Vivekanand Vidya Vihar

Session 2019-20

Computer Science (283)

Project On Desktop Assistent

Submitted by
Kanha Tomar

Submitted On
10 november 2019

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher mr. Anshul Atre as well as our principal mrs. Rashmi Kaushal Baveja who gave me the golden opportunity to do this wonderful project on the topic Desktop Assistent, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them. Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

CERTIFICATE

This is to certify that Kanha Tomar of class XII has successfully completed this Computer Science project on the topic "DESKTOP ASSISTENT" prescribed by Mr. Anshul Atre Sir, during academic session 2019-2020 as per the guidelines issues by Central Board of Secondary Education.

Mr. Anshul Atre
(P.G.T, computer)

External
Examiner

Index

Introduction

Desktop Assistent is a program which can do our basics tasks through our voice commands. It has been written in Python 3.7 which has lots of modules to make our work easy. I didn’t work on AI, it is just an simple program which uses other’s AI systems like the Wolfram Alpha and Google. I did’t use any framework, i wrote my code on Vim Text Editor which is mainly distributed with UNIX systems and run it with the help of inbuilt Python through Terminal.
It can do such type of task for a user:-

1. Open Youtube.
2. Open Whatsapp.
3. Open Facebook.
4. Open Gmail.
5. Play Songs.
6. Narrate the Wikipedia.
7. It can tell you the Weather Forecast.
8. Tell you the Jokes.
9. Solve the basic Maths problems.
10. Answer the G.K Questions.
11. Send Emails over your voice command.
12. Browse to any site.

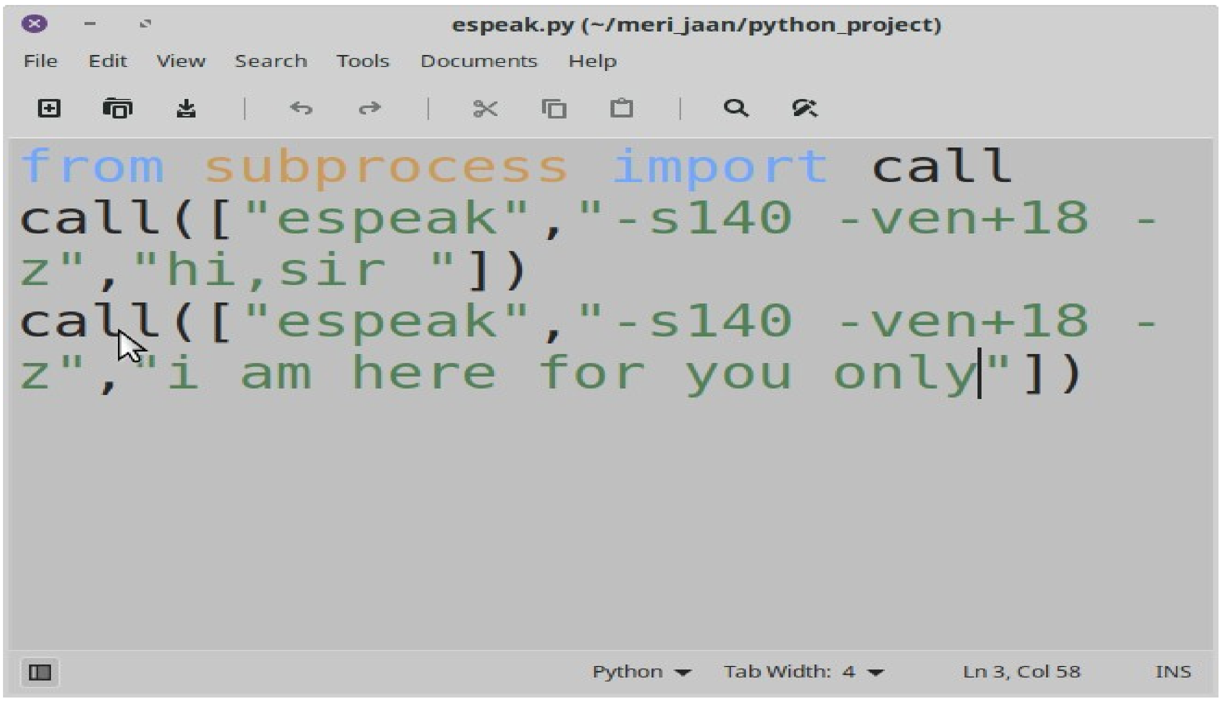
PIP(preferred installer program)

when we talk about what make python easy so may be pip is there. pip can installs all the modules which python have just by the command "pip install name of the module".
python comes with two version 2.7 or any in that series and 3.7 or updated versions, so for installing modules for a perticular version of Python there is speacified syntex that we have to use.
Syntex for installing modules in python2 is only pip while for python3 we just have to write pip3.
We are using python 3.7 which is already installed in the system, so we use the pip3
 We can install it by the command "sudo apt get install pip3"

About The Modules

- **Espeak:** It is an open source library which is used to make our program speakable. We can directly install it from the terminal through the command "sudo apt get install espeak", here "pip3 install espeak" isn't work because it is not specified for Python.
  It can directly run on the terminal with the help of commands because it is an terminal based software.

  For make it working with Pyhton the code is as follow


espeak.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 3, Col 58 INS

```
from subprocess import call
call(["espeak","-s140 -ven+18 -z","hi,sir "])
call(["espeak","-s140 -ven+18 -z","i am here for you only"])
```

- **Speech Recognition:** It is a Python module which we can use to recognise or we can say that to give command to our program. Most modern speech recognition systems rely on what is known as a *Hidden Markov Model* (HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process—that is, a process in which statistical properties do not change over time.

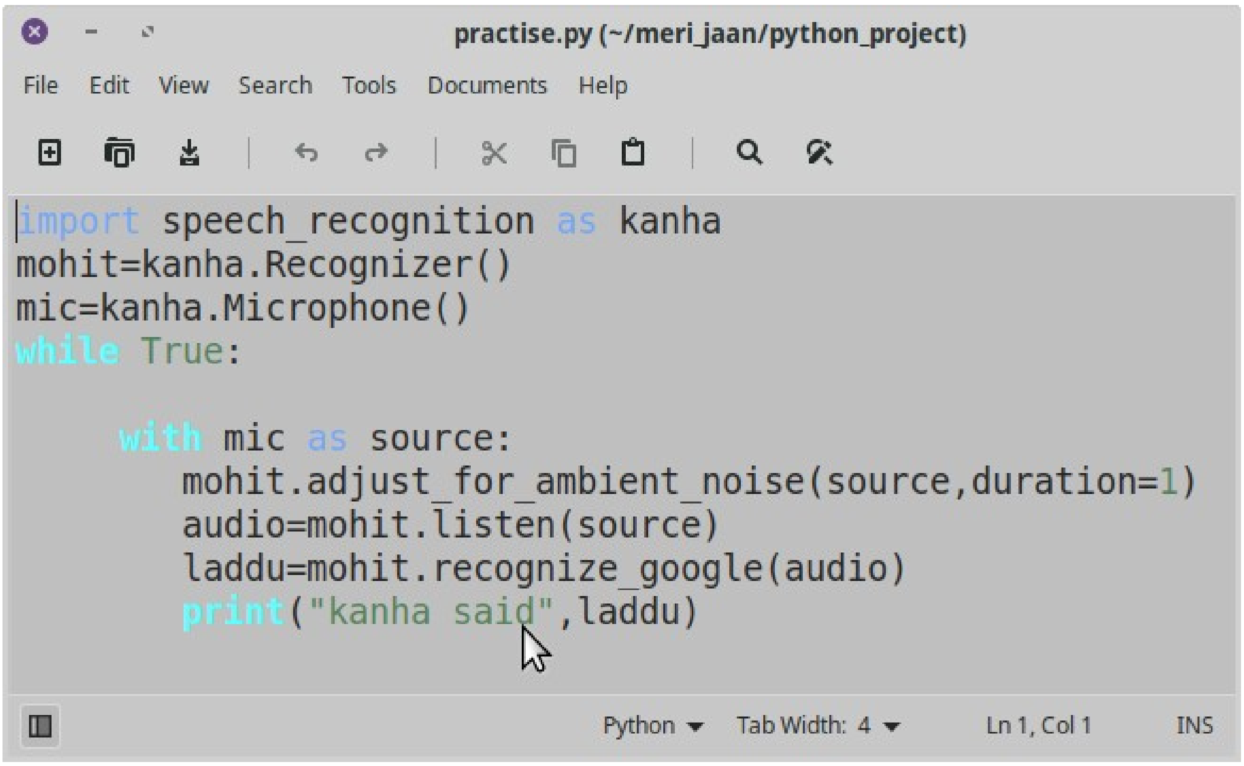
  In a typical HMM, the speech signal is divided into 10-millisecond fragments. The power spectrum of each fragment, which is essentially a plot of the signal's power as a function of frequency, is mapped to a vector of real numbers known as cepstral coefficients. The dimension of this vector is usually small—sometimes as low as 10, although more accurate systems may have dimension 32 or more. The final output of the HMM is a sequence of these vectors.

