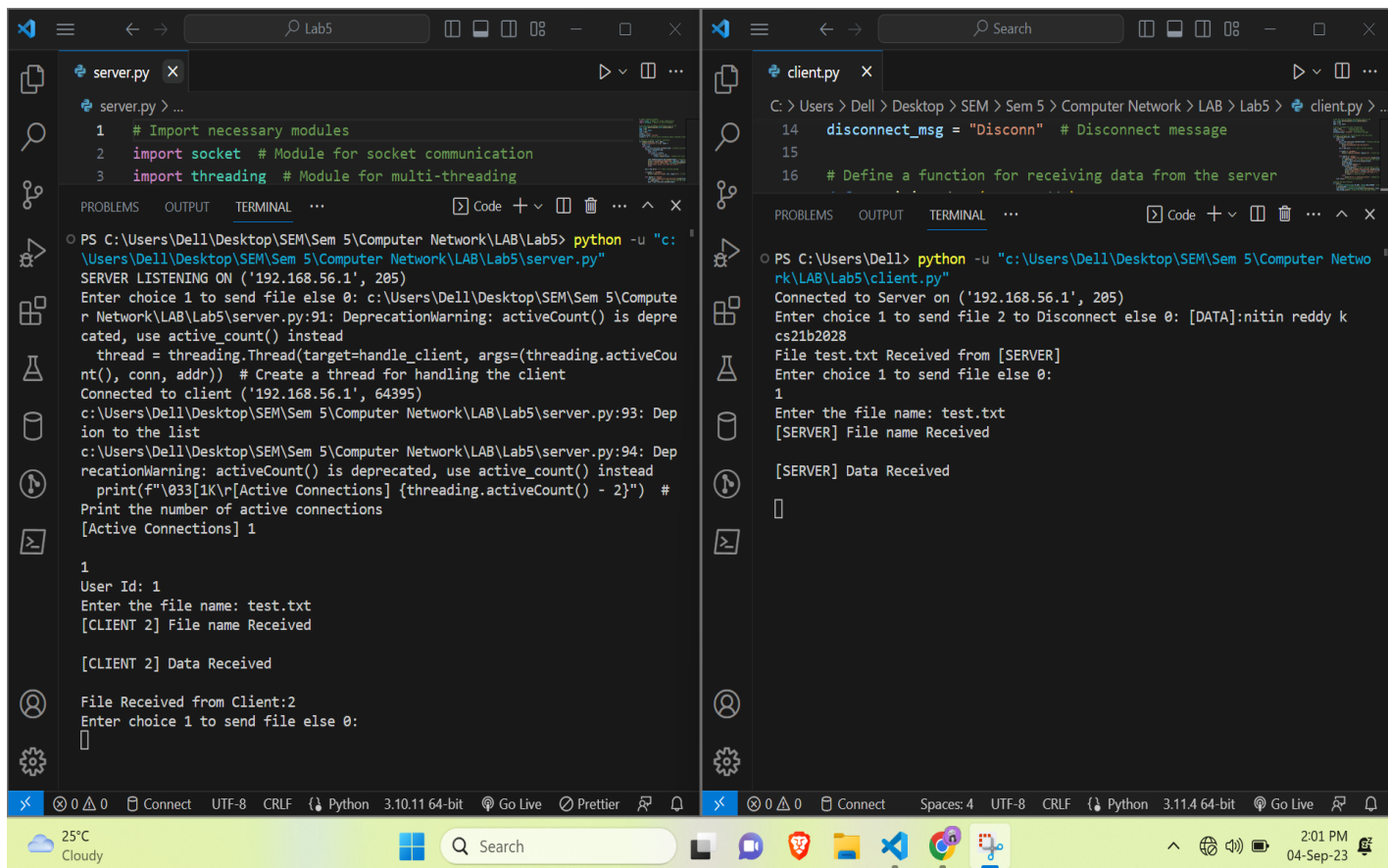


NITIN REDDY K

CS21B2028

LAB 5:

Terminal After Execution:



The image shows two side-by-side terminal windows in Visual Studio Code. The left window is titled 'server.py' and the right window is titled 'client.py'. Both windows show the execution of a Python script that implements a simple network communication protocol.

