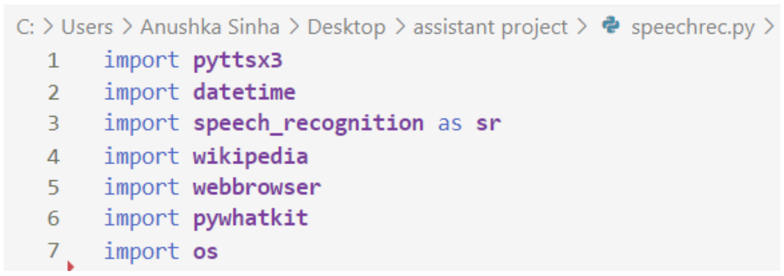
EMMA was developed as an automated tool and personal virtual assistant. The roles played by Emma are:

1. Send Whatsapp message with voice interaction.
2. Search Wikipedia from Web and shows the result in summary form.
3. Plays music from YouTube.
4. Interacts with you.
5. Tells the current time.
6. Opens our daily use application.
7. Shuts down the system for us.

LINK TO GITHUB REPOSITORY:
https://github.com/anu332002/assistant_hackathon.git

PROJECT DESCRIPTION:

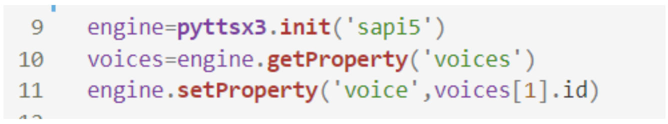
IMPORTS:



C: > Users > Anushka Sinha > Desktop > assistant project > speechrec.py >

```python
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import pywhatkit
import os
```

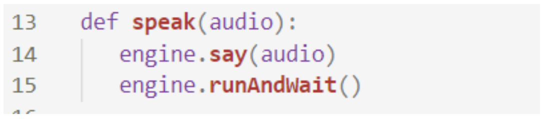
INITIALISING THE VOICE OF OUR VOICE ASSISTANT:



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

The pyttsx3 module will help us to convert text to speech.
In this we are initializing the voice of our assistant with the female voice present in sapi5 API which is used for speech recognition and speech synthesis within Windows application.
engine.getProperty sets the property of the model.

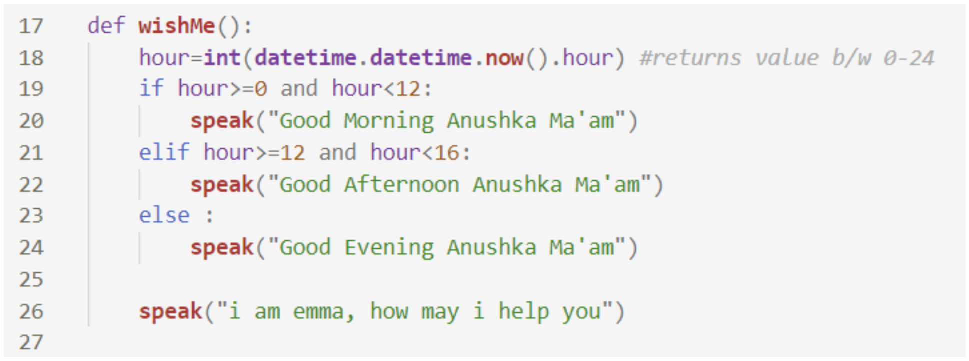
DEFINING SPEAK FUNCTION:



```python
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
```

The speak() function takes audio as argument and then pronounce it.
engine.runAndWait will make the speech audible in the system, if we don’t write this command then the speech will not be audible to you.

DEFINING WISHME FUNCTION:



