Odd Semester (2024)



**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | |  |  |
| **Student Information**: **Surname** | | | | **Given Names**  Edward | **Student ID Number**  2440032316 |
| 1. | | Kurniawan |  |
|  |  |
| **Course Code** | **:** COMP6056 |  |  | **Course Name** | **:** Program Design Methods |
| **Class** | **:** L1AC |  |  | **Name of Lecturer(s)** | **:** Jude Joseph Lamug Martinez |
|  |  |  |  |  |  |
| **Major** | **:** Computer Science |  |  |  |  |
| **Title of Assignment**  (if any) | : Voice assistant (Pickles) |  |  |  |  |
| **Type of Assignment** | **:** Final project |  |  |  |  |
| **Due Date** | **:** 13-01-2021 |  |  | **Submission Date** | **:** |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Edward Matthew Kurniawan

“Voice assistant (Pickles)”

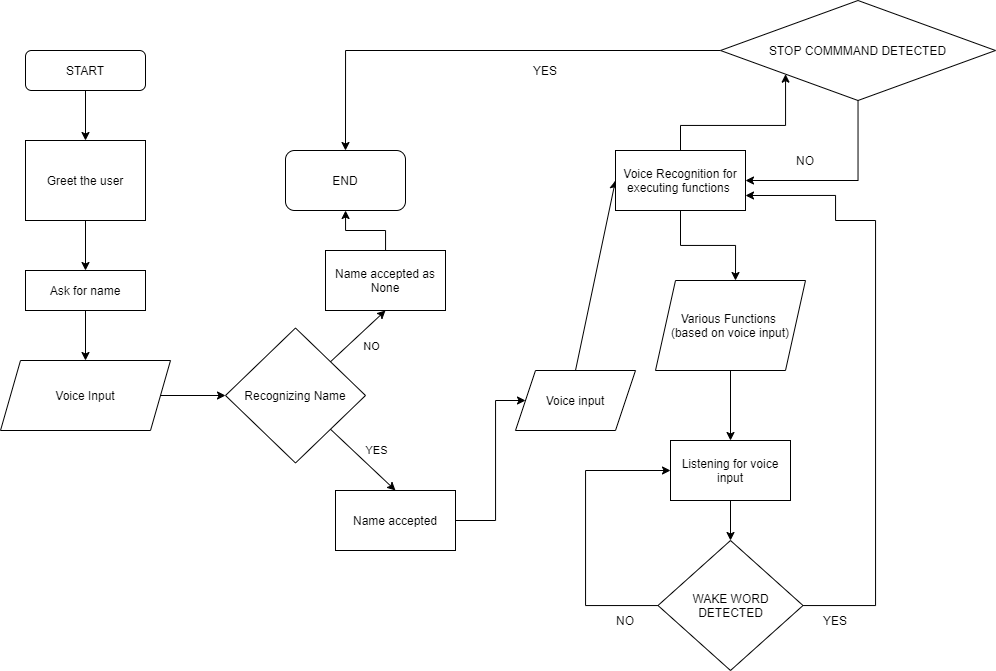
Name : Edward Matthew Kurniawan

ID : 2440032316

1. Project Specifications

Pickles is a voice assistant that is designed to help the user in doing basic everyday tasks on their PC. It is built using python using the Speech Recognition module to take in the voice input, pyttsx3 for the text to speech function, and a bunch of other modules for the other functions. This program is made to help the user do some simple task that people usually do every day using their PC.

1. Implementation



1. BASIC INITIALIZATION PROCESS

There are 3 basic processes in this voice assistant program. The first process is greeting the user, the second process is asking for a username, and the third process is recognizing voice input the run the functions.

1. Greeting the user

The program will first greet the user. The greeting itself changes according to the current time as well. The program will say “Good morning” if it is run at any given time point from midnight until noon. Running at any given time point from noon until 6pm will result in the program greeting the user with “Good afternoon”. Any other time points beside that will result in the program greeting the user with “Good evening”.

