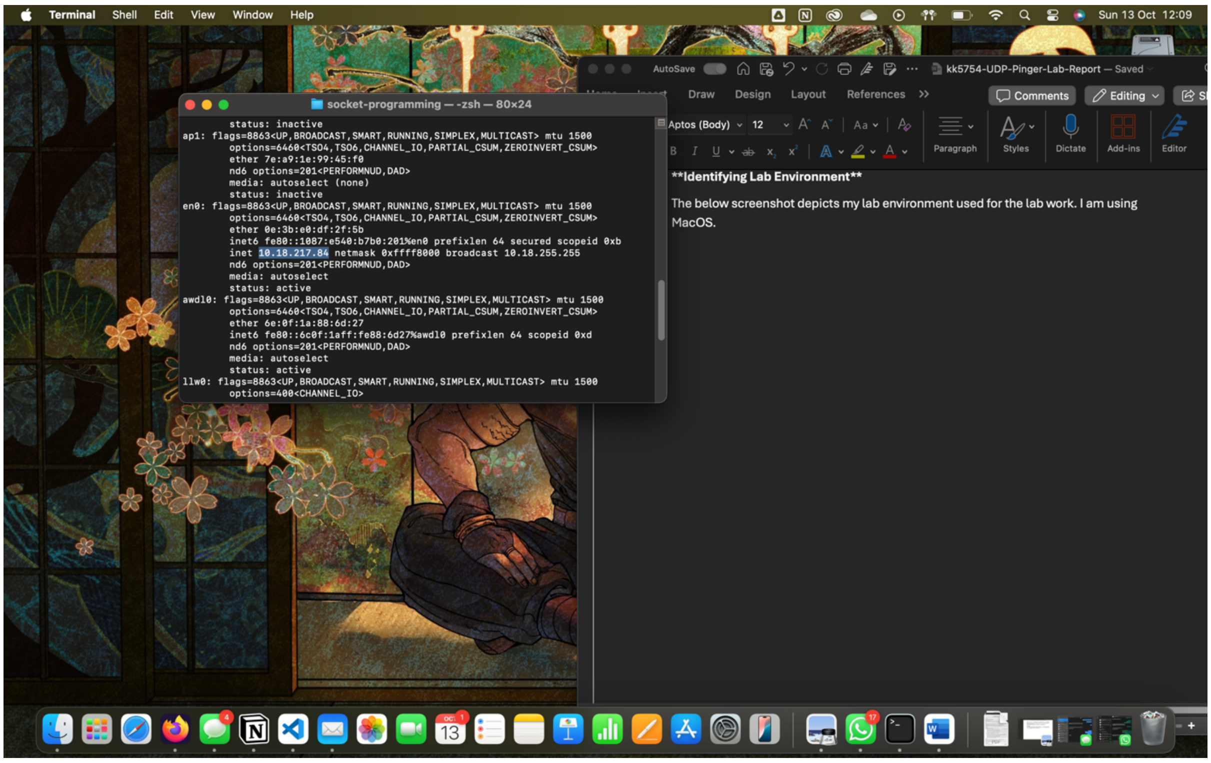
**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

# Computer Networking Socket Programming Lab 2: UDP Pinger Lab

## **Identifying Lab Environment**

The below screenshot depicts my lab environment used for the lab work. I am using MacOS with Python version 3.9.11.


Terminal Shell Edit View Window Help
Sun 13 Oct 12:09
socket-programming — -zsh — 80×24
AutoSave
kk5754-UDP-Pinger-Lab-Report — Saved
Draw Design Layout References
Comments
Editing
Aptos (Body) 12
Paragraph Styles Dictate Add-ins Editor

```
        status: inactive
ap1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 7e:a9:1e:99:45:f0
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (none)
        status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 0e:3b:e0:df:2f:5b
        inet6 fe80::1087:e540:b7b0:201%en0 prefixlen 64 secured scopeid 0xb
        inet 10.18.217.84 netmask 0xffff8000 broadcast 10.18.255.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
awdl0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6460<TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
        ether 6e:0f:1a:88:6d:27
        inet6 fe80::6c0f:1aff:fe88:6d27%awdl0 prefixlen 64 scopeid 0xd
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
llw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
```

**Identifying Lab Environment**

The below screenshot depicts my lab environment used for the lab work. I am using MacOS.