  To decode the speech into text, groups of vectors are matched to one or more phonemesa fundamental unit of speech. This calculation requires training, since the sound of a pho_witneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.

  One can imagine that this whole process may be computationally expensive. In many modern speech

recognition systems, neural networks are used to simplify the speech signal using techniques for feature transformation and dimensionality reduction *before* HMM recognition. _witVoice activity detectors (VADs) are also used to reduce an audio signal to only the portions that are likely to contain speech. This prevents the recognizer from wasting time analyzing unnecessary parts of the signal.
we can install it through the command "pip3 install Speech Recognition".

<u>Here is the trial code</u>


practise.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 1, Col 1 INS

```
import speech_recognition as kanha
mohit=kanha.Recognizer()
mic=kanha.Microphone()
while True:

     with mic as source:
        mohit.adjust_for_ambient_noise(source,duration=1)
        audio=mohit.listen(source)
        laddu=mohit.recognize_google(audio)
        print("kanha said",laddu)
```

There are many recognizer are given by the module we just change the code little bit like in line no. 8 we have to change _google to _google_cloud,_houndify,_ibm _sphinx and _wit.

We can also check the help section of module by typing following coomands in terminal

```
python3 # to python in terminal
>>import speech_recognition as sr
>>help(sr)
```

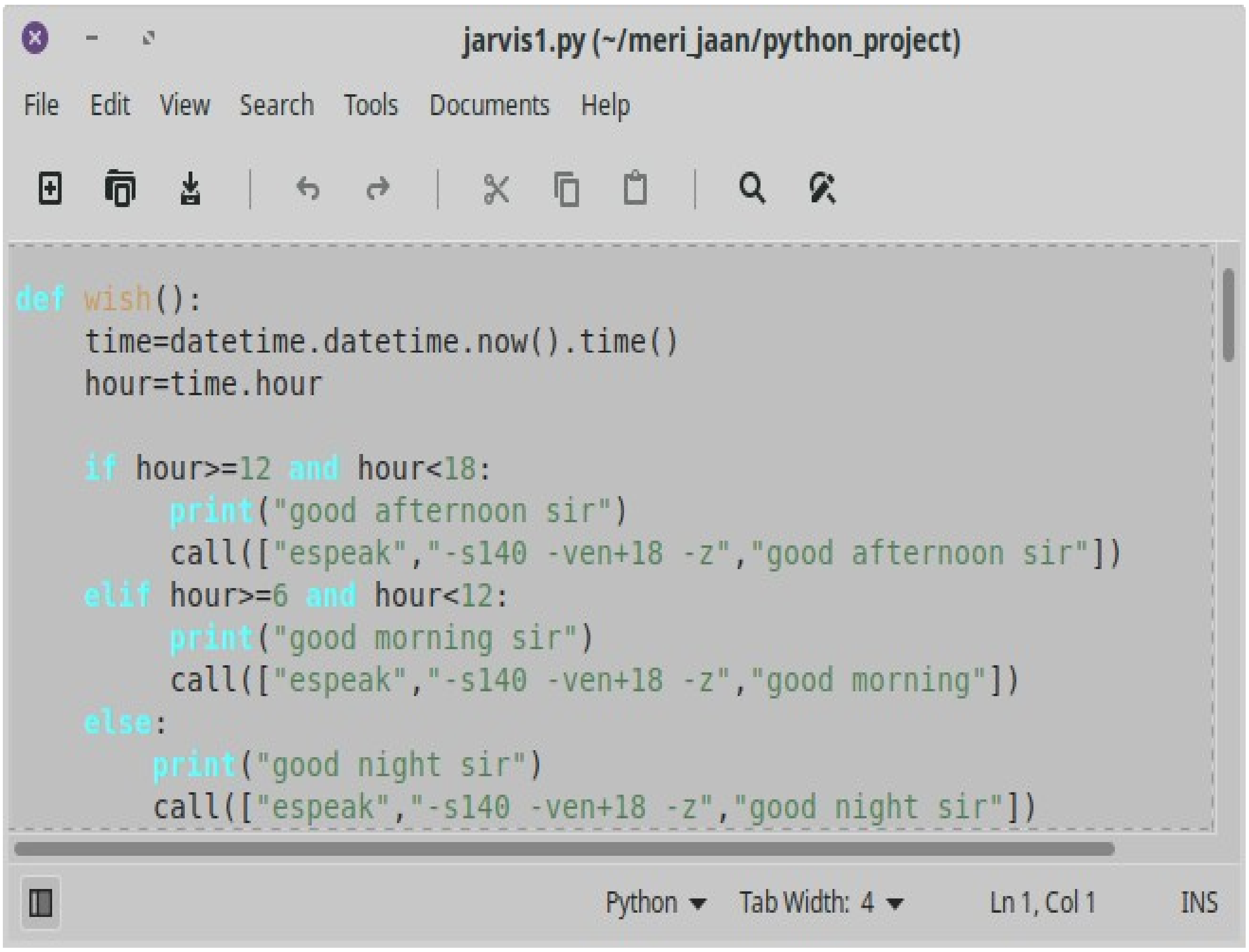
This help give us a good understanding of the module.

- **Date time:** In Python, date, time and datetime classes provides a number of function to deal with dates, times and time intervals. Date and datetime are an object in Python, so when we manipulate them, we are actually manipulating objects and not string or timestamps. Whenever we manipulate dates or time, we need to import datetime function.

  The datetime classes in Python are categorized into main 5 classes.

