

PERSONAL VIRTUAL ASSISTANT: JARVIS USING PYTHON

*A Report submitted in partial fulfillment of the requirements to complete
Term Work & Practical work of Project Based Learning (PBL) in the
department of*

FIRST YEAR ENGINEERING

As prescribed by

SAVITRIBAI PHULE PUNE UNIVERSITY

By

HOLE VIDISH PANDURANG	72286287D
CHAVAN JAYDEEP VIJAY	72286139H
TODKARI SHIVAM VARDHASHANKAR	72286804K
PARDESHI GAYATRI RAJENDRA	72286574M
HARKE ATHARWA SANJAY	72286049J
NINGDALI ABHISHEK HANUMANT	72286556C
PAWAR GANESH AVACHIT	72286607M

Under the supervision of

Prof. A.V. Gunjal



**First Year Engineering Department
SINHGAD COLLEGE OF ENGINEERING**

*44/1, Vadgaon (Bk), Off Sinhgad Road,
Pune - 411041.*

SINHGAD COLLEGE OF ENGINEERING

S.No. 44/1, Vadgaon (Bk), Off Sinhgad Road, Pune – 411 041.


Department of First Year Engineering

Certificate

This is to certify that the following students,

HOLE VIDISH PANDURANG	Roll No: 104016
CHAVAN JAYDEEP VIJAY	Roll No: 104017
TODKARI SHIVAM VARDHASHANKAR	Roll No: 104018
PARDESHI GAYATRI RAJENDRA	Roll No: 104019
HARKE ATHARWA SANJAY	Roll No: 104020
NINGDALI ABHISHEK HANUMANT	Roll No: 104021
PAWAR GANESH AVACHIT	Roll No: 104022

have completed all the Term Work & Practical Work in the subject **Project Based Learning (PBL)** satisfactorily in the department of First Year Engineering as prescribed by Savitribai Phule Pune University, in the academic year 2021 -2022.



Prof. A.V. Gunjal
Faculty-in-charge



Prof. V.K. Chaudhari
Head of Department



Dr. S.D. Lokhande
Principal

Date: 13/06/2022.

INDEX

Sr. No	Title	Page No
1.	Introduction 1.1 Introduction 1.2 Objective 1.3 Purpose 1.4 Scope	4...
2.	Software Details and Resources 2.1 Software Details 2.1.1 Visual Studio Code 2.1.2 Qt Designer 2.2 Programming language: Python 2.3 Libraries and Modules used in Python	9...
3.	System Design 3.1 Work Flow/ Methodology	16...
4.	Source code 4.1 Main code 4.2 Code for Gui	18...
5.	Output Analysis	29...

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Artificial Intelligence shows us the capability of thinking like humans. In this a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Virtual Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the Alexa Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favourite website with the help of a single voice command. In the current scenario, advancement in technologies are such that they can perform any task with same effectiveness or can say more effectively than us. By making this project. I realized that the concept of AI in every field in decreasing human effort and saving time.

As the voice assistant is using Artificial Intelligence hence the result that it is providing and highly accurate and efficient. The assistant can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual us whom we are talking and asking to perform task. The assistant is on less than a human assistant but we can say that this is more effective and efficient to perform any task. The libraries and packages and to make this assistant focuses on the time complexities and reduces time.

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which uses concept of language processing, and voice recognition. They listen the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner. As these voice assistants are using Artificial Intelligence

hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time, but for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

It was an interesting task to make my own assistant. It became easier to Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favourite IDE with the help of a single voice command. Jarvis is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this, it does not require any internet connection while getting the instructions to perform any specific task. The IDE used in this project is VISUAL STUDIO CODE. All the python files were created in VISUAL STUDIO CODE and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e. pyttsx3, SpeechRecognition, Datetime, Wikipedia, pyautogui, PyQt etc. I have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation. With the advancement JARVIS can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project

include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favourite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc. in a web browser, t can have some basic conversation.

1.2 OBJECTIVE

Main objective of building personal assistant software (a virtual assistant) is using data sources available on the web, user generated content and providing knowledge from knowledge data bases. The main purpose of an intelligent virtual assistant is to answer questions that users may have. This may be done in a business environment, for example, on the business website, with a chat interface. On the mobile platform, the intelligent virtual assistant is available as a call-button operated service where a voice asks the user “What can I do for you?” and then responds to verbal input. Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. JARVIS can do that for you. Provide a topic for research and continue with your tasks while JARVIS does the research. One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time¹⁵. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

1.3 PURPOSE

Purpose of virtual assistant is to being capable of voice interaction, music playback, setting alarms, and other real-time information, such as news. Virtual assistants enable users to speak natural language voice commands in order to operate the device and its apps. There is an increased overall awareness and a

higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it's clear that we are moving towards less screen interaction.

1.4 SCOPE

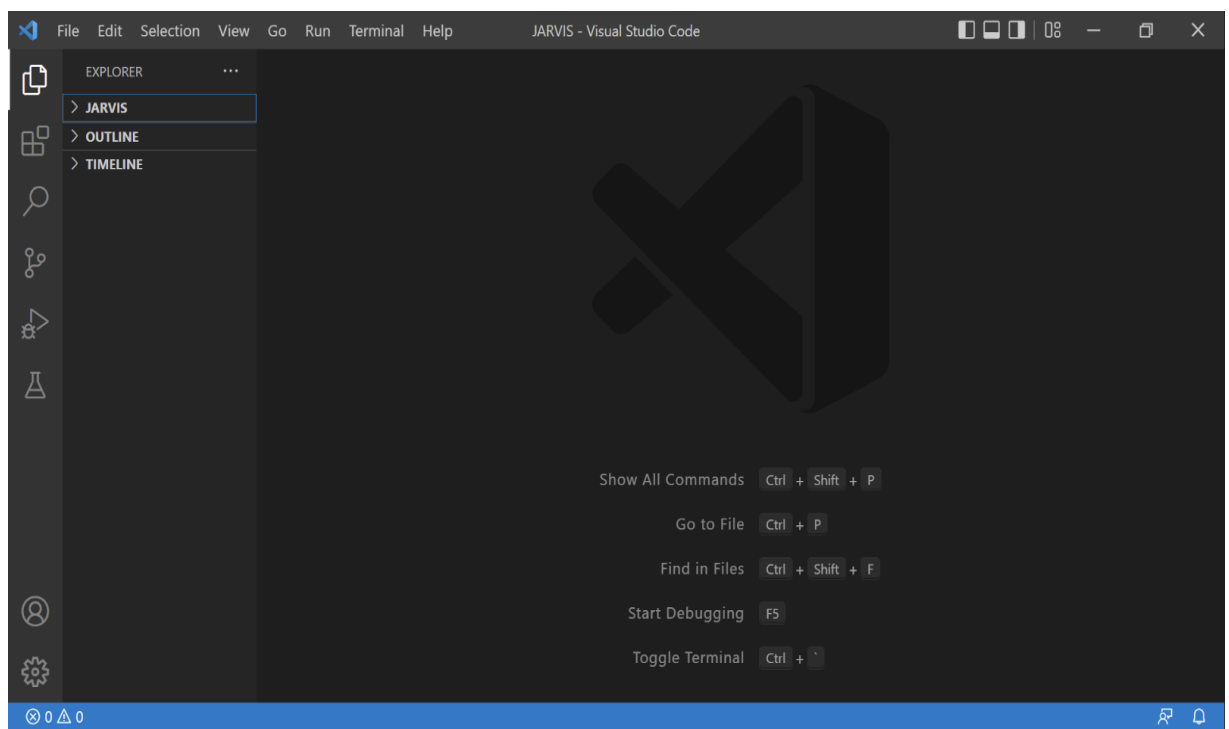
Voice assistants will continue to offer more individualized experiences as they get better at differentiating between voices. However, It's not just developers that need to address the complexity of developing for voice as brands also need to understand the capabilities of each device and integration and if it makes sense for their specific brand. They will also need to focus on maintaining a user experience that is consistent within the coming years as complexity becomes more of a concern. This is because the visual interface with voice assistants is missing. Users simply cannot see or touch a voice interface.

CHAPTER 2: SOFTWARE DETAILS AND RESOURCES

2.1 SOFTWARE DETAILS

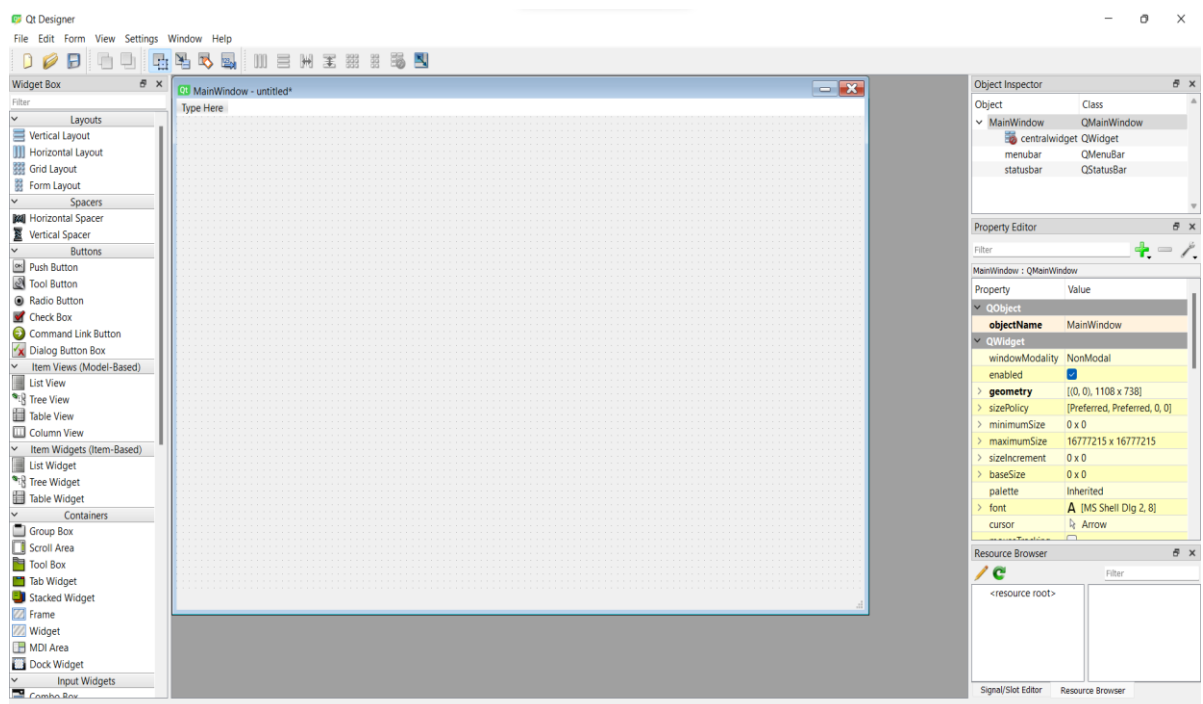
2.1.1 VISUAL STUDIO CODE

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop made by MICROSOFT and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity)



