**Socket Programming Assignment 1**

The screenshots below are the python UDP client and server source code: A picture containing text, electronics

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

UDP Client Analysis:

* For the client side I established a server name and server port number.
* Client socket is then created using AF\_INET and SOCK\_DGRAM.
* Using a variable we store raw input from keyboard.
* Send message with server name/port into socket.
* Read received reply message into string.
* Print out received message and close the socket

UDP Server Analysis:

* Establish server port that is the same as client.
* Create UDP socket via AF\_INET and SOCK\_DGRAM.
* Bind socket to local port that has been established.
* Print message that server is active and ready to receive.
* Create loop that will loop forever.
* Variable message stores whats read from UDP socket.
* Variable client address stores whats read from UDP socket.
* Modified message var changes it to all caps.
* Message is sent back via serverSocket.sendto command

The screenshots below are the python TCP client and server source code:

A picture containing text, electronics, screenshot, computer

Description automatically generated

Graphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generated

TCP Client Analysis:

* Established server name/port number.
* Created TCP socket for server via AF\_INET and SOCK\_STREAM.
* Connect socket using clientSocket.connect command.
* Message variable stores user raw input.
* Message sent via clientSocket.send command.
* Modified message var stores what is received from server.
* Print modified message.
* Close socket via clientSocket.close() command

TCP Server Analysis:

* Establish port number to connect to client.
* Create TCP welcoming socket via AF\_INET and SOCK\_STREAM.
* Bind the socket to the port via serverSocket.bind command.
* Server then listens via serverSocket.listen command.
* Print to screen that server is ready to receive.
* Create loop that will loop forever.
* Create socket to accept incoming request via serverSocket.accept command.
* Message var stores what is received from client.
* MessageCap var modifies message to all caps.
* Send modified message via connectionSocket.send command.
* Close connection socket via connectionSocket.close command.
* Does not close welcoming socket