Mini Project
(2019-2020)

Report
on

Iron Man Jarvis AI Desktop Voice Assistant

Institute of Engineering & Technology

Team Members:

Aditi Gupta(171500017)
Ayush Pandey(171500077)
Vaishali Jain(171500369)

Supervised By:
Mr. Piyush Vashistha
(Assistant Professor)

Department of Computer Engineering & Applications

GLA University
Mathura-281406, India

Department of computer Engineering and Applications
GLA UNIVERSITY, MATHURA
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

We hereby declare that the work which is being presented in the Mini Project"Iron Man Jarvis AI Desktop Voice Assistant", in partial fulfillment of the requirements for Mini Project viva voice, is an authentic record of our own work carried under the supervision of "Mr. Piyush Vashistha".

Signature of Candidate:

Name of Candidate:

Aditi Gupta(171500017)

Ayush Pandey(171500077)

Vaishali Jain(171500369)

Course: B.Tech. (CSE)

Year: 3rd

Semester: 6th

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. On the completion of this project. We would like to extend our sincere thanks to all of them. We are highly indebted to this project guide **Mr. Piyush Vashistha Assistant Professor of Department of Computer Engineering and Applications of GLA University** for their guidance and constant supervision as well as for providing necessary information regarding the project. We wish to extend our sincere gratitude to **Prof. Anand Singh Jalal, Head of Department of Computer Engineering and Applications and faculty of CEA Department of GLA University** for their guidance, encouragement and give this opportunity and valuable suggestion which prove extremely useful and helpful in the completion of this report. We would also like to thank all those who directly or indirectly supported or helped us in completing our project in time. We would like to express our gratitude towards our parents and member of our college for their kind cooperation and encouragement which helped me in completion of this project. All of them have willingly helped us out with their abilities.

Thanks

Aditi Gupta
Ayush Pandey
Vaishali Jain

Abstract

In the Modern Era of fast moving technology we can do things which we never thought we could do before but, to achieve and accomplish these thoughts there is a need for a platform which can automate all our tasks with ease and comfort. Thus we humans developed applications like Personal Voice Assistant having the ability to inter act with the surroundings just by one of the materialistic form of human interaction i.e .HUMAN VOICE. The most famous application of android mobile phone is “Google Assistant”, “Google Voice Search” which is developed by the Google .Various applications like Microsoft Cortana, Amazon Alexa is also used as an voice assistant .The voice application of iphone is “SIRI” which helps the end user to communicate end-user mobile with voice and it also responds to the voice commands of the user. We are going to develop a web application were the voice assistant would be available for a particular desktop user. It can change the way of inter actions between user and the system. The Application is being designed in such a way that all the services provided by the system are accessible by the desktop user on the user's voice commands.

Table Of Content

1.0 Introduction

About the Project

As we know Python is a suitable language for script writers and developers. Let's write a script for Personal Voice Assistant using Python. The query for the assistant can be manipulated as per the user's need.

The implemented assistant can open up the application (if it's installed in the system), search Google, Wikipedia and YouTube about the query, calculate any mathematical question, etc by just giving the voice command. We can process the data as per the need or can add the functionality, depends upon how we code things.

We are using Google speech recognition API and google text to speech for voice input and output respectively.

Also, for calculating mathematical expression WolframAlpha API can be used.

Playsound Package is used to play the saved mp3 sound from the system.

2.0 About Desktop Voice Assistant

What is a Voice Assistant?

A **voice assistant** or **intelligent personal assistant** is a software agent that can perform tasks or services for an individual based on verbal commands i.e. by interpreting human speech and respond via synthesized voices. Users can ask their assistants' questions, control home automation devices, and media playback via voice, and manage other basic tasks such as email, to-do lists, open or close any application etc with verbal commands.

Let me give you the example of Braina (Brain Artificial) which is an intelligent personal assistant, human language interface, automation and **voice recognition software** for Windows PC. Braina is a multi-functional AI software that allows you to interact with your computer using **voice commands** in most of the languages of the world. Braina also allows you to accurately convert speech to text in over 100 different languages of the world.

Motivation

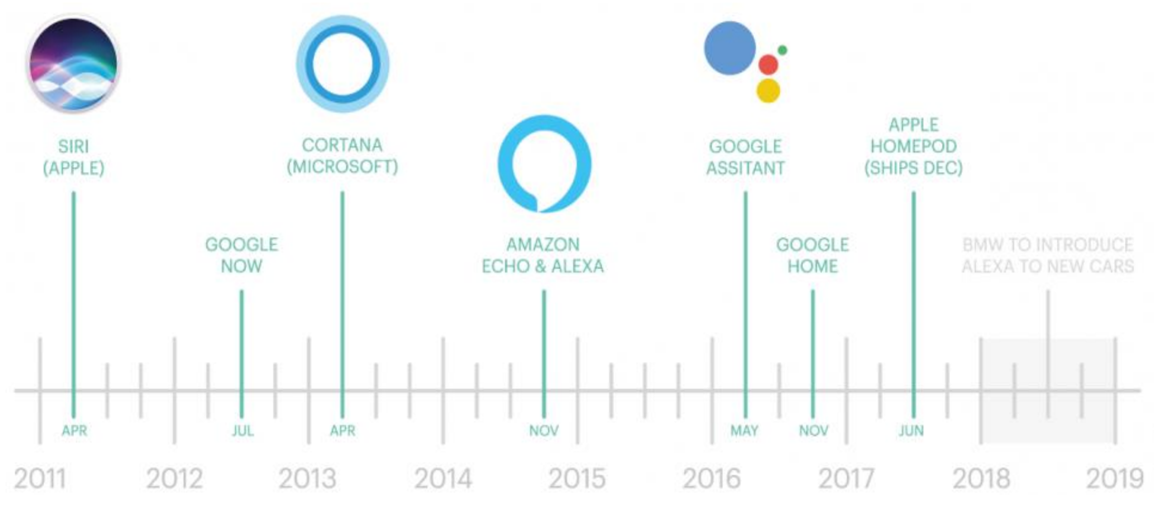
Who doesn't want to have the luxury to own an assistant who always listens for your call, anticipates your every need, and takes action when necessary? That luxury is now available thanks to artificial intelligence-based voice assistants.