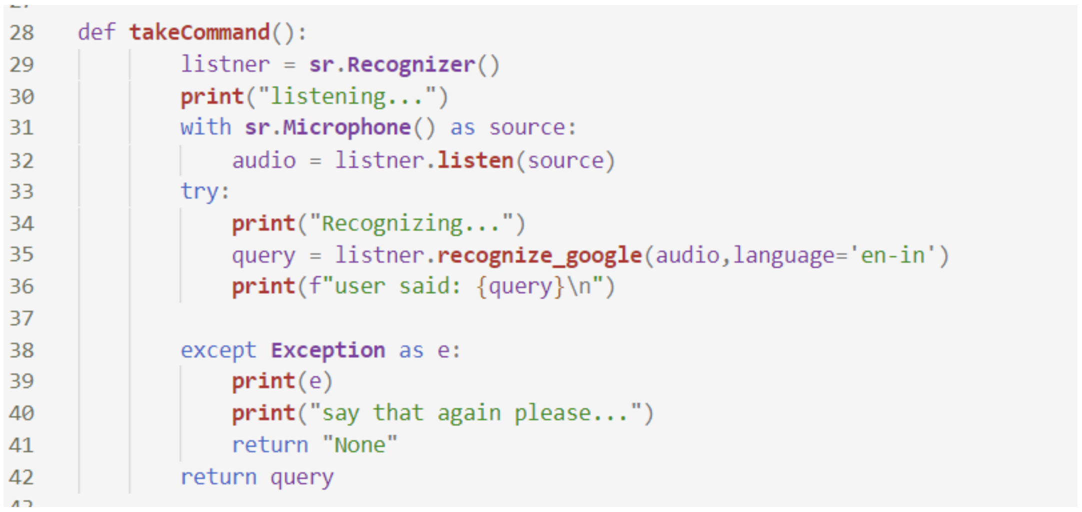
```python
def wishMe():
    hour=int(datetime.datetime.now().hour) #returns value b/w 0-24
    if hour>=0 and hour<12:
        speak("Good Morning Anushka Ma'am")
    elif hour>=12 and hour<16:
        speak("Good Afternoon Anushka Ma'am")
    else :
        speak("Good Evening Anushka Ma'am")

    speak("i am emma, how may i help you")

```

To use this function we first imported datetime module using **import datetime** statement. One of the class defined in the datetime module is datetime class we then used now() method to create a datetime object containing the current local date and time and then typecasted it to integer.
After that we simply print the messages according to the time on the computer.

DEFINING TAKECOMMAND FUNCTION:



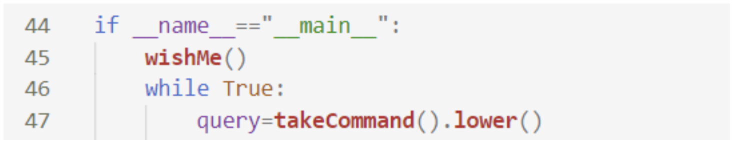
```python
def takeCommand():
        listner = sr.Recognizer()
        print("listening...")
        with sr.Microphone() as source:
            audio = listner.listen(source)
        try:
            print("Recognizing...")
            query = listner.recognize_google(audio,language='en-in')
            print(f"user said: {query}\n")

        except Exception as e:
            print(e)
            print("say that again please...")
            return "None"
        return query
```

With the help of this function our AI assistant will be able to return a string output by taking microphone input from us. Before defining the takecommand() function we need to install a module called speechRecognition. After successfully installing this module, import that module so that our AI assistant should be able to take commands from the user.

At last we will add try and except block to our program to handle errors effectively.

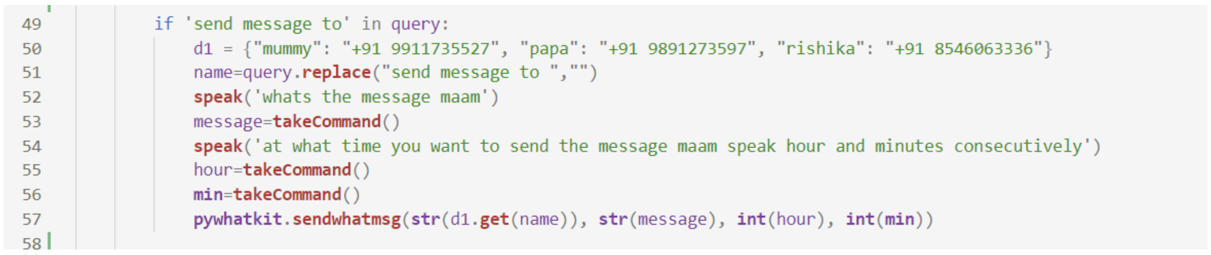
CREATING OUR MAIN FUNCTION:



```python
if __name__=="__main__":
    wishMe()
    while True:
        query=takeCommand().lower()
```

Now we will create a main() function to call our wishme() function and then we will run a while loop till the user says thank you to the assistant.