1. Username asking

The program will then ask the user for a name. Not saying anything into the microphone as the input device within a certain time limit will result in the program accepting a None type as the user. The main function itself will always check for an existing username. In the event a None type is recognized as a user, the program will terminate.

1. Voice input recognizing

The user could then say anything into the voice recognition software. If it matches one of the commands listed in the program, the program will then execute a function according to the command given. If a command is not given within a certain time period, the program will terminate. Giving a stop command will also terminate the program

1. AVAILABLE FUNCTIONS

The voice assistant is currently programmed with 3 subcategory of functions in mind: web automation, file automation, and miscellaneous functions.

1. Web automation

Pickles is currently capable of opening Google, YouTube, and Outlook at a browser. Other web related functions pickles is capable of doing is searching Google, playing a video from YouTube, getting a definition from Wikipedia, and sending messages using the users Gmail account.

1. File automation

Inputting the make a note or write this down command invokes Pickles’ note making function. The function itself is pretty simple and straightforward. The program will ask the user what to take note of, and a new file will be created. The file are stored in a folder named notes, and each file created will be named the date and time when the file was created. This allows the user to easily sort out the file they no longer need

1. Miscellaneous

The miscellaneous functions are not stored in separate files due to the sheer simplicity of the functions. An example of a miscellaneous function programmed into the voice assistant is telling a joke. The user could also ask for the current time, empty their recycle bin, shutdown or restart their computer, and get responses with certain commands like “how are you”.

1. LIBRARIES USED
2. Pyttsx3

Used for the voice of Pickles. Text to speech module using Google

1. Selenium

For web browser automation work. Used to add search codes for the search google function. It is also used to play a video from YouTube.

1. Webbrowser

To open internet related browsers. In the case of the voice assistant, used to open Google, YouTube, and Outlook

1. Base64

For email message encoding

1. Gmail API

To send email messages and authorize the users account

1. Pyjokes

To have the Pickles tell programming related jokes

1. Datetime

For getting the date and time

1. Speech Recognition

The main voice recognition library. Used for recognizing verbal inputs for the program to recognize

1. OS

To use some built in functions, including but not limited to, making a new directory for the notes, making the notes in the right directories, and clearing the console of any previous commands before initializing Pickles.

1. Subprocess

Mainly used to open notepad for Pickles to be able to write something in.

1. Winshell

For operation relating to windows, in this case specifically used for emptying the recycle bin.

1. PROJECT TECHNICAL DESCRIPTION
2. Getting the voice assistant up and running

To get the voice assistant to run successfully, there are two main functions that needs to be created: The speech recognition and the text to speech function. Using these two basic feature, we could create a very basic voice assistant that could take a voice input, and have it respond with an audio output

I am using the pyttsx3 module as the text to speech of choice for the output of the voice assistant due to it being easier to use, as well as the module itself being usable offline. Another advantage of using pyttsx3 over a module like GoogleTTS is that pyttsx3 doesn’t save an audio file to the machine and it is also usable when the computer isn’t connected to the internet. This makes the voice assistant itself usable even when the machine isn’t connected to the internet, although some functions that open the web browser could be impaired and rendered unusable.

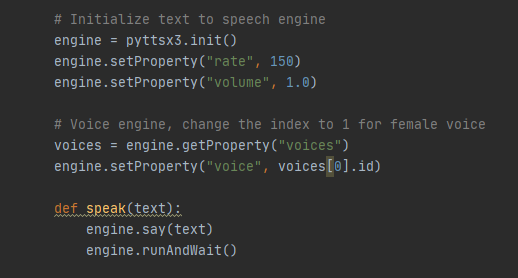
1. Adding functionality to the voice assistant

After the basic framework of the voice assistant is established, additional functionalities are added by classifying command words detected in the input. Unlike AI which uses machine learning to compute the commands, and could cater to each user, a simple voice assistant detects exact commands in the voice input and responds with a specific output corresponding to the command detected in the input.

