Exercise 1

1. False
2. True
3. False
4. False

Exercise 2

1)

A. http://gaia.cs.umass.edu/cs453/index.html
B. HTTP/1.1
C. Persistent Connection - because “Keep-Alive”
D. The IP address is not present in the HTTP message
E. Mozilla/5.0. The browser type in an HTTP request helps servers tailor content for compatibility and optimize user experience. The server may have to send a different types of the same object to different browsers

2)

A. Yes, the server was able to find the document because we got a HTTP 200 OK. Date of Reply: : Tue, 07 Mar 2008 12:39:45 GMT
B. Last-Modified: Sat, 10 Dec2005 18:27:46 GMT
C. 3874 bytes
D. First 5 bytes: <!doc
   The server did agree to a persistent connection, since “Connection: Keep-Alive” indicates a persistent connection.

Exercise 3

A. MTA stands for Mail Transfer Agents. MTA is the mail transport service that is provided by clients and servers in SMTP.
B. The malicious host is the one with IP address 58.88.21.177 and email address inbnd55.exchangeddd.com. The host is disguising themselves as tennis5@pp33head.com, to prevent being caught.

Exercise 4

A.

1. dig +trace nyu.edu @a.root-servers.net

```
; <<>> DiG 9.10.6 <<>> +trace nyu.edu @a.root-servers.net
;; global options: +cmd
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      RRSIG   NS 8 0 518400 20241030050000 20241017040000 61050 . Q14j8GNv8yP4XW+
74i8CaXnNYKAqj8nl+gYjLqPnthOm0WMl4epLEh7g H2r689khiuYOBfP+9vip9OdLj5pmNJGopYX1dP40jTDYYkWl5cN+ejaf CI4fvlZKuOoOr16o
VNee/vLNIZ0iHwfsUnvQmLTJT59/Qn9KySXjjRgP +HevKORtHrjNsZuPQu70nSt95Son8Le0WkUU+ZK5Ox3wqdYsIyDs7Don 3XvA8Fr90cqYb0vUK
ELfnC4LkYIiO4GVczCJgWMDRDodn6QBYtlIQ15p Jv+uac4IubqPTCVqeY/ng0nIsSLGQ2oTSD/3eCYKw9u/uLt8DJ55etNl XDabyw==
;; Received 1097 bytes from 198.41.0.4#53(a.root-servers.net) in 18 ms


edu.                    172800  IN      NS      a.edu-servers.net.
edu.                    172800  IN      NS      b.edu-servers.net.
edu.                    172800  IN      NS      c.edu-servers.net.
edu.                    172800  IN      NS      d.edu-servers.net.
edu.                    172800  IN      NS      e.edu-servers.net.
edu.                    172800  IN      NS      f.edu-servers.net.
edu.                    172800  IN      NS      g.edu-servers.net.
edu.                    172800  IN      NS      h.edu-servers.net.
edu.                    172800  IN      NS      i.edu-servers.net.
edu.                    172800  IN      NS      j.edu-servers.net.
edu.                    172800  IN      NS      k.edu-servers.net.
edu.                    172800  IN      NS      l.edu-servers.net.
edu.                    172800  IN      NS      m.edu-servers.net.
edu.                    86400   IN      DS      35663 13 2 A2E1614291831A4746B5AC52B4B345357687271E85353082741F1CF3
 D06A4C1D
edu.                    86400   IN      RRSIG   DS 8 1 86400 20241030050000 20241017040000 61050 . db+EcHTgLhLm8d9h
LXmRUSorm7M45IrIw2rdAmo8hfi/tHs2LVbCRMHB tSmTrX7Yfc3fz6rxSrYdsZnjep26OTasswC6r1y2UwfefHvxvPJHQWrx 0rAcSLwiLB16hQfZ1
Ko7cFyyCxWSEN2TJXwR55kcy/T7ll4KnL8VTG33 G5DMUYW8uXSBVP/GzP18VSLvdeBjsog3aIKT6b1IXpZ0Gc0YRSnruu9V TI62JeqXDCAeG2nZQJ
oL2gTvS7vYvNZ6y3KvhgiPxiBLDATR7Qa6NxIO dDxrkzVdxlmz8difwAaJBSBHP1ED0XJwb01+xGkFD+royAo/W5DxdSAL OY8K2g==
;; Received 1166 bytes from 192.203.230.10#53(e.root-servers.net) in 18 ms


nyu.edu.                172800  IN      NS      ns1.nyu.net.
nyu.edu.                172800  IN      NS      ns2.nyu.org.
nyu.edu.                172800  IN      NS      ns4.nyu.edu.
9DHS4EP5G85PF9NUFK06HEK0O48QGK77.edu. 900 IN NSEC3 1 1 0 - 9DQL407IJB4JHADMF7CJMUFNQMR4DF7M  NS SOA RRSIG DNSKEY NS
EC3PARAM
9DHS4EP5G85PF9NUFK06HEK0O48QGK77.edu. 900 IN RRSIG NSEC3 13 2 900 20241021024907 20241014013907 30299 edu. VFhLOXOR
dYmRRwJLydxQ/f7E8bf+grSEt2TwIWS1rSRh34IL3xfKYiyo lA5TfZ5WaAv3ToeclDpSUoV2xch4yw==
DHG9JFD11U5N1TFRFRED3USJP8LBR960.edu. 900 IN NSEC3 1 1 0 - DJQJOL9OHIS0EOBL9S4M43IPCGUJT1S0  NS DS RRSIG
DHG9JFD11U5N1TFRFRED3USJP8LBR960.edu. 900 IN RRSIG NSEC3 13 2 900 20241024024747 20241017013747 26800 edu. kypCD4qO
HFMyMmMFjnvp5YYgLgvh2SCOWBGfD5Cn2OQ7OpqV+pDEbGuO JLw3V95ntTM4lMGr6o2F+axgKrvhTQ==
;; Received 477 bytes from 192.42.93.30#53(g.edu-servers.net) in 26 ms

nyu.edu.                60      IN      A       216.165.61.24
nyu.edu.                86400   IN      NS      ns1.nyu.net.
nyu.edu.                86400   IN      NS      ns4.nyu.edu.
nyu.edu.                86400   IN      NS      ns2.nyu.org.
;; Received 136 bytes from 128.122.0.76#53(ns2.nyu.org) in 84 ms
```