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

1. UDP Pinger Lab: Client Code Completed with commented lines for explanation of code

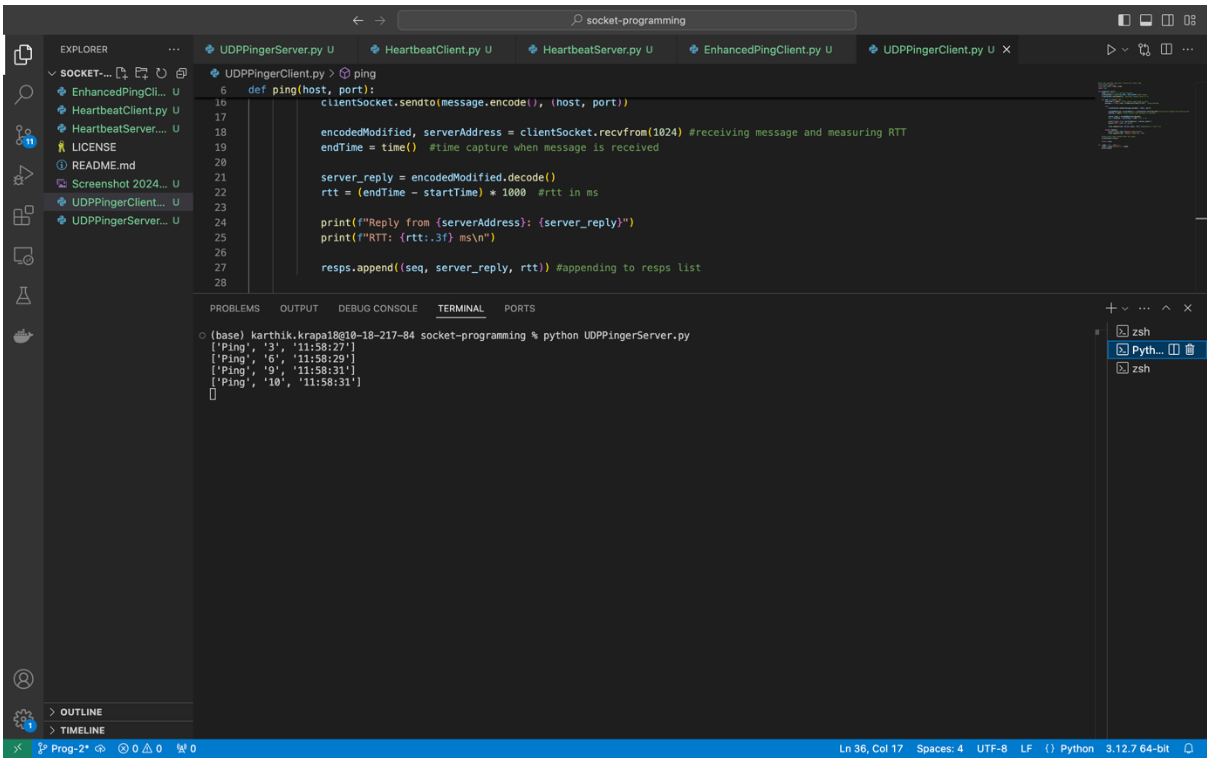
   client.py

```python
from socket import *
import time
import sys

def ping(host, port):
    resps = []
    clientSocket = socket(AF_INET, SOCK_DGRAM)
    clientSocket.settimeout(1)

    for seq in range(1, 11):
        startTime = time.time()  # Retrieve the current time
        message = "Ping {} {}".format(seq, startTime)

        try:
            # Sending the message and waiting for the answer
            clientSocket.sendto(message.encode(), (host, port))
            encodedModified, serverAddress = clientSocket.recvfrom(1024)

            # Checking the current time and if the server answered
            endTime = time.time()
            modifiedMessage = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  # Convert to milliseconds
            resps.append((seq, modifiedMessage, rtt))
        except timeout:
            resps.append((seq, 'Request timed out', 0))

    clientSocket.close()
    return resps

if __name__ == '__main__':
    resps = ping('127.0.0.1', 12000)
    print(resps)
```