After defining the exact commands which could be detected by the voice assistant, user defined functions could then be added and executed when the voice assistant detects a matching command within the voice input. All of the capabilities of the voice assistant will be explained in the code explanation below.

1. CODE EXPLANATION

“voice\_functions.py” file

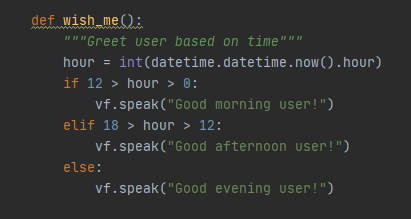


The first thing I did was making the text to speech engine for Pickles to be able to speak through. I initialized the engine, and set the speed and volume of the speech to the optimal settings. I then iterate through the voices to get the voice for the voice assistant. Changing the index in the square brackets to one results in Pickles speaking with a female voice. I then defined a function “speak” that passes a text parameter and runs the text to speech engine to give the voice output.

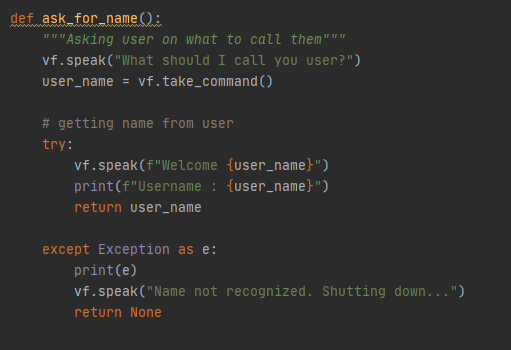


Then I wrote another function called “take\_command” that uses the Speech Recognition module to get a voice input from the user. I created a variable that acts takes a Microphone as the input device, and I set it to listen for voice inputs for 2 minutes before the voice recognition software times out. The try – exception function checks if a voice is heard and recognized by the program. Once the program hears and recognizes a word, the program will print out “Recognizing…” to the console, and reviews the command. When the program finally accepts the voice input, “User said : command” will then be printed on the console. In the event of Pickles not recognizing a word, the exception will occur and print out the error.

“greetings.py” file



This function is used by the voice assistant to greet the user according to the current time the user initializes the voice assistant. I first checked the current hour using the datetime module, and then assigned it as an integer to a variable called hour. I then wrote an if – else statements that checks whether the current hour is morning, afternoon, or night. An appropriate response is then given by the program according to the current hour



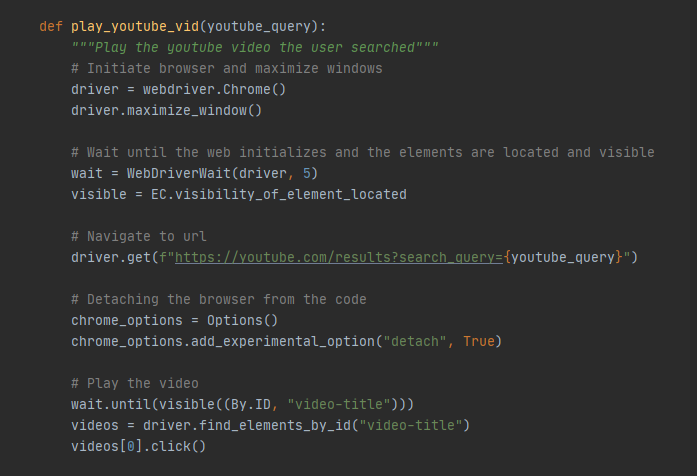
The “ask\_for\_name” function is used to verify if someone is actually using the voice assistant or not. I wrote a code that asks the user for their name, or whatever they preferred to be called as, then the program will try running a code based on if a username is recognized or not. The username itself doesn’t have to be an actual name, as long as the program could recognize it. If a name isn’t recognized by the program, the program will let the user know that a name is not recognized and the voice assistant will shut down.

