



LearnKartS
A Training Services Company

SOLUTION:

SPEECH RECOGNITION AS PERSONAL VOICE AI ASSISTANCE

Steps to perform:

1. First install all the required modules using pip.

Before we proceed, we will need to install some external modules.

- ✓ gTTS – Google Text To Speech, for converting the given text to speech
- ✓ speech_recognition – for recognizing the voice command and converting to text
- ✓ selenium – for web based work from browser
- ✓ wolframalpha – for calculation given by user
- ✓ playsound – for playing the saved audio file.
- ✓ pyaudio – for voice engine in python

Commands to install the packages

```
pip install gTTS
```

```
pip install SpeechRecognition
```

```
pip install selenium
```

```
pip install wolframalpha
```

```
pip install playsound
```

```
conda install pyaudio
```

2. Well, Now let's get started with code. We will divide each function as a single code for easy understanding.

Here's the main function, with readCommand() and talk function.

- ✓ **readCommand()** function is created to get the audio from user using microphone

```
def readCommand():
```

```
    r = sr.Recognizer()
```

```
    audio = "
```

```
with sr.Microphone() as source:
```

```
    print("Speak...")
```

```
    audio = r.listen(source, phrase_time_limit=5)
```

```
    print("Stop.")
```

```
    try:
```

```
        text = r.recognize_google(audio,language='en-US')
```

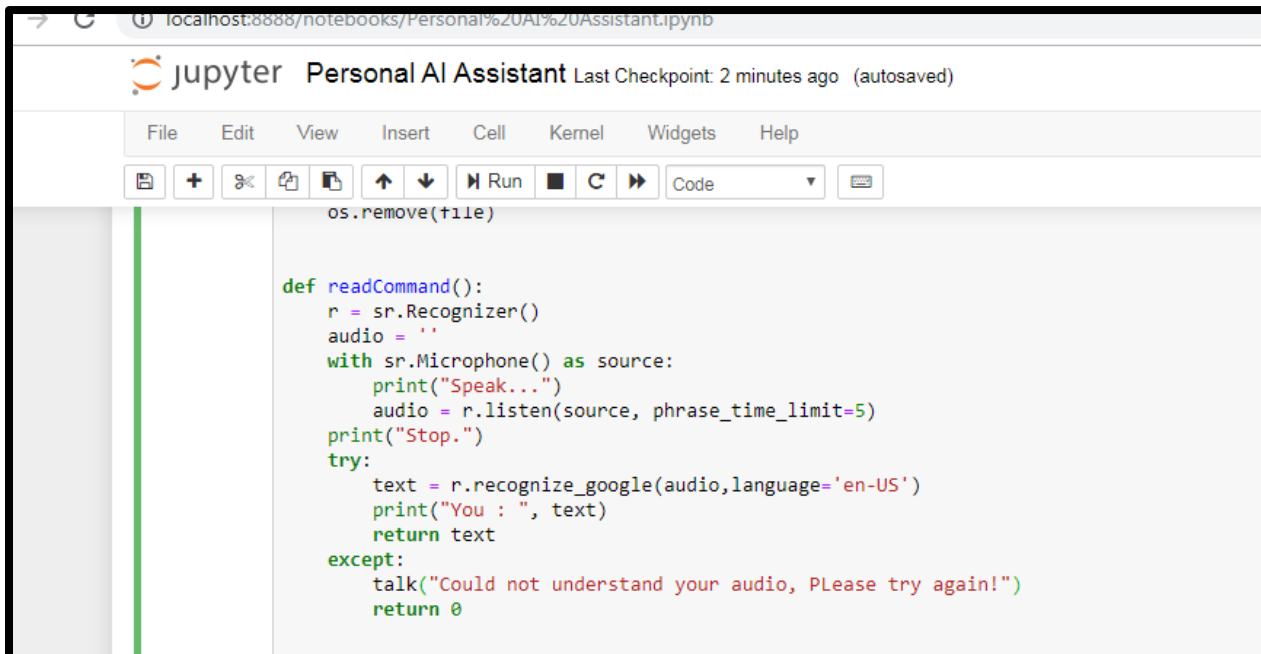
```
        print("You : ", text)
```

```
    return text
```

```
    except:
```

```
        talk("Could not understand your audio, Please try again!")
```

```
    return 0
```

