Jarvis_task

```
import datetime
import speak as s
import takingcommand as t
import wikipedia
import webbrowser
import pywhatkit
import os
import pyjokes
import third as ts
import threading
import clap as cl
import mains as ms


def main(master):
    ts.stop_threads = False

    t10 = threading.Thread(target=ts.run)
    t10.start()

    while (True):

        try:
            s.speak("  give me any command sir ")
            query = t.takeCommand()
            if ('wikipedia'in query  or "who" in query):
                s.speak('Okay ,  i found something , it says ')
                if 'wikipedia'in query:
                    query = query.replace('wikipedia', '')
                elif 'something' in query:
                    query = query.replace('something','')
```

Jarvis_task

```
    elif 'who' in query:
        query = query.replace('who','')
        results = wikipedia.summary(query, sentences=2)
        s.speak(results)
       print("query="+query)
   elif 'open youtube' in query.lower():
    s.speak('working on it , opening youtube ')
    webbrowser.open('youtube.com')
    elif 'open google' in query.lower():
    webbrowser.open('google.com')

elif 'open fitgirl repack' in query.lower():
    webbrowser.open('fitgirl-repacks.site')
elif 'open gmail' in query.lower():
    webbrowser.open('gmail.com')
elif 'play song' in query.lower():
    song = query.replace('play song', '')
    pywhatkit.playonyt(song)
elif "time" in query.lower():
    strtimes = datetime.datetime.now().strftime("%H:%M:%S")
    s.speak(f"{master} the time is {strtimes}")
elif 'open spotify' in query.lower():

  os.startfile('C:\\Users\\rnmod\\AppData\\Roaming\\Spotify\\Spotify.exe')

elif 'joke' in query.lower():
  my_joke=pyjokes.get_joke('en',category='neutral')
  s.speak(my_joke)
elif 'create ' in query.lower():
    s.speak('Okay working on it')
  s.speak('Tell me the appropriate file name which you want me to create')
  fname = t.takeCommand()
  s.speak('Okay. what you want me to write inside it')
```

```
data = t.takeCommand()
        f = open(fname, 'wt')
        f.write(data)
        s.speak("file successfully created")
        os.startfile(fname)
  elif 'read ' in query.lower():
     while True:
        try:
    s.speak('can u tell me the name of the file, which you want me to read it for you ')
             fname = t.takeCommand()
               f = open(fname, 'rt')
               s.speak(f.readlines())
               break
           except:
               s.speak(f"{master} im sorry to say, file with that name, does not exist")
               # s.speak("file does not exist")
               s.speak("you want to create the file with that name  or you want to exit")
               fi=t.takeCommand()
               if "create" in fi:
                   # s.speak('Tell me the appropriate file name which you want me to create')
                   # fname = t.takeCommand()

                   s.speak('Okay. what you want me to write inside it')
                   data = t.takeCommand()
                   f = open(fname, 'wt')
                   f.write(data)
                   s.speak("file successfully created")
                   break

               else:
                   break
```

Jarvis_task

```
    elif 'rest' in query.lower() or 'exit' in query.lower():

        s.speak("it was pleasure,   talking to you...  Sir ")
        ts.w.state(newstate='iconic')
        ts.stop_threads = True
        ms.stop_mains = False
        cl.stop_clap = False

        t10.join()
break

    elif 'screenshot' in query.lower():
        s.speak('by what name you want to save that screenshot')
        sc_name = t.takeCommand()
        s.speak('taking Screen shot')
        pywhatkit.take_screenshot(sc_name, 2, True)

    elif 'email' in query.lower():
        while True:
            try:
                s.speak('can you tell me his email address')
                receiver_email=t.takeCommand()
                receiver_email=receiver_email.replace('at the rate','@')
                receiver_email=receiver_email.replace(" ","")
                print(receiver_email)
                s.speak('check you email address on console')
                s.speak('tell me to yes to continue, or no to repeat you email address')
                c=t.takeCommand()
                if 'yes' in c:
                    s.speak('what you want to write inside subject')
```