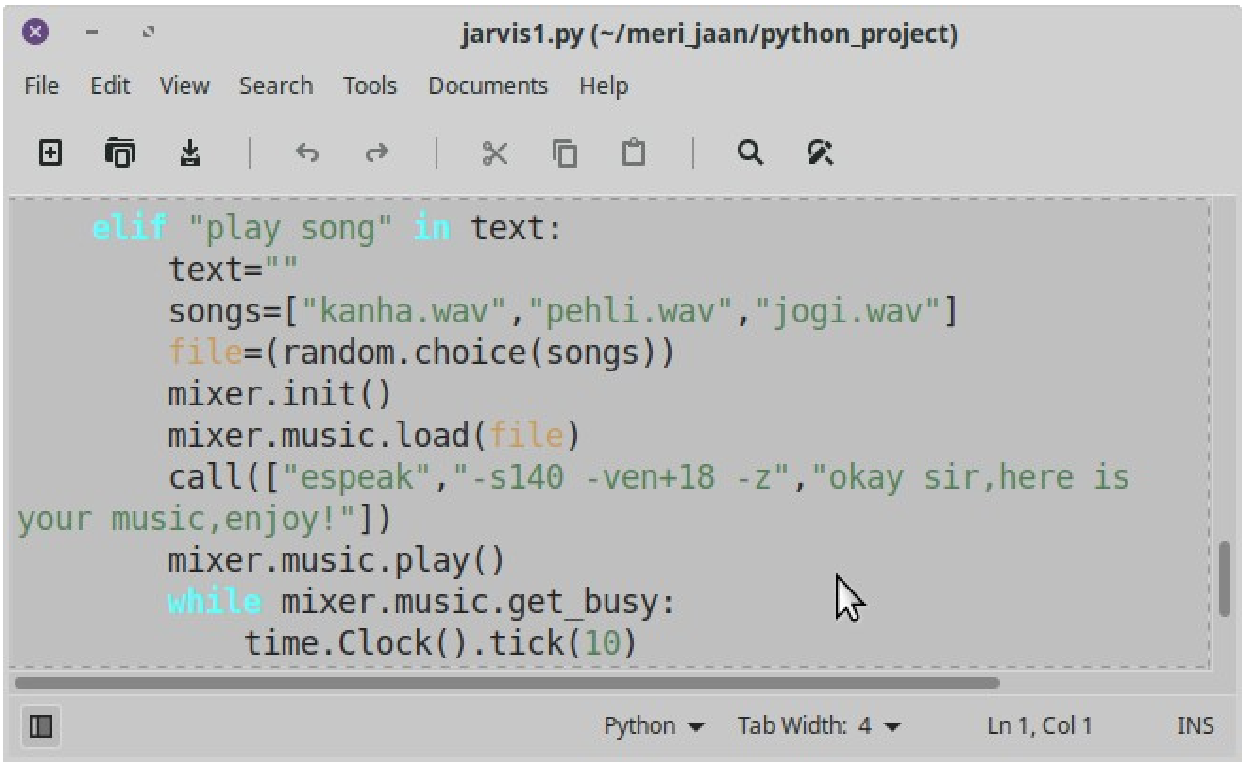
  date – Manipulate just date (Month, day, year)
  time – Time independent of the day (Hour, minute, second, microsecond)
  datetime - Combination of time and date (Month, day, year, hour, second, microsecond)
  timedelta - A duration of time used for manipulating dates
  tzinfo - An abstract class for dealing with time zones

Here we wants that at he starting jarvis will wish us that'swhy we only need time as a output from the datetime module and then send it to the espeak to tell us.


jarvis1.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 1, Col 1 INS

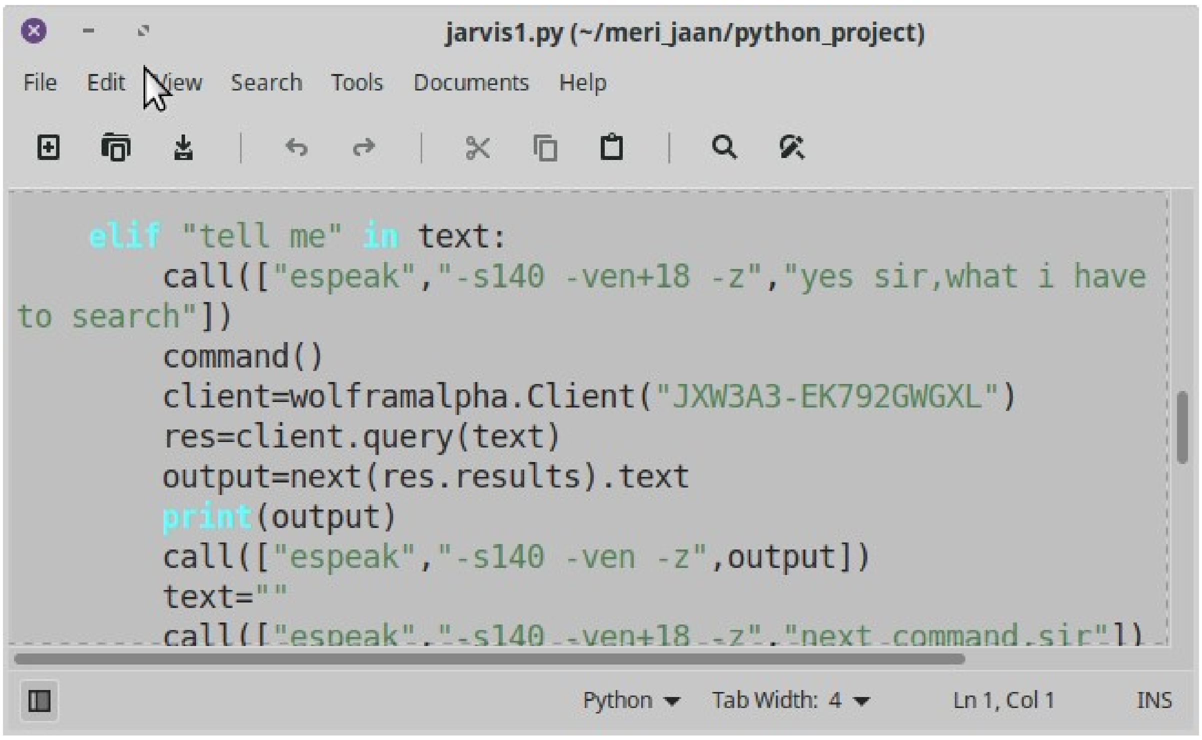
```
def wish():
    time=datetime.datetime.now().time()
    hour=time.hour

    if hour>=12 and hour<18:
         print("good afternoon sir")
         call(["espeak","-s140 -ven+18 -z","good afternoon sir"])
    elif hour>=6 and hour<12:
         print("good morning sir")
         call(["espeak","-s140 -ven+18 -z","good morning"])
    else:
        print("good night sir")
        call(["espeak","-s140 -ven+18 -z","good night sir"])
```

- **Pygame:** It is used for building games but here we use it to play songs.
  This module also uses the random module to pick up a random song to play. We play the songs which are only in wav format other format files are not allowed. We have to put the songs file in the same directory from where our program is running.
  We also need to install pyaudio for better working sometimes it's push an error.
  So for listning the songs by just a command we have to write this commands in terminal to install all required repository.
  "pip3 install pygame" or in linux we can install it from the software manager.
  "pip3 install pyaudio"


jarvis1.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 1, Col 1 INS

```
    elif "play song" in text:
        text=""
        songs=["kanha.wav","pehli.wav","jogi.wav"]
        file=(random.choice(songs))
        mixer.init()
        mixer.music.load(file)
        call(["espeak","-s140 -ven+18 -z","okay sir,here is 
your music,enjoy!"])
        mixer.music.play()
        while mixer.music.get_busy:
            time.Clock().tick(10)
```

- **Wolfram Alpha:**Wolfram Alpha is a unique engine for computing answers and providing knowledge. It works by using its vast store of expert-level knowledge and algorithms to automatically answer questions, do analysis and generate reports. First of all we have to sign in to wolfram alpha
  then we have to generate a key as a student and key we have to use in our code to recognise ourself.
  We can do basic maths calculation with this api.
  Get answers of the general knowledge questions.
  Check weather condition.
  Tell us the jokes.
  We can install it through the pip like this pip3 install wolframalpha.
  The code is like this


jarvis1.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 1, Col 1 INS

```
    elif "tell me" in text:
        call(["espeak","-s140 -ven+18 -z","yes sir,what i have
to search"])
        command()
        client=wolframalpha.Client("JXW3A3-EK792GWGXL")
        res=client.query(text)
        output=next(res.results).text
        print(output)
        call(["espeak","-s140 -ven -z",output])
        text=""
        call(["espeak","-s140 -ven+18 -z","next command sir"])
```

- **Wikipedia:** In order to extract data from Wikipedia, we must first install the Python Wikipedia library, which wraps the official Wikipedia API. This can be done by entering the command below in your command prompt or terminal:

```
pip install wikipedia
```

Once the installation is done, we can use the Wikipedia API in Python to extract information from Wikipedia. In order to call the methods of the Wikipedia module in Python, we need to import it using the following command.

```
import wikipedia
```

Searching Titles and Suggestions:

The search() method does a Wikipedia search for a query that is supplied as an argument to it. As a result, this method returns a list of all the article's titles that contain the query. For example:

```
import wikipedia
print(wikipedia.search("Bill"))
```

Output:

```
['Bill', 'The Bill', 'Bill Nye', 'Bill Gates', 'Bills, Bills, Bills',
'Heartbeat bill', 'Bill Clinton', 'Buffalo Bill', 'Bill & Ted',
'Kill Bill: Volume 1']
```

As you see in the output, the searched title along with the related search suggestions are displayed. You can configure the number of search titles returned by passing a value for the results parameter, as shown here:

```
import wikipedia
print(wikipedia.search("Bill", results=2))
```
Output:
```
['Bill', 'The Bill']
```
The above code prints only 2 search results of the query since that is how many we requested to be returned.