“file\_automation.py”



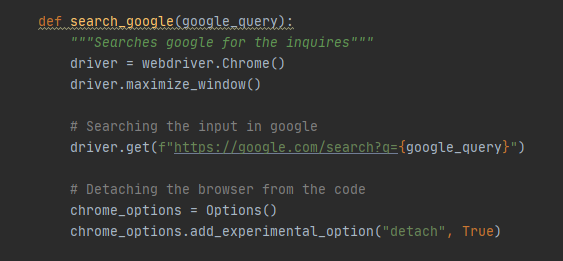
This file contains only one function at the moment, to make notes. I first used the datetime module to create the filename. I then changed the colons in the filename and appended “-note.txt” to the end of the file. The changing of the colons is to prevent filename error because Windows doesn’t permit a filename with a colon in it. I then checked the path of the voice assistant for a folder called notes. If such folder doesn’t exist, the program will create a new folder called notes (using the os module) in the same working directory. The note will then be created with the current datetime as the filename and it will be opened using notepad (using the subprocess module)

“web\_automation.py”

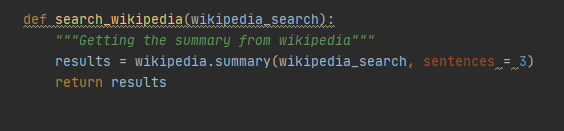


This module deals with everything related to the web and contains most of the functions available in the voice assistant. The first function I created (play\_youtube\_vid) plays a YouTube video requested by the user, and it takes one argument called (youtube\_query). I used the selenium module with the chromedriver provided in the selenium website to open Google Chrome as the web of choice due to its speed. I also chose to maximize the Chrome window opened for more visibility. After successfully opening the Chrome web driver, I then programmed it to wait for 5 seconds and to check for text visibility to prevent the program from checking the loading page, which usually results in an error, before navigating to YouTube.

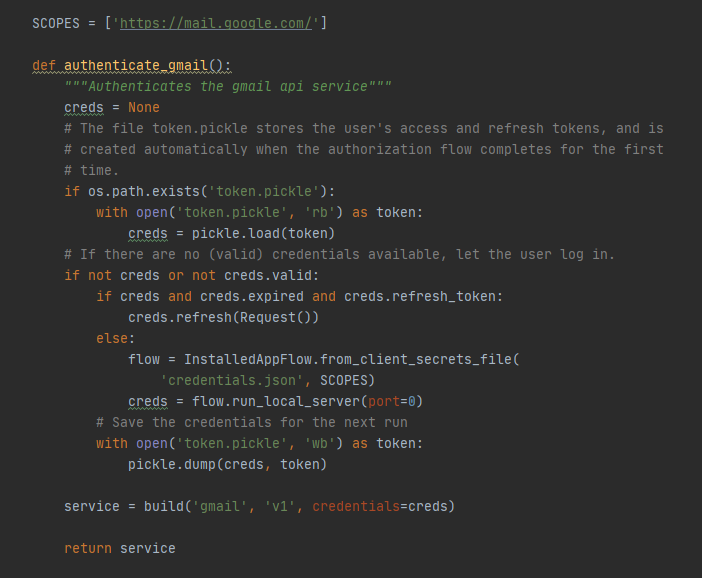
The program knows what video to play by using the video the user wants as the search keyword in the YouTube URL (the argument of the function tells it which keyword the user searched), which is accomplished by using string formatting. After that, I detached the driver from the module, to keep it running even when another command is processed by the voice assistant. I then programmed the voice assistant to wait until it finds the ID of the video title that the user requested before processing the users request, and used the selenium module automation feature to click the top most video that pops out when someone searches that keyword on YouTube.