Voice assistants come in somewhat small packages and can perform a variety of actions after hearing your command. They can turn on lights, answer questions, play music, place online orders and do all kinds of AI-based stuff.

Voice assistants are not to be confused with virtual assistants, which are people who work remotely and can, therefore, handle all kinds of tasks. Rather, voice assistants are technology based. As voice assistants become more robust, their utility in both the personal and business realms will grow as well.

3.0 History of Voice Assistants

History of Voice Assistants



A modern history of Voice Assistants

In recent times, Voice assistants got the major platform after Apple integrated the most astonishing Virtual Assistant — Siri which is officially a part of Apple Inc. But the timeline of greatest evolution began with the year 1962 event at the Seattle World Fair where IBM displayed a unique apparatus called Shoebox. It was the actual size of a shoebox and could perform scientific functions and can perceive 16 words and also speak them in the human recognizable voice with 0 to 9 numerical digits.

During the period of the 1970s, researchers at Carnegie Mellon University in Pittsburgh, Pennsylvania — with the considerable help of the U.S Department of Defence and its Defence Advanced Research Projects Agency (DARPA) — made Harpy. It could understand almost 1,000 words, which is approximately the vocabulary of a three-year-old child.

Big organizations like Apple and IBM sooner in the 90s started to make things that utilized voice acknowledgment. In 1993, Macintosh began to building speech recognition with its Macintosh PCs with Plain Talk.

In April 1997, Dragon Naturally Speaking was the first constant dictation product which could comprehend around 100 words and transform it into readable content.