Let's say we need to get the Wikipedia search suggestions for a search title, "Bill Cliton" that is incorrectly entered or has a typo. The suggest() method returns suggestions related to the search query entered as a parameter to it, or it will return "None" if no suggestions were found.

Let's try it out here:

```
import wikipedia
print(wikipedia.suggest("Bill cliton"))
```

Output:
```
bill clinton
```

You can see that it took our incorrect entry, "Bill cliton", and returned the correct suggestion of "bill clinton".

Extracting Wikipedia Article Summary

We can extract the summary of a Wikipedia article using the summary() method. The article for which the summary needs to be extracted is passed as a parameter to this method.

Let's extract the summary for "Ubuntu":

```
print(wikipedia.summary("Ubuntu"))
```

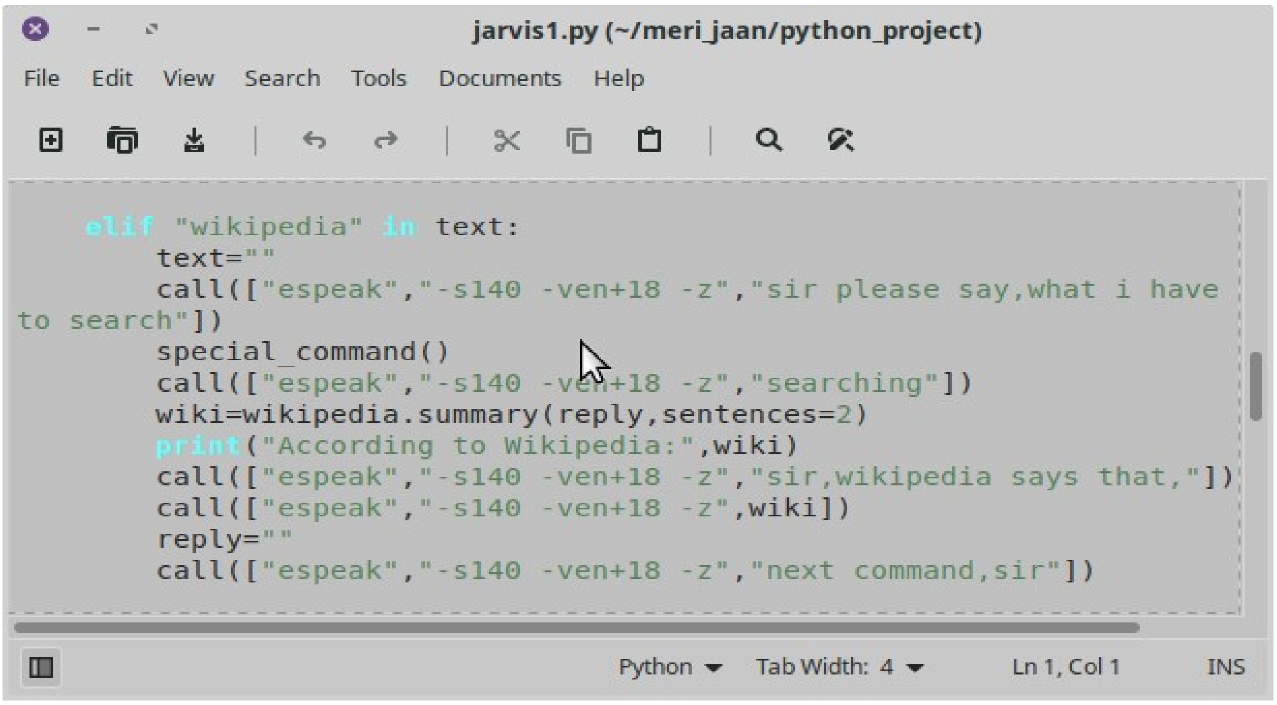
Output:

```
Ubuntu ( (listen)) is a free and open-source Linux
distribution based on Debian. Ubuntu is officially released
in three editions: Desktop, Server, and Core (for the internet
of things devices and robots). Ubuntu is a popular operating
system for cloud computing, with support for
OpenStack.Ubuntu is released every six months, with long-
term support (LTS) releases every two years. The latest
release is 19.04 ("Disco Dingo"), and the most recent long-
term support release is 18.04 LTS ("Bionic Beaver"), which
is supported until 2028. Ubuntu is developed by Canonical
and the community under a meritocratic governance model.
Canonical provides security updates and support for each
Ubuntu release, starting from the release date and until the
release reaches its designated end-of-life (EOL) date.
Canonical generates revenue through the sale of premium
services related to Ubuntu. Ubuntu is named after the
African philosophy of Ubuntu, which Canonical translates
as "humanity to others" or "I am what I am because of who
we all are".
```

The whole summary is printed in the output. We can customize the number of sentences in the summary text to be displayed by configuring the sentences argument of the method.

```
print(wikipedia.summary("Ubuntu", sentences=2))
```

Output:
Ubuntu ((listen)) is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: Desktop, Server, and Core (for the internet of things devices and robots).
As you can see, only 2 sentences of Ubuntu's text summary is printed.
However, keep in mind that wikipedia.summary will raise a "disambiguation error" if the page does not exist or the page is disambiguous.
Here in our project we use only summary class to check the wikipedia. and the output which returns from the module we take that result in a variable and pass it to espeak module so it can tell us the wikipedia.


jarvis1.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 1, Col 1 INS

```
    elif "wikipedia" in text:
        text=""
        call(["espeak","-s140 -ven+18 -z","sir please say,what i have
to search"])
        special_command()
        call(["espeak","-s140 -ven+18 -z","searching"])
        wiki=wikipedia.summary(reply,sentences=2)
        print("According to Wikipedia:",wiki)
        call(["espeak","-s140 -ven+18 -z","sir,wikipedia says that,"])
        call(["espeak","-s140 -ven+18 -z",wiki])
        reply=""
        call(["espeak","-s140 -ven+18 -z","next command,sir"])
```

- **Webbrowser:** The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.
  Under Unix, graphical browsers are preferred under X11, but text-mode browsers will be used if graphical browsers are not available or an X11 display isn’t available. If text-mode browsers are used, the calling process will block until the user exits the browser.

  If the environment variable BROWSER exists, it is interpreted as the os.pathsep-separated list of browsers to try ahead of the platform defaults. When the value of a list part contains the string %s, then it is interpreted as a literal browser command line to be used with the argument URL substituted for %s; if the part does not contain %s, it is simply interpreted as the name of the browser to launch.

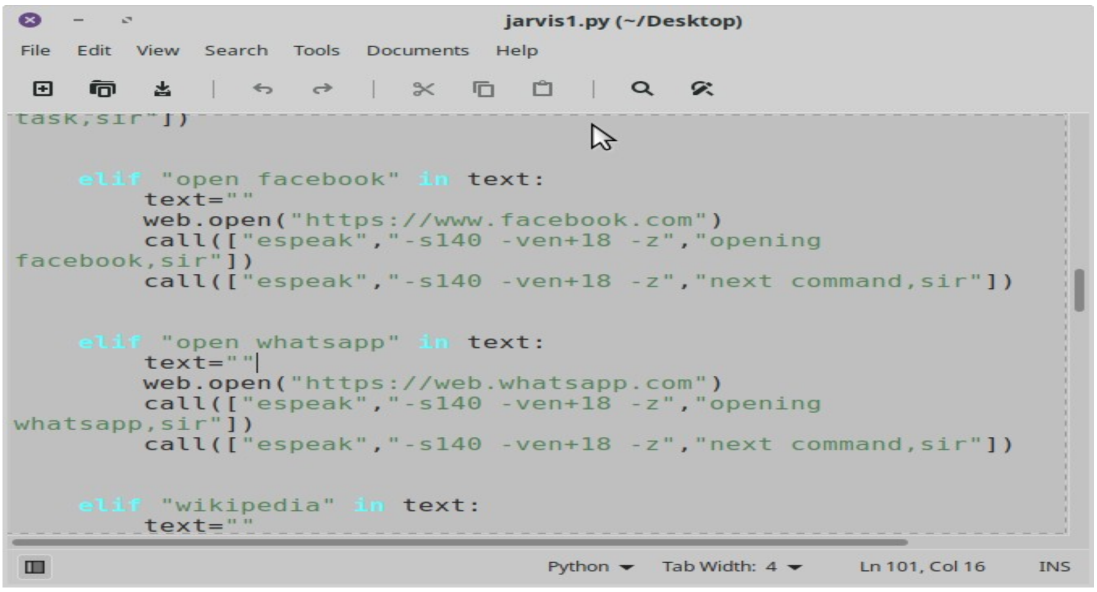
  For non-Unix platforms, or when a remote browser is available on Unix, the controlling process will not wait for the user to finish with the browser, but allow the remote browser to maintain its own windows on the display. If remote browsers are not available on Unix, the controlling process will launch a new browser and wait.

