CS3530

COMPUTER NETWORKS

SOCKET PROGRAMMING ASSIGNMENT

Team Members **:**

Dheekhsitha Bheemnath	- *CS18BTECH11006*
Darapaneni Krishna Pawan	- *CS18BTECH11008*
Karuturi Havya Sree	- *CS18BTECH11022*
Akash Tadwai	- *ES18BTECH11019*
Vinta Reethu	- *ES18BTECH11028*
Krishnam Dhanush	- *ME18BTECH11022*

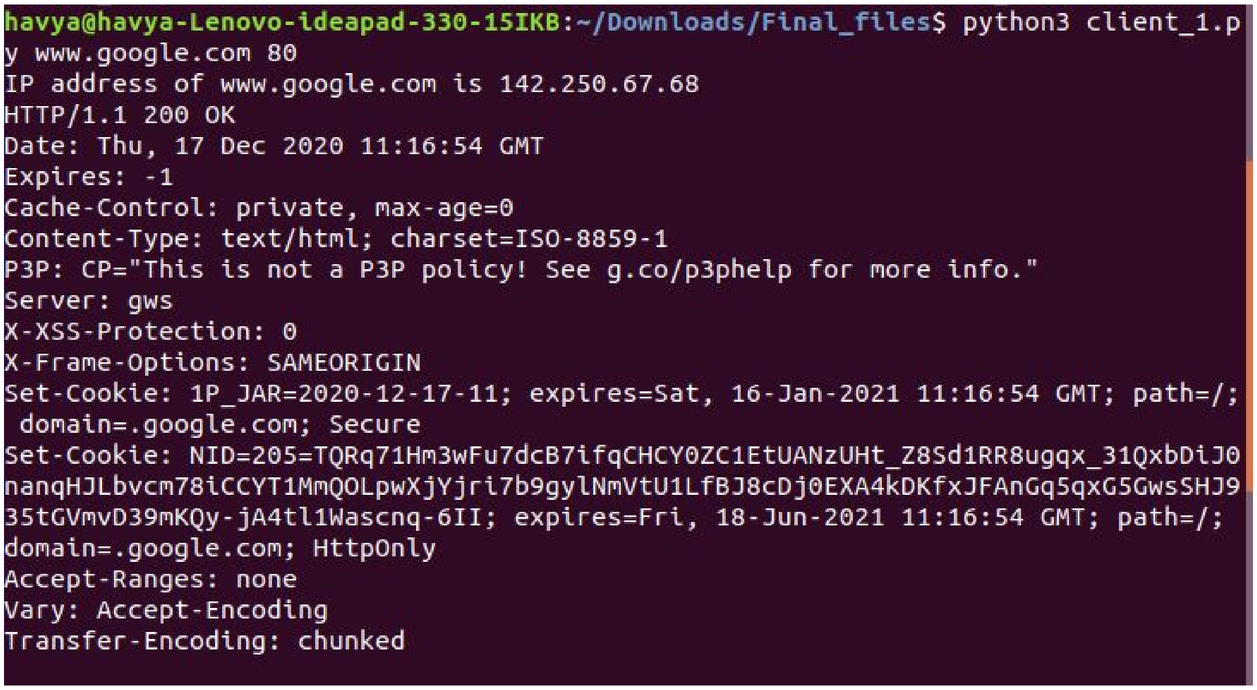
Question #1:

Integrate getaddrinfo() as part of your client software so that the hostname of the server can be given as the command line option.

Command: $ **python3 client_1.py <hostname> <port number>**

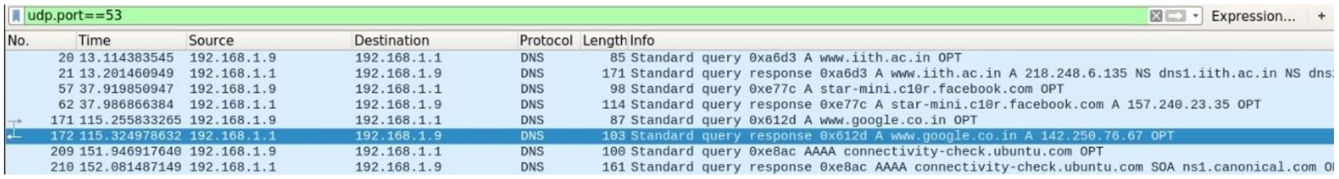
The client gets the ip address using getaddrinfo() and then sends a GET request to that corresponding IP address to verify the IP address.

1) **the screenshot of successful execution of your client software with hostname**



```
havya@havya-Lenovo-ideapad-330-15IKB:~/Downloads/Final_files$ python3 client_1.p
y www.google.com 80
IP address of www.google.com is 142.250.67.68
HTTP/1.1 200 OK
Date: Thu, 17 Dec 2020 11:16:54 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2020-12-17-11; expires=Sat, 16-Jan-2021 11:16:54 GMT; path=/;
 domain=.google.com; Secure
Set-Cookie: NID=205=TQRq71Hm3wFu7dcB7ifqCHCY0ZC1EtUANzUHt_Z8Sd1RR8ugqx_31QxbDiJ0
nanqHJLbvcm78iCCYT1MmQOLpwXjYjri7b9gylNmVtU1LfBJ8cDj0EXA4kDKfxJFAnGq5qxG5GwsSHJ9
35tGVmvD39mKQy-jA4tl1Wascnq-6II; expires=Fri, 18-Jun-2021 11:16:54 GMT; path=/;
domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked
```

2) **the screenshot of wireshark/tcpdump to prove that your client software sends a DNS query and receives a response.**



```
udp.port==53
No.     Time            Source          Destination     Protocol Length Info
     20 13.114383545    192.168.1.9     192.168.1.1     DNS       85 Standard query 0xa6d3 A www.iith.ac.in OPT
     21 13.201460949    192.168.1.1     192.168.1.9     DNS      171 Standard query response 0xa6d3 A www.iith.ac.in A 218.248.6.135 NS dns1.iith.ac.in NS dns
     57 37.919850947    192.168.1.9     192.168.1.1     DNS       98 Standard query 0xe77c A star-mini.c10r.facebook.com OPT
     62 37.986866384    192.168.1.1     192.168.1.9     DNS      114 Standard query response 0xe77c A star-mini.c10r.facebook.com A 157.240.23.35 OPT
    171 115.255833265   192.168.1.9     192.168.1.1     DNS       87 Standard query 0x612d A www.google.co.in OPT
    172 115.324978632   192.168.1.1     192.168.1.9     DNS      103 Standard query response 0x612d A www.google.co.in A 142.250.76.67 OPT
    209 151.946917640   192.168.1.9     192.168.1.1     DNS      100 Standard query 0xe8ac AAAA connectivity-check.ubuntu.com OPT
    210 152.081487149   192.168.1.1     192.168.1.9     DNS      161 Standard query response 0xe8ac AAAA connectivity-check.ubuntu.com SOA ns1.canonical.com O
```

References :
https://docs.python.org/3/library/socket.html
https://gitlab.com/wireshark/wireshark/-/wikis/CaptureFilters

---

Question #2:

Add at least two features to Echo Client /Server, and demonstrate them.

Feature 1 : Execute shell commands on client side from server.
Feature 2 : Timestamp of message exchanges
Feature 3 : Chat box

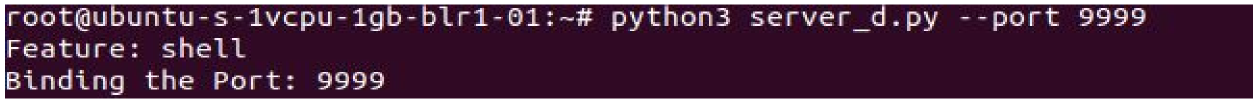
<u>Server command:</u>
$ python3 server_2.py --port <port number>

Creates a socket on the server and you are prompted to enter a feature which can be of two types:

1. **shell :** The server gets access to the remote client's shell
   The server also prints the timestamp for every message sent to the client.