4.0 System Requirements Specification

Hardware Requirements Specification

Processor: Intel Pentium III or later

Main Memory (RAM) : 8 GB

Cache Memory: 512 KB

Keyboard: 108 Keys

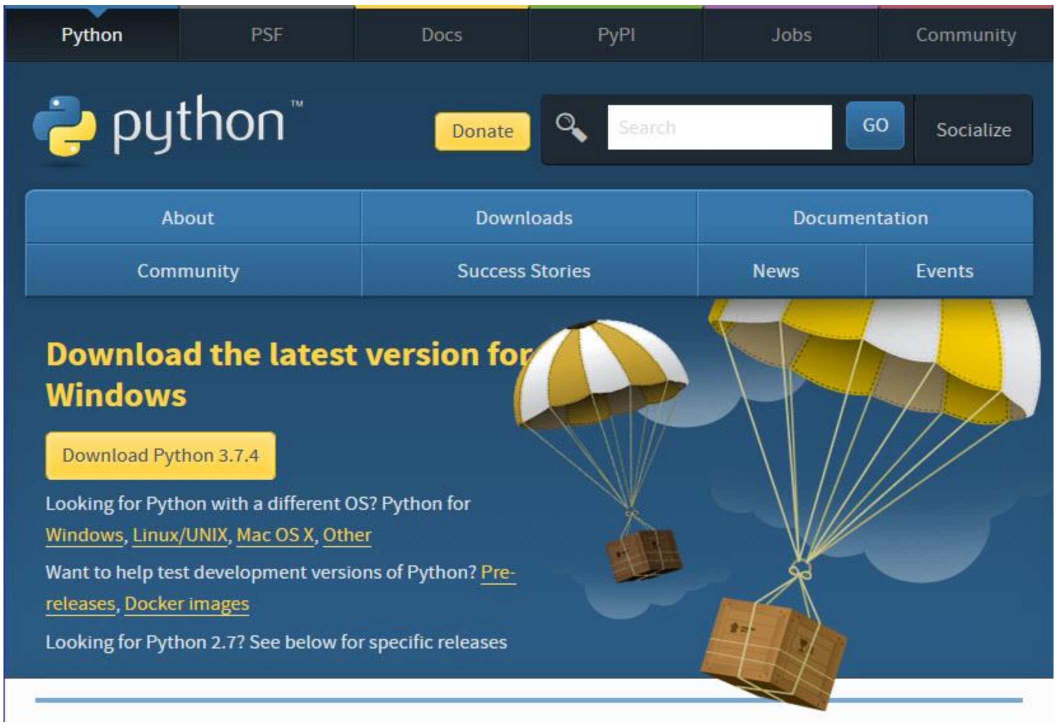
Software Requirements Specification

Technology: Python 3, Different Libraries

Platform: Jupyter Notebook

Operating System: Windows 7, 8, 9, 10, XP

Installation of Python

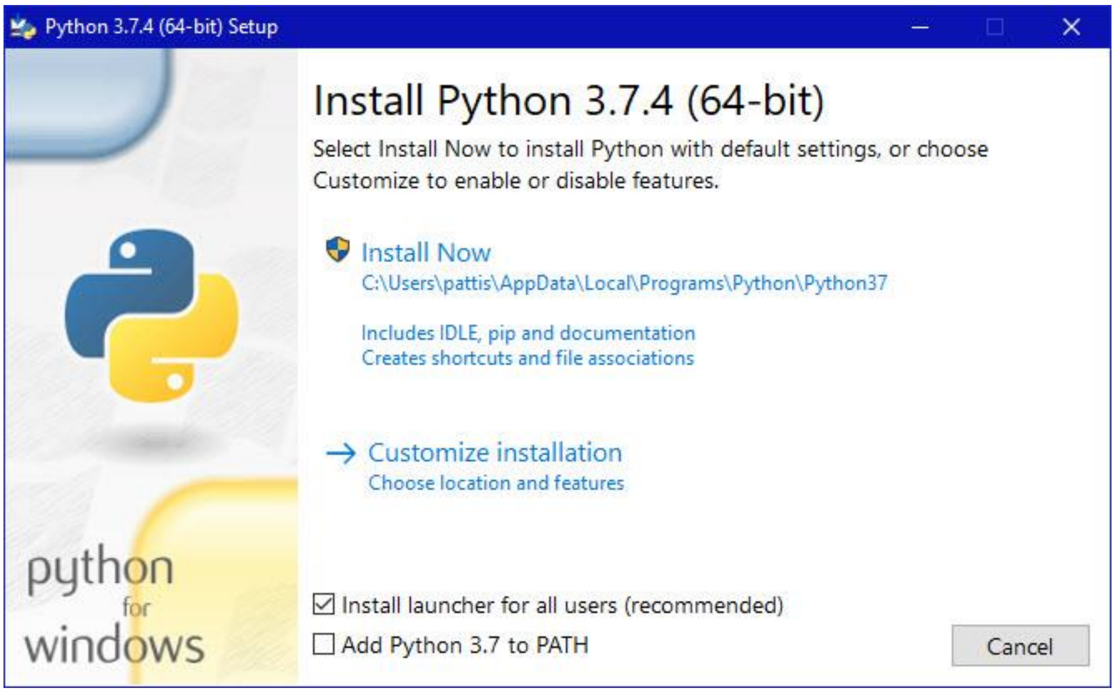


Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\pattis\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.7 to PATH

Installation of Jupyter Notebook

For new users, we highly recommend installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook. To run the notebook:

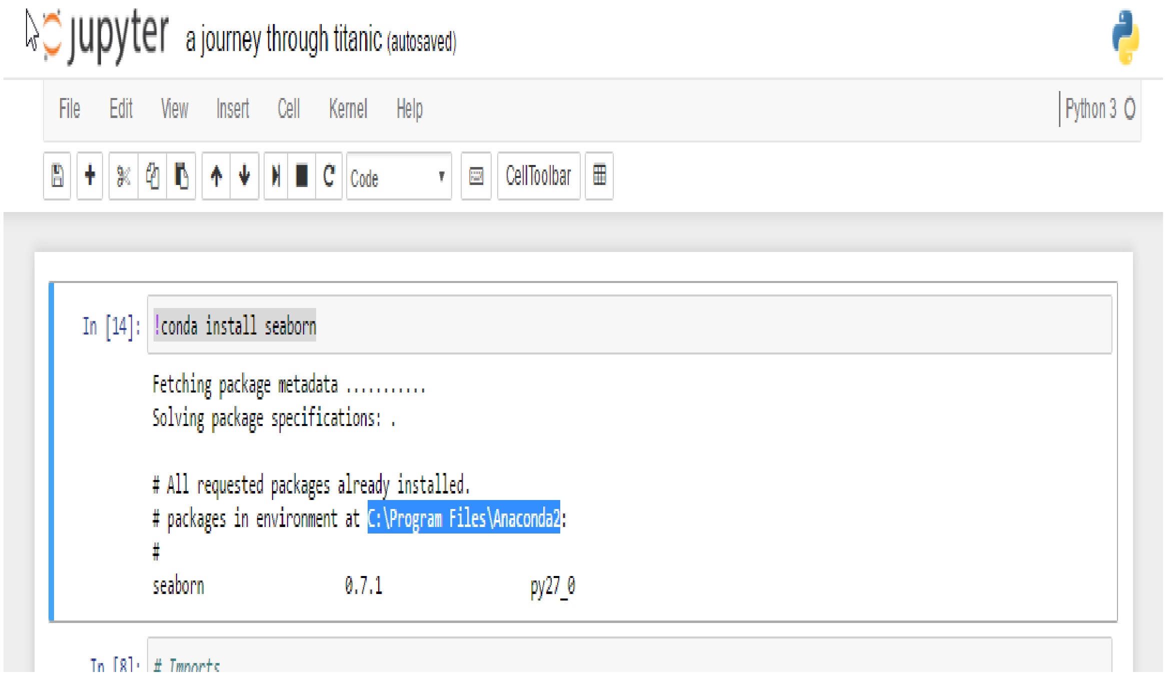
As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

```
pip3 install --upgrade pip
```

