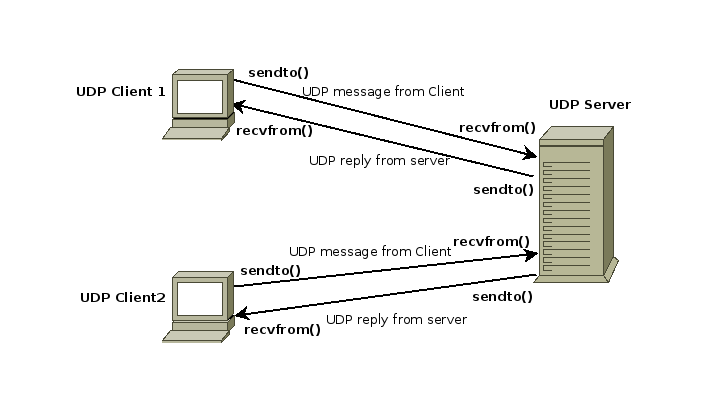
Administrator

[Company name]  [Company address]

|  |
| --- |
| **Subject: Computer Network And Programming**  **Name: MD AYNUL ISLAM**  **Chinese Name: 叶子**  **Student ID: 4420190030**  Topic Name: python-udp-server |

UDP Overview:

UDP is the abbreviation of User Datagram Protocol. UDP makes use of Internet Protocol of the TCP/IP suit. In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.



This document describe the functioning of the client.py and server.py included with it.

Properties of UDP:

1. The UDP does not provide guaranteed delivery of message packets. If for some issue in a network if a packet is lost it could be lost forever.
2. Since there is no guarantee of assured delivery of messages, UDP is considered an unreliable protocol.
3. The underlying mechanisms that implement UDP involve no connection-based communication. There is no streaming of data between a UDP server or and an UDP Client.
4. An UDP client can send "n" number of distinct packets to an UDP server and it could also receive "n" number of distinct packets as replies from the UDP server.

The Server

The server.py is a program that acts as a UDP server. It listens to port 5454 on the localhost until he receives a total of 3 UDP packets. Those packets are assumed to contain short strings.

Once the three packets are received, one reply packet is sent back to the client containing the concatenation of the 3 messages previously received, with an hash symbol (#) between them.

A second reply packet is sent containing the sum of the length of the three packets previously received. At the end the server clear the received messages and restart the process.

The Client

The client.py is a program that acts as a UDP client. It creates a UDP socket, and send a total of 3 packets (containing string entered by the user) to the server (localhost) on port 5454.

Once the messages are sent it awaits for a reply packet that is assumed contains a message with the concatenation of the 3 previously sent messages, separated by a hash symbol (#).

After receiving the first packet it awaits for a second packet which is assumed contains the sum of the length of the three previously sent messages.

Once the second packet is received the client check whether the received message is equal to the concatenation of the sent messages and displays a string with the answer.

It also check wether the checksum received is the same as the checksum calculated and displays another message with the answer. After that the client terminates.

Code Implementation

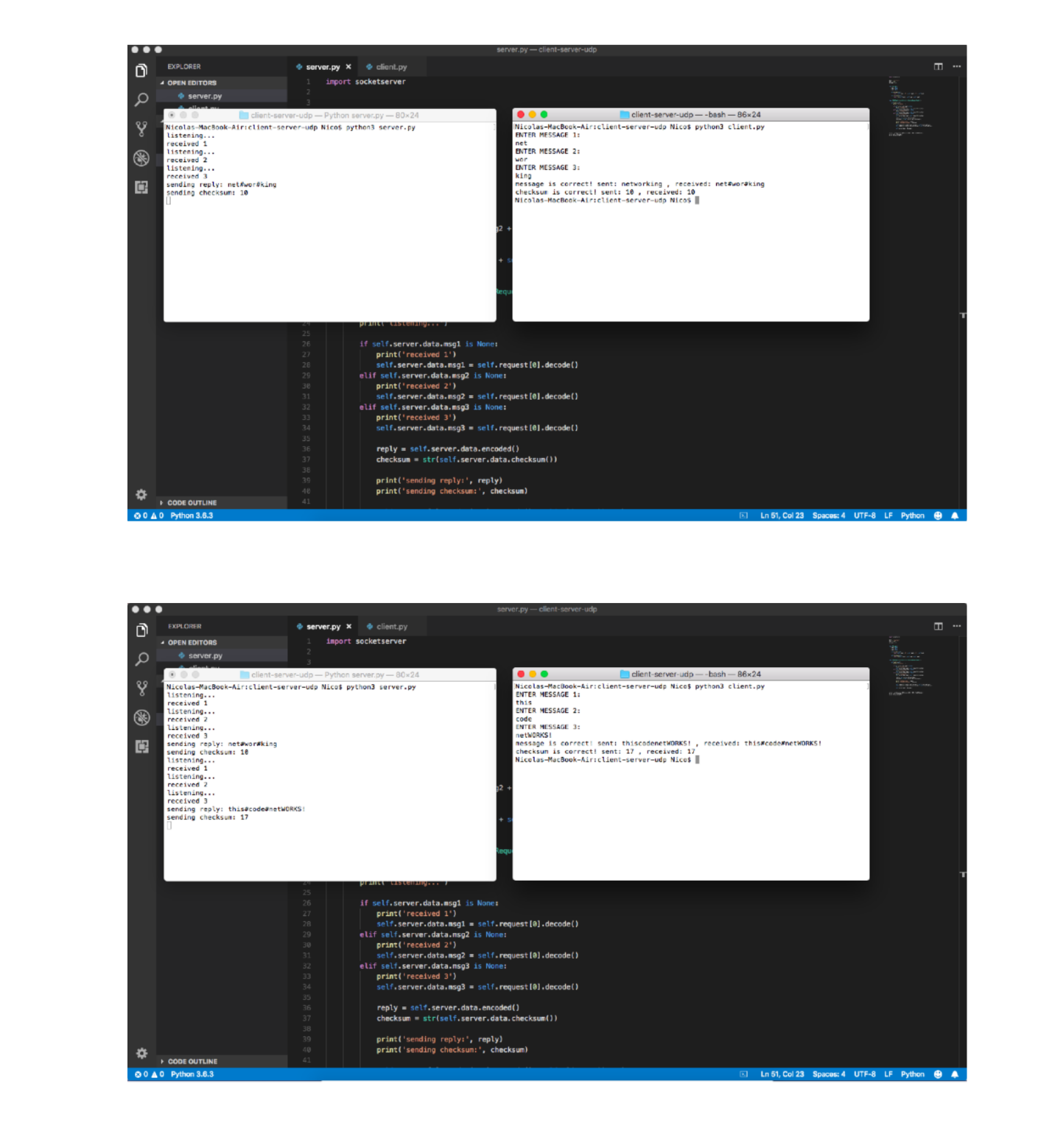
For client:

|  |
| --- |
| **import** **socket**  **import** **time**  HOST = 'localhost'  PORT = **5454**  sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  t1 = input('ENTER MESSAGE 1:**\n**')  t2 = input('ENTER MESSAGE 2:**\n**')  t3 = input('ENTER MESSAGE 3:**\n**')  message = t1 + t2 + t3  checksum = len(message)  sock.sendto(t1.encode(), (HOST, PORT))  sock.sendto(t2.encode(), (HOST, PORT))  sock.sendto(t3.encode(), (HOST, PORT))  recv\_message = ''  recv\_checksum = ''  **while** len(recv\_message) == **0** : recv\_message = sock.recv(**50**).decode()  **while** len(recv\_checksum) == **0** : recv\_checksum = sock.recv(**50**).decode()  **if** (recv\_message.replace('#', '') == message): **print**('message is correct! sent:', message, ', received:', recv\_message)  **else**: **print**('message is wrong', recv\_message)  **if** (int(recv\_checksum) == checksum): **print**('checksum is correct! sent:', checksum, ', received:', recv\_checksum)  **else**: **print**('checksum is wrong', recv\_checksum) |

For server:

|  |
| --- |
| **import** **socketserver**  HOST = 'localhost'  PORT = **5454**  messages = []  **class** **Message**():  msg1 = None  msg2 = None  msg3 = None  **def** **encoded**(self):  **return** self.msg1 + '#' + self.msg2 + '#' + self.msg3  **def** **checksum**(self):  **return** len(self.msg1 + self.msg2 + self.msg3)  **class** **MyUDPHandler**(socketserver.DatagramRequestHandler):  **def** **handle**(self):  **print**('listening...')  **if** self.server.data.msg1 **is** None:  **print**('received 1')  self.server.data.msg1 = self.request[**0**].decode()  **elif** self.server.data.msg2 **is** None:  **print**('received 2')  self.server.data.msg2 = self.request[**0**].decode()  **elif** self.server.data.msg3 **is** None:  **print**('received 3')  self.server.data.msg3 = self.request[**0**].decode()  reply = self.server.data.encoded()  checksum = str(self.server.data.checksum())  **print**('sending reply:', reply)  **print**('sending checksum:', checksum)  self.request[**1**].sendto(reply.encode(), self.client\_address)  self.request[**1**].sendto(checksum.encode(), self.client\_address)  self.server.data = Message()  server = socketserver.UDPServer((HOST, PORT), MyUDPHandler)  server.data = Message()  server.serve\_forever() |

Programme result screenshot:



Reference :

1. More About UDP - Client And Server . [click here.](https://pythontic.com/modules/socket/udp-client-server-example)
2. For converts The code snippets into pretty-printed HTML format, [click here.](http://hilite.me/)