Introduction

In Today's World almost all Works are digitalized. And having a Smartphone is like having the world at your fingertips. Technology keeps developing day by day where now we can do various tasks with the use of technology. There exist systems where we can say "Call Mom", and it does the task without even operating by touching. That is the Ultimate Goal of a Virtual Assistant. It also supports specialized task such as booking a flight, or finding cheapest book online from various e-commerce sites and then providing an interface to book an order are helping automate search, discovery and online order operations.

Personal Assistants are software programs that help you ease your day-to-day tasks, such as showing weather report, creating reminders, and creating shopping lists etc. They can take commands via text and reply through voice. Voice searches have dominated over text search. The Personal Assistant with voice over, made by our team is named as "***Leon***".

Common Features of a Personal Assistant:

The concept of all voice recognition systems is similar, but each system has its own unique features.

Google Assistant does particularly well in understanding the context of what has been said. The system benefits from the gigantic database of search entries.

On other assistants you have to send commands more precisely so that the desired task is completed correctly.

Google is most likely to know what you want if you try it in different ways. Moreover, it will soon be possible to communicate with Google assistant in several languages. Recently, the system has also included so-called routines. These are voice commands that can involve a chain of actions. The sequence of commands is programmed by the user.

However, in the past year, we've been witnessing the evolution of the humble chatbot from merely being a customer support tool to being touted as the next big thing in business.

The reason for this is quite simple: chatbots, and the artificial intelligence that powers them, are getting smarter and smarter. Huge innovations in the industry have given businesses the ability to use chatbots in more exciting and strategic ways, chief among them as a tool for sales and marketing.

Python Libraries used for Building Leon

Python Libraries are a set of useful functions that eliminate the need for writing codes from scratch. There are over 137,000 **python libraries** present today. **Python libraries** play a vital role in developing machine learning, data science, data visualization, image and data manipulation **applications** and more.

Essential Ingredients In our Project:

Libraries used in this project:

1. **tkinter** *(in-built module)*
2. **webbrowser** *(in-built module)*
3. **youtube_search** *(external module)*
4. **pyttsx3** *(external module)*
5. **pygames** *(external module)*
6. **Pandas** *(external module)*
7. **datetime** *(in-built module)*
8. **subprocess** *(in-built module)*

These are the clever tools to simplify our project beautiful and Divine….

1. tkinter (GUI):

This framework provides Python users with a simple way to create GUI(Graphical User Interface) elements **using** the widgets found in the Tk toolkit. Tk widgets can be **used** to construct buttons, menus, data fields, etc. in a Python application.

2. webbrowser:

The **web browser 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.

3. Youtube_search:

Python function for searching for youtube videos to avoid using their heavily rate-limited API

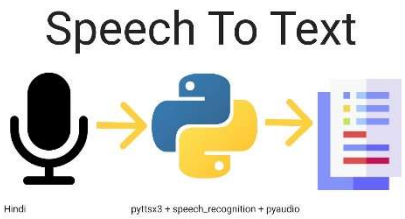
To avoid using the API, this uses the form on the youtube homepage and scrapes the resulting page.

4. pygames:

Pygame is a cross-platform set of **Python modules** which is **used** to create video games.It consists of computer graphics and sound **libraries** designed to be **used** with the **Python** programming language. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

5. pyttsx3:

pyttsx3 is a text-to-speech conversion **library** in **Python**. Unlike alternative **libraries**, it works offline and is compatible with both **Python** 2 and 3. An application invokes the **pyttsx3**. init() factory function to get a reference to a **pyttsx3**.



6. Pandas:

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like ActiveState's ActivePython.

- Data fill
- Data normalization
- Merges and joins
- Data visualization
- Statistical analysis
- Data inspection

7. datetime:

DateTime objects represent instants in time and provide interfaces for controlling its representation without affecting the absolute value of the object.

DateTime objects may be created from a wide variety of string or numeric data, or may be computed from other DateTime objects. DateTimes support the ability to convert their representations to many major timezones, as well as the ablility to create a DateTime object in the context of a given timezone.

8. subprocess:

The subprocess module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.

It offers a higher-level interface than some of the other available modules, and is intended to replace functions such as **os.system(), os.spawn*(), os.popen*(), popen2.*() and commands.*().** To make it easier to compare subprocess with those other modules, many of the examples here re-create the ones used **for os and popen**.

What can Leon Do?

- ✓ *Tell date and time.*
- ✓ *Play local music randomly and can be stopped using a command.*
- ✓ *Play popular YouTube videos mentioning the keywords of the video followed by the command.*
- ✓ *Pop-ups websites with *.com extension when mentioned along with the command.*
- ✓ *Shows the current weather report of the location which is detected based on IP Address.*
- ✓ *Open File Explorer with a single command.*
- ✓ *Open calculator with a single command, for performing mathematical calculations.*

Commands of Leon

These commands can be used to assign a particular task to Neon. The list of commands and their actions are mentioned below:

i. **open:** The keyword open is assigned to open a particular website holding a *.com extension. This action takes place only when the website name is mentioned followed by ***"open"***.