   Output:

   Running the UDPPingerServer.py

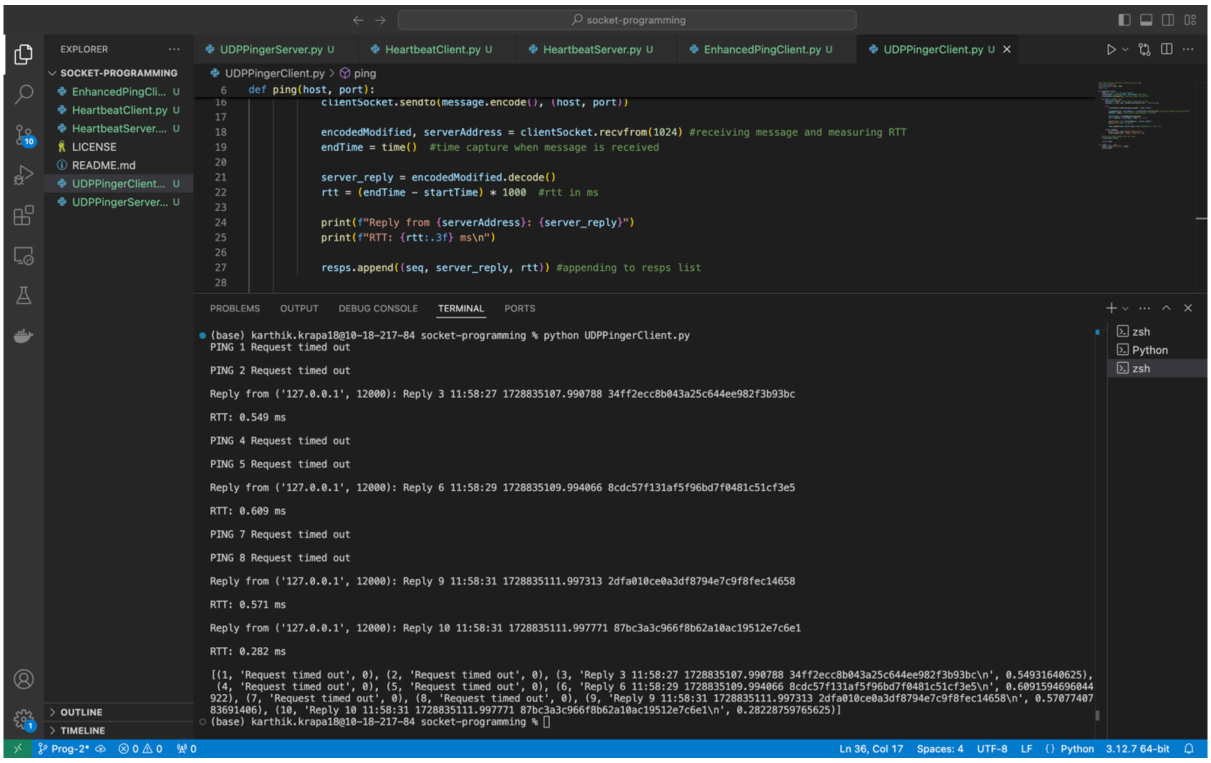
**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754



```python
def ping(host, port):
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list
```

```
(base) karthik.krapa18@10-18-217-84 socket-programming % python UDPPingerServer.py
['Ping', '3', '11:58:27']
['Ping', '6', '11:58:29']
['Ping', '9', '11:58:31']
['Ping', '10', '11:58:31']
```