This code is written in python language and it is used to perform various tasks using voice commands given by the user. It consists of a main() function which is the starting point of execution. It has threads and functions imported from other files. The main() function has an infinite loop which runs and takes voice commands from the user with the help of takeCommand() function. If the voice command contains certain keywords then the code will execute tasks accordingly. For example, if the voice command contains 'Wikipedia' then the code will search for the query on Wikipedia and will return the results. Similarly, if the voice command contains 'open Youtube' then the code will open the Youtube website. If the voice command contains 'time' then the code will return the current time. It also contains code to create, read, take screenshot, send email and search a query on the internet. The code also has a break condition which will break the loop and end the code, if the user says 'rest' or 'exit'.

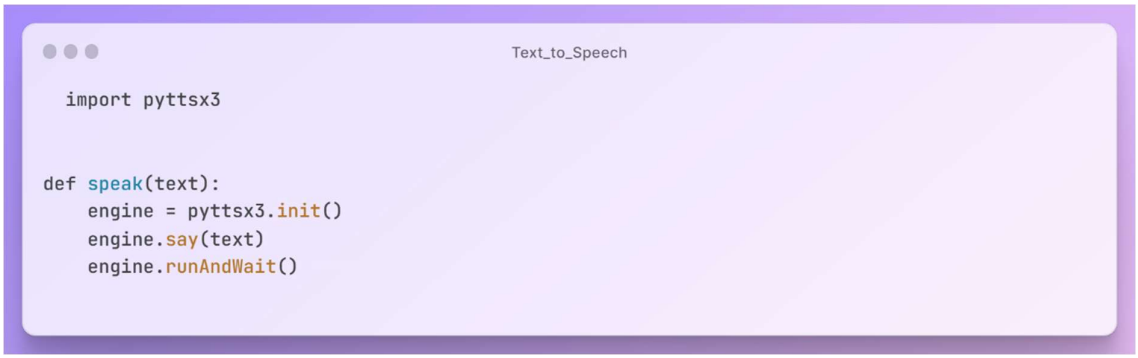
Clap_detect_code

```
import speak as s
import login_register as lr
import sounddevice as sd
import numpy as np
import  threading
import pyttsx3
import mains as ms
import main as m
import wishme as w
import third as ts
global stop_clap



def speak(text):
    engine = pyttsx3.init()
    engine.say(text)
    engine.runAndWait()




def detect_clap():
    while True:
        try:
            duration = 1.5  # seconds
            fs = 44100
            myrecording = sd.rec(int(duration * fs), samplerate=fs, channels=2)
            # print("Recording Audio...")
            sd.wait()
            # print("Audio recording complete , Play Audio")
            # sd.play(myrecording, fs)
            # sd.wait()
            # print("Play Audio Finished")
            audio = myrecording[:,0]
            clap_count = 0
            for i in range(len(audio)):
                if audio[i] > 0.5:
                    clap_count += 1
            if clap_count > 5:
                ms.stop_main = True

                ts.w.state(newstate='normal')
                print("Clap Detected")
                speak('Welcome BAck Sir ')
                s.speak("initiating Authenticating Phase....")
```

Clap_detect_code

```
s.speak(" do you would like to, login... or register")
                master = lr.login_reg()
                print("master=" + master)
                w.wishMe(master)
                s.speak("....sir .... is there anything i can do for you")
                m.main(master)


                t12=threading.Thread(target=ms.body)
                if t12.is_alive()==False:

                    t12.start()




            else:
                print("Clap Not Detected")



            if stop_clap:
                break


        except:
            pass
t5=threading.Thread(target=detect_clap)
```

This code is used to detect a clap. It uses sounddevice library for recording and playing audio. A duration of 1.5 seconds is set for recording audio. If audio amplitude is greater than 0.5, then clap_count is incremented by 1. If clap_count is greater than 5, then a clap is detected. When a clap is detected, it initiates the welcome back message and Authenticating Phase. It then calls login_reg() function to login or register. After that wishMe() and main() functions are called. Finally, a thread is created to start the main body loop. The loop breaks when stop_clap variable is set to true.



This is a Python program which uses pyttsx3 library to convert text into speech. It has a function called 'speak()' which takes a string as an argument and converts it into speech using the pyttsx3 library. This program can be used to create applications that can talk, such as voice-controlled assistants, automated customer service agents, etc. It can also be used to create educational applications like text-to-speech reading applications.

Speech_to_text

```
import speech_recognition as sr


def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("listening...")
        audio = r.listen(source,0,8)
    try:
        print("recognizing...")
        query = r.recognize_google(audio, language='en-in')
        query=query.lower()
        print(f" user spoke: {query}\n")
    except:
        pass
    return query
```

This code takes a speech input from the user, recognizes it using Google Speech Recognition, and prints the recognized text. It then returns the recognized text as a string in lowercase. specifically, it uses the speechrecognition library to take input from a microphone and uses the recognize_google() method to recognize the speech and convert it to text. more specifically, it uses the speechrecognition library to take input from a microphone and uses the recognize_google() method to recognize the speech and convert it to text. After the input is recognized, the code converts the input to lowercase and prints it. Finally, the code returns the recognized text as a string.

Background_audio_video

```
 import playsound as p
from threading import *
from tkinter import *
from tkvideo import tkvideo
from time import *



def run():
    while True:

        p.playsound('jarvis_mu.mp3')

        global stop_threads

        if stop_threads:
            break

class jarvis_intro(Thread):
    def run(self):
        try:
            p.playsound(sound='jarvis.mp3')

        except:
            print("exception called at intro")




class jarvis_video(Thread):
    def run(self):
        try:
            global w
            w = Tk()
            w.title("JARVIS")
            lblVideo = Label(w)
            lblVideo.pack()
            player = tkvideo("Matrix - 100218 _2.mp4",
                             lblVideo,
                             loop=1,
                             size=(400, 300))
            player.play()

            w.mainloop()
            # sleep(2)
        except:
            print("excpetion called at jarv_vdo")




t2=jarvis_video()
t2.start()
sleep(1)
```

This code is to create a Jarvis-like interface in python. Firstly, the 'playsound' module is imported which is used to play sound files and then the 'Threading' and 'tkinter' modules are imported to create a video in the background. A function 'run' is then defined which plays the sound file 'jarvis_mu.mp3' in an infinite loop. Then two classes namely 'jarvis_intro' and 'jarvis_video' are defined which are derived from the 'Thread' class. In the 'jarvis_intro' class, the 'playsound' module is used to play the 'jarvis.mp3' sound file. In the 'jarvis_video' class, a window is created using the 'Tk' class and a label is created which will contain the video. The 'tkvideo' module is then used with parameters such as the file name, label, loop and size to play the video. Finally, the main loop is called and the video is played. Finally, two threads are created and started with the 'jarvis_intro' and 'jarvis_video' classes and the 'sleep' function is called to provide a delay of one second.

Wish_ME

```
import datetime
import speak as s




def wishMe(master):
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        s.speak("good Morning" +master)
    elif hour >= 12 and hour < 18:
        s.speak("Good Afternoon " +master)
    else:
        s.speak("good evening " + master)
```

This code is used to wish a person according to the time of the day. It takes the "master" name as an argument. It then gets the current hour from the datetime library. Depending on the hour, the code will print the appropriate greeting for the person. If the hour is between 0 and 12, it will print "Good Morning" followed by the master's name. If the hour is between 12 and 18, it will print "Good Afternoon" followed by the master's name. Otherwise, it will print "Good Evening" followed by the master's name. The code uses the speak library to print the greeting. Finally, the code returns nothing.