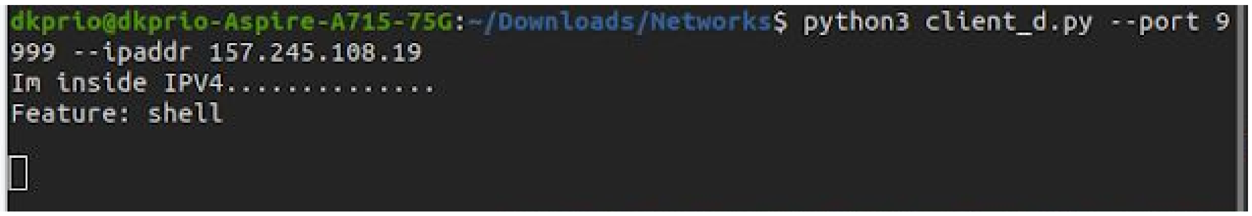
<u>Clients command:</u>
$ python3 client_2.py --port <port number> --ipaddr <ip address of server>

Server:



```
root@ubuntu-s-1vcpu-1gb-blr1-01:~# python3 server_d.py --port 9999
Feature: shell
Binding the Port: 9999
```

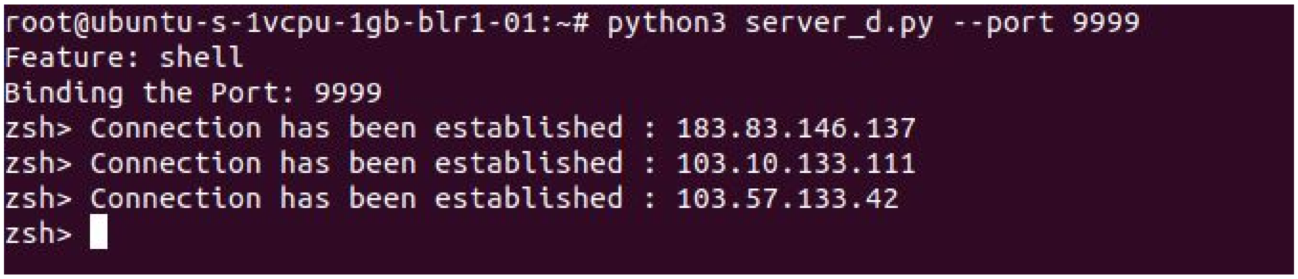
Client 0:



```
dkprio@dkprio-Aspire-A715-75G:~/Downloads/Networks$ python3 client_d.py --port 9
999 --ipaddr 157.245.108.19
Im inside IPV4..............
Feature: shell

```

- A message appears saying that client-server connection is established when a connection request sent by the client is accepted by the server.

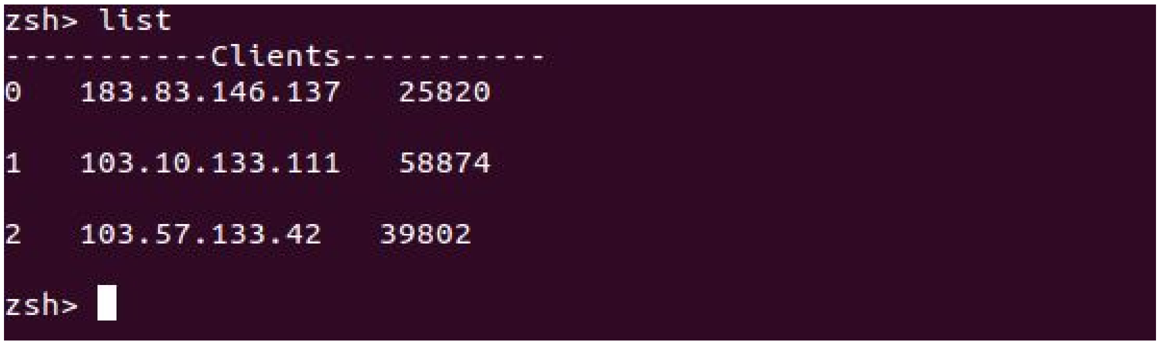
Server:



```
root@ubuntu-s-1vcpu-1gb-blr1-01:~# python3 server_d.py --port 9999
Feature: shell
Binding the Port: 9999
zsh> Connection has been established : 183.83.146.137
zsh> Connection has been established : 103.10.133.111
zsh> Connection has been established : 103.57.133.42
zsh> 
```

- If we type "**list**", all the connected client ips are listed

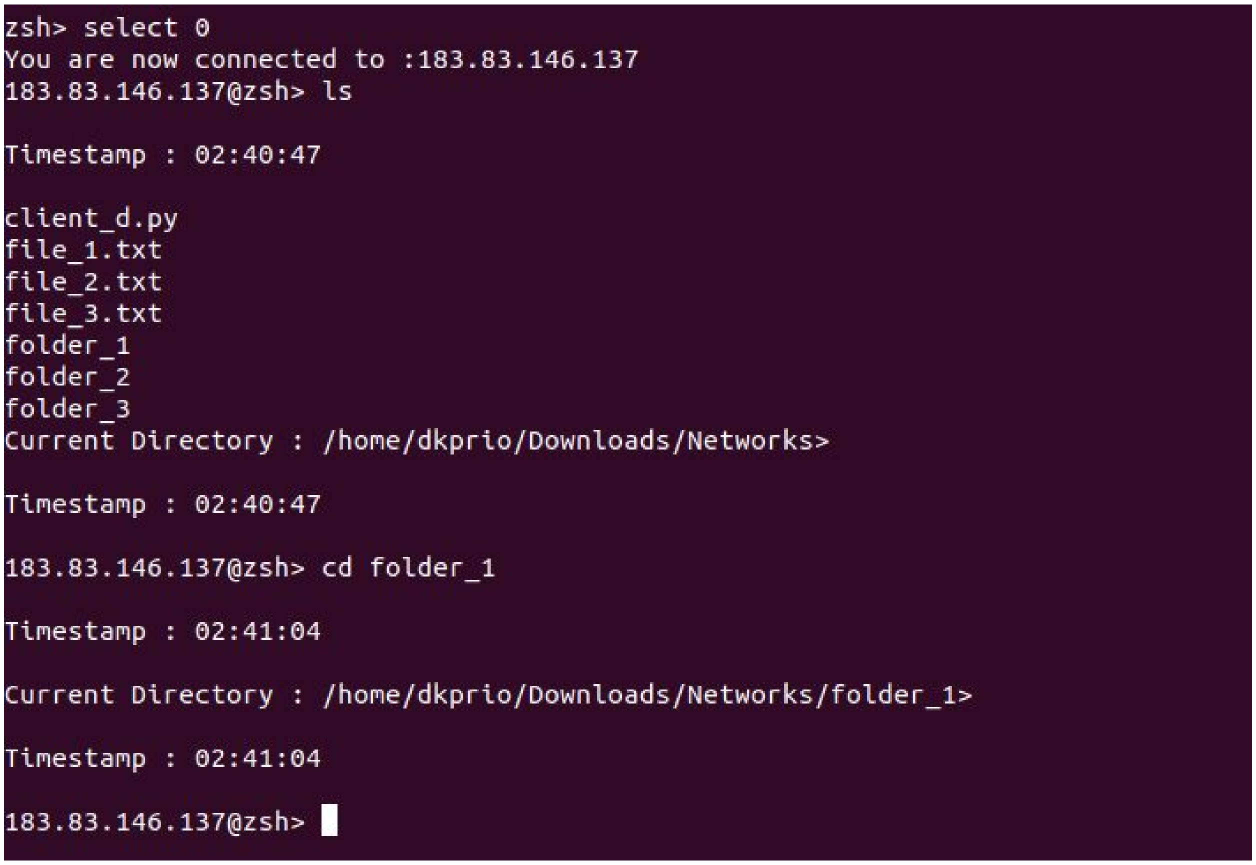
Server:



```
zsh> list
-----------Clients-----------
0   183.83.146.137   25820

1   103.10.133.111   58874

2   103.57.133.42   39802

zsh> 
```

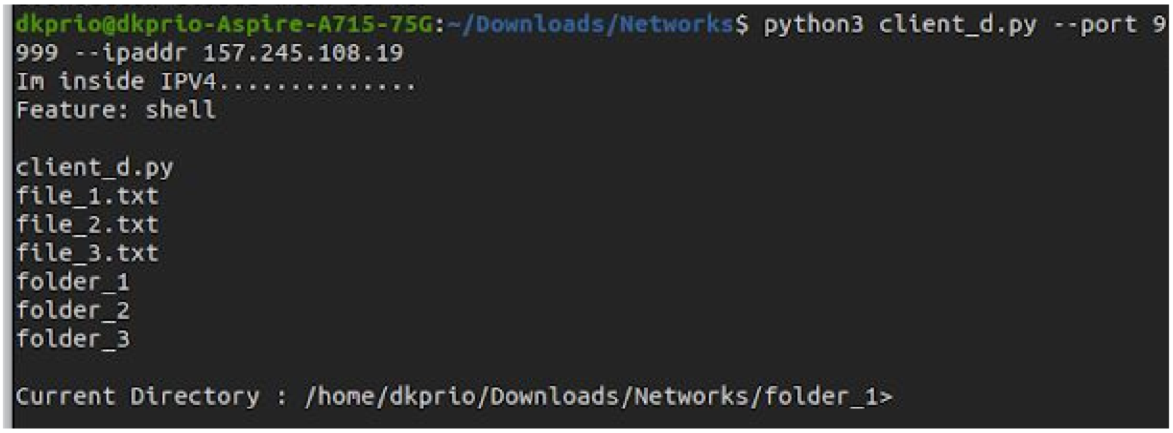
- If we type "**select <serial number>**", the server is connected to the client we selected and the server can now access the shell of the selected client and execute commands like cd, ls, mkdir, pwd, etc... remotely on its shell.

