Speech Recognition Code Document

This instruction document is for all the Speech recognition related code, where we have systems that can convert the text into speech or speech into text and methods that can make conversation with users. Here we used gTTS, playsound, NLTK, etc. module for speech recognition concepts.

gTTS (Google Text-to-Speech) is a Python library with Google Translate text-to-speech API that writes spoken mp3 data to a file, a file-like object for further audio manipulation, or stdout. It features flexible pre-processing and tokenizing, as well as automatic retrieval of supported languages.

Play sound on Python is natural. Several modules can play a sound file. These solutions are cross-platform (Windows, Mac, Linux). The main difference is in the ease of use and supported file formats. All of them should work with Python 3. The audio file should be in the same directory as your python program unless you specify a path.

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical Natural Language Processing (NLP) for English written in the Python programming language.

Note: This project is developed on a Linux System (Ubuntu), so it is advisable to use Linux System.

System Set-Up

There are a couple of Python packages that need to validate before running any above program in this folder. Please go to the Installation Documents folder and follow all the steps.

Pre-Requisites

- A system running on Windows/Ubuntu APP/Ubuntu OS
- A user account with sudo/administration privileges
- Access to a terminal window/command-line
- Internet connection

Before continuing with this tutorial, make sure you are logged in as root or a user with sudo or administration privileges and completed all the necessary steps from the Installation Documents folder**.**

In this tutorial, we will explain all the Speech Recognition Codes that are written in Python3 on Ubuntu. We have some codes related to Speech Recognition as follows:

1. Speech_Question.py
2. Speech_Normal.py
3. Speech_Name_Organization.py
4. Speech_Emergency_Evacuation_Procedures.py

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$ ls -lrt
total 72
-rwxrwxr-x 1 somak somak  7052 Jun 25 22:13 Speech_Normal.py
-rwxrwxr-x 1 somak somak 10297 Jun 26 02:17 Speech_Emergency_Evacuation_Procedures.py
-rw-rw-r-- 1 somak somak  4352 Jun 26 02:23 README.rst
drwxrwxr-x 2 somak somak  4096 Jun 29 19:45 Emergency_Evacuation_Procedures
-rwxrwxr-x 1 somak somak 16019 Jun 29 20:27 Speech_Question.py
-rwxrwxr-x 1 somak somak 22528 Jun 29 20:27 Speech_Name_Organization.py
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$
```

Each of the codes in this folder has one function called **process_parameter_set(),** which contains the different parameter values. Depending on the parameter value, the program will perform another process. Hence it is essential to validate this function and all the parameters before running any codes.

```
def process_parameter_set():
    """
```

```
    yes_syn_words = ['all right', 'alright', 'very well', 'of course', 'by all means', 'sure', 'certainly',
                     'absolutely', 'indeed', 'affirmative', 'in the affirmative', 'agreed', 'roger', 'aye',
                     'aye aye', 'yeah', 'yah', 'yep', 'yup', 'uh-huh', 'okay', 'ok', 'right', 'surely',
                     'yea', 'well', 'course', 'yes', 'please']

    no_syn_words = ['no', 'not', "n't", 'nae', 'na', 'never', 'nope']

    stop_words = set(stopwords.words("english"))
    record = sr.Recognizer()
    mp3_filename = "Speech_Question"
    text = "I am going to ask few question to you. You can answer with Yes or No. If I do not get an input from you " \
           "within 5 second, then, I will prompt a pop up message to you."

    device_list = sr.Microphone.list_microphone_names()

    if 'pulse' in device_list[0:7]:
        device_index = device_list.index('pulse')
    elif 'USB PnP Sound Device: Audio (hw:2,0)' in device_list[0:7]:
        device_index = device_list.index('USB PnP Sound Device: Audio (hw:2,0)')
    else:
        device_index = 0

    # device_index = 2

    output_file = "Speech_Question_Output.txt"

    return yes_syn_words, no_syn_words, stop_words, record, mp3_filename, text, device_index, output_file
```

Also, it is crucial to test the microphone index. In this process, we are automatically taking all the microphone devices present into the device_list variable. Later, checking if "pulse" or "USB PnP" is present in the device list or not. If that is present, we are assigning the device_index to the index position of "pulse" or "USB PnP," but if that is not present then, we are setting as 0.

For your case, this value (USB details) may change. So, it is better to validate those in python console as below:

```
>> device_list = sr.Microphone.list_microphone_names()
>> print(device_list)
```

If you find your microphone device details (With working), then you can directly assign that value to device_index in this function.

All the images of emergency and evacuation procedures are present inside the Emergency_Evacuation_Procedures folder. So, you need to verify that directory is present, and all the pictures are there. If not, then create that directory and put all the images inside of it. Please check the program to verify the name of the photos.

Speech Recognition Code

Speech Recognition codes written in Python as follow:

1. **Speech_Question.py**

   Speech_Question.py is to ask YES or NO related questions to the user. If this program does not get input from users within a given time, it will prompt a pop-up message to click OK or Cancel. This process will be called from Main_Process.py with the question as an argument. Depending on the person's response, this program will write an output file (Speech_Question_Output.txt) as YES or NO or NONE.

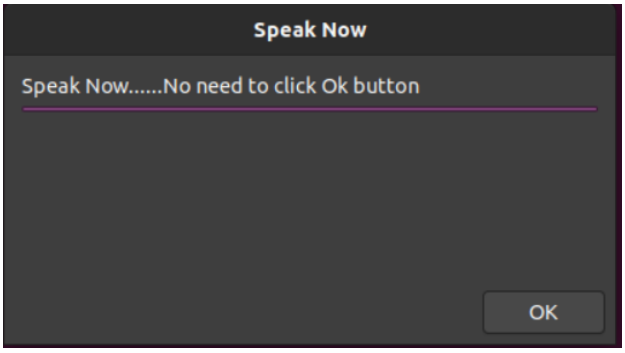
   Execution:

   - To run this program, run the below command from your command prompt.

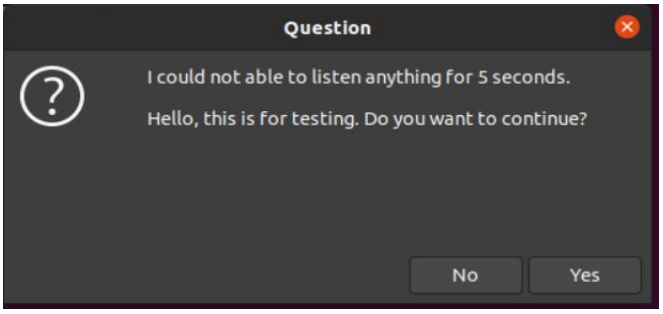
```
$ python3 Speech_Question.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$ python3 Speech_Question.py
Starting program : Speech_Question.py - At : 20:07:41 On : 29/06/2020
Expression 'alsa_snd_pcm_hw_params_set_buffer_size_near( pcm, hwParams, &alsaBufferFrames )' failed in 'src/
Expression 'alsa_snd_pcm_hw_params_set_buffer_size_near( pcm, hwParams, &alsaBufferFrames )' failed in 'src/
Speak:
yes please
['yes', 'please']
Yes!! Person wants to continue
Deleting mp3 and output file. Value of stand_alone_flag :  1
Ending program : Speech_Question.py - at : 20:07:53 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$
```

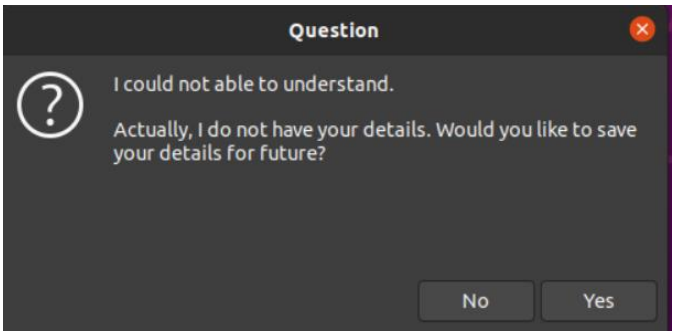
   - After the process asks the question, it will prompt a Speak Now message for the users as below. User no need to click OK button, they can only speak once they can see the pop-up message.



   - If the process cannot able to get an input from the users for 5 seconds, then it will pop-up another screen for the user as below:



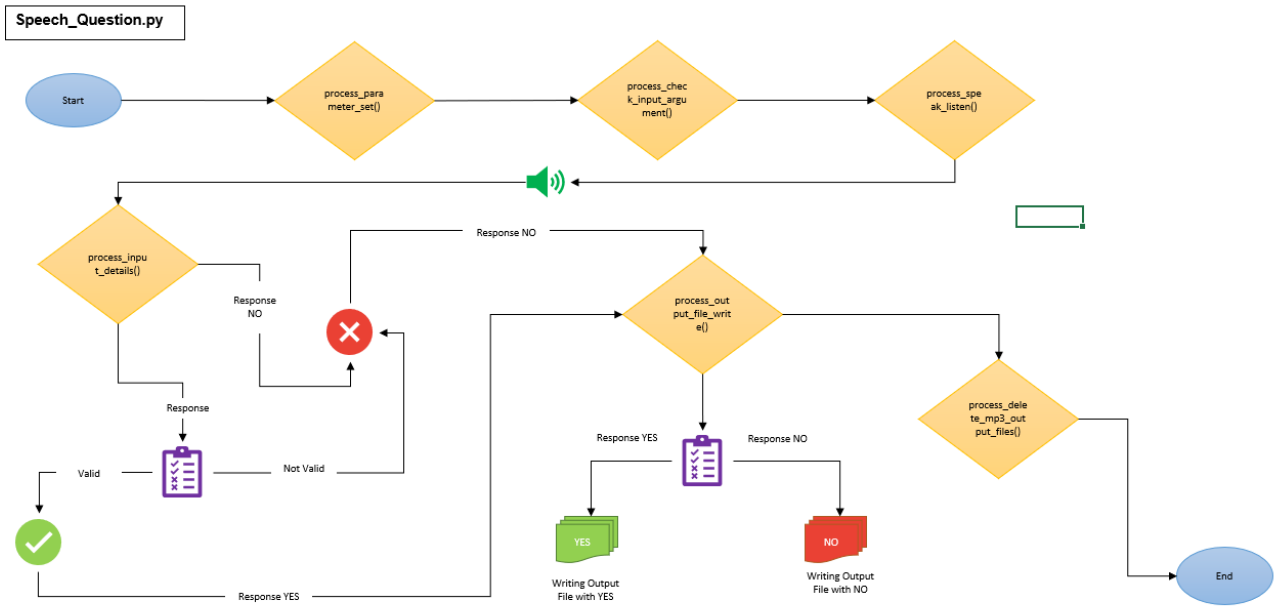
- If the process cannot able to understand, then also it will pop-up another screen for the user as below:



- In this pop-up screen, users can directly click Yes to continue or No to cancel.

Flowchart

Following is the flowchart architecture of the Speech_Question.py program.



2. Speech_Normal.py

Speech_Normal.py is a standard speech-related program. It will be called from the Main_Process.py with a text message that Robot needs to speak. This process will not ask anything. It will talk whatever text message it will receive.

Execution:

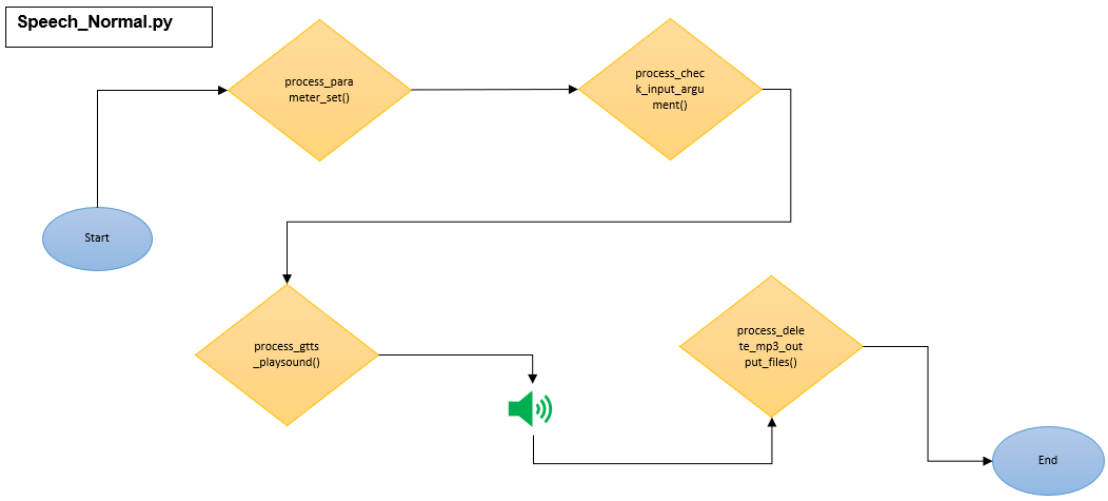
- To run this program, run the below command from your command prompt.

```
$ python3 Speech_Normal.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$ python3 Speech_Normal.py
Starting program : Speech_Normal.py - at : 19:42:05 on : 29/06/2020
Deleting mp3 and output file. Value of stand_alone_flag :  1
Ending program : Speech_Start_End.py - at : 19:42:10 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$
```

Flowchart

Following is the flowchart architecture of the Speech_Normal.py program.



3. Speech_Name_Organization.py

Speech_Name_Organization.py is to ask the Name & Organization to Unknown Person. Later, it will ask the person if they want to save their details or not. If this program does not get input from users within a given time, then it will prompt a pop-up message to enter the details or to click OK or Cancel. Depending on the person's response, this program will write an output file (Speech_Name_Organization_Output.text) as YES or NO or NONE.

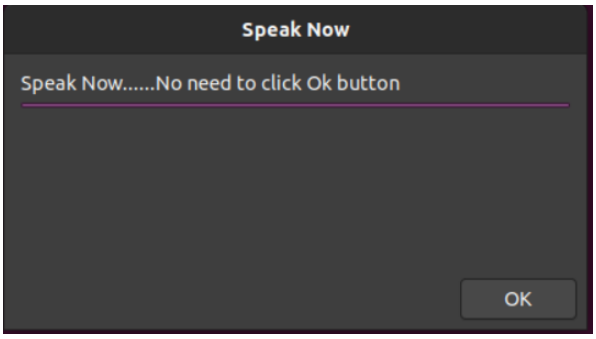
Execution:

- To run this program, run the below command from your command prompt.

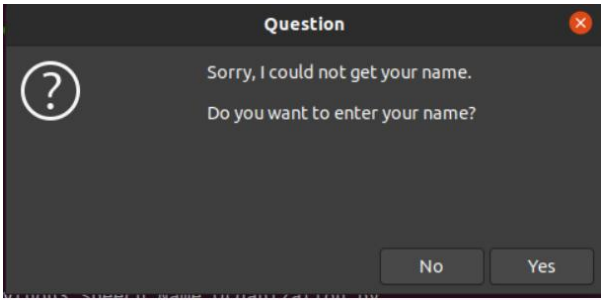
```
$ python3 Speech_Name_Organization.py
```

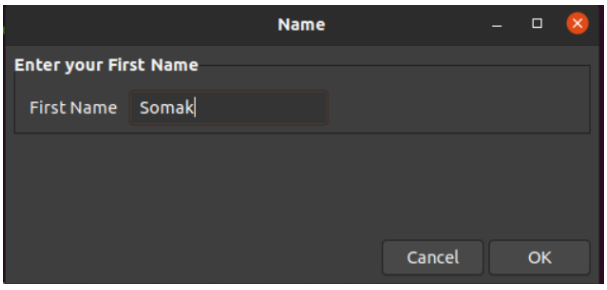
```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$ python3 Speech_Name_Organization.py
Starting program : Speech_Name_Organization.py - at : 20:10:02 on : 29/06/2020
Expression 'alsa_snd_pcm_hw_params_set_buffer_size_near( pcm, hwParams, &alsaBufferFrames )' failed in 'src/hos
Expression 'alsa_snd_pcm_hw_params_set_buffer_size_near( pcm, hwParams, &alsaBufferFrames )' failed in 'src/hos
Speak:
hi my name is Bob
{'Bob'}
Speak:
McMaster University
{'McMaster University'}
Speak:
no thank you
No!! Do not want to continue
Deleting mp3 and output file. Value of stand_alone_flag :  1
Ending program : Speech_Name_Organization.py - at : 20:10:51 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$
```

- After the process asks the question, it will prompt a Speak Now message for the users as below. User no need to click OK button, they can only speak once they can see the pop-up message.

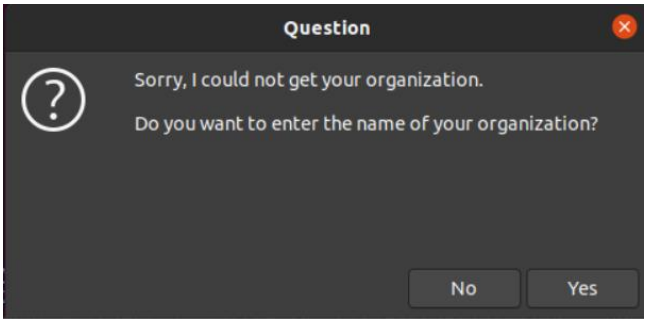


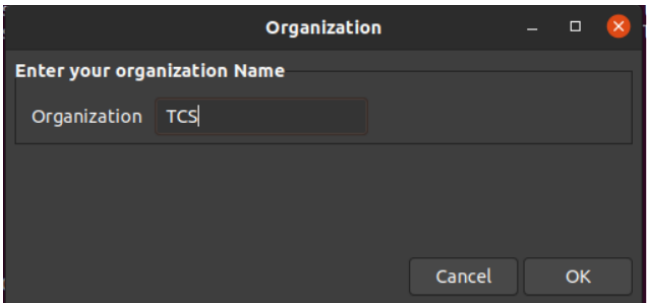
- If the process cannot be able to get input from the users for 5 seconds or cannot able to understand, then it will ask the user if they want to fill their names. Click yes to enter the name or No to exit.





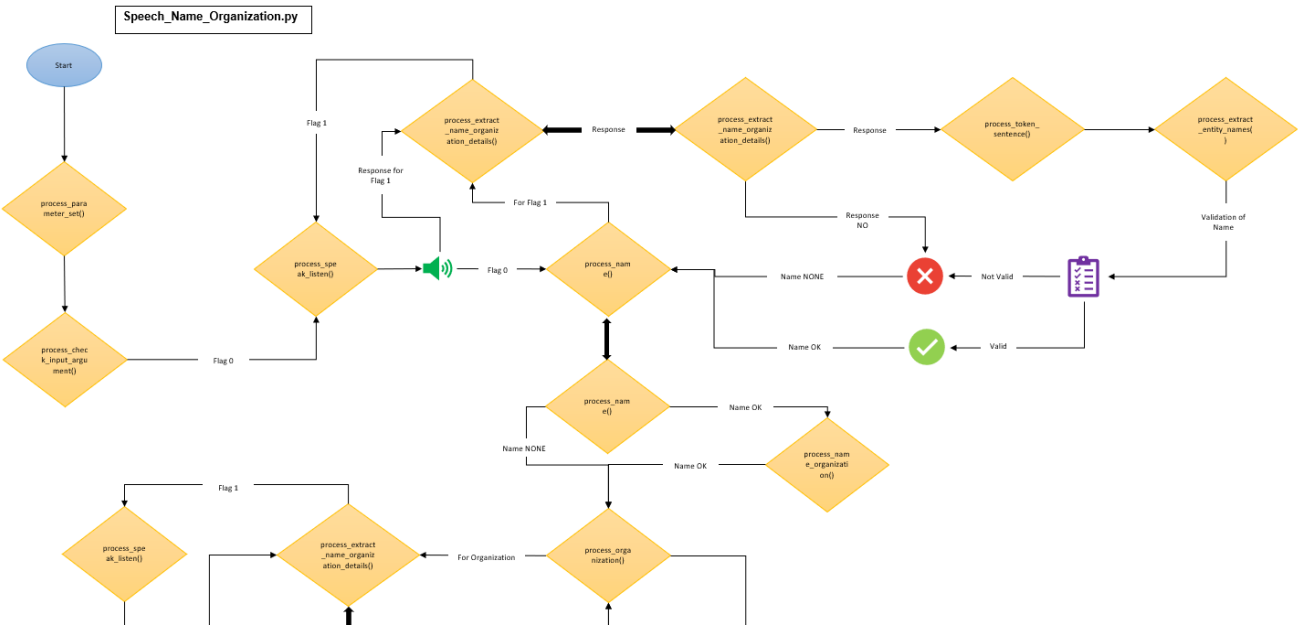
- If the process cannot be able to get input from the users for 5 seconds or cannot able to understand, then it will ask the user if they want to fill their organization details. Click yes to enter the details or No to exit.

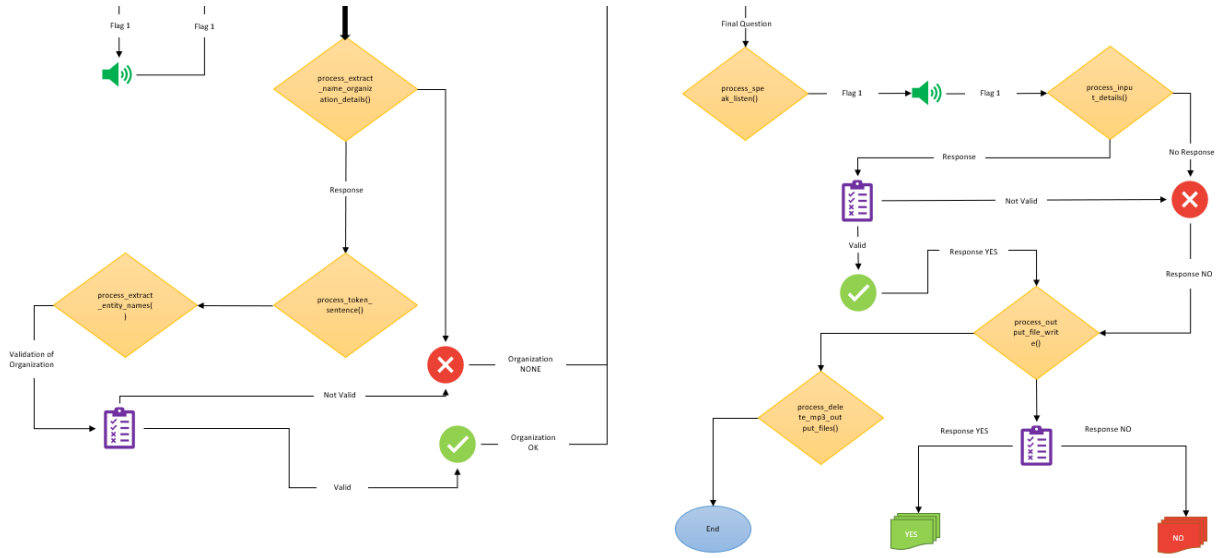




Flowchart

Following is the flowchart architecture of the Speech_Name_Organization.py program.





4. Speech_Emergency_Evacuation_Procedures.py

Speech_Emergency_Evacuation_Procedures.py program is to show the emergency and evacuation procedure to the users, and while showing the image, this program will describe the image by gTTS. All the pictures of emergency and evacuation procedures are present inside the Emergency_Evacuation_Procedures folder. Please check the def process_parameter_set() function and check the path before executing this process.

Inside of the Emergency_Evacuation_Procedures folder, five images are present for this process to run successfully. In function emergency_evacuation_process(), this program will first check if this folder present or not. If the folder is If the path is present, then it will enter the path and will validate each image. If the images exist, then it will open all the photos one by one and call process_gtts_playsound to describe it.

Path & Images are in follow :

Path/Folder/Directory	Images Name
Emergency_Evacuation_Procedures	Emergency_Evacuation_Procedures.png
Emergency_Evacuation_Procedures	Employee_and_Occupants_Responsibilities.png
Emergency_Evacuation_Procedures	Designated_Meeting_Area.png
Emergency_Evacuation_Procedures	Main_Floor_Layout_and_Emergency_Exits.png
Emergency_Evacuation_Procedures	Second_Floor_Layout_and_Emergency_Exits.png

If you are changing the name of the images, then make sure to turn it inside of the program also.

Execution:

- To run this program, run the below command from your command prompt.

```
$ python3 Speech_Emergency_Evacuation_Procedures.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$ python3 Speech_Emergency_Evacuation_Procedures.py
Starting program : Speech_Emergency_Evacuation_Procedures.py - at : 19:43:58 on : 29/06/2020
Emergency_Evacuation_Procedures.png is present
Showing Image - Emergency Evacuation Procedure
Employee_and_Occupants_Responsibilities.png is present
Showing Image - Employee and Occupants Responsibilities
Designated_Meeting_Area.png is present
Showing Image - Designated Meeting Area
Main_Floor_Layout_and_Emergency_Exits.png is present
Second_Floor_Layout_and_Emergency_Exits.png is present
Showing Image - Main Floor and Second Floor Layout and Emergency Exits
Ending program : Speech_Emergency_Evacuation_Procedures.py - at : 19:45:12 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Speech_Recognition_Code$
```

Flowchart

Following is the flowchart architecture of the Speech_Emergency_Evacuation_Procedures.py program.