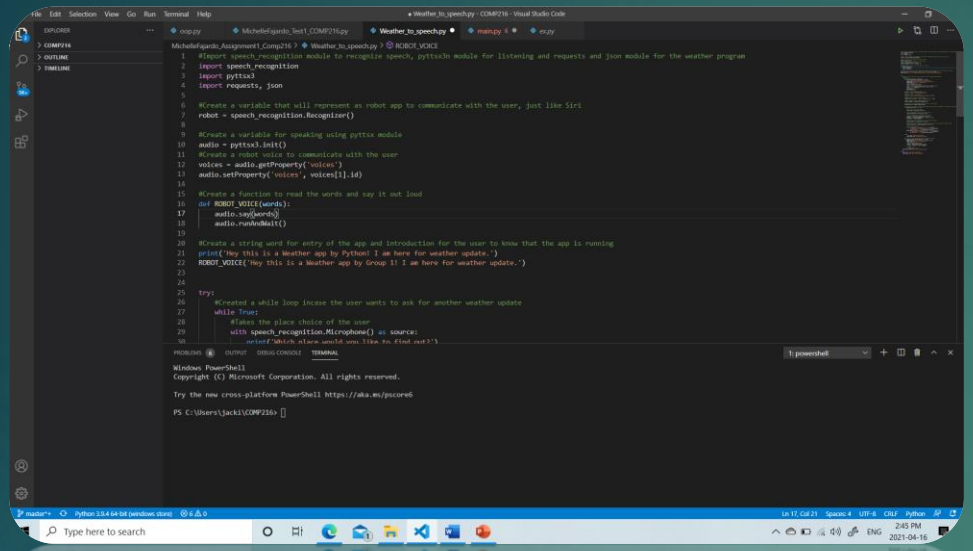
Voice Command Web Scraper

Group 1

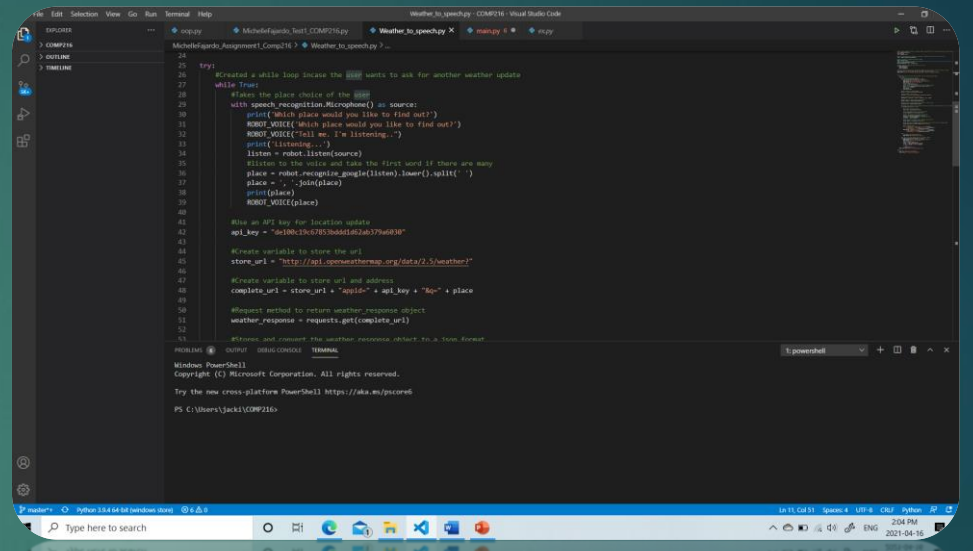
CASRON JOHNSON
CHRISTO JACOB
SHANIQUO MCKENZIE
MUSTAFA BUTT
MICHELLE FAJARDO

Codebase for the Project Start-Up

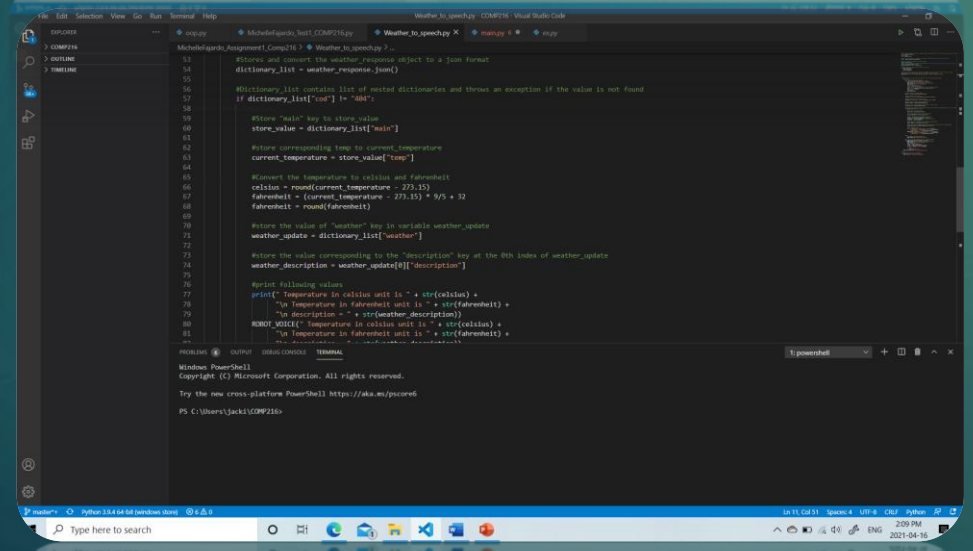
1



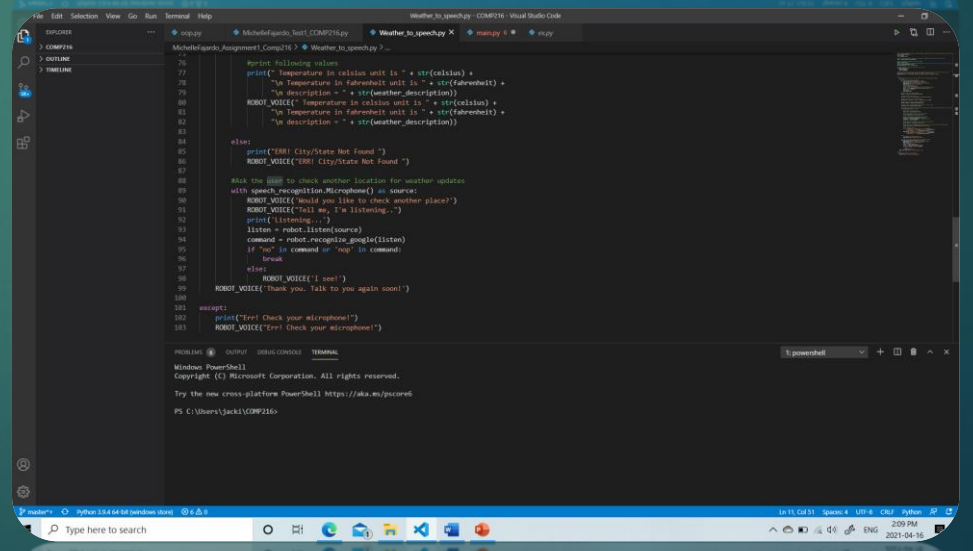
2



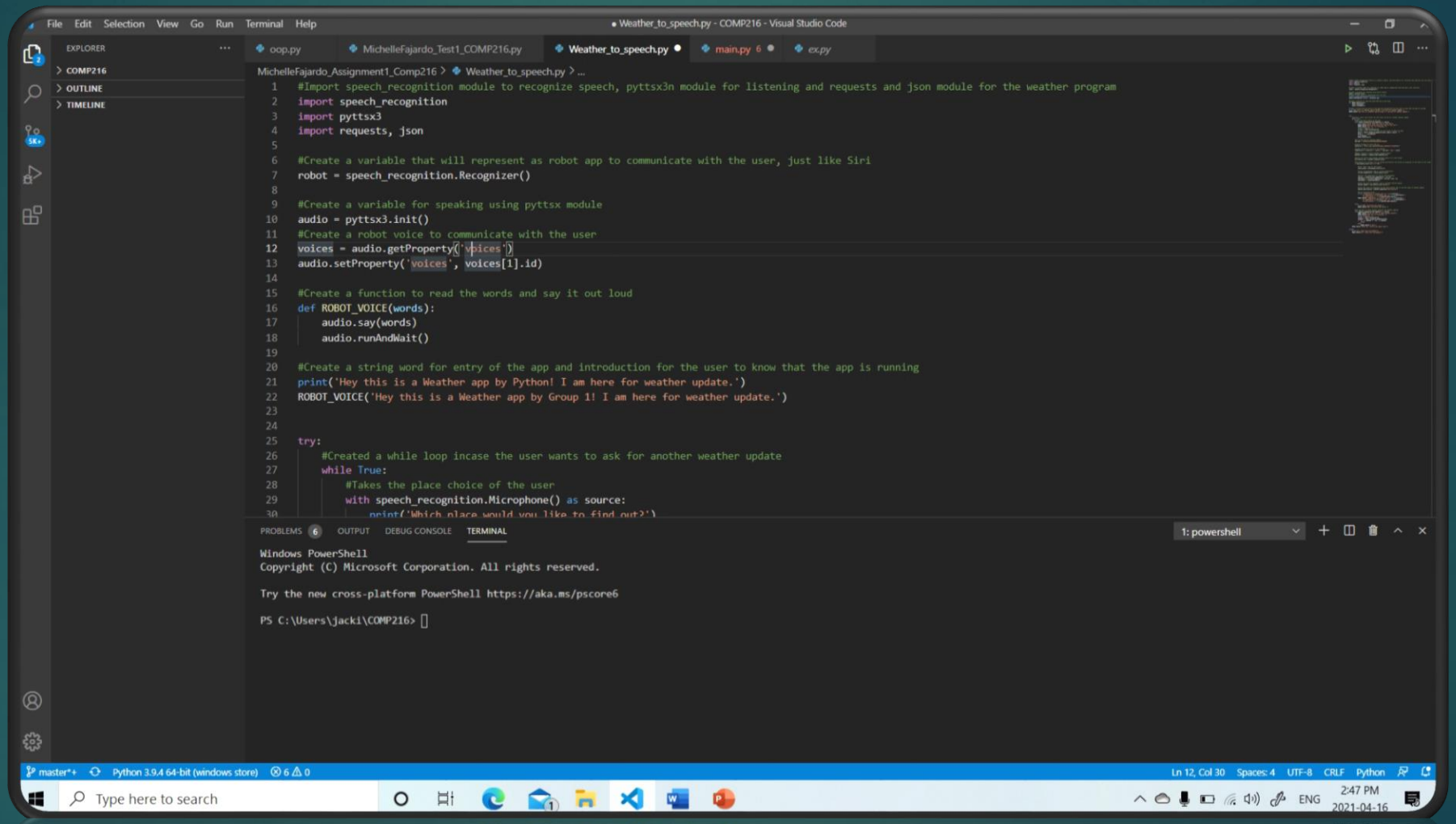
3



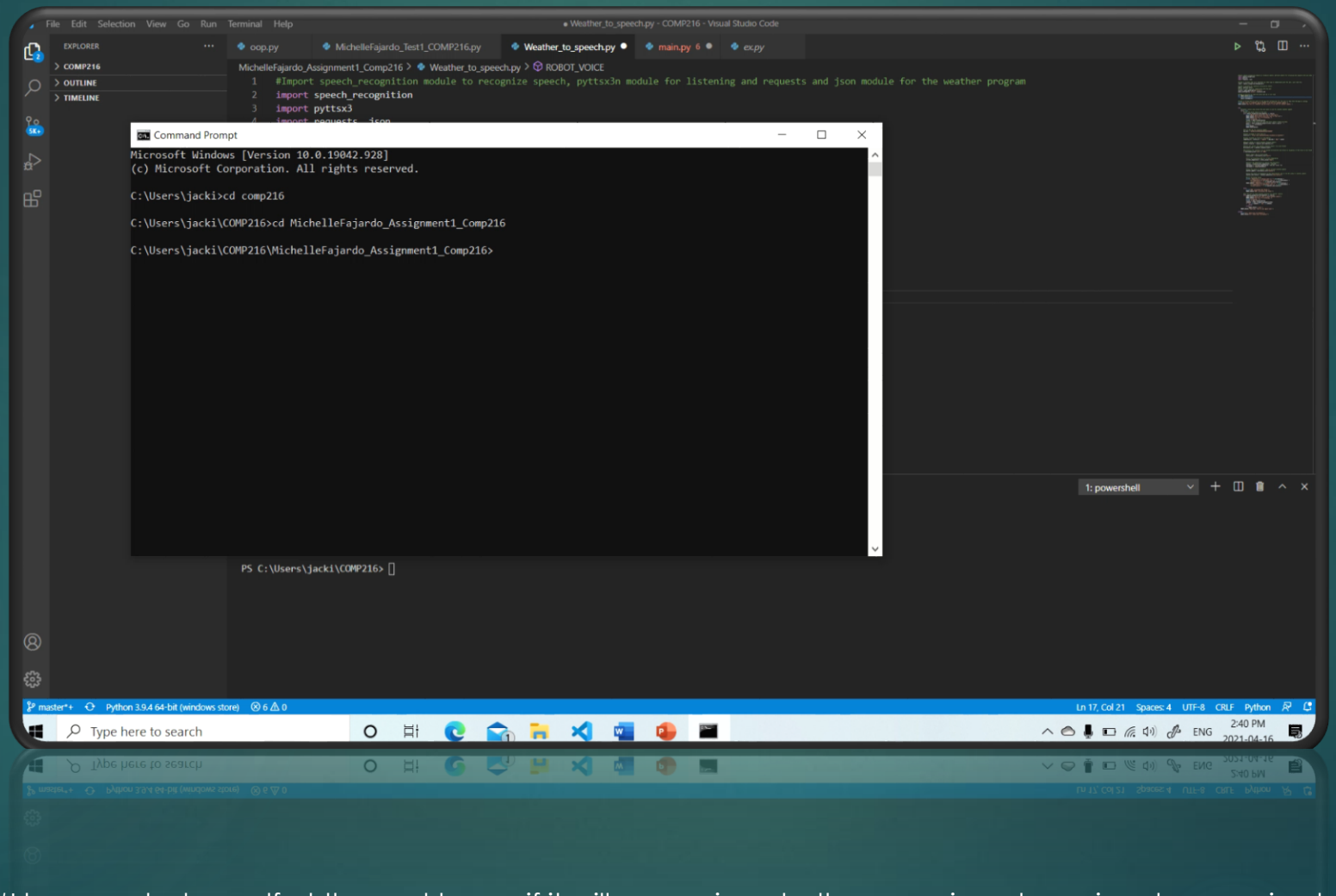
3



Demonstration on how the code looks like at the beginning of the project



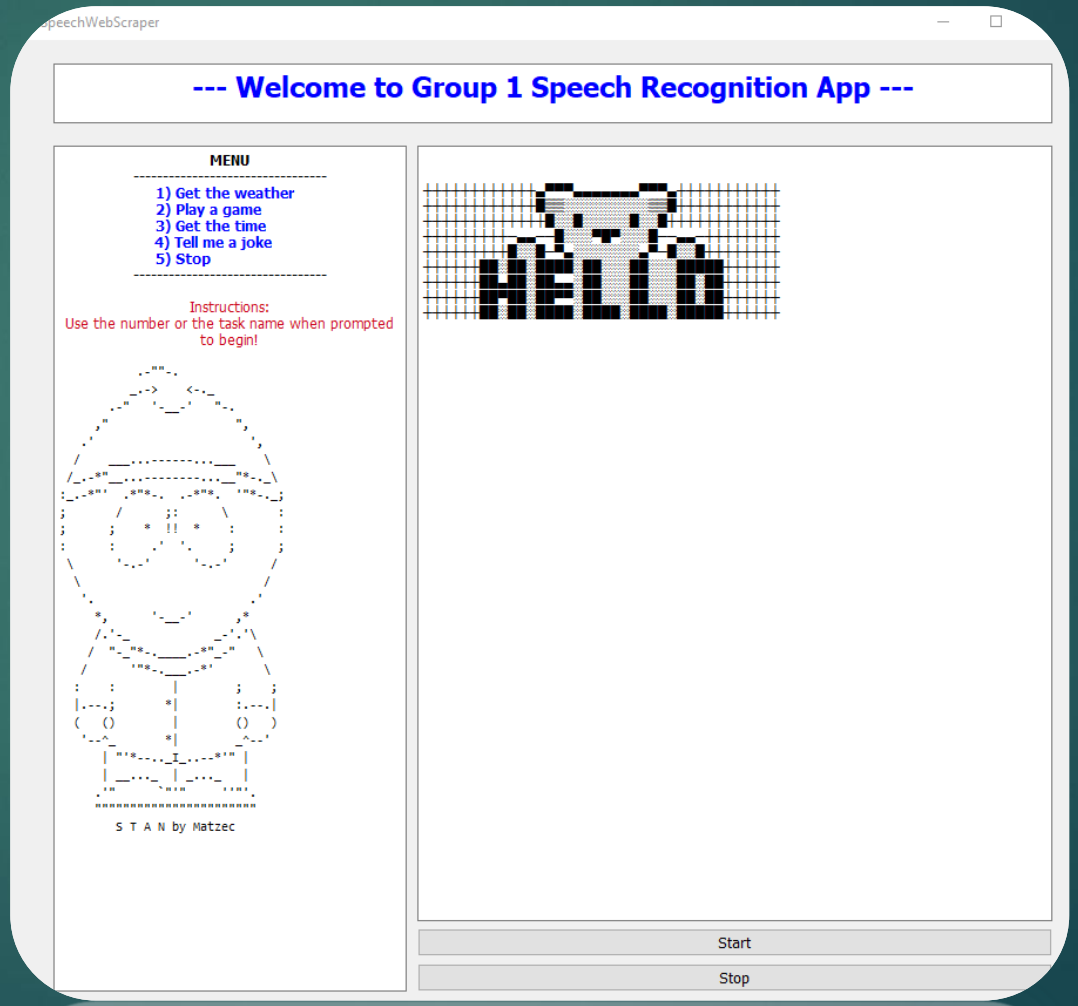
Demo of the first code



#I have muted myself at the end to see if it will recognize whether my microphone is not recognized or not.

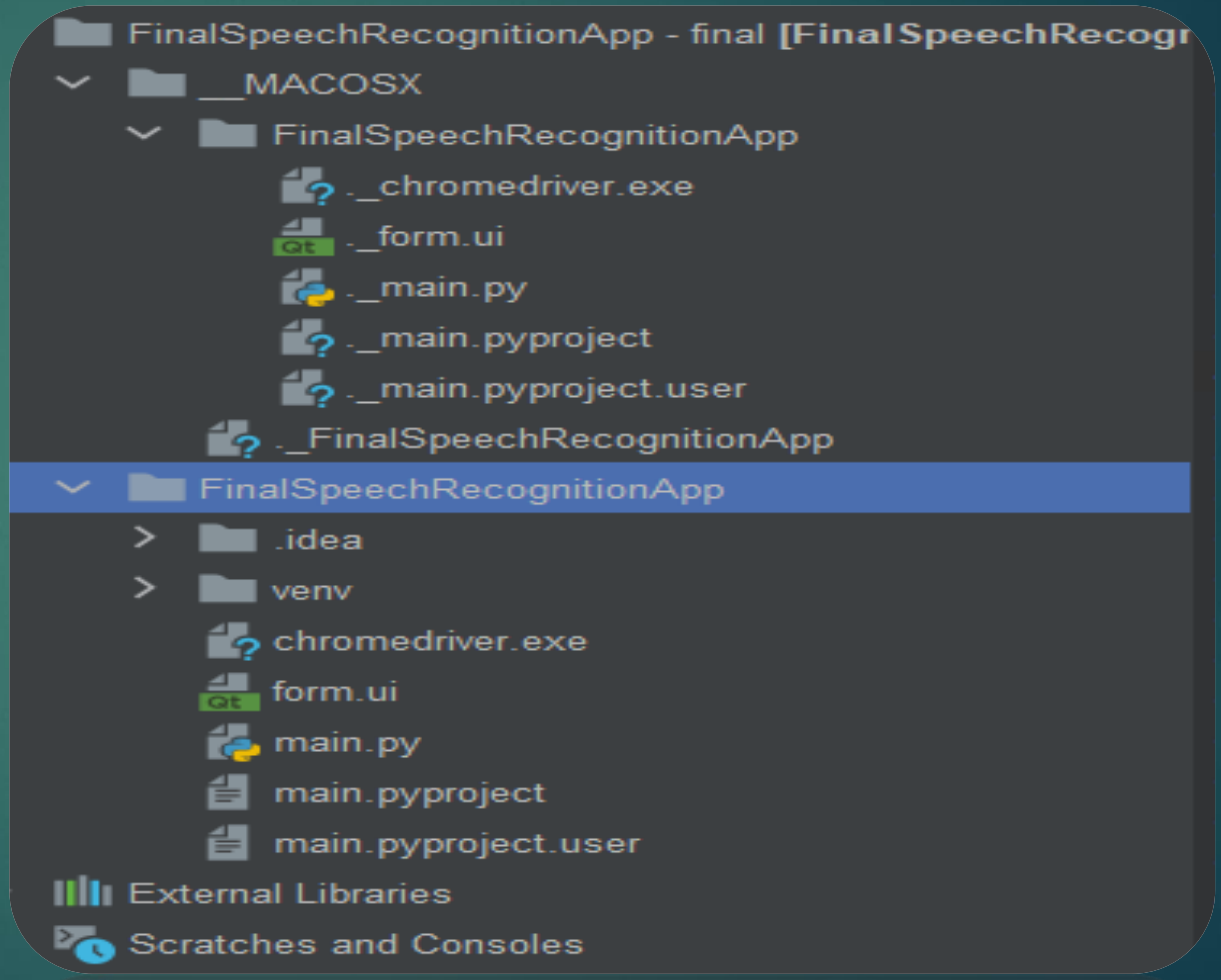
App Functionality

- Weather API
- Game
- Time
- Jokes



MacOS Compatibility

Programmed With MacOS Compatibility



Demo of the Application including all functionality , displayed in GUI

Main Packages


```python
import speech_recognition
import pyttsx3
from selenium import webdriver
from bs4 import BeautifulSoup
import pycountry
import requests
import random
import sys
from PyQt5.QtWidgets import QApplication
from PyQt5 import uic
import os
```

Speech_Recognition
Pyttsx3
Selenium
BeautifulSoup
Requests
PyQt5
PyAudio

Performing Speech Recognition

SpeechRecognition 3.8.1

pip install SpeechRecognition

- Library for performing speech recognition, with support for several engines and APIs, online and offline.

- **Speech recognition engine/API support:**

  Google Speech Recognition
  Google Cloud Speech API
  Microsoft Bing Voice Recognition

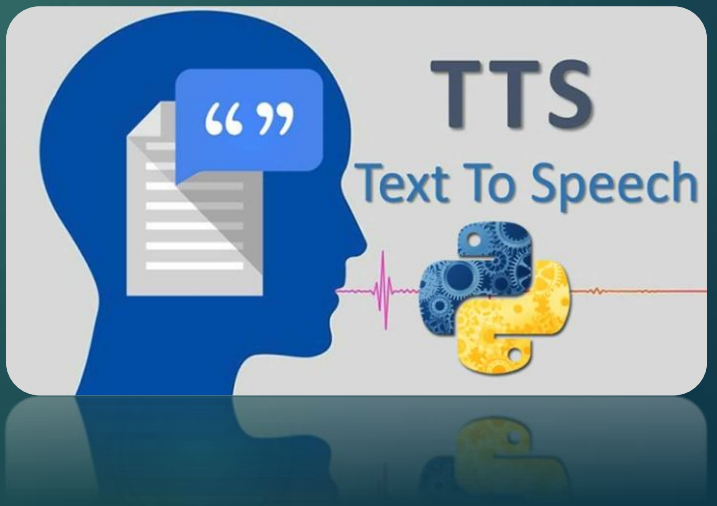
Text to Speech (TTS)

pyttsx3 2.90

pip install pyttsx3

pyttsx3: text-to-speech conversion library in Python.

- Pros:, it works offline
- Compatible with both Python 2 and 3.



Web Scraper

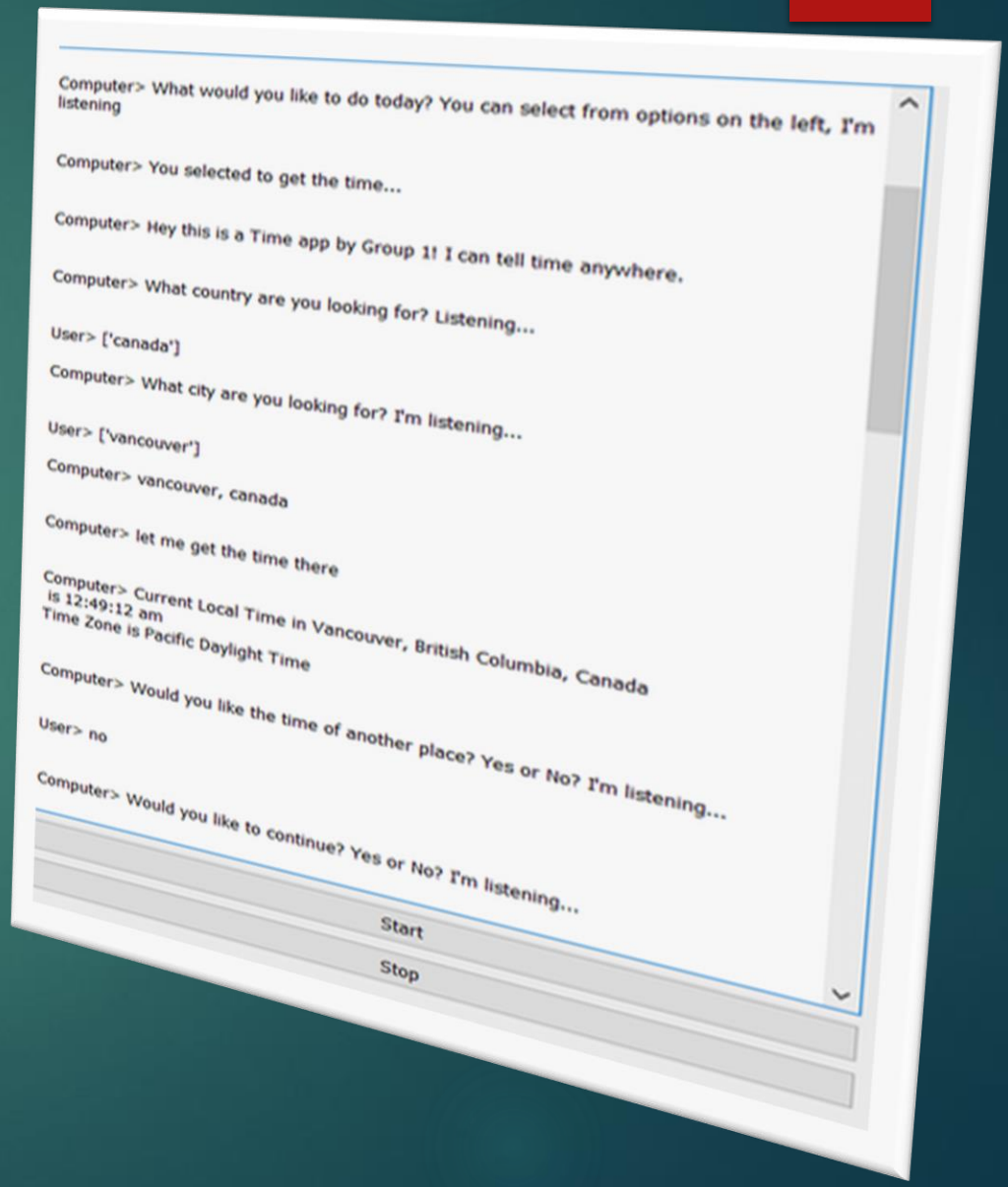
Selenium: Web testing library, Web driver loads web page like a browser would

- pip install selenium

BeautifulSoup: Python library for extracting data out of HTML & XML

- pip install beautifulsoup4

- Selenium was used to get the dynamic HTML code, which was parsed & searched with BeautifulSoup. Requests library was used for all data requests.



API Calls

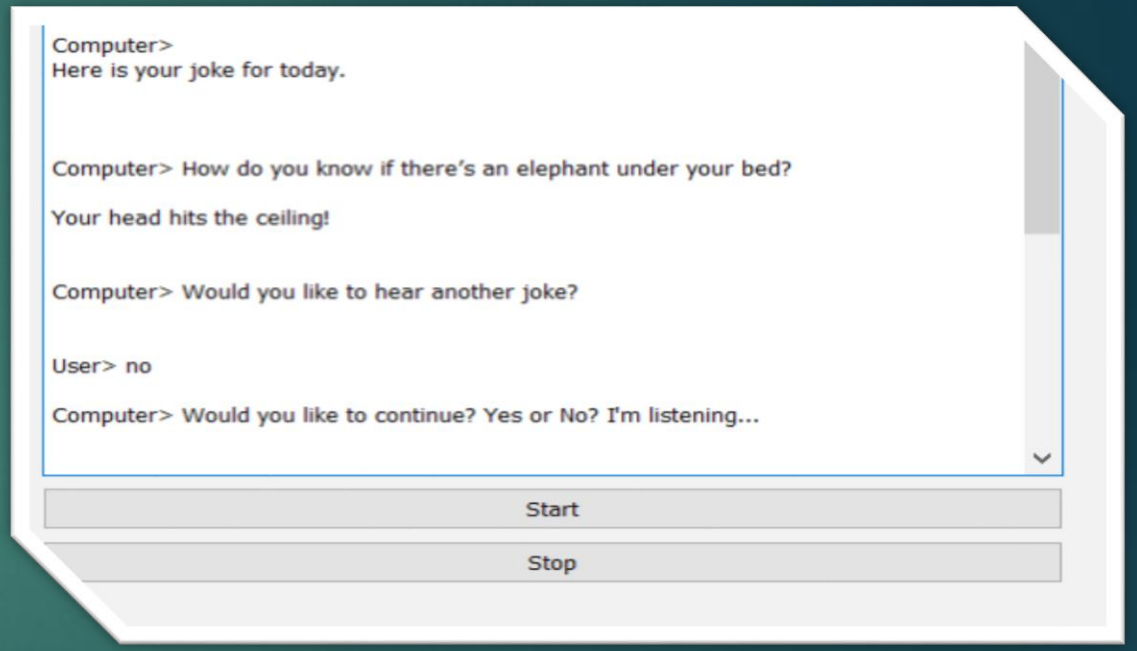
Requests: HTTP library, Send HTTP requests and return a Response Object, Response object easily converted to JSON

- pip install requests

JSON: Module for working with JSON strings and objects, Python native support, Name/Value pair format

- import json

- Requests are used to make API calls and response is parsed as a JSON object with the JSON module



PyQt5 GUI

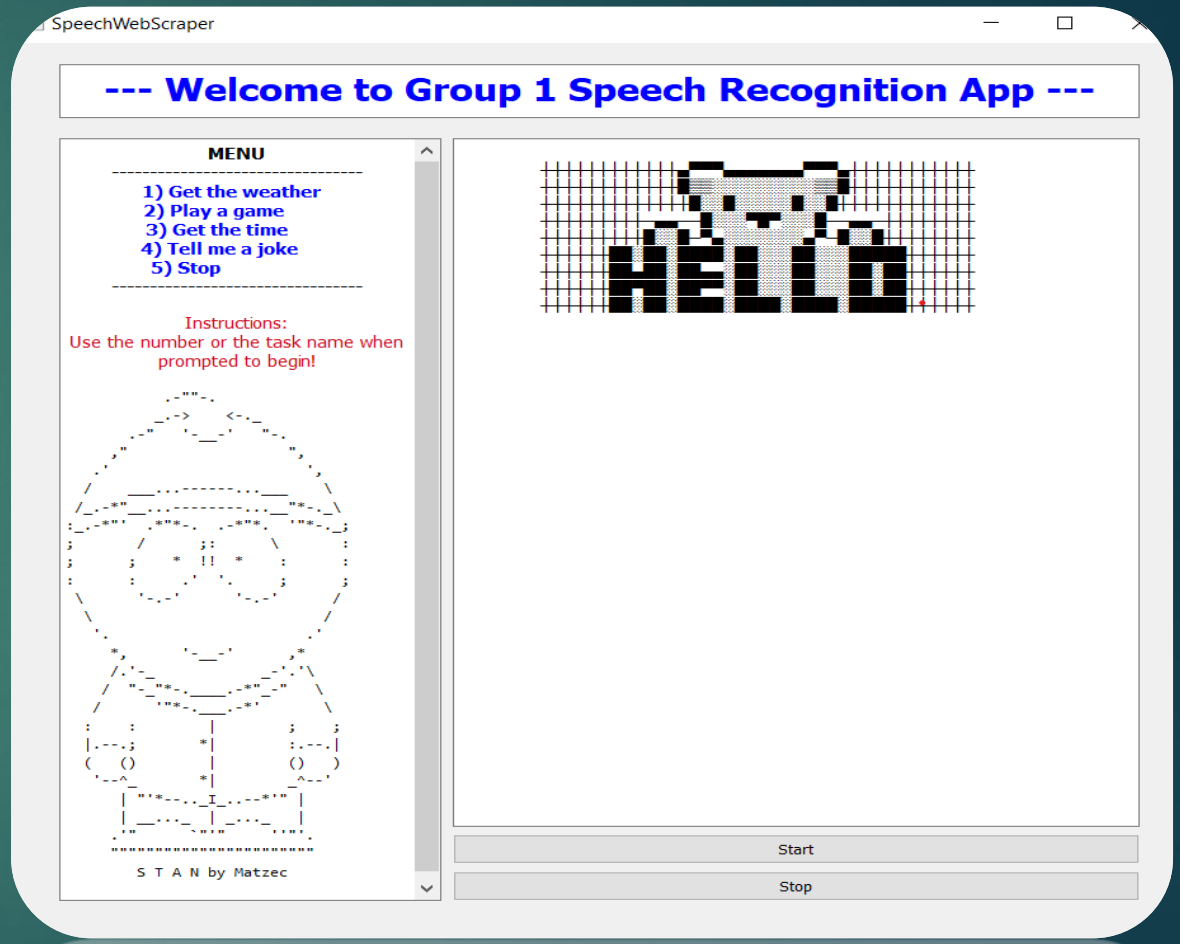
PyQt5: GUI library, Uses Qt framework, Most used GUI builiding module

- pip install pyqt5

QtDesigner: Qt tool, Built in to PyQt5 library, Scratch pad for designing applications

- Found in Lib folder of Python directory on drive

- QtDesigner was used to design the application. The output file, a .ui file, was then used in the application code by importing the uic module from PyQT5 library, with the Qpplication class



Application Breakdown

SpeechWebScraper class inherits from 2 GUI Classes. The constructor initializes & loads the required components. ROBOT_VOICE method allows for text-to-speech speaking

```
20  class SpeechWebScraper(MainWindowBase, MainWindowUI):
21
22      def __init__(self, parent=None):
23          MainWindowBase.__init__(self, parent)
24
25          self.setupUi(self)
26          # Create a variable that will represent as robot app to communicate with the user, just like Siri
27          self.robot = speech_recognition.Recognizer()
28          # Create a variable for speaking using pyttsx module
29          self.audio = pyttsx3.init()
30          # Create a self.robot voice to communicate with the user
31          self.voices = self.audio.getProperty('voices')
32          self.audio.setProperty('voices', self.voices[0].id)
33          self.pushButton_start.clicked.connect(self.menu_selection)
34          self.pushButton_stop.clicked.connect(self.stop_event)
35          self.textBrowser_conversation.append("""
36                  ┼┼┼┼┼┼┼┼┼┼┼┼▄▄▄▄▄▄▄▄┼┼┼┼┼┼┼┼┼┼┼┼
37                  ┼┼┼┼┼┼┼┼┼┼┼┼▄▓▓▓▓▓▓▄┼┼┼┼┼┼┼┼┼┼┼┼
38                  ┼┼┼┼┼┼┼┼┼┼┼┼▓▓▄▓▓▓▓▄▓▓┼┼┼┼┼┼┼┼┼┼┼┼
39                  ┼┼┼┼┼┼┼┼┼┼┼┼▀▄▓▓▄▓▓▄▀▄┼┼┼┼┼┼┼┼┼┼┼┼
40                  ┼┼┼┼┼┼┼┼┼┼┼┼┼┼▓▓▓▓▓▓┼┼┼┼┼┼┼┼┼┼┼┼┼┼
41                  ┼┼┼┼┼┼┼┼┼┼┼┼▄▓▓▄▓▓▄▓▓▄▓▓▄┼┼┼┼┼┼┼┼┼┼
42                  ┼┼┼┼┼┼┼┼┼┼┼┼▓▓▄▀▓▓▄▓▓▄▓▓▄┼┼┼┼┼┼┼┼┼┼
43                  ┼┼┼┼┼┼┼┼┼┼┼┼▓▓▀▄▓▓▄▓▓▄▓▓▄┼┼┼┼┼┼┼┼┼┼
44                  ┼┼┼┼┼┼┼┼┼┼┼┼▓▓▄▓▓▄▓▓▄▓▓▄▓▓┼┼┼┼┼┼┼┼┼┼
45          """)
46          self.continueLoop = True
47
48      # Create a function to read the words and say it out loud
49      def ROBOT_VOICE(self, words):
50          self.audio.say(words)
51          self.audio.runAndWait()
```

Location_getter: method

Location_getter method uses speech recognition to get user input of city and country and returns them as 2 string variables. Used in the weather and time modules.

```
307     def location_getter(self, source):
308         while True:
309             self.general_display('What country are you looking for? Listening...')
310             # Try condition for invalid country and unrecognized speech exceptions
311             try:
312                 listen = self.robot.listen(source)
313                 # listen to the voice and take the first word if there are many
314                 place = self.robot.recognize_google(listen).lower().split(' ')
315                 country = '-'.join(place)
316                 self.textBrowser_conversation.append(f"\nUser> {place}")
317                 # If statement validates input country with pycountry module
318                 if pycountry.countries.search_fuzzy(country):
319                     # Url takes america as usa, so this if condition check for that
320                     if pycountry.countries.search_fuzzy(country) == pycountry.countries.search_fuzzy("America"):
321                         country = "usa"
322                     break
323                 else:
324                     # Raise error if country is not validated
325                     raise LookupError
326             except LookupError or AttributeError:
327                 # Raised if country not validated
328                 self.general_display("Invalid Country")
329             except speech_recognition.UnknownValueError:
330                 # Raised if speech not recognized
331                 self.general_display("Please Say a Country")
332         # Get city from the user
333         while True:
334             self.general_display("What city are you looking for? I'm listening...")
335             # Try condition for invalid city and unrecognized speech exceptions
336             try:
337                 listen = self.robot.listen(source)
338                 # listen to the voice and take the first word if there are many
339                 place = self.robot.recognize_google(listen).lower().split(' ')
340                 self.textBrowser_conversation.append(f"\nUser> {place}")
341                 city = '-'.join(place)
342                 self.general_display(city + ", " + country)
343                 break
344             except:
345                 # Raised if speech not recognized
346                 self.general_display("Please Say a City")
347
348         return country, city
```

Time_scraper: method – Scrapes the given website using BeautifulSoup and requests.

BeautifulSoup parses the webpage's code to extract the information we need. Only works on webpages which are not loaded with dynamic code. Implemented with location_getter method in the time module.

```
83
84      # Method to statically web scrape time from user input of country & city
85      def time_scraper(self, country: str, city: str):
86          # Create variable to store url with country and city appended which will be opened
87          URL = 'https://www.timeanddate.com/worldclock/' + country + '/' + city
88          # Request gets page's full code
89          page = requests.get(URL)
90          # Variable creates a soup object based on source code in analyzable format
91          soup = BeautifulSoup(page.content, features="lxml")
92          # Web page's title was a good format so I grabbed it as is
93          title = soup.title.text.strip()
94          # Variables to store extracted weather details
95          time = soup.find(id='ct').text.strip()
96          time_zone = soup.find(id='cta').find('a')['title']
97          return f'{title}\n is {time}\nTime Zone is {time_zone}'
98
```

Weather_scraper: method - dynamically scrapes the given website using requests, Selenium and BeautifulSoup

Selenium loads the full pages code and BeautifulSoup parses the webpage's code to extract the information we need.

```
# Create function to scrape weather data from the web, takes country & city parameters & returns forecast as st
def weather_scraper(self, country: str, city: str):
    # Variable holds path to webdriver for selenium use
    PATH = "./chromedriver.exe"
    # Create variable to store url with country and city appended which will be opened
    URL = 'https://www.timeanddate.com/weather/' + country + '/' + city
    # Option variable to pass driver object to suppress the opening of the browser
    option = webdriver.ChromeOptions()
    option.add_argument('headless')
    # Variable represents Chrome browser
    driver = webdriver.Chrome(PATH, options=option)
    # Webdriver opens url
    driver.get(URL)
    # Page loads generates full code, source code is then stored as a variable
    soup_source = driver.page_source
    # Variable creates a soup object based on source code in analyzable format
    soup = BeautifulSoup(soup_source, features="lxml")
    # Webdriver closed as beautifulsoup will process remaining code
    driver.quit()
    # Web page's title was a good format so I grabbed it as is
    title = soup.title.text.strip()
    # Element id of HTML code block needed to analyze
    focus = soup.find(id='qlook')
    # Variables to store extracted weather details
    temp = focus.find('div', class_='h2').text.strip()
    rest_info = focus.find_all('p')
    condition = rest_info[0].text.strip()
    description = rest_info[1].text.strip().split('F')
    # Create a formatted string variable to return forecast in speakable form
    feels = 'F' + description[1]
    return f'{title}\n {temp}\nCondition is {condition}\n {feels}'
```

Joke Module: method – Uses API request to get joke

Joke Module gets a random joke from an open-source joke API. The request library is used to get the joke and punchline, while json library is used to parse, process and display the joke and punchline.

```
187     def joke_module(self, source):
188         self.continueJoke = True
189         while self.continueJoke:
190             response = requests.get("https://official-joke-api.appspot.com/random_joke").json()
191             self.general_display(f"\nHere is your joke for today.\n")
192             joke = f"{response['setup']} \n\n{response['punchline']}"
193             self.general_display(joke)
194             self.general_display("Would you like to hear another joke?")
195             listen = self.robot.listen(source)
196             command = self.robot.recognize_google(listen)
197             if "no" in command or "exit" in command or "stop" in command or 'nope' in command or '5' in command \
198                     or 'five' in command or 'nop' in command or 'cancel' in command or 'end' in command:
199                 self.continueJoke = False
200                 self.textBrowser_conversation.append(f"\nUser> {command}")
```

Game Module: method – Simple guess-the-random-number game

Game Module is a simple guess-the-randomly-generated-number game within the allotted attempts and integrates taking of user speech from speech recognition.

```
    def game_module(self, source):
        while True:
            num_guess = 0
            number = random.randint(1, 20)  # returns random integer within range
            # print the number so you can see it
            print(number)
            self.general_display('I am thinking of a number between 1 and 20.')
            self.general_display("How many attempts would you like to guess the number? I'm listening ...")
            listen = self.robot.listen(source)
            attempts = self.robot.recognize_google(listen)
            attempts = int(attempts)
            self.textBrowser_conversation.append(f'\nUser> {attempts}')
            guess = 0
            while num_guess < attempts:
                try:
                    self.general_display("Guess a number. I'm listening ")
                    listen = self.robot.listen(source)
                    guess = self.robot.recognize_google(listen)
                    guess = int(guess)
                    self.textBrowser_conversation.append(f'\nUser> {guess}')
                    num_guess += 1
                    if guess < number:
                        self.general_display('Your guess is too low.')
                    if guess > number:
                        self.general_display('Your guess is too high.')
                    if guess == number:
                        self.win_msg(num_guess)
                        break
                except:
                    # Generic exception block w/o error type catch all errors & mic speech_recognition errors
                    self.general_display("Something went Wrong. Please try again!")
            if guess != number:
                number = str(number)
                self.lose_msg(number)
            # Ask the user to continue checking other locations for weather updates
            self.general_display("Would you like to play another round? Yes or No? I'm listening...")
            try:
                listen = self.robot.listen(source)
                command = self.robot.recognize_google(listen)
                if "no" in command or "exit" in command or "stop" in command or 'nop' in command or 'cancel' in command or 'end' in command:
                    self.textBrowser_conversation.append(f"\nUser> {command}\n")
                    break
                else:
                    self.general_display('I see!')
```

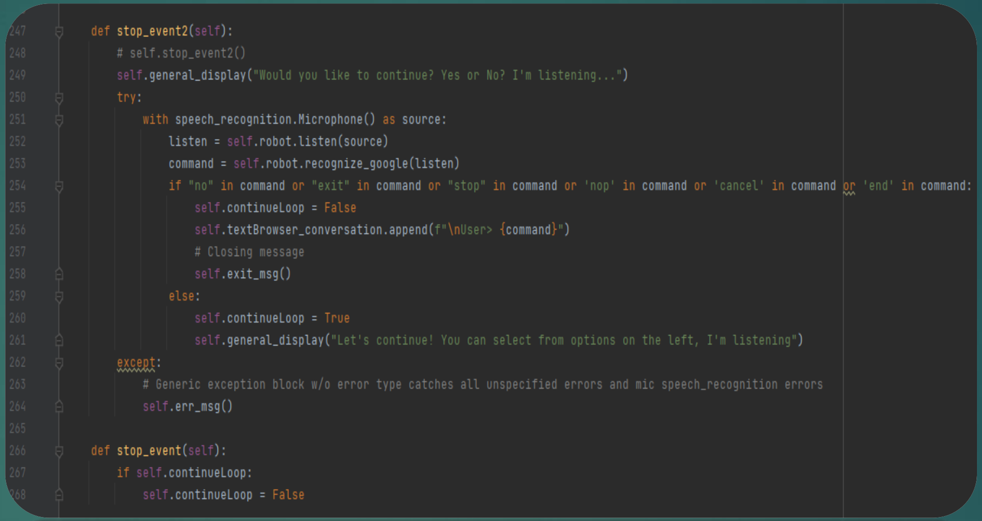
Menu_selection : method - Application loop allows operation of multiple modules in a single run

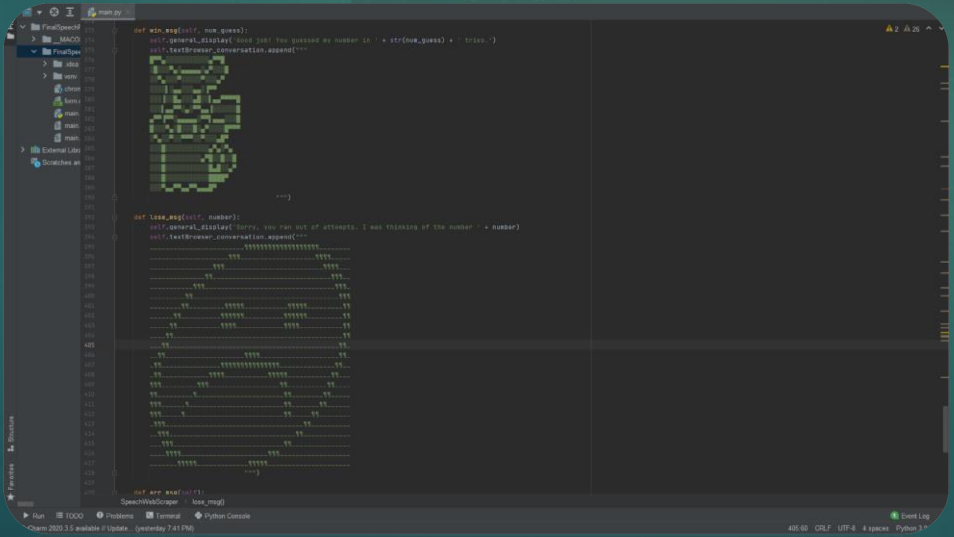
Menu_selection method allows the user to select the function they would like to use and opens the corresponding module. Method loop will ask the user to continue exploring functionality or exit loop and end app.

```
    def menu_selection(self):
        self.general_display("What would you like to do today? You can select from options on the left, I'm listening")
        try:
            with speech_recognition.Microphone() as source:
                while self.continueLoop:
                    listen = self.robot.listen(source)
                    command = self.robot.recognize_google(listen)
                    if "no" in command or "exit" in command or "stop" in command or 'nope' in command or '5' in command \
                            or 'five' in command or 'nop' in command or 'cancel' in command or 'end' in command:
                        self.continueLoop = False
                        self.textBrowser_conversation.append(f"\nUser> {command}")
                        self.exit_msg()
                    elif "weather" in command or "get the weather" in command or "1" in command or "one" in command:
                        self.continueLoop = True
                        self.general_display("You selected to get the weather...")
                        self.weather_module()
                    elif "time" in command or "get the time" in command or "3" in command or "three" in command:
                        self.continueLoop = True
                        self.general_display("You selected to get the time...")
                        self.time_module()
                    elif "game" in command or "play a game" in command or "play" in command or "2" in command or "two" in command:
                        self.continueLoop = True
                        self.general_display("You selected to play a game...")
                        self.game_module(source)
                    elif "joke" in command or "tell me a joke" in command or "please tell me a joke" in command or "4" in command or "four" in command:
                        self.joke_module(source)
                    else:
                        self.continueLoop = True
                        self.general_display('I did not understand so I will tell you a joke!')
                        self.joke_module()

                    self.stop_event2()
        except:
            # Generic exception block without error type catches all unspecified errors and mic speech_recognition errors
            self.err_msg()
```

Adding more Features





Thanks
For
Watching!

+ REFERENCES

https://icehp.org/application/files/8915/0394/7405/image.JPG

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwabisabilearning.com%2Fblogs%2Finquiry%2F100-awesome-essential-questions&psig=AOvVaw2kYSR0A6OyJko0wKFvF2pf&ust=1618195040358000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNCOrryU9e8CFQAAAAAdAAAAABBC

https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40butteredwaffles%2Fpython-packages-and-modules-explained-part-1-ff304c4f19dd&psig=AOvVaw0dOUCBTRSr0lRgD4GCgA_y&ust=1618196104832000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNi567eY9e8CFQAAAAAdAAAAABAE

https://www.google.com/url?sa=i&url=https%3A%2F%2Fpy-pkgs.org%2F03-how-to-package-a-python.html&psig=AOvVaw0dOUCBTRSr0lRgD4GCgA_y&ust=1618196104832000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNi567eY9e8CFQAAAAAdAAAAABAJ

https://i.morioh.com/55e6442638.png