Mysql_Database_connection

```
import mysql.connector
con = mysql.connector.connect(host='localhost', user='root', passwd='', database='test1')
mycur = con.cursor()
```

1. This code connects to a MySQL database.

2. The host name is 'localhost', which means that the database is running on the same computer as the program.

3. The user name is 'root', which is the default user name for a MySQL database.

4. The password is blank, so no password is required to connect to the database.

5. The database name is 'test1'.

6. The connection is stored in a variable called 'con'.

7. A cursor object is created, which allows us to execute SQL commands on the database.

8. The cursor is stored in a variable called 'mycur'.

9. The connection and cursor objects can be used to execute SQL commands on the database.

Login_OR_Register

```
import speak as s
import takingcommand as t
import registration as r
import login as l



def login_reg():
    while True:
        try:
            command=t.takeCommand()
            if 'register' in command:
                s.speak("registration process initiated")
                r.registration()
                s.speak("now u can easily login")
                s.speak("do u want to login or exit")
                while True:
                    y=t.takeCommand()
                    if 'login' in y:
                        i=l.login()
                       # master=i
                        # wishMe()
                       # return i
                        break
                    elif 'exit' in y:
                        s.speak("have a nice day.. sir")
                        break
                    else:
                        s.speak("is it login or exit sir can you please repeat")
                return i
            elif 'login' in command:
                s.speak("initiating login process")
                i=l.login()
                # master=i
                # wishMe()
                # return i
                break
            else:
                s.speak("can you please repeat sir")
                # l.login_reg()
                # break
        except:
            s.speak("Are you there sir...  Can you please tell me to register or login ")
    return i
```

This code is used to login or register a user. It starts by taking a command from the user using the function takeCommand() from takingcommand module. If the command is 'register', it starts the registration process by calling the registration() function from the registration module. After successful registration, it informs the user that he/she can now login and asks the user if he/she wants to login or exit. If the user wants to login, the login() function from the login module is called and the user is logged in. If the command is 'login', the login() function from the login module is called and the user is logged in. If neither of the commands is given, it asks the user to repeat the command. After successful login, it returns the user's details.

Login

```
import speak as s
import takingcommand as t
import database as d


def login():
    x1 = 3
    while (x1 > 0):
        try:
            s.speak("whats the user id")
            luser = t.takeCommand()
            s.speak("whats the password")
            lpassword = t.takeCommand()

            d.mycur.execute(f"select * from xyz where user like '{luser}'&& password
like'{lpassword}'")
            #print(luser, lpassword)
            d.myresult = d.mycur.fetchall()
            #print(d.myresult)
            if (d.myresult != []):
                s.speak("authorization successsfull")

                # wishMe()
                return luser
                break
            if(d.myresult == []):
                # print("empty")
                x1 = x1 - 1
                s.speak(f"Login details Incorrect you got {x1} tries")

            for i in d.myresult:
                i
                break
            if (x1 == 0):
                s.speak("you are unauthorized ")
                break
        except:
            s.speak("speak louder... Sir")
```

This code is used for login authentication and is used to authenticate the user by taking user id and password from the user. It utilizes the database module to search the database for the credentials entered by the user. The code takes 3 attempts from the user to authenticate the credentials. It starts by initializing the variable x1 = 3 which stores the number of attempts. Using while loop, the code keeps running until x1 is greater than 0. It then calls the speak function to ask the user for user id and password and stores it in luser and lpassword variables. Using execute function, it searches the database for the credentials entered by the user. If the credentials are found in the database, the user is authorized and wishMe function is called. Otherwise, x1 is reduced by 1 and the user is asked to re-enter the credentials. The code then checks if the user has attempted the login 3 times and if it is true, the user is declared as unauthorized. Finally, the loop is broken and the user is logged in or out depending on the credentials entered by the user.