2.1.2 QT DESIGNER

Qt Designer is a tool for quickly building graphical user interfaces with widgets from the Qt GUI framework. It gives you a simple drag-and-drop interface for laying out components such as buttons, text fields, combo boxes and more. Here is a screenshot of Qt Designer on Windows: Qt Designer produces .ui files.



2.2 PROGRAMMING LANGUAGE: PYTHON

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level builtin data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI). Machine Learning (ML). natural language processing, data science etc Python has a lot of libraries for every need of this project.

2.3 LIBRARIES AND MODULES USED IN PYTHON

```
import pyttsx3
import speech_recognition
import datetime
import wikipedia
import webbrowser
from playsound import playsound
import pyautogui
from PIL import Image
import requests
from bs4 import BeautifulSoup
import os
import time
import wolframalpha
import sys
```

2.3.1 pyttsx3:

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible

2.3.2 speech_recognition:

It allows computers to understand human language. Speech recognition is a machine's ability to listen to spoken words and identify them and then convert into text.

2.3.3 datetime:

Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals.

2.3.4 wikipedia:

It is an open-source platform which manages by the community of volunteer editors using a wiki-based editing system. It is a multi-lingual encyclopedia. Python provides the Wikipedia module (or API) to scrap the data from the Wikipedia pages. This module allows us to get and parse the information from Wikipedia.

2.3.5 webbrowser:

The webbrowser module provides a high-level interface to allow displaying web-based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.

2.3.6 playsound:

The playsound module contains only a single function named playsound(). It requires one argument: the path to the file with the sound we have to play. It can be a local file, or a URL

2.3.7 pyautogui:

PyAutoGUI is a Python module which can automate your GUI and programmatically control your keyboard and mouse.

2.3.8 requests:

The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data.

2.3.9 bs4:

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

2.3.10 os:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

2.3.11 time:

The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.

2.3.12 wolframalpha:

Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledgebase and AI technology. It is made possible by the Wolfram Language.

2.3.13sys:

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.

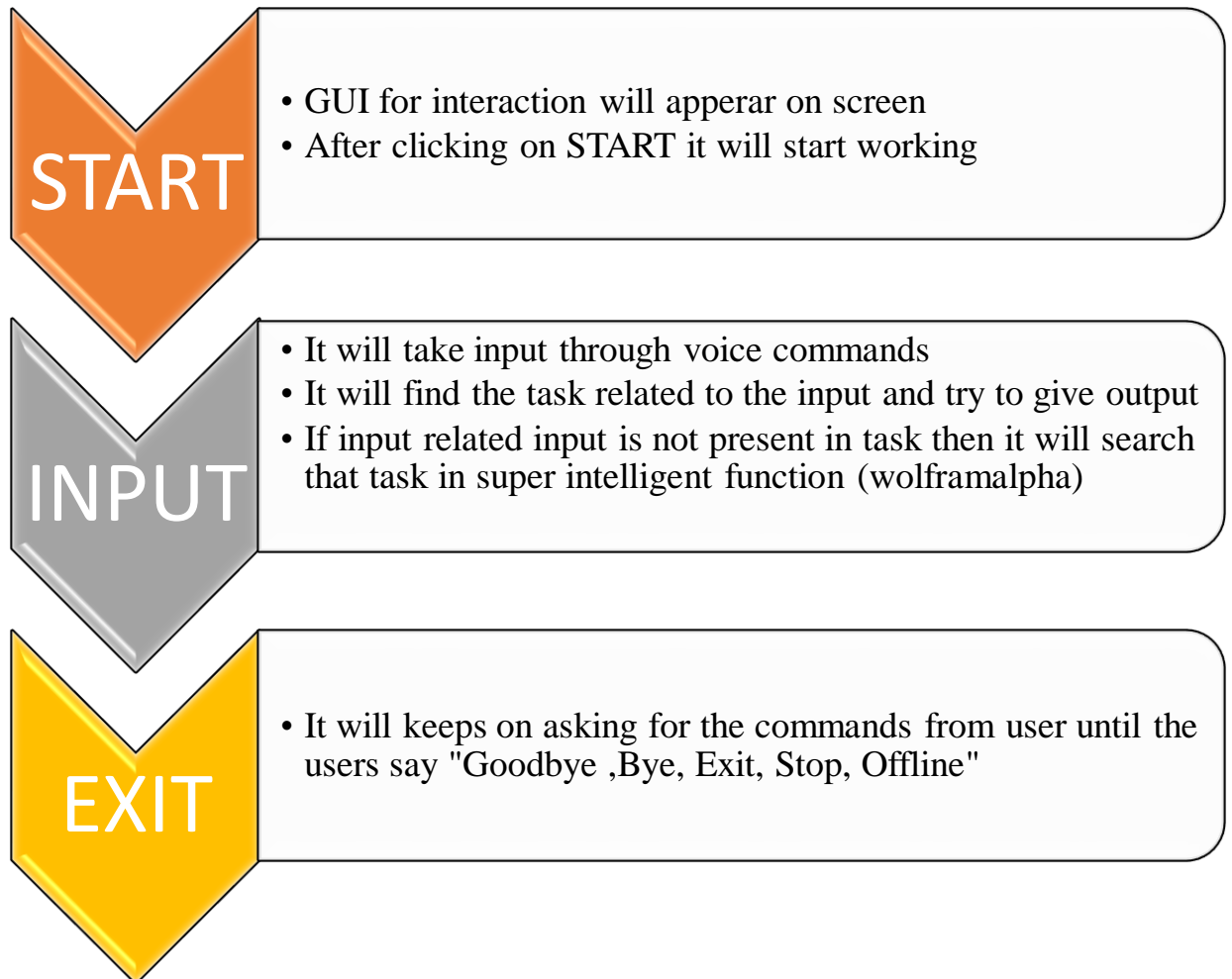
2.3.14 PyQt5:

PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. A GUI application consists of Front-end and Back-end.

CHAPTER 3: SYSTEM DESIGN

3.1 WORK FLOW/ METHODOLOGY

The data flow of JARVIS is as follows:



CHAPTER 4: SOURCE CODE

4.1MAIN CODE

```
import pyttsx3
import speech_recognition
import datetime
import wikipedia
import webbrowser
from playsound import playsound
import pyautogui
from PIL import Image
import requests
from bs4 import BeautifulSoup
import os
import time
import wolframalpha
import sys
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtCore import QTimer , QTime,QDate,Qt
from PyQt5.QtGui import QMovie
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.uic import loadUiType
from jarvisui import Ui_MainWindow
from jarvisui import Ui_MainWindow

assistant = pyttsx3.init("sapi5")
assistant.setProperty("rate",175)
assistant.setProperty("volume",1.0)
voices = assistant.getProperty('voices')
assistant.setProperty('voice', voices[0].id)

def speak(audio):
    assistant.say(audio)
    assistant.runAndWait()

def welcome():
    location = "Pune"
    search = f'Weather in {location}'
    url = f'https://www.google.com/search?&q={search}'
    html = requests.get(url).content
    soup = BeautifulSoup(html, 'html.parser')
```

```

temp = soup.find('div', attrs={'class': 'BNeawe iBp4i AP7Wnd'}).text
a=time.strftime("%I:%M %p")
playsound("C:\\Users\\gapaw\\Downloads\\WhatsApp-Audio-2022-04-26-at-5.23.01-PM.wav")
playsound("C:\\Users\\gapaw\\Downloads\\WhatsApp-Audio-2022-04-26-at-5.23.05-PM.wav")
hour = int(datetime.datetime.now().hour)
if hour>=0 and hour<=12:
    speak(f"Good Morning sir. It's {a}. The temperature of Pune is {temp}")
    print(f"Good Morning sir. It's {a}. The temperature of Pune is {temp}")
elif hour>12 and hour<18:
    speak(f"Good afternoon sir. It's {a}. The temperature of Pune is {temp}")
    print(f"Good afternoon sir. It's {a}. The temperature of Pune is {temp}")
else:
    speak(f"Good evening sir, it's {a}. The temperature of Pune is {temp}")
    print(f"Good evening sir, it's {a}. The temperature of Pune is {temp}")

speak("Tell me how may I help you")
def super_intelligent_ai(que):
    try:
        my_id="2VX6AR-J87WL3LGE5"
        client = wolframalpha.Client(my_id)
        res = client.query(que)
        ans = next(res.results).text
        return ans
    except:
        return "None"
class MainThread(QThread):
    def __init__(self) -> None:
        super(MainThread,self).__init__()
    def run(self) -> None:
        self.Main()

    def querytaker(self):
        #It takes microphone input from the user and returns string output

        r = speech_recognition.Recognizer()
        with speech_recognition.Microphone(device_index=0) as source:
            print("Listening...")
            speak("Listening...")

            r.pause_threshold = 1
            audio = r.listen(source)