A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Personal%20AI%20Assistant.ipynb'. The notebook title is 'Personal AI Assistant' with a subtitle 'Last Checkpoint: 2 minutes ago (autosaved)'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The toolbar contains icons for saving, adding cells, undo, redo, running, and other standard Jupyter actions. The code editor shows the following Python code:

```
os.remove(t11e)

def readCommand():
    r = sr.Recognizer()
    audio = ''
    with sr.Microphone() as source:
        print("Speak...")
        audio = r.listen(source, phrase_time_limit=5)
    print("Stop.")
    try:
        text = r.recognize_google(audio,language='en-US')
        print("You : ", text)
        return text
    except:
        talk("Could not understand your audio, Please try again!")
        return 0
```

✓ talk() function is created to provide the output according to the processed data.

```
def talk(audioString):
```

```
    global num
```

```
num +=1
```

```
print("You : ", audioString)
```

```
toSpeak = gTTS(text=audioString, lang='en-US', slow=False)
```

```
file = str(num)+".mp3"
```

```
toSpeak.save(file)
```

```
'''mixer.init()
```

```
mixer.music.load('D:\Speech\\audio\spoken.mp3')
```

```
mixer.music.play()
```

```
time.sleep(5)
```

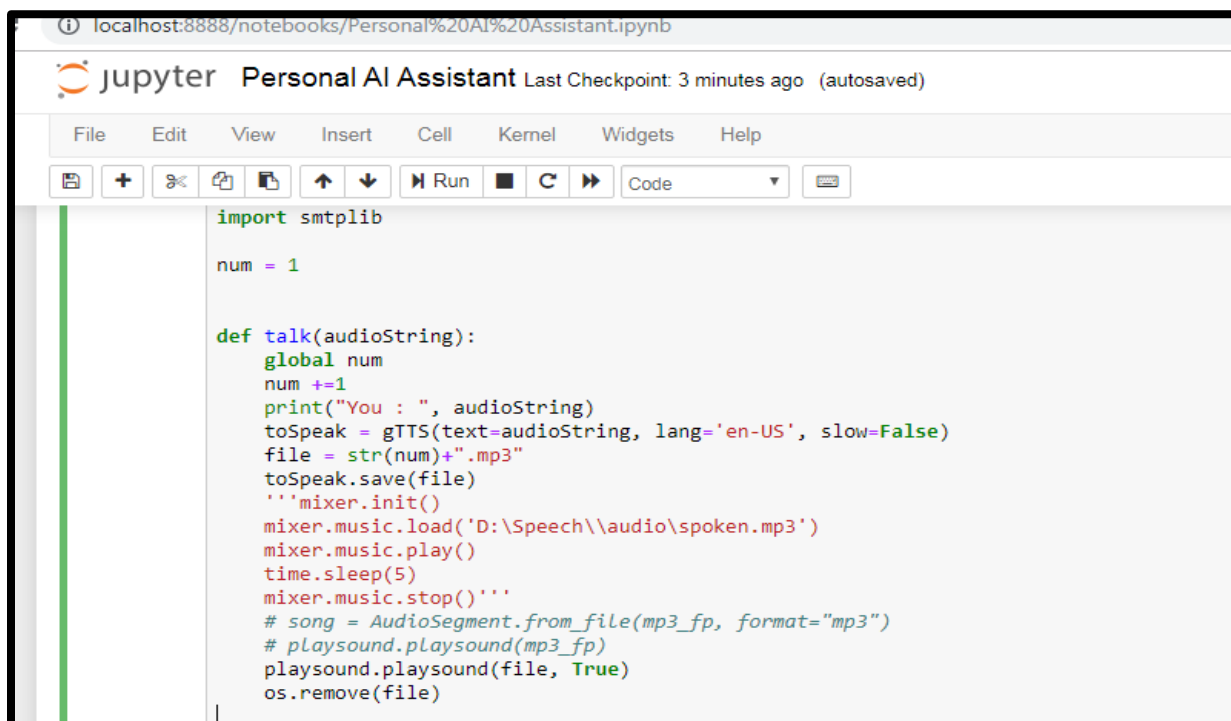
```
mixer.music.stop()'''
```

```
# song = AudioSegment.from_file(mp3_fp, format="mp3")
```

```
# playsound.playsound(mp3_fp)
```

```
playsound.playsound(file, True)
```

```
os.remove(file)
```



The screenshot shows a Jupyter Notebook titled "Personal AI Assistant" running on a local host. The notebook contains a Python script that defines a function named `talk`. This function takes an `audioString` as input and performs several actions: it increments a global counter `num`, prints a message, uses `gTTS` to generate speech, saves it as an MP3 file, initializes a mixer, loads the file, plays it, waits for 5 seconds, stops it, and then uses `AudioSegment` and `playsound` to play the file before deleting it. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, running, and other notebook functions.