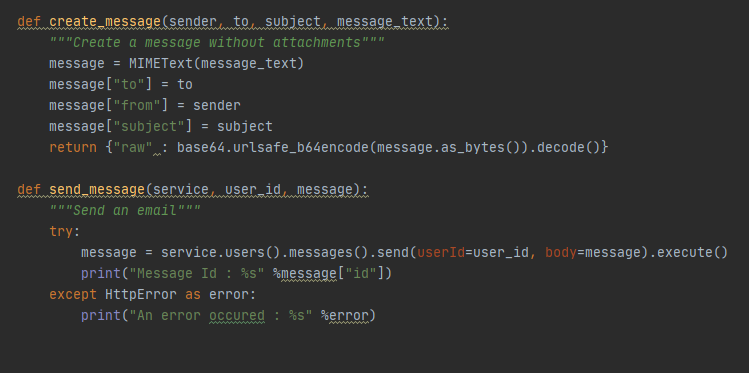
The second function in this module is similar in code to the first one, but instead of playing a YouTube video, this one searches Google. The function itself accepts one parameter, which is “google\_query”. The overall principle is the same as the first function. And like the first function, this function also uses selenium and chromedriver to execute the search. First I initialized the Chrome driver, and maximized the newly opened Chrome window, then I navigated to the search page according to the user’s query (google\_query). Like the first one, I also detached the window so that it doesn’t close when this function finished executing.



The third function is a simple one made with the Wikipedia module. The function itself accepts one argument “wikipedia\_search” that acts as a keyword that will be inputted by the user using the voice assistant. I then made a variable called “result” that gets the summary of something from Wikipedia. The summary is the first 3 sentences from Wikipedia.



The fourth function of this module is to authenticate the Gmail API service. This particular function enables python to be able to send emails from one account (mainly the user account) to an account of their choosing. I chose to use Gmail API for sending emails because the old method, which is establishing a connection using smtplib is outdated, and you would have to feed your username and password into the function as well, while using Gmail API uses access tokens and an encrypted credentials file to authorize the email sending. I first checked the system for an access token file, and would create one if it’s not found in the machine, then it would request an access to one of your gmail account, which is used for sending the emails, and would connect the API with that account. The function would then return a service which will then be used for the email sending function.



The 2 final functions in this module is responsible for the actual sending emails part. They are:

* The first one out of the two is for creating the message. I assigned the actual email message as a MIMEText to a variable called message. MIME (Multipurpose Internet Mail Extensions) itself is an Internet standard that extends the format of email message to support character sets other than ASCII. After that I defined each part of the email into the appropriate format. The return function returns the raw message in the form of bytes that is encoded using base64 (this is due to the message accepted has to be in the form of bytes) which I decoded for email itself to read.
* The second function is to send the newly created message to the intended recipient. It is done using the class provided by the API. Using service.users().messages().send() will invoke the send email function of the class. I passed the necessary arguments into the send function to get the correct message format just like the ones you would get when you send an email to someone. The execute function of the class executes the sending of the email. The function itself accepts three arguments:

1. service

This is the service that is used by the function to send the emails. In this case, the authenticate\_gmail function is the service function that is used here.

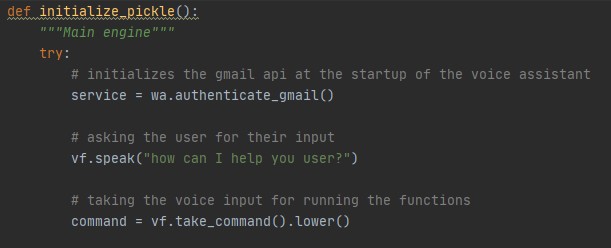
1. user\_id

The user id basically just means the sender email. Using the special variable “me” provides this argument with the account you activated Gmail API with using the access token as a login system.

