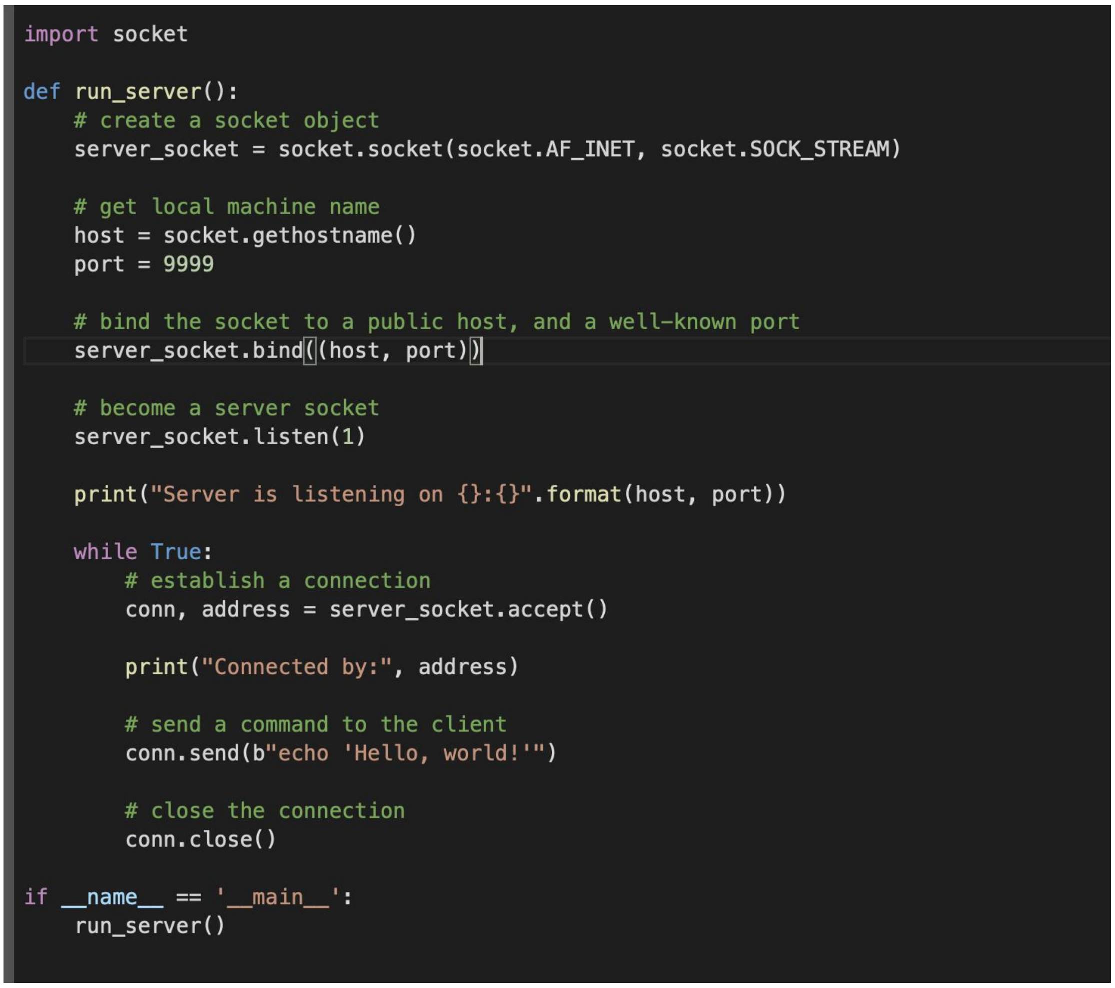
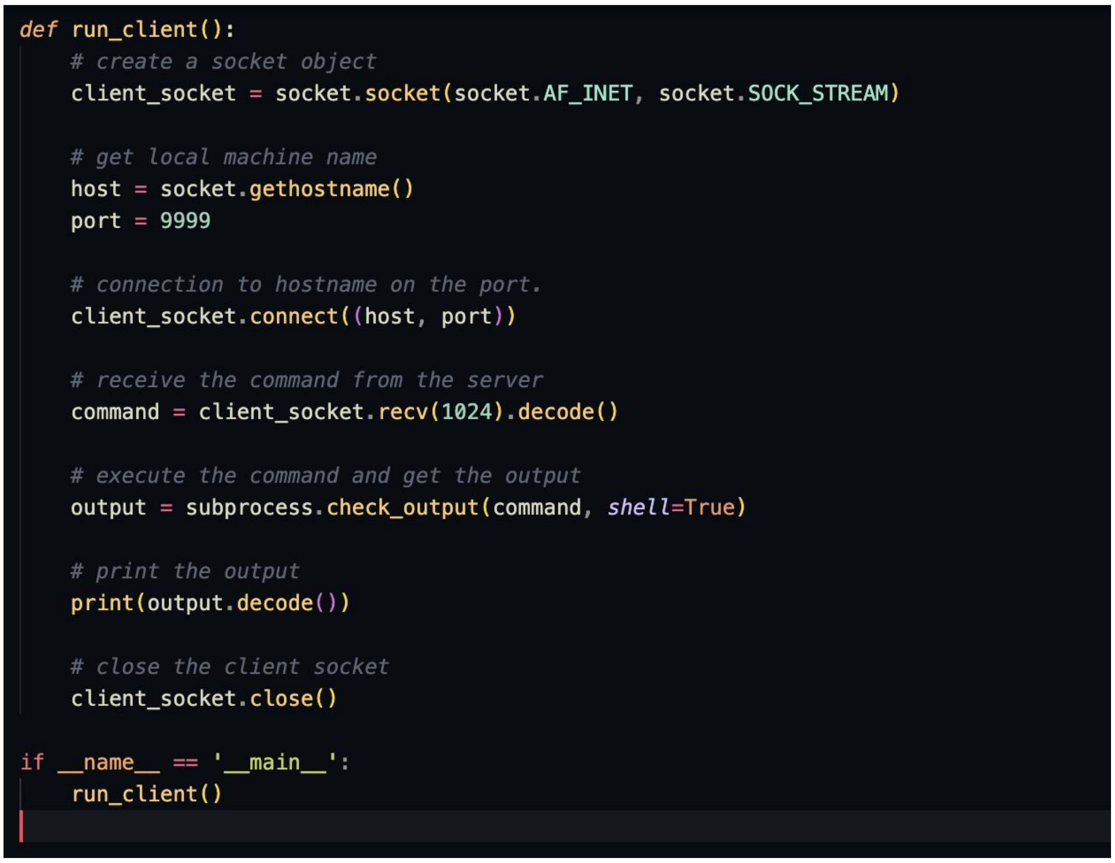
Week 12
Networking Programming
Name: Ronit Kumar **Reg: RA2111032010009** **Batch: CSE w/s IOT [T2]**

