**Technical Training Module**

**Project Report**

**On**

**“…………… ( AI PERSONAL ASSISTANT)……………..”**

**Using**

**…..(Python Programming Language.)………..**

**Submitted by:**

**Name: AMRIT RAJ SHARMA; ANMOL SADANA;KRISHNA AGARWAL**

**Branch:C.S.E.**

**Semester: Fourth semester.**

**Submitted To:**

**Mr. Rajat Tyagi**

**( Mentor – CETPA Infotech Ltd. )**

**Mr. Satendra Kumar**

**(H.O.D – CSE Deptt. – Quantum School of Technology, Roorkee )**

**ACKNOWLEDGEMENT**

First of all we are very thankful to our prestigious college Quantum School of technology,for bringing this great training in our reach for learning, moreover we would also like to express our gratitude towards cetpa infotech for their training and guidance in python programming language. Last but not the least we would also like to thank our trainers for teaching us in a friendly and amazing manner(with no doubt that we not only learned but also enjoyed a lot in the training sessions).

Student name: ***ANMOL SADANA; AMRIT RAJ SHARMA;***

***KRISHNA AGARWAL.***

Submitted to:-

***Mr. Rajat Tyagi***

***Mr. Satendra Kumar***

**PROJECT TOPIC :**

***AI PERSONAL ASSISTANT***

TABLE OF CONTENTS :

**1.ABSTRACT**

**2.INTRODUCTION**

2.1.Objective

**3. PROJECT DESCRIPTION**

3.1. Problem Definition

3.2 Overview of the project 3.3 Module Description 3.4 Data Flow Diagram(DFD) 3.5 Use case Diagram 3.6 Input Design 3.7 Output Design

**4. SYSTEM IMPLEMENTATION**

4.1 Purpose 4.2 System Maintenance

**5.CONCLUSIONS AND FUTURE ENHANCEMENT**  5.1. Conclusion 5.2. Scopes for Future Enhancement

**6. APPENDICES**

6.1 Program Listing

6.2 Screen Shots

**Abstract**

The subject of the paper is a personal assistant design. The goal of personal assistant is a daily activities list planning. The personal assistant will assist the costumer (human operator) for planning an optimum succession of desired activities. The solution is based on objective and subjective information. The paper focus on problem definition, mathematical structure of the concepts which are involved and a first solution analyze.

**Introduction**

The basic idea behind this project is to create a simple stand-alone application  
that helps less tech savvy people in the world to use the computer without feeling ignorant or computer illiterate.  
Computers have became a very important devices and as well as less expensive over time.  
The application works same like Siri/ Google Assistant etc. But the application deals with the computer itself mainly.  
The U.I of the application is self-explanatory and minimal.  
Currently it takes text as input as most of the people are not very good at speaking.

