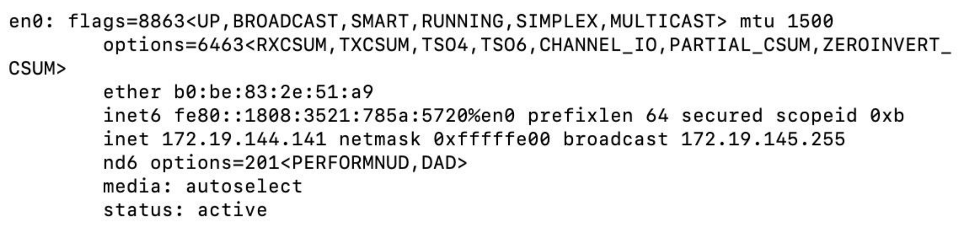
Week4 Lab: Socket Programming

Procedure for running the Python UDP and TCP client and server

Setup Project:

You should have installed Python and PyCharm. Note that the Linux and MacOS machines has existing Python 2 and Windows machines are running python 3, so there are some minor differences between the programs for the two operating systems. Our sample codes use Python 3.

1) In PyCharm, create a project, ClientServer
2) Copy and paste the code from the end of this document (Appendix) into **four separate files** in the project: TCPServer.py, UDPServer.py, TCPClient.py and UDPClient.py
3) Find your PC's IP address
    a. in Windows:
        i. type "cmd" in the search box on windows and pressing <ENTER>
        ii. In the command window type "ipconfig" to find your IP address.
    b. in MacOS:
        i. type "terminal" in the search box and pressing <ENTER>
        ii. in the terminal window, type "ifconfig" to find you IP address

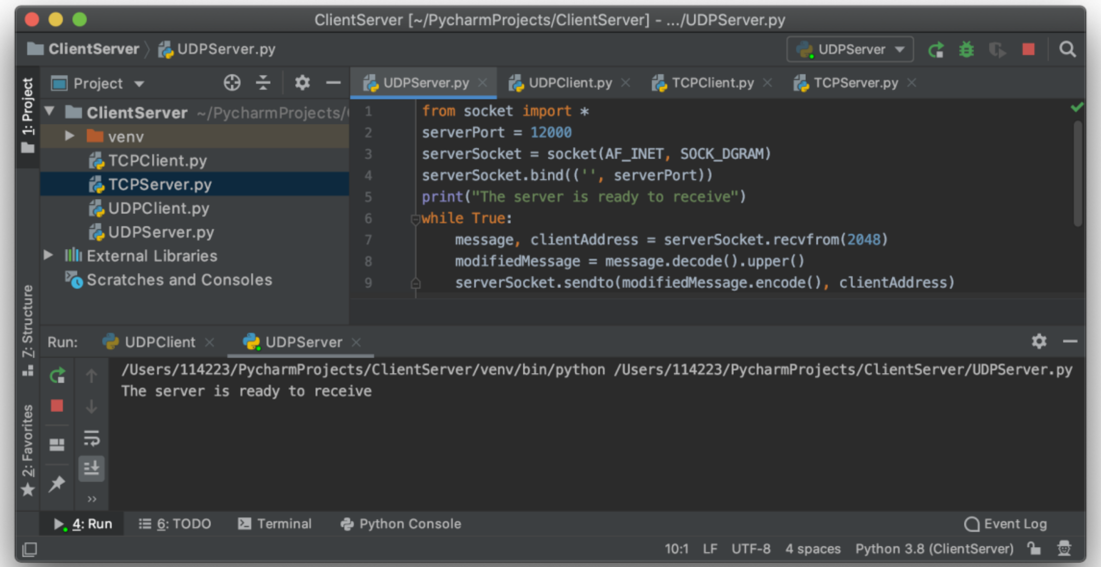


```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=6463<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_
CSUM>
        ether b0:be:83:2e:51:a9
        inet6 fe80::1808:3521:785a:5720%en0 prefixlen 64 secured scopeid 0xb
        inet 172.19.144.141 netmask 0xfffffe00 broadcast 172.19.145.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

IP address: 172.19.126.27

Run your programs

4) run UDP server in PyCharm – screenshot below:



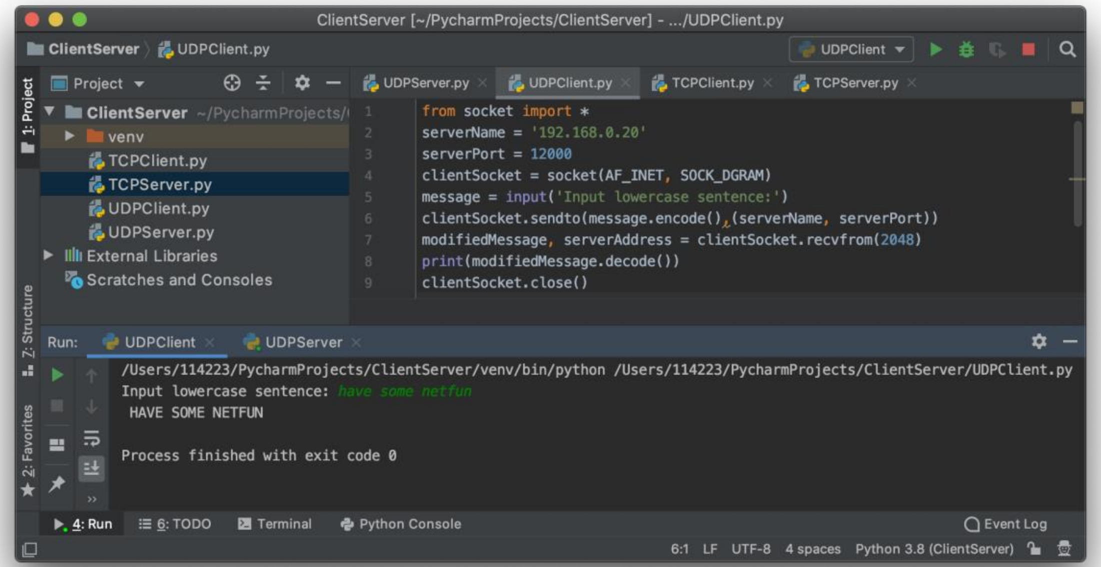
```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

```
/Users/114223/PycharmProjects/ClientServer/venv/bin/python /Users/114223/PycharmProjects/ClientServer/UDPServer.py
The server is ready to receive
```

5) Run UDP Client in PyCharm

Add server IP address to the Client file, and run the client file.

The program will prompt you for a string of characters. Enter lower case sentence and <ENTER>. If everything has been properly configured you string should be returned in all upper case.



```
from socket import *
serverName = '192.168.0.20'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

```
/Users/114223/PycharmProjects/ClientServer/venv/bin/python /Users/114223/PycharmProjects/ClientServer/UDPClient.py
Input lowercase sentence: have some netfun
 HAVE SOME NETFUN

Process finished with exit code 0
```

6) For TCP Server and Client: the procedure is the same for setting up and running.

Lab Tasks

(T1) Install and compile the Python programs TCPClient and UDPClient on one host, and TCPServer and UDPServer on another host.

a. Suppose you run TCPClient before you run TCPServer. What happens? Why?

- **Nothing happens when both programs are running at the same time (they're not dependent on each other- doesn't matter which program is running first)**

b. Suppose you run UDPClient before you run UDPServer. What happens? Why?

- **The same thing happened for a.**

c. What happens if you use different port numbers for the client and server sides?

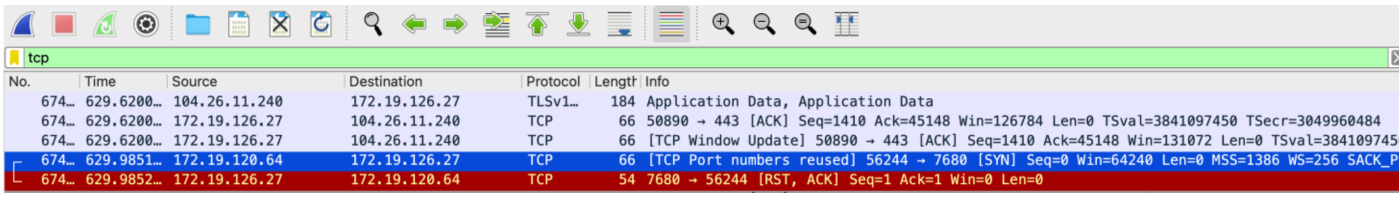
- **The message will not be able to be sent (both port numbers assigned to the server and client must be the same for communication)**

d. (optional) Use Wireshark at client PC to view the messages between the client and server.

(T2) Suppose that in UDPClient.py, after we create the socket, we add the line:

```
clientSocket.bind(('', 5432))
```

Will it become necessary to change UDPServer.py? **Yes, it needs the same port number or else the client wouldn't be able to communicate with the server.**



```
tcp
No.   Time        Source          Destination     Protocol  Length Info
674…  629.6200…   104.26.11.240   172.19.126.27   TLSv1…    184  Application Data, Application Data
674…  629.6200…   172.19.126.27   104.26.11.240   TCP        66  50890 → 443 [ACK] Seq=1410 Ack=45148 Win=126784 Len=0 TSval=3841097450 TSecr=3049960484
674…  629.6200…   172.19.126.27   104.26.11.240   TCP        66  [TCP Window Update] 50890 → 443 [ACK] Seq=1410 Ack=45148 Win=131072 Len=0 TSval=384109745
674…  629.9851…   172.19.120.64   172.19.126.27   TCP        66  [TCP Port numbers reused] 56244 → 7680 [SYN] Seq=0 Win=64240 Len=0 MSS=1386 WS=256 SACK_P
674…  629.9852…   172.19.126.27   172.19.120.64   TCP        54  7680 → 56244 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

What are the **port numbers** for the sockets in UDPClient and UDPServer?

(client) Source Port: 56244

(server) Destination Port: 7680

What were they before making this change?

(client) Source Port: 25305

(server) Destination Port: 7680

RESULTS: (Client) Source port is the only one that changed, the (server) destination port stayed the same

(use Wireshark to observe and confirm your answers)

(T3) Modify the Servers so they flag all client accesses and/or print the string they receive from the client.

UDP:

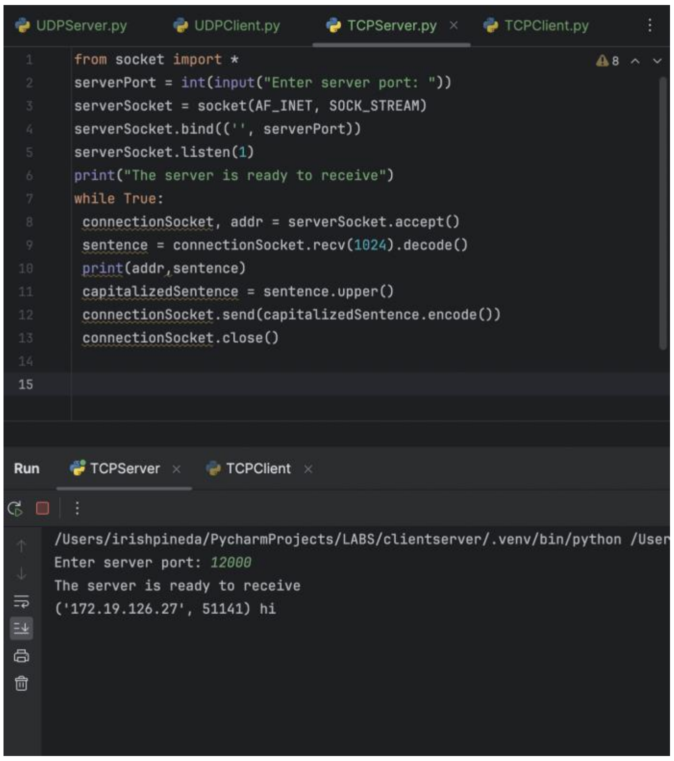
Modified lines: **print(addr,sentence)**

- Enables message received by the server to be printed
- Both under the variables addr and sentence: needs the message to be sent through before being able to print it

TCP:

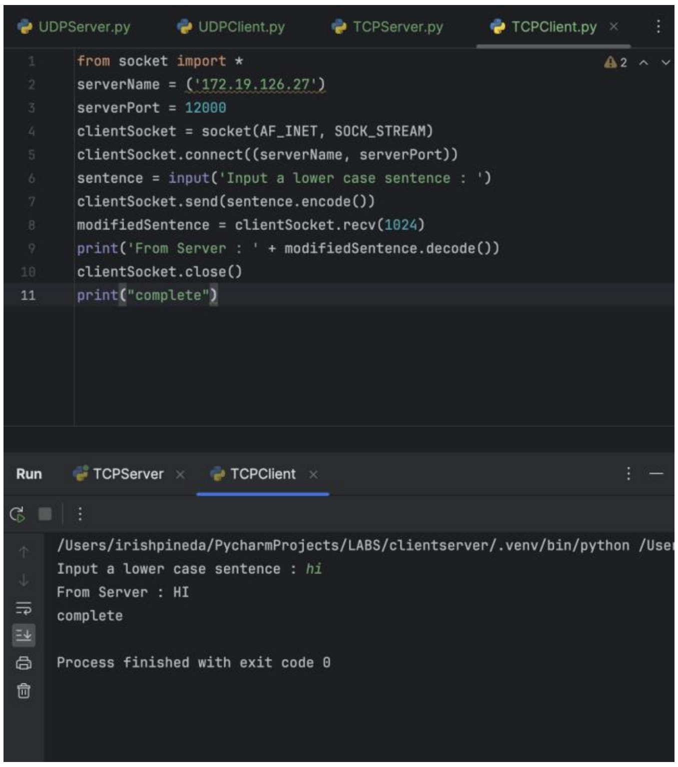
Modified lines: **print(message.decode())**

- This enables the message received by the server to be printed
- Under the variable ModifiedMessage: message is being decoded



```
from socket import *
serverPort = int(input("Enter server port: "))
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while True:
 connectionSocket, addr = serverSocket.accept()
 sentence = connectionSocket.recv(1024).decode()
 print(addr,sentence)
 capitalizedSentence = sentence.upper()
 connectionSocket.send(capitalizedSentence.encode())
 connectionSocket.close()
```

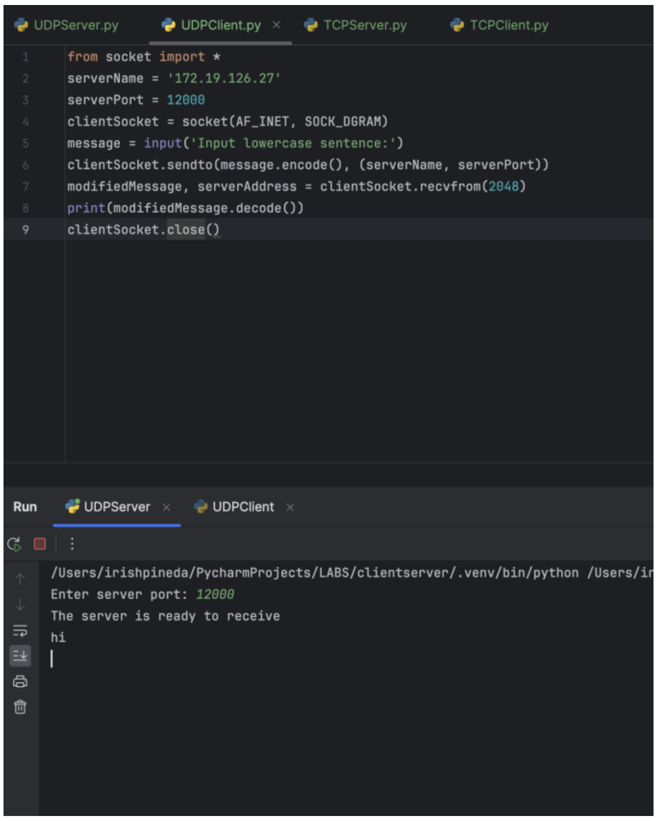
```
/Users/irishpineda/PycharmProjects/LABS/clientserver/.venv/bin/python /User
Enter server port: 12000
The server is ready to receive
('172.19.126.27', 51141) hi
```



```
from socket import *
serverName = ('172.19.126.27')
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('Input a lower case sentence : ')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server : ' + modifiedSentence.decode())
clientSocket.close()
print("complete")
```

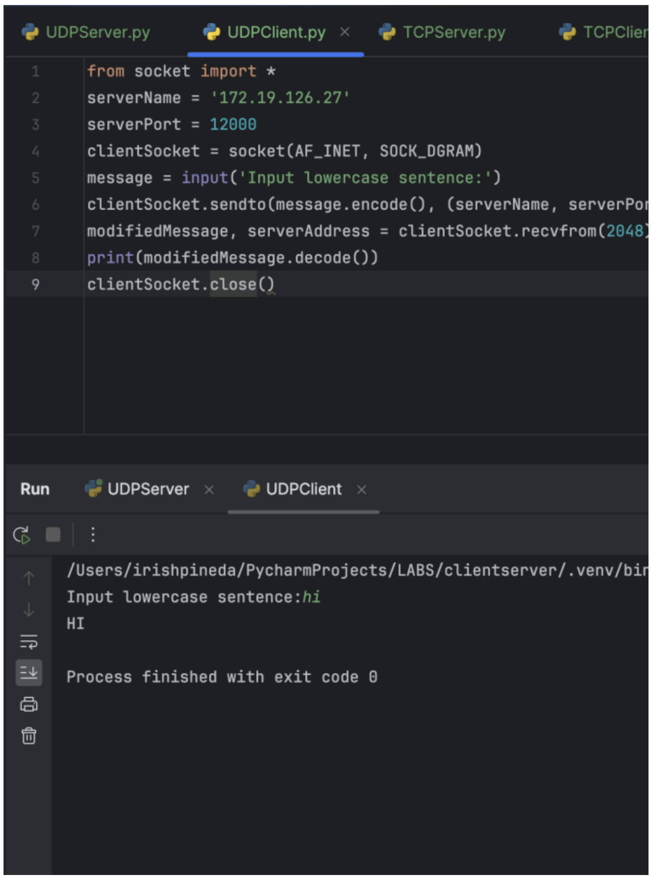
```
/Users/irishpineda/PycharmProjects/LABS/clientserver/.venv/bin/python /Use
Input a lower case sentence : hi
From Server : HI
complete

Process finished with exit code 0
```



```
from socket import *
serverName = '172.19.126.27'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(), (serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

```
/Users/irishpineda/PycharmProjects/LABS/clientserver/.venv/bin/python /Users/i
Enter server port: 12000
The server is ready to receive
hi
```



```
from socket import *
serverName = '172.19.126.27'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(), (serverName, serverPor
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

```
/Users/irishpineda/PycharmProjects/LABS/clientserver/.venv/bir
Input lowercase sentence:hi
HI

Process finished with exit code 0
```

(T4) By default the two servers use port 12000. Change the code so the port can be set from the command line. (Use a port number above 1024)

RESULTS: both server and client must have this code: **serverPort = int(input("Enter server port: "))**

- Can be input by user: enables communication

(T5) (Optional) – Use another PC as an additional client. Now you have two clients communicating with the server. Try sending messages from both clients to the server and have the server print out received messages.

Appendix. Python Code for Week4 Lab

UDPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("The server is ready to receive")
while True:
   message, clientAddress = serverSocket.recvfrom(2048)
   modifiedMessage = message.decode().upper()
   serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

UDPClient.py

```
from socket import *
serverName = '192.168.0.20'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

TCPServer.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print("The server is ready to receive")
while True:
   connectionSocket, addr = serverSocket.accept()
   sentence = connectionSocket.recv(1024).decode()
   capitalizedSentence = sentence.upper()
   connectionSocket.send(capitalizedSentence.encode())
   connectionSocket.close()
```

TCPClient.py

```
from socket import *
serverName = '192.168.0.20'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('Input a lower case sentence : ')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server : ' + modifiedSentence.decode())
clientSocket.close()
print("complete")
```