1. Create a Simple Client Server Application using TCP Socket where the server issues
a command which will be executed at the client slide as a process of remote command execution.

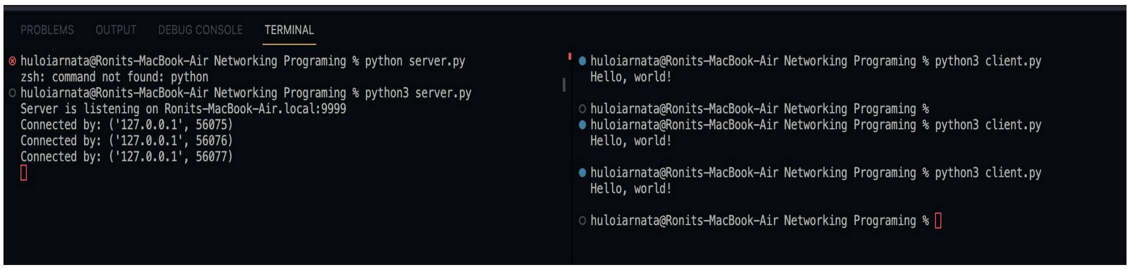


```
import socket

def run_server():
    # create a socket object
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()
    port = 9999

    # bind the socket to a public host, and a well-known port
    server_socket.bind((host, port))

    # become a server socket
    server_socket.listen(1)

    print("Server is listening on {}:{}".format(host, port))

    while True:
        # establish a connection
        conn, address = server_socket.accept()

        print("Connected by:", address)

        # send a command to the client
        conn.send(b"echo 'Hello, world!'")

        # close the connection
        conn.close()

if __name__ == '__main__':
    run_server()
```



```python
def run_client():
    # create a socket object
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # get local machine name
    host = socket.gethostname()
    port = 9999

    # connection to hostname on the port.
    client_socket.connect((host, port))

    # receive the command from the server
    command = client_socket.recv(1024).decode()

    # execute the command and get the output
    output = subprocess.check_output(command, shell=True)

    # print the output
    print(output.decode())

    # close the client socket
    client_socket.close()

if __name__ == '__main__':
    run_client()
```

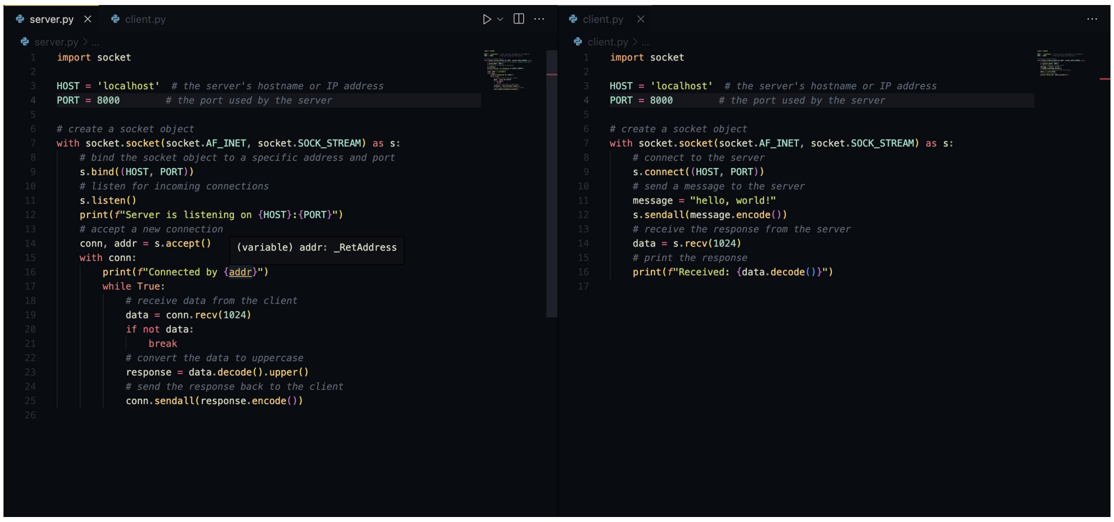
Output:



```
huloiarnata@Ronits-MacBook-Air Networking Programing % python server.py
zsh: command not found: python
huloiarnata@Ronits-MacBook-Air Networking Programing % python3 server.py
Server is listening on Ronits-MacBook-Air.local:9999
Connected by: ('127.0.0.1', 56075)
Connected by: ('127.0.0.1', 56076)
Connected by: ('127.0.0.1', 56077)
```

```
huloiarnata@Ronits-MacBook-Air Networking Programing % python3 client.py
Hello, world!

huloiarnata@Ronits-MacBook-Air Networking Programing %
huloiarnata@Ronits-MacBook-Air Networking Programing % python3 client.py
Hello, world!

huloiarnata@Ronits-MacBook-Air Networking Programing % python3 client.py
Hello, world!

huloiarnata@Ronits-MacBook-Air Networking Programing %
```

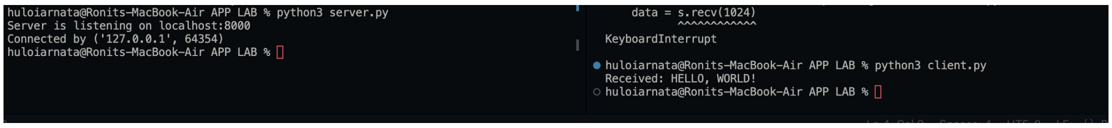
**2. Write a Socket-based Python server program that responds to client messages as
follows: When it receives a message from a client, it simply converts the message into
all uppercase leters and sends back the same to the client. Write both client and
server programs demonstrating this.**



```python
import socket

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # bind the socket object to a specific address and port
    s.bind((HOST, PORT))
    # listen for incoming connections
    s.listen()
    print(f"Server is listening on {HOST}:{PORT}")
    # accept a new connection
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            # receive data from the client
            data = conn.recv(1024)
            if not data:
                break
            # convert the data to uppercase
            response = data.decode().upper()
            # send the response back to the client
            conn.sendall(response.encode())
```

```python
import socket

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # connect to the server
    s.connect((HOST, PORT))
    # send a message to the server
    message = "hello, world!"
    s.sendall(message.encode())
    # receive the response from the server
    data = s.recv(1024)
    # print the response
    print(f"Received: {data.decode()}")
```

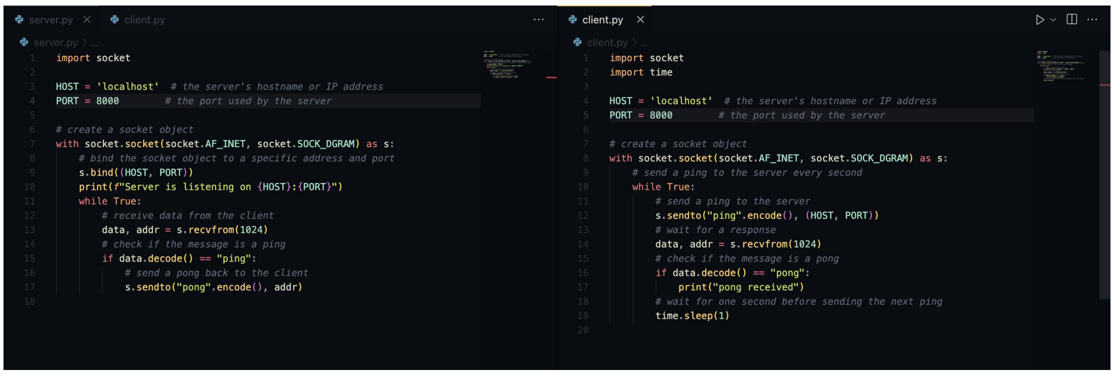
Output:



```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 server.py
Server is listening on localhost:8000
Connected by ('127.0.0.1', 64354)
huloiarnata@Ronits-MacBook-Air APP LAB %
```

```
    data = s.recv(1024)
           ^^^^^^^^^^^^
KeyboardInterrupt

huloiarnata@Ronits-MacBook-Air APP LAB % python3 client.py
Received: HELLO, WORLD!
huloiarnata@Ronits-MacBook-Air APP LAB %
```

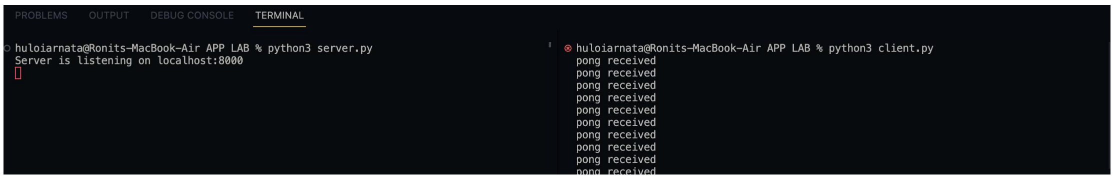
**3. Write a ping-pong client and server application. When a client sends a ping message to the server, the server will respond with a pong message. Other messages
sent by the client can be safely dropped by the server.**



```python
import socket

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    # bind the socket object to a specific address and port
    s.bind((HOST, PORT))
    print(f"Server is listening on {HOST}:{PORT}")
    while True:
        # receive data from the client
        data, addr = s.recvfrom(1024)
        # check if the message is a ping
        if data.decode() == "ping":
            # send a pong back to the client
            s.sendto("pong".encode(), addr)
```

```python
import socket
import time

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    # send a ping to the server every second
    while True:
        # send a ping to the server
        s.sendto("ping".encode(), (HOST, PORT))
        # wait for a response
        data, addr = s.recvfrom(1024)
        # check if the message is a pong
        if data.decode() == "pong":
            print("pong received")
        # wait for one second before sending the next ping
        time.sleep(1)
```

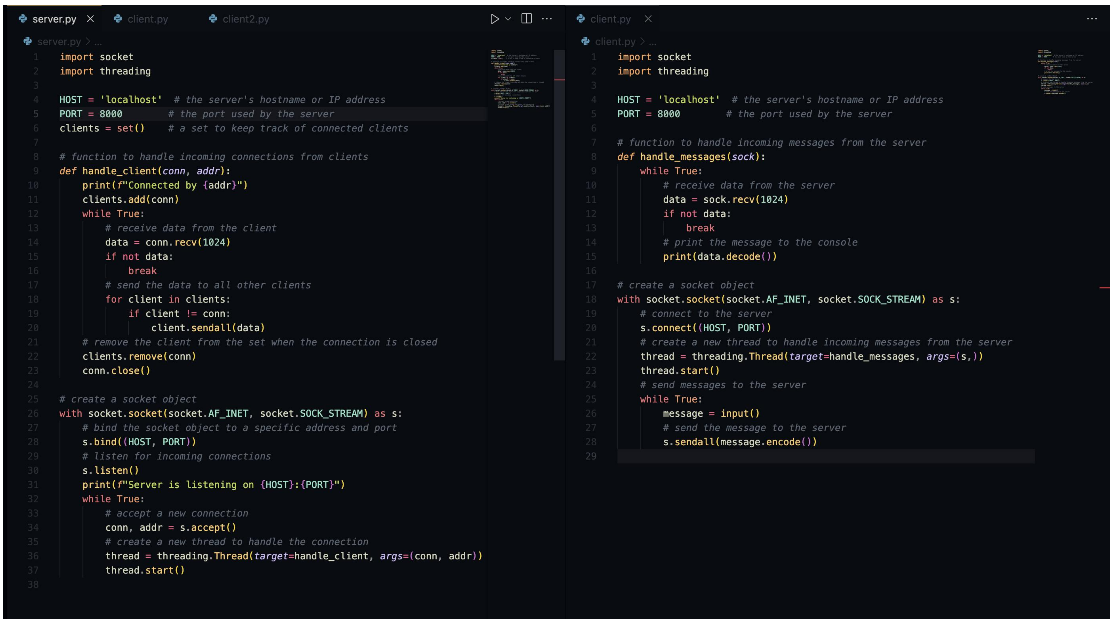
Output:



```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 server.py
Server is listening on localhost:8000
```

```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 client.py
pong received
pong received
pong received
pong received
pong received
pong received
pong received
pong received
pong received
pong received
```

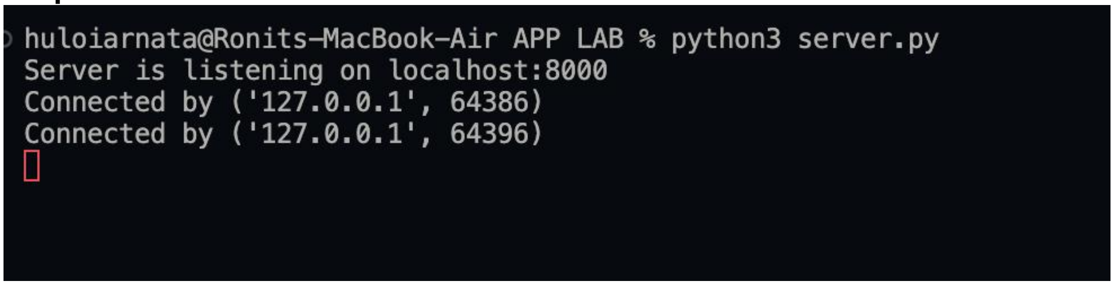
4. Write a Socket based program server-client to simulate a simple chat application where the server is multithreaded which can serve multiple clients at the same time.



```python
import socket
import threading

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server
clients = set()     # a set to keep track of connected clients

# function to handle incoming connections from clients
def handle_client(conn, addr):
    print(f"Connected by {addr}")
    clients.add(conn)
    while True:
        # receive data from the client
        data = conn.recv(1024)
        if not data:
            break
        # send the data to all other clients
        for client in clients:
            if client != conn:
                client.sendall(data)
    # remove the client from the set when the connection is closed
    clients.remove(conn)
    conn.close()

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # bind the socket object to a specific address and port
    s.bind((HOST, PORT))
    # listen for incoming connections
    s.listen()
    print(f"Server is listening on {HOST}:{PORT}")
    while True:
        # accept a new connection
        conn, addr = s.accept()
        # create a new thread to handle the connection
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

```

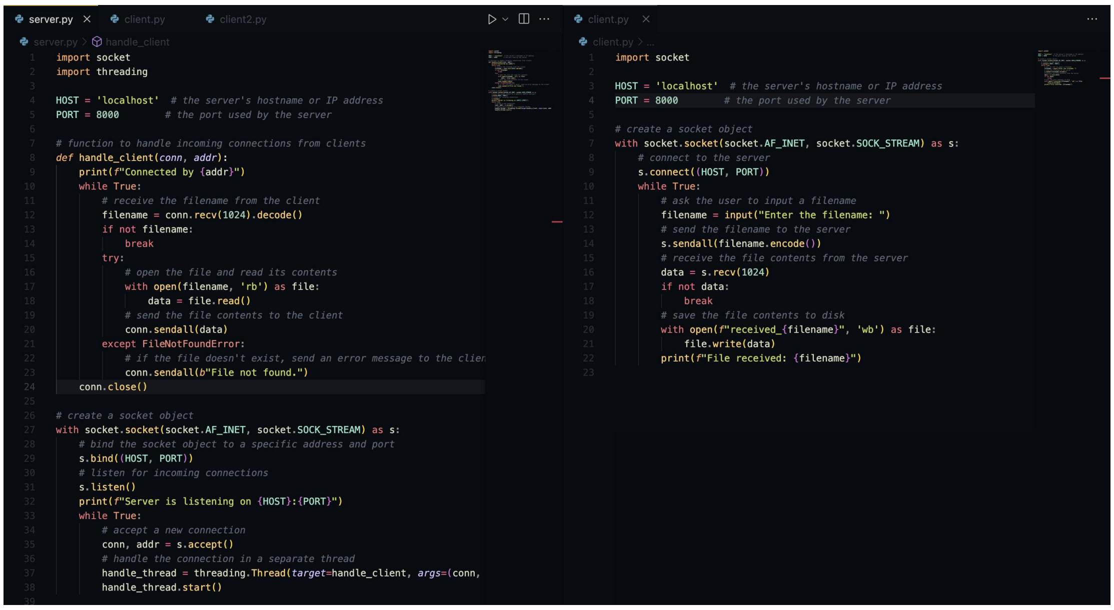
```python
import socket
import threading

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000         # the port used by the server

# function to handle incoming messages from the server
def handle_messages(sock):
    while True:
        # receive data from the server
        data = sock.recv(1024)
        if not data:
            break
        # print the message to the console
        print(data.decode())

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # connect to the server
    s.connect((HOST, PORT))
    # create a new thread to handle incoming messages from the server
    thread = threading.Thread(target=handle_messages, args=(s,))
    thread.start()
    # send messages to the server
    while True:
        message = input()
        # send the message to the server
        s.sendall(message.encode())

```

Output:



```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 server.py
Server is listening on localhost:8000
Connected by ('127.0.0.1', 64386)
Connected by ('127.0.0.1', 64396)
```

5. Write a Socket based program server-client to simulate Simple File Transfer Protocol using TCP Sockets.



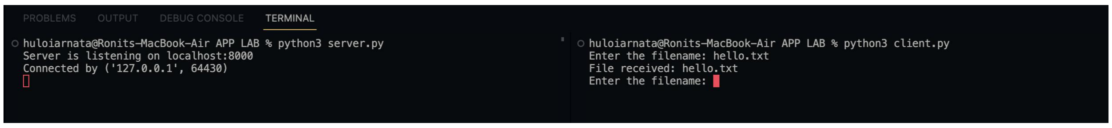
server.py

```python
import socket
import threading

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000        # the port used by the server

# function to handle incoming connections from clients
def handle_client(conn, addr):
    print(f"Connected by {addr}")
    while True:
        # receive the filename from the client
        filename = conn.recv(1024).decode()
        if not filename:
            break
        try:
            # open the file and read its contents
            with open(filename, 'rb') as file:
                data = file.read()
            # send the file contents to the client
            conn.sendall(data)
        except FileNotFoundError:
            # if the file doesn't exist, send an error message to the clien
            conn.sendall(b"File not found.")
    conn.close()

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # bind the socket object to a specific address and port
    s.bind((HOST, PORT))
    # listen for incoming connections
    s.listen()
    print(f"Server is listening on {HOST}:{PORT}")
    while True:
        # accept a new connection
        conn, addr = s.accept()
        # handle the connection in a separate thread
        handle_thread = threading.Thread(target=handle_client, args=(conn,
        handle_thread.start()
```

client.py

```python
import socket

HOST = 'localhost'  # the server's hostname or IP address
PORT = 8000        # the port used by the server

# create a socket object
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # connect to the server
    s.connect((HOST, PORT))
    while True:
        # ask the user to input a filename
        filename = input("Enter the filename: ")
        # send the filename to the server
        s.sendall(filename.encode())
        # receive the file contents from the server
        data = s.recv(1024)
        if not data:
            break
        # save the file contents to disk
        with open(f"received_{filename}", 'wb') as file:
            file.write(data)
        print(f"File received: {filename}")
```

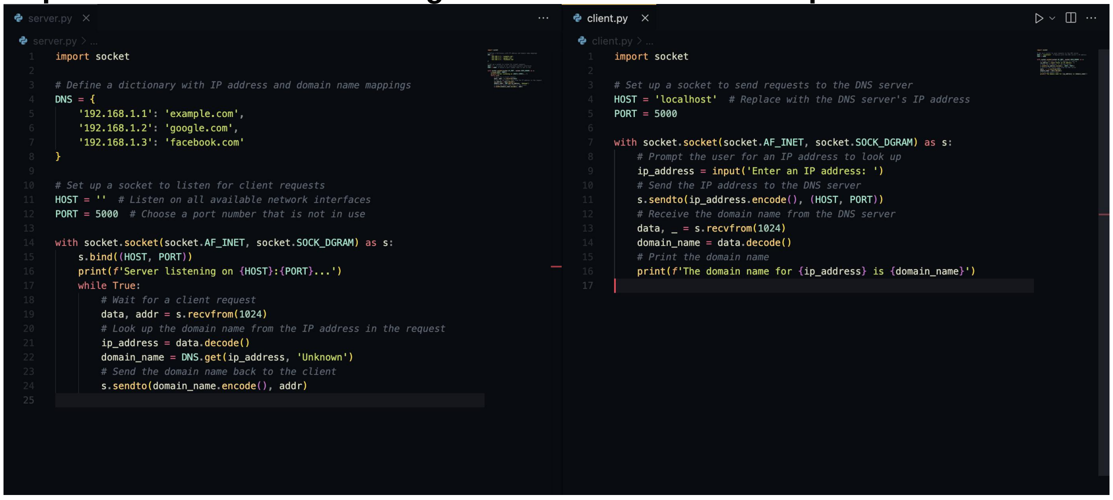
Output:



```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 server.py
Server is listening on localhost:8000
Connected by ('127.0.0.1', 64430)
```

```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 client.py
Enter the filename: hello.txt
File received: hello.txt
Enter the filename:
```

6. Write a Socket based program server-client to simulate DNS Service where client
request for Domain name using IP address and server responds with the Name.



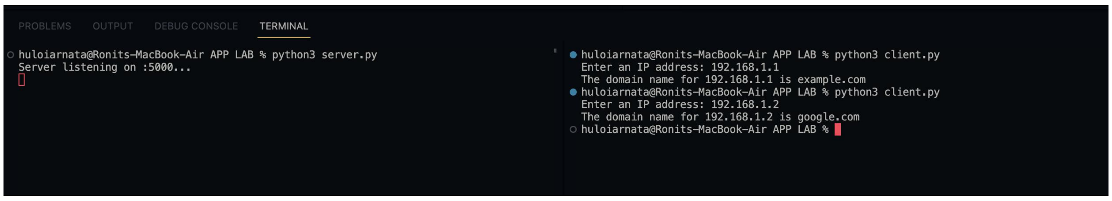
server.py

```python
import socket

# Define a dictionary with IP address and domain name mappings
DNS = {
    '192.168.1.1': 'example.com',
    '192.168.1.2': 'google.com',
    '192.168.1.3': 'facebook.com'
}

# Set up a socket to listen for client requests
HOST = ''  # Listen on all available network interfaces
PORT = 5000  # Choose a port number that is not in use

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    s.bind((HOST, PORT))
    print(f'Server listening on {HOST}:{PORT}...')
    while True:
        # Wait for a client request
        data, addr = s.recvfrom(1024)
        # Look up the domain name from the IP address in the request
        ip_address = data.decode()
        domain_name = DNS.get(ip_address, 'Unknown')
        # Send the domain name back to the client
        s.sendto(domain_name.encode(), addr)
```

client.py

```python
import socket

# Set up a socket to send requests to the DNS server
HOST = 'localhost'  # Replace with the DNS server's IP address
PORT = 5000

with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
    # Prompt the user for an IP address to look up
    ip_address = input('Enter an IP address: ')
    # Send the IP address to the DNS server
    s.sendto(ip_address.encode(), (HOST, PORT))
    # Receive the domain name from the DNS server
    data, _ = s.recvfrom(1024)
    domain_name = data.decode()
    # Print the domain name
    print(f'The domain name for {ip_address} is {domain_name}')
```

Output:



```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 server.py
Server listening on :5000...
```

```
huloiarnata@Ronits-MacBook-Air APP LAB % python3 client.py
Enter an IP address: 192.168.1.1
The domain name for 192.168.1.1 is example.com
huloiarnata@Ronits-MacBook-Air APP LAB % python3 client.py
Enter an IP address: 192.168.1.2
The domain name for 192.168.1.2 is google.com
huloiarnata@Ronits-MacBook-Air APP LAB %
```