webbrowser.open(url, new=0, autoraise=True)

Display url using the default browser. If new is 0, the url is opened in the same browser window if possible. If new is 1, a new browser window is opened if possible. If new is 2, a new browser page (“tab”) is opened if possible. If autoraise is True, the window is raised if possible (note that under many window managers this will occur regardless of the setting of this variable).

webbrowser.open_new(url)
Open url in a new window of the default browser, if possible, otherwise, open url in the only browser window.

webbrowser.open_new_tab(url)
Open url in a new page (“tab”) of the default browser, if possible, otherwise equivalent to open_new().
<u>The code is like that</u>



```
task,sir"])


    elif "open facebook" in text:
        text=""
        web.open("https://www.facebook.com")
        call(["espeak","-s140 -ven+18 -z","opening
facebook,sir"])
        call(["espeak","-s140 -ven+18 -z","next command,sir"])


    elif "open whatsapp" in text:
        text=""
        web.open("https://web.whatsapp.com")
        call(["espeak","-s140 -ven+18 -z","opening
whatsapp,sir"])
        call(["espeak","-s140 -ven+18 -z","next command,sir"])


    elif "wikipedia" in text:
        text=""
```

- **SMTPLIB**: SMTP (Simple Mail Transfer Protocol) is an application-level protocol (on top of TCP) used to communicate with mail servers from external services, like an email client on our phone. SMTP is a delivery protocol only, so we can't actually retrieve email with it, we can only send email.

  Opening the Connection
  As already mentioned, Python conveniently comes with the smtplib, which handles all of the different parts of the protocol, like connecting, authenticating, validation, and of course, sending emails.

  Using this library, there are a few different ways we can create a connection to our mail server. In this section, we'll focus on creating a plain, insecure connection (which should rarely, if ever, be used). This connection is unencrypted and defaults to port 25. However, the protocol for mail submission actually uses 587, which is what we'll use.

  These connections are really simple to create with smtplib. The unencrypted version can be created with:

```
import smtplib

try:
server = smtplib.SMTP('smtp.gmail.com', 587)
server.ehlo()
```

```
except:
print 'Something went wrong...'
```

And that's it. There really isn't much more to it than passing the server address, port, and calling .helo(), which identifies us to the SMTP server. Using this server object we can now send email over an insecure connection.

Using a Secure Connection
When an SMTP connection is secured via TLS/SSL, it is done over port 465 and is typically called SMTPS. Needless to say, we should always use a secured connection.

There are a few different ways we can secure our SMTP connections in the smtplib library. The first way is to first create an insecure connection and then upgrading to TLS. This is done using the .starttls() method.

```
import smtplib

try:
server = smtplib.SMTP('smtp.gmail.com', 587)
server.ehlo()
server.starttls()
# ...send emails
except:
print 'Something went wrong...'
```
Notice that while this is very similar to the previous

insecure connection we created, all that's different is that we're using the .starttls() method to upgrade the connection to secure.

Creating the Email

Emails, at the very core, are just strings of text connected by newline characters. Most emails will at least have the "From", "To", "Subject" and a body fields. Here is a simple example:

```
From: you@gmail.com
To: me@gmail.com,bill@gmail.com
Subject: OMG Super Important Message

Hey, what's up?

- You
```

As we can see, each line contains a new field with its data. No binary protocol, no XML, no JSON, just line-separated strings.

A simple way to parameterize these fields is to use string formatting in Python:

```
sent_from = 'you@gmail.com'
to = ['me@gmail.com', 'bill@gmail.com']
subject = 'OMG Super Important Message'
body = 'Hey, what's up?\n\n- You'

email_text = """\
From: %s
To: %s
Subject: %s

%s
""" % (sent_from, ", ".join(to), subject, body)
```

Authenticating with Gmail

There are a few steps we need to take before we can send emails through Gmail with SMTP, and it has to do with authentication. If we're using Gmail as the provider, we'll need to tell Google to allow we to connect via SMTP, which is considered a "less secure" method.

You can't really blame Google for setting it up this way since our application (or some other 3rd party app) will need to have our plaint-text password for this to work, which is definitely not ideal. It's not like the OAuth protocol where a revocable token is issued, so they have to find another way to ensure no unauthorized parties access our data.

For many other email providers we won't need to do any of the extra steps I describe here.

First, we'll want to allow less secure apps to access we account. For detailed instructions on how to do this, we should check out this page:

Allowing less secure apps to access our account

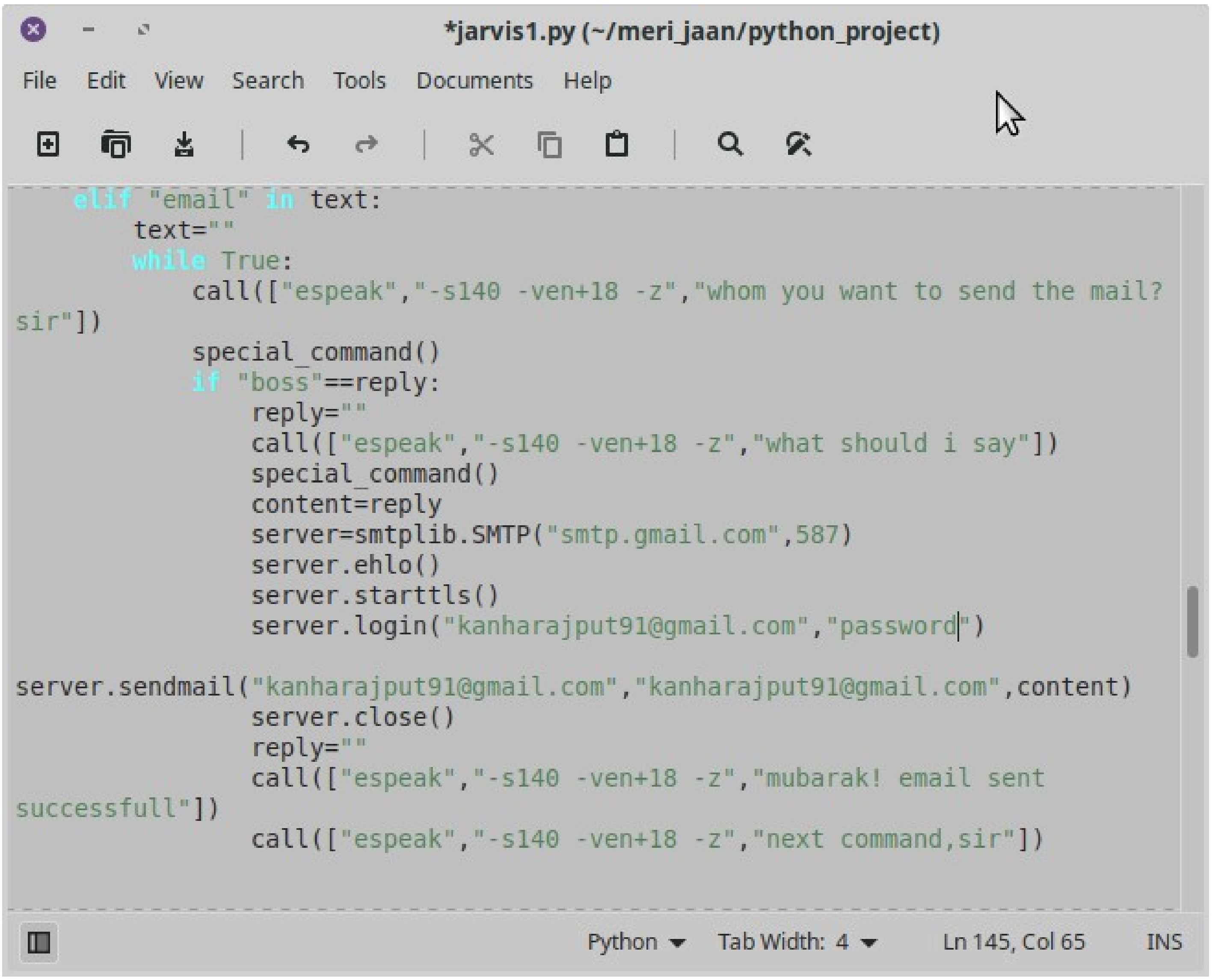
Sending the Email
Now that we have our SMTP connection set up and authorized we app with Google, we can finally use Python to send email with Gmail.