```

```

try:
    print("Recognizing...")
    speak("Recognizing...")
    query = r.recognize_google(audio, language='en-in')
    print(f"User said: {query}\n")
except Exception as e:
    print("Say that again please...")
    return "None"
return query

def Main(self):
    # welcome()
    while True:
        self.query = self.querytaker().lower()

        if "information about" in self.query or "tell me" in self.query:
            speak('Searching Wikipedia...')
            try:
                self.query = self.query.replace("information about", "").replace("tell
me", "")
                results = wikipedia.summary(self.query, sentences=2)
                speak("According to Wikipedia")
                print(results)
                speak(results)
            except:
                print("An Unexpexted Error!")
                speak("An Unexpexted Error!")

        elif 'none' in self.query:
            print(f"You what to search ,{self.query}?")
            speak(f"You what to search ,{self.query}?")
            a=self.querytaker().lower()
            if a=="yes":
                speak("ok")
                webbrowser.open(f"{self.query}.com")

        elif "temperature" in self.query :
            print("Temperature of which city?")
            speak("Temperature of which city?")
            location = self.querytaker().lower()

            search = f'Weather in {location}'

```

```

url = f'https://www.google.com/search?&q={search}'
html = requests.get(url).content

soup = BeautifulSoup(html, 'html.parser')
temp = soup.find('div', attrs={'class': 'BNeawe iBp4i AP7Wnd'}).text
speak(f'Current Temperature of {location} is {temp}')
print(f'Current Temperature of {location} is {temp}')

elif 'open youtube' in self.query or 'youtube' in self.query :
    speak("Launching youtube for you sir!")
    webbrowser.open("youtube.com")

elif 'open dashboard' in self.query or 'dashboard' in self.query :
    speak("Launching dashboard for you sir!")
    webbrowser.open('https://docs.google.com/document/d/14qNvVJIW
GXZCRII5_58ZrPRW8UpbAM3MEQbKq4obUL0/edit')

elif 'open google' in self.query or 'google' in self.query :
    speak("Launching google for you sir!")
    webbrowser.open("google.com")

elif 'open vs code' in self.query or 'code' in self.query :
    speak("Launching visual studio code for you sir!")
    codePath = "C:\\Users\\gapaw\\AppData\\Local\\Programs\\Microsoft
VS Code\\Code.exe"
    os.startfile(codePath)

elif 'open whatsapp' in self.query or "open what's app" in self.query or
'whats app' in self.query or "what's aap" in self.query :
    speak("Launching what's app for you sir!")
    codePath =
"C:\\Users\\gapaw\\AppData\\Local\\WhatsApp\\WhatsApp.exe"
    os.startfile(codePath)

elif 'open telegram' in self.query or 'telegram' in self.query :
    speak("Launching telegram for you sir!")

```

```
codePath = "C:\\Users\\gapaw\\AppData\\Roaming\\Telegram
Desktop\\Telegram.exe"
os.startfile(codePath)
```

```
elif 'open chrome' in self.query or 'chrome' in self.query :
    speak("Launching chrome for you sir!")
    codePath = "C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe"
    os.startfile(codePath)
```

```
elif 'open excel' in self.query or 'excel' in self.query :
    speak("Launching excel for you sir!")
    codePath = "C:\\Program Files\\Microsoft
Office\\root\\Office16\\EXCEL.EXE"
    os.startfile(codePath)
```

```
elif 'open powerpoint' in self.query or 'powerpoint' in self.query :
    speak("Launching powerpoint for you sir!")
    codePath = "C:\\Program Files\\Microsoft
Office\\root\\Office16\\POWERPNT.EXE"
    os.startfile(codePath)
```

```
elif 'open word' in self.query :
    speak("Launching word for you sir!")
    codePath = "C:\\Program Files\\Microsoft
Office\\root\\Office16\\WINWORD.EXE"
    os.startfile(codePath)
```

```
elif "set alarm" in self.query or "alarm" in self.query:
    speak("Enter the hours !")
    hrs=int(input("Enter the hours (00) :"))
    speak("Enter the minutes !")
    min=int(input("Enter the minutes (00) :"))
    speak("Enter the PM or AM")
```

```

pmam=input("Enter the pm or am :")
if pmam=="pm":
    hrs+=12

while True:
    if hrs==datetime.datetime.now().hour and
min==datetime.datetime.now().minute:
        playsound("C:\\Users\\gapaw\\Downloads\\ring.mp3.wav")
    elif min+1==datetime.datetime.now().minute:
        break

    elif "take screenshot" in self.query or "take a screenshot" in self.query or
"capture the screen" in self.query:
        speak("By what name do you want to save the screenshot?")
        name = self.querytaker().lower()
        speak("Alright sir, taking the screenshot")
        img = pyautogui.screenshot()
        name = f"{name}.png"
        img.save(name)
        speak("The screenshot has been succesfully captured")

    elif "show me the screenshot" in self.query or "show screenshot" in
self.query:
        speak("Name of the image is?")
        nam =self.querytaker().lower
        try:
            img =
Image.open(f"C:\\Users\\gapaw\\Desktop\\JARVIS\\{nam}.png")
            img.show(img)
            speak("Here it is sir")
            time.sleep(2)
        except IOError:
            speak("Sorry sir, I am unable to display the screenshot")

    elif "volume up" in self.query or "increase volume" in self.query:
        pyautogui.press("volumeup")
        speak('volume increased')

```



```

elif "volume down" in self.query or "decrease volume" in self.query:
    pyautogui.press("volumedown")
    speak('volume decreased')

elif 'your name' in self.query or "whats your name" in self.query:
    speak('My name is JARVIS')

elif 'who made you' in self.query:
    speak('I was created by my AI master ganesh in 2022')

elif 'jarvis stands for' in self.query or "full form of jarvis" in self.query:
    speak('J.A.R.V.I.S stands for JUST A RATHER VERY
INTELLIGENT SYSTEM')

elif "close the window" in self.query or "close window" in self.query:
    speak("Okay sir, closing the window")
    pyautogui.keyDown("alt")
    pyautogui.press("f4")
    time.sleep(1)
    pyautogui.keyUp("alt")

elif "wait" in self.query:
    speak("Waiting")
    time.sleep(30)
    print("tell me how may I help you")
    speak("tell me how may I help you")

elif "goodbye" in self.query or "bye" in self.query or "stop" in self.query
or "offline" in self.query or "exit" in self.query:
    speak("Alright sir, going offline. It was nice working with you, have a
nice day!")
    print("Alright sir, going offline. It was nice working with you, have a
nice day!")
    sys.exit()

```

```

else:
    que=self.query
    a= super_intelligent_ai(que)
    if "noun" in a or "Noun" in a:
        print(f"You what to search ,{self.query}?")
        speak(f"You what to search ,{self.query}?")
        a=self.querytaker().lower()
        if a=="yes":
            speak("ok")
            webbrowser.open(f"{self.query}.com")
    elif a!="None":
        print(a)
        speak(a)
    else:
        print(f"You what to search ,{self.query}?")
        speak(f"You what to search ,{self.query}?")
        a=self.querytaker().lower()
        if a=="yes":
            speak("ok")
            webbrowser.open(f"{self.query}.com")

startExecution=MainThread()
class Main(QMainWindow):
    def __init__(self) -> None:
        super().__init__()
        self.ui=Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.pushButton.clicked.connect(self.startTask)
        self.ui.pushButton_2.clicked.connect(self.close)
    def startTask(self):
        self.ui.movie=QtGui.QMovie("2264c-jarvis-1.gif")
        self.ui.label.setMovie(self.ui.movie)
        self.ui.movie.start()
        startExecution.start()

app= QApplication(sys.argv)
jarvis=Main()
jarvis.show()
exit(app.exec_())

```

4.2CODE FOR GUI

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'jarvisui.ui'
#
# Created by: PyQt5 UI code generator 5.15.6
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again. Do not edit this file unless you know what you are doing.

