Write a Socket program in python to connect to www.google.co.in and print "the socket has successfully connected to Google".

Program:-

```
import Socket

def main():
    target_host = 'www.google.co.in'
    target_port = 80

client = Socket.socket(Socket.AF_INET, Socket.
                       SOCK_STREAM)

try:
  client.connect((target_host, target_port))

print("The Socket has successfully connected
      to googe")

expect Exception as e:
    print(f"Error: {e}")
finally:
    client.close()
if __name__ == "__main__":
  main()
```

Output:- The socket has successfully connected to google.

9(9) write Socket programs in python to implement a simple TCP client server application and send "Hello" messages.

Server:-

```
import socket
s = socket.socket()
print ("Socket successfully created")
port = 12345
s.bind ('', port)
print ("Socket binded to %s" %(port))
s.listen(5)
print ("Socket is listening")
while True:
    c, addr = s.accept()
    print ('Got connection from', addr)
    c.send ("Thank you for connecting".encode())
    c.close()
    Break
```

Client Side:-

```
import socket
s = socket.socket()
port = 12345
s.connect (('127.0.0.1', port))
print (s.recv (1024).decode())
s.close()
```

- First of all we make a socket object.
- Then we connet to localhost on port 12345 and lastly we received data from the server and close connection.

Now save this file as client.py and run it from the terminal after starting the server script.

Output:

```
#start the server:
$Python Server.py
Socket Successfully created
Socket binded to 12345
Socket is listenning
Got connection from ('127.0.0.1', 52617)

# start the client:
$python client.py
Thank you for connecting.
```

9. (10) write socket program in python to implement a single UDP client server application and send "welcome" messages.

UDP client Server Programming: Server side:-

```
import Socket
localIP = "127.0.0.1"
localPort = 20001
buffersize = 1024
msgFromServer = "Hello UDP client"
bytesTO Send = Str. encode (msgFrom Server)
UDPServerSocket = Socket . Socket (family = Socket.
AF_INET, type = Socket.SOCK_DGRAM)
UDPServer socket . bind ((localIP, localPort))
print ("UDP Server up and listenning")
```

```
while (True):
    bytesAddressPair = UDPServerSocket.recvfrom
    (bufferSize)
    message = bytesAddressPair[0]
    address = bytesAddressPair[1]
    clientMsg = "Message from client: {}".format
    (message)
    clientIP = "client IP Address: {}".format
    (address)
    print(clientMsg)
    print(clientIP)

    UDPServerSocket.sendto(bytesToSend, address)
```

output:

```
UDP server up and listening.
Message from client: "Hello UDP server"
client IP Address: ("127.0.0.1", 51696)
```

UDP client server Programming: client side

```
import socket
msgFromClient = "Hello UDP Server"
bytesToSend = str.encode(msgFromClient)
ServerAddressPort = ("127.0.0.1", 20001)
bufferSize = 1024
UDPClientSocket = socket.socket(family = socket.
    AF_INET, type = socket.SOCK_DGRAM)
```

socket.recvfrom

0]

1]

ient: {}", format

ess: {}". format

To Send, address)

ening.

ello UDP Server"

.0.0.1", 51696)

lient Side

DP Server"

ode (msg From(client)

.0.1", 20001)

ket(family = Socket

SOCK_DGRAM)

UDPClientSocket.sendto(bytes TO Send, Servere address
Port)

msgFromServer = UDPClientSocket.recvfrom(buffersize)

msg = "Message from Server {}". format(msgFrom
Server[0])

print(msg)

output

Message From Server "Hello UDP client"

Write client Server Socket programs to implement file transfer using TCP/IPV4.

TCP-SERVER.py:-

```
import socket
if __name__ == '__main__':
host = '127.0.0.1'
port = 8080
totalclient = int(input("Enter number of clients"))
Sock = Socket.Socket(Socket.AF_INET, Socket.
SOCK_STREAM)
Sock.bind((host, port))
Sock.listen(totalclient)
connections = []
print('Initiating clients')
for i in range(totalclient):
 Conn = Sock.accept()
connections.append(conn)
print('connected with client', i+1)
```

```
fileno = 0
idx = 0
for conn in connections:
  idx += 1
  data = conn[0].recv(1024).decode()
  if not data:
    continue
  filename = 'output'+str(fileno)+'.txt'
  fileno = fileno + 1
  fo = open(filename, "w")
  while data:
    if not data:
      break
    else:
      fo.write(data)
      data = conn[0].recv(1024).decode()
  print()
  print('Receiving file from client', idx)
  print()
  print('Received successfully! New filename
  is:', filename)
  fo.close()
for conn in connections:
  conn[0].close()
```