Server:



```
zsh> select 0
You are now connected to :183.83.146.137
183.83.146.137@zsh> ls

Timestamp : 02:40:47

client_d.py
file_1.txt
file_2.txt
file_3.txt
folder_1
folder_2
folder_3
Current Directory : /home/dkprio/Downloads/Networks>

Timestamp : 02:40:47

183.83.146.137@zsh> cd folder_1

Timestamp : 02:41:04

Current Directory : /home/dkprio/Downloads/Networks/folder_1>

Timestamp : 02:41:04

183.83.146.137@zsh>
```

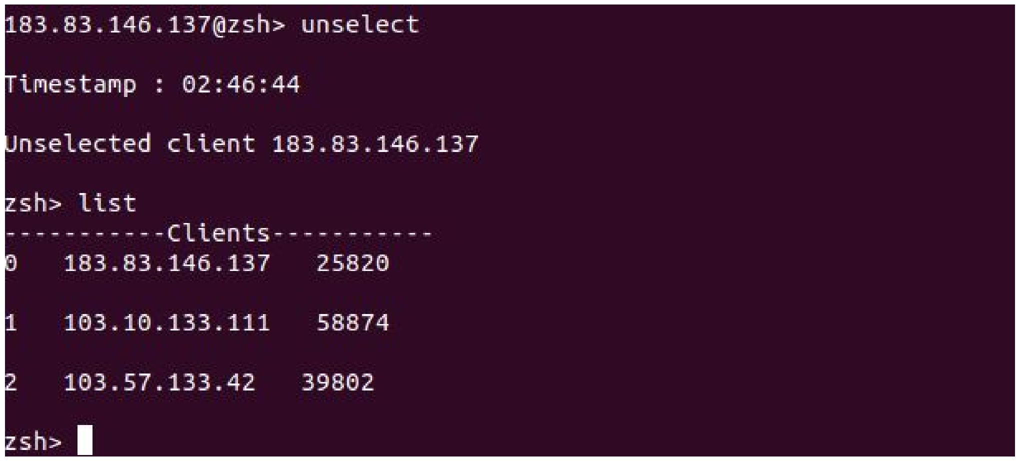
Client 0:



```
dkprio@dkprio-Aspire-A715-75G:~/Downloads/Networks$ python3 client_d.py --port 9
999 --ipaddr 157.245.108.19
Im inside IPV4.............
Feature: shell

client_d.py
file_1.txt
file_2.txt
file_3.txt
folder_1
folder_2
folder_3

Current Directory : /home/dkprio/Downloads/Networks/folder_1>
```

- If we type “**unselect**” then the clients shell will be unselected and the server will be ready to select other clients again.

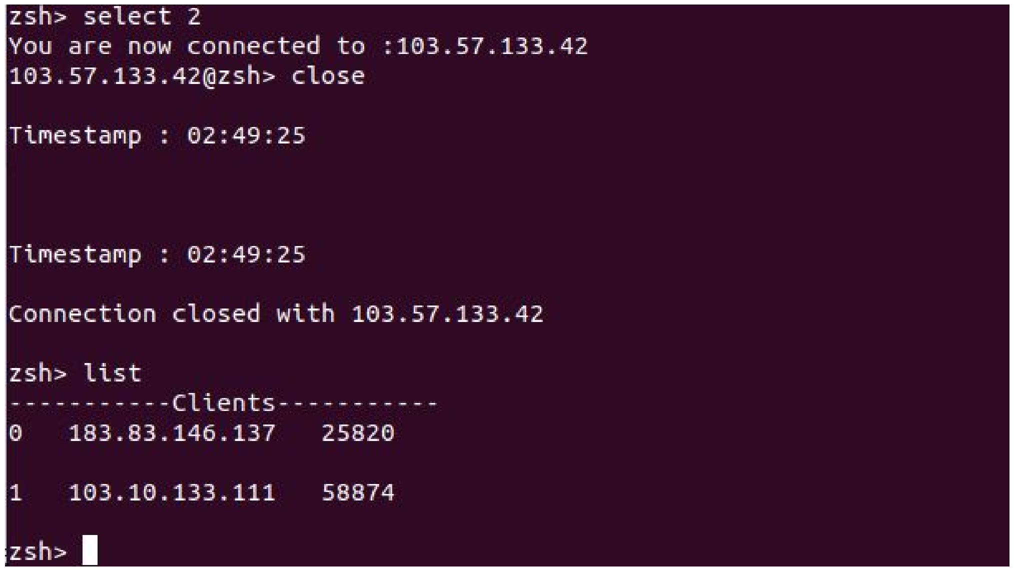
Server:



```
183.83.146.137@zsh> unselect

Timestamp : 02:46:44

Unselected client 183.83.146.137

zsh> list
-----------Clients-----------
0   183.83.146.137   25820

1   103.10.133.111   58874

2   103.57.133.42   39802

zsh> 
```

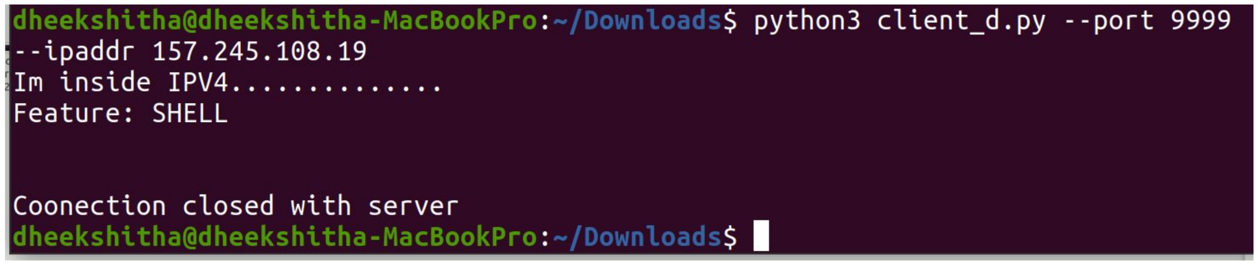
- If we type "**close**" then the client socket will be closed.

Server:



```
zsh> select 2
You are now connected to :103.57.133.42
103.57.133.42@zsh> close

Timestamp : 02:49:25



Timestamp : 02:49:25

Connection closed with 103.57.133.42

zsh> list
-----------Clients-----------
0   183.83.146.137   25820

1   103.10.133.111   58874

zsh> 
```

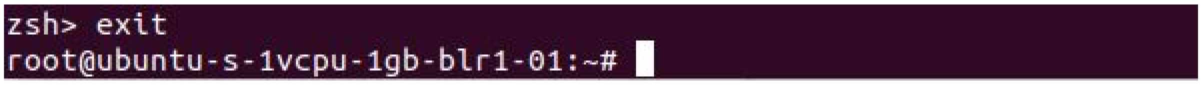
Client 2:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr 157.245.108.19
Im inside IPV4..............
Feature: SHELL


Coonection closed with server
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ 
```

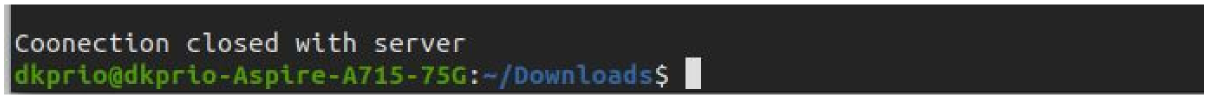
- If we type “**exit**” when we aren't in control of any of the client’s shells, then all the clients socket will be closed and the server socket closes its socket and the program ends.

Server:



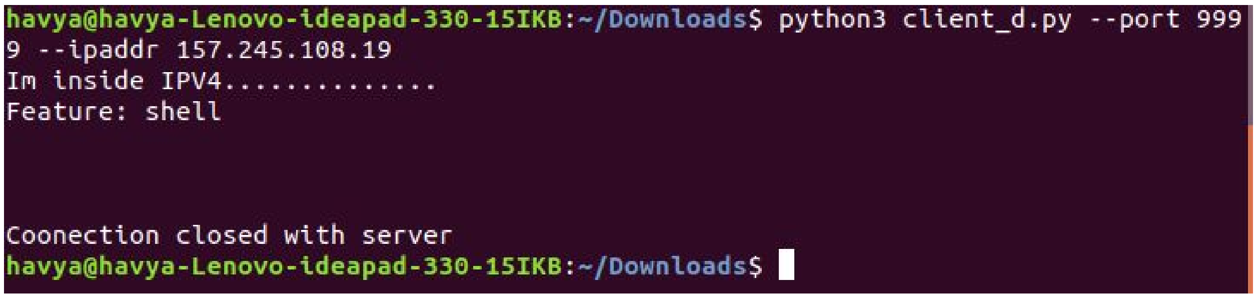
```
zsh> exit
root@ubuntu-s-1vcpu-1gb-blr1-01:~#
```

Client 0:



```
Coonection closed with server
dkprio@dkprio-Aspire-A715-75G:~/Downloads$
```

Client 1:



```
havya@havya-Lenovo-ideapad-330-15IKB:~/Downloads$ python3 client_d.py --port 999
9 --ipaddr 157.245.108.19
Im inside IPV4.............
Feature: shell