Syntax:

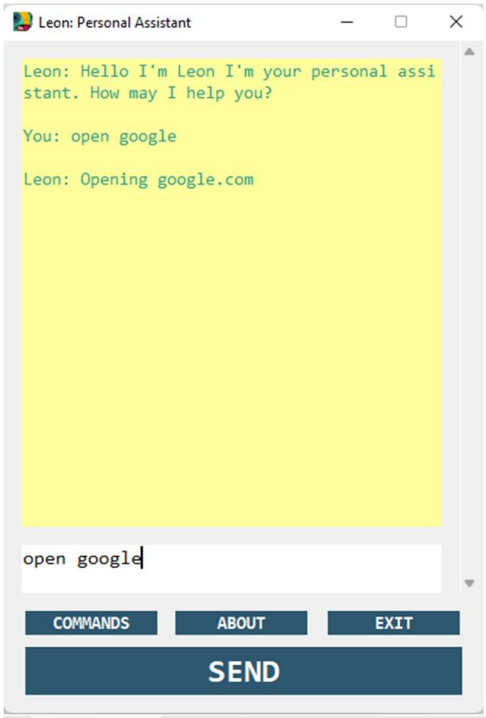
open name_of_website

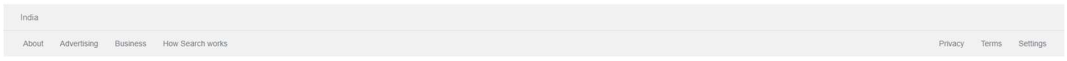
Example:

open google

When ***open google*** is entered, ***google.com*** is opened in the default web browser.

OUTPUT:





ii. **yt:** The keyword yt is assigned to open popular video of a particular search result from YouTube. This action takes place only when the keywords are mentioned followed by ***"yt"***.

Syntax:

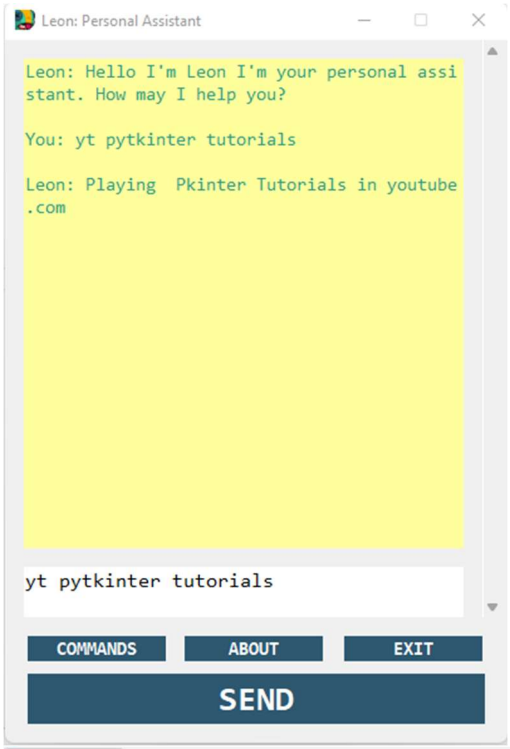
yt keywords_of_video

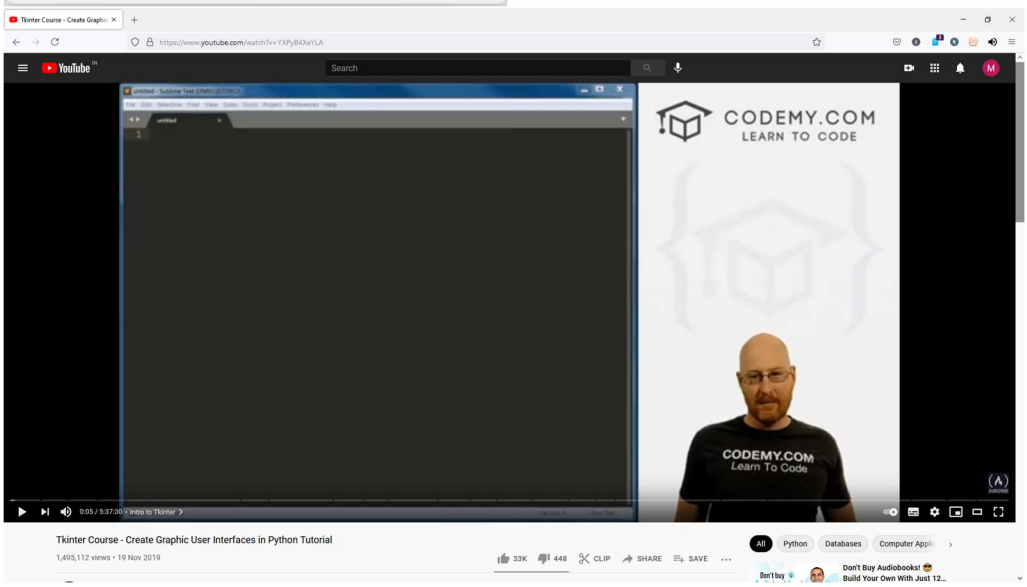
Example:

yt pytkinter tutorials

When ***yt pytkinter tutorials*** is entered the ***pytkinter tutorials*** is searched in YouTube and the 1st result of the search performed is played in the default web browser. The searching process happens in background. *Note: Search results may vary based on the historical activities of YouTube account.*

OUTPUT:





iii. **date:** The keyword date is assigned to say and display the current date. This action takes place even when the command has words mentioned followed by ***"open"***.

Syntax:

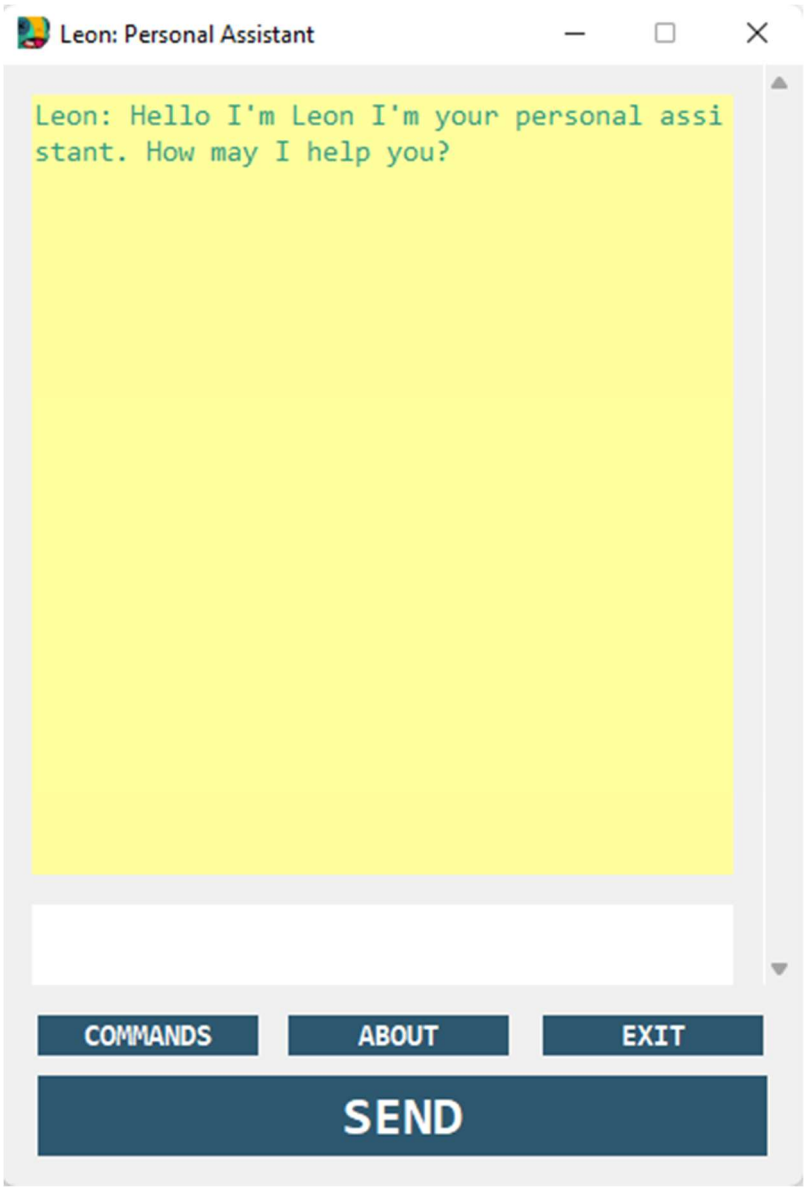
date

Example:

date today, date now, date etc..

When ***date today*** is entered, then the current date is displayed.

OUTPUT:



iv. **time:** The keyword time is assigned to say and display the current time. This action takes place even when the command has words mentioned followed by ***"time"***.

Syntax:

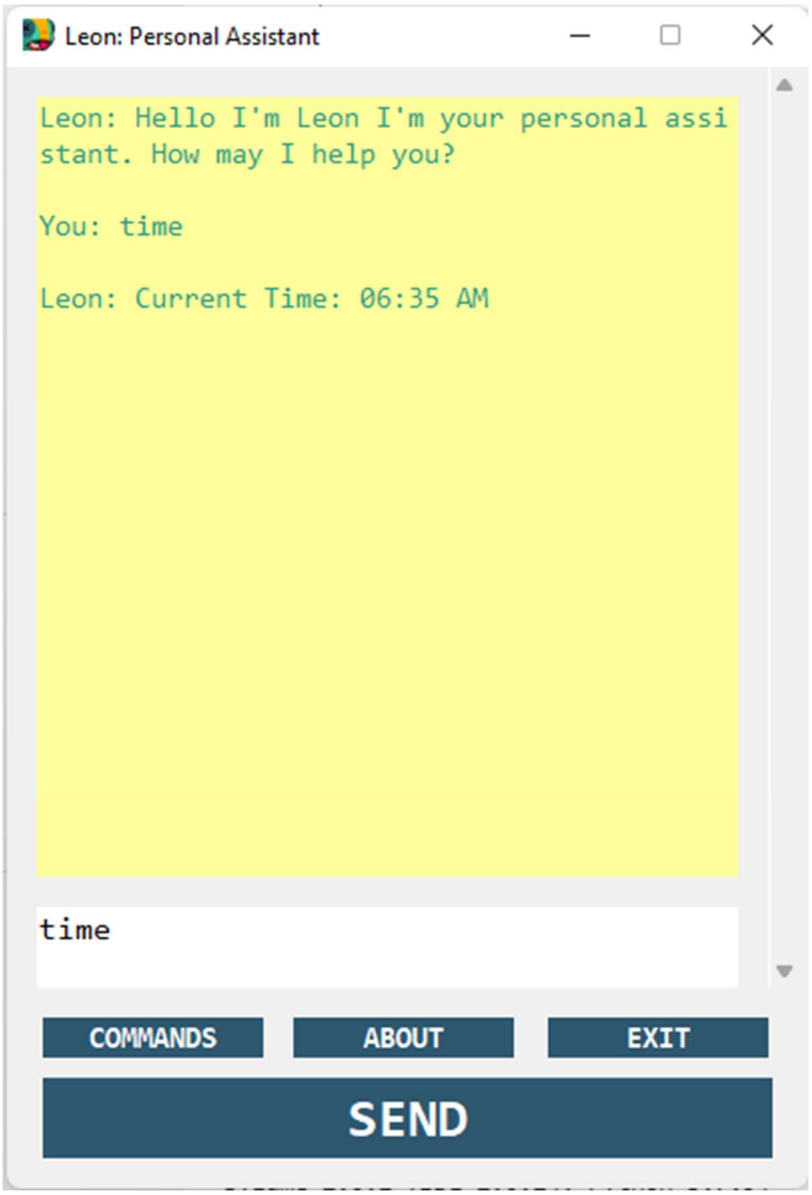
time

Example:

time today, time now, time etc..

When ***time today*** is entered, then the current time is displayed.

OUTPUT:



v. **weather:** The keyword weather is assigned to open the weather report in google.com. This action takes place even when the command has words mentioned followed by ***"weather"***. The location the may/may not be accurate because weather is shown based on IP Address of the PC.

Syntax:

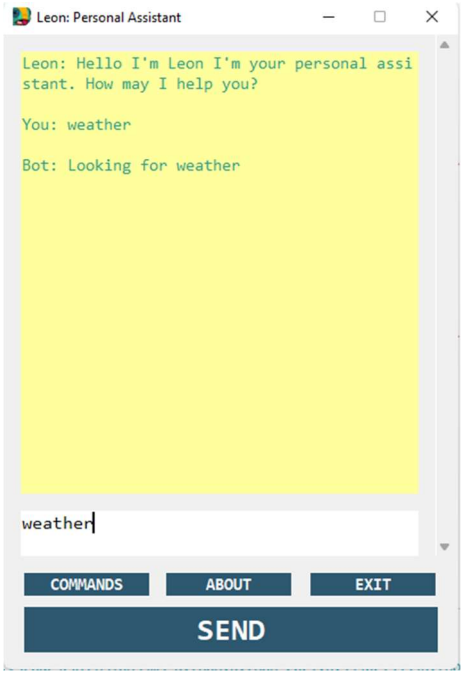
weather

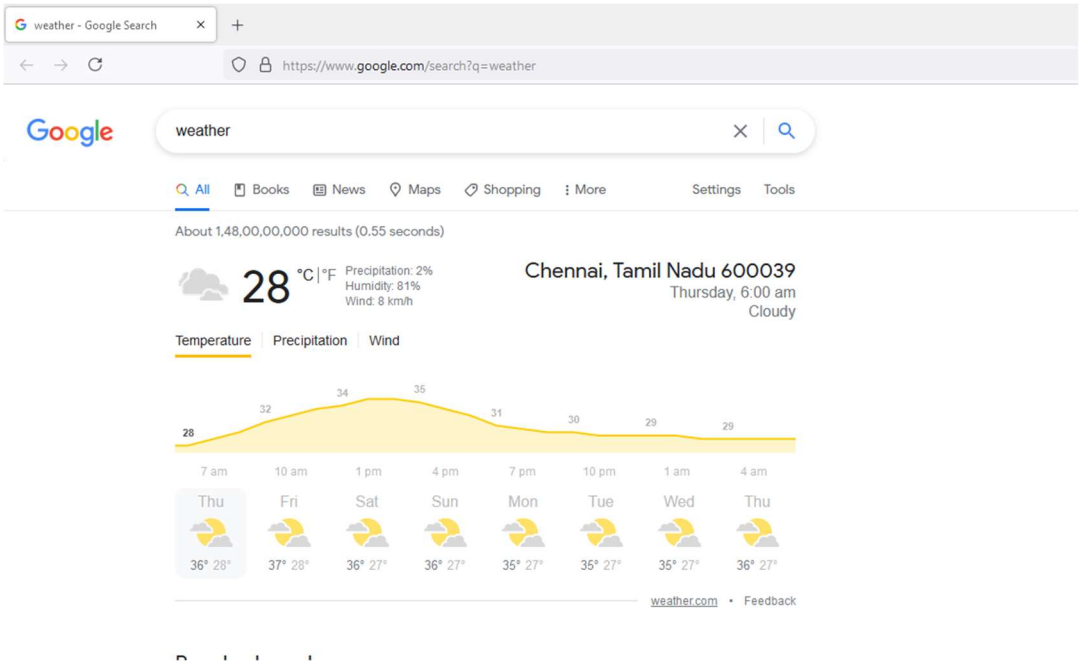
Example:

weather today, weather now, weather etc..

When ***weather today*** is entered, then the current weather is displayed.

OUTPUT:





vi. **music:** The keyword music is assigned to play local music of a particular folder. This action takes place even when the command has words mentioned followed by ***"music"***.

Syntax:

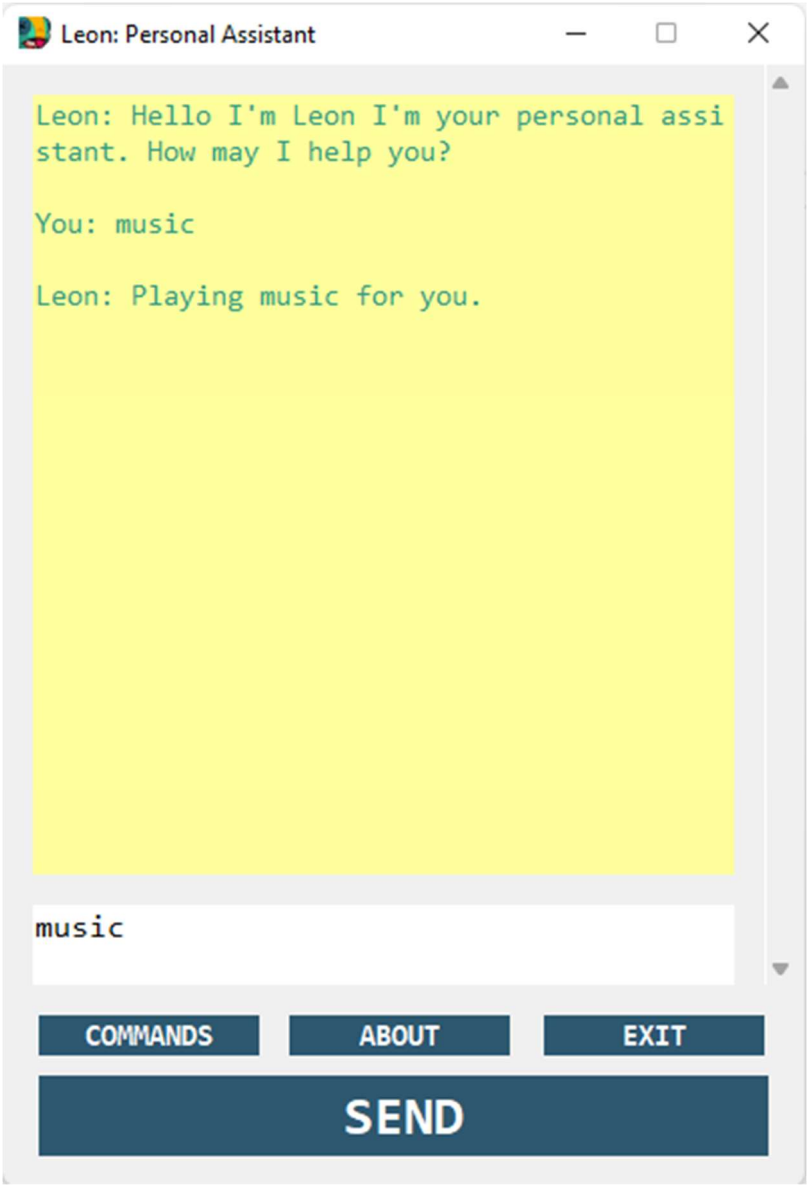
music

Example:

music play, play music, music etc..

When ***music play*** is entered, then the local music files is played randomly.

OUTPUT:



vii. **mstop:** The keyword mstop is assigned to stop the music played by the music command. This action takes place even when the command has words mentioned followed by ***"mstop"***.

Syntax:

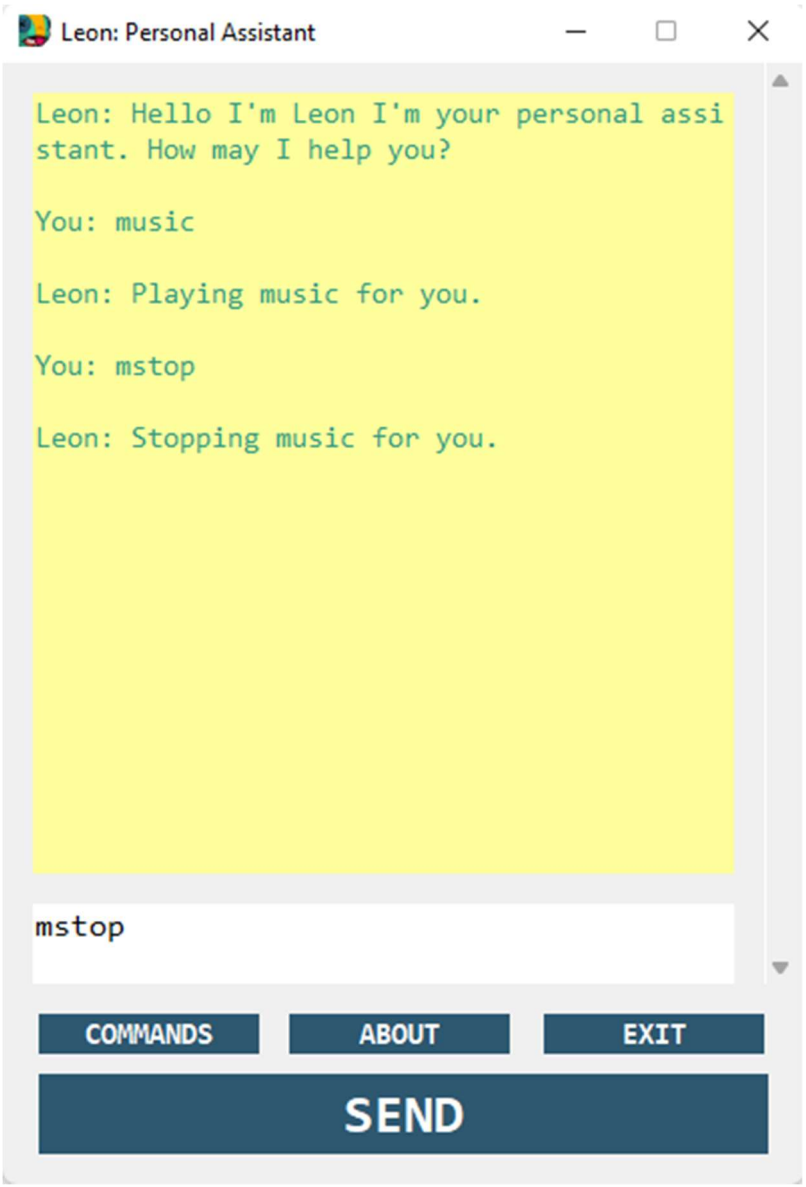
mstop

Example:

mstop stop, stop mstop, mstop etc..

When ***mstop stop*** is entered, then the music playing is stopped.

OUTPUT:



viii. **files:** The keyword files is assigned to open the File Explorer. This action takes place even when the command has words mentioned followed by ***"files"***.

Syntax:

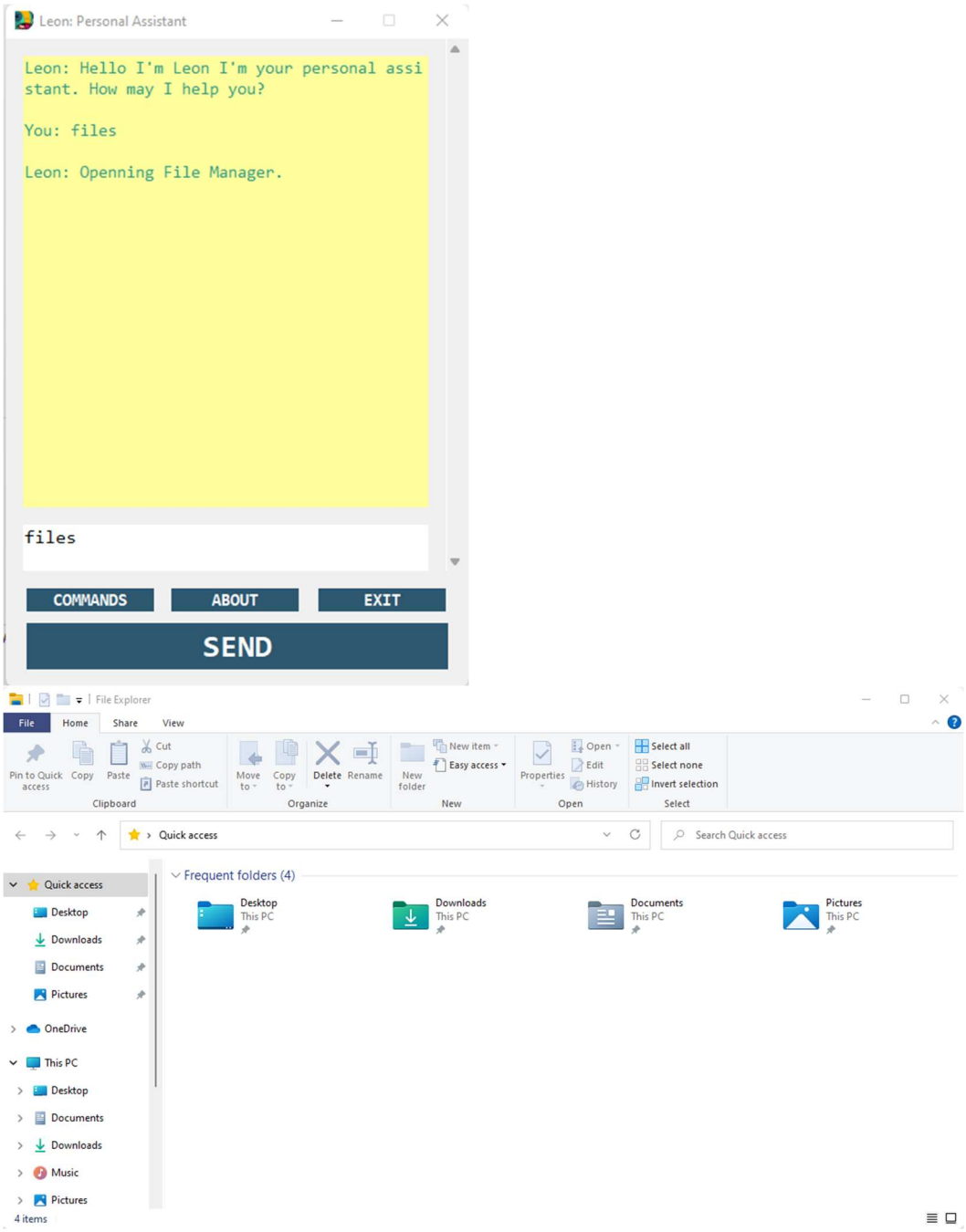
files

Example:

files manager, show files, files etc..

When ***files manager*** is entered, then the File Explorer app is opened.

OUTPUT:



ix. **calc:** The keyword calc is assigned to open the File Explorer. This action takes place even when the command has words mentioned followed by ***"calc"***.

Syntax:

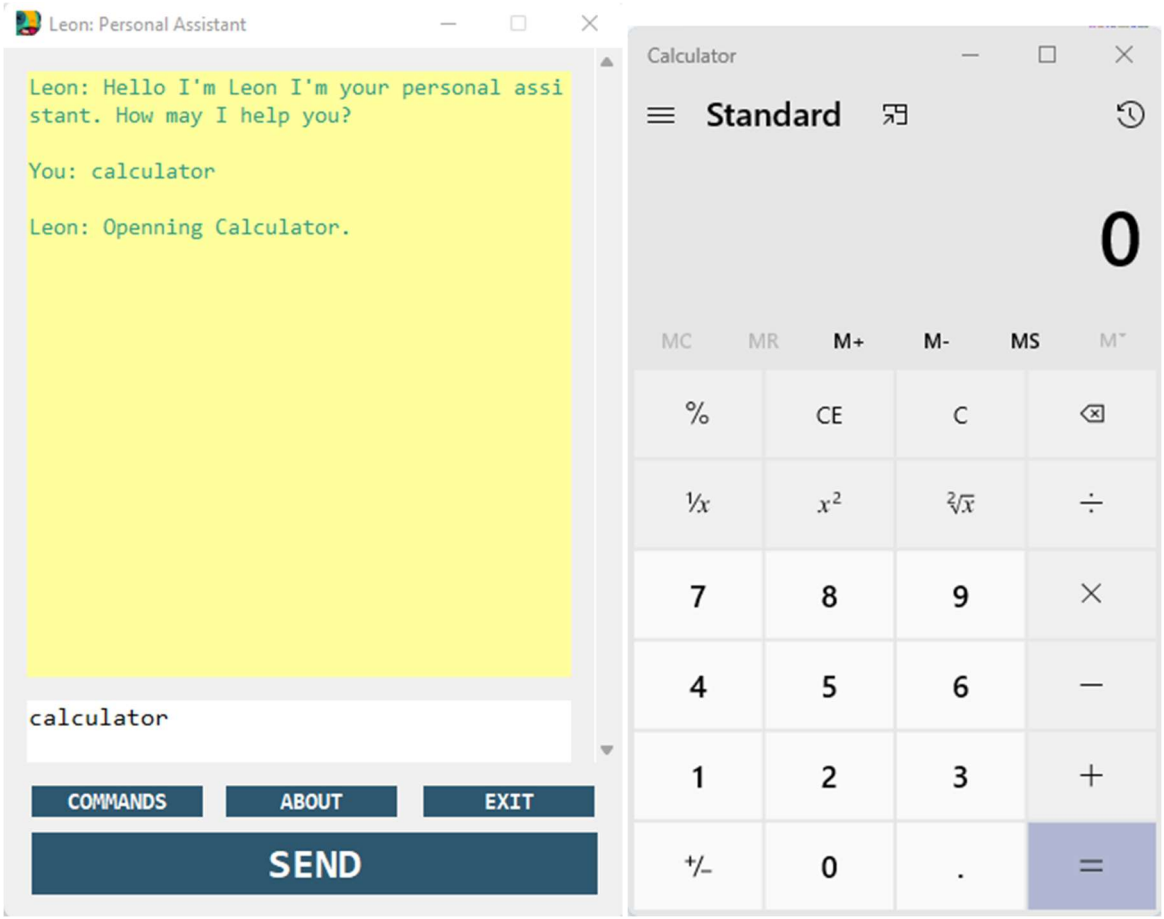
calc

Example:

calculator, show calc, calc etc..

When ***calculator*** is entered, then the calculator app is opened.

OUTPUT:



SOURCE CODE:

```
from tkinter import *
import pyttsx3
import pandas as pd
from datetime import *
import pyttsx3
import webbrowser
from pygame import mixer
import subprocess
from youtube_search import YoutubeSearch
engine = pyttsx3.init()
def speak(text):
    engine.setProperty("rate", 150)
    engine.say(text)
    engine.runAndWait()
def send():
    data = pd.read_csv("sources/trainer.csv")
    msg = e.get("1.0",'end-1c').strip()
    out = 0
    if msg != '':
        t.config(state=NORMAL)
        t.insert(END, "You: " + msg + '\n\n')
        if "open" in msg.lower() and out==0:
                mg = msg.replace(" ", "").replace("open", "")
                mg = mg.replace(" ", "")
                mg = mg.lower()
                t.insert(END, "Leon: Opening " + mg +".com"+ '\n\n')
                speak("Opening {}.com".format(mg))
                webbrowser.open('{}.com'.format(mg))
                out = 1
        elif "yt" in msg.lower() and out==0:
                mg = msg.replace("yt", "")
                mg = mg.title()
                t.insert(END, "Leon: Playing " + mg +" in youtube.com"+ '\n\n')

                speak("Playing {} in youtube.com".format(mg))
                results = YoutubeSearch('{}'.format(mg), max_results=1).to_dict()
                for v in results:
                    webbrowser.open('https://www.youtube.com/watch?v=' + v['id'])
                out = 1
        elif "date" in msg.lower() and out==0:
                dt = datetime.now()
                d = dt.strftime("%d")
                m = dt.strftime("%B")
                y = dt.strftime("%Y")
                t.insert(END, "Leon: Today's Date: "+"{} {} {}".format(d,m,y)+ '\n
\n')
```


```
                speak("Today's Date is {} {} {}".format(d,m,y))
                out = 1
        elif "time" in msg.lower() and out==0:
                dt = datetime.now()
                h = dt.strftime("%I")
                m = dt.strftime("%M")
                p = dt.strftime("%p")
                t.insert(END, "Leon: Current Time: "+"{}:{} {}".format(h,m,p)+ '\n
\n')
                speak("Current Time is {} {} {}".format(h,m,p))
                out = 1
        elif "weather" in msg.lower() and out==0:
                t.insert(END, "Bot: Looking for " + msg + '\n\n')
                speak("Opening weather in google.com")
                webbrowser.open('https://www.google.com/search?q={}'.format(msg))
                out = 1
        elif "music" in msg.lower() and out==0:
                t.insert(END, "Leon: Playing " + msg + " for you." +'\n\n')
                speak("Playing Music for you. Enjoy")
                mixer.init()
                mixer.music.load('sources/music/1.mp3')
                mixer.music.play()
                out = 1
        elif "mstop" in msg.lower() and out==0:
                t.insert(END, "Leon: Stopping music for you." +'\n\n')
                mixer.music.stop()
                speak("Stopped Muisc for you.")
                out = 1
        elif "files" in msg.lower() and out==0:
                t.insert(END, "Leon: Openning File Manager." +'\n\n')
                subprocess.call(["explorer.exe"])
                speak("Openning File Manager for you.")
                out = 1
        elif "calc" in msg.lower() and out==0:
                t.insert(END, "Leon: Openning Calculator." +'\n\n')
                subprocess.call(["calc.exe"])
                speak("Openning Calculator for you.")
                out = 1
        else:
            for i in range(len(data)):
                if data['input'].iloc[i] in msg.lower() and out==0:
                    t.insert(END, "Leon: " + data['output'].iloc[i] + '\n\n')
                    speak(data['output'].iloc[i])
                    t.config(state=DISABLED)
                    out = 1
        t.yview(END)
def About():
        top = Toplevel()
        top.title("About Leon's Developers")
```


```
        top.iconbitmap('sources/icon.ico')
        top.geometry("222x160")
        top.resizable(width=FALSE, height=FALSE)
        ta = Text(top, bd=0, height="8", width="50", font="Consolas",)
        ta.config(background="#FFFF9D",foreground="#319981", font=("Consolas", 11,
 'bold' ))
        ta.insert(END, ' Crafted by:\n\n')
        ta.insert(END, '\tHarish M\n\tGirish Raj M\n\tRahul V\n')
        ta.insert(END, '~For Python Mini Project~')
        ebot= Button(top, font=("Consolas",11,'bold'), text="CLOSE", width="12", h
eight=4,bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff', command=top.dest
roy)
        ebot.place(x=80, y=130, height=20, width=60)
        ta.place(x=10,y=10, height=110, width=202, anchor=NW)
def Commands():
        top = Toplevel()
        top.title("Leon's Commands")
        top.iconbitmap('sources/icon.ico')
        top.geometry("300x300")
        top.resizable(width=FALSE, height=FALSE)
        ta = Text(top, bd=0, height="8", width="50", font="Consolas",)
        ta.config(background="#FFFF9D",foreground="#319981", font=("Consolas", 11,
 'bold' ))
        ta.insert(END, 'Commands:\n\n')
        ta.insert(END, '  1. open - open name_of_website\n')
        ta.insert(END, '  2. yt - yt keywords_of_video\n')
        ta.insert(END, '  3. date - date\n')
        ta.insert(END, '  4. time - time\n')
        ta.insert(END, '  5. weather - weather\n')
        ta.insert(END, '  6. music - music\n')
        ta.insert(END, '  7. mstop - mstop\n')
        ta.insert(END, '  8. files - files\n')
        ta.insert(END, '  9. calc - calc\n')
        ta.insert(END, '  [commands - syntax]')
        ebot= Button(top, font=("Consolas",11,'bold'), text="CLOSE", width="12", h
eight=4,bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff', command=top.dest
roy)
        ebot.place(x=110, y=270, height=20, width=60)
        ta.place(x=10,y=10, height=250, width=280, anchor=NW)

layout = Tk()
layout.title("Leon: Personal Assistant")
layout.iconbitmap('sources/icon.ico')
layout.geometry("400x560")
layout.resizable(width=FALSE, height=FALSE)
t = Text(layout, bd=0, bg="#fff", height="8", width="50", font="Consolas",)
t.config(background="#FFFF9D",foreground="#319981", font=("Consolas", 11 ))
t.insert(END, "Leon: Hello I'm Leon I'm your personal assistant. How may I help yo
u? "+ '\n\n')
```


```
t.config(state=DISABLED)
sb = Scrollbar(layout, command=t.yview)
t['yscrollcommand'] = sb.set
e = Text(layout, bd=0, bg="#fff",width="29", height="3", font="Consolas")
s = Button(layout, font=("Consolas",20,'bold'), text="SEND", width="12", height=4,
bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff',command=send)
eb= Button(layout, font=("Consolas",11,'bold'), text="EXIT", width="12", height=4,
bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff', command=layout.destroy)
ab= Button(layout, font=("Consolas",11,'bold'), text="ABOUT", width="12", height=4
,bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff', command=About)
cb= Button(layout, font=("Consolas",11,'bold'), text="COMMANDS", width="12", heigh
t=4,bd=0, bg="#2d5870", activebackground="#2d5870",fg='#fff', command=Commands)
sb.place(x=380,y=0, height=460,width=15)
t.place(x=15,y=15, height=390, width=350)
e.place(x=15, y=420, height=40, width=350)
cb.place(x=18, y=475, height=20, width=110)
eb.place(x=270, y=475, height=20, width=110)
ab.place(x=143, y=475, height=20, width=110)
s.place(x=18, y=505, height=40, width=364)
speak("Hello I'm Leon. I'm your personal assistant. How may I help you?")
layout.mainloop()
```