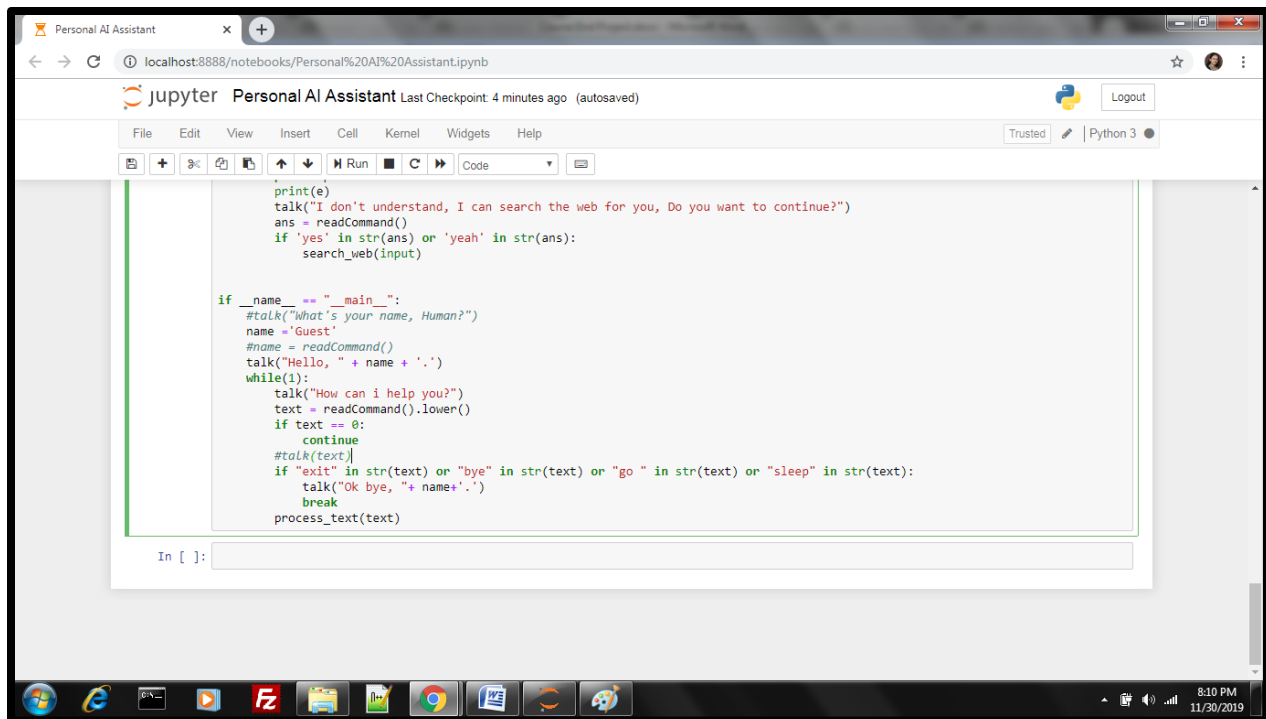
```
import smtplib

num = 1

def talk(audioString):
    global num
    num +=1
    print("You : ", audioString)
    toSpeak = gTTS(text=audioString, lang='en-US', slow=False)
    file = str(num)+".mp3"
    toSpeak.save(file)
    '''mixer.init()
    mixer.music.load('D:\Speech\\audio\spoken.mp3')
    mixer.music.play()
    time.sleep(5)
    mixer.music.stop()'''
    # song = AudioSegment.from_file(mp3_fp, format="mp3")
    # playsound.playsound(mp3_fp)
    playsound.playsound(file, True)
    os.remove(file)
```

✓ this is how we will call these two functions from our main function

```
if __name__ == "__main__":  
    #talk("What's your name, Human?")  
    name = 'Guest'  
    #name = readCommand()  
    talk("Hello, " + name + '.')  
    while(1):  
        talk("How can i help you?")  
        text = readCommand().lower()  
        if text == 0:  
            continue  
        #talk(text)  
        if "exit" in str(text) or "bye" in str(text) or "go " in str(text) or "sleep" in str(text):  
            talk("Ok bye, " + name + '.')  
            break
```



3. So, above we have shown the cases how we are giving the input voice to system and take input from user. Now we are going to show you how process your sound input and get Output as results . Here we are going to use "Wolframalpha api" to calculate the calculations part.

```
def process_input(input):
```

```
    try:
```

```
        if "tell me about yourself" in input or "brief me about you" in input:
```

```
            speak = "Hello, I am Person. Your personal Assistant.
```

```
            I am here to make your life easier.
```

```
            i can perform various tasks such as sum calculations or opening Apps etcetra as per your command"
```

```
            talk(speak)
```

```
            return
```

```
        elif "who is your creator" in input or "who created" in input:
```

```
speak = "I have been created by Narinder."
```

```
talk(speak)
```

```
return
```

```
elif "From where you belong" in input:
```

```
speak = """"I belong to india.""""
```

```
talk(speak)
```

```
return
```

```
elif "calculate" in input.lower():
```

```
app_id= "E46YXW-T5LG6RT7K7"
```

```
client = wolframalpha.Client(app_id)
```

```
indx = input.lower().split().index('calculate')
```

```
query = input.split()[indx + 1:]
```

```
res = client.query(' '.join(query))
```

```
answer = next(res.results).text
```

```
talk("The answer is " + answer)
```

```
return
```

```
elif 'open' in input:
```

```
openApp(input.lower())
```

```
return
```

```
elif 'search' in input or 'play' in input:
```

```
internetSearch(input.lower())
```

```
return
```

```
elif 'email' in input or 'send message' in input:
```

```
s = smtplib.SMTP('smtp.gmail.com', 587)
```

```
s.starttls()
```

```
s.login("sender_email_id", "sender_email_id_password")
```

```
message = "Message_you_need_to_send"
```

```
s.sendmail("sender_email_id", "receiver_email_id", message)
```

```
s.quit()
```

```
else:
```

```
talk("I can perform web search for you, Can i do it now?")
```

```
ans = readCommand()
```

```
if 'yes' in str(ans) or 'yeah' in str(ans):
```

```
    internetSearch(input)
```

```
else:
```

```
    return
```

```
except Exception as e:
```

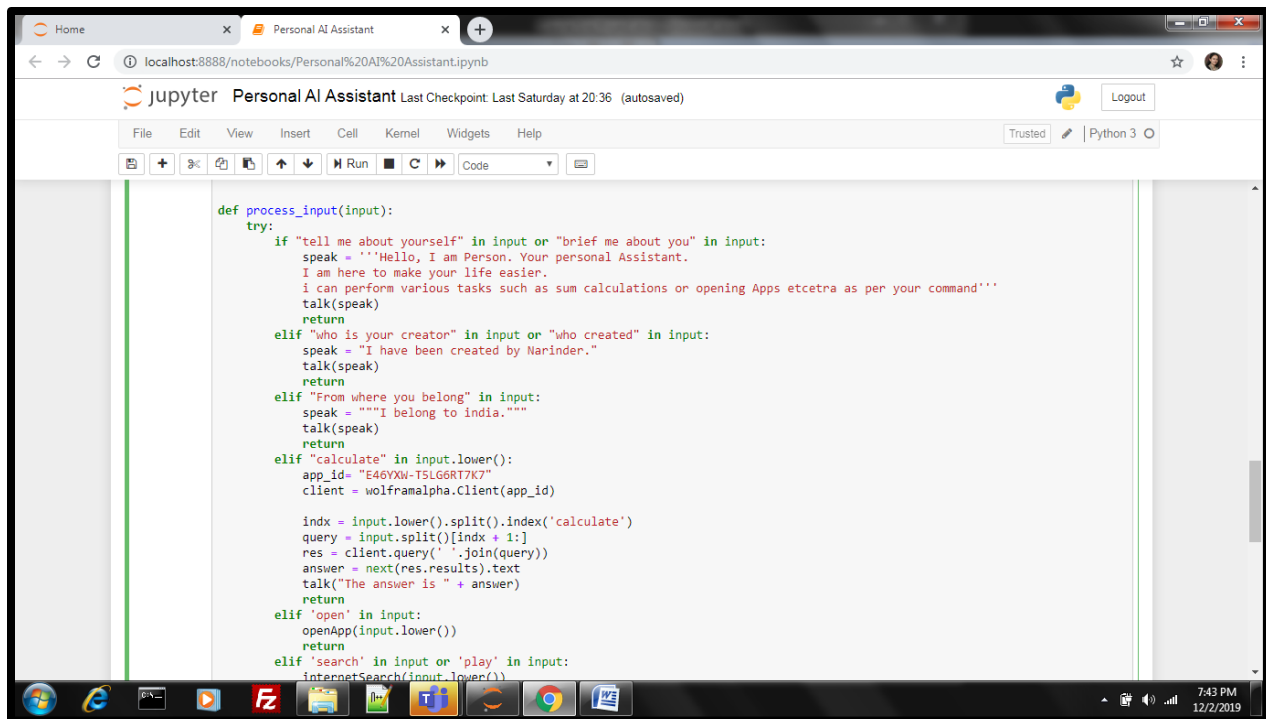
```
    print(e)
```

```
talk("I don't understand, I can perform web search for you, Can i do it now?")
```

```
ans = readCommand()
```

```
if 'yes' in str(ans) or 'yeah' in str(ans):
```

```
    internetSearch(input)
```

4. Well since we have processed the information now we need to take necessary actions like Doing internet Search, Opening installed applications or sending email Events , We are going to define here two functions ;- openApp() & internetSearch()

✓ " internetSearch " just act like a web crawler in which we are using selenium package to process the incoming inputs. It will process the coming input text and search in chrome, wikipedia youtube etc , You can define your own set of information sources from where you want to do search

```
def internetSearch(input):
```

```
    driver = webdriver.Firefox()
```

```
    driver.implicitly_wait(1)
```

```
    driver.maximize_window()
```

```
    if 'youtube' in input.lower():
```

```
        talk("Opening in youtube")
```

```
        indx = input.lower().split().index('youtube')
```

```
        query = input.split()[indx+1:]
```

```
driver.get("http://www.youtube.com/results?search_query=" + '+'.join(query))
```

```
return
```

```
elif 'wikipedia' in input.lower():
```

```
    talk("Opening Wikipedia")
```

```
    indx = input.lower().split().index('wikipedia')
```

```
    query = input.split()[indx + 1:]
```

```
    driver.get("https://en.wikipedia.org/wiki/" + '_'.join(query))
```

```
    return
```

```
else:
```

```
    if 'google' in input:
```

```
        indx = input.lower().split().index('google')
```

```
        query = input.split()[indx + 1:]
```

```
        driver.get("https://www.google.com/search?q=" + '+'.join(query))
```

```
    elif 'search' in input:
```

```
        indx = input.lower().split().index('google')
```

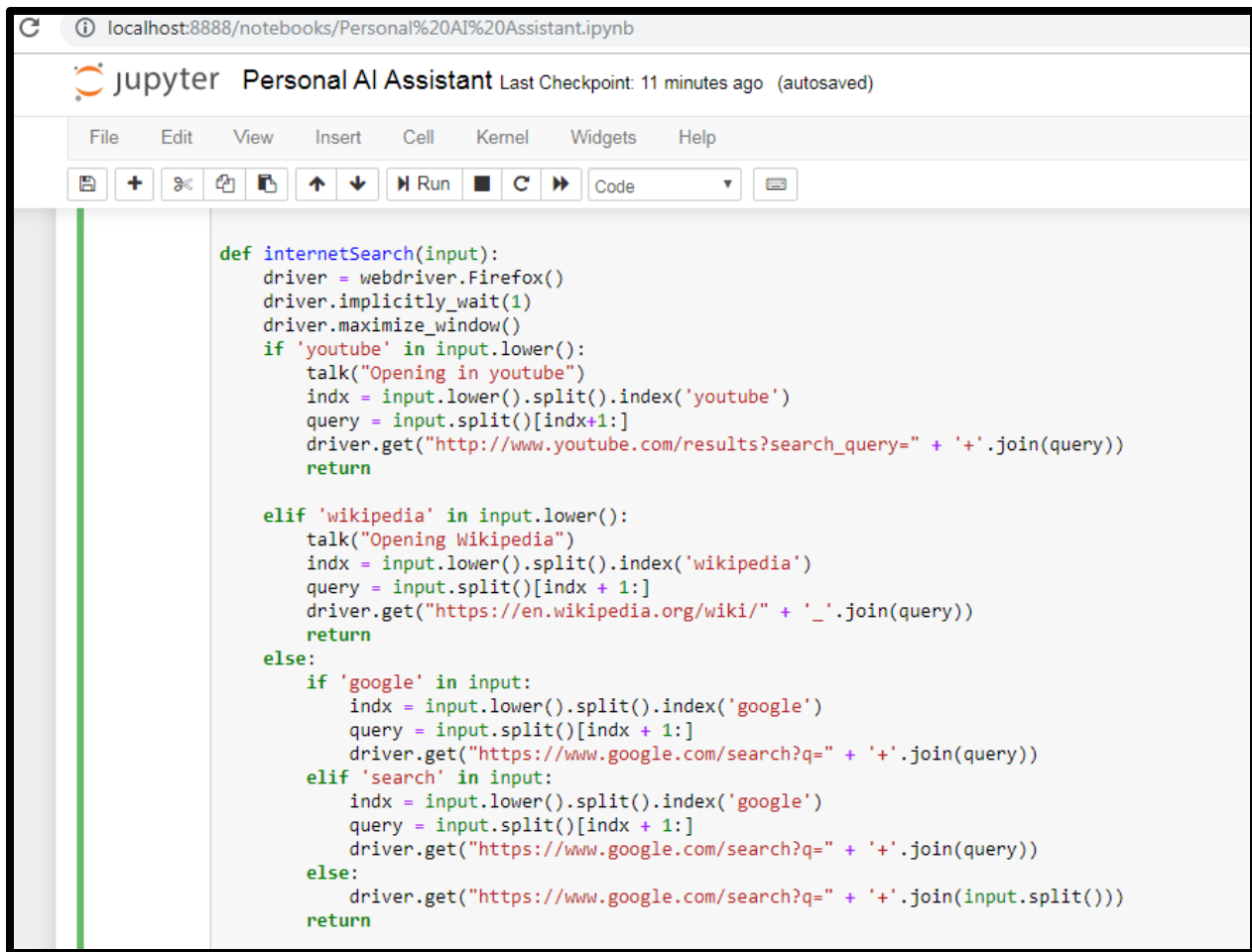
```
        query = input.split()[indx + 1:]
```

```
        driver.get("https://www.google.com/search?q=" + '+'.join(query))
```

```
    else:
```

```
        driver.get("https://www.google.com/search?q=" + '+'.join(input.split()))
```

```
    return
```



The screenshot shows a Jupyter Notebook titled "Personal AI Assistant" running on a local host. The notebook contains a Python function named `internetSearch` that uses Selenium to perform searches on YouTube, Wikipedia, or Google based on the input. The function uses `webdriver.Firefox()` and `WebDriverWait` to interact with the browser. It splits the input string to identify the search engine and the search query, then constructs the appropriate URL and performs a `driver.get()` action.

```
def internetSearch(input):
    driver = webdriver.Firefox()
    driver.implicitly_wait(1)
    driver.maximize_window()
    if 'youtube' in input.lower():
        talk("Opening in youtube")
        indx = input.lower().split().index('youtube')
        query = input.split()[indx+1:]
        driver.get("http://www.youtube.com/results?search_query=" + '+'.join(query))
        return

    elif 'wikipedia' in input.lower():
        talk("Opening Wikipedia")
        indx = input.lower().split().index('wikipedia')
        query = input.split()[indx + 1:]
        driver.get("https://en.wikipedia.org/wiki/" + '_'.join(query))
        return

    else:
        if 'google' in input:
            indx = input.lower().split().index('google')
            query = input.split()[indx + 1:]
            driver.get("https://www.google.com/search?q=" + '+'.join(query))
        elif 'search' in input:
            indx = input.lower().split().index('google')
            query = input.split()[indx + 1:]
            driver.get("https://www.google.com/search?q=" + '+'.join(query))
        else:
            driver.get("https://www.google.com/search?q=" + '+'.join(input.split()))
        return
```

Run the code and you will see the response in the form of the output.