

26 February 2023

SOCKET PROGRAMMING ASSIGNMENT 1

For this assignment, we were assigned to apply the conceptual knowledge with socket to build client/server applications to communicate. This assignment is divided into two: UDP and TCP. Within the assignment, we must input lowercase sentence and receive the same input in all uppercase.

For the first part, we are required to use UDP. Below is the source code for UDP client and server.

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind('', serverPort)
print 'The server is ready to receive'
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
~
~
~
~
~
"udp_server.py" 9L, 321C 9,1 All
```

Figure 1. UDP server.py source code

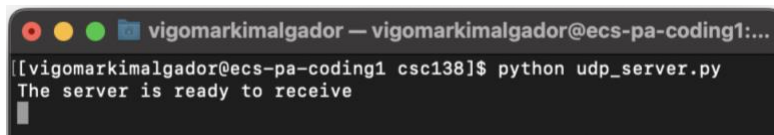
```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message, (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()

~
~
~
~
~

"udp_client.py" 9L, 315C 4,1 All
```

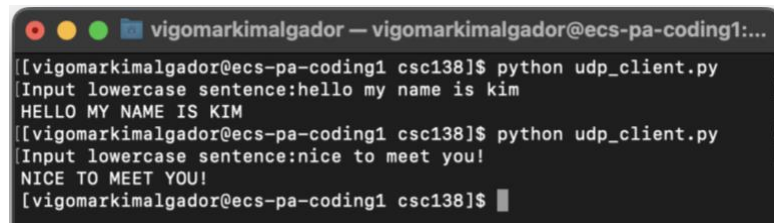
Figure 2. UDP client.py source code

After that, we are required to run first the server and then run the client showed below.

A terminal window titled 'vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...' showing the execution of 'python udp_server.py'. The output is 'The server is ready to receive' followed by a cursor.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...\n[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python udp_server.py\nThe server is ready to receive\n█
```

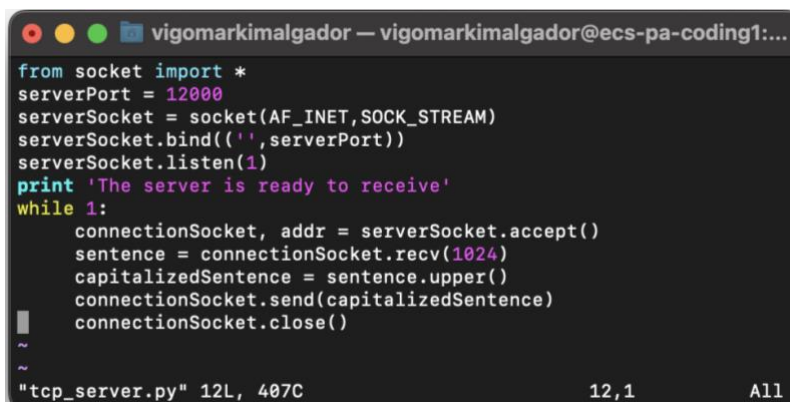
Figure 3. UDP Server-side snapshot

A terminal window titled 'vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...' showing the execution of 'python udp_client.py'. It prompts for a lowercase sentence, which is entered as 'hello my name is kim', and outputs 'HELLO MY NAME IS KIM'. This is repeated with 'nice to meet you!' and 'NICE TO MEET YOU!'.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...\n[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python udp_client.py\nInput lowercase sentence:hello my name is kim\nHELLO MY NAME IS KIM\n[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python udp_client.py\nInput lowercase sentence:nice to meet you!\nNICE TO MEET YOU!\n[[vigomarkimalgador@ecs-pa-coding1 csc138]$ █
```

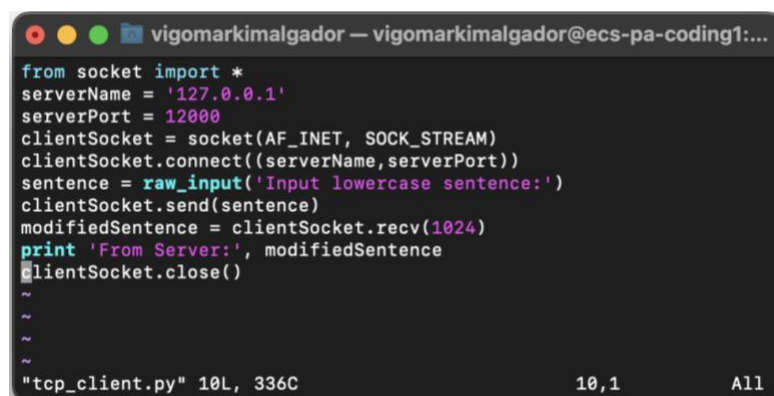
Figure 4. UDP client-side snapshot

The second part, we are required to program with TCP. Below is the source code for TCP client and server.

A terminal window showing the source code for 'tcp_server.py'. The code imports socket, sets serverPort to 12000, binds to all interfaces, listens for one connection, and enters a loop to accept, receive, capitalize, send, and close connections.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...\nfrom socket import *\nserverPort = 12000\nserverSocket = socket(AF_INET, SOCK_STREAM)\nserverSocket.bind(('', serverPort))\nserverSocket.listen(1)\nprint 'The server is ready to receive'\nwhile 1:\n    connectionSocket, addr = serverSocket.accept()\n    sentence = connectionSocket.recv(1024)\n    capitalizedSentence = sentence.upper()\n    connectionSocket.send(capitalizedSentence)\n    connectionSocket.close()\n~\n~\n~\n"tcp_server.py" 12L, 407C                                12,1                All
```

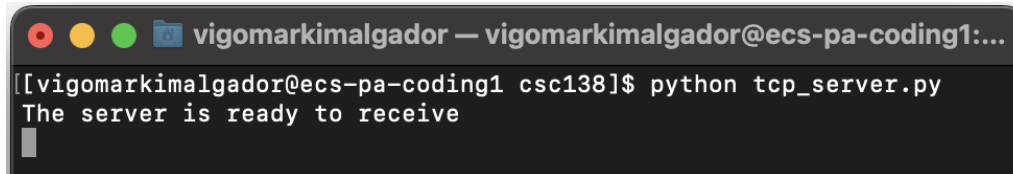
Figure 5. TCP server.py source code

A terminal window showing the source code for 'tcp_client.py'. The code imports socket, sets serverName to '127.0.0.1' and serverPort to 12000, connects to the server, sends a lowercase sentence, receives the modified sentence, prints it, and closes the connection.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...\nfrom socket import *\nserverName = '127.0.0.1'\nserverPort = 12000\nclientSocket = socket(AF_INET, SOCK_STREAM)\nclientSocket.connect((serverName, serverPort))\nsentence = raw_input('Input lowercase sentence:')\nclientSocket.send(sentence)\nmodifiedSentence = clientSocket.recv(1024)\nprint 'From Server:', modifiedSentence\nclientSocket.close()\n~\n~\n~\n"tcp_client.py" 10L, 336C                                10,1                All
```

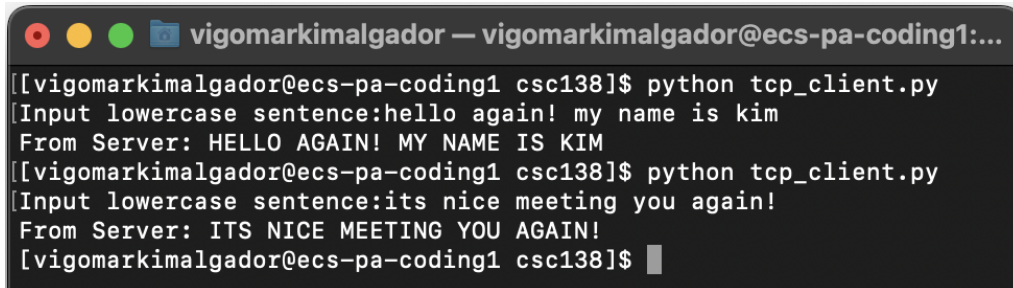
Figure 6. TCP client.py source code

After that, we need to do the same thing to run both client and server.

A terminal window with a dark background and light text. The title bar shows three colored circles (red, yellow, green) and a folder icon, followed by the text "vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...". The terminal content shows a prompt "[vigomarkimalgador@ecs-pa-coding1 csc138]" followed by the command "python tcp_server.py". The output is "The server is ready to receive" followed by a cursor.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...  
[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python tcp_server.py  
The server is ready to receive  
█
```

Figure 7. TCP Server-side snapshot

A terminal window with a dark background and light text. The title bar shows three colored circles (red, yellow, green) and a folder icon, followed by the text "vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...". The terminal content shows two instances of running "python tcp_client.py". In the first instance, the input is "hello again! my name is kim" and the output is "HELLO AGAIN! MY NAME IS KIM". In the second instance, the input is "its nice meeting you again!" and the output is "ITS NICE MEETING YOU AGAIN!". The prompt "[vigomarkimalgador@ecs-pa-coding1 csc138]" is shown at the end of each line.

```
vigomarkimalgador — vigomarkimalgador@ecs-pa-coding1:...  
[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python tcp_client.py  
[Input lowercase sentence:hello again! my name is kim  
From Server: HELLO AGAIN! MY NAME IS KIM  
[[vigomarkimalgador@ecs-pa-coding1 csc138]$ python tcp_client.py  
[Input lowercase sentence:its nice meeting you again!  
From Server: ITS NICE MEETING YOU AGAIN!  
[vigomarkimalgador@ecs-pa-coding1 csc138]$ █
```

Figure 8. TCP Client-side snapshot