**VOCAL ASSISTANT Python**

Student: Uriciuc Mihai

Group: 2031 / Semigroup II

# Introduction

Voice assistants have become an integral part of our daily lives, blending technology and human interaction. From helping users set reminders to answering questions, these assistants provide intuitive interfaces for a broad range of tasks. In this documentation, I will present my own version of a vocal assistant, done in python.

To achieve what I wanted for this project, I created multiple functions that when assembled in the main function will give me a functional voice assistant. The functions can be found below:

1. *speak()* - function that converts text to speech using gTTS library
2. *get\_audio()* - function that gets microphone input
3. *password\_required*() - function that authenticates the user so he can use the voice assistant
4. *authenticate\_google() -* function for authenticating with google calendar
5. *get\_events()* - function that gets the events from google calendar
6. *get\_date() and* *get\_final\_date()* - functions for getting and process the date inpout from the user
7. *note()* and *get\_note()* - functions for letting the user to say something that bill be saved in a notepad
8. *calculator*() and get\_calculator() – functions that open the calculator by voice
9. exit\_*program*() – function that closes the program after too many tries

Also, so I can build this program easily, I used a variety of imports that can be found below:

A screen shot of a computer

Description automatically generated

# Functions speak() and get\_audio()

A screen shot of a computer program

Description automatically generated

*speak()* function gets as input the string text and coverts it into a .mp3 file, it saves it in the workspace folder and then it plays the audio in the file in English.

*get\_audio()* is used for user voice input. With the microphone of the laptop as source, this function helps me store inside a string (said) words that I spoke. Of course, I forgot to mention, this is possible because of the *speech\_recognition library*, which I reduced to only *sr* for an easier coding. The function waits for the user input 10 seconds (*timeout=10)* and he can talk for about 7 seconds *(pharse\_time\_limit=7).*

*recognize.pause\_threshold = 3* allows me to have some delay in my sentence, so I don’t have to hurry and to express myself clearly.

# Functions authenticate\_google() and get\_events()

The first utility that I integrated in my program is Google Calendar. With the help of Google Calendar API I wanted to be able to get the next events that will occur in my schedule. To integrate this in my program I followed the steps that Google provides for personal implementation. Below is the code necessary for authenticating with Google:

A screen shot of a computer program

Description automatically generated

I transformed the *main*() function in the left picture into two functions called *authenticate\_google()* and *get\_events()* so I can use them more easily.

# 3. Function get\_date()

A screenshot of a computer program

Description automatically generated

This function is used for the recognition of date and time in the input *(text).* Using the constants shown in page 7, the code above is able to find those constants in text and then return a valid date so that with the function *get\_events()* the program will return the event that was asked by the user.

# 1.Functions note() and calculator()

The next actions that I implemented in my program are opening the calculator on my computer and opening the TextEdit app to write and save a note in the workspace.

A screen shot of a computer program

Description automatically generated

# 1.Functions get\_final\_date(), get\_note() and get\_calculator()

A screen shot of a computer program

Description automatically generated

Those functions are the ones that search in the constants below for a word to trigger them. For example, if *text=”Hey can you open the calculator”,* the *“open the calculator*” constant will be recognized and the function will be used, opening the calculator.

A computer screen shot of a program code

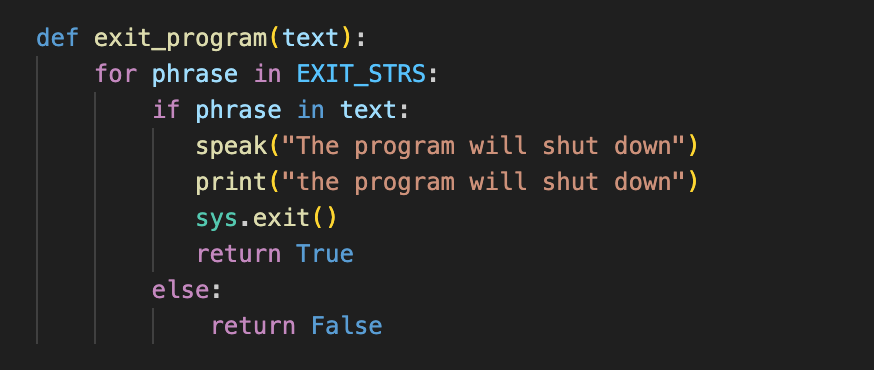
Description automatically generated

# 1.Functions password\_required() and exit\_program()

To add a secure aspect to my project and so that the program can be used as long as a certain command is transmitted, I created these two functions:A screen shot of a computer program

Description automatically generated

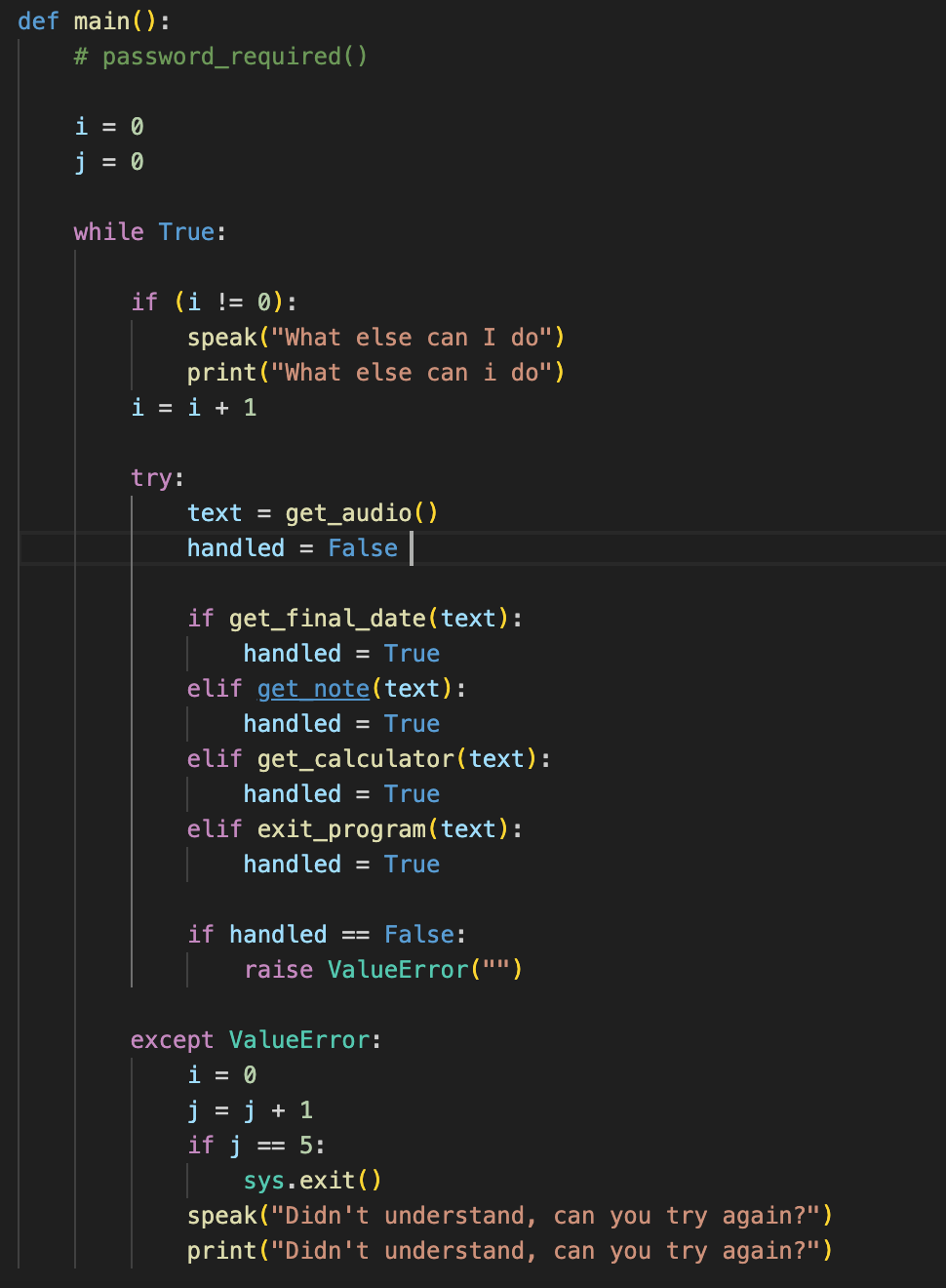
This code asks the user for a password, in this case “happy”. The user has three tries to input the correct answer, after that the program will exit. After the password is communicated correctly, the program will ask the user a question.



The exit\_*program()* will search for a string in constant *EXIT\_STRS* so that when the value is found, the program shuts down.

# 1.Function main()

In *main()* I assembled all the functions that I created so the program can work properly. So the program can run continuously unless the *exit\_program()* function is called, I created a while loop. In this while loop I also created a try/except so that if the input is not valid the user can try again but no more than 5 times. After 5 times the program will shut down.



# 1.Simulation and bibliography

Below I made a screenshot of the project running. Because it’s a voice assistant, I printed what it is supposed to say so I can attach it to my documentation.

A screenshot of a computer

Description automatically generated

Bibliography:

<https://www.youtube.com/@TechWithTim>

<https://console.cloud.google.com/apis/credentials/consent?inv=1&invt=Abmpjg&project=voice-assistant-project-445711>

<https://developers.google.com/calendar/api/quickstart/python>

<https://github.com/termux/termux-app/discussions/3306>

https://pyttsx3.readthedocs.io/en/latest/engine.html#module-pyttsx3.voice