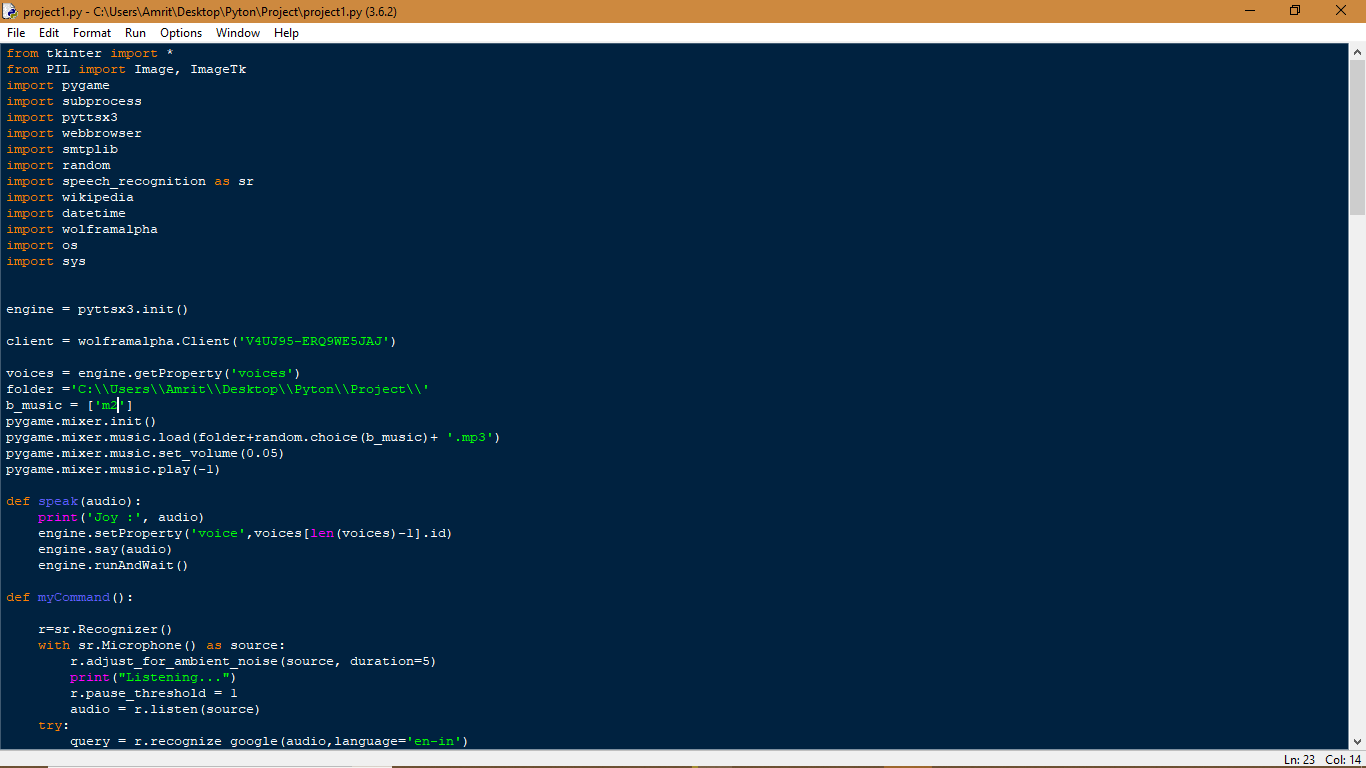
**Problem definition:**