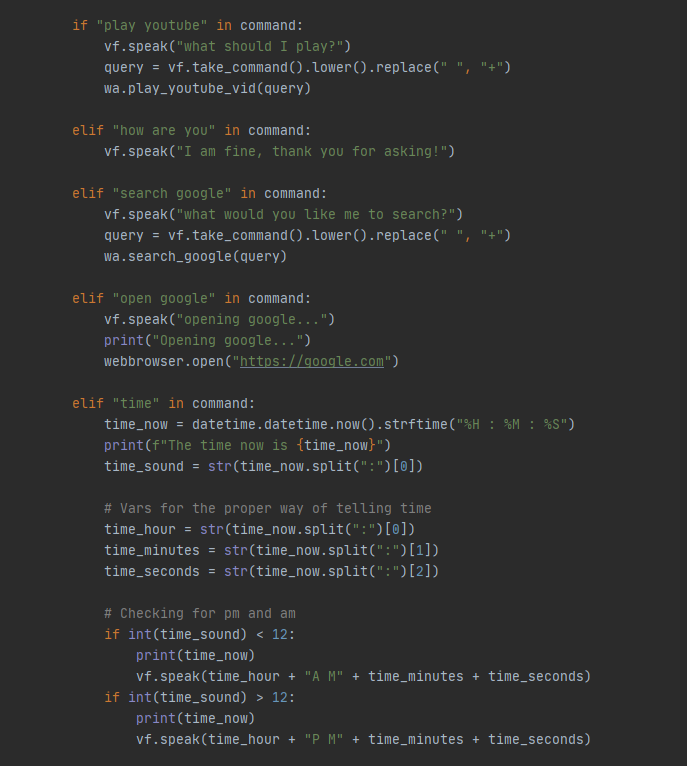
1. message

The message the user intends to send. Passing the create\_message function as the third argument, will send the message created in the mentioned function to the recipient

“main\_functions.py” file

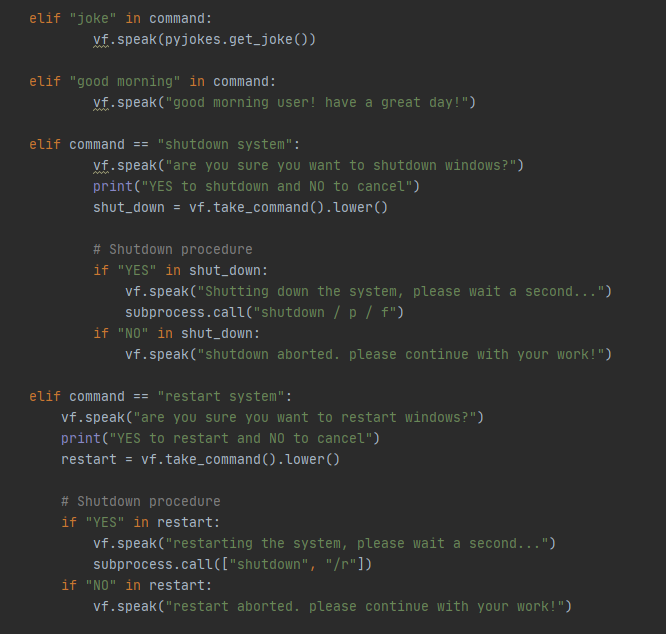


This is the main initialization of the voice assistant. This function is the whole mainframe of how the voice assistant works. These three function deals with the initial prerequisite initialization so that the voice assistant could work properly. First I authenticate the Gmail API service at the start of the voice assistant so that if the access token file isn’t found on the user machine, it would create one right away, thus enabling the send email function to work properly. The second and third line is just the voice assistant asking the user to instruct the voice assistant to do something.