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1269, 830)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(-100, -10, 1451, 851))
        self.label.setText("")
        self.label.setPixmap(QtGui.QPixmap("2264c-jarvis-1.gif"))
        self.label.setScaledContents(True)
        self.label.setObjectName("label")
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(10, 670, 301, 111))
        self.pushButton.setText("")
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("Start.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.pushButton.setIcon(icon)
        self.pushButton.setIconSize(QtCore.QSize(300, 400))
        self.pushButton.setObjectName("pushButton")
        self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(970, 680, 281, 91))
        self.pushButton_2.setText("")
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap("Quit.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.pushButton_2.setIcon(icon1)
```

```

self.pushButton_2.setIconSize(QtCore.QSize(300, 400))
self.pushButton_2.setObjectName("pushButton_2")
MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)
QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))

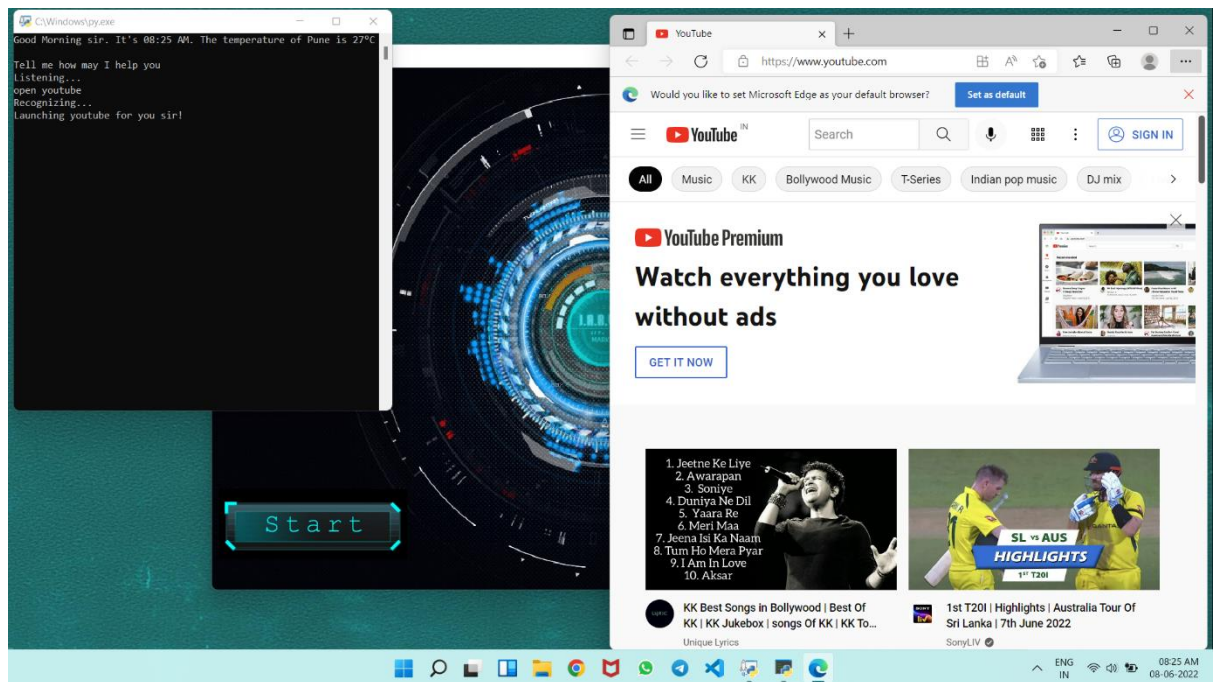
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

```

CHAPTER 2: OUTPUT ANALYSIS

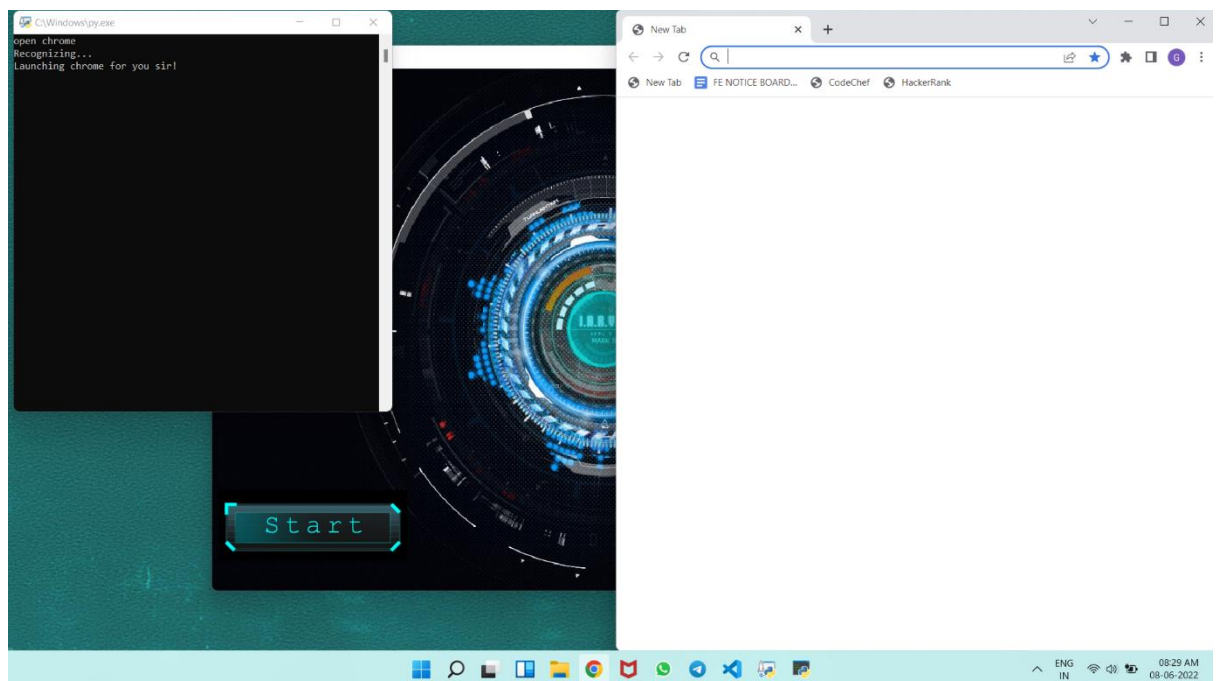
INPUT: Open youtube

OUTPUT:



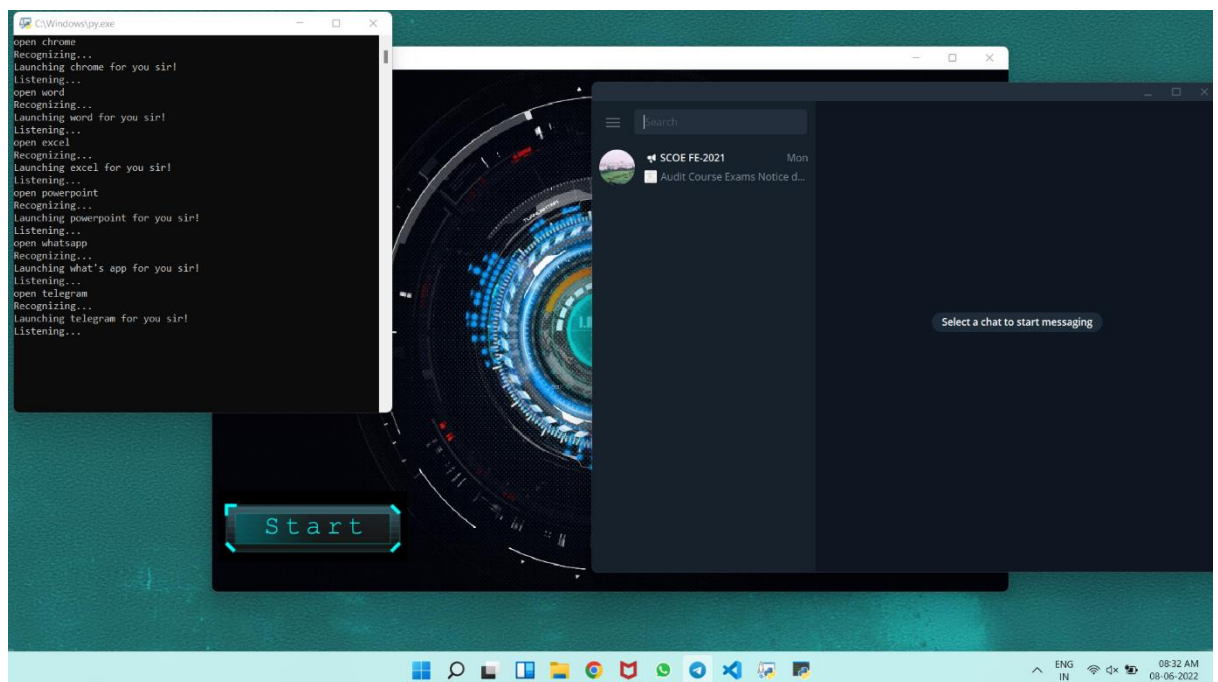
INPUT: Open chrome

OUTPUT:



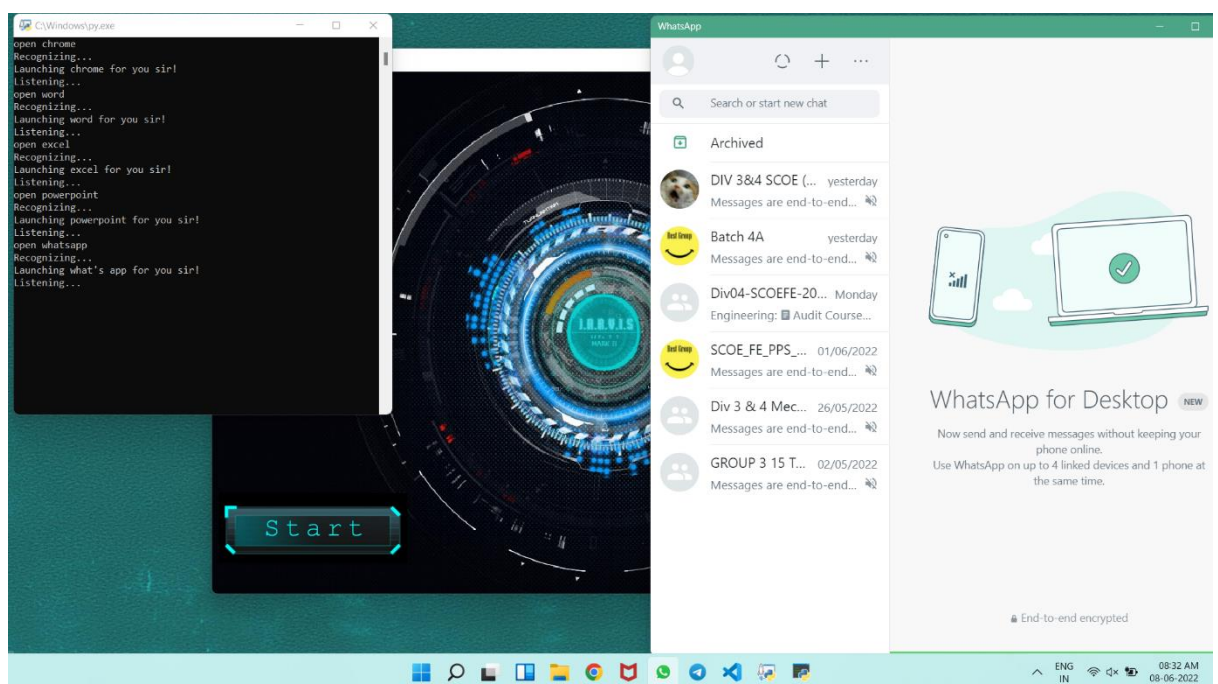
INPUT: Open Telegram

OUTPUT:



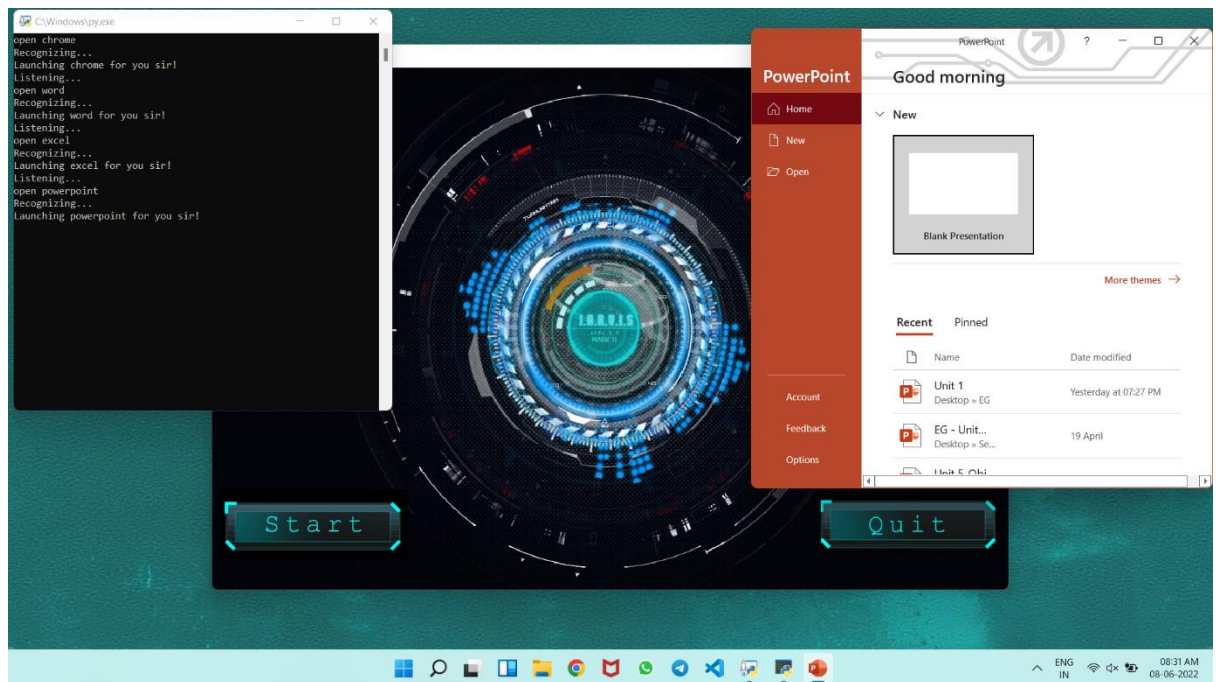
INPUT: Open Whatsapp

OUTPUT:



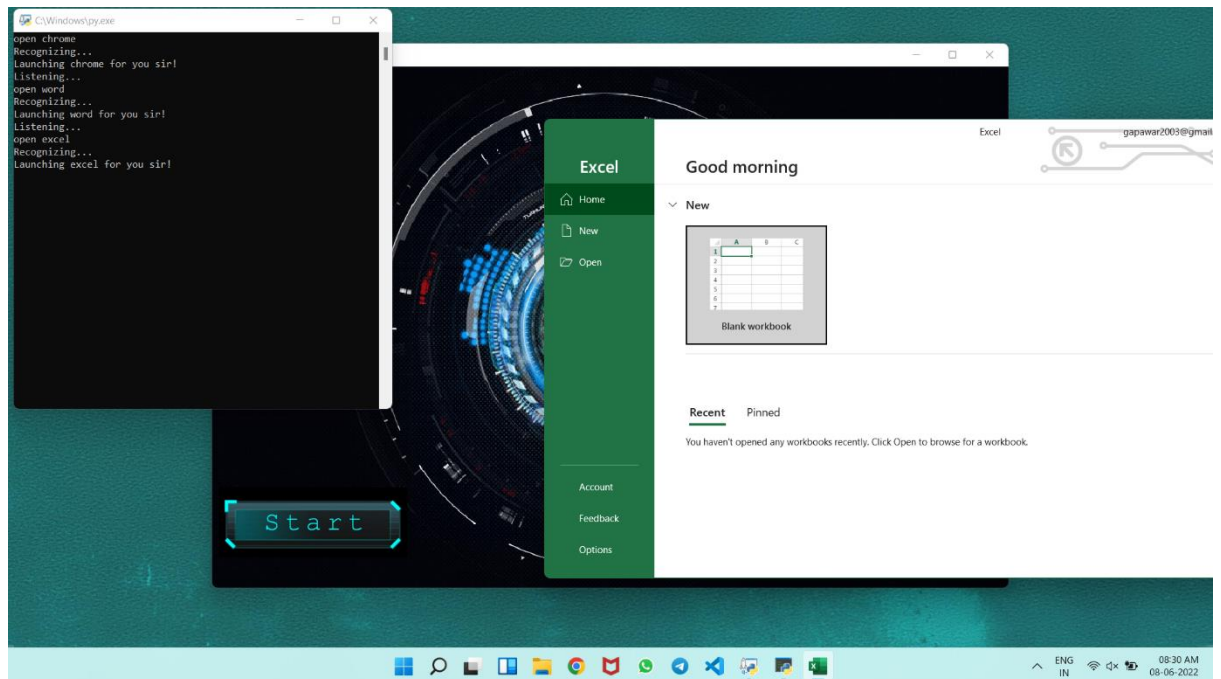
INPUT: Open PowerPoint

OUTPUT:



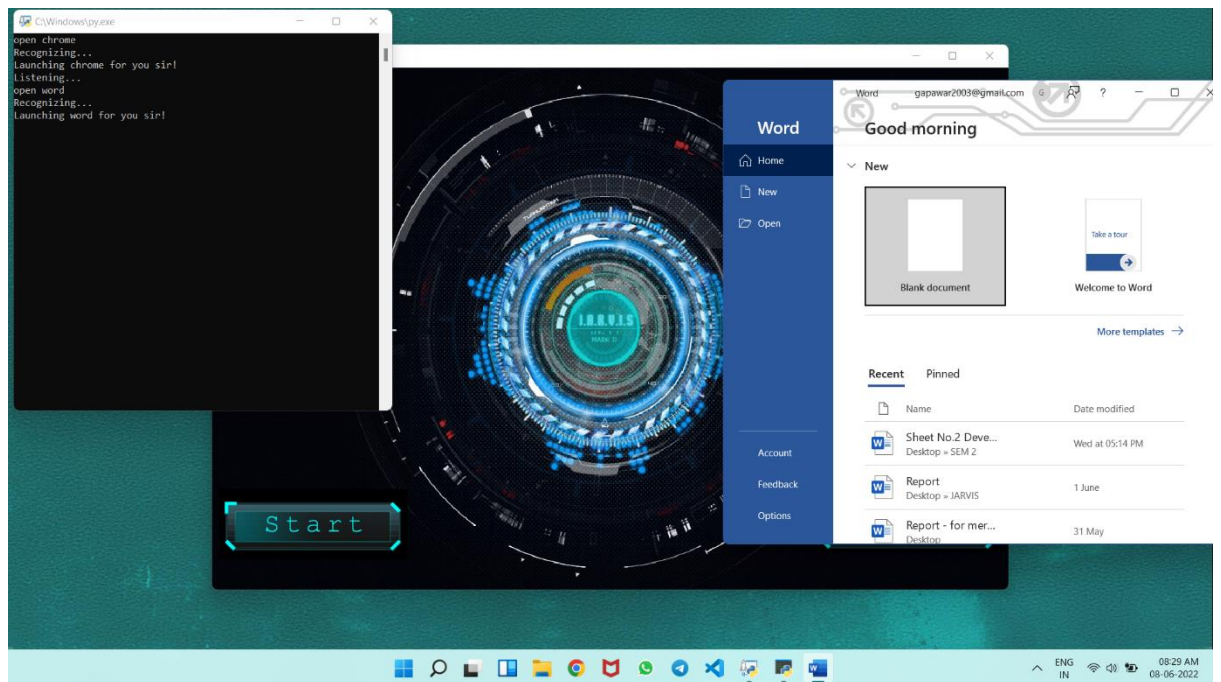
INPUT: Open Excel

OUTPUT:



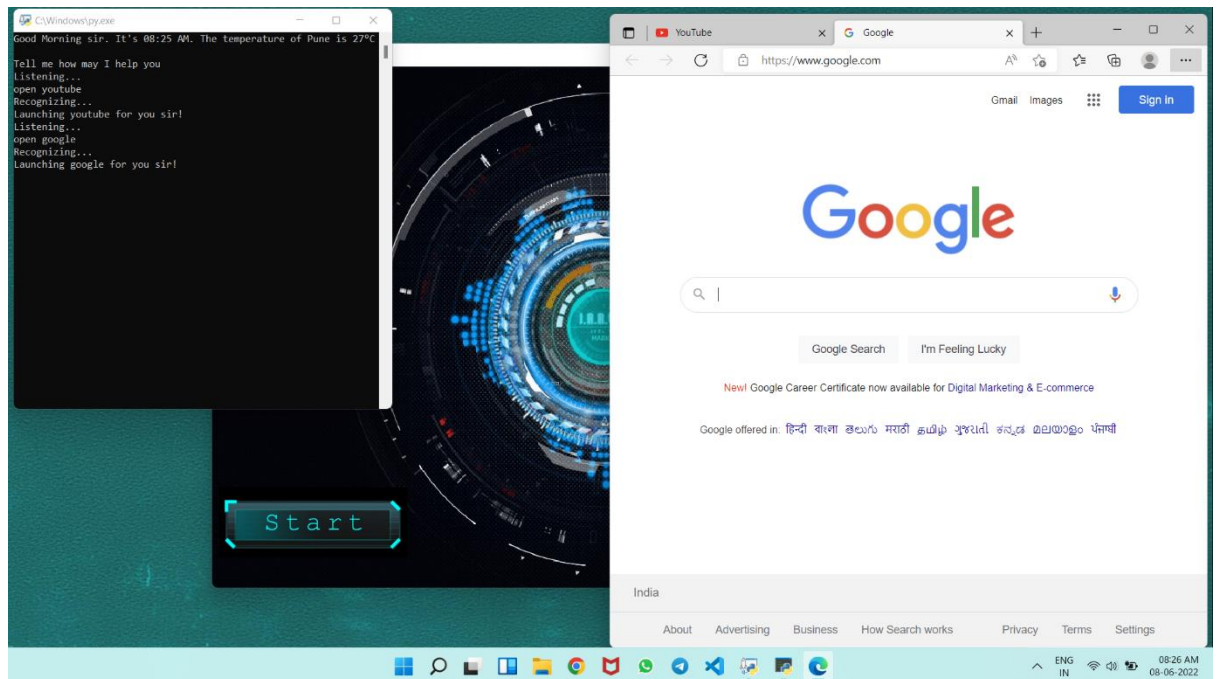
INPUT: Open Word

OUTPUT:



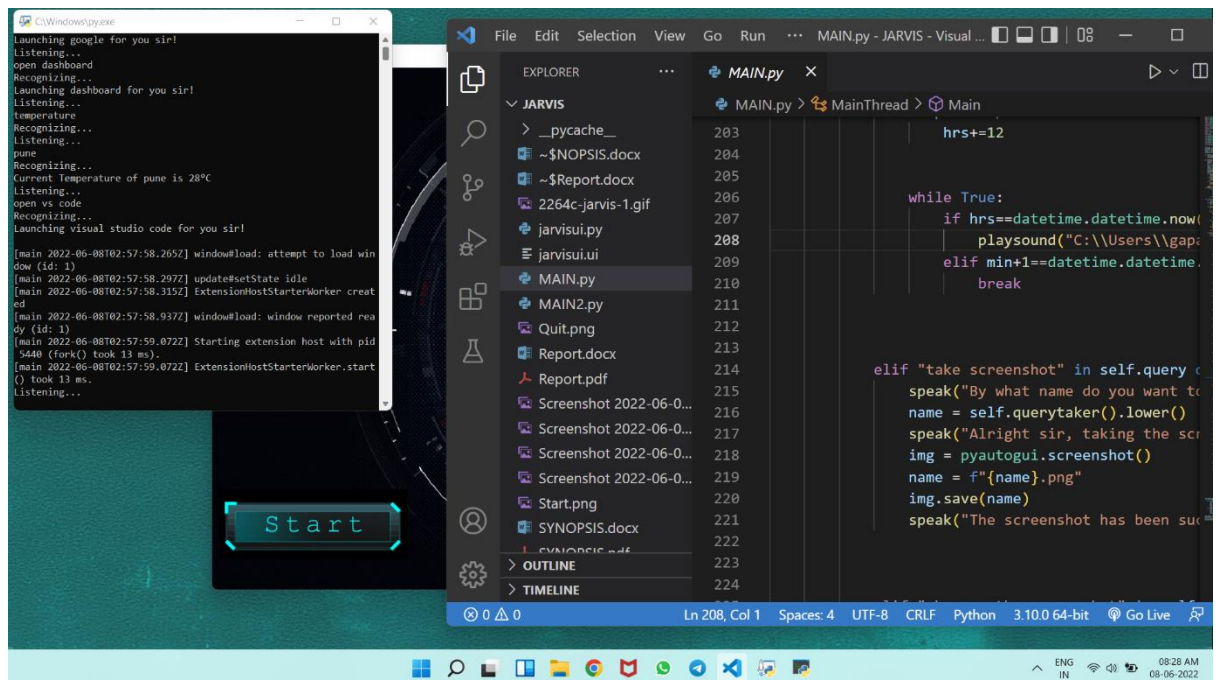
INPUT: Open Google

OUTPUT:



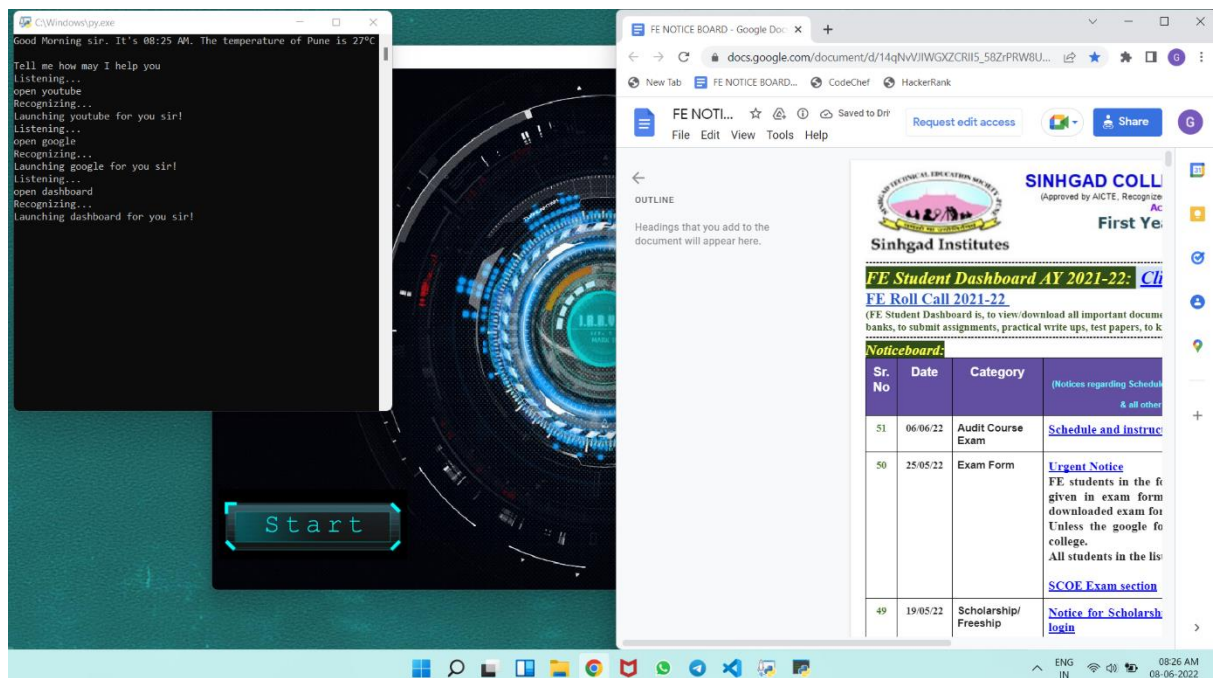
INPUT: Open Vs Code

OUTPUT:



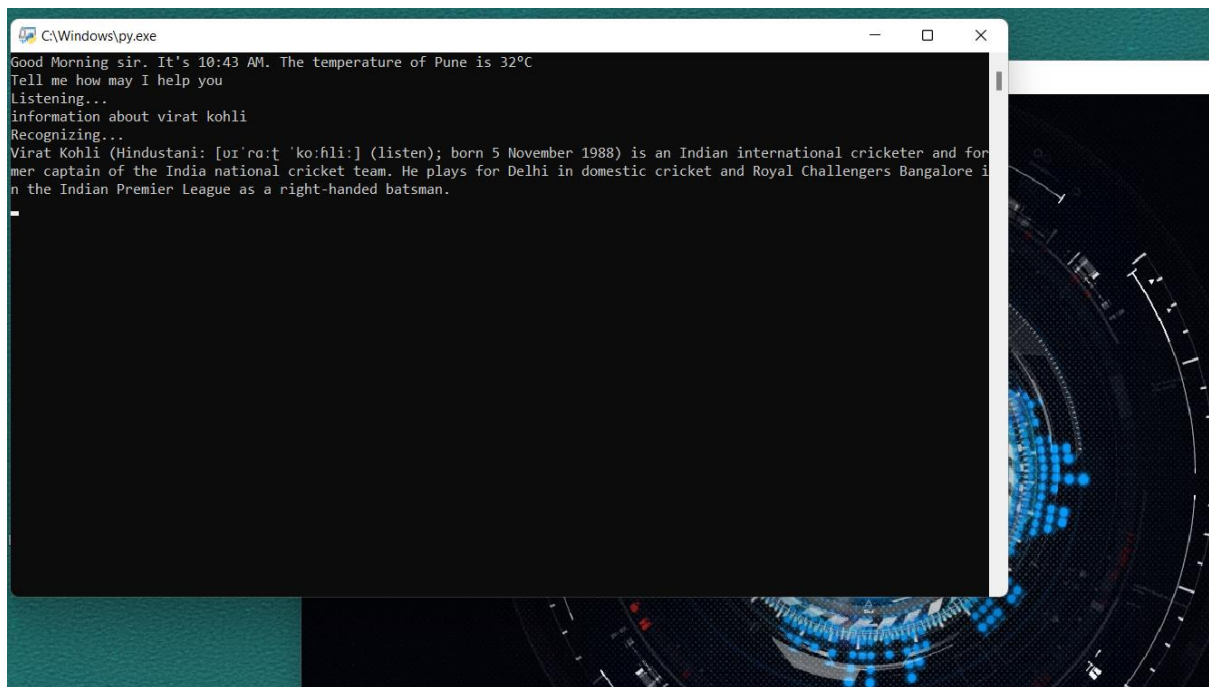
INPUT: Open Dashboard

OUTPUT:



INPUT: Information about virat kohli

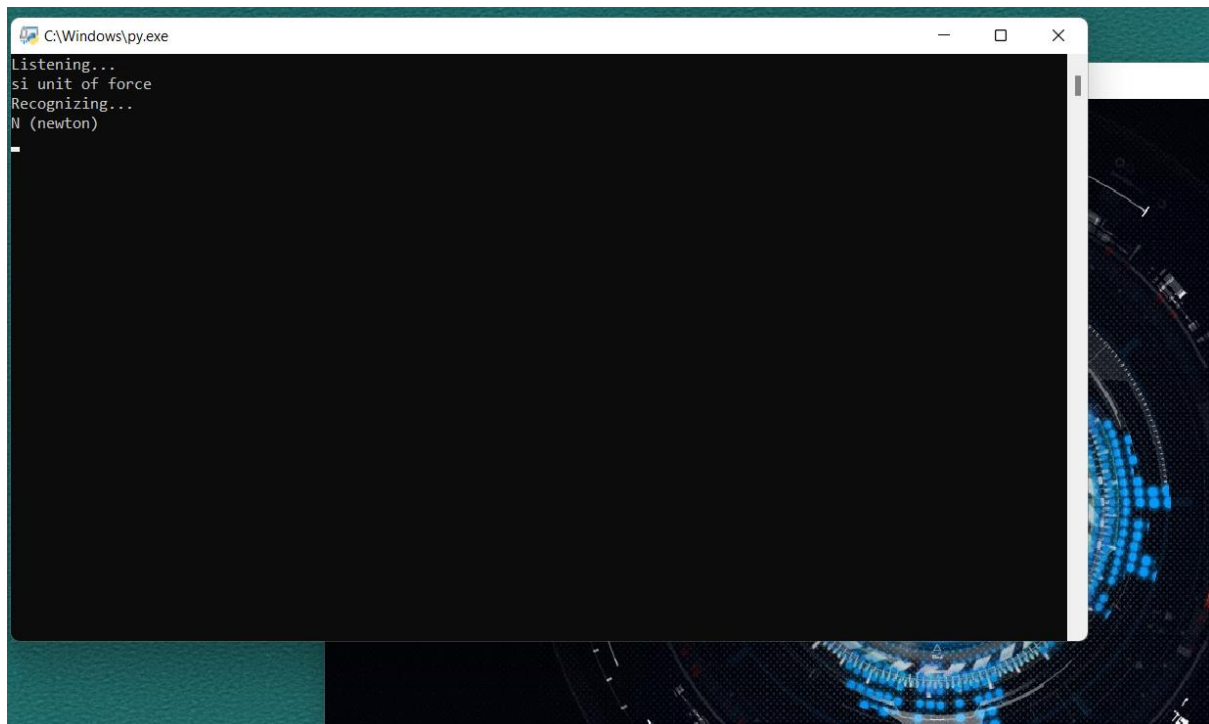
OUTPUT:



```
C:\Windows\py.exe
Good Morning sir. It's 10:43 AM. The temperature of Pune is 32°C
Tell me how may I help you
Listening...
information about virat kohli
Recognizing...
Virat Kohli (Hindustani: [ur'ro:t 'ko:hli:] (listen); born 5 November 1988) is an Indian international cricketer and former captain of the India national cricket team. He plays for Delhi in domestic cricket and Royal Challengers Bangalore in the Indian Premier League as a right-handed batsman.
```

INPUT: SI unit of force

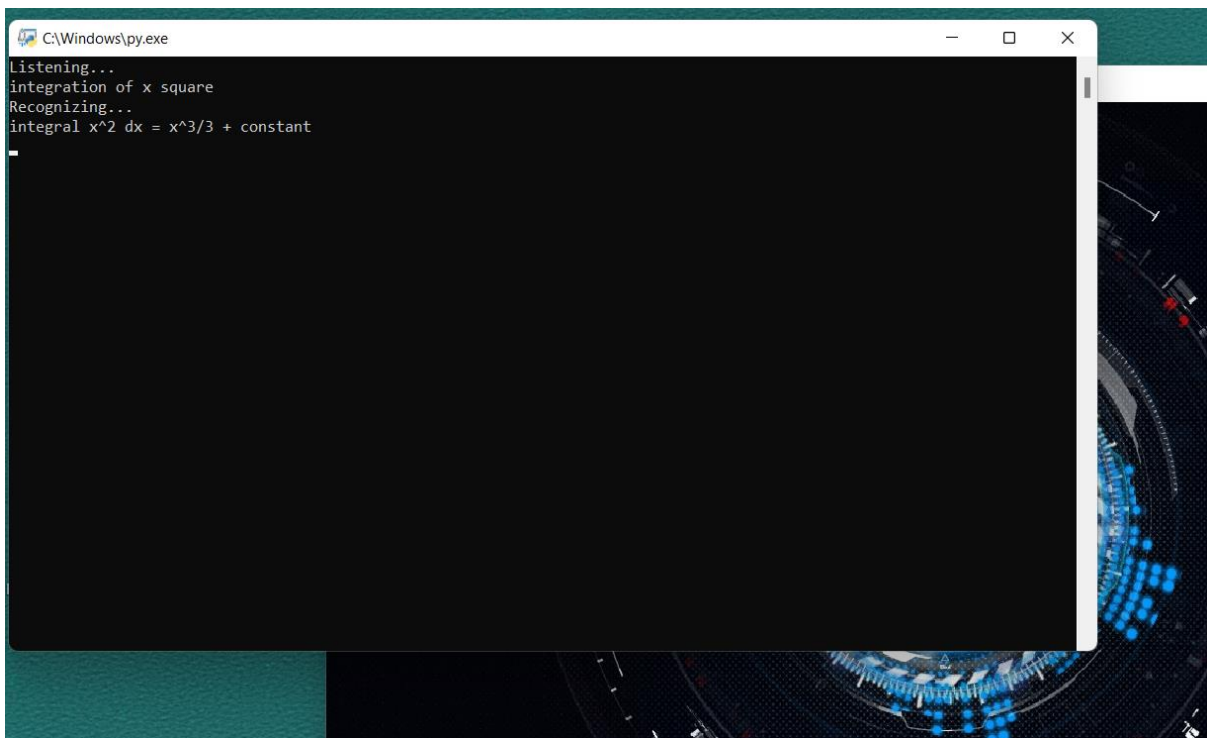
OUTPUT:



```
C:\Windows\py.exe
Listening...
si unit of force
Recognizing...
N (newton)
```


INPUT: Integration of x square

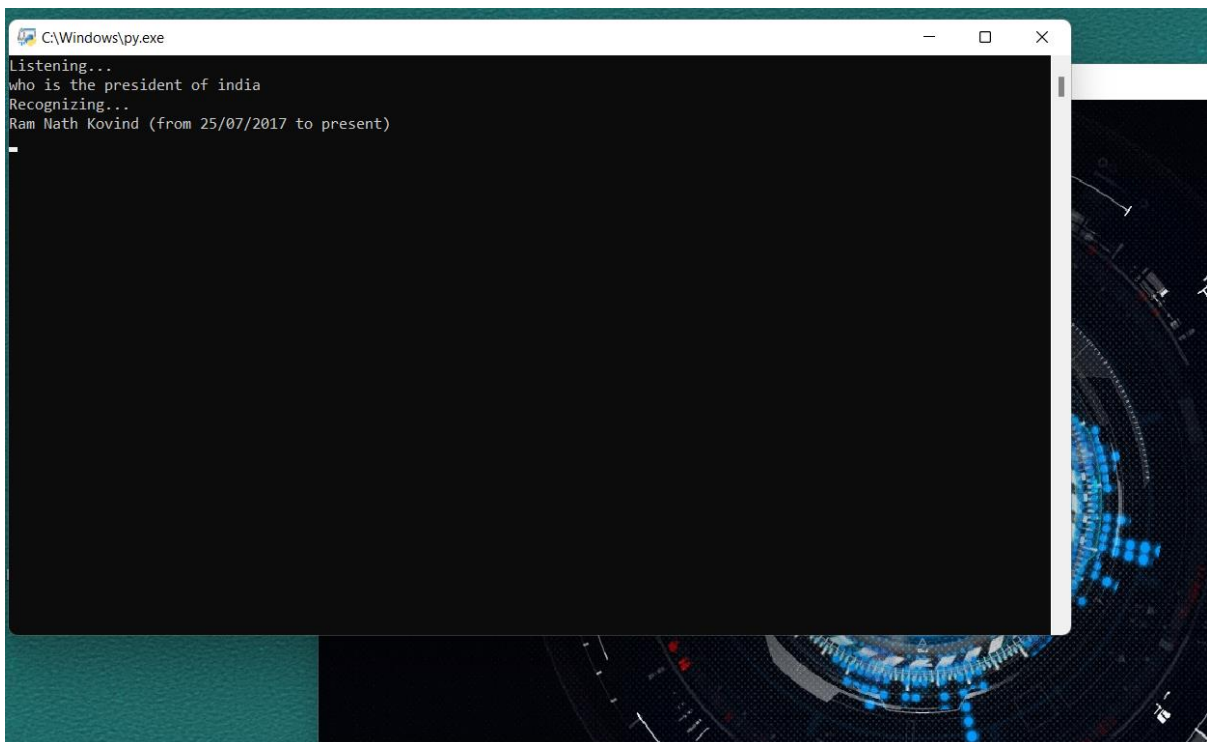
OUTPUT:

A screenshot of a Windows command prompt window titled 'C:\Windows\py.exe'. The window has a black background with white text. The text shows the process of listening for a command, recognizing the input 'integration of x square', and then outputting the mathematical result: 'integral x^2 dx = x^3/3 + constant'.