Coonection closed with server
havya@havya-Lenovo-ideapad-330-15IKB:~/Downloads$
```

2. **chatbox :** Two clients can chat with each other via our server

__**Client 1 command:**__

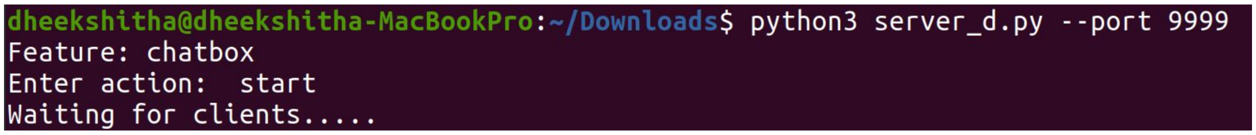
$ python3 client_2.py --port <port number> --ipaddr <ipv4 address of server>

__**Client 2 command:**__

$ python3 client_2.py --port <port number> --ipaddr <ipv6 address of server>

- Server waits for the 2 clients to connect.

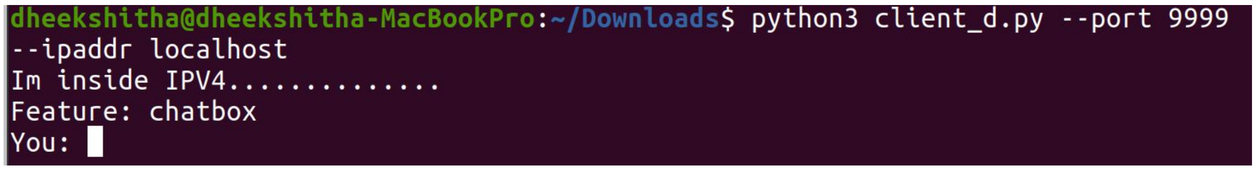
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
```

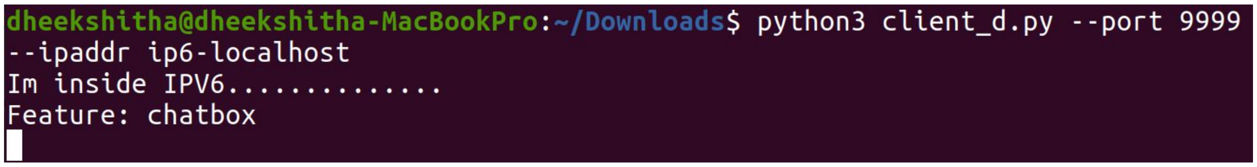
- For two clients to connect, one has to give IPv4 address while other has to give IPv6 address of the server as argument.

Client 1:



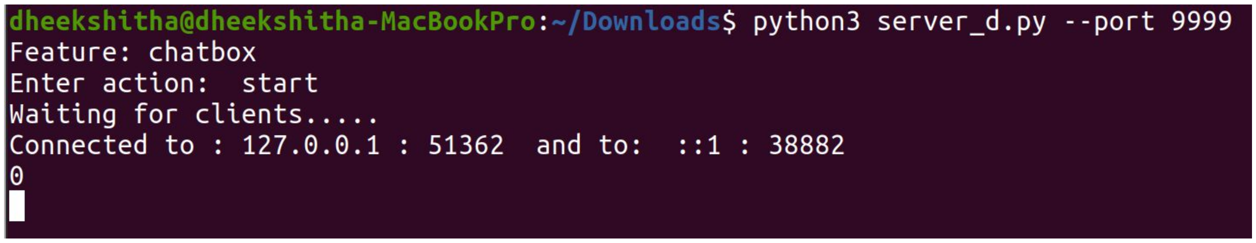
```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr localhost
Im inside IPV4.............
Feature: chatbox
You: 
```

Client 2:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr ip6-localhost
Im inside IPV6.............
Feature: chatbox
```

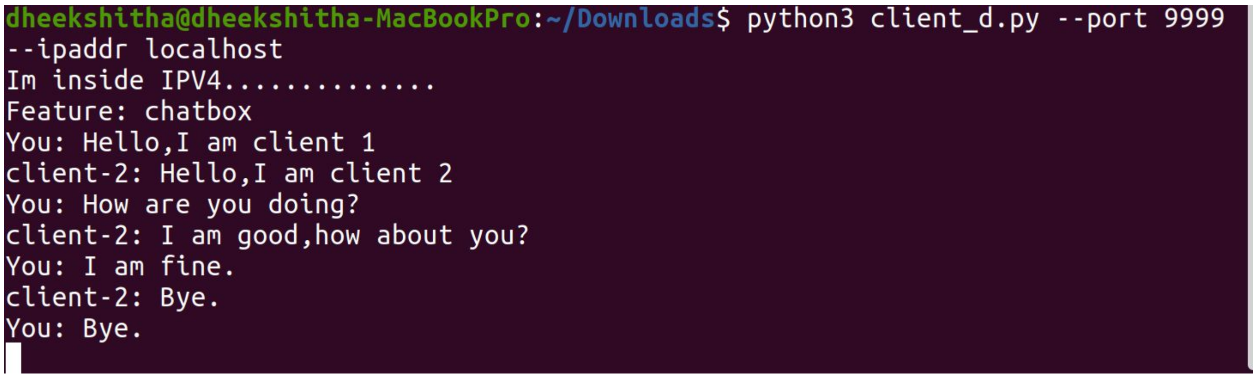
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51362  and to:  ::1 : 38882
0
```

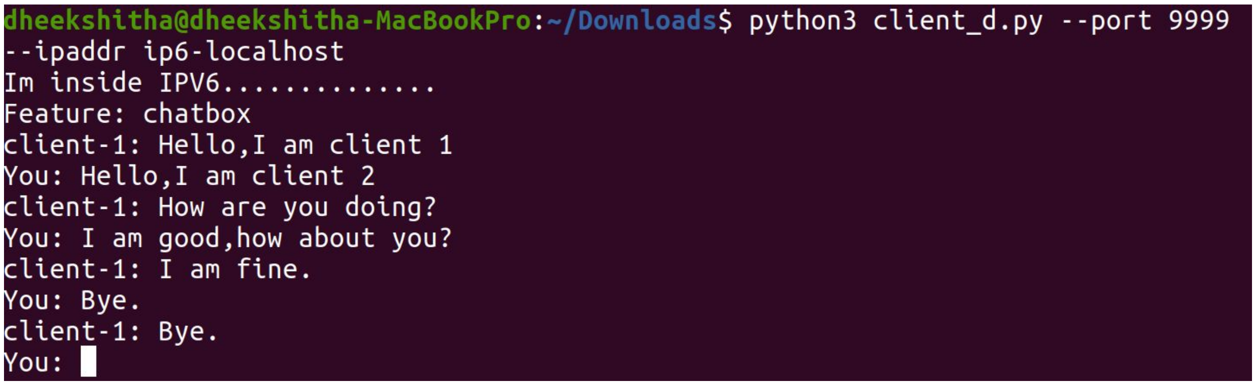
- Clients can now chat with each other and server keeps track of the number of messages sent by the server

Client 1:



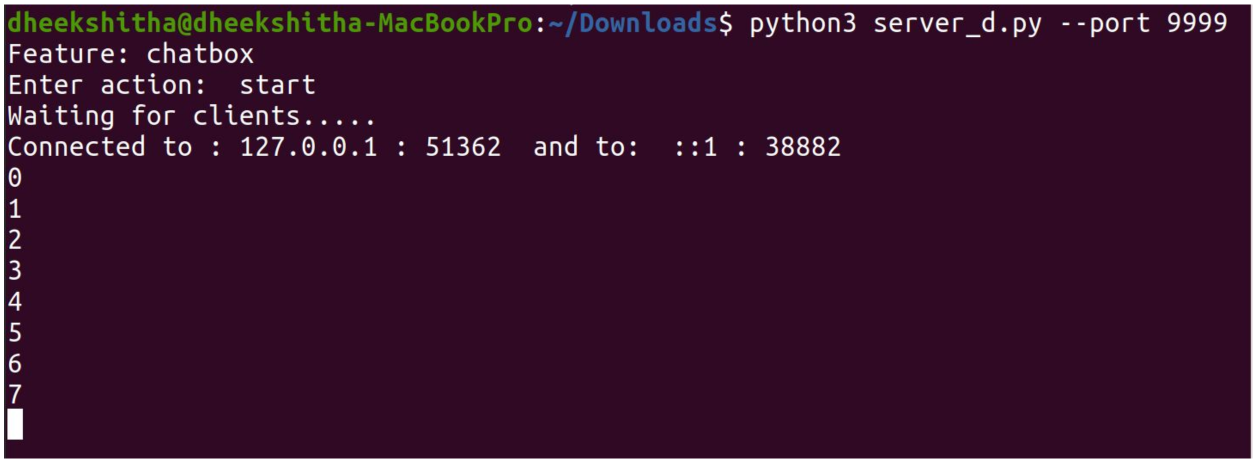
```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr localhost
Im inside IPV4.............
Feature: chatbox
You: Hello,I am client 1
client-2: Hello,I am client 2
You: How are you doing?
client-2: I am good,how about you?
You: I am fine.
client-2: Bye.
You: Bye.
```

Client 2:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr ip6-localhost
Im inside IPV6..............
Feature: chatbox
client-1: Hello,I am client 1
You: Hello,I am client 2
client-1: How are you doing?
You: I am good,how about you?
client-1: I am fine.
You: Bye.
client-1: Bye.
You: 
```

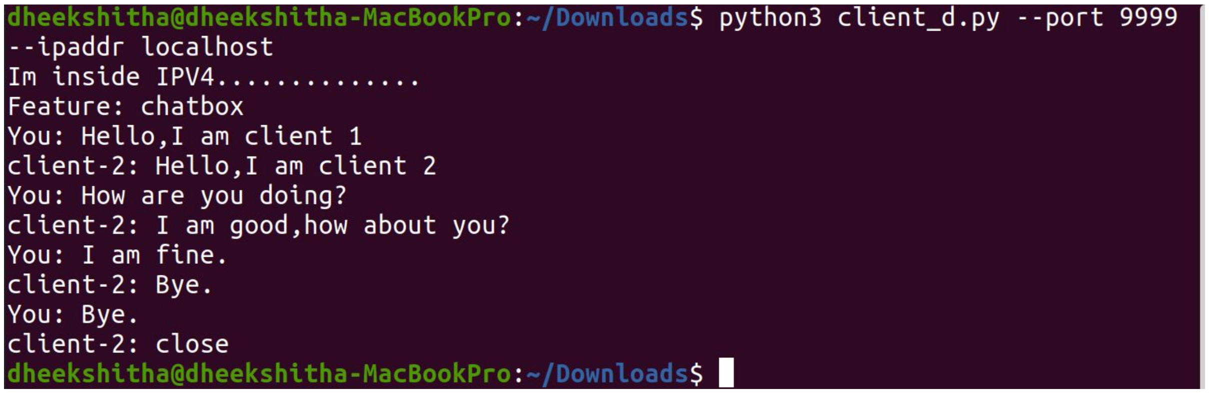
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51362  and to:  ::1 : 38882
0
1
2
3
4
5
6
7
```

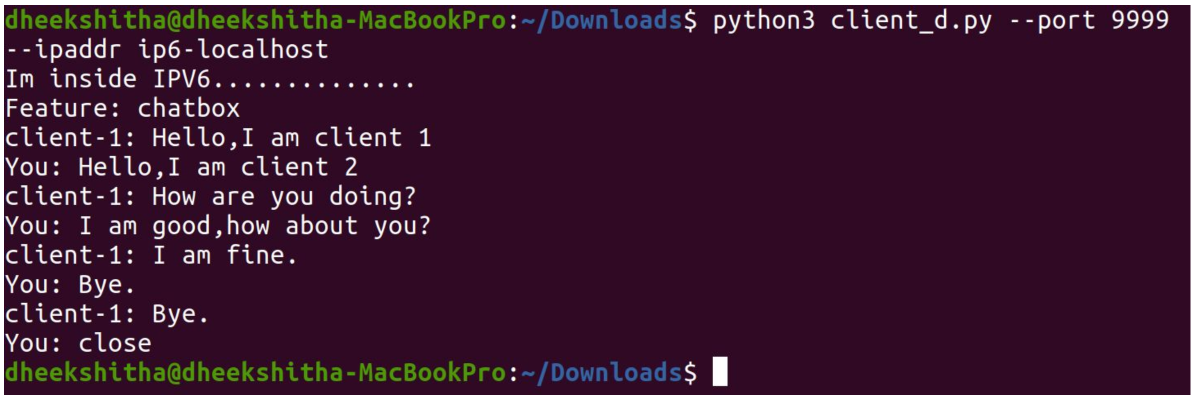
- If we type "**close**" in any of the clients, the clients will be closed and the server will be waiting for its next action.

Client 1:



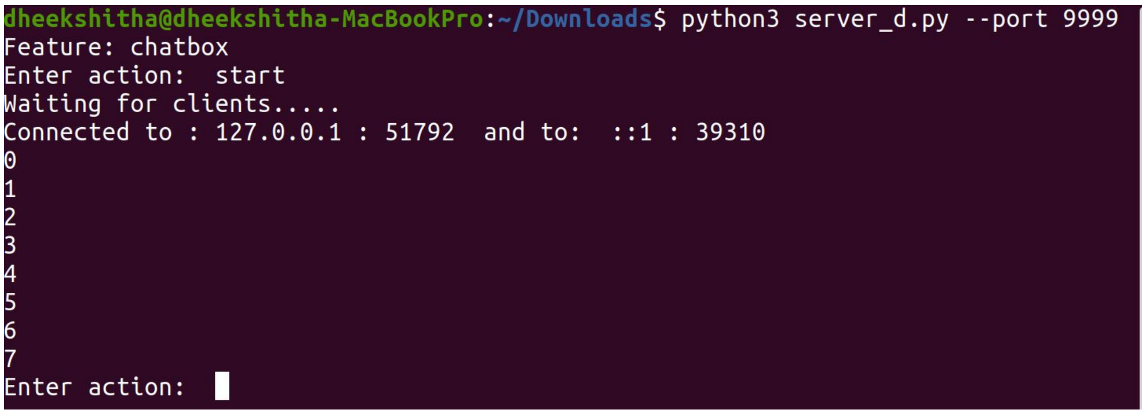
```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr localhost
Im inside IPV4..............
Feature: chatbox
You: Hello,I am client 1
client-2: Hello,I am client 2
You: How are you doing?
client-2: I am good,how about you?
You: I am fine.
client-2: Bye.
You: Bye.
client-2: close
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ 
```

Client 2:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 client_d.py --port 9999
--ipaddr ip6-localhost
Im inside IPV6..............
Feature: chatbox
client-1: Hello,I am client 1
You: Hello,I am client 2
client-1: How are you doing?
You: I am good,how about you?
client-1: I am fine.
You: Bye.
client-1: Bye.
You: close
dheekshitha@dheekshitha-MacBookPro:~/Downloads$
```

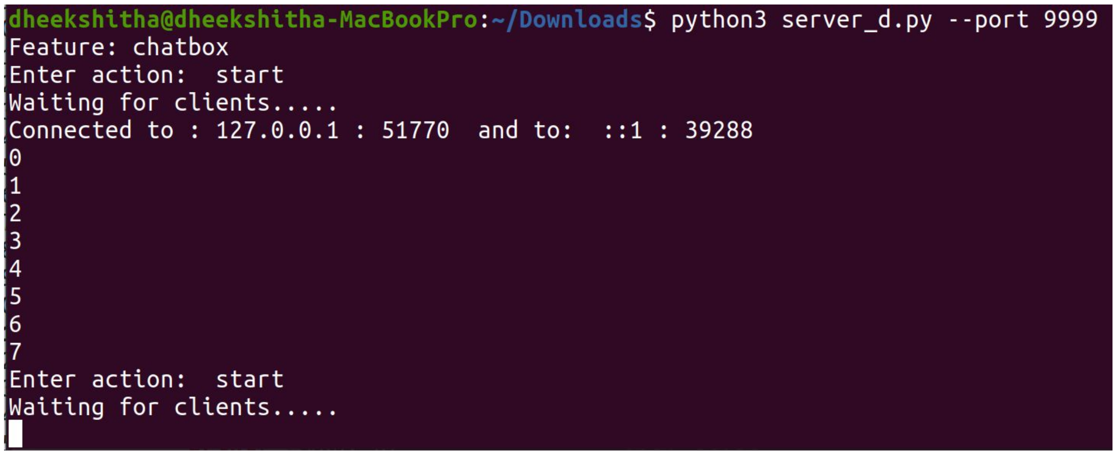
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51792  and to:  ::1 : 39310
0
1
2
3
4
5
6
7
Enter action:
```

- If we again type "**start**" in the server, it will wait for the any two clients to join.

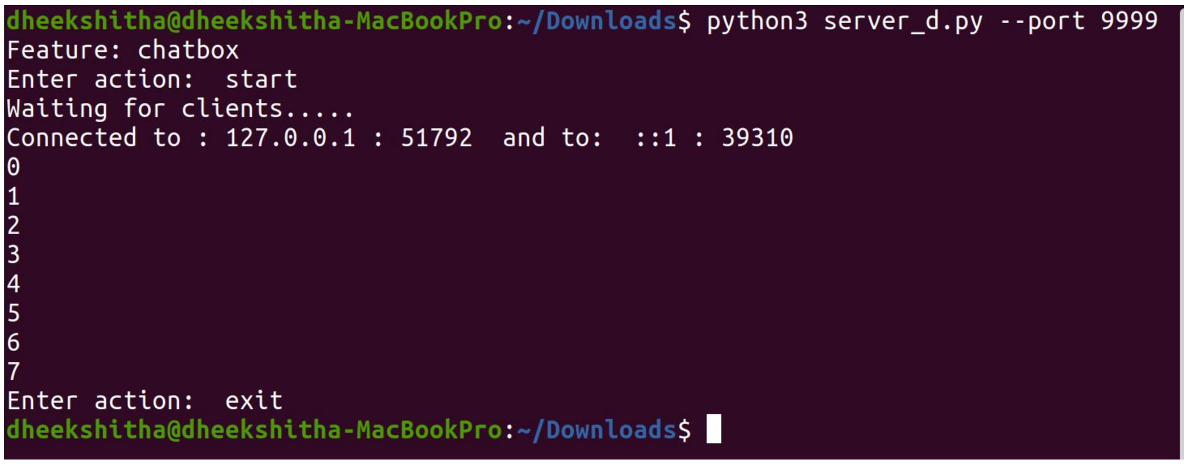
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51770  and to:  ::1 : 39288
0
1
2
3
4
5
6
7
Enter action:  start
Waiting for clients.....
```

- If we type “**exit**” in the server, the server closes the socket and the program ends.

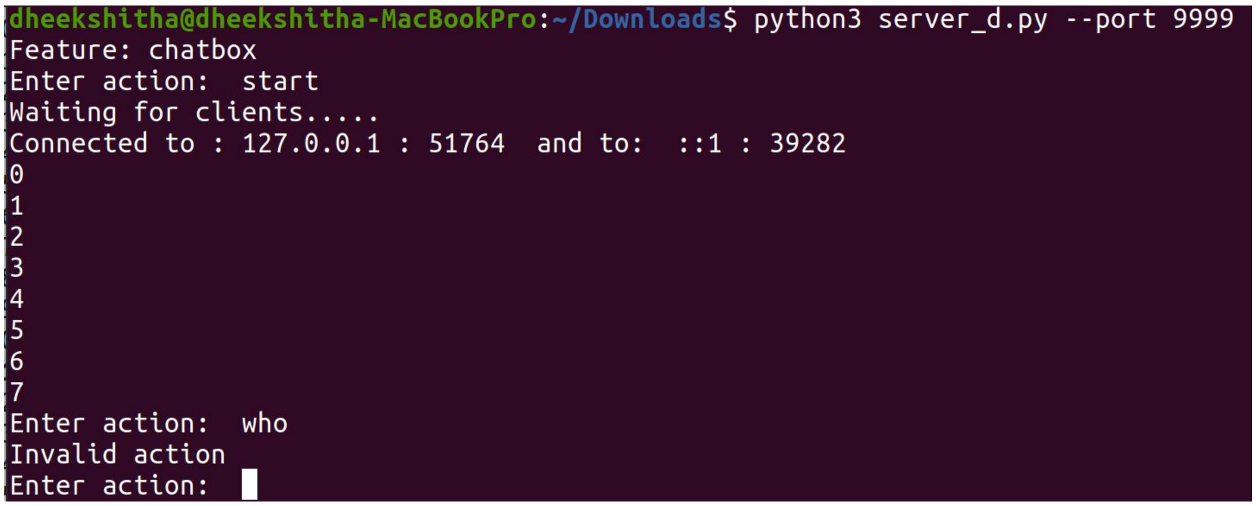
Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51792  and to:  ::1 : 39310
0
1
2
3
4
5
6
7
Enter action:  exit
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ 
```

- If we type anything other than “exit” or “start” in the server, the server prints an error message and waits for the user to give a valid action.

Server:



```
dheekshitha@dheekshitha-MacBookPro:~/Downloads$ python3 server_d.py --port 9999
Feature: chatbox
Enter action:  start
Waiting for clients.....
Connected to : 127.0.0.1 : 51764  and to:  ::1 : 39282
0
1
2
3
4
5
6
7
Enter action:  who
Invalid action
Enter action:  
```

References :

- [Threading in Python]
- https://docs.python.org/3/howto/argparse.html#id1
- https://docs.python.org/3/library/queue.html

Question #3:

Revise echo client and server to be protocol independent (support both IPv4 and IPv6).

Command :

```
$ python server_3.py <port_number>
$ python client_3.py <port_number> <ipv4/ipv6 address>
```

Example :

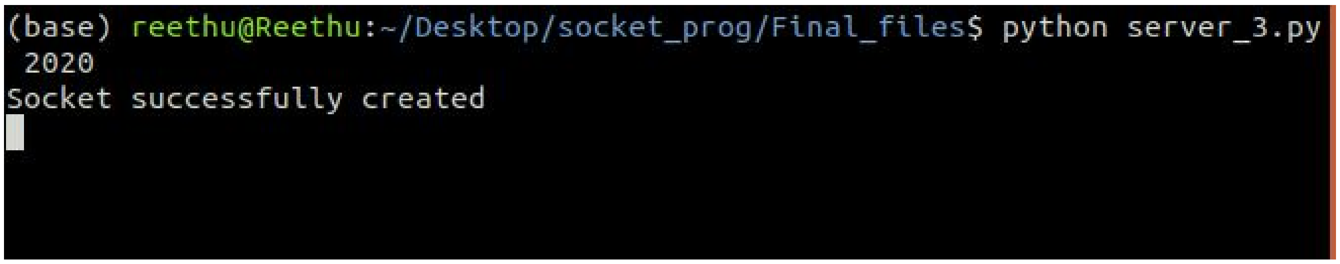
1. python server_3.py 1500
   python client_3.py 1500 127.0.0.1
2. python server_3.py 1500
   python client_3.py 1500 ::1

- If you have given IPv4 and this correctly creates a socket,binds to it then you can see successful creation along with the message I'm in IPv4.
- If you have given IPv6 and this correctly creates a socket,binds to it then you can see successful creation along with the message I'm in IPv6.
- Server is executed on two threads. One for IPv4 and the other for IPv6.
- It prompts to type a message in the client, after which that message is displayed on the server. Basically enabled messaging operations between server and client.

Connecting to a IPv4:

- Creating a socket on the server.

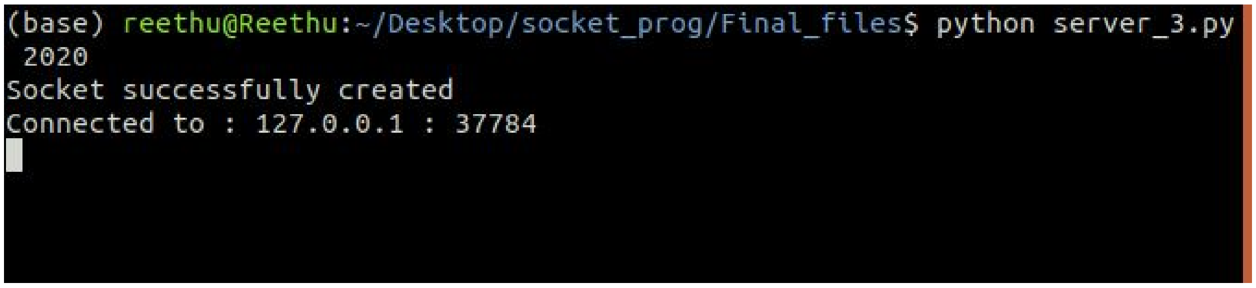
Server:



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2020
Socket successfully created
```

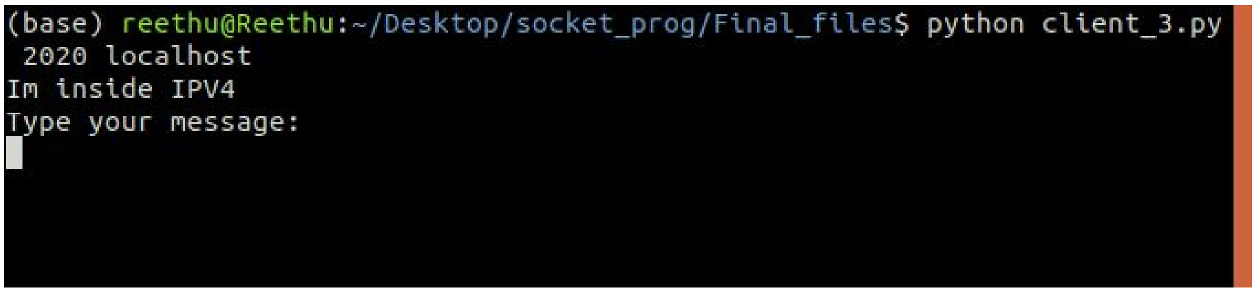
- Connecting to an IPv4 client.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2020
Socket successfully created
Connected to : 127.0.0.1 : 37784
```

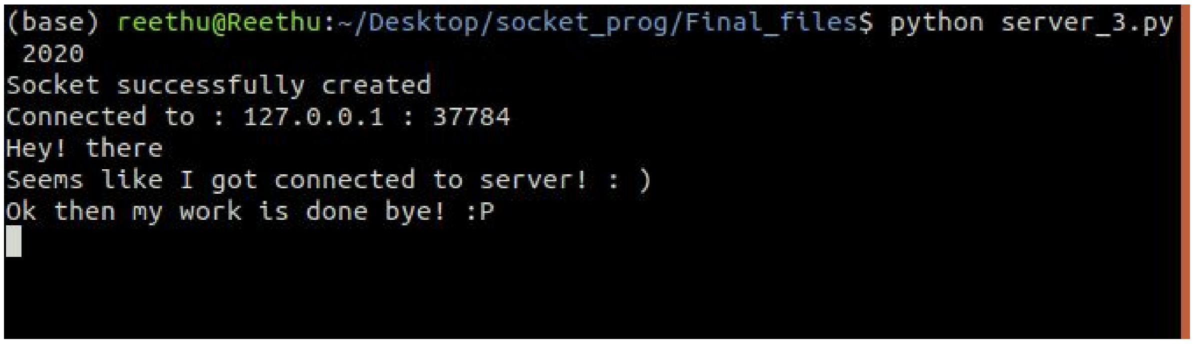
Client :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2020 localhost
Im inside IPV4
Type your message:
```

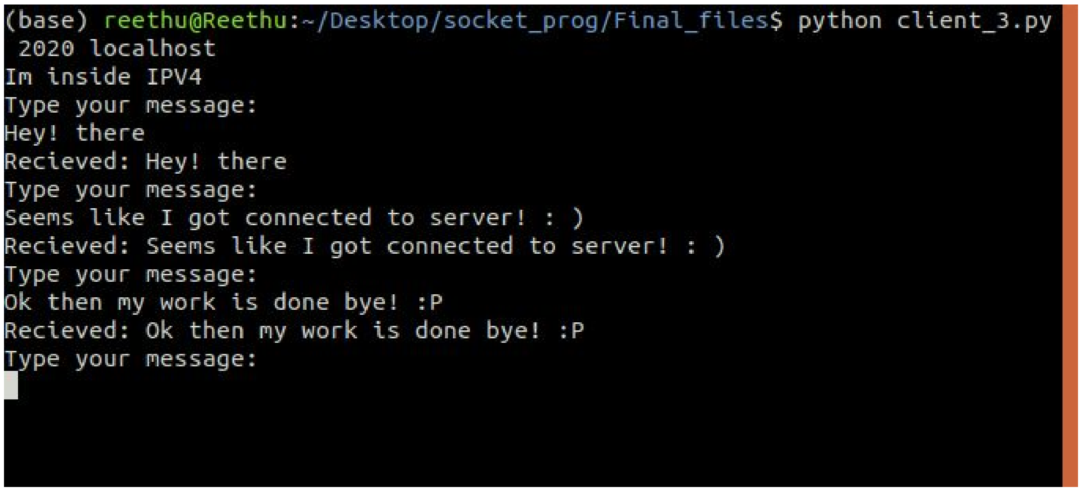
- Typing some messages from client to server.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2020
Socket successfully created
Connected to : 127.0.0.1 : 37784
Hey! there
Seems like I got connected to server! : )
Ok then my work is done bye! :P
```

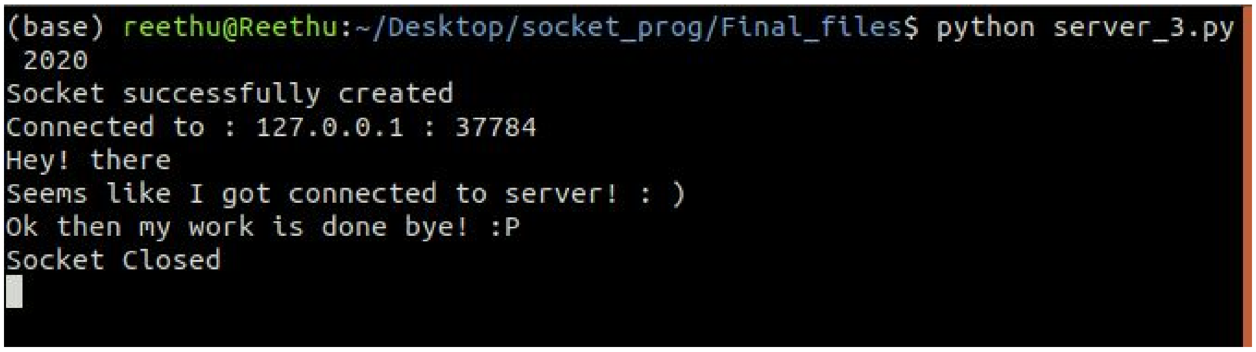
Client :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2020 localhost
Im inside IPV4
Type your message:
Hey! there
Recieved: Hey! there
Type your message:
Seems like I got connected to server! : )
Recieved: Seems like I got connected to server! : )
Type your message:
Ok then my work is done bye! :P
Recieved: Ok then my work is done bye! :P
Type your message:
```

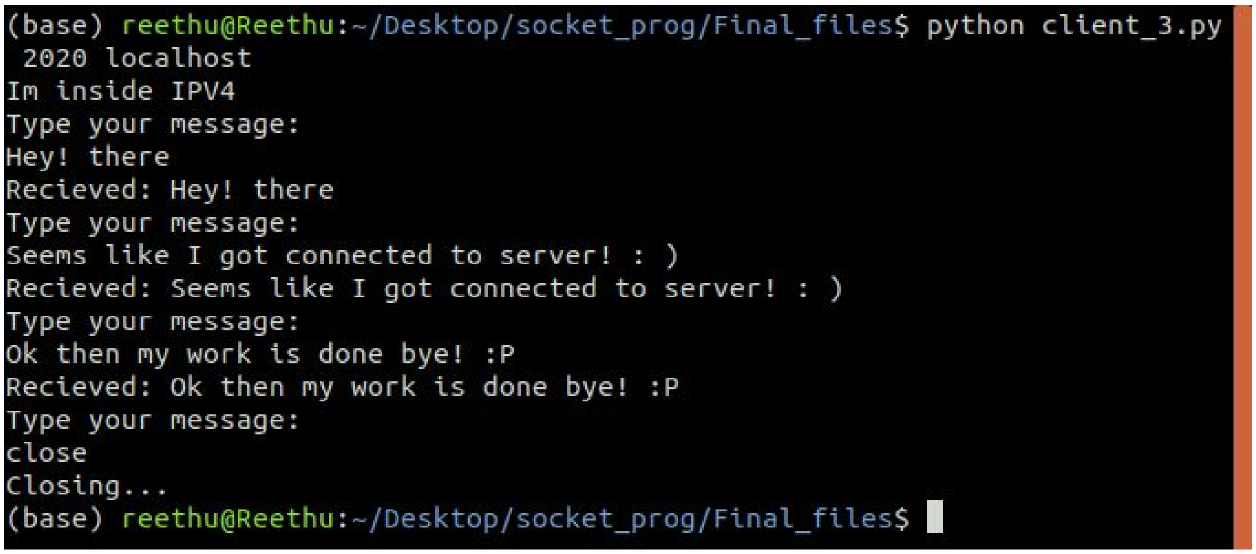
- If we type “close” then the client socket will be closed.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2020
Socket successfully created
Connected to : 127.0.0.1 : 37784
Hey! there
Seems like I got connected to server! : )
Ok then my work is done bye! :P
Socket Closed
```

Client :

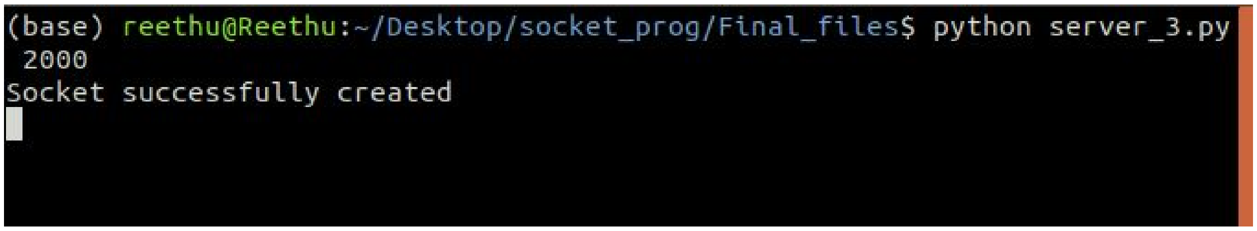


```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2020 localhost
Im inside IPV4
Type your message:
Hey! there
Recieved: Hey! there
Type your message:
Seems like I got connected to server! : )
Recieved: Seems like I got connected to server! : )
Type your message:
Ok then my work is done bye! :P
Recieved: Ok then my work is done bye! :P
Type your message:
close
Closing...
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ 
```

Connecting to a IPv6:

- Creating a socket on the server.

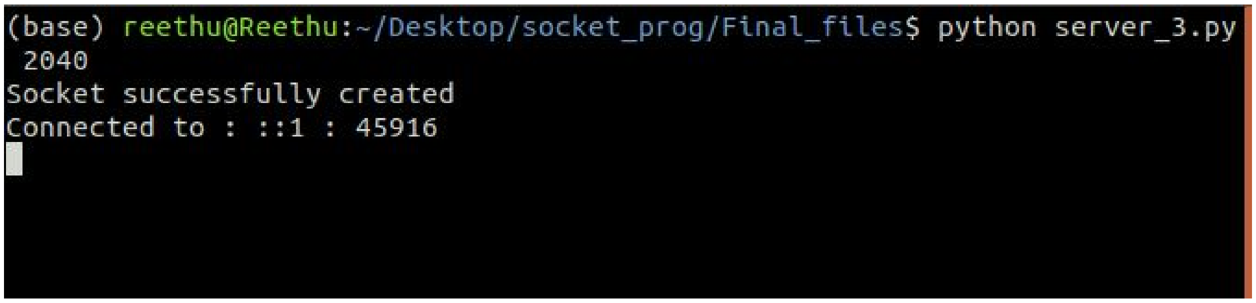
Server:



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2000
Socket successfully created
```

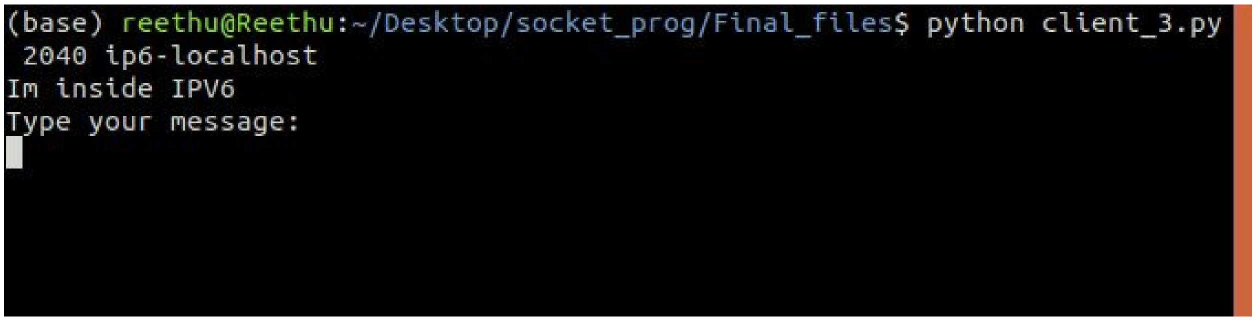
- Connecting to an IPv6 client.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2040
Socket successfully created
Connected to : ::1 : 45916
```

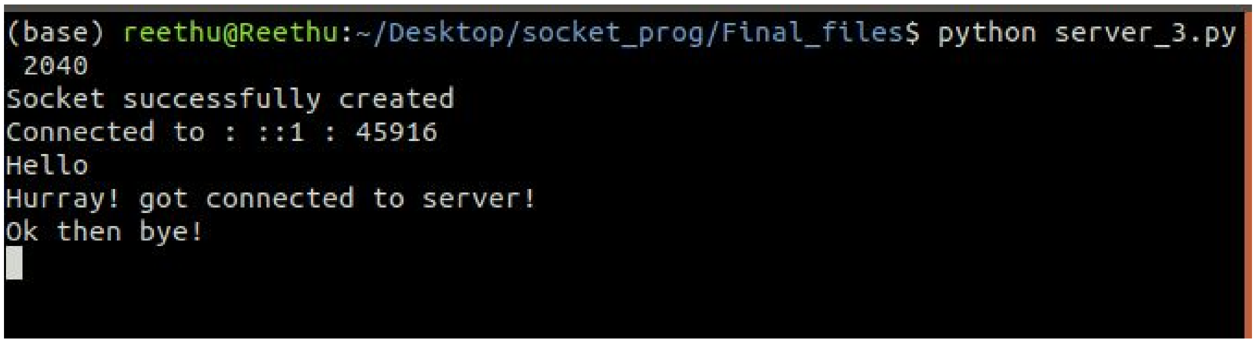
Client :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2040 ip6-localhost
Im inside IPV6
Type your message:
```

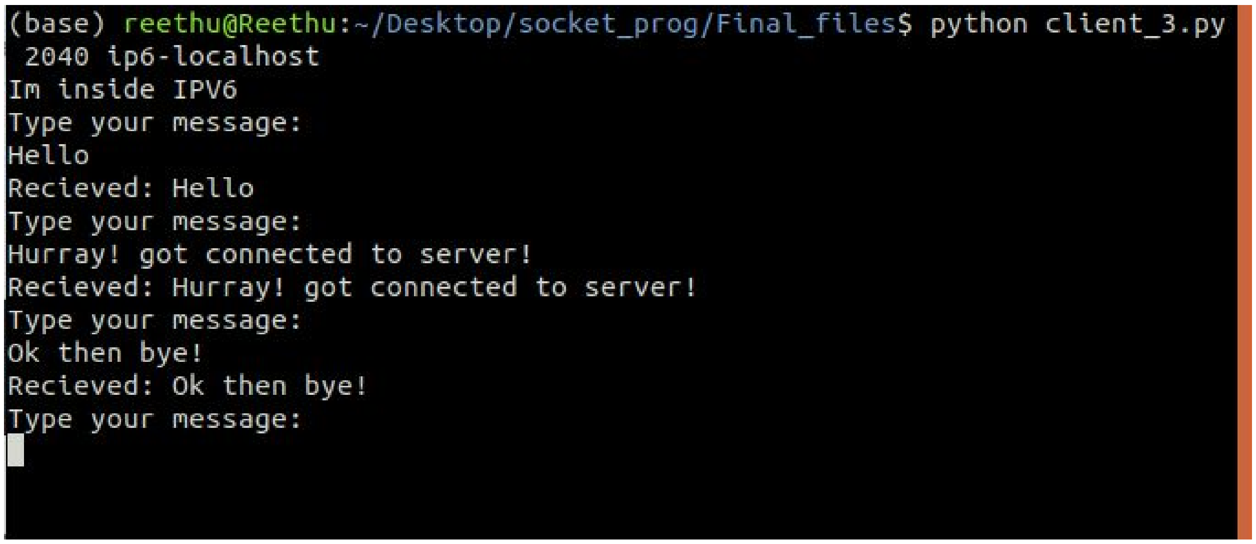
- Typing some messages from client to server.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2040
Socket successfully created
Connected to : ::1 : 45916
Hello
Hurray! got connected to server!
Ok then bye!
```

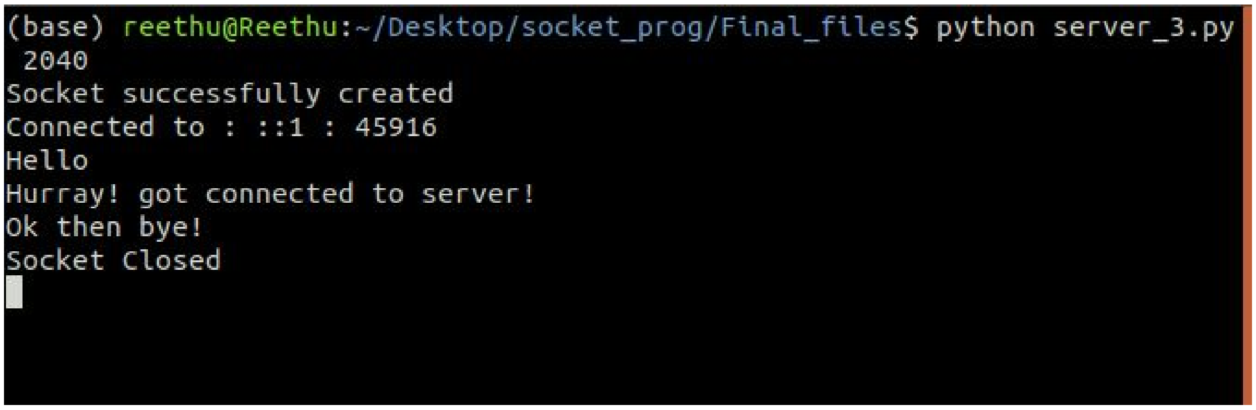
Client :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2040 ip6-localhost
Im inside IPV6
Type your message:
Hello
Recieved: Hello
Type your message:
Hurray! got connected to server!
Recieved: Hurray! got connected to server!
Type your message:
Ok then bye!
Recieved: Ok then bye!
Type your message:
```

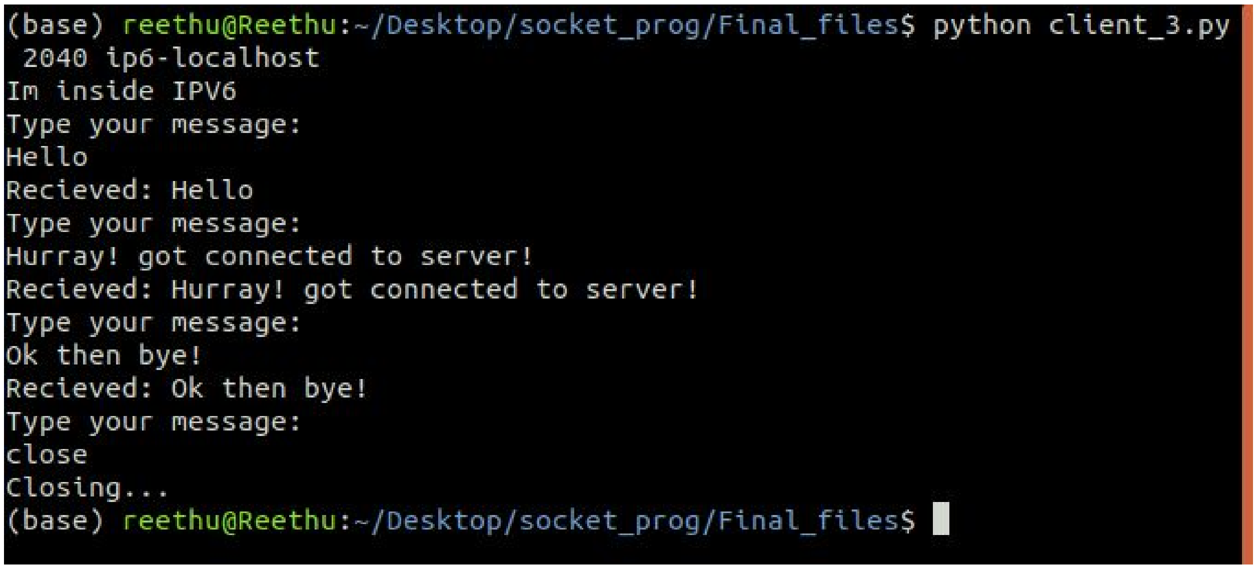
- If we type “close” then the client socket will be closed.

Server :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python server_3.py
 2040
Socket successfully created
Connected to : ::1 : 45916
Hello
Hurray! got connected to server!
Ok then bye!
Socket Closed
```

Client :



```
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$ python client_3.py
 2040 ip6-localhost
Im inside IPV6
Type your message:
Hello
Recieved: Hello
Type your message:
Hurray! got connected to server!
Recieved: Hurray! got connected to server!
Type your message:
Ok then bye!
Recieved: Ok then bye!
Type your message:
close
Closing...
(base) reethu@Reethu:~/Desktop/socket_prog/Final_files$
```

References :

- Arg Parse
- Socket Programming with Multithreading in Python
- Threading library in Python

.