Registration

```
import speak as s
import takingcommand as t
import database as d



def registration():
    while True:
        try:
            s.speak("registration name please")
            regname=t.takeCommand()
            s.speak("registration password please")
            regpas=t.takeCommand()
            #print(regname,regpas)
            s.speak("Reconform your  details please")
            s.speak("your name is"+regname)
            s.speak("your password is"+regpas)
            s.speak("please tell yes to conform, no to retry,exit to terminate from here")
            conform=t.takeCommand()
            if(conform=='yes'):
                s.speak("inserting your details in database")
                d.mycur.execute(f"insert into xyz values('{regname}','{regpas}')")
                s.speak("you are successfully registered to use jarvis")
                break


            elif'no' in conform:
                s.speak("Okay.... retrying your registration process..... your")
                #print("no")
                registration()
                break

            elif 'exit' in conform:
                s.speak('okay, terminating registration process')
                break
            else:
                s.speak('looks like you spoke something else ')
                s.speak('i will terminate registration process from here ')
                break
        except:
            s.speak("speak louder Sir...")
            # registration()
```

This code is used to register a user in database. Firstly it will take user's name and password. Then it will ask user to reconform the details and if user says yes, it will insert the details in database by using execute() method of database. Finally user will be registered successfully. In case user says no, it will restart the registration process. If user says exit then it will terminate the registration process. It also contains exception handling in case user doesn't speak loud enough.

Code to wake Jarvis threw voice

```
import speak as s
import takingcommand as t
import wishme as w
import login_register as lr
import threading

import main as m
import third as ts
import clap as c



global stop_main


def body():


    while True:



        try:

            wak=t.takeCommand()

            if "wake" in wak or 'uthe' in wak :
                c.stop_clap = True
                ts.w.state(newstate='normal')
                # t3 = ts.jarvis_intro()
                # t3.start()
                # t3.join()
                master="sir"
                #
                # s.speak("initiating Authenticating Phase....")
                s.speak(" do you would like to, login... or register")
                master = lr.login_reg()
                print("master="+master )
                w.wishMe(master)
                s.speak("....sir .... is there anything i can do for you")
                m.main(master)
                # c.stop_clap = False

                t15 = threading.Thread(target=c.detect_clap)
                t15.start()
                stop_main=False




            if stop_main:
                break


        except:

            print('Jarvis is sleeping........')


t9 = threading.Thread(target=body)
```

This code is used to make a voice assistant(Jarvis) which responds to the commands given by the user. It is using various modules such as speak, takingcommand, wishme, login_register, main, third and clap. A while loop is used to continuously take command from the user and perform the tasks accordingly. If user says "wake" or "uthe" then it will initiate the authentication phase. It will ask the user whether he wants to login or register. After that it will wish the user and then go to main function to perform other tasks. It is also using a threading module to make a separate thread for detecting claps. If a clap is detected then it will break the loop and go back to the main loop. At the end of the loop, it will check the stop_main variable which will be set to true if the user says "stop". This will break the loop and the program will terminate.

Main file to run all Task of jarvis

```
import clap as c
import threading
import  mains as ms









while True:
    if(c.t5.is_alive()==False):
        c.t5.start()
    if(ms.t9.is_alive()==False):
        ms.t9.start()
    if(c.t5.is_alive()==True):
        print('t5==', c.t5.is_alive())
        c.t5.join()
    if(ms.t9.is_alive()==True):
        print('t9==', ms.t9.is_alive())
        ms.t9.join()
```

This code is an infinite loop which checks the status of two threads namely t5 and t9 and starts them if they are not alive. The loop runs infinitely and checks if the threads are alive or not and if not, it starts them. The code also prints the status of the threads to notify the user whether the threads are alive or not. Finally, it uses the join() method to ensure that the threads are completed before the infinite loop starts again.

This is the code which will create,execute,run,break all loops,function,audio,video,which means this is the main file outof all file created, without this file you can’t run this Jarvis Program created in python. Using pyinstaller you only need to create exe file of above code only which will run the whole program .