FUNCTIONS USED IN THE SOURCE CODE [User-Defined]:

- ***speak():***
  speak() function is defined specifically for the module *pyttsx3.* Pyttsx3 is the module used for the voice of Personal Assistant. For the voice to say without any exceptions, the following code snippet is required.

  ```
  engine = pyttsx3.init()
  engine.setProperty("rate", 150)
  engine.say(“Hello”)
  engine.runAndWait()
  ```

  These above mention code is required to say a single sentence. Instead of using everywhere and in order to keep the program short function speak() is used.

  ```
  def speak(text):
      engine.setProperty("rate", 150)
      engine.say(text)
      engine.runAndWait()
  speak(“Hello”)
  ```

- ***send():***
  Inside the send() function the commands are defined under many if-elif-else cases. speak() functions is also used in send() function. The send() function is a heart to ***Leon***. Without send function ***Leon*** will not work.

```
def send():
    data = pd.read_csv("sources/trainer.csv")
    msg = e.get("1.0",'end-1c').strip()
    out = 0
    if msg != '':
        t.config(state=NORMAL)
        t.insert(END, "You: " + msg + '\n\n')
        if "open" in msg.lower() and out==0:
         .
         .
         .
         .
        elif ... in msg.lower() and out==0:
         .
         .
         .
         .
      else:
         for ...
         .
         .
         .
         .
         out = 1
t.yview(END)
```

In the above snippet, the variable out is used for not performing the other cases. This can lead to conflicts when the user inputs two commands at a time.

Example: When user enters ***"open music"***, where open and music both are commands, in this case ***"open"*** is taken as a command and ***"music"*** is taken as value for the command.

- ***About():***
  The About() function acts as another layout linked to this main app. In the About() function there are names of the team members. This event takes place when the button ***"ABOUT"*** is clicked.
- ***Commands()***
  The Commands() function acts as another layout linked to this main app. In the Commands() function there are commands used in ***Leon*** and their syntax. This event takes place when the button ***"ABOUT"*** is clicked.

Google Drive link for the project files:

https://drive.google.com/file/d/1NmeGMJfJ1SheUdxnam8zSpcnuzOWsmHh/view?usp=sharing