Using the email string we constructed above, and the connected/authenticated server object, we need to call the .sendmail() method. Here is the full code, including the methods for closing the connection:

```
import smtplib

gmail_user = 'you@gmail.com'
gmail_password = 'P@ssword!'

sent_from = gmail_user
to = ['me@gmail.com', 'bill@gmail.com']
subject = 'OMG Super Important Message'
body = 'Hey, what's up?\n\n- You'

email_text = """\
From: %s
To: %s
Subject: %s

%s
""" % (sent_from, ", ".join(to), subject, body)
```

```
try:
server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
server.ehlo()
server.login(gmail_user, gmail_password)
server.sendmail(sent_from, to, email_text)
server.close()

print 'Email sent!'
except:
print 'Something went wrong...'
```

<u>The code is like that</u>


*jarvis1.py (~/meri_jaan/python_project)
File Edit View Search Tools Documents Help
Python Tab Width: 4 Ln 145, Col 65 INS

```
    elif "email" in text:
        text=""
        while True:
            call(["espeak","-s140 -ven+18 -z","whom you want to send the mail?
sir"])
            special_command()
            if "boss"==reply:
                reply=""
                call(["espeak","-s140 -ven+18 -z","what should i say"])
                special_command()
                content=reply
                server=smtplib.SMTP("smtp.gmail.com",587)
                server.ehlo()
                server.starttls()
                server.login("kanharajput91@gmail.com","password")

server.sendmail("kanharajput91@gmail.com","kanharajput91@gmail.com",content)
                server.close()
                reply=""
                call(["espeak","-s140 -ven+18 -z","mubarak! email sent
successfull"])
                call(["espeak","-s140 -ven+18 -z","next command,sir"])
```

DESKTOP ASSISTENT PROGRAM

```
from subprocess import call
from pygame import *
import speech_recognition as sr
import datetime
import webbrowser as web
import wikipedia
import wolframalpha
import smtplib
import random

def wish():
time=datetime.datetime.now().time()
hour=time.hour

 if hour>=12 and hour<18:
     print("good afternoon sir")
       call(["espeak","-s140 -ven+18 - z","good
                                              afternoon sir"])
  elif hour>=6 and hour<12:
         print("good morning sir")
          call(["espeak","-s140 -ven+18- z","good
                                                     morning"])
```

```
   else:
       print("good night sir")
       call(["espeak","-s140 -ven+18 -z","good night
                                                    sir"])

   def special_command():
   global reply
   recog=sr.Recognizer()
   mic1=sr.Microphone()

   with mic1 as source:
       print("listning....")
        recog.adjust_for_ambient_noise
        (source,duration=1)
        audio=recog.listen(source)

  try:
 call(["espeak","-s140 -ven+18 -z",
                                          "reconizing."])
       print("recognizing...")

REPLY=recog.recognize_google(audio,language="en-
IN")
       reply=REPLY.lower()
```

```
        print("you said:",reply)

    except Exception as e:
        return()

def command():
    global text
    rec=sr.Recognizer()
    mic=sr.Microphone()

    with mic as source:
        print("listening...")
        rec.adjust_for_ambient_noise(source,duration=1)
        audio=rec.listen(source)

    try:
        print("recognizing....")
        call(["espeak","-s140 -ven+18 -z",
                                          "recognizing"])
        TEXT=rec.recognize_google(audio,language="en-
                                                   IN")
        text=TEXT.lower()
         print("you said:",text)
    except Exception as e:
        print("sir please say that again.")
        call(["espeak","-s140 -ven+18 - z","sir please
                                             say that again"])

        return()

wish()
call(["espeak","-s140 -ven+18 -z","i am your jarvis
how can i help you"])
while True:
    command()
    if "open youtube" in text:
        text=""
        web.open("https://www.youtube. com")

        call(["espeak","-s140 -ven+18 -z","opening
                                              youtube,sir"])
        call(["espeak","-s140 -ven+18 -z","next
                                               command,sir"])

    elif "search" in text:
        text=""
    call(["espeak","-s140 -ven+18 - z","what i
                                 have to search sir"])
    special_command()
      site="https://www.google.com/
      searchsource=hp&ei=0bAJXa
     G9O4W8rQG1mZxA&q="+reply
     web.open(site)
     reply=""
     call(["espeak","-s140 -ven+18 -z","here is
                                            it,sir"])
     call(["espeak","-s140 -ven+18 -z","now
                                 what my  next task,sir"])


 elif "open facebook" in text:
    text=""
    web.open("https://www.facebook.com")
    call(["espeak","-s140 -ven+18 - z","opening
                                        facebook,sir"])
     call(["espeak","-s140 -ven+18 -z","next
                                        command,sir"])

 elif "open whatsapp" in text:
     text=""
    web.open("https://web.whatsapp.com")
  call(["espeak","-s140 -ven+18 -   z","opening
                                          whatsapp,sir"])
   call(["espeak","-s140 -ven+18 -z","next
                                   command,sir"])

   elif "wikipedia" in text:
       text=""
       call(["espeak","-s140 -ven+18 - z", "sir
                     please say,what i have to search"])
       special_command()
      call(["espeak","-s140 -ven+18 -z",
                                      "searching"])
      wiki=wikipedia.summary(reply,
                                      sentences=2)
       print("According to Wikipedia:",wiki)
      call(["espeak","-s140 -ven+18 -z",
                           "sir,wikipedia says that,"])
      call(["espeak","-s140 -ven+18 -z",wiki])
      reply=""
      call(["espeak","-s140 -ven+18 -z","next
                                        command,sir"])
 elif "tell me" in text:
     call(["espeak","-s140 -ven+18 -
                z","yes sir,what i have to search"])
      command()
      client=wolframalpha.Client("JXW
      3A3-EK792GWGXL")
      res=client.query(text)
      output=next(res.results).text
      print(output)
      call(["espeak","-s140 -ven -z",output])
      text=""
      call(["espeak","-s140 -ven+18 -z","next
                                    command,sir"])


    elif "email" in text:
      text=""
      while True:
        call(["espeak","-s140 -ven+18 -z","whom
                    you want to send the mail?sir"])
         special_command()
          if "boss"==reply:
            reply=""
            call(["espeak","-s140 -ven+18
                              -z","what should i say"])
            special_command()
            content=reply
            server=smtplib.SMTP("smtp.gmail.
                                            com",587)
            server.ehlo()
            server.starttls()
            server.login("kanharajput91@gmail.
                                  com","password")
            server.sendmail("kanharajput91@gmail
                                        .com","kanharajput
                                  91@gmail.com",content)
             server.close()
             reply=""
             call(["espeak","-s140 -ven+18
                                  z","mubarak! email  sent
                                            successfull"])
             call(["espeak","-s140 -ven+18
                              -z","next command,sir"])

       elif "no one"==reply:
            call(["espeak","-s140 -
                      ven+18 -z","what can i do for
                                  sending a mail"])
              reply=""
              break
```