Left Window (server.py):

```
1 # Import necessary modules
2 import socket # Module for socket communication
3 import threading # Module for multi-threading

PS C:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5> python -u "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5\server.py"
SERVER LISTENING ON ('192.168.56.1', 205)
Enter choice 1 to send file else 0: c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5\server.py:91: DeprecationWarning: activeCount() is deprecated, use active_count() instead
  thread = threading.Thread(target=handle_client, args=(threading.activeCount(), conn, addr)) # Create a thread for handling the client
Connected to client ('192.168.56.1', 64395)
c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5\server.py:93: DeprecationWarning: activeCount() is deprecated, use active_count() instead
  print(f"\033[1K\r[Active Connections] {threading.activeCount() - 2}") # Print the number of active connections
[Active Connections] 1

1
User Id: 1
Enter the file name: test.txt
[CLIENT 2] File name Received

[CLIENT 2] Data Received

File Received from Client:2
Enter choice 1 to send file else 0:
0
```

Right Window (client.py):

```
14 disconnect_msg = "Disconn" # Disconnect message
15
16 # Define a function for receiving data from the server

PS C:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5> python -u "c:\Users\Dell\Desktop\SEM\Sem 5\Computer Network\LAB\Lab5\client.py"
Connected to Server on ('192.168.56.1', 205)
Enter choice 1 to send file 2 to Disconnect else 0: [DATA]:nitin reddy k cs21b2028
File test.txt Received from [SERVER]
Enter choice 1 to send file else 0:
1
Enter the file name: test.txt
[SERVER] File name Received

[SERVER] Data Received

0
```

Client Code:

```
client.py
C:\Users> Dell > Desktop > SEM > Sem 5 > Computer Network > LAB > Lab5 > client.py > ...
1 # Import the necessary modules
2 import socket # Module for socket communication
3 import threading # Module for multi-threading
4 import time # Module for handling time
5
6 # Get the local IP address and define a port to use
7 ip = socket.gethostbyname(socket.gethostname())
8 port = 205
9 addr = (ip, port)
10
11 # Define constants for data transmission
12 size = 1024 # Maximum data size
13 format = 'utf-8' # Data encoding format
14 disconnect_msg = "Disconn" # Disconnect message
15
16 # Define a function for receiving data from the server
17 def receiving_data(conn, addr):
18     flag = True
19     while flag:
20         msg = conn.recv(size).decode(format) # Receive and decode data
21         if msg == 'Disconn':
22             print("Disconnected From the Server")
23             break
24         msg = msg.split(':') # Split received message
25
26         if msg[0] == "[ACKNOW]":
27             print(f"033[1K\r[SERVER] {msg[1]}\n") # Print server acknowledgment message
28
29         elif msg[0] == "[FILE]":
30             conn.send("[ACKNOW]:File name Received".encode(format)) # Send acknowledgment to server
31             f = open(msg[1], 'w') # Open a file with the received filename
32             ed_data = conn.recv(size).decode(format)
33             print(ed_data)
34             data = ed_data.split(':')
35             if data[0] == '[DATA]':
36                 f.write(data[1]) # Write received data to the file
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
client.py
C:\Users> Dell > Desktop > SEM > Sem 5 > Computer Network > LAB > Lab5 > client.py > ...
37
38     conn.send("[ACKNOW]:Data Received".encode(format)) # Send acknowledgment to server
39     print(f"033[1K\rFile {msg[1]} Received from [SERVER]")
40     print("Enter choice 1 to send file else 0: ")
41     f.close() # Close the file
42
43     else:
44         print("Error in receiving file\n") # Handle error in receiving file
45     conn.close() # Close the connection
46
47 # Define the main function
48 def main():
49     c = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Create a socket object
50     c.connect(addr) # Connect to the server
51     rcv_thread = threading.Thread(target=receiving_data, args=(c, addr)) # Create a thread for receiving data
52     rcv_thread.start() # Start the receiving data thread
53     print(f"Connected to Server on {addr}")
54     while True:
55         k = int(input('Enter choice 1 to send file 2 to Disconnect else 0: ')) # Get user input
56         if k == 1:
57             file_name = input("Enter the file name: ") # Prompt for the file name
58             try:
59                 f = open(file_name, 'r') # Open the file for reading
60                 ed_file_name = '[FILE]:' + file_name
61                 c.send(ed_file_name.encode(format)) # Send the file name to the server
62                 data = f.read() # Read the file contents
63                 data = '[DATA]:' + data
64                 c.send(data.encode(format)) # Send the file data to the server
65                 f.close() # Close the file
66             except FileNotFoundError:
67                 print("File Not Found\n") # Handle file not found error
68         elif k == 2:
69             c.send("Disconn".encode(format)) # Send a disconnect message to the server
70             break # Exit the loop and close the program
71
72 # Check if the script is run as the main program
73 if __name__ == '__main__':
74     main() # Call the main function
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Server Code:

```
File Edit Selection View Go ... Lab5
server.py x
server.py > ...
1 # Import necessary modules
2 import socket # Module for socket communication
3 import threading # Module for multi-threading
4
5 # Get the local IP address and define a port to use
6 IP = socket.gethostname(socket.gethostname())
7 PORT = 205
8 ADDR = (IP, PORT)
9 size = 1024
10 format = 'utf-8'
11 disconnect_msg = "Disconn"
12 clients = [] # List to store information about connected clients
13
14 # Function to handle a client's connection
15 def handle_client(id, conn, addr):
16     print(f"connected to client {addr}")
17     flag = True
18     while flag:
19         msg = conn.recv(size).decode(format) # Receive and decode data from the client
20         if msg == disconnect_msg:
21             flag = False
22             for client in clients:
23                 if client['id'] == id:
24                     clients.remove(client) # Remove the client from the list when disconnected
25
26         conn.send(disconnect_msg.encode(format)) # Send disconnect acknowledgment to the client
27         print(f"[DISCONNECTED CONNECTION] {addr[0]} {addr[1]} USER {id} is disconnected from the server")
28         print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 2}") # Count active connections
29         print("Enter choice 1 to send file else 0: ")
30         break
31
32     msg = msg.split(':') # Split received message
33
34     if msg[0] == "[ACKNOW]":
35         print(f"\033[1K\r[CLIENT {id}] {msg[1]}\n") # Print client acknowledgment message
36
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit Go Live Prettier
```

```
File Edit Selection View Go ... Lab5
server.py x
server.py > ...
36
37 elif msg[0] == "[FILE]":
38     conn.send("[ACKNOW]:File name Received".encode(format)) # Send acknowledgment to the client
39     f = open(msg[1], 'w') # Open a file with the received filename for writing
40     data = conn.recv(size).decode(format)
41     data = data.split(':')
42     if data[0] == '[DATA]':
43         f.write(data[1]) # Write received data to the file
44         conn.send("[ACKNOW]:Data Received".encode(format)) # Send acknowledgment to the client
45         print(f"\033[1K\rFile Received from Client:{id}")
46         print("Enter choice 1 to send file else 0: ")
47     else:
48         print("Error in receiving file\n") # Handle error in receiving file
49
50     f.close() # Close the file
51
52     conn.close() # Close the connection when done
53
54 # Function to handle sending data to clients
55 def handle_send_data():
56     while True:
57         k = int(input('Enter choice 1 to send file else 0: ')) # Get user input
58         if k == 1:
59             x = int(input("User Id: ")) # Get the user ID
60             for client in clients:
61                 if client['id'] == x:
62                     c = client['conn'] # Get the client's connection
63                     a = client['addr'] # Get the client's address
64                     break
65             else:
66                 print("User does not exist")
67                 x = -1 # Set x to -1 if the user does not exist
68             if x != -1:
69                 file_name = input("Enter the file name: ") # Prompt for the file name
70                 try:
71                     f = open(file_name, 'r') # Open the file for reading
```

```
File Edit Selection View Go ... Lab5
server.py x
server.py > ...
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

try:
    f = open(file_name, 'r') # Open the file for reading
    ed_file_name = '[FILE]:' + file_name
    c.send(ed_file_name.encode(format)) # Send the file name to the client
    data = f.read() # Read the file contents
    ed_data = '[DATA]:' + data
    c.send(ed_data.encode(format)) # Send the file data to the client
    f.close() # Close the file
except FileNotFoundError:
    print("File Not Found\n") # Handle file not found error

# Main function
def main():
    print(f"SERVER LISTENING ON {ADDR}")
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # Create a socket object
    s.bind(ADDR) # Bind the socket to the server address
    s.listen() # Listen for incoming connections
    send_thread = threading.Thread(target=handle_send_data) # Create a thread for sending data
    send_thread.start() # Start the sending data thread
    while True:
        conn, addr = s.accept() # Accept a client connection
        thread = threading.Thread(target=handle_client, args=(threading.activeCount(), conn, addr)) # Create a thread for handling the client
        thread.start() # Start the client handling thread
        clients.append({'id': threading.activeCount() - 2, 'conn': conn, 'addr': addr}) # Add client information to the list
        print(f"\033[1K\r[Active Connections] {threading.activeCount() - 2}") # Print the number of active connections
        print('')
    # Check if the script is run as the main program
    if __name__ == "__main__":
        main() # Call the main function
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit Go Live Prettier

25°C Cloudy Search 2:04 PM 04-Sep-23