Running the client.py



```python
def ping(host, port):
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list
```

```
(base) karthik.krapa18@10-18-217-84 socket-programming % python UDPPingerClient.py
PING 1 Request timed out

PING 2 Request timed out

Reply from ('127.0.0.1', 12000): Reply 3 11:58:27 1728835107.990788 34ff2ecc8b043a25c644ee982f3b93bc

RTT: 0.549 ms

PING 4 Request timed out

PING 5 Request timed out

Reply from ('127.0.0.1', 12000): Reply 6 11:58:29 1728835109.994066 8cdc57f131af5f96bd7f0481c51cf3e5

RTT: 0.609 ms

PING 7 Request timed out

PING 8 Request timed out

Reply from ('127.0.0.1', 12000): Reply 9 11:58:31 1728835111.997313 2dfa010ce0a3df8794e7c9f8fec14658

RTT: 0.571 ms

Reply from ('127.0.0.1', 12000): Reply 10 11:58:31 1728835111.997771 87bc3a3c966f8b62a10ac19512e7c6e1

RTT: 0.282 ms

[(1, 'Request timed out', 0), (2, 'Request timed out', 0), (3, 'Reply 3 11:58:27 1728835107.990788 34ff2ecc8b043a25c644ee982f3b93bc\n', 0.54931640625),
 (4, 'Request timed out', 0), (5, 'Request timed out', 0), (6, 'Reply 6 11:58:29 1728835109.994066 8cdc57f131af5f96bd7f0481c51cf3e5\n', 0.6091594696044
922), (7, 'Request timed out', 0), (8, 'Request timed out', 0), (9, 'Reply 9 11:58:31 1728835111.997313 2dfa010ce0a3df8794e7c9f8fec14658\n', 0.57077407
83691406), (10, 'Reply 10 11:58:31 1728835111.997771 87bc3a3c966f8b62a10ac19512e7c6e1\n', 0.28228759765625)]
(base) karthik.krapa18@10-18-217-84 socket-programming % 
```

2. Optional Exercise: Enhanced UDPPingerClient with RTT Statistics and Packet Loss Rate

EnhancedPingClient.py

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

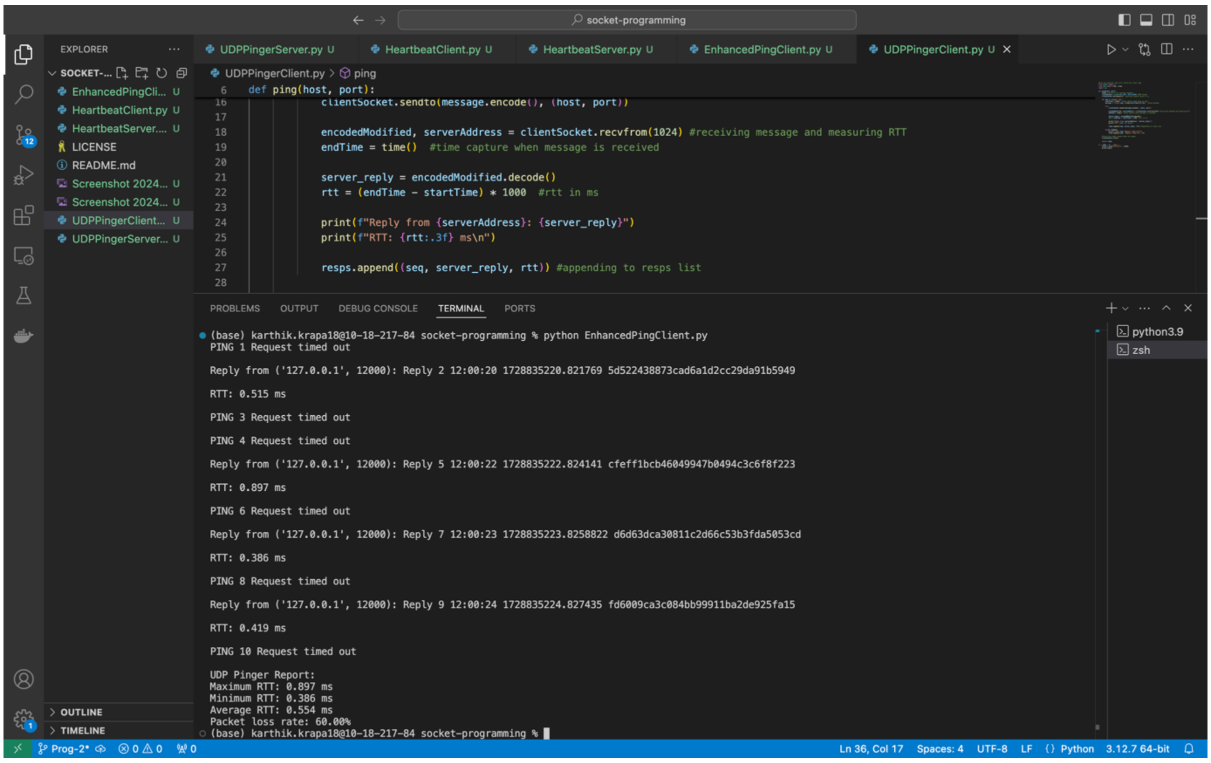
```python
from socket import *
from time import time, ctime
import sys

def ping(host, port):
    rtts = []  # to store the RTTs
    packet_loss_count = 0  # packet lost counter

    clientSocket = socket(AF_INET, SOCK_DGRAM)
    clientSocket.settimeout(1)

    for seq in range(1, 11):
        startTime = time()
        message = f"Ping {seq} {ctime(startTime)[11:19]}"

        try:
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024)
            endTime = time()

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  # RTT in ms
            rtts.append(rtt)  # storing the RTT in list for getting stat report on
this

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

        except timeout:
            print(f"PING {seq} Request timed out\n")
            packet_loss_count += 1

    # Closing client
    clientSocket.close()

    # RTT stats
    if rtts:
        print("UDP Pinger Report:")
        print(f"Maximum RTT: {max(rtts):.3f} ms")  # max value from the list
        print(f"Minimum RTT: {min(rtts):.3f} ms")  # min value from the list
        print(f"Average RTT: {sum(rtts) / len(rtts):.3f} ms")  # avg calculation
from the list
    else:
        print("No successful pings, all requests timed out.")

    # Calculating Packet Loss Rate
    total_packets = 10  # since 10 pings were made from the client skeleton code
earlier
    packet_loss_rate = (packet_loss_count / total_packets) * 100
    print(f"Packet loss rate: {packet_loss_rate:.2f}%")
```

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

```python
if __name__ == '__main__':
    ping('127.0.0.1', 12000)
```

Output:

Running EnhancedPingClient.py



```python
def ping(host, port):
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list
```

```
(base) karthik.krapa18@10-18-217-84 socket-programming % python EnhancedPingClient.py
PING 1 Request timed out

Reply from ('127.0.0.1', 12000): Reply 2 12:00:20 1728835220.821769 5d522438873cad6a1d2cc29da91b5949

RTT: 0.515 ms

PING 3 Request timed out

PING 4 Request timed out

Reply from ('127.0.0.1', 12000): Reply 5 12:00:22 1728835222.824141 cfeff1bcb46049947b0494c3c6f8f223

RTT: 0.897 ms

PING 6 Request timed out

Reply from ('127.0.0.1', 12000): Reply 7 12:00:23 1728835223.8258822 d6d63dca30811c2d66c53b3fda5053cd

RTT: 0.386 ms

PING 8 Request timed out

Reply from ('127.0.0.1', 12000): Reply 9 12:00:24 1728835224.827435 fd6009ca3c084bb99911ba2de925fa15

RTT: 0.419 ms

PING 10 Request timed out

UDP Pinger Report:
Maximum RTT: 0.897 ms
Minimum RTT: 0.386 ms
Average RTT: 0.554 ms
Packet loss rate: 60.00%
(base) karthik.krapa18@10-18-217-84 socket-programming %
```

3. Optional Exercise: UDP Heartbeat

HeartbeatServer.py

```python
from socket import *
import time
import sys

def serve(port, heartbeat_timeout=5):
    serverSocket = socket(AF_INET, SOCK_DGRAM)
    serverSocket.bind(('', port))
    last_seq = -1  # to keep track of sequence number for detecting lost packets
    last_heartbeat_time = time.time()  # the last heartbeat was received time

    print("Server is ready to receive heartbeat signals.")

    while True:
        try:
            message, address = serverSocket.recvfrom(1024)
            current_time = time.time()
```

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

```
            heartbeat = message.decode().split()  # heartbeat message decode
            seq = int(heartbeat[1])
            c_time = float(heartbeat[2])

            print(f"Received Heartbeat {seq} from {address} at {current_time}, sent
at {c_time}")

            if seq != last_seq + 1 and last_seq != -1:  # checking packet loss
                print(f"Warning: Packet loss detected! Expected seq {last_seq + 1},
but got {seq}.")

            if current_time - last_heartbeat_time > heartbeat_timeout:  # checking
heartbeat timeout
                print("Warning: Heartbeat timeout. Client may have stopped.")

            # Updating last sequence number and heartbeat time
            last_seq = seq
            last_heartbeat_time = current_time

        except KeyboardInterrupt:
            serverSocket.close()
            sys.exit()

if __name__ == '__main__':
    serve(12000)
```

HeartbeatClient.py

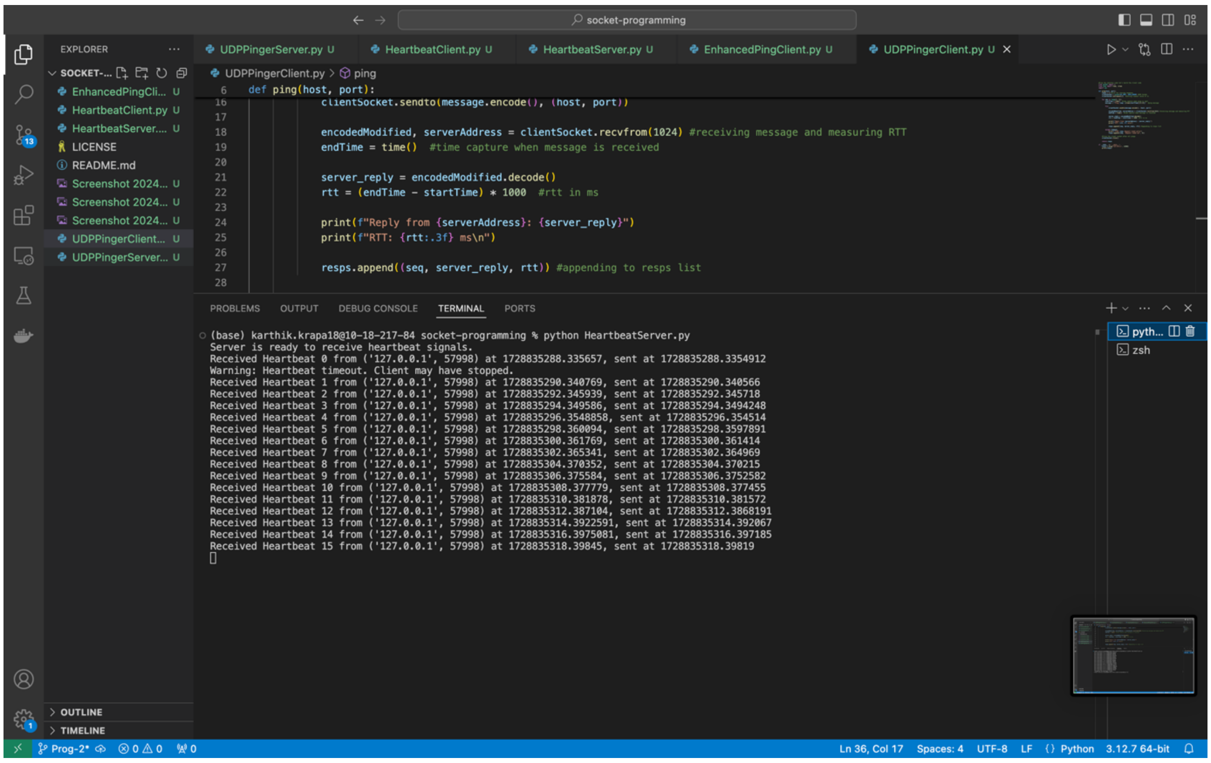
```
from socket import *
import time
import sys

def heartbeat(host, port, interval=2):
    seq = 0  # Sequence number for heartbeat messages

    clientSocket = socket(AF_INET, SOCK_DGRAM)

    while True:
        try:
            current_time = time.time()  # Current time
            message = f"Heartbeat {seq} {current_time}"

            # Sending heartbeat to server
            clientSocket.sendto(message.encode(), (host, port))

            print(f"Sent Heartbeat {seq} at {current_time}")
            seq += 1

            # Wait for the specified interval before sending the next heartbeat
```

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754

```
            time.sleep(interval)

        except KeyboardInterrupt:
            print("Stopping the heartbeat client.")
            clientSocket.close()
            sys.exit()

if __name__ == '__main__':
    heartbeat('127.0.0.1', 12000)
```

Output:

Running HeartbeatServer.py

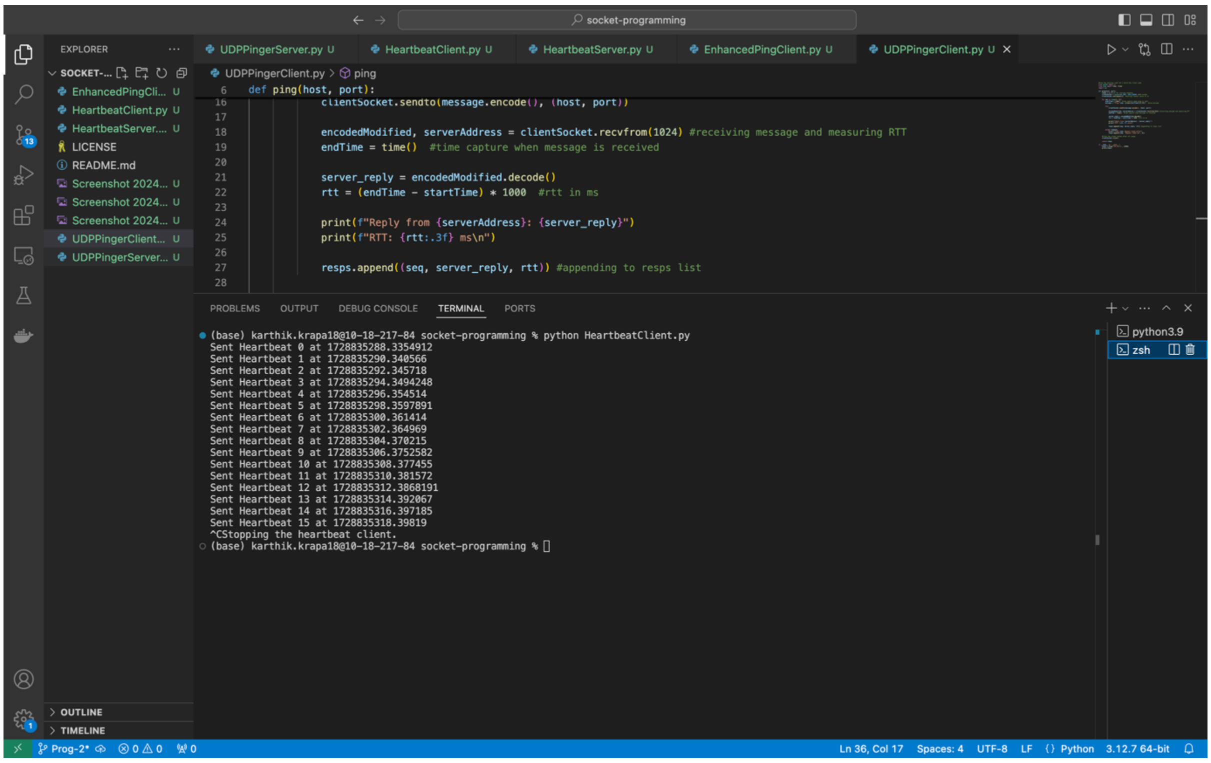


```
def ping(host, port):
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list
```

```
(base) karthik.krapa18@10-18-217-84 socket-programming % python HeartbeatServer.py
Server is ready to receive heartbeat signals.
Received Heartbeat 0 from ('127.0.0.1', 57998) at 1728835288.335657, sent at 1728835288.3354912
Warning: Heartbeat timeout. Client may have stopped.
Received Heartbeat 1 from ('127.0.0.1', 57998) at 1728835290.340769, sent at 1728835290.340566
Received Heartbeat 2 from ('127.0.0.1', 57998) at 1728835292.345939, sent at 1728835292.345718
Received Heartbeat 3 from ('127.0.0.1', 57998) at 1728835294.349586, sent at 1728835294.3494248
Received Heartbeat 4 from ('127.0.0.1', 57998) at 1728835296.3548858, sent at 1728835296.354514
Received Heartbeat 5 from ('127.0.0.1', 57998) at 1728835298.360094, sent at 1728835298.3597891
Received Heartbeat 6 from ('127.0.0.1', 57998) at 1728835300.361769, sent at 1728835300.361414
Received Heartbeat 7 from ('127.0.0.1', 57998) at 1728835302.365341, sent at 1728835302.364969
Received Heartbeat 8 from ('127.0.0.1', 57998) at 1728835304.370352, sent at 1728835304.370215
Received Heartbeat 9 from ('127.0.0.1', 57998) at 1728835306.375584, sent at 1728835306.3752582
Received Heartbeat 10 from ('127.0.0.1', 57998) at 1728835308.377779, sent at 1728835308.377455
Received Heartbeat 11 from ('127.0.0.1', 57998) at 1728835310.381878, sent at 1728835310.381572
Received Heartbeat 12 from ('127.0.0.1', 57998) at 1728835312.387104, sent at 1728835312.3868191
Received Heartbeat 13 from ('127.0.0.1', 57998) at 1728835314.3922591, sent at 1728835314.392067
Received Heartbeat 14 from ('127.0.0.1', 57998) at 1728835316.3975081, sent at 1728835316.397185
Received Heartbeat 15 from ('127.0.0.1', 57998) at 1728835318.39845, sent at 1728835318.39819
```

Running HeartbeatClient.py

**Name:** Krapa Karthik **NYU ID:** N12039854 **NET ID:** kk5754



```python
def ping(host, port):
            clientSocket.sendto(message.encode(), (host, port))

            encodedModified, serverAddress = clientSocket.recvfrom(1024) #receiving message and measuring RTT
            endTime = time()  #time capture when message is received

            server_reply = encodedModified.decode()
            rtt = (endTime - startTime) * 1000  #rtt in ms

            print(f"Reply from {serverAddress}: {server_reply}")
            print(f"RTT: {rtt:.3f} ms\n")

            resps.append((seq, server_reply, rtt)) #appending to resps list
```

```
(base) karthik.krapa18@10-18-217-84 socket-programming % python HeartbeatClient.py
Sent Heartbeat 0 at 1728835288.3354912
Sent Heartbeat 1 at 1728835290.340566
Sent Heartbeat 2 at 1728835292.345718
Sent Heartbeat 3 at 1728835294.3494248
Sent Heartbeat 4 at 1728835296.354514
Sent Heartbeat 5 at 1728835298.3597891
Sent Heartbeat 6 at 1728835300.361414
Sent Heartbeat 7 at 1728835302.364969
Sent Heartbeat 8 at 1728835304.370215
Sent Heartbeat 9 at 1728835306.3752582
Sent Heartbeat 10 at 1728835308.377455
Sent Heartbeat 11 at 1728835310.381572
Sent Heartbeat 12 at 1728835312.3868191
Sent Heartbeat 13 at 1728835314.392067
Sent Heartbeat 14 at 1728835316.397185
Sent Heartbeat 15 at 1728835318.39819
^CStopping the heartbeat client.
(base) karthik.krapa18@10-18-217-84 socket-programming % 
```