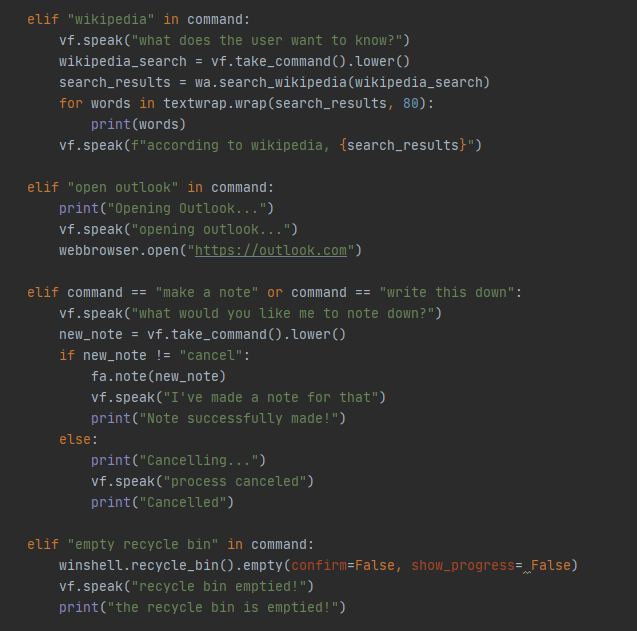


Things that the voice assistant is capable of doing:

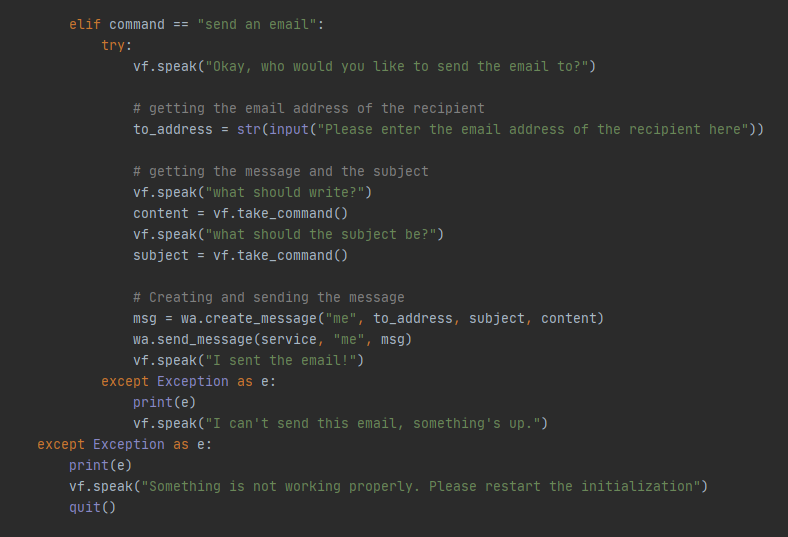
* playing a video from YouTube (using the play\_youtube\_vid function from the web automation module)
* responding to a “how are you” command
* searching something in google (using the search\_google function from the web automation module)
* opening google using the webbrowser module
* tell the user what the current time is. The time told is in the format of am and pm.



* telling a joke
* responding to a good morning command
* shutting down the computer
* restarting the computer

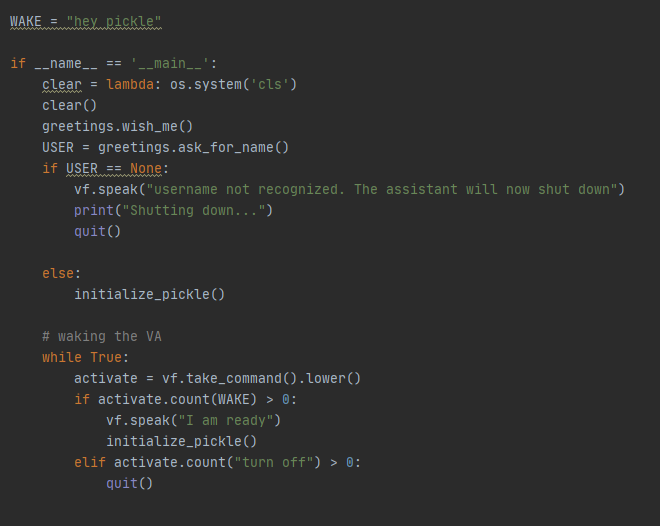


* getting a summary from Wikipedia
* open outlook
* making a note
* emptying the recycle bin



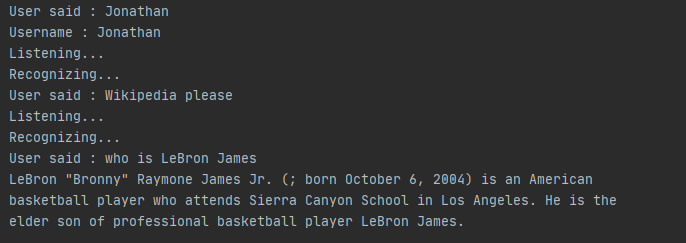
* sending the email (using gmail api and the email related function in the web automation module)

The exception checks whether the voice assistant is run properly or not.

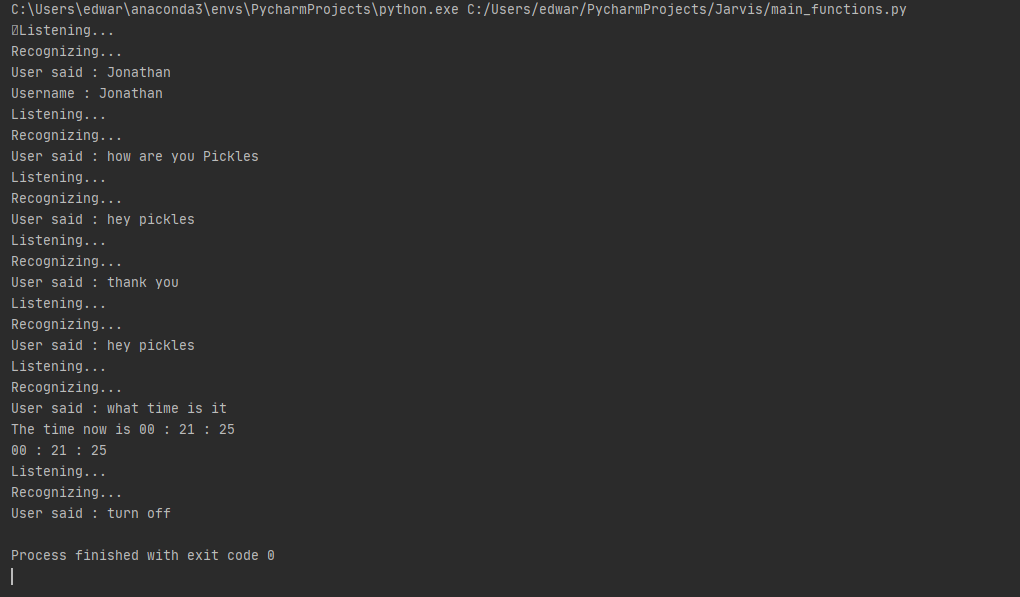


This is the main driver of the code. The lambda function clears any console commands before the initialization of the voice assistant. I also checked whether a username is accepted or not before activating the voice assistant. I also added a wake word that would reactivate the main functions of the voice assistant whenever the recognizer hears the wake word.

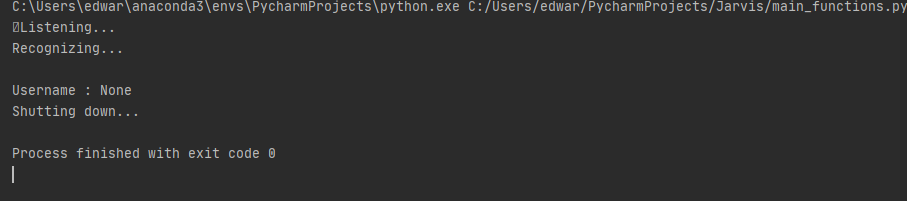
1. EVIDENCE OF WORKING PROGRAM



Successful use of the Wikipedia function



Sucessful use of the “how are you”, time, and wake word methods



No username error

1. PROJECT LINK

<https://github.com/EdwardMatthew/Jarvis>

1. REFERENCES

<https://developers.google.com/gmail/api/guides/sending>

<https://realpython.com/python-speech-recognition/>