Python Training

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by

Pankaj Panwar: (Roll No. 16UCSE4016)

Under the Supervision of

Mr. Virendra Soni

Department of Computer Science and Engineering
M.B.M. Engineering College
Faculty of Engineering & Architecture
Jai Narain Vyas University, Jodhpur

DECLARATION

I, *Pankaj Panwar* hereby declare that this seminar/project titled “Python Training” is a record of original work done by me under the supervision and guidance of Mr. Virendra Soni.

I, further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

Pankaj Panwar
VIIITH Semester, CSE
Roll No. - 16UCSE4010

ACKNOWLEDGEMENT

I take immense pleasure in thanking Dr. Shrikant Ojha, Dean, Faculty of Engineering & Architecture, Jodhpur for permitting me to carry out work of this project. I would like to thank Dr. N C BARWAR, HOD of Computer Science & Engineering, Jodhpur for support and facilities made available.

I wish to express our deep sense of gratitude to Mr. Virendra Soni for her able guidance and useful suggestions, which helped us in completing the project work, in time.

Finally, yet most importantly, I would like to express my heartfelt thanks to my beloved parents and family for their blessings, wishes and support for the successful completion of this project

Contents

S.no	Topic	Page No.
1.	HISTORY OF PYTHON	1
2.	Why you should use python	3
3.	Characteristics of Python	6
4.	Library Used	9
5.	Implementation	14
6.	Conclusion	21
7	References	22

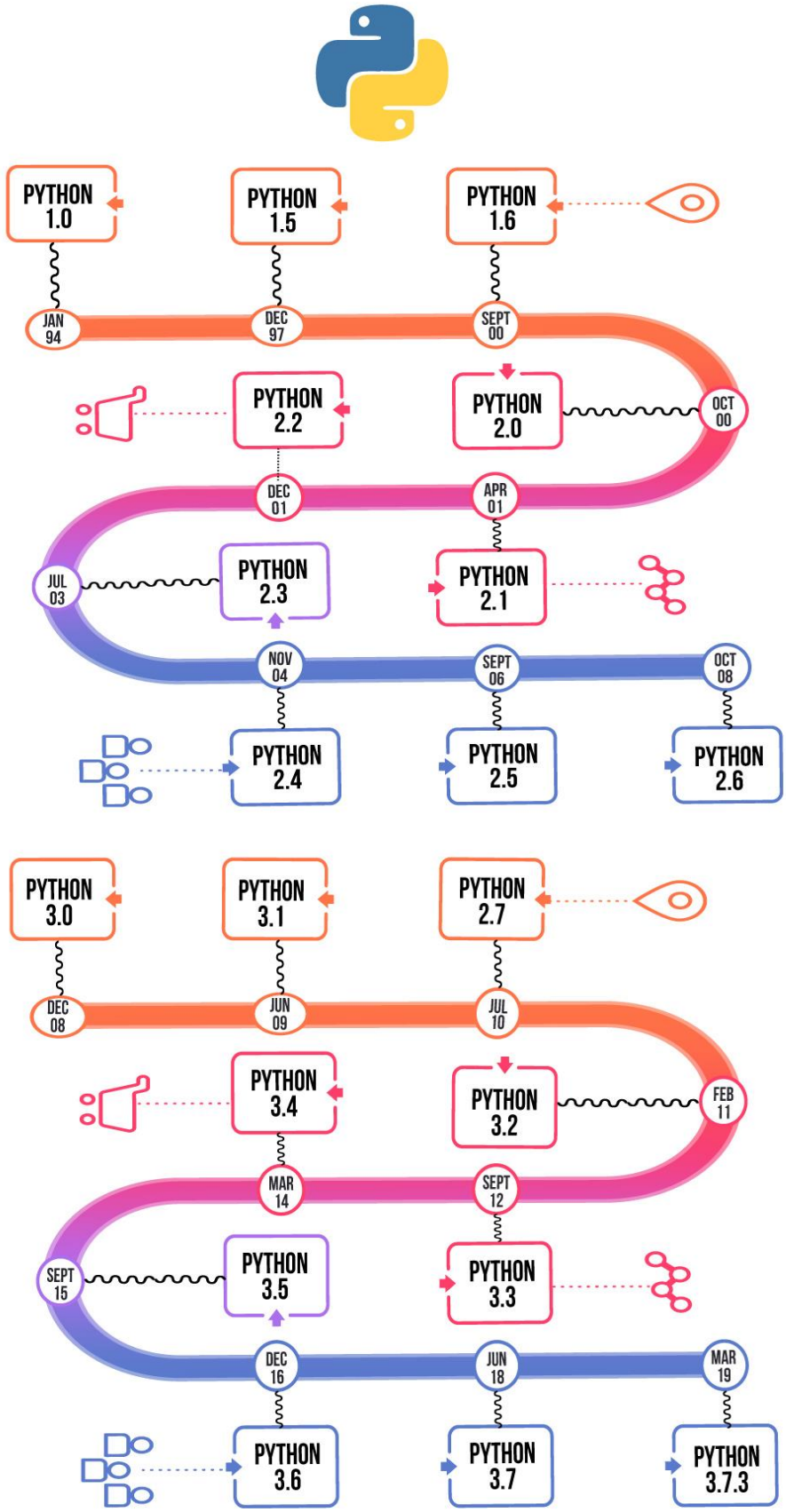
CHAPTER 1

HISTORY OF PYTHON

Python is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Let's dig deeper –

In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 by at Centrum Wiskunde & Informatica (CWI) which is situated in Netherland. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language which Python is said to have succeeded is ABC Programming Language, which had the interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did as really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws. The inspiration for the name came from BBC's TV Show – 'Monty Python's Flying Circus', as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the "Benevolent dictator for life" (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox. The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data type exception handling and functions.


PYTHON 1.0
PYTHON 1.5
PYTHON 1.6
JAN 94
DEC 97
SEPT 00
PYTHON 2.2
PYTHON 2.0
OCT 00
DEC 01
APR 01
JUL 03
PYTHON 2.3
PYTHON 2.1
NOV 04
SEPT 06
OCT 08
PYTHON 2.4
PYTHON 2.5
PYTHON 2.6
PYTHON 3.0
PYTHON 3.1
PYTHON 2.7
DEC 08
JUN 09
JUL 10
PYTHON 3.4
PYTHON 3.2
FEB 11
MAR 14
SEPT 12
SEPT 15
PYTHON 3.5
PYTHON 3.3
DEC 16
JUN 18
MAR 19
PYTHON 3.6
PYTHON 3.7
PYTHON 3.7.3

CHAPTER 2

WHY YOU SHOULD USE PYTHON

According to the latest TIOBE Programming Community Index, Python is one of the top 10 popular programming languages of 2017. Python is a general purpose and high-level programming language. You can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high-level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further make it easier for you to keep the code base readable and application maintainable. There are also a number of reasons why you should prefer Python to other programming languages.

Seven Reasons Why You Must Consider Writing Software Applications in Python

1) Readable and Maintainable Code

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

2**) Multiple Programming Paradigms**

Like other modern programming languages, Python also supports several programming paradigms. It supports object oriented and structured programming fully. Also, its language features support various concepts in functional and aspect-oriented programming. At the same time, Python also features a dynamic type system and automatic memory management. The programming paradigms and language features help you to use Python for developing large and complex software applications.

3**) Compatible with Major Platforms and Systems**

At present, Python is supporting many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.

4) Robust Standard Library

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code. For instance, while writing a web application in Python, you can use specific modules to implement web services, perform string operations, manage operating system interface or work with internet protocols. You can even gather information about various modules by browsing through the Python Standard Library documentation.

5**) Many Open Source Frameworks and Tools**

As an open source programming language, Python helps you to curtail software development cost significantly. You can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost. You even have option to choose from a wide range of open source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy. Likewise, you can accelerate desktop GUI application development using Python GUI frameworks and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.

6) **Simplify Complex Software Development**

Python is a general-purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to

facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many Python developers even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

7**) Adopt Test Driven Development**

You can use Python to create prototype of the software application rapidly. Also, you can build the software application directly from the prototype simply by refactoring the Python code. Python even makes it easier for you to perform coding and testing simultaneously by adopting test driven development (TDD) approach. You can easily write the required tests before writing code and use the tests to assess the application code continuously. The tests can also be used for checking if the application meets predefined requirements based on its source code.

However, Python, like other programming languages, has its own shortcomings. It lacks some of the built-in features provided by other modern programming language. Hence, you have to use Python libraries, modules, and frameworks to accelerate custom software development. Also, several studies have shown that Python is slower than several widely used programming languages including Java and C++. You have to speed up the Python application by making changes to the application code or using custom runtime. But you can always use Python to speed up software development and simplify software maintenance.

CHAPTER 3

CHARACTERISTICS OF PYTHON

Python Features

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

1) Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type print ("Hello World"). It will take only one line to execute, while Java or C takes multiple lines.

3) Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

8) Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, NumPy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C, C++ Java. It makes easy to debug the code.

11. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

12. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

CHAPTER 4

LIBRARY USED

Python DateTime

In Python, date and time are not a data type of its own, but a module named datetime can be imported to work with the date as well as time. Datetime module comes built into Python, so there is no need to install it externally.
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. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The datetime classes are categorize into 6 main classes –

- date – An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
- time – An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
- datetime – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
- timedelta – A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.
- tzinfo – It provides time zone information objects.
- timezone – A class that implements the tzinfo abstract base class as a fixed offset from the UTC.

PyAudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms.
PyAudio is inspired by:

To use PyAudio, first instantiate PyAudio using pyaudio.PyAudio(), which sets up the portaudio system.

To record or play audio, open a stream on the desired device with the desired audio parameters using pyaudio.PyAudio.open(). This sets up a pyaudio.Stream to play or record audio.

Play audio by writing audio data to the stream using pyaudio.Stream.write(), or read audio data from the stream using pyaudio.Stream.read().

Note that in “blocking mode”,

each pyaudio.Stream.write() or pyaudio.Stream.read() blocks until all the given/requested frames have been played/recorded. Alternatively, to generate audio data on the fly or immediately process recorded audio data, use the “callback mode” outlined below.

Use pyaudio.Stream.stop_stream() to pause playing/recording, and pyaudio.Stream.close() to terminate the stream. (4)

Finally, terminate the portaudio session using pyaudio.PyAudio.terminate()

Wikipedia

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

Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the MediaWiki API so you can focus on using Wikipedia data, not getting it.

Note: this library was designed for ease of use and simplicity, not for advanced use. If you plan on doing serious scraping or automated requests, please use Pywikipediabot (or one of the other more advanced Python MediaWiki API wrappers), which has a larger API, rate limiting, and other features so we can be considerate of the MediaWiki infrastructure.

Installation

To install Wikipedia, simply run:

```
$ pip install wikipedia
```

PyAutoGUI

PyAutoGUI lets your Python scripts control the mouse and keyboard to automate interactions with other applications. The API is designed to be as simple. PyAutoGUI works on Windows, macOS, and Linux, and runs on Python 2 and 3.

To install with pip, run: *pip install pyautogui*

The source is available on: https://github.com/asweigart/pyautogui

PyAutoGUI has several features:

- Moving the mouse and clicking or typing in the windows of other applications.
- Sending keystrokes to applications (for example, to fill out forms).
- Take screenshots, and given an image (for example, of a button or checkbox), find it on the screen.
- Locate an application's window, and move, resize, maximize, minimize, or close it (Windows-only, currently)
- Display message boxes for user interaction while your GUI automation script runs.

Pyjokes

One line jokes for programmers (jokes as a service)

Installation

Install the *pyjokes* module with pip.

See the documentation for installation instructions.

Usage

Once installed, simply call *pyjoke* from the command line or add it to your .bashrc file to see a joke every time you open a terminal.

Use the *-c* flag to get jokes from a specific category. Options:

```
-c neutral [default] (neutral geek jokes)
```

-c chuck (Chuck Norris geek jokes)

-c all (all jokes)

-c twister (Tongue-twister)

Win10toast

Python is a general-purpose language, can be used to develop both desktop and web applications. By using a package available in Python named win10toast , we can create desktop notifications. It is an easy way to get notified when some event occurs.
The package is available in Pypi and it is installed using pip.

```
pip install win10toast
```

About show_toast() function:
Syntax: show_toast(title='Notification', message='Here comes the message', icon_path=None, duration=5, threaded=False)
Parameters:
title: It contains notification title.
message: It contains notification message.
icon_path: It contains the path to .ico file.
duration":

It specifies the notification destruction active duration.

To create notifications we have to import the win10toast module. Then create an object to ToastNotifier class and by using the method show_toast we create a notification. It contains *header* or title of that notification, actual message, duration of that notification and icon for that notification. show_toast method is a instance of notification settings.

Speech Recognition

Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. This article aims to provide an introduction on how to make use of the SpeechRecognition library of Python. This is useful as it can be used on microcontrollers such as Raspberri Pis with the help of an external microphone.

Speech Input Using a Microphone and Translation of Speech to Text

1. Configure Microphone (For external microphones): It is advisable to specify the microphone during the program to avoid any glitches.
   Type lsusb in the terminal. A list of connected devices will show up. Make a note of this as it will be used in the program.
2. Set Chunk Size: This basically involved specifying how many bytes of data we want to read at once. Typically, this value is specified in powers of 2 such as 1024 or 2048
3. Set Sampling Rate: Sampling rate defines how often values are recorded for processing
4. Set Device ID to the selected microphone: In this step, we specify the device ID of the microphone that we wish to use in order to avoid ambiguity in case there are multiple microphones. This also helps debug, in the sense that, while running the program, we will know whether the specified microphone is being recognized. During the program, we specify a parameter device_id. The program will say that device_id could not be found if the microphone is not recognized.
5. Allow Adjusting for Ambient Noise: Since the surrounding noise varies, we must allow the program a second or too to adjust the energy threshold of recording so it is adjusted according to the external noise level.
6. Speech to text translation: This is done with the help of Google Speech Recognition. This requires an active internet connection to work. However, there are certain offline Recognition systems such as PocketSphinx, but have a very rigorous installation process that requires several dependencies. Google Speech Recognition is one of the easiest to use.

Chapter 5
Code

```
import pyttsx3
import datetime  #module
import speech_recognition as sr
import wikipedia
import webbrowser as wb
import os  #inbuilt
import pyautogui
import psutil  #pip install psutil
import pyjokes  # pip install pyjokes
import requests, json  #inbuilt
import win10toast
toaster = win10toast.ToastNotifier()


engine = pyttsx3.init()
engine.setProperty('rate', 190)
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
engine.setProperty('volume', 1)


#change voice
def voice_change(v):
    x = int(v)
    engine.setProperty('voice', voices[x].id)
    speak("done sir")


#speak function
def speak(audio):
    engine.say(audio)
    engine.runAndWait()


#time function
def time():
    Time = datetime.datetime.now().strftime("%H:%M:%S")
    speak("The current time is")
    speak(Time)


#date function
def date():
```

```
    year = int(datetime.datetime.now().year)
    month = int(datetime.datetime.now().month)
    date = int(datetime.datetime.now().day)
    speak("The current date is")
    speak(date)
    speak(month)
    speak(year)


def checktime(tt):
    hour = datetime.datetime.now().hour
    if ("morning" in tt):
        if (hour >= 6 and hour < 12):
            speak("Good morning sir")
        else:
            if (hour >= 12 and hour < 18):
                speak("it's Good afternoon sir")
            elif (hour >= 18 and hour < 24):
                speak("it's Good Evening sir")
            else:
                speak("it's Goodnight sir")
    elif ("afternoon" in tt):
        if (hour >= 12 and hour < 18):
            speak("it's Good afternoon sir")
        else:
            if (hour >= 6 and hour < 12):
                speak("Good morning sir")
            elif (hour >= 18 and hour < 24):
                speak("it's Good Evening sir")
            else:
                speak("it's Goodnight sir")
    else:
        speak("it's night sir!")


#welcome function
def wishme():
    speak("Welcome Back")
    hour = datetime.datetime.now().hour
    if (hour >= 6 and hour < 12):
        speak("Good Morning sir!")
    elif (hour >= 12 and hour < 18):
        speak("Good afternoon sir")
    elif (hour >= 18 and hour < 24):
        speak("Good Evening sir")
    else:
        speak("Goodnight sir")
    toaster.show_toast("  Voice Assistant Started  ", duration=4)
    speak("At your service sir, Please tell me how can i help you?")
```

```
def wishme_end():
    speak("signing off")
    hour = datetime.datetime.now().hour
    if (hour >= 6 and hour < 12):
        speak("Good Morning")
    elif (hour >= 12 and hour < 18):
        speak("Good afternoon")
    elif (hour >= 18 and hour < 24):
        speak("Good Evening")
    else:
        speak("Goodnight.. Sweet dreams")
    toaster.show_toast("  terminating voice assistant  ", duration=10)
    quit()


#command by user function
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listing...")
        r.pause_threshold = 0.5
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        #speak(query)
        print(query)
    except Exception as e:
        print(e)
        speak("Say that again please...")

        return "None"

    return query

#screenshot function
def screenshot():
    img = pyautogui.screenshot()
    img.save(
        "C:\\Users\\Pankaj Panwar\\Desktop\\ss.png"
    )


#battery and cpu usage
def cpu():
    usage = str(psutil.cpu_percent())
```

```
    speak('CPU usage is at ' + usage)
    print('CPU usage is at ' + usage)
    battery = psutil.sensors_battery()
    speak("Battery is at")
    speak(battery.percent)
    print("battery is at:" + str(battery.percent))


#joke function
def jokes():
    j = pyjokes.get_joke()
    print(j)
    speak(j)



def personal():
    speak(
        " hi,I am an AI assistent, I am developed by Pankaj Panwar for python training"
    )
    speak("Now i hope you know me")


if __name__ == "__main__":
    wishme()
    while (True):
        query = takeCommand().lower()

        #time