Then install the Jupyter Notebook using:

```
pip3 install jupyter
```



```
In [14]: !conda install seaborn

Fetching package metadata ..........
Solving package specifications: .

# All requested packages already installed.
# packages in environment at C:\Program Files\Anaconda2:
#
seaborn                   0.7.1                    py27_0

In [8]: # Imports
```

Dependencies and requirements:

Install all these python libraries :

pip install SpeechRecognition

pip install beautifulsoup4

pip install vlc

pip install youtube-dl

pip install pyowm

pip install wikipedia

5.0 System Implementation

1. Open the subreddit Reddit in the browser.

The user will give any command to open any subreddit from Reddit and the command should be “Hey Jarvis! Can you please open Reddit subreddit_name”. only the quoted phrase should be used as it is. You can use any kind of prefix, just take care of the italic bold phrase.

How it works : If you have said the phrase open reddit in your command then it will search for subreddit name in the user command using re.search(). The subreddit will be searched using www.reddit.com and will get opened in the browser using pythons Web browser module. The Web browser module provides a high-level interface to allow displaying Web-based documents to users.

2. Open any website in the browser.

You can open any website just be saying “open website.com” or “open website.org”. For example: “Please open facebook.com” or “Hey, can you open linkedin.com” like this you can ask Jarvis to open any website for you.

How it works : If you have said the word open in your command then it will search for website name in the user command using re.search(). Next, it will append the website name to https://www. and using web browser module the complete URL gets opened in the browser.

3. Send Email.

You can also ask your desktop assistant to send the email.

How it works : If you have said the word email in your command then the bot will ask for receipient, If my response is rajat, the bot will use pthons smtplib library.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. Sending mail is done with Python's smtplib using an SMTP server. First it will intiate gmail SMTP using smtplib. SMTP(), then identify the server using ehlo() function, then encypting the session starttls(), then login to your mailbox using login(), then sending the message using sendmail().

4. Launch any system application.

Say "launch calendar" or "can you please launch skype" or "Jarvis launch finder" etc. and Jarvis will launch that system application for you.

How it works : If you have said the word launch in your command then it will search for application name(if it is present in your system) in the user command using re.search(). It will then append the suffix ".app" to the application name. Now your application name is for example say calender.app(In macOS the executable files end with extension .app unlike in Windows which ends with .exe). So the executable application name will be launched using python subprocess's Popen() function. The subprocess module enables you to start new applications from your Python program.

5. Tells you the current time.

"Jarvis can you tell me the current time ?" or "what is the time now ?" and Jarvis will tell you the current time of your timezone.

How it works : Its pretty simple

6. Greetings/ leave

Say " hello Jarvis" to greet your voice assistant or when you want the program to terminate say something like "shutdown Jarvis" or "Jarvis please shutdown" etc.

How it works : If you have said the word hello in your command, then depending on the time of the day, the bot will greet the user. If the time is more than 12 noon, the bot will respond "Hello Sir. Good afternoon", likewise if the time is more than 6 ck pm, the bot will respond "Hello Sir. Good evening". And when you give command as shutdown, sys.exit() will be called to terminate the program.

7. Tells you latest news feeds

Jarvis can also tell you the latest news update. The user just has to say "Jarvis what are the top news for today?" or "tell me the news for today".

How it works : If you have said the phrase "news for today" in your command then it will scrape data using Beautiful Soup from Google News RSS() and read it for you. For convenience I have set number of news limit to 15.

8. Tells you the current weather and temperature of almost any city

Jarvis can also tell you the weather, maximum and minimum temperature of any city around the world. The user just needs to say something like "what is the current weather in London" or "tell me the current weather in Delhi".

How it works : If you have said the phrase "current weather" in your command then it will search for city name using re.search(). I have used pythons pyowm library to get the weather of any city. get_status() will tell you about the weather condition like haze, cloudy, rainy etc and get_temperature() will tell you about the max and min temperature of the city.

9. Play you a song on VLC media player

This feature allows your voice bot to play your desired song in VLC media player. The user will say "jarvis play me a song", the bot will ask "What song shall I play Sir?". Just say the name of the song and jarvis will download the song from youtube in your local drive, play that song on the VLC media player and if you again play a song the previously downloaded song will get deleted automatically.

How it works : If you have said the phrase “play me a song” in your command, then it will ask you what video song to play. The song you will ask will be searched in youtube.com, If found than the song will be downloaded in your local directory using pythons youtube_dl library. The youtube-dl is a command-line program to download videos from YouTube.com and a few more sites. Now the song will be played as soon as it gets downloded using pythons VLC library and play(path_to__videosong) module actually plays the song.

Now if the next time you ask for any other song, the local directory will be flushed and a new song will be downloaded in that directory.