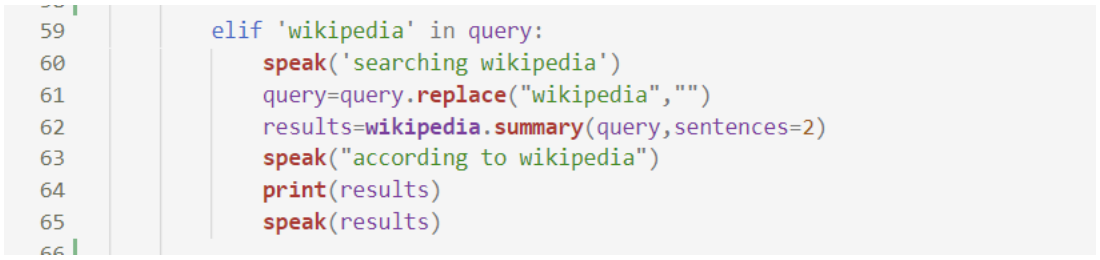
TO SEND AUTOMATED MESSAGES ON WHATSAPP



```python
        if 'send message to' in query:
            d1 = {"mummy": "+91 9911735527", "papa": "+91 9891273597", "rishika": "+91 8546063336"}
            name=query.replace("send message to ","")
            speak('whats the message maam')
            message=takeCommand()
            speak('at what time you want to send the message maam speak hour and minutes consecutively')
            hour=takeCommand()
            min=takeCommand()
            pywhatkit.sendwhatmsg(str(d1.get(name)), str(message), int(hour), int(min))
```

To send automated messages on whatsapp we first need to install pywhatkit into our program. In the above code, we have used an if statement to check whether “send message” is in the query of the user or not. If “send message” is found in the user’s query, then first it will create a dictionary named d1 then replace the string “send message to” to a blank string so that we can get the name of the person to which the user want to send the message to.
Then simply it takes the input message and then asks at what time the user wants to send the message and then invoke the sendwhatmsg function for sending automated messages on whatsapp.

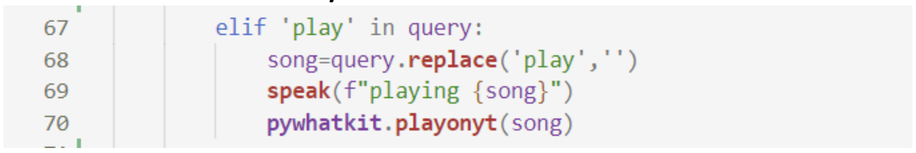
TO SEARCH SOMETHING ON WIKIPEDIA



```python
        elif 'wikipedia' in query:
            speak('searching wikipedia')
            query=query.replace("wikipedia","")
            results=wikipedia.summary(query,sentences=2)
            speak("according to wikipedia")
            print(results)
            speak(results)
```

To do Wikipedia searches, we need to install and import the Wikipedia module into our program. If Wikipedia is in the search query of the user then it speaks searching Wikipedia then it replace the word Wikipedia to a blank string to get the topic to search on Wikipedia then finally it prints and speaks the result in the form of summary.

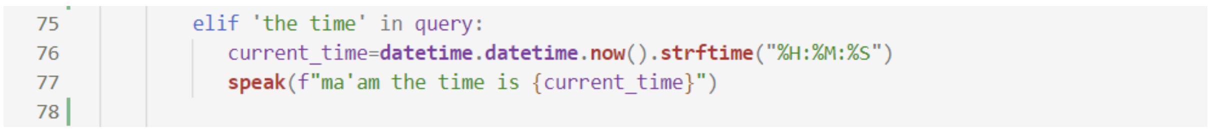
TO PLAY YOUTUBE VIDEOS/SONGS:



```python
        elif 'play' in query:
            song=query.replace('play','')
            speak(f"playing {song}")
            pywhatkit.playonyt(song)
```

To play youtube videos and songs we need to install and import pywhatkit module into our program. If play is in the query then it fetches the song by replacing the word play from user's query then at last uses the function playonyt to play youtube videos.

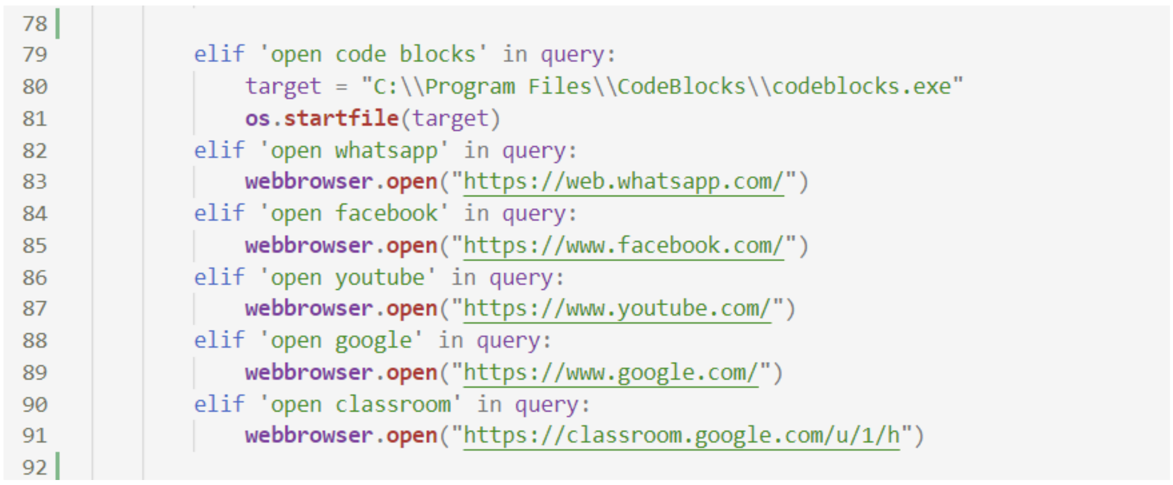
TO KNOW THE CURRENT TIME:



```python
        elif 'the time' in query:
            current_time=datetime.datetime.now().strftime("%H:%M:%S")
            speak(f"ma'am the time is {current_time}")
```