B.

2. dig +trace google.com @a.root-servers.net

```
; <<>> DiG 9.10.6 <<>> +trace google.com @a.root-servers.net
;; global options: +cmd
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      RRSIG   NS 8 0 518400 20241030050000 20241017040000 61050 . Q14j8GNv8yP4XW+
74i8CaXnNYKAqj8nl+gYjLqPnthOm0WMl4epLEh7g H2r689khiuYOBfP+9vip9OdLj5pmNJGopYX1dP40jTDYYkWl5cN+ejaf CI4fvlZKuOoOr16o
VNee/vLNIZ0iHwfsUnvQmLTJT59/Qn9KySXjjRgP +HevKORtHrjNsZuPQu70nSt95Son8Le0WkUU+ZK5Ox3wqdYsIyDs7Don 3XvA8Fr90cqYb0vUK
ELfnC4LkYIiO4GVczCJgWMDRDodn6QBYtlIQ15p Jv+uac4IubqPTCVqeY/ng0nIsSLGQ2oTSD/3eCYKw9u/uLt8DJ55etNl XDabyw==
;; Received 1097 bytes from 198.41.0.4#53(a.root-servers.net) in 21 ms

com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    86400   IN      DS      19718 13 2 8ACBB0CD28F41250A80A491389424D341522D946B0DA0C0291F2D3D7
 71D7805A
com.                    86400   IN      RRSIG   DS 8 1 86400 20241030050000 20241017040000 61050 . WnI82CB5ZZ+xI/+E
YDnaVrKu2nNSOkaClf4zfr7xritRFdF73Rr1t56w rBUKRhYcH39OCXNg5X5J3LgpfoaWk6BdhKEWSVj+fXtIfy2K1K2wPixm GIFDJgtf4IeuBiXMb
uCCHCNtIEN+B0pC68eyeVuGutOYtecoaNL2+3S6 MdiP3oBqF3oDE/xmewxdoe/iehsak3GjeaT8YZcAjC61VjUulWE+KBW8 5T02fhZekzsiX/oqdn
QcFEvHPCS4anUvFu76gu/GppBEeIAxtOAQGh8D tUGjVEWzmjbw2aysrnNCnROR7yxJR5NuE5qccCx83khwTW51fe+LURCB 7NJxiw==
;; Received 1170 bytes from 192.36.148.17#53(i.root-servers.net) in 13 ms

google.com.             172800  IN      NS      ns2.google.com.
google.com.             172800  IN      NS      ns1.google.com.
google.com.             172800  IN      NS      ns3.google.com.
google.com.             172800  IN      NS      ns4.google.com.
CK0POJMG874LJREF7EFN8430QVIT8BSM.com. 900 IN NSEC3 1 1 0 - CK0Q3UDG8CEKKAE7RUKPGCT1DVSSH8LL  NS SOA RRSIG DNSKEY NS
EC3PARAM
CK0POJMG874LJREF7EFN8430QVIT8BSM.com. 900 IN RRSIG NSEC3 13 2 900 20241023002604 20241015231604 29942 com. u4BMzBMt
Fesf5REVhScwGXI5fM1yKufHinBGJEDWRhIQOWGZMCG2xpHG KZhLEedvZR3uvCuA8qRNFbF4B3pWDw==
S84BOR4DK28HNHPLC218O483VOOOD5D8.com. 900 IN NSEC3 1 1 0 - S84BR9CIB2A20L3ETR1M2415ENPP99L8  NS DS RRSIG
S84BOR4DK28HNHPLC218O483VOOOD5D8.com. 900 IN RRSIG NSEC3 13 2 900 20241024014710 20241017003710 29942 com. ZcqTn2B8
HWWW/syCSLL0/gf7EIdKlIsaKZdK7k4TCfhsScePmJRCgL4S bg+Q8VNFoMegk9+dND6Mn3kVm7xZKw==
;; Received 644 bytes from 192.55.83.30#53(m.gtld-servers.net) in 23 ms

google.com.             300     IN      A       216.58.214.174
;; Received 55 bytes from 216.239.36.10#53(ns3.google.com) in 15 ms
```