10. Change desktop wallpaper.

You guys can also change your desktop wallpaper using this feature. When you say something like “change wallpaper” or “jarvis please change wallpaper” the bot will download random wallpaper from unsplash.com and sets it as your desktop background.

How it works : If you have said the phrase “change wallpaper” in your command, the program will download a random wallpaper from unsplash.com, store it in local directory and set it as your desktop wallpaper using subprocess.call(). I have used unsplash API to get access to its content.

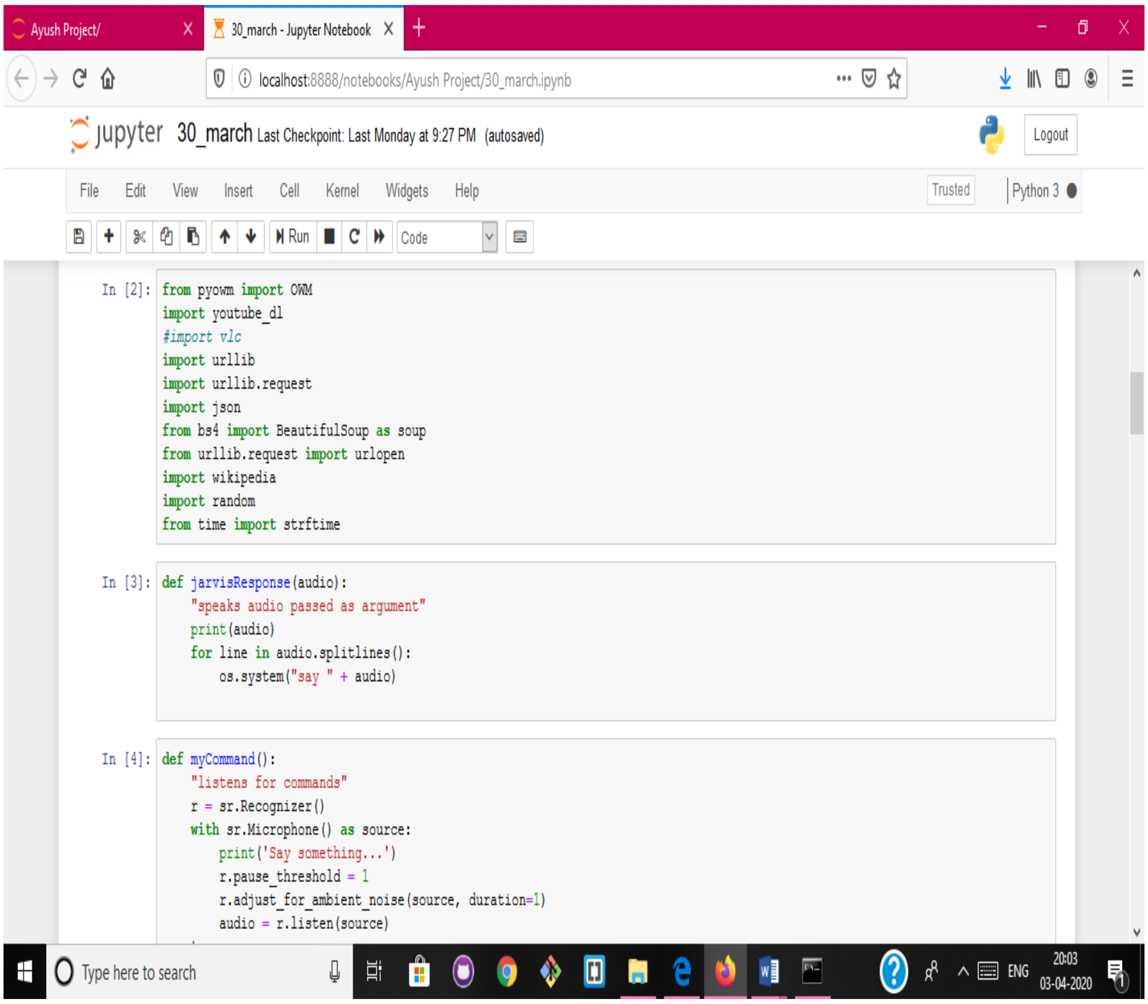
Now if the next time you ask to change the wallpaper again, your local directory will be flushed and a new wallpaper will be downloaded in that directory.

11. Tells you about almost anything you ask.

Your bot can fetch details of almost anything you ask her. Like “jarvis tell me about Google” or “Please tell me about Supercomputers” or “please tell me about the Internet”. So as you can see you can ask about almost anything.

How it works : If you have said the phrase tell me about in your command then it will search for the keyword in the user command using re.search(). Using pythons wikipedia library it will search for that topic and extract first 500 characters(if you don’t specify the limit the bot will read the whole page for you). Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia.

