



CS330—Computer Networks Project

TCP-based client server application

Students Names:

Sarah Khalid Alaridi - 440023365
Sadeem Faisal Alqahtani - 440021429
Sarah Abdullah Alsarami - 440020811
Asia Omar Alrajeh - 440020948

Section: 374

11 December 2021

1. Setting up the Programming Environment

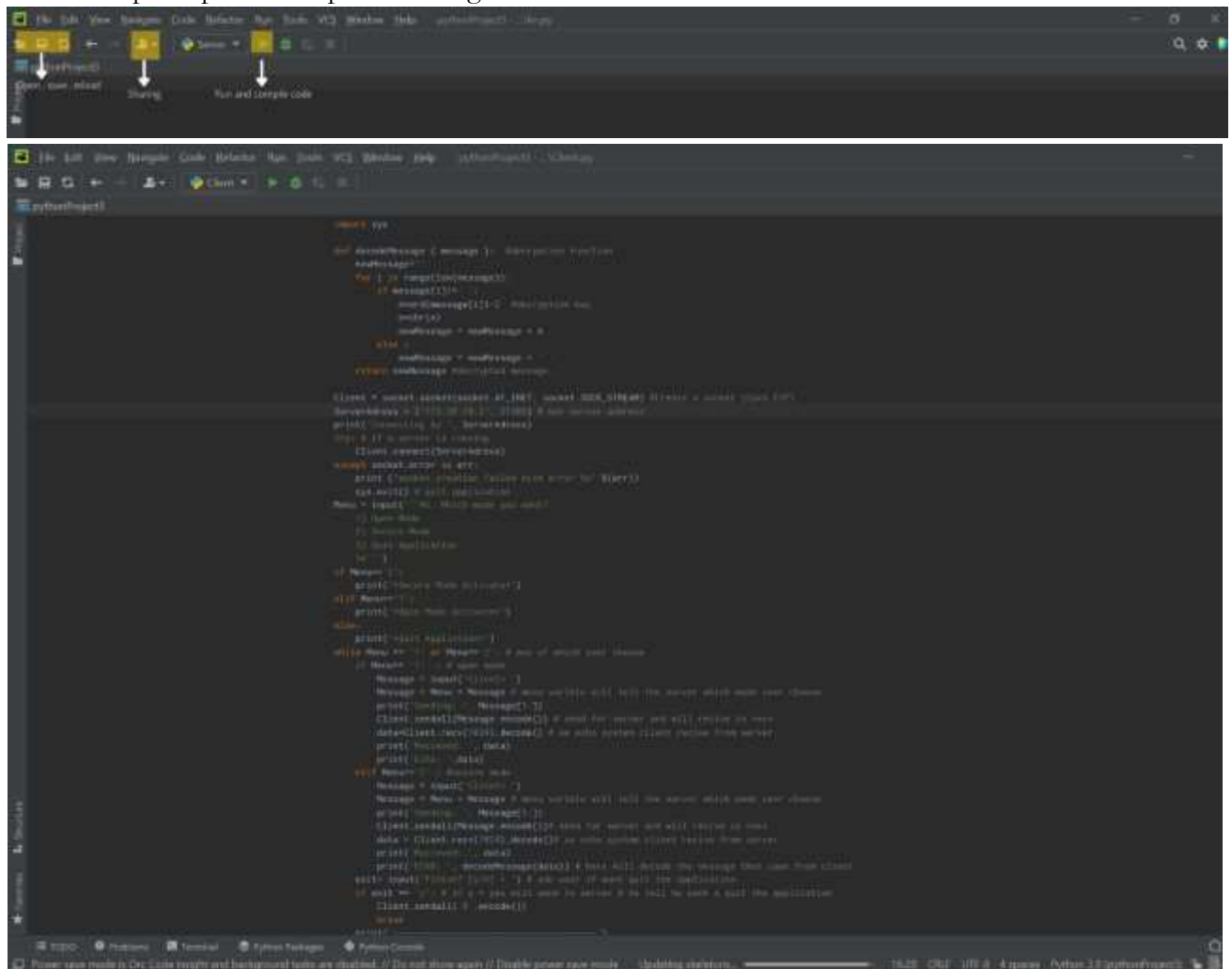
In this project we use python as a programming language. it is easy, and we familiar with it. In python, a lot of libraries that help us.

We use pycharm IDE to write and run python program. These are steps how to download pycharm:

Visit this website and download the suitable version you want:

<https://www.jetbrains.com/pycharm/download/#section=windows>

This is simple explain of important things in IDE:

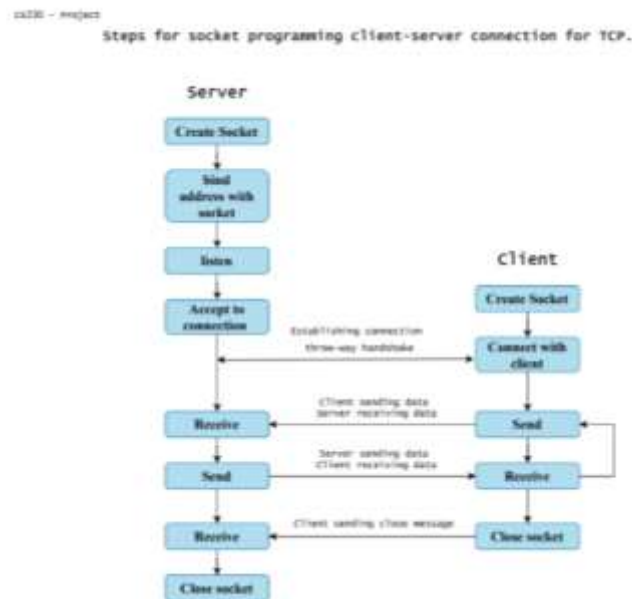


2. Steps for TCP socket programming for client-server connection

We use **Python** socket programming to implement the client- server communication over TCP protocol. we use some sites to help us to write our code:

<https://www.youtube.com/watch?v=6DtinPYTZBY>

<https://www.youtube.com/watch?v=jgaQAIP4toU>



Server side:

Create socket:

```
Server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Create socket type. AF_INET is the Internet address family for IPv4. SOCK_STREAM is the socket type for TCP.

Bind address with socket:

```
ServerAddress = ('localhost', 21265) #localhost is Ip of host device
```

```
Server.bind(ServerAddress)
```

Bind parameter is dependent on Internet address family, we use IPv4 so must fill (Ip, port).

Listen:

```
Server.listen(1) # 1 = maximum length for the queue of pending connections
```

creates a connection request queue of length backlog to queue incoming connection requests.

Accept to connection:

```
connection, ClientAddress = Server.accept()
```

returns a new socket object representing the connection and a tuple holding the address of the client.

Receive:

```
data = connection.recv(1024)
```

Server reads whatever data that the client sends.

Send:

```
connection.sendall(data)
```

Server works as a echo, so if client send something the server will send it back by using sendall().

Receive:

```
data = connection.recv(1024)
```

we send a close connection message to tell server this connection is finish.

Close:

```
connection.close()
```

```
Server.close()
```

After receive connection message the connection will close and the server socket.

Client side:

Create socket:

```
Client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

As we said in server side, Create socket type. AF_INET is the Internet address family for IPv4. SOCK_STREAM is the socket type for TCP.

Connect with server:

```
Client.connect(ServerAddress) # Server Address = (Ip,port)
```

Connect with server address to establish a connection.

Send:

```
Client.sendall(Message)
```

Send to server a message.

Receive:

```
data=Client.recv(1024)
```

Receive from server what it sends.

Close Socket:

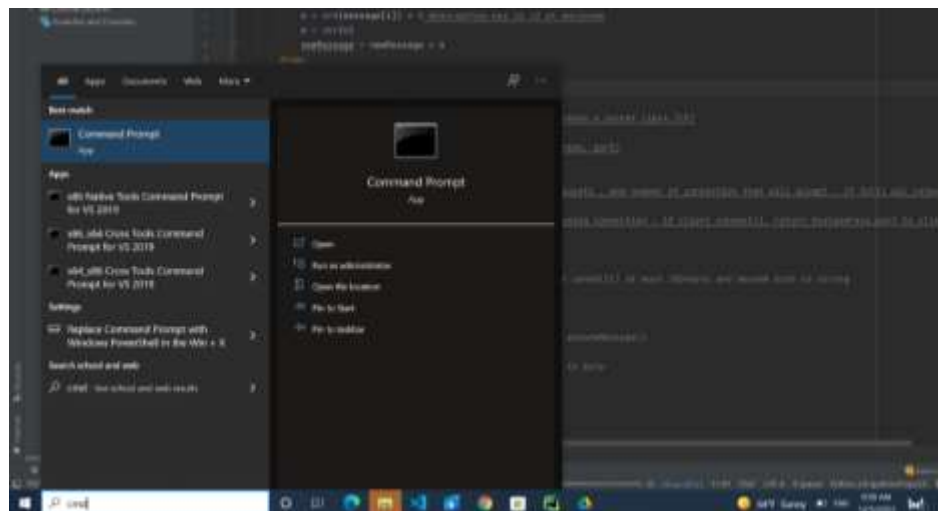
```
Client.sendall('o') # close message
```

```
Client.close()
```

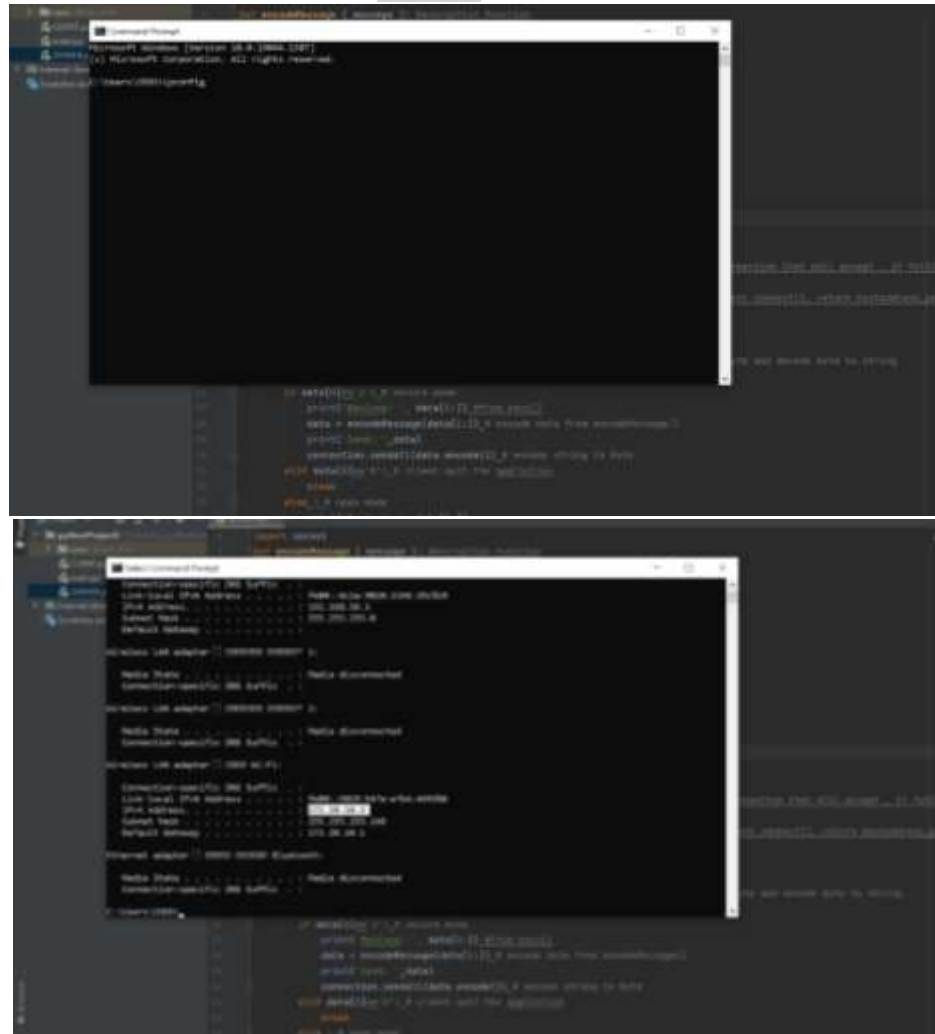
connection will close and the client socket.

3. Steps for setting up the network

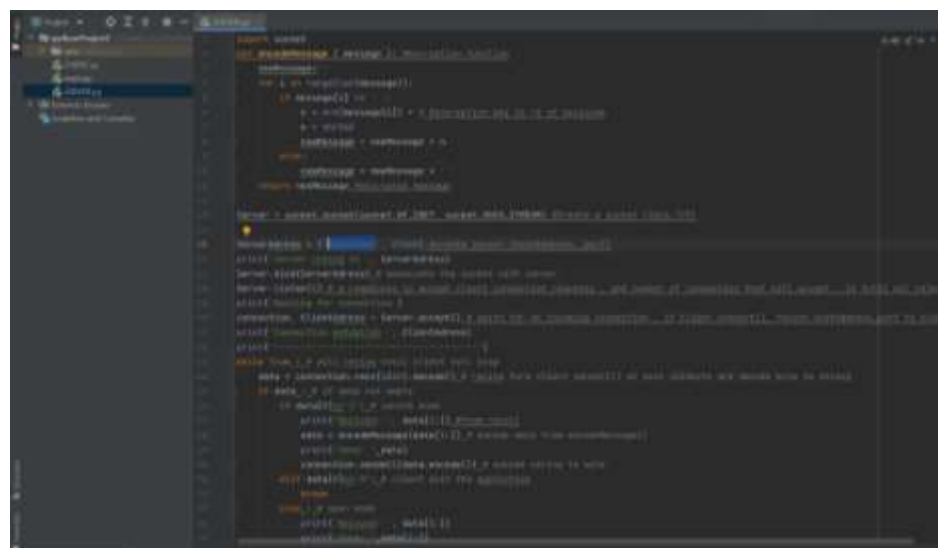
First, we connect in the same network and go to command prompt in server device:



And type `ipconfig` to take Ip server:



Then change the server address in server code and client




```

server_socket
def receive_message(message):
    """
    Receives a message from the client and returns it.
    """
    # If the message is empty, return an empty string
    if message == '':
        return ''
    # If the message is not empty, return the message
    return message

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8080))
server_socket.listen(1)
print('Server ready on 0.0.0.0:8080')
while True:
    # Accept a connection from the client
    connection, client_address = server_socket.accept()
    print('Connection from', client_address)
    # Receive a message from the client
    message = connection.recv(1024)
    # If the message is empty, return an empty string
    if message == '':
        continue
    # Decode the message
    data = message.decode('utf-8')
    # If the data is not empty, process it
    if data != '':
        # Print the data
        print('Received data: %s' % data)
        # Decode the data
        data = data.decode('utf-8')
        # Print the data
        print('Decoded data: %s' % data)
        # Encode the data
        data = data.encode('utf-8')
        # Send the data
        connection.sendall(data)
        # Print the data
        print('Sent data: %s' % data)
    # Close the connection
    connection.close()
    # Print the client address
    print('Client address: %s' % client_address)
    # Print the message
    print('Message: %s' % message)

```

```

def receive_message(message):
    """
    Receives a message from the client and returns it.
    """
    # If the message is empty, return an empty string
    if message == '':
        return ''
    # If the message is not empty, return the message
    return message

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 8080))
server_socket.listen(1)
print('Server ready on 0.0.0.0:8080')
while True:
    # Accept a connection from the client
    connection, client_address = server_socket.accept()
    print('Connection from', client_address)
    # Receive a message from the client
    message = connection.recv(1024)
    # If the message is empty, return an empty string
    if message == '':
        continue
    # Decode the message
    data = message.decode('utf-8')
    # If the data is not empty, process it
    if data != '':
        # Print the data
        print('Received data: %s' % data)
        # Decode the data
        data = data.decode('utf-8')
        # Print the data
        print('Decoded data: %s' % data)
        # Encode the data
        data = data.encode('utf-8')
        # Send the data
        connection.sendall(data)
        # Print the data
        print('Sent data: %s' % data)
    # Close the connection
    connection.close()
    # Print the client address
    print('Client address: %s' % client_address)
    # Print the message
    print('Message: %s' % message)

```

Code for client side:

```

import socket
import sys

def decodeMessage(message, decodeMessage):
    messages =
    for i in range(len(message)):
        if message[i] == '\n':
            decodeMessage[i] = decodeMessage[i] + '\n'
            decodeMessage = decodeMessage + 1
        else:
            decodeMessage = decodeMessage + 1
    return decodeMessage.decode('utf-8')

if __name__ == '__main__':