To know the current time we need to install and import datetime module into our program and then store the current time returned by the datetime function in current time and then simply make the assistant speak the current time.

TO OPEN APPLICATIONS:

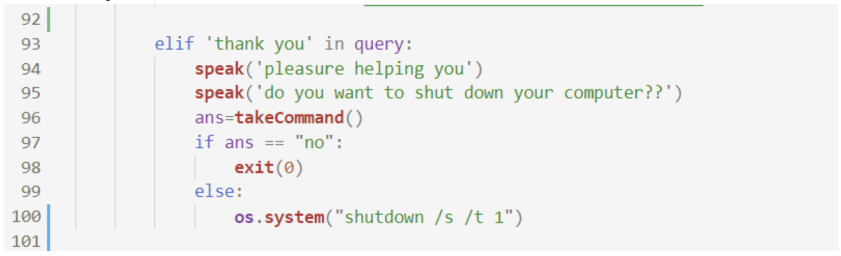


```python
        elif 'open code blocks' in query:
            target = "C:\\Program Files\\CodeBlocks\\codeblocks.exe"
            os.startfile(target)
        elif 'open whatsapp' in query:
            webbrowser.open("https://web.whatsapp.com/")
        elif 'open facebook' in query:
            webbrowser.open("https://www.facebook.com/")
        elif 'open youtube' in query:
            webbrowser.open("https://www.youtube.com/")
        elif 'open google' in query:
            webbrowser.open("https://www.google.com/")
        elif 'open classroom' in query:
            webbrowser.open("https://classroom.google.com/u/1/h")
```

To open various application we first need to import the os module then pass the path of that application in the argument of the startfile function.

To open various websites on the internet we need to first install and import the webbrowser module and then pass the URL of that website in the open function present in webbrowser module.

TO QUIT/SHUT DOWN THE SYSTEM:



```
        elif 'thank you' in query:
            speak('pleasure helping you')
            speak('do you want to shut down your computer??')
            ans=takeCommand()
            if ans == "no":
                exit(0)
            else:
                os.system("shutdown /s /t 1")
```

In this we check if thank you is present in the query of the user. If thank you is present in the query the assistant will speak pleasure helping you then ask if the user wants to shut down the system or not.
If the user says no then the program gets terminated if the user says yes then the systems shuts down with the help of system function present in the os module.

PROJECT WORKING:
https://drive.google.com/file/d/1Ez1N-IKvzbmboevnOfvr9q56Z-IkPxW9/view?usp=sharing

CHALLENGES FACED:

Throughout the making of the project I faced many challenges like I was facing an error installing pyaudio module as I was not having the required build tools to build pyaudio module. So to fix that I first installed pipwin module by writing the command
$ pip install pipwin
on command prompt then installed pyaudio through that module by writing
$ pipwin install pyaudio
Secondly while writing the function for sending automated whatsapp messages, I faced the problem while taking the time as input from the user. The sendwhatmsg function of pywhatkit modules takes the first two arguments as string and last two as integers.

I wanted that the user should give command to the assistant when to send message to the specific contact without pre defining the inputs in hour and minute fields. So I invoked takeCommand() Function to take voice input from the user. But by doing so the problem arises is that the takeCommand function returns a dictionary with the key 'alternative' that points to a list of possible transcripts. So to solve this I had typecast this dictionary to a string and then passed that to the argument in the sendwhatmsg function. I also created a dictionary of some of my family members to send the messages to so that I do not have to predefine their numbers in the argument list of the sendwhatmsg function.

CONCLUSION:

Through this voice assistant, it had automated various services using a single voice command. It eases most of the tasks of the user like sending automated messages through our single voice command, search Wikipedia, play music etc. The main objective was to make this project a complete server assistant and make it smart enough to act as a replacement for a general server administration. The future plans include to use framework like AngularJS to provide a better interactive interface and use backend stack like machine learning to improve its performance and user system interaction. The functionality would be seamless enough to replace the Server Administration with EMMA.