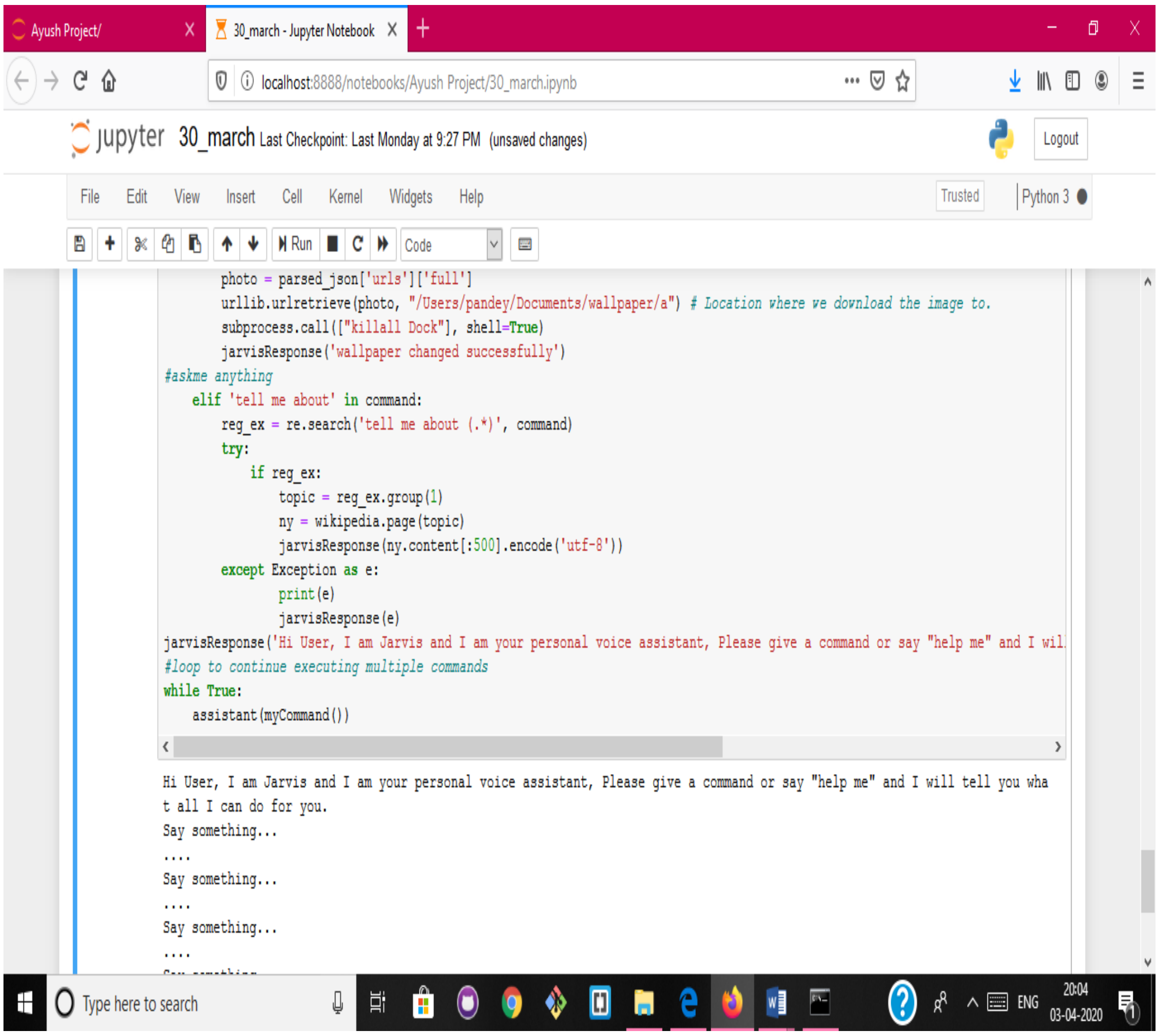
Execution


Ayush Project/
30_march - Jupyter Notebook
localhost:8888/notebooks/Ayush Project/30_march.ipynb
jupyter 30_march Last Checkpoint: Last Monday at 9:27 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
ENG
20:03
03-04-2020

```
In [2]: from pyowm import OWM
        import youtube_dl
        #import vlc
        import urllib
        import urllib.request
        import json
        from bs4 import BeautifulSoup as soup
        from urllib.request import urlopen
        import wikipedia
        import random
        from time import strftime

In [3]: def jarvisResponse(audio):
            "speaks audio passed as argument"
            print(audio)
            for line in audio.splitlines():
                os.system("say " + audio)

In [4]: def myCommand():
            "listens for commands"
            r = sr.Recognizer()
            with sr.Microphone() as source:
                print('Say something...')
                r.pause_threshold = 1
                r.adjust_for_ambient_noise(source, duration=1)
                audio = r.listen(source)
```


Ayush Project/
30_march - Jupyter Notebook
localhost:8888/notebooks/Ayush Project/30_march.ipynb
jupyter 30_march Last Checkpoint: Last Monday at 9:27 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Type here to search
ENG
20:04
03-04-2020

```
        photo = parsed_json['urls']['full']
        urllib.urlretrieve(photo, "/Users/pandey/Documents/wallpaper/a") # Location where we download the image to.
        subprocess.call(["killall Dock"], shell=True)
        jarvisResponse('wallpaper changed successfully')
#askme anything
    elif 'tell me about' in command:
        reg_ex = re.search('tell me about (.*)', command)
        try:
            if reg_ex:
                topic = reg_ex.group(1)
                ny = wikipedia.page(topic)
                jarvisResponse(ny.content[:500].encode('utf-8'))
        except Exception as e:
                print(e)
                jarvisResponse(e)
jarvisResponse('Hi User, I am Jarvis and I am your personal voice assistant, Please give a command or say "help me" and I wil
#loop to continue executing multiple commands
while True:
    assistant(myCommand())
```

```
Hi User, I am Jarvis and I am your personal voice assistant, Please give a command or say "help me" and I will tell you wha
t all I can do for you.
Say something...
....
Say something...
....
Say something...
....
```

6.0 Testing

The implementation phase of software development is concerned with translating design specification into source code. The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages.

The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description. Each team member must understand the objectives of implementation.

Terms in Testing Fundamental

Error

The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also to used to refer to human action that result in software containing a defect or fault.

Fault

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

Failure

Failure is the inability of a system or component to perform a required function according to its specifications.

a. Unit Testing

The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system.

A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

b. Module Testing

A module and encapsulates related component. So can be tested without other system module.

There are four categories of tests that a programmer will typically perform on a program unit.

i Functional test

ii Performance test

iii Stress test

iv Structure test

Functional Test

Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

Performance Test

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit.

Stress Test

Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

Structure Test

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test date to exercise those paths, determining the test coverage criterion to be used, executing the test, and measuring the test coverage achieved when the test cases are exercised.

7.0 Conclusion: What the future holds

Throughout the history of computing, user interfaces have become progressively natural to use. The screen and keyboard were one step in this direction. The mouse and graphical user interface were another. Touch screens are the most recent development. The next step will most likely consist of a mix of augmented reality, gestures and voice commands. After all, it is often easier to ask a question or have a conversation than it is to type something or enter multiple details in an online form.

The more a person interacts with voice-activated devices, the more trends, and patterns the system identifies based on the information it receives. Then, this data can be utilized to determine user preferences and tastes, which is a long-term selling point for making a home smarter. Google and Amazon are looking to integrate voice-enabled artificial intelligence capable of analyzing and responding to human emotion.

8.0 References/Bibliography

https://towardsdatascience.com/build-your-first-voice-assistant-85a5a49f6cc1/
https://stackoverflow.com/
https://www.geeksforgeeks.org/personal-voice-assistant-in-python/
https://clearbridgemobile.com/7-key-predictions-for-the-future-of-voice-assistants/

Appendices

Iron Man Jarvis AI Desktop Voice Assistant

```
import speech_recognition as sr

import os

import sys

import re

import webbrowser

import smtplib

import requests

import subprocess

from pyowm import OWM

import youtube_dl

#import vlc

import urllib

import urllib.request

import json
```

```
from bs4 import BeautifulSoup as soup

from urllib.request import urlopen

import wikipedia

import random

from time import strftime

def jarvisResponse(audio):

    "speaks audio passed as argument"

    print(audio)

    for line in audio.splitlines():

        os.system("say " + audio)

def myCommand():

    "listens for commands"

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print('Say something...')

        r.pause_threshold = 1

        r.adjust_for_ambient_noise(source, duration=1)

        audio = r.listen(source)

    try:
        command = r.recognize_google(audio).lower()

        print('You said: ' + command + '\n')

    #loop back to continue to listen for commands if unrecognizable
speech is received

    except sr.UnknownValueError:

        print('....')

        command = myCommand();

    return command

def assistant(command):

    "if statements for executing commands"

#open subreddit Reddit

    if 'open reddit' in command:

        reg_ex = re.search('open reddit (.*)', command)

        url = 'https://www.reddit.com/'

        if reg_ex:

            subreddit = reg_ex.group(1)

            url = url + 'r/' + subreddit

        webbrowser.open(url)

        jarvisResponse('The Reddit content has been opened for you Sir.')
    elif 'shutdown' in command:

        jarvisResponse('Bye bye Sir. Have a nice day')

        sys.exit()

#open website

    elif 'open' in command:

        reg_ex = re.search('open (.+)', command)

        if reg_ex:

            domain = reg_ex.group(1)

            print(domain)

            url = 'https://www.' + domain

            webbrowser.open(url)

            jarvisResponse('The website you have requested has been
opened for you Sir.')

        else:

            pass

#greetings

    elif 'hello' in command:

        day_time = int(strftime('%H'))

        if day_time < 12:
            jarvisResponse('Hello Sir. Good morning')
        elif 12 <= day_time < 18:
            jarvisResponse('Hello Sir. Good afternoon')
        else:
            jarvisResponse('Hello Sir. Good evening')
    elif 'help me' in command:
        jarvisResponse("""
        You can use these commands and I'll help you out:
        1. Open reddit subreddit : Opens the subreddit in default browser.
        2. Open xyz.com : replace xyz with any website name
        3. Send email/email : Follow up questions such as recipient name, content will be asked in order.
        4. Current weather in {cityname} : Tells you the current condition and temperture
        5. Hello
        6. play me a video : Plays song in your VLC media player
        7. change wallpaper : Change desktop wallpaper
        8. news for today : reads top news of today
        9. time : Current system time
        10. top stories from google news (RSS feeds)
        11. tell me about xyz : tells you about xyz

        """)

#joke

    elif 'joke' in command:

        res = requests.get(

                'https://icanhazdadjoke.com/',

                headers={"Accept":"application/json"})

        if res.status_code == requests.codes.ok:

            jarvisResponse(str(res.json()['joke']))

        else:

            jarvisResponse('oops!I ran out of jokes')

#top stories from google news

    elif 'news for today' in command:

        try:

            news_url="https://news.google.com/news/rss"

            Client=urlopen(news_url)

            xml_page=Client.read()

            Client.close()

            soup_page=soup(xml_page,"xml")
            news_list=soup_page.findAll("item")

            for news in news_list[:15]:

                jarvisResponse(news.title.text.encode('utf-8'))

        except Exception as e:

                print(e)

#current weather

    elif 'current weather' in command:

        reg_ex = re.search('current weather in (.*)', command)

        if reg_ex:

            city = reg_ex.group(1)

            owm = OWM(API_key='ab0d5e80e8dafb2cb81fa9e82431c1fa')

            obs = owm.weather_at_place(city)

            w = obs.get_weather()

            k = w.get_status()

            x = w.get_temperature(unit='celsius')

            jarvisResponse('Current weather in %s is %s. The maximum
temperature is %0.2f and the minimum temperature is %0.2f degree
celcius' % (city, k, x['temp_max'], x['temp_min']))

#time

    elif 'time' in command:
        import datetime

        now = datetime.datetime.now()

        jarvisResponse('Current time is %d hours %d minutes' %
(now.hour, now.minute))

    elif 'email' in command:

        jarvisResponse('Who is the recipient?')

        recipient = myCommand()

        if 'ayush' in recipient:

            jarvisResponse('What should I say to him?')

            content = myCommand()

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

            mail.ehlo()

            mail.starttls()

            mail.login('your_email_address', 'your_password')

            mail.sendmail('sender_email', 'receiver_email', content)

            mail.close()

            jarvisResponse('Email has been sent successfuly. You can check
your inbox.')

        else:

            jarvisResponse('I don\'t know what you mean!')
```

```
#launch any application

    elif 'launch' in command:

        reg_ex = re.search('launch (.*)', command)

        if reg_ex:

            appname = reg_ex.group(1)

            appname1 = appname+".app"

            subprocess.Popen(["open", "-n", "/Applications/" + appname1],
stdout=subprocess.PIPE)

            jarvisResponse('I have launched the desired application')

#play youtube song

    elif 'play me a song' in command:

        path = '/Users/nageshsinghchauhan/Documents/videos/'

        folder = path

        for the_file in os.listdir(folder):

            file_path = os.path.join(folder, the_file)

            try:

                if os.path.isfile(file_path):

                    os.unlink(file_path)

            except Exception as e:
```

```
            print(e)

        jarvisResponse('What song shall I play Sir?')

        mysong = myCommand()

        if mysong:

            flag = 0

            url = "https://www.youtube.com/results?search_query=" +
mysong.replace(' ', '+')

            response = urllib2.urlopen(url)

            html = response.read()

            soup1 = soup(html,"lxml")

            url_list = []

            for vid in soup1.findAll(attrs={'class':'yt-uix-tile-link'}):

                if ('https://www.youtube.com' +
vid['href']).startswith("https://www.youtube.com/watch?v="):

                    flag = 1

                    final_url = 'https://www.youtube.com' + vid['href']

                    url_list.append(final_url)

                    url = url_list[0]

                    ydl_opts = {}

                    os.chdir(path)
                with youtube_dl.YoutubeDL(ydl_opts) as ydl:

                    ydl.download([url])

                    vlc.play(path)

                if flag == 0:

                    jarvisResponse('I have not found anything in Youtube ')

#change wallpaper

    elif 'change wallpaper' in command:

        folder = '/Users/pandey/Documents/wallpaper/'

        for the_file in os.listdir(folder):

            file_path = os.path.join(folder, the_file)

            try:

                if os.path.isfile(file_path):

                    os.unlink(file_path)

            except Exception as e:

                print(e)

        api_key =
'fd66364c0ad9e0f8aabe54ec3cfbed0a947f3f4014ce3b841bf2ff6e20948
795'

        url = 'https://api.unsplash.com/photos/random?client_id=' +
api_key #pic from unspalsh.com
```

```
        f = urllib.request.urlopen(url)

        json_string = f.read()

        f.close()

        parsed_json = json.loads(json_string)

        photo = parsed_json['urls']['full']

        urllib.urlretrieve(photo, "/Users/pandey/Documents/wallpaper/a")
# Location where we download the image to.

        subprocess.call(["killall Dock"], shell=True)

        jarvisResponse('wallpaper changed successfully')

#askme anything

    elif 'tell me about' in command:

        reg_ex = re.search('tell me about (.*)', command)

        try:

            if reg_ex:

                topic = reg_ex.group(1)

                ny = wikipedia.page(topic)

                jarvisResponse(ny.content[:500].encode('utf-8'))

        except Exception as e:

                print(e)
            jarvisResponse(e)

jarvisResponse('Hi User, I am Jarvis and I am your personal voice assistant, Please give a command or say "help me" and I will tell you what all I can do for you.')

#loop to continue executing multiple commands

while True:

   assistant(myCommand())
```