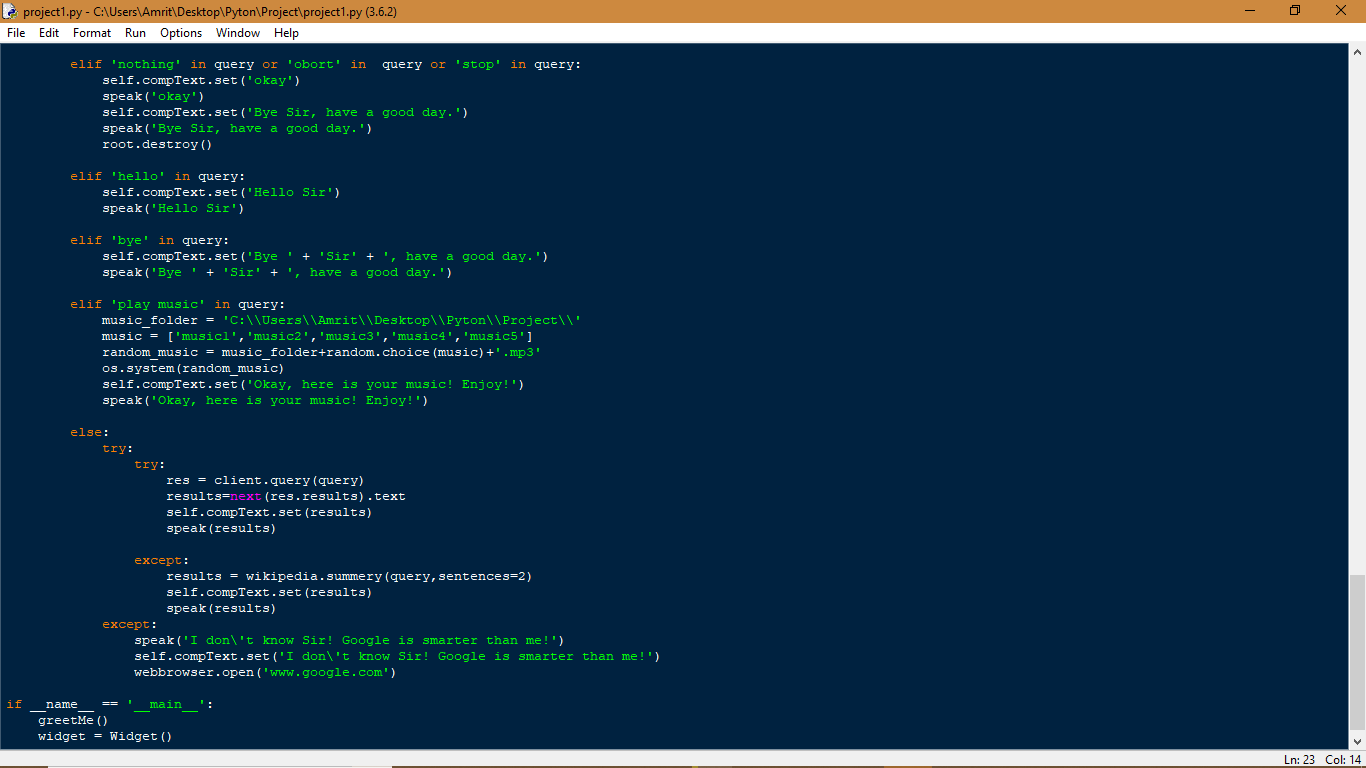
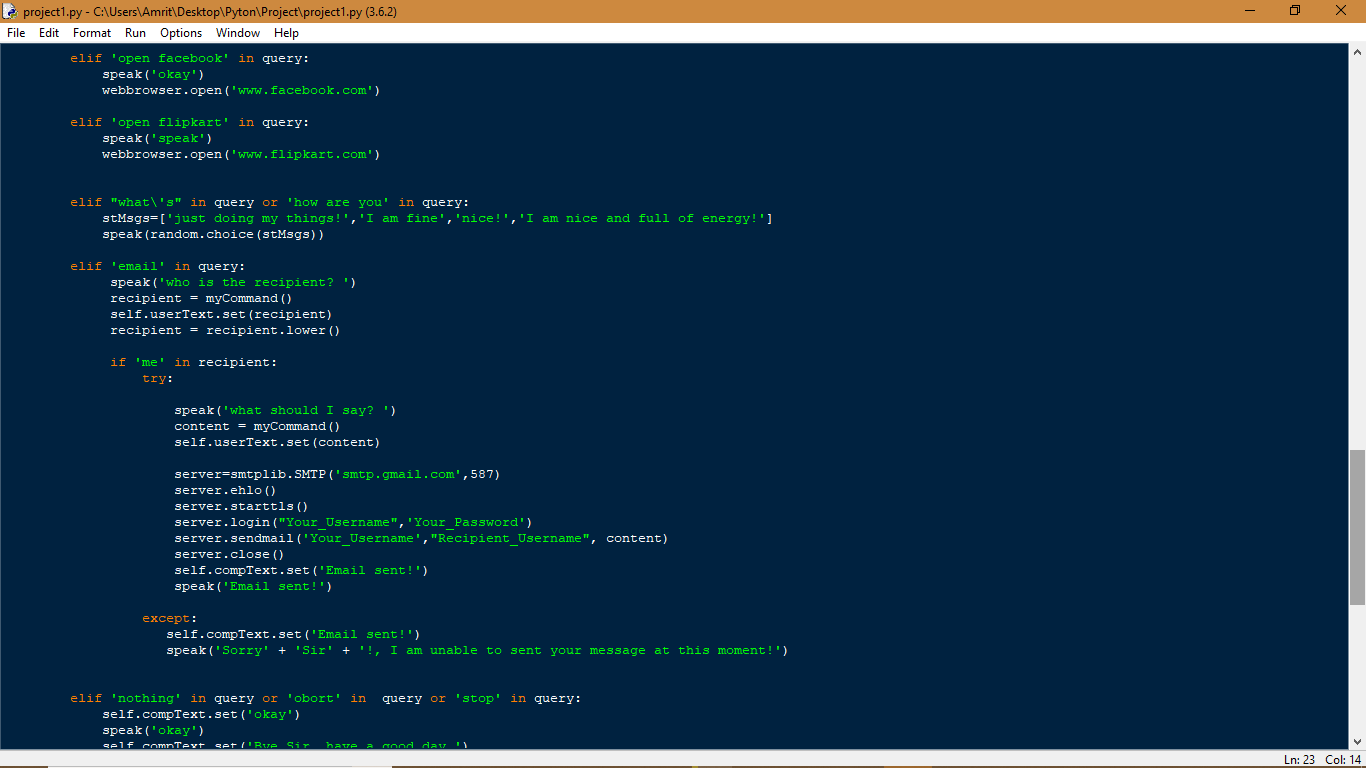
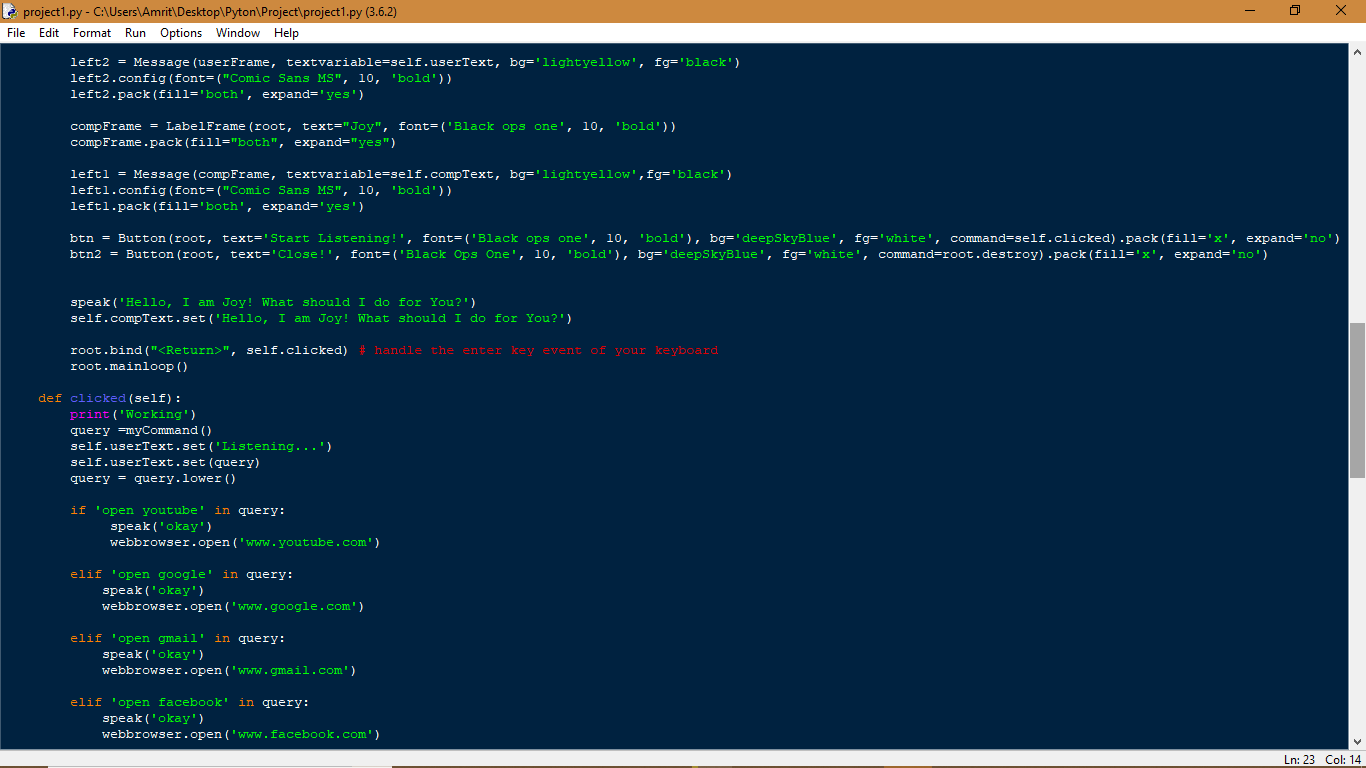
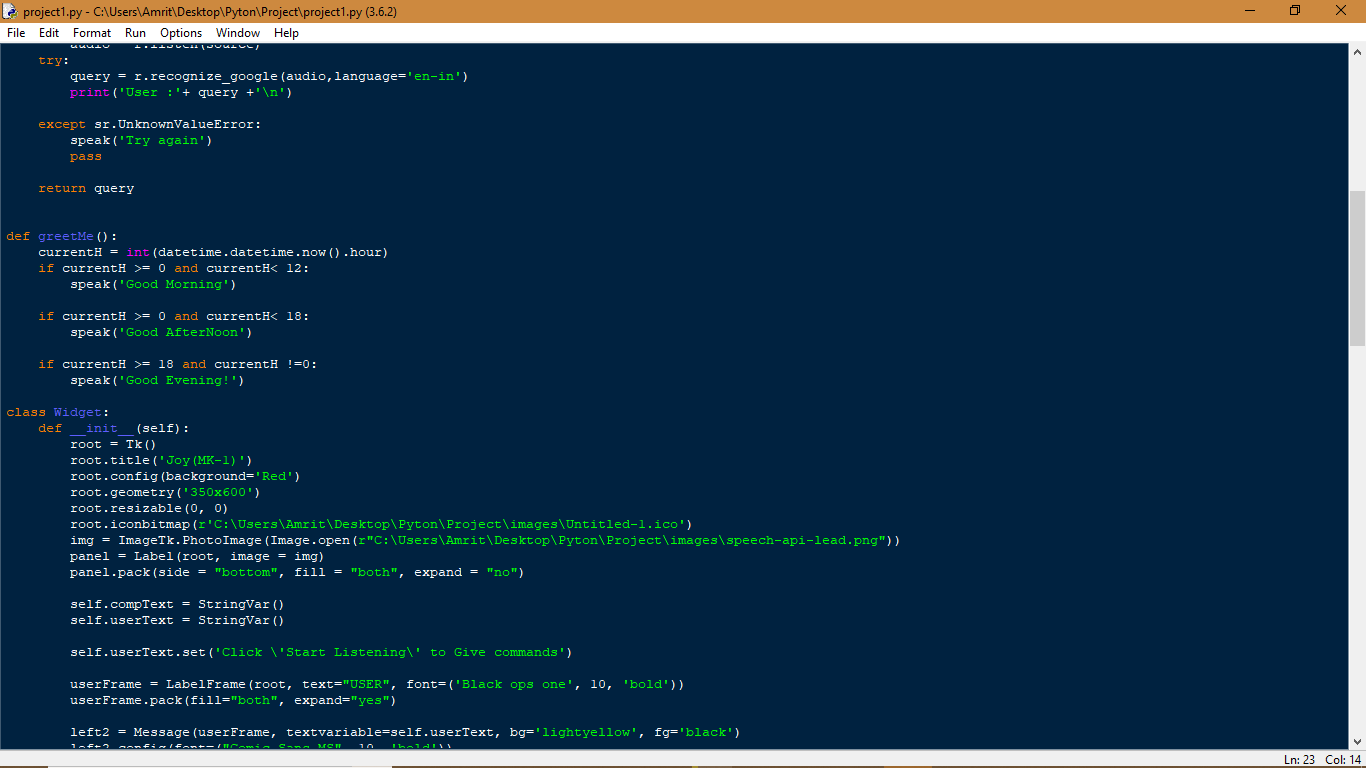
# Personal assistant (AI assistant)

A personal assistant, also called AI assistant or digital assistant, is an [application program](https://searchsoftwarequality.techtarget.com/definition/application) that understands [natural language](https://whatis.techtarget.com/definition/natural-language) voice commands and completes tasks for the user.  
  
Such tasks, historically performed by a personal assistant or secretary, include taking dictation, reading text or email messages aloud, looking up phone numbers, scheduling, placing phone calls and reminding the end user about appointments. Popular personal assistants currently include Amazon [Alexa](https://whatis.techtarget.com/definition/Alexa-Voice-Services-AVS), Apple's [Siri](https://searchmobilecomputing.techtarget.com/definition/Siri), [Google Now](https://whatis.techtarget.com/definition/Google-Now) and Microsoft's [Cortana](https://searchenterprisedesktop.techtarget.com/definition/Cortana) -- the digital assistant built into Windows Phone 8.1 and Windows 10.

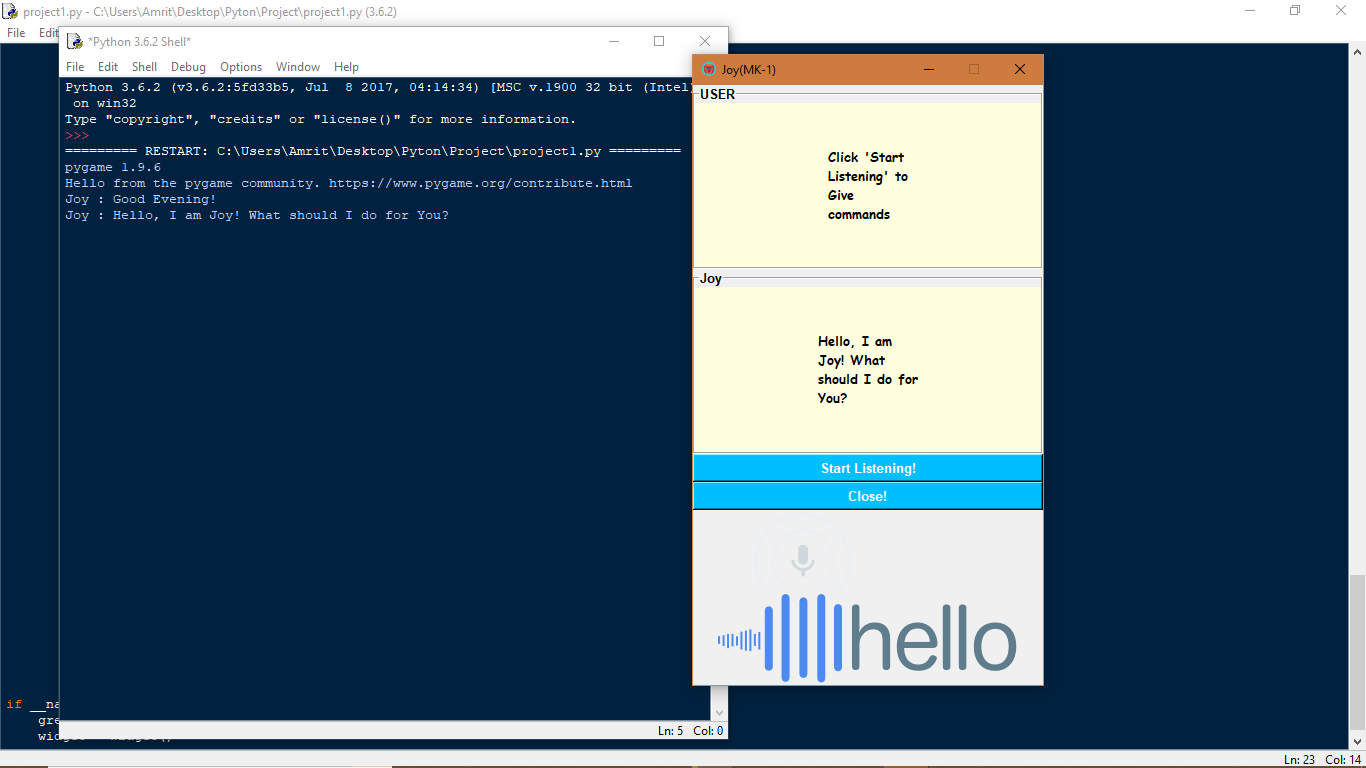
**Overview of the project:**

**Source Code :-**

****

****

**Output :-**

****

**MODULE DISCRIPTION**

**Libraries included**

**tkinter :-**

tkinter is the Python interface to the Tk GUI toolkit shipped with Python.

## **Tkinter Programming**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the *Tkinter* module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

## **Example**

#!/usr/bin/python

import Tkinter

top = Tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



**PIL**

The **Python** Imaging **Library** (**PIL**) is image manipulation **library in python**

**Installation :-**

**pip install Pillow**

>>> from PIL import Image

>>> im = Image.open("lena.ppm")

**pygame:**

Game creation in any programming language is very rewarding, and also makes for a great teaching tool. With game development, you often have quite a bit of logic, mathematics, physics, artificial intelligence, and other things, all of which come together for game creation. Not only this, but the topic is games, so it can be very fun.

Many times people like to visualize the programs they are creating, as it can help people to learn programming logic quickly. Games are fantastic for this, as your are specifically programming everything you see.

First, you're going to need **[PyGame](http://pygame.org/news.html" \t "_blank" \o "PyGame module for game creating in Python)**!

Installation may vary by operating system. Macs historically have had trouble with PyGame, so you may need to get an earlier version of Python and PyGame.

Nowadays, pip is probably the best method for installing PyGame, since it comes with the latest versions of Python 2 and Python 3.

## **Simple PyGame Application**

The following is a very simple app that is built using PyGame pipeline. Check it out:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import pygame    pygame.init()  screen = pygame.display.set\_mode((400, 300))  done = False    while not done:          for event in pygame.event.get():                  if event.type == pygame.QUIT:                          done = True            pygame.display.flip() |

Here is where you can make more sense out of the syntax:

**import pygame** – This is of course needed to access the PyGame framework.

**pygame.init()** – This initializes all the modules required for PyGame.

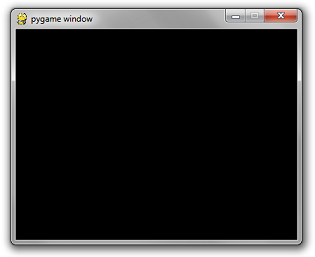
**pygame.display.set\_mode((width, height))** – This will launch a window of the desired size. The return value is a Surface object which is the object you will perform graphical operations on.

**pygame.event.get()** – This empties the event queue. If you do not call this, the windows messages will start to pile up and your game will become unresponsive in the opinion of the operating system.

**pygame.QUIT** – This is the event type that is fired when you click on the close button in the corner of the window.

**pygame.display.flip()** – PyGame is double-buffered so this swaps the buffers. All you need to know is that this call is required in order for any updates that you make to the game screen to become visible.

So what is the output like, when we execute the above code? It looks something like this:



**subprocess :-**

The subprocess module enables you to start new applications from your Python program

In the example below the full command would be “ls -l”

|  |
| --- |
| *#!/usr/bin/env python*    **import** subprocess  subprocess.call(["ls", "-l"]) |

**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.

**Installation :**

pip install pyttsx3

If you recieve errors such as No module named win32com.client, No module named win32, or No module named win32api, you will need to additionally install pypiwin32.

**Usage :**

import pyttsx3

engine = pyttsx3.init()

engine.say("I will speak this text")

engine.runAndWait()

**Web-browser**

To display web based documents to users by using python, there is a module called webbrowser. It provides high level interface to handle web documents.

To use this module, we need to import the following module.

import webbrowser

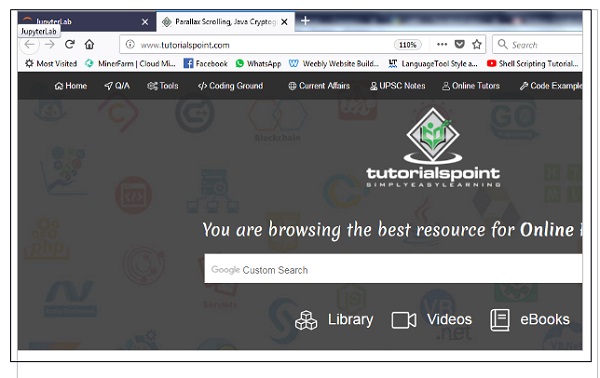
## Example Code

import webbrowser as browser

my\_browser = browser.get('windows-default')

my\_browser.open\_new('http://www.tutorialspoint.com')

## Output



**SMTP**

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail –

import smtplib

smtpObj = smtplib.SMTP( [host [, port [, local\_hostname]]] )

## **Example**

Here is a simple way to send one e-mail using Python script. Try it once −

#!/usr/bin/python

import smtplib

sender = 'from@fromdomain.com'

receivers = ['to@todomain.com']

message = """From: From Person <from@fromdomain.com>

To: To Person <to@todomain.com>

Subject: SMTP e-mail test

This is a test e-mail message.

"""

try:

smtpObj = smtplib.SMTP('localhost')

smtpObj.sendmail(sender, receivers, message)

print "Successfully sent email"

except SMTPException:

print "Error: unable to send email"

## **Random**

The method **random()** returns a random float r, such that 0 is less than or equal to r and r is less than 1.

## **Syntax**

Following is the syntax for **random()** method −

random ( )

**Note** − This function is not accessible directly, so we need to import random module and then we need to call this function using random static object

## **Example**

The following example shows the usage of random() method.

#!/usr/bin/python

import random

# First random number

print "random() : ", random.random()

# Second random number

print "random() : ", random.random()

When we run above program, it produces following result −

random() : 0.281954791393

random() : 0.309090465205

# **Speech Recognition**

Speech recognition is the process of converting spoken words to text. Python supports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API,  
Microsoft Bing Voice Recognition and IBM Speech to Text.

## **Installation**

A library that helps is named “SpeechRecognition”. You should install it with pyenv, pipenv or virtualenv. You can also install it system wide:

|  |
| --- |
| pip install SpeechRecognition |

**Wikipedia**

**Wikipedia** is a Python library that makes it easy to access and parse data from Wikipedia.

## Installation

To install Wikipedia, simply run:

$ pip install wikipedia

Date\_time

A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

Example

Import the datetime module and display the current date:

import datetime  
  
x = datetime.datetime.now()  
print(x)

**Wolframalpha**

Wolfram Alpha is a computational search engine that tends to evaluate what the user asks.

Installation

pip install wolframalpha  
pip install wikipedia  
pip install requests

* Create a python file and open it with any code editor of your choice
* Import the pre-installed packages

import wolframalpha  
import wikipedia  
import requests

**OS**

The OS module in Python provides a way of using operating system dependent

functionality.

The functions that the OS module provides allows you to interface with the

underlying operating system that Python is running on – be that Windows, Mac or

Linux.

# **Sys Module**

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. You will learn some of the important features of this module here.

## **sys.argv**

sys.argv returns a list of command line arguments passed to a Python script. The item at index 0 in this list is always the name of the script. The rest of the arguments are stored at the subsequent indices.

Here is a Python script (test.py) consuming two arguments from the command line.

Example: test.py

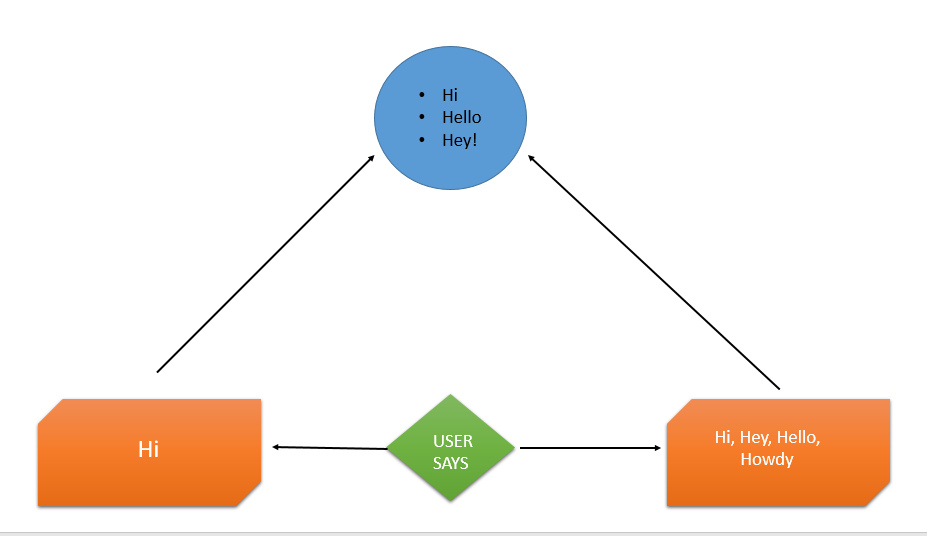
import sys

print("Hello {}. Welcome to {} tutorial".format(sys.argv[1], sys.argv[2]))

**Diagrams**

**UML Diagrams**

**Use Case Diagram:**  
A use case is a set of scenarios that describe an interaction between a user and a system.  A use case diagram displays the relationship among actors and use cases.  The two main components of a use case diagram are use cases and actors.



**Reference :-**

<https://www.google.com/>, <https://www.tutorialspoint.com/>,<https://www.geeksforgeeks.org/>

<https://www.wikipedia.org/>,<https://www.youtube.com/> “Er. Rajat Tyagi”

***Thank you***