TCP- CLIENT. by:-

```
import socket
if __name__ == '__main__':
    host = '127.0.0.1'
    port = 8080
sock = socket.socket(socket.AF_INET, socket.
SOCK_STREAM)
sock.connect((host, port))
while True:
    filename = input('Input filename you want
    to send:')
    try:
    fi = open(filename, "rb")
    data = fi.read()
    if not data:
        break
    while data:
        sock.send(str(data).encode())
        data = fi.read()
    fi.close()
except IOError:
    print('you entered an invalid filename!
        please enter a valid name')
```

Q(12) Build a simple chat room using python socket programming and allow multiple clients to connect and transfer messages.

Step 1: Setting up the project

First, let create a new directory for our project and navigate into it:

```
Mkdir chat_app
Cd Chat_app
```

Step 2: Implementing the Server

Create a new file called Server.py and open it in your favourite text editor. We'll start by importing the necessary libraries:

```
import Socket
import threading
```

Next let's define the server code. We'll implement a function to handle each client connection and communication:

```
def handle_client (client_socket):
    while True:
        data = client_socket.recv (1024)
        if not data:
            break
        message = data.decode ('utf-8')
        print (f"received message: {message}")
        response = "Server received your message:" + message
        client_socket.Sendall (response.encode('utf-8'))
    client_socket.close()
```

python socket
tiple clients to
s.

tory for our project

ver.py and open it
we'll start by
es:

ode. we'll implement
lient connection

Socket):

.recv(1024)

('utf-8')
sage: {message}")
eceived your

onse.encode('utf-8'))

In the handle_client function, we use a while loop to continuously receive data from the client. If the data received is empty, it means the client has disconnected, and we break out of the loop. Otherwise, we decode data and print it to the server's console.

Now, let's create the main function to set up the server:

```
def main():
    server_socket = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
    host = '127.0.0.1'
    port = 12345
    server_socket.bind((host, port))
    server_socket.listen(5)
    print(f"Server listening on {host}:{port}")

    while True:
        client_socket, client_address = server_socket.
        accept()
        print(f"Accepted connection from {client_
        address}")
        client_handler = threading.Thread(target=
        handle_client, args=(client_socket,))
        client_handler.start()
if __name__ == "__main__":
    main()
```

Step-3 · Implementing the client

Next create a new file called client.py and open it in your text editor.

```
import socket
import threading
```

Next implement the main function for the client

```
def main():
    client_socket = socket.socket(socket.AF_
INET, socket.SOCK_STREAM)
host = '127.0.0.1'
port = 12345
client_socket.connect((host, port))

while True:
    message = input("Enter your message:")
    client_socket.sendall(message.encode
('utf-8'))
    data = client_socket.recv(1024)
    response = data.decode('utf-8')
    print(f"Server response: {response}")
if __name__ == "__main__":
main()
```

Step 4: Testing the chat Application

Now, let's test our chat application with some snap shots to see how it works.

1. Start the server by running python.py in the terminal. open a new terminal window and run the client script using python client.py:

```
C:\Users\Admin\Documents\Socket_Programming_
  python > Python client.py
Enter your message:
```

2. The Server terminal will show a message indicating that it has accepted the client connection:

```
C:\Users\Admin\Documents\Socket_Programming_
python > Python Server.py
Server listenning on 127.0.0.1:12345
Accepted connection from ('127.0.0.1', 58366)
```

4. To test with multiple clients, open more terminal windows and run client.py in each of them.

5. once multiple clients are connected, the server terminal will display messages for each client connection:

```
C:\Users\Admin\Documents\Socket_Programming_Python
> Python Server.py
Server listenning on 127.0.0.1:12345
```

Accepted connection from ('127.0.0.1', 58366)
Accepted connection from ('127.0.0.1', 58374)
Accepted connection from ('127.0.0.1', 58385)

In the above examples, I have created 3 instances

6. Send a message from one of the clients:

C:\Users\Admin\Documents\Socket-Programming-Python > python client.py
Enter your message : Hi
Server response : Server received your message : Hi
Enter your message :

7. The server will receive the message and send a response back to the client

C:\Users\Admin\Documents\Socket-Programming-Python > python server.py
Server listening on 127.0.0.1:12345
Accepted connection from ('127.0.0.1', 58402)
Received message : Hi