```
C:\Windows\py.exe
Listening...
integration of x square
Recognizing...
integral x^2 dx = x^3/3 + constant
_
```

INPUT: Who is the president of india

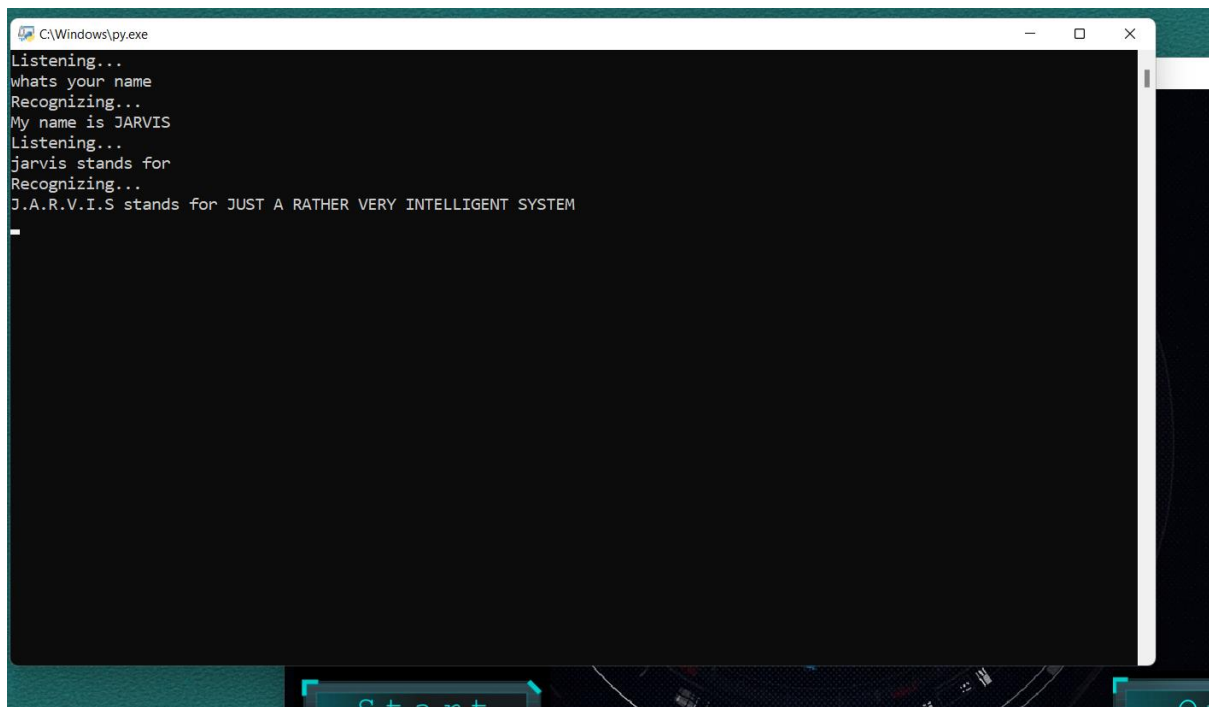
OUTPUT:

A screenshot of a Windows command prompt window titled 'C:\Windows\py.exe'. The window has a black background with white text. The text shows the process of listening for a command, recognizing the input 'who is the president of india', and then outputting the name 'Ram Nath Kovind' along with a time period: '(from 25/07/2017 to present)'.

```
C:\Windows\py.exe
Listening...
who is the president of india
Recognizing...
Ram Nath Kovind (from 25/07/2017 to present)
_
```

INPUT: Jarvis stands for

OUTPUT:

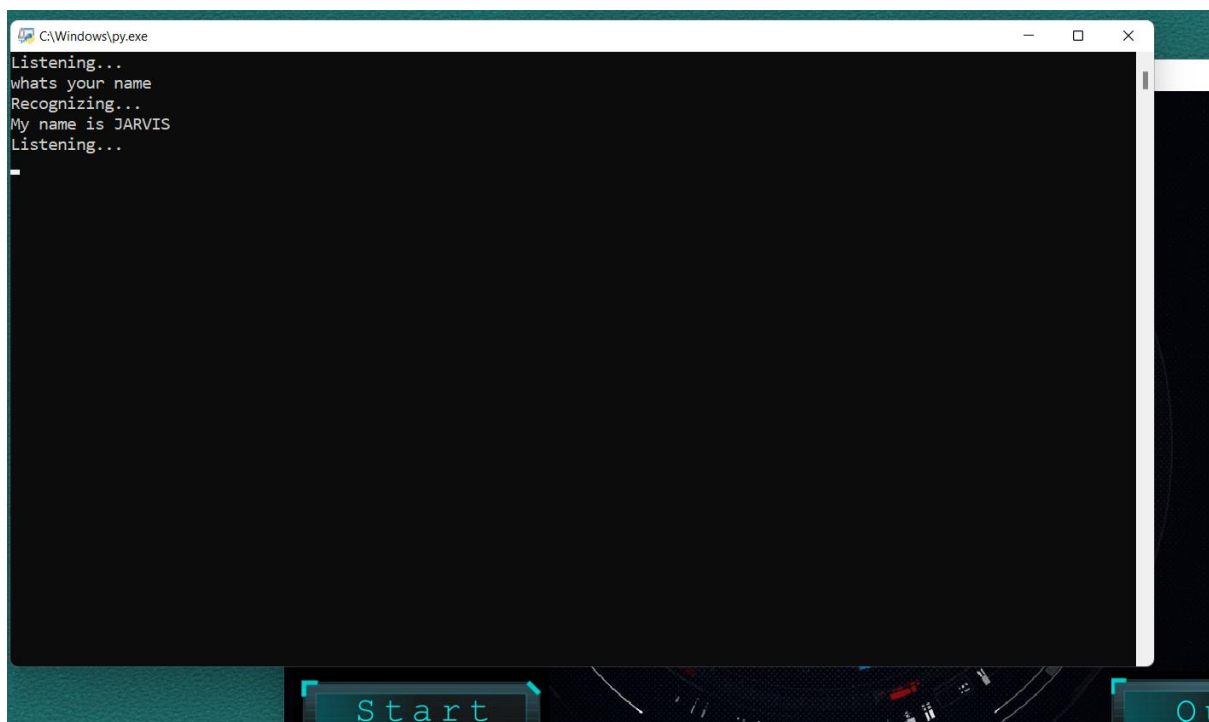


```
C:\Windows\py.exe
Listening...
whats your name
Recognizing...
My name is JARVIS
Listening...
jarvis stands for
Recognizing...
J.A.R.V.I.S stands for JUST A RATHER VERY INTELLIGENT SYSTEM
```

A screenshot of a Windows command prompt window titled "C:\Windows\py.exe". The window has a black background with white text. The text shows a sequence of commands and responses: "Listening...", "whats your name", "Recognizing...", "My name is JARVIS", "Listening...", "jarvis stands for", "Recognizing...", and "J.A.R.V.I.S stands for JUST A RATHER VERY INTELLIGENT SYSTEM". The window is partially overlapping a game interface with a "Start" button visible at the bottom.

INPUT: what's your name

OUTPUT:



```
C:\Windows\py.exe
Listening...
whats your name
Recognizing...
My name is JARVIS
Listening...
```

A screenshot of a Windows command prompt window titled "C:\Windows\py.exe". The window has a black background with white text. The text shows a sequence of commands and responses: "Listening...", "whats your name", "Recognizing...", "My name is JARVIS", and "Listening...". The window is partially overlapping a game interface with a "Start" button visible at the bottom.

LIMITATIONS

- Background voice can interfere
- Misinterpretation because of accents and may cause inaccurate res
- JARVIS cannot be called externally anytime like other traditional assistants like Google Assistant can be called just by saying. "Ok Google"

FUTURE SCOPE & ENHANCEMENT OF THE PROJECT

The virtual assistants which are currently available are fast and responsive but we still have to go a long way. The understanding and reliability of the current Jarvis is need to be improved a lot. The Jarvis available nowadays are still not reliable in critical scenarios. The future of these assistant will have the virtual assistants incorporated with Artificial Intelligence which includes Machine Learning, Neural Networks, etc. and lots of things.

REFERENCES

WEBSITES:

www.stackoverflow.com

www.python.org

www.geeksforgeeks.com

www.github.com

www.walframalpha.com

www.google.co.in

www.pythonprogramming.net

YOUTUBE CHANNELS:

Codewithharry

Avi Upadhyay