**Chapter 01: Introduction**

**Chapter 01**

**Introduction**

* 1. **Introduction to python**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its use of significant indentation. Its [language](https://en.wikipedia.org/wiki/Language_construct) [constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects. Python is dynamically-typed and [garbage-](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) [collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [structured](https://en.wikipedia.org/wiki/Structured_programming) (Particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), object-oriented and [functional-programming](https://en.wikipedia.org/wiki/Functional_programming). It is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a [cycle detecting](https://en.wikipedia.org/wiki/Cycle_detection) garbage collection system.

Python consistently ranks as one of the most popular programming languages.

* 1. Introduction to client-server programming using python:

In the project the main idea is of the client-server /socket programming. Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server. They are the real backbones behind web browsing. In simpler terms, there is a server and a client. Socket programming is started by importing the socket library and making a simple socket.

**import sockets = socket.Socket(socket.AF\_INET, socket.SOCK\_STREAM)**

* 1. **Introduction to “Tkinter” Module**

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. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. There are currently 15 types of widgets in Tkinter. Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications.

* 1. Introduction to other python modules used in the code:
     1. Socket module:

This module provides access to the BSD socket interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms. It is used for low-level networking interfaces. Depending on the system and the build options, various socket families are supported by this module. The socket module provides various objects, constants, functions and related exceptions for building full-fledged network applications including client and server programs.

* + 1. Thread module:

Thread module is used for creating, controlling and managing threads in python. In this tutorial, we will discuss about various functions and object types defined by the thread module.

* 1. Scope and importance of work:

Communication has become an integral part in today’s world. We can't rely on the same methods. Today many applications are being developed to improve communication. Socket programming is applicable in this environment.

We can chat with the desired individuals more securely and there is no threat of hacking as the system with IP address can only view the messages. Not only the two end-users can communicate but we can use this program to connect server, desired URL, dns system. It doesn't require much network bandwidth and is cost effective.

**2**

Chapter 02: Implementation

* 1. PYTHON CODE:

**CLIENT SIDE:**

from tkinter import \* from socket import \* import \_thread

def initialize\_client():

s = socket (AF\_INET, SOCK\_STREAM)

host = '172.16.23.223'

port = 80 s.connect((host, port))

return s

def update\_chat(msg, state): global chatlog

chatlog.config(state=NORMAL) if state==0:

chatlog.insert(END, 'YOU: ' + msg)

else:

chatlog.insert(END, 'OTHER: ' + msg) chatlog.config(state=DISABLED)

chatlog.yview(END) def send():

global textbox

msg = textbox.get("0.0", END) update\_chat(msg, 0)

s.send(msg.encode('ascii')) textbox.delete("0.0", END)

def receive(): while 1:

try:

data = s.recv(1024) msg = data.decode('ascii')

if msg != "": update\_chat(msg, 1)

except:

pass

def press(event): send()

def GUI(): global chatlog global textbox

gui = Tk () gui.title("Client Chat") gui.geometry("380×430")

chatlog = Text(gui, bg='white') chatlog.config(state=DISABLED)

sendbutton = Button(gui, bg='orange', fg='red', text='SEND', command=send) textbox = Text(gui, bg='white')

chatlog. place (x=6, y=6, height=386, width=370) textbox. place(x=6, y=401, height=20, width=265) sendbutton.place(x=300, y=401, height=20, width=50)

textbox.bind("<Key Release-Return>", press)

\_thread. start\_new\_thread(receive, ()) gui.mainloop()

if name == ' main ': chatlog = textbox = None s = initialize\_client()

GUI ()

**SERVER SIDE:**

from tkinter import \* from socket import \* import \_thread

def initialize server ():

s = socket (AF\_INET, SOCK\_STREAM)

host = '172.16.37.128' port = 80

s.bind((host, port))

s.listen(1)

conn, addr = s.accept()

return conn

def update\_chat(msg, state): global chatlog

chatlog.config(state=NORMAL) if state==0:

chatlog. insert (END, 'YOU: ' + msg) else:

chatlog.insert(END, 'OTHER: ' + msg) chatlog.config(state=DISABLED)

chatlog.yview(END)

**6**

def send (): global textbox

msg = textbox.get("0.0", END) update\_chat(msg, 0)

conn.send(msg.encode('ascii')) textbox. delete("0.0", END)

def receive (): while 1:

try:

data = conn. recv(1024) msg = data. decode('ascii')

if msg != "": update\_chat (msg, 1)

except:

pass

def press(event):

send ()

def GUI (): global chatlog global textbox

gui = Tk () gui. title("Server Chat")

gui.geometry("380×430")

**7**

chatlog = Text(gui, bg='white') chatlog.config(state=DISABLED)

sendbutton = Button(gui, bg='orange', fg='red', text='SEND', command=send)

textbox = Text(gui, bg='white')

chatlog.place(x=6, y=6, height=386, width=370)

textbox.place(x=6, y=401, height=20, width=265)

sendbutton. place(x=300, y=401, height=20, width=50)

textbox.bind("<KeyRelease-Return>", press)

\_thread. start\_new\_thread(receive, ())

gui.mainloop()

if name == ' main ': chatlog = textbox = None conn = initialize\_server ()

GUI()

**8**

* 1. CODE EXPLANATION

Client side:

The following application involves executing programs at two different end system. The following modules has to be installed using the “pip - install” command.

* Tkinter
* Socket
* Thread

Step 1:

Initialize the server connection by writing a function def initialize\_client()

Initialize the socket using the command S = socket (AF\_inet, sock\_Stream)

Configure the necessary details of server like the Ip address of the host like “192.024.098” Know the details of necessary port numbers

Connect to the server using the command by entering the valid port number and host Ip address

* 1. onnect ((host, port))

Update the chat box after every successful message is sent. Update the message window and display the latest messages

Step 2:

Write a function to send message using def send()

Using the command “msg = textbox.get (“0.0”, END) “you will be able to get message Update the chat box

Send message using “s.send” and encode the message to ascii code

**9**

Step 3:

Write a function to receive message Define a function receive

def receive()

The data is in order of 1024 bytes and the ascii code is decoded using conditional statements

Step 4:

Write a function “def gui() “to define the GUI function Initialize the tkinter object

Gui = tk()

Set the tittle using gui. tittle (client. chat) Set the size using geometry

AS the user desires set the color of chat box buttons and their respective axes. Bind the text box to use ‘ENTER’ key

Create a thread to capture messages continuously Keep window in a loop using a function

**10**

SERVER SIDE:

Generally, the same steps followed in client side is followed here with minute changes The steps are summarized here:

* + - Import the modules mentioned as in client program.
    - Initialize the server connection, server, update the messages chat log as done in the client side
    - Define a function to send message
    - Define a function to receive message
    - Define a GUI function to create a GUI application
    - Initialize the tkinter objects
    - Define textboxes, button to write a message send a message to use ENTER key
    - Create thread to capture messages
    - Using Gui. Loop ()
    - Keep the window in the loop

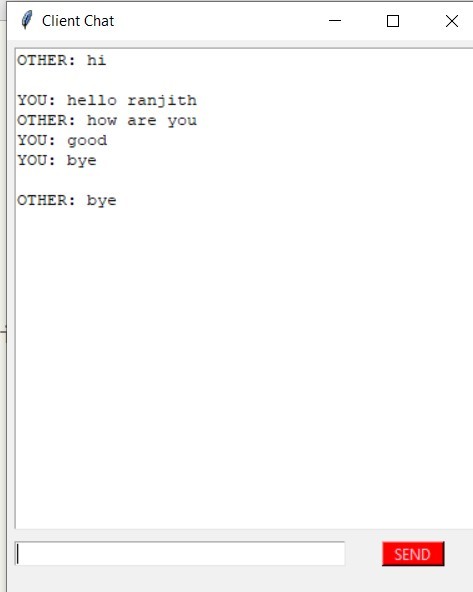
Chapter 03: Results and snapshots

**AT SERVER:**



**12**

AT CLIENT:



**Chapter: 04**

**CONCLUSION**

By completing the mini project, we could get to know the usage of many modules. Especially modules like tkinter socket and thread. The magic that the tkinter module does is enchanting. From the chosen topic we could learn many interesting facts and gained knowledge on new concepts. We learnt how we can connect two different end users under a same network to communicate. We learnt the GUI interfacing.

In the following topic communication is inevitable today. But data security is more necessary. May be not for common transactions but many has to be dealt with secrecy. We got an insight of how we can communicate using modern software and technologies. Today we all use social media but we got an insight of how actually we can connect. I would like to thank our python Professor Mr. Saravana for providing an opportunity to venture a step forward in our career.

**Chapter: 05**

**REFERENCES**

|  |  |
| --- | --- |
| **1** | https://github.com/ashwintecguy/Python-Projects |
| **2** | www.geeksforgeeks.org/client-server/Python |
| **3** | www.pythonic.com/libraries |