        if ('time' in query):
            time()

#date

        elif ('date' in query):
            date()

#personal info
        elif ("tell me about yourself" in query):
            personal()
        elif ("about you" in query):
            personal()
        elif ("who are you" in query):
            personal()
        elif ("yourself" in query):
            personal()

        elif ("developer" in query or "tell me about your developer" in query
```

```
            or "father" in query or "who develop you" in query
            or "developer" in query):
            speak("here is the details: " + res.read())

#searching on wikipedia

        elif ('wikipedia' in query or 'what' in query or 'who' in query
            or 'when' in query or 'where' in query):
            speak("searching...")
            query = query.replace("wikipedia", "")
            query = query.replace("search", "")
            query = query.replace("what", "")
            query = query.replace("when", "")
            query = query.replace("where", "")
            query = query.replace("who", "")
            query = query.replace("is", "")
            result = wikipedia.summary(query, sentences=2)
            print(query)
            print(result)
            speak(result)


#sysytem logout/ shut down etc

        elif ("logout" in query):
            os.system("shutdown -1")
        elif ("restart" in query):
            os.system("shutdown /r /t 1")
        elif ("shut down" in query):
            os.system("shutdown /r /t 1")

#play songs

        elif ("play songs" in query):
            speak("Playing...")
            songs_dir = "C:\\Music"
            songs = os.listdir(songs_dir)
            os.startfile(os.path.join(songs_dir, songs[1]))
            quit()

#reminder function

        elif ("create a reminder list" in query or "reminder" in query):
            speak("What is the reminder?")
            data = takeCommand()
            speak("You said to remember that" + data)
            reminder_file = open("data.txt", 'a')
            reminder_file.write('\n')
            reminder_file.write(data)
```

```
            reminder_file.close()

#reading reminder list

        elif ("do you know anything" in query or "remember" in query):
            reminder_file = open("data.txt", 'r')
            speak("You said me to remember that: " + reminder_file.read())

#screenshot
        elif ("screenshot" in query):
            screenshot()
            speak("Done!")

#cpu and battery usage
        elif ("cpu and battery" in query or "battery" in query
              or "cpu" in query):
            cpu()

#jokes
        elif ("tell me a joke" in query or "joke" in query):
            jokes()


#Voice assistant features
        elif ("tell me your powers" in query or "help" in query
              or "features" in query):
            features = ''' i can help to do lot many things like..
            i can tell you the current time and date,
            i can tell you battery and cpu usage,
            i can create the reminder list,
            i can take screenshots,
            i can shut down or logout or hibernate your system,
            i can tell you non funny jokes,
            i can open any website,
            i can search the thing on wikipedia,
            i can change my voice from male to female and vice-versa
            tell me what can i do for you??
            '''
            print(features)
            speak(features)

        elif ("hii" in query or "hello" in query or "goodmorning" in query
              or "goodafternoon" in query or "goodnight" in query
              or "morning" in query or "noon" in query or "night" in query):
            query = query.replace("hey", "")
            query = query.replace("hi", "")
            query = query.replace("hello", "")
            if ("morning" in query or "night" in query or "goodnight" in query
                    or "afternoon" in query or "noon" in query):
```

```
            checktime(query)
        else:
            speak("what can i do for you")

#changing voice
    elif ("voice" in query):
        speak("for female say female and, for male say male")
        q = takeCommand()
        if ("female" in q):
            voice_change(1)
        elif ("male" in q):
            voice_change(0)
    elif ("male" in query or "female" in query):
        if ("female" in query):
            voice_change(1)
        elif ("male" in query):
            voice_change(0)

#exit function to terminate assitant

    elif ('i am done' in query or 'bye bye ' in query
          or 'go offline ' in query or 'bye' in query
          or 'nothing' in query):
        wishme_end()
```

Chapter 6
Conclusion

Lastly, I conclude that I go through the Basics of Python programming language and successfully implementing a personal voice assistant which take voice commands by using some python libraries and obtained corresponding output which is shown in the above sections with their respective python code.

Further improvements possible:

1. More features can be added
2. Support for more than one language
3. Better quality of speech recognition
4. Increase vocabulary size.
5. More number of commands
6. More interactive

References

1. https://people.csail.mit.edu/hubert/pyaudio/docs/
2. https://www.programiz.com/python-programming/datetime
3. https://docs.python.org/3/library/datetime.html
4. https://realpython.com/python-speech-recognition/
5. https://www.geeksforgeeks.org/speech-recognition-in-python-using-google-speech-api/
6. https://pypi.org/project/pyjokes/
7. https://pyautogui.readthedocs.io/en/latest/