3. dig +trace amazon.com @a.root-servers.net

```
; <<>> DiG 9.10.6 <<>> +trace amazon.com @a.root-servers.net
;; global options: +cmd
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      RRSIG   NS 8 0 518400 20241030050000 20241017040000 61050 . Q14j8GNv8yP4XW+74i8CaXnNY
KAqj8nl+gYjLqPnthOm0WMl4epLEh7g H2r689khiuYOBfP+9vip9OdLj5pmNJGopYX1dP40jTDYYkWl5cN+ejaf CI4fvlZKuOoOr16oVNee/vLNIZ0iHwfsUnvQ
mLTJT59/Qn9KySXjjRgP +HevKORtHrjNsZuPQu70nSt95Son8Le0WkUU+ZK5Ox3wqdYsIyDs7Don 3XvA8Fr90cqYb0vUKELfnC4LkYIiO4GVczCJgWMDRDodn6Q
BYtlIQ15p Jv+uac4IubqPTCVqeY/ng0nIsSLGQ2oTSD/3eCYKw9u/uLt8DJ55etNl XDabyw==
;; Received 1097 bytes from 198.41.0.4#53(a.root-servers.net) in 37 ms

com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    86400   IN      DS      19718 13 2 8ACBB0CD28F41250A80A491389424D341522D946B0DA0C0291F2D3D7 71D7805A
com.                    86400   IN      RRSIG   DS 8 1 86400 20241030050000 20241017040000 61050 . WnI82CB5ZZ+xI/+EYDnaVrKu2n
NSOkaClf4zfr7xritRFdF73Rr1t56w rBUKRhYcH39OCXNg5X5J3LgpfoaWk6BdhKEWSVj+fXtIfy2K1K2wPixm GIFDJgtf4IeuBiXMbuCCHCNtIEN+B0pC68eye
VuGutOYtecoaNL2+3S6 MdiP3oBqF3oDE/xmewxdoe/iehsak3GjeaT8YZcAjC61VjUulWE+KBW8 5T02fhZekzsiX/oqdnQcFEvHPCS4anUvFu76gu/GppBEeIAx
tOAQGh8D tUGjVEWzmjbw2aysrnNCnROR7yxJR5NuE5qccCx83khwTW51fe+LURCB 7NJxiw==
;; Received 1170 bytes from 199.7.91.13#53(d.root-servers.net) in 22 ms

amazon.com.             172800  IN      NS      ns1.amzndns.org.
amazon.com.             172800  IN      NS      ns2.amzndns.org.
amazon.com.             172800  IN      NS      ns1.amzndns.co.uk.
amazon.com.             172800  IN      NS      ns2.amzndns.co.uk.
amazon.com.             172800  IN      NS      ns1.amzndns.net.
amazon.com.             172800  IN      NS      ns2.amzndns.net.
amazon.com.             172800  IN      NS      ns1.amzndns.com.
amazon.com.             172800  IN      NS      ns2.amzndns.com.
CK0POJMG874LJREF7EFN8430QVIT8BSM.com. 900 IN NSEC3 1 1 0 - CK0Q3UDG8CEKKAE7RUKPGCT1DVSSH8LL  NS SOA RRSIG DNSKEY NSEC3PARAM
CK0POJMG874LJREF7EFN8430QVIT8BSM.com. 900 IN RRSIG NSEC3 13 2 900 20241023002604 20241015231604 29942 com. u4BMzBMtFesf5REVhS
cwGXI5fM1yKufHinBGJEDWRhIQOWGZMCG2xpHG KZhLEedvZR3uvCuA8qRNFbF4B3pWDw==
K2018RPS5825HNTF5IBTBK1G1T6BMOAB.com. 900 IN NSEC3 1 1 0 - K201OL9RR8G60SVOKTU0G0BE7M05JF51  NS DS RRSIG
K2018RPS5825HNTF5IBTBK1G1T6BMOAB.com. 900 IN RRSIG NSEC3 13 2 900 20241024023633 20241017012633 29942 com. LVTQqjxpxEucESlFQG
75sNz5sRB2ads7g6lIiPdKjqtMMQIljWhdXFen 2TrFE4h7tQRV/bal5aNbcDmETpGXow==
;; Received 671 bytes from 192.35.51.30#53(f.gtld-servers.net) in 27 ms

amazon.com.             900     IN      A       52.94.236.248
amazon.com.             900     IN      A       54.239.28.85
amazon.com.             900     IN      A       205.251.242.103
amazon.com.             7200    IN      NS      ns1.amzndns.co.uk.
amazon.com.             7200    IN      NS      ns1.amzndns.com.
amazon.com.             7200    IN      NS      ns1.amzndns.net.
amazon.com.             7200    IN      NS      ns1.amzndns.org.
amazon.com.             7200    IN      NS      ns2.amzndns.co.uk.
amazon.com.             7200    IN      NS      ns2.amzndns.com.
amazon.com.             7200    IN      NS      ns2.amzndns.net.
amazon.com.             7200    IN      NS      ns2.amzndns.org.
;; Received 274 bytes from 156.154.64.10#53(ns1.amzndns.com) in 18 ms
```

Exercise 5

TCP

reliable_server.py

```python
import socket

# Server-side code (TCP)
def start_tcp_server():
    # Createing a new socket object that supports TCP over IPv4.
    # AF_INET specifies the address family, in this case, IPv4.
    # It indicates that the socket will use an IP address.
    # SOCK_STREAM specifies that the socket will use TCP protocol.
    tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Binding the socket to a specific IP address and port number
    # It associates the socket with a specific local network interface and port number,
    # making it available to listen for incoming connections.
    # In this case, the IP address is the loopback address (localhost).
    # It limits the server to only accept connections from the same machine
    # The port 12345 uniquely identifies the server process running on the local
machine.
    tcp_server_socket.bind(("127.0.0.1", 12345))
    # This tells the socket to listen for incoming connections. The argument 1
specifies
    # the backlog or the maximum number of queued connections before the server starts
refusing new ones.
    # If 1 connection is waiting and another client tries to connect,
    # it will be refused until the previous connection is handled or the queue is
cleared.
    tcp_server_socket.listen(1)
    print(f"TCP Server up at {tcp_server_socket.getsockname()} and listening.")

    # Predefining chatbot responses
    chatbot_responses = {
        "Hey": "Hello! How can I help you today?",
        "What's your name?": "It's nice to meet you!",
        "Goodbye": "Goodbye! Talk soon!"
    }

    # Waiting for client connections and handling them where conn is the new socket
object
    conn, client_address = tcp_server_socket.accept()
    print(f"Connected to {client_address}")

    while True:
```

```
        # The recv() method is used to receive data from the client through the
connection.
        # 1024 specifies the maximum amount of data (in bytes) to be received in one
call.
        # The decode() method is called to convert the received byte stream (raw binary
data)
        # into a string format, by default using 'utf-8'
        client_message = conn.recv(1024).decode()
        # Checks if a response exists in chatbot's dictionary of responses,
        # otherwise, it returns a default message
        response = chatbot_responses.get(client_message, "Sorry, I don't understand.")
        # Sends the response back to the client as an encoded byte stream, instead of a
string
        conn.send(response.encode())

        # This message is used to breakout from the chatbot's conversational loop
        if client_message == "Goodbye":
            break

    # The close() method for conn closes the connection with the client. This means
that no further data can
    # be sent or received using this socket. It frees up resources associated with this
specific connection.
    conn.close()
    # The close() method for the server socket shuts down the server socket, meaning it
will stop listening
    # for new client connections.
    tcp_server_socket.close()

if __name__ == "__main__":
    start_tcp_server()
```

reliable_client.py

```
import socket

# Client-side code (TCP)
def start_tcp_client():
    # Creating a new socket object that supports TCP over IPv4.
    # AF_INET specifies the address family, in this case, IPv4.
    # It indicates that the socket will use an IP address.
```

```
    # SOCK_STREAM specifies that the socket will use TCP protocol.
    tcp_client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # It attempts to establish a connection between the client socket and a server
running
    # on the local machine (localhost) at port 12345.
    tcp_client_socket.connect(("127.0.0.1", 12345))

    while True:
        # Send a message to the server
        message = input("You: ")
        # The send() method is used to send data to the server through the connection
        # It sends the data as an encoded byte stream, instead of a string, by default
        # 'utf-8'
        tcp_client_socket.send(message.encode())

         # The recv() method is used to receive data from the server through the
connection.
        # 1024 specifies the maximum amount of data (in bytes) to be received in one
call.
        # The decode() method is called to convert the received byte stream (raw binary
data)
        # into a string format, by default using 'utf-8'
        server_response = tcp_client_socket.recv(1024).decode()
        print("Bot:", server_response)

        # This message is used to breakout from the chatbot's conversational loop
        if message == "Goodbye":
            break

    # The close() method for the client socket closes the connection with the server.
    tcp_client_socket.close()

if __name__ == "__main__":
    start_tcp_client()
```

UDP

non_reliable_server.py

```
import socket
```

```python
# Server-side code (UDP)
def start_udp_server():
    # Createing a new socket object that supports TCP over IPv4.
    # AF_INET specifies the address family, in this case, IPv4.
    # It indicates that the socket will use an IP address.
    # SOCK_DGRAM specifies that the socket will use UDP protocol.
    udp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    # Binding the socket to a specific IP address and port number
    # It associates the socket with a specific local network interface and port number,
    # making it available to listen for incoming connections.
    # In this case, the IP address is the loopback address (localhost).
    # It limits the server to only accept connections from the same machine
    # The port 12345 uniquely identifies the server process running on the local
machine.
    udp_server_socket.bind(("127.0.0.1", 12345))
    print("UDP Server up and listening.")

    # Predefining chatbot responses
    chatbot_responses = {
        "Hey": "Hello! How can I help you today?",
        "What's your name?": "It's nice to meet you!",
        "Goodbye": "Goodbye! Talk soon!"
    }

    while True:
        # Receive data from the client
        # The recvfrom() method is used to receive data from the client with no
specific connection.
        # 1024 specifies the maximum amount of data (in bytes) to be received in one
call.
        # Unlike recv(), recvfrom() provides the client address along with the message
        # The decode() method is called to convert the received byte stream (raw binary
data)
        # into a string format, by default using 'utf-8'
        message, client_address = udp_server_socket.recvfrom(1024)
        client_message = message.decode()

        # Checks if a response exists in chatbot's dictionary of responses,
        # otherwise, it returns a default message
        response = chatbot_responses.get(client_message, "Sorry, I don't understand.")

        # The sendto() method is used to send data to a specific client when using UDP.
```

```
        # Since UDP is a connectionless protocol, unlike TCP, the server doesn't
maintain a
        # continuous connection with the client. So, the server must specify the target
        # client's address every time it sends data.
        udp_server_socket.sendto(response.encode(), client_address)

        # This message is used to breakout from the chatbot's conversational loop
        if client_message == "Goodbye":
            break

    # The close() method for the server socket shuts down the server socket gracefully
and
    # releases any resources associated with it.
    udp_server_socket.close()

if __name__ == "__main__":
    start_udp_server()
```

non_reliable_client.py

```
import socket

# Client-side code (UDP)
def start_udp_client():
    # Creating a new socket object that supports TCP over IPv4.
    # AF_INET specifies the address family, in this case, IPv4.
    # It indicates that the socket will use an IP address.
    # SOCK_DGRAM specifies that the socket will use UDP protocol.
    udp_client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Since this is UDP, we don't establish any connections

    while True:
        # Send a message to the server
        message = input("You: ")

        # The sendto() method is used to send data to a specific server's port when
using UDP.
        # Since UDP is a connectionless protocol, unlike TCP, the server doesn't
maintain a
        # continuous connection with the client. So, the client must specify the target
```

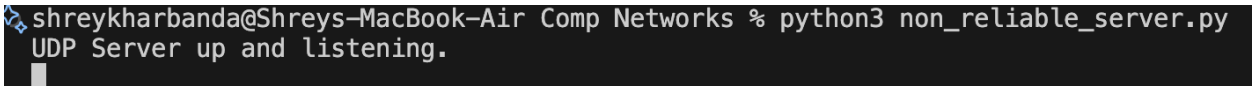
```
        # server's address every time it sends data.
        udp_client_socket.sendto(message.encode(), ("127.0.0.1", 12345))

        # Receive data from the server
        # The recvfrom() method is used to receive data from the server with no
specific connection.
        # 1024 specifies the maximum amount of data (in bytes) to be received in one
call.
        # Unlike recv(), recvfrom() provides the server address along with the message
        # The decode() method is called to convert the received byte stream (raw binary
data)
        # into a string format, by default using 'utf-8'
        server_response, _ = udp_client_socket.recvfrom(1024)
        print("Bot:", server_response.decode())

        # This message is used to breakout from the chatbot's conversational loop
        if message == "Goodbye":
            break

    # The close() method for the server socket shuts down the server socket gracefully
and
    # releases any resources associated with it.
    udp_client_socket.close()

if __name__ == "__main__":
    start_udp_client()
```

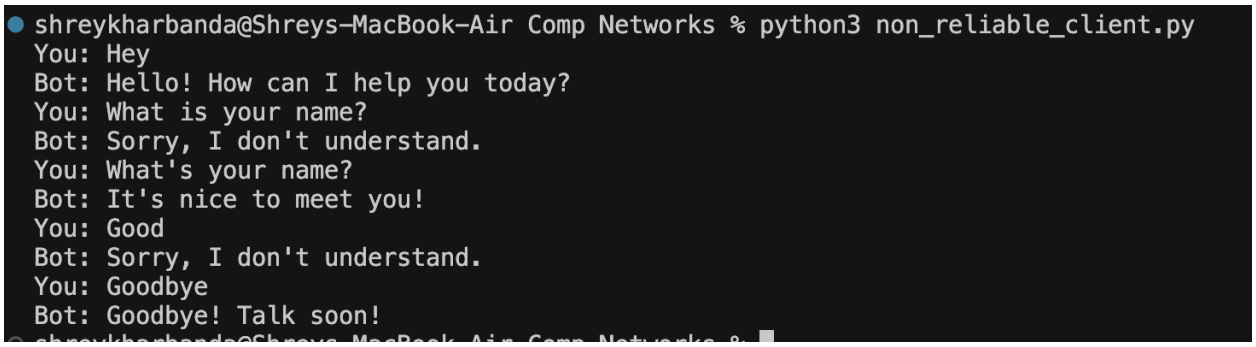
Steps to run the code:

1. Open a terminal (or command prompt) and navigate to the directory where your server script is located.
2. Run the server script using Python:
   python3 reliable_server.py (For the TCP server)
   python3 non_reliable_server.py (For the UDP server)
3. Open another terminal (or command prompt) and navigate to the directory where your client script is located.
4. Run the client script using Python:
   python3 reliable_client.py (For the TCP client)
   python3 non_reliable_client.py (For the UDP client)
5. In the client terminal, you can start typing messages. The server will respond based on the logic defined in the server code.

6. Type “Goodbye” to exit the conversation and close the connection.

UDP Example:





TCP Example: