Title and Authors:

Programming Project Phase 1: File Transfer between a UDP Client and a UDP Server

Submitted by: Luis D. Pena Mateo

Purpose of The Phase

Project description: The TCP/IP stack has five layers, namely application, transport, network, link, and physical. The phase 1 is divided into two parts.

Phase 1(a):

In this part of the project, each student must individually implement the standard user datagram protocol (UDP) sockets. The intention is to transfer a message (Say "HELLO") from the UDP client to the UDP server and then ECHO the message back from the UDP server to the UDP client. Note that the client and server process can reside on the same host but have to use different port numbers. Make sure that your program can send and receive messages in both directions.

Phase 1(b):

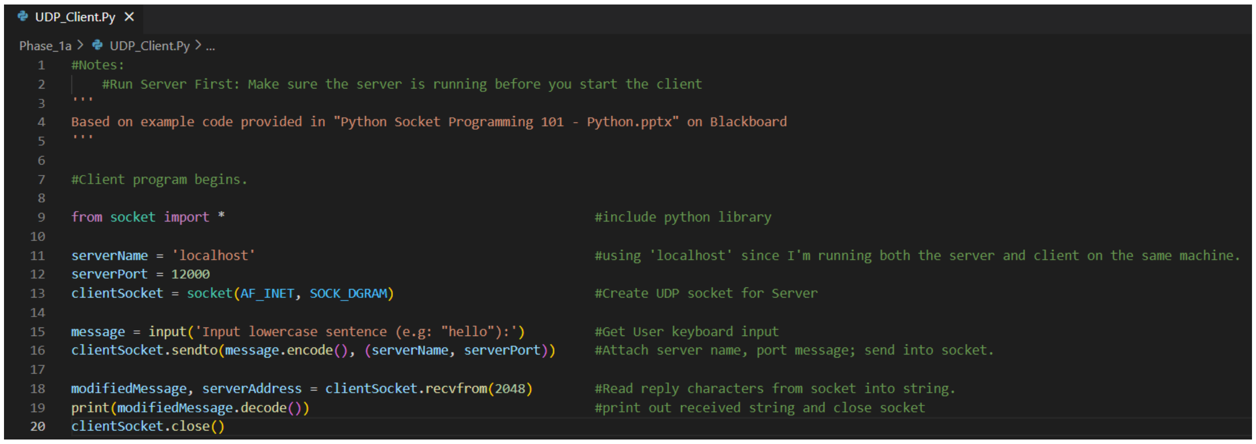
In the second part of this phase, the intention is to transfer a file (say BMP) between a UDP client process and a UDP server process. Each student must implement the RDT 1.0 protocol described in Section 3.4.1 of the course textbook.

Code Explanation

Phase 1(a):

List of Files:

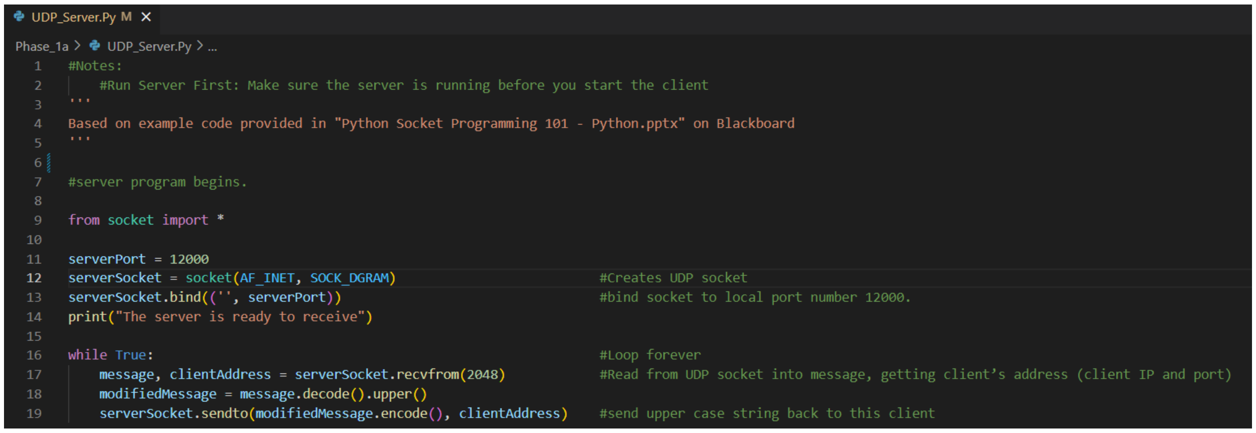
- **UDP_Client.py**: This script implements the UDP client for Phase 1(a). It sends a message to the UDP server and receives the echoed message back.



```
#Notes:
    #Run Server First: Make sure the server is running before you start the client
'''
Based on example code provided in "Python Socket Programming 101 - Python.pptx" on Blackboard
'''


#Client program begins.

from socket import *                                            #include python library

serverName = 'localhost'                                        #using 'localhost' since I'm running both the server and client on the same machine.
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)                      #Create UDP socket for Server

message = input('Input lowercase sentence (e.g: "hello"):')     #Get User keyboard input
clientSocket.sendto(message.encode(), (serverName, serverPort)) #Attach server name, port message; send into socket.

modifiedMessage, serverAddress = clientSocket.recvfrom(2048)    #Read reply characters from socket into string.
print(modifiedMessage.decode())                                 #print out received string and close socket
clientSocket.close()
```

Figure 1: Client Source Code. phase 1(a)

- **UDP_Server.py:** This script implements the UDP server for Phase 1(a). It receives a message from the UDP client, converts it to uppercase, and sends it back to the client.



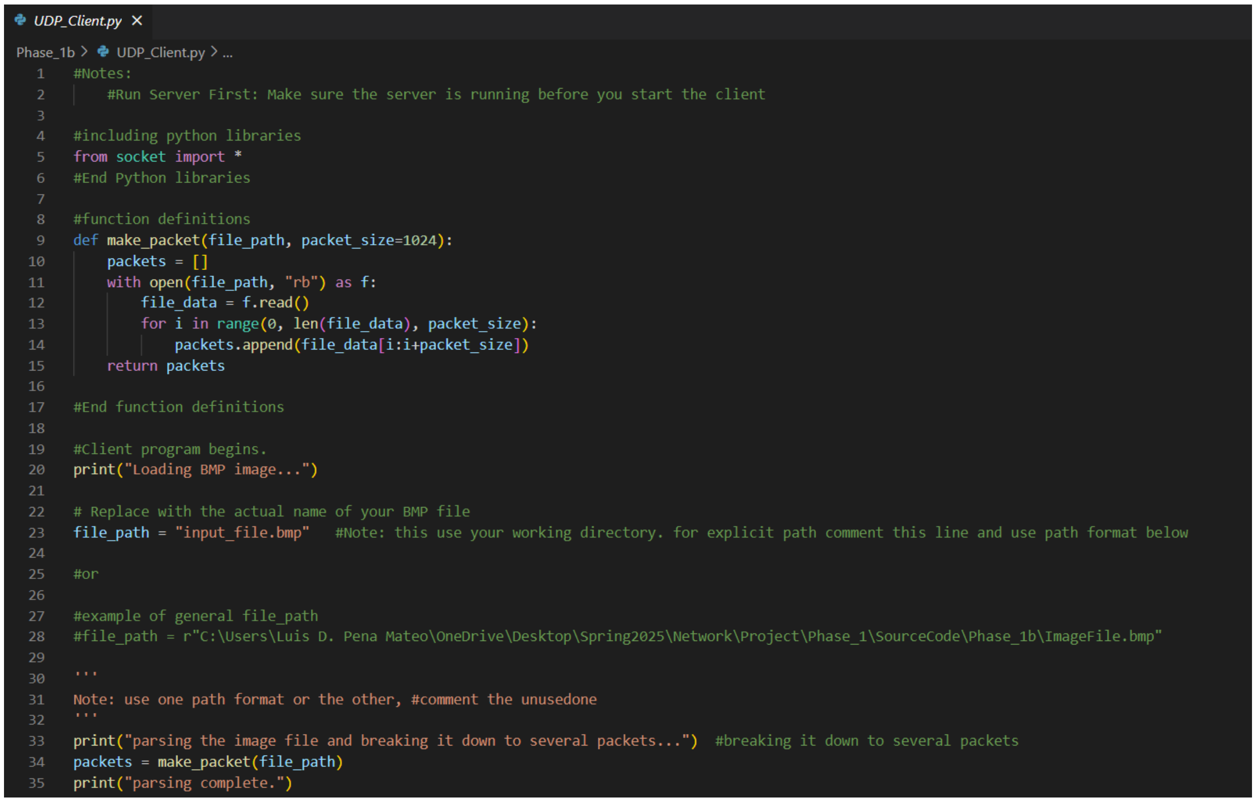
```python
#Notes:
    #Run Server First: Make sure the server is running before you start the client
'''
Based on example code provided in "Python Socket Programming 101 - Python.pptx" on Blackboard
'''

#server program begins.

from socket import *

serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)                         #Creates UDP socket
serverSocket.bind(('', serverPort))                                #bind socket to local port number 12000.
print("The server is ready to receive")

while True:                                                        #Loop forever
    message, clientAddress = serverSocket.recvfrom(2048)           #Read from UDP socket into message, getting client’s address (client IP and port)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)   #send upper case string back to this client
```

Figure 2: Server Source Code. phase 1(a)

Phase 1(b):

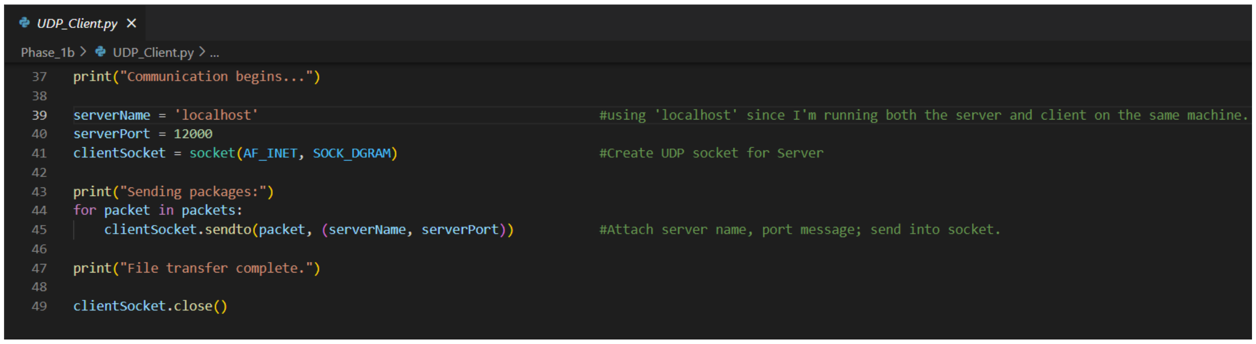
List of Files:

- **UDP_Client.py:** This script implements the UDP client for Phase 1(b). It sends a BMP file to the UDP server, one packet at a time.



```python
#Notes:
    #Run Server First: Make sure the server is running before you start the client

#including python libraries
from socket import *
#End Python libraries

#function definitions
def make_packet(file_path, packet_size=1024):
    packets = []
    with open(file_path, "rb") as f:
        file_data = f.read()
        for i in range(0, len(file_data), packet_size):
            packets.append(file_data[i:i+packet_size])
    return packets

#End function definitions

#Client program begins.
print("Loading BMP image...")

# Replace with the actual name of your BMP file
file_path = "input_file.bmp"   #Note: this use your working directory. for explicit path comment this line and use path format below

#or

#example of general file_path
#file_path = r"C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1b\ImageFile.bmp"

'''
Note: use one path format or the other, #comment the unusedone
'''
print("parsing the image file and breaking it down to several packets...")  #breaking it down to several packets
packets = make_packet(file_path)
print("parsing complete.")
```

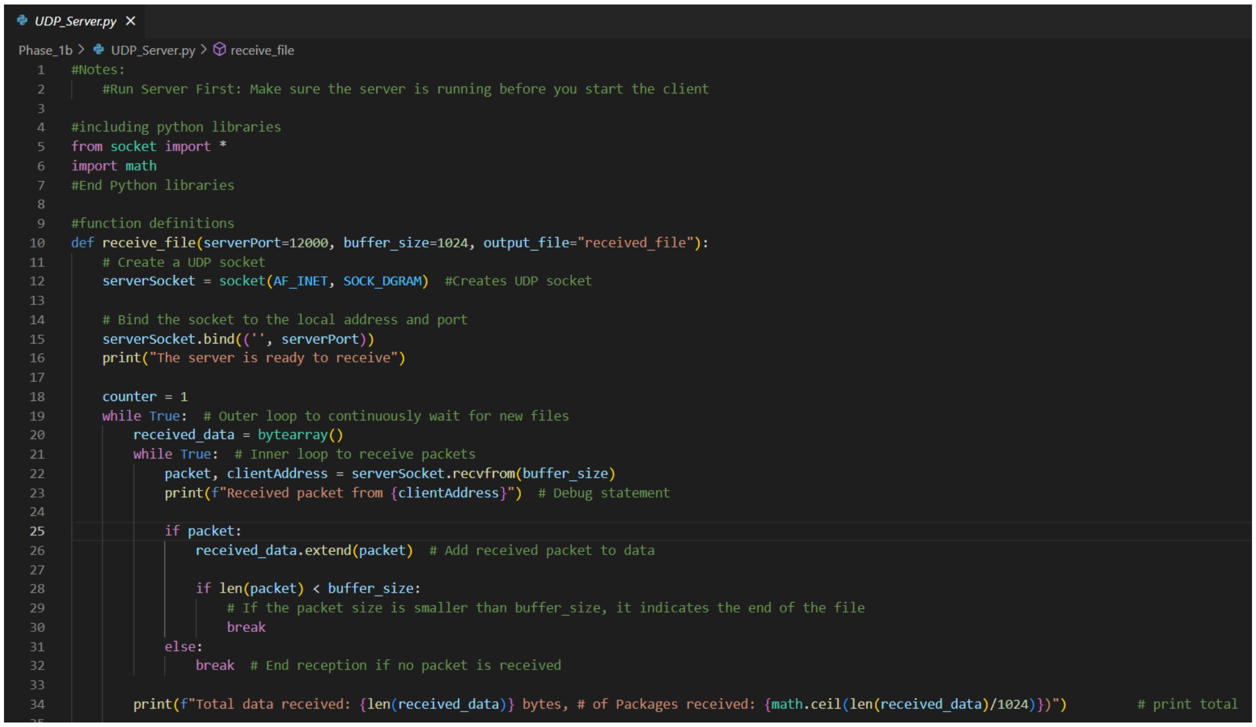
Figure 3: Source code Client. Phase 1(b), lines 1 to 35



```python
print("Communication begins...")

serverName = 'localhost'                                    #using 'localhost' since I'm running both the server and client on the same machine.
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)                  #Create UDP socket for Server

print("Sending packages:")
for packet in packets:
    clientSocket.sendto(packet, (serverName, serverPort))           #Attach server name, port message; send into socket.

print("File transfer complete.")

clientSocket.close()
```

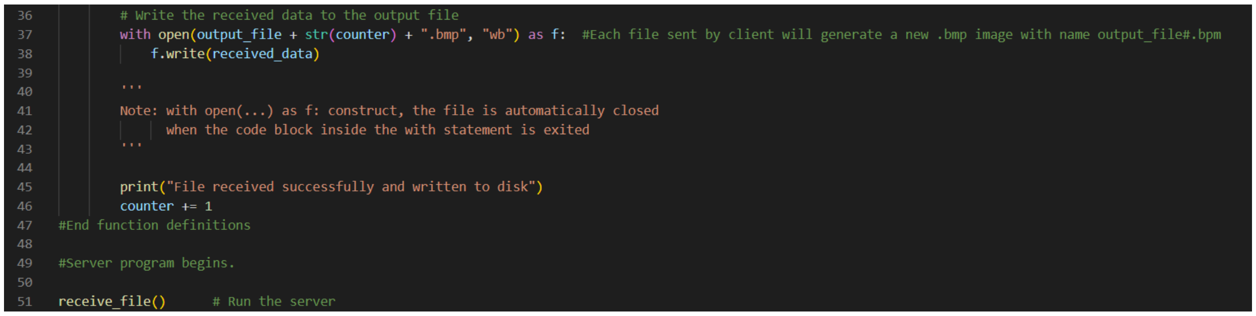
Figure 4: Continuation of Source code Client. Phase 1(b), lines 37 to 49

- **UDP_Server.py:** This script implements the UDP server for Phase 1(b). It receives BMP file packets from the UDP client and then writes data received into a new file.



```python
#Notes:
    #Run Server First: Make sure the server is running before you start the client

#including python libraries
from socket import *
import math
#End Python libraries

#function definitions
def receive_file(serverPort=12000, buffer_size=1024, output_file="received_file"):
    # Create a UDP socket
    serverSocket = socket(AF_INET, SOCK_DGRAM)  #Creates UDP socket

    # Bind the socket to the local address and port
    serverSocket.bind(('', serverPort))
    print("The server is ready to receive")

    counter = 1
    while True:  # Outer loop to continuously wait for new files
        received_data = bytearray()
        while True:  # Inner loop to receive packets
            packet, clientAddress = serverSocket.recvfrom(buffer_size)
            print(f"Received packet from {clientAddress}")  # Debug statement

            if packet:
                received_data.extend(packet)  # Add received packet to data

                if len(packet) < buffer_size:
                    # If the packet size is smaller than buffer_size, it indicates the end of the file
                    break
            else:
                break  # End reception if no packet is received

        print(f"Total data received: {len(received_data)} bytes, # of Packages received: {math.ceil(len(received_data)/1024)})")          # print total
```

Figure 5: Source code Server. Phase 1(b), lines 1 to 35



```python
        # Write the received data to the output file
        with open(output_file + str(counter) + ".bmp", "wb") as f:  #Each file sent by client will generate a new .bmp image with name output_file#.bpm
            f.write(received_data)

        '''
        Note: with open(...) as f: construct, the file is automatically closed
              when the code block inside the with statement is exited
        '''

        print("File received successfully and written to disk")
        counter += 1
#End function definitions

#Server program begins.

receive_file()      # Run the server
```

Figure 6: Source code Client. Phase 1(b), lines 36 to 51

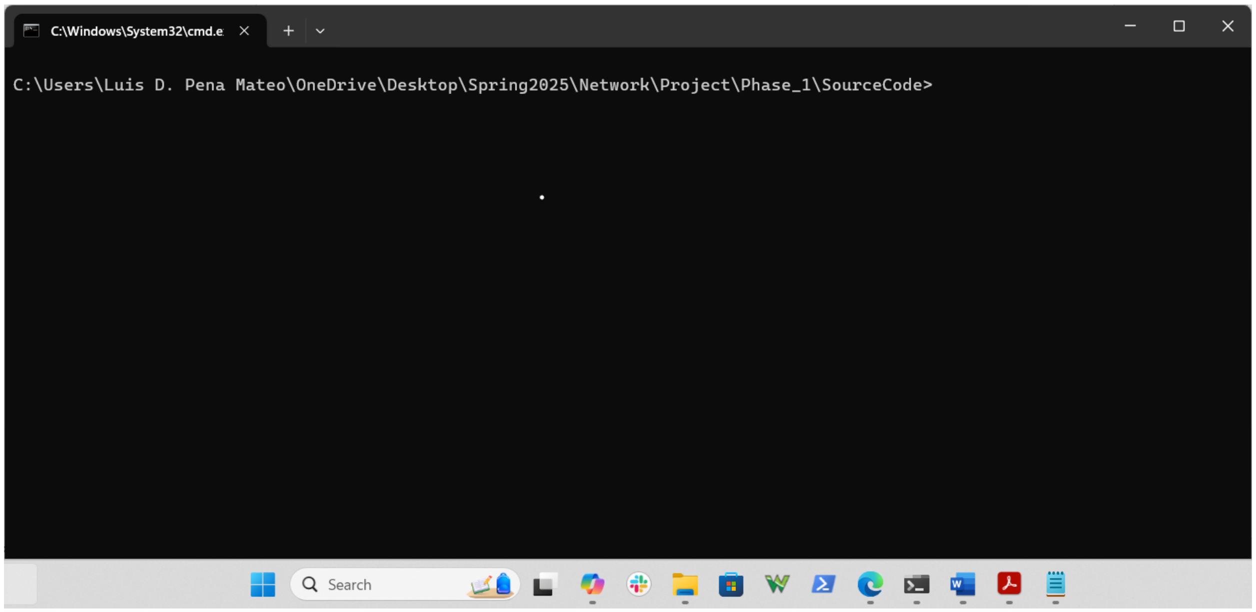
- **Input_file:** This is a BMP file to test the code (Phase_1b)

Figure 7: BMP file trasfered Using client and server python scripts presented above.

Execution Example

Instructions to Set Up and Execute Phase 1(a): Message Transfer

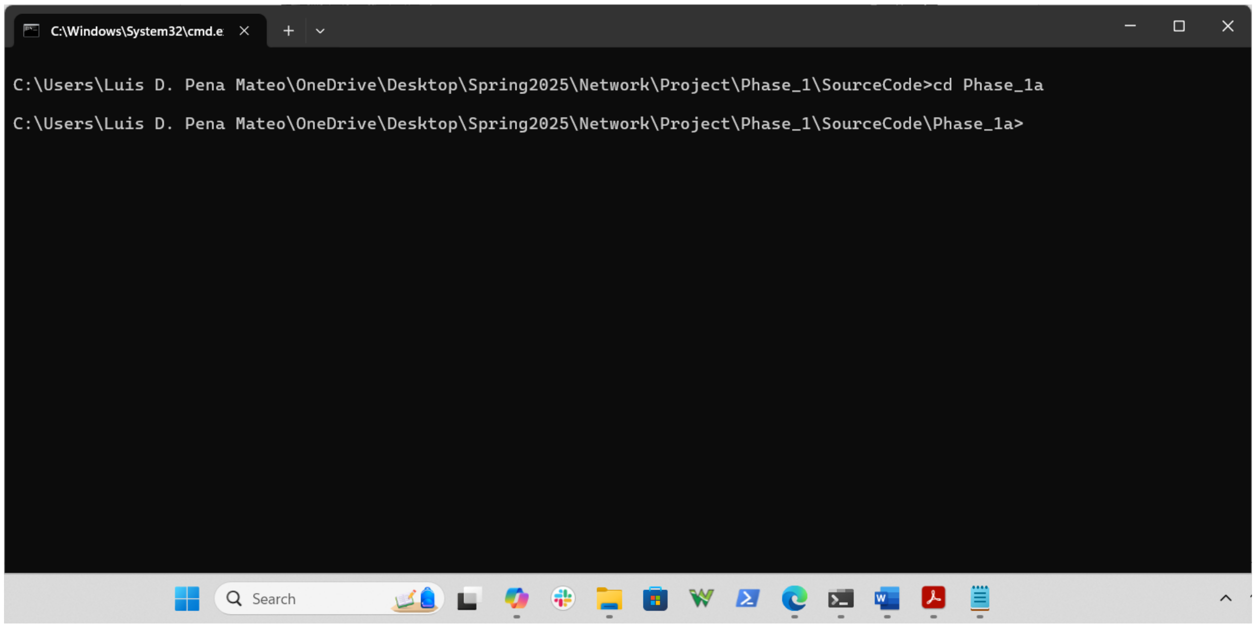
- **Run the Server:**
    1) Open a terminal or command prompt.



```
C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode>
```

Figure 8: opening terminal

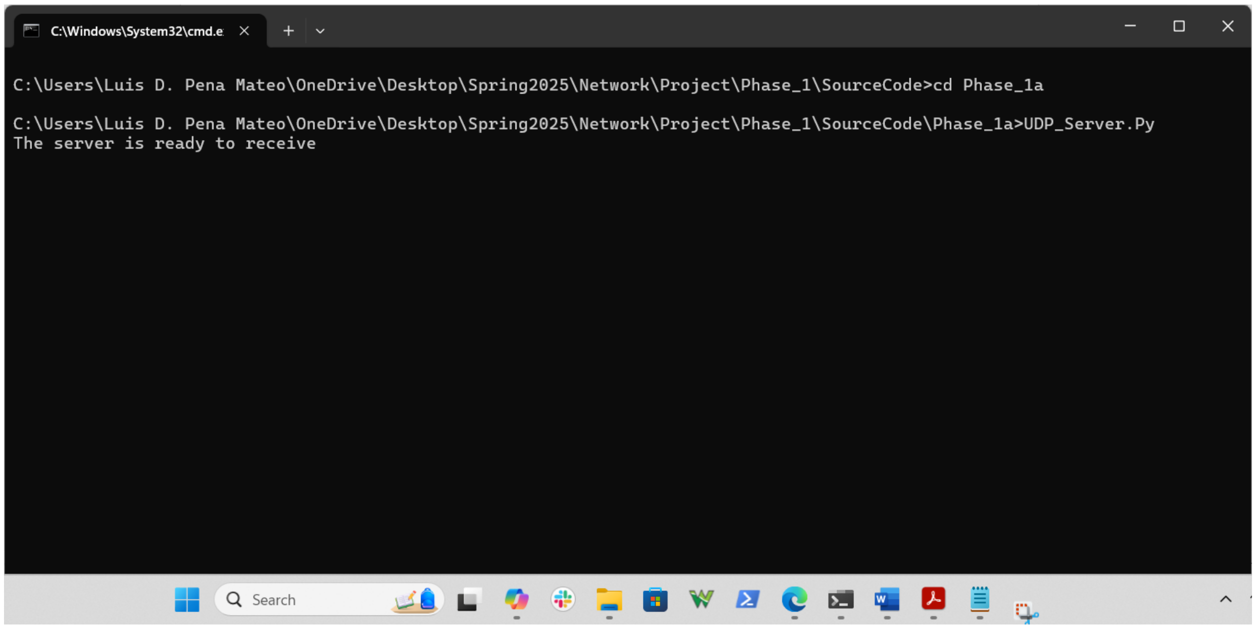
2) Navigate to the folder Phase_1a directory containing UDP_Server.py



```
C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode>cd Phase_1a

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>
```

Figure 9: Navigating to Phase_1a directory

3) Execute the server script by running python UDP_Server.py

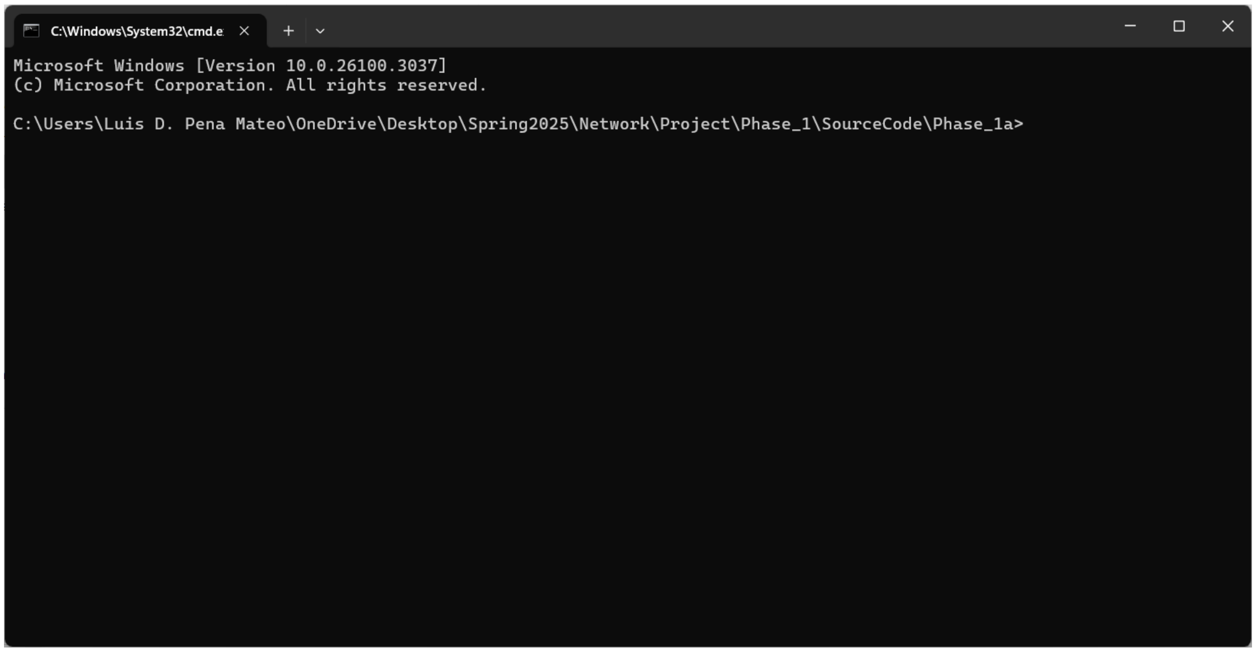


```
C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode>cd Phase_1a

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>UDP_Server.Py
The server is ready to receive
```

Figure 10: Executing Server.py script. Server is Ready to Receive a file now.

4) The server will bind to port 12000 and wait for incoming messages.

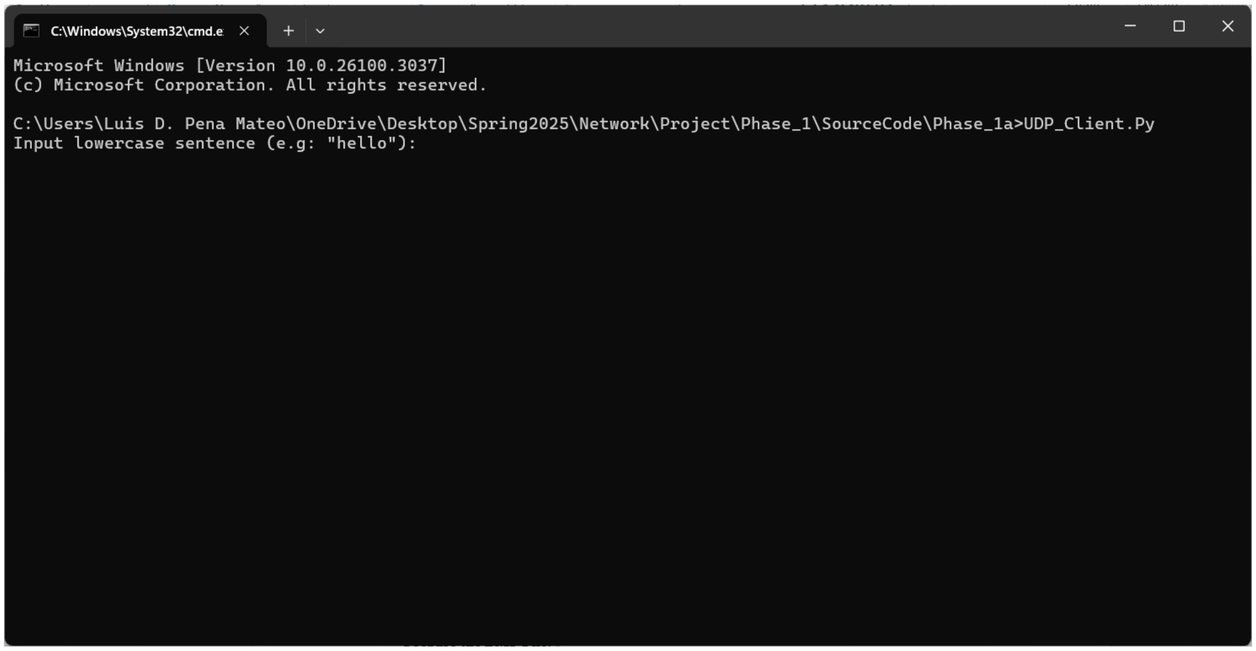
- **Run the Client:**
  1) Open another terminal or command prompt.



```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>
```

Figure 11: Opening another Terminal in the Phase_1a directory

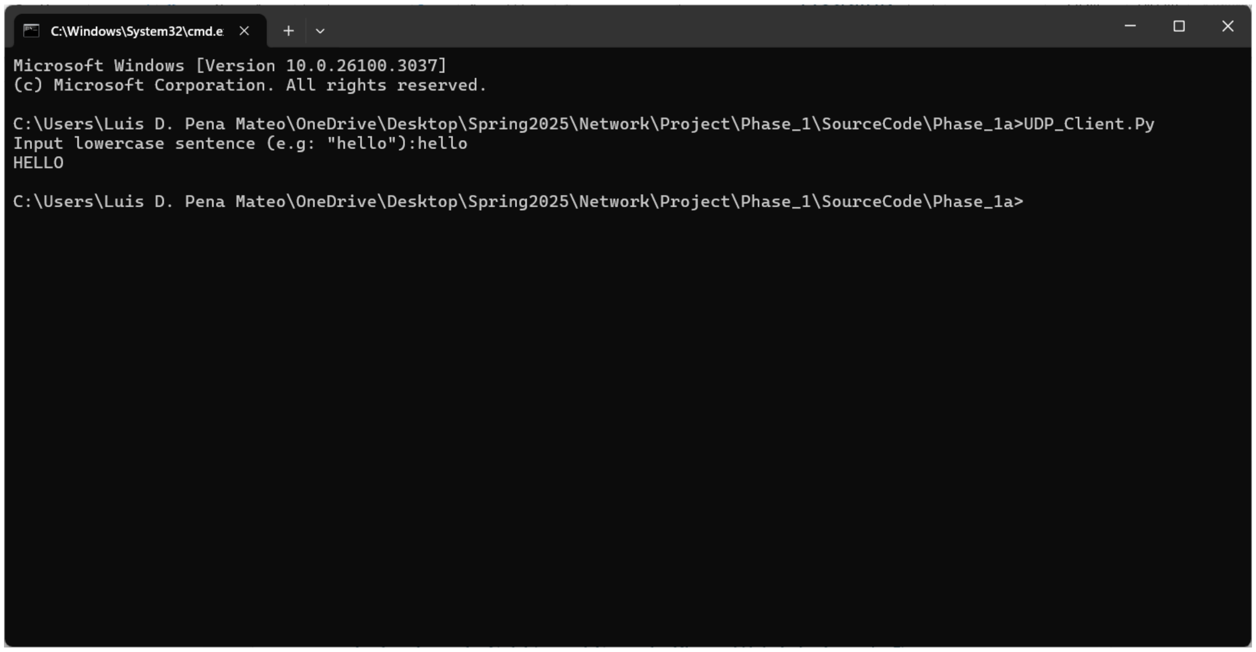
2) Navigate to the folder Phase_1a directory containing UDP_Client.py
3) Execute the client script by running python UDP_Client.py



```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>UDP_Client.Py
Input lowercase sentence (e.g: "hello"):
```

Figure 12: Executing Client script

4) Enter a lowercase sentence when prompted. e.g.: "hello"
5) The client will send the message to the server, receive the echoed message, and print it in upper case.

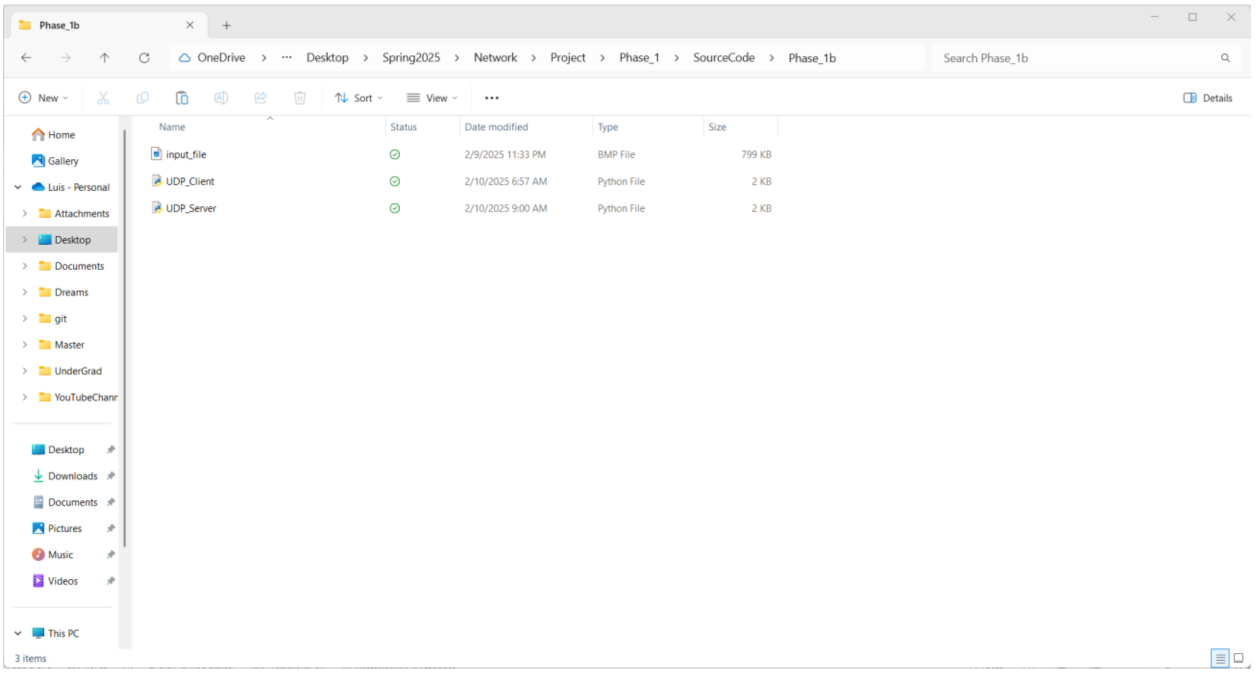


```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>UDP_Client.Py
Input lowercase sentence (e.g: "hello"):hello
HELLO

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1a>
```

Figure 13: After typing hello (lowercase) in client, Client received Server Echo HELLO response (Upper case)

Instructions to Set Up and Execute Phase 1(b): File Transfer

- **Prepare the BMP File:**
    1) Place the BMP file (input_file.bpm) to be transferred in the same directory as UDP_Client.py (Phase_1b)

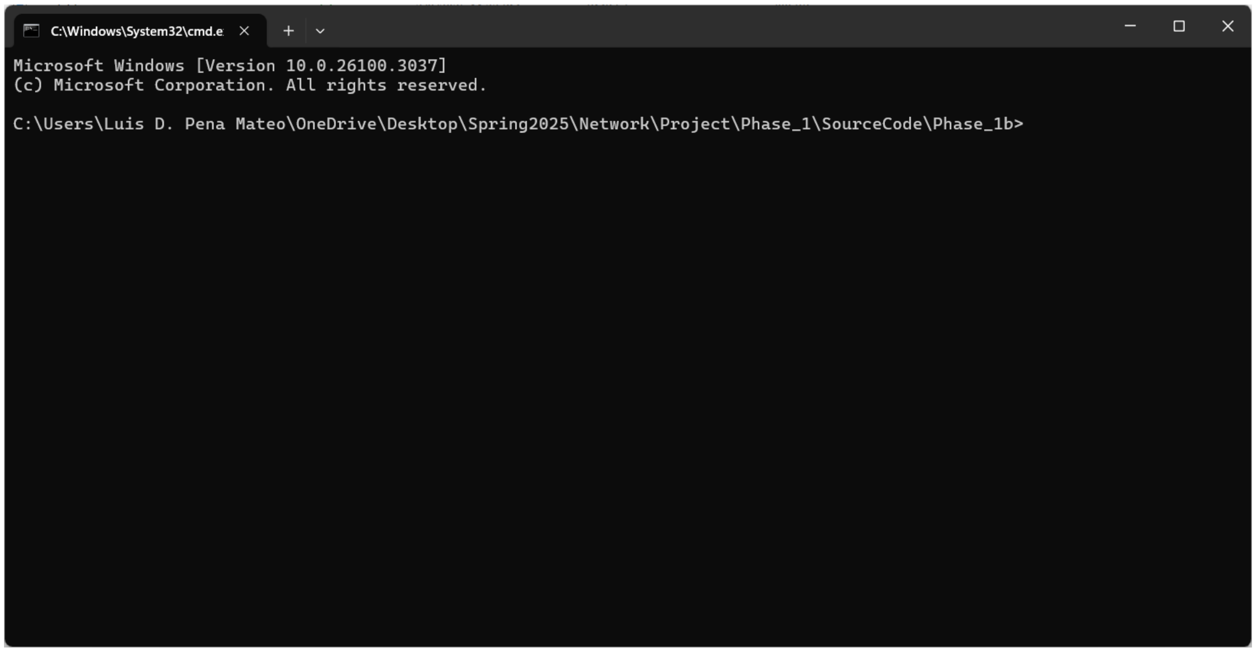


Name	Status	Date modified	Type	Size
input_file	✓	2/9/2025 11:33 PM	BMP File	799 KB
UDP_Client	✓	2/10/2025 6:57 AM	Python File	2 KB
UDP_Server	✓	2/10/2025 9:00 AM	Python File	2 KB

Figure 14: input_file.bmp in the same directory as client and server scripts.

    2) Or alternatively, you could also update the file_path variable in client_part1b.py to match the BMP file name or its explicit.

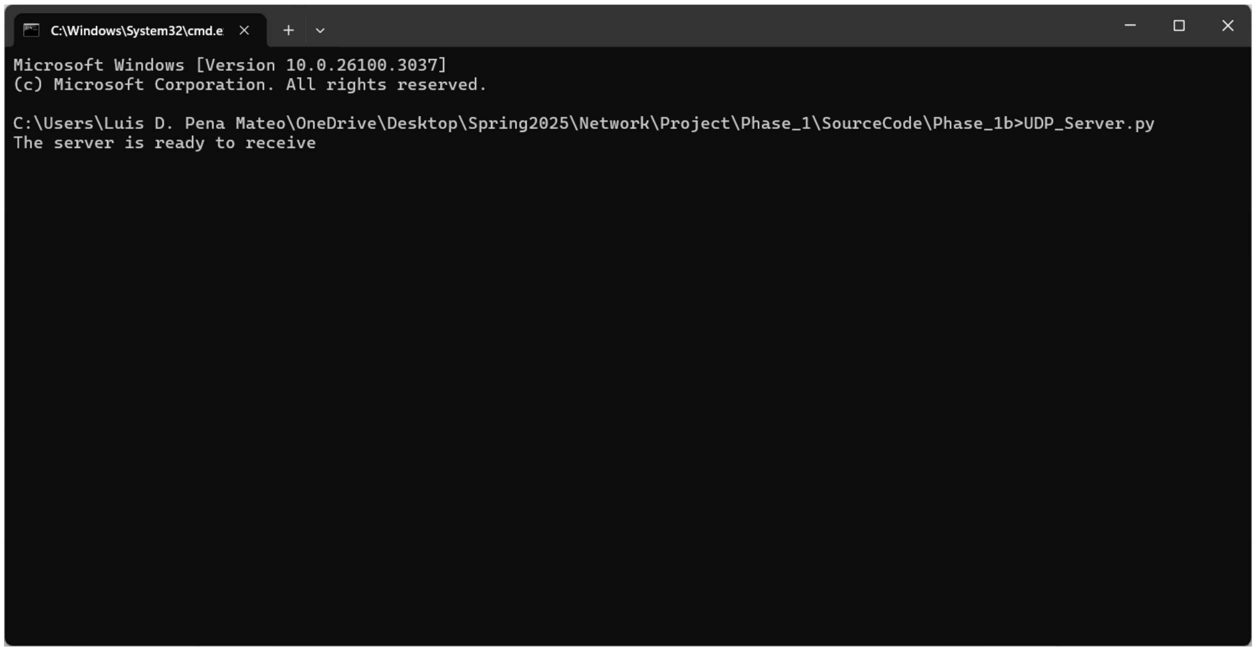
- **Run the Server:**
    1) Open a terminal or command prompt.
    2) Navigate to the Phase_1b directory containing UDP_Server.py



```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1b>
```

Figure 15: Opening terminal in Phase_1b directory.

3) Execute the server script by running python UDP_Server.py

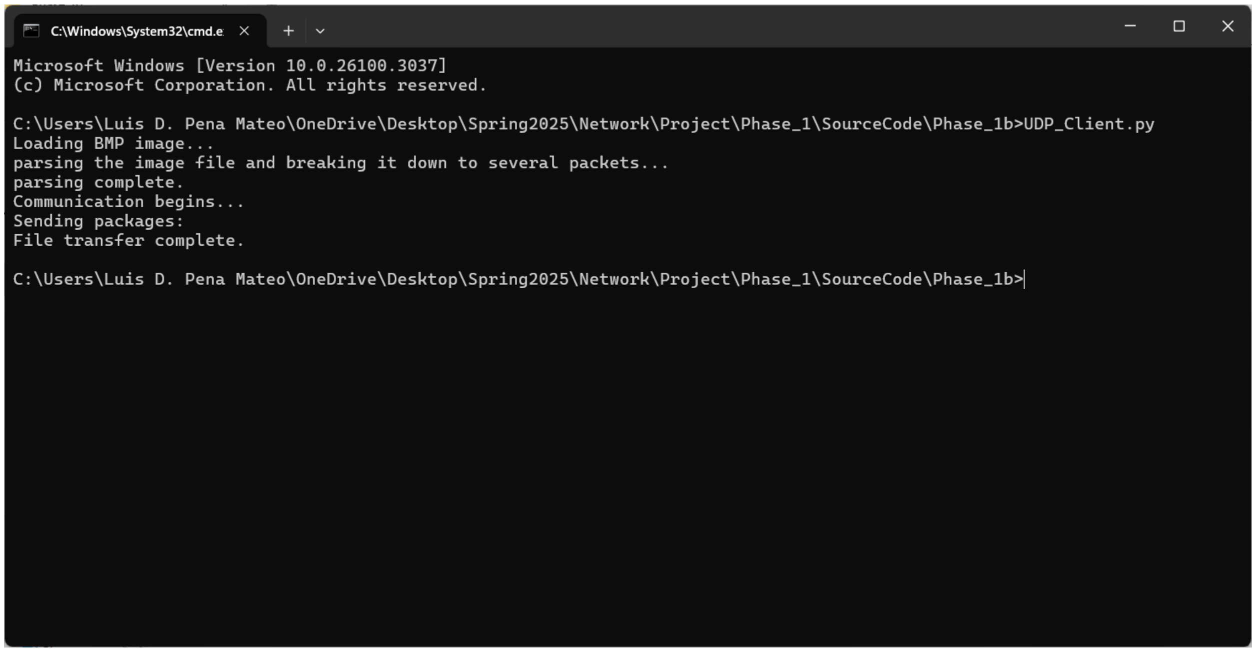


```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1b>UDP_Server.py
The server is ready to receive
```

Figure 16: Executing Server script.