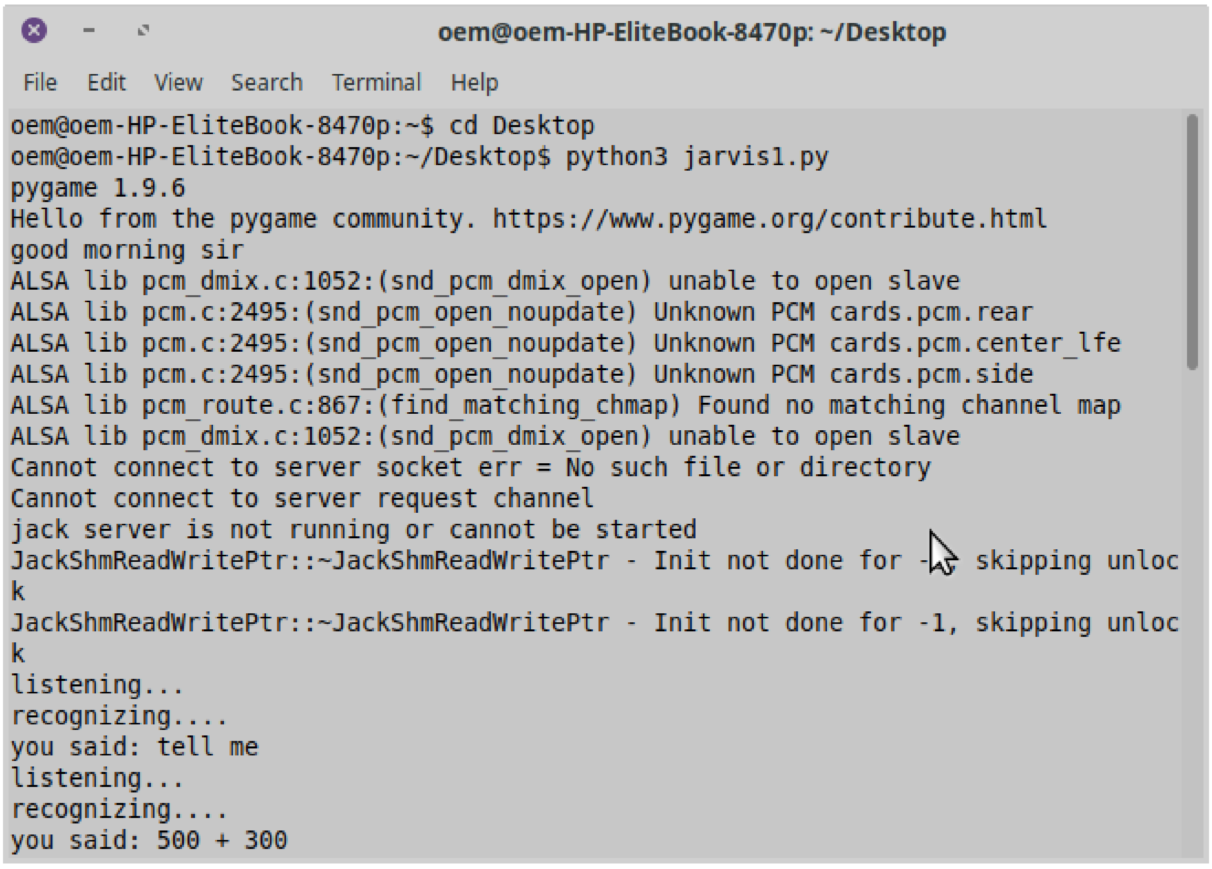
```
else:
    reply=""
    call(["espeak","-s140 -ven+18 -
                              z","please type the
                              email adress ,sir"])
     email=str(input("email:"))

        if email=="exit":
           call(["espeak","-s140 -ven+18
                  z","what can i do for you sir
                         other then sending mail"])
              break

         else:
           call(["espeak","-s140 -ven+18 -
                       z","what should i say?sir"])
             special_command()
             content=reply
              server=smtplib.SMTP
                            ("smtp.gmail.com",587)
              server.ehlo()
              server.starttls()
             server.login("kanharajput91@
                        gmail.com", "password")
                        server.sendmail("kanharajput91
                           @ gmail.com",email,content)
                        server.close()
                        call(["espeak","-s140 ven+18-z"
                                ,"mubarak,email sent
                                      successfully!"])
                         reply=""

           elif "open gmail" in text:
              text=""
              call(["espeak","-s140 -ven+18 -z","opening
                                               gmail,sir"])
               web.open("https://www.gmail.com")
              call(["espeak","-s140 -ven+18 - z","next
                                           command,sir"]
           elif "play song" in text:
              text=""
              songs=["kanha.wav","pehli.wav", "jogi.wav"]
              file=(random.choice(songs))
              mixer.init()
              mixer.music.load(file)
              call(["espeak","-s140 -ven+18 -z","okay sir,
                                 here is your music,enjoy!"])
              mixer.music.play()
```

```
   while mixer.music.get_busy:
         time.Clock().tick(10)

elif  "stop" in text:
    call(["espeak","-s140 -ven+18 -z","okay sir,
                                as you wish,bye-bye"])
      break

elif "flow" in text: #to open stackoverflow site
     text=""
   call(["espeak","-s140 -ven18 - z","opening
                          the stackoverflow site,sir"])
      web.open("https://stackoverflow.com")
      call(["espeak","-s140 -ven+18 -z","next
                                         command,sir"])
```

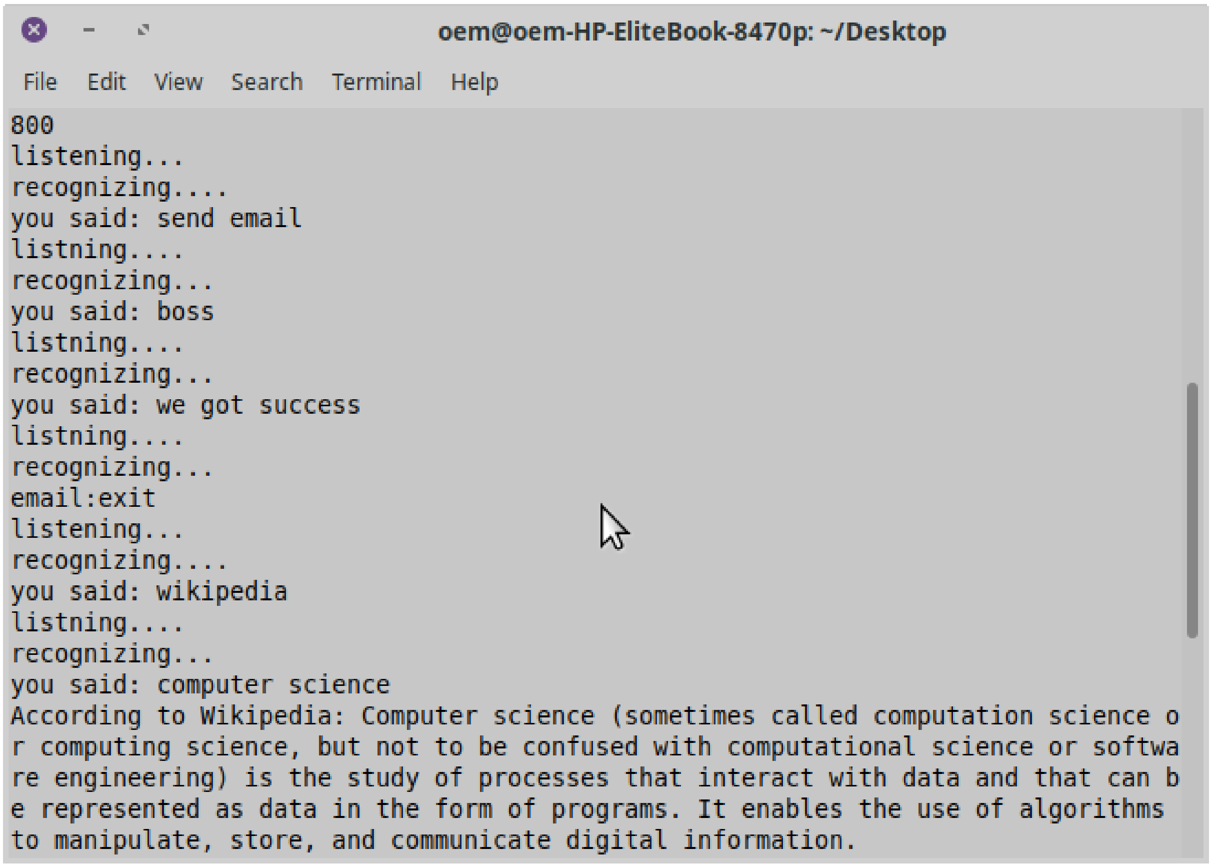
CONCLUSION

Did as we thought to make a project which works on our command atlast we did it.The program run like we thought.sometimes errors are occurs due to the modules errors means when we not get the data from the particular api then we have to keep patience and run the program this thing mostly happens with speech recognition, wolframalpha api because our input not make sense some times.But for first time in the life to make a project and get success is great.

It works like that:

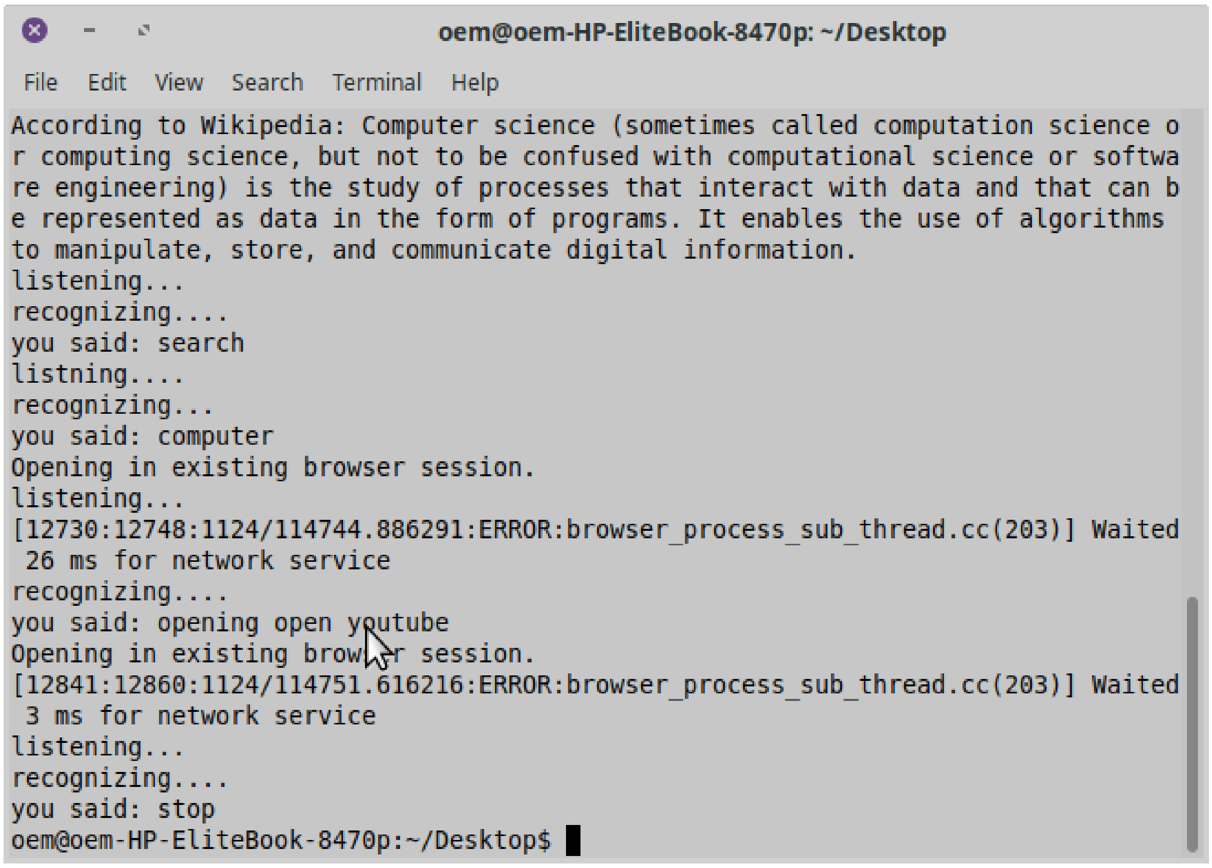

oem@oem-HP-EliteBook-8470p: ~/Desktop
File Edit View Search Terminal Help

```
oem@oem-HP-EliteBook-8470p:~$ cd Desktop
oem@oem-HP-EliteBook-8470p:~/Desktop$ python3 jarvis1.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
good morning sir
ALSA lib pcm_dmix.c:1052:(snd_pcm_dmix_open) unable to open slave
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_route.c:867:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_dmix.c:1052:(snd_pcm_dmix_open) unable to open slave
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for - skipping unloc
k
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unloc
k
listening...
recognizing....
you said: tell me
listening...
recognizing....
you said: 500 + 300
```



oem@oem-HP-EliteBook-8470p: ~/Desktop
File Edit View Search Terminal Help

```
800
listening...
recognizing....
you said: send email
listning....
recognizing...
you said: boss
listning....
recognizing...
you said: we got success
listning....
recognizing...
email:exit
listening...
recognizing....
you said: wikipedia
listning....
recognizing...
you said: computer science
According to Wikipedia: Computer science (sometimes called computation science o
r computing science, but not to be confused with computational science or softwa
re engineering) is the study of processes that interact with data and that can b
e represented as data in the form of programs. It enables the use of algorithms
to manipulate, store, and communicate digital information.
```



oem@oem-HP-EliteBook-8470p: ~/Desktop
File Edit View Search Terminal Help

```
According to Wikipedia: Computer science (sometimes called computation science o
r computing science, but not to be confused with computational science or softwa
re engineering) is the study of processes that interact with data and that can b
e represented as data in the form of programs. It enables the use of algorithms
to manipulate, store, and communicate digital information.
listening...
recognizing....
you said: search
listning....
recognizing...
you said: computer
Opening in existing browser session.
listening...
[12730:12748:1124/114744.886291:ERROR:browser_process_sub_thread.cc(203)] Waited
 26 ms for network service
recognizing....
you said: opening open youtube
Opening in existing browser session.
[12841:12860:1124/114751.616216:ERROR:browser_process_sub_thread.cc(203)] Waited
 3 ms for network service
listening...
recognizing....
you said: stop
oem@oem-HP-EliteBook-8470p:~/Desktop$
```

SYNOPSIS

Desktop assistant will be a terminal based project, we will just use the

python(3.6.8) which is alreay come with the system(linux_mint) and some

modules of python to get,what we want?

The required modules are:-

1) Espeak

2) Pygame

3) Speech Recognition

4) Wikipedia

5) Wolframalpha

6) SMTPLIB

7) Webbrowser

We want to run the program by voice commands like the J.A.R.V.I.S in Iron

man(movie) and the program also gives the answer in suitable manner by

saying and showing in terminal.

The program will speakable by the use of modules Espeak and Pygame and the

listening work will be done by the Speech Recognition module. The program need the internet connectivity because the modules Speech Recognition,Wolframalpha

and Wikipedia are internet dependent. Speech Recognition is a google API which

transmits our speech in text which can be the command for our program.

Wolframaplha ia also an API. It can gives us the answers of our queries like

wheather forcasting, adding the numbers,general knowledge, and it can also tell us

the jokes.

Pygame is a gaming module but we can use that here to play music.

Wikipedia is also make our program interesting it can tell us the information

which we want from wikipedia, we can set how much lines it have to read from

wikipedia.

Smtplib come with python package. We will use it to send the emails.

Webbrowser is also an preinstalled module, we can serve sites by this module,

which we want like google, facebook, stake overflow etc .