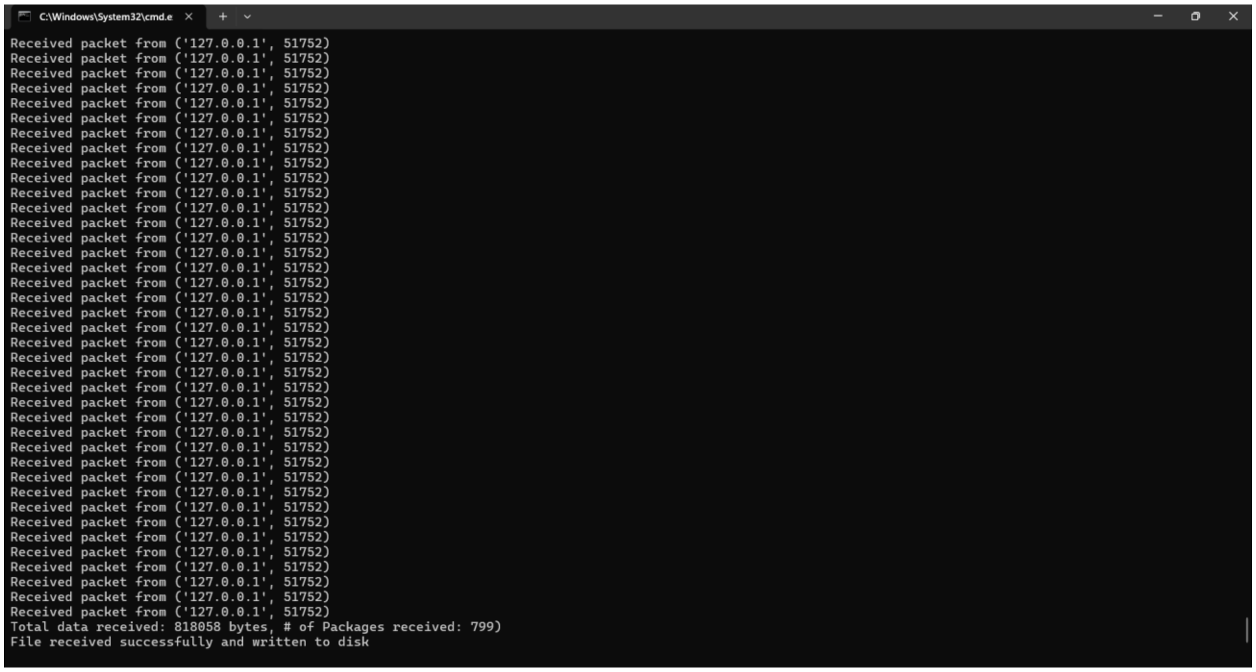
4) The server will bind to port 12000 and continuously wait for incoming file packets.

- **Run the Client:**
    1) Open another terminal or command prompt.
    2) Navigate to the directory Phase_1b containing UDP_Client.py
    3) Execute the client script by running python UDP_Client.py



```
Microsoft Windows [Version 10.0.26100.3037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1b>UDP_Client.py
Loading BMP image...
parsing the image file and breaking it down to several packets...
parsing complete.
Communication begins...
Sending packages:
File transfer complete.

C:\Users\Luis D. Pena Mateo\OneDrive\Desktop\Spring2025\Network\Project\Phase_1\SourceCode\Phase_1b>
```

Figure 17: After Executing client Script in a new terminal the following message is displayed.



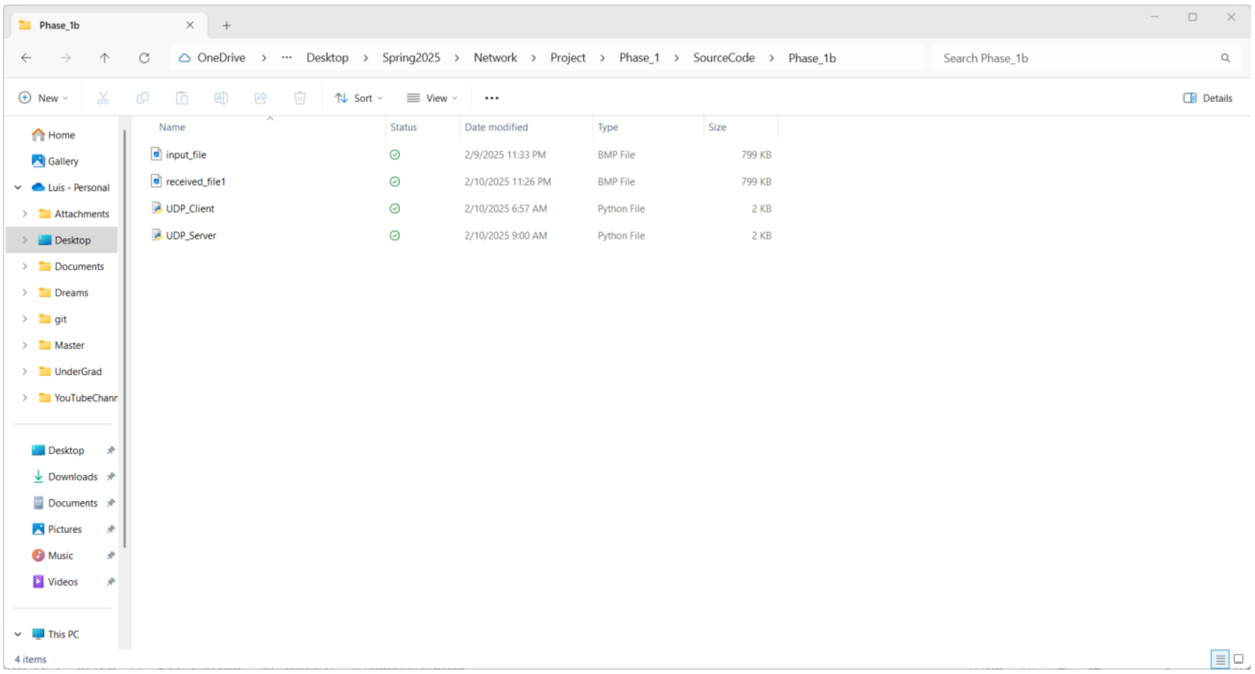
```
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Received packet from ('127.0.0.1', 51752)
Total data received: 818058 bytes, # of Packages received: 799)
File received successfully and written to disk
```

Figure 18:Server terminal generates the following messages confirming is receiving the packets from the file sent by the client.

4) The client will read the BMP file, break it into packets, and send them to the server.

- **Receive the File:**

1) The server will receive the file packets and write them to disk as received_file#.bmp in your python active directory.



Name	Status	Date modified	Type	Size
input_file	✓	2/9/2025 11:33 PM	BMP File	799 KB
received_file1	✓	2/10/2025 11:26 PM	BMP File	799 KB
UDP_Client	✓	2/10/2025 6:57 AM	Python File	2 KB
UDP_Server	✓	2/10/2025 9:00 AM	Python File	2 KB

Figure 19: File transferred successfully. A new file is created by the server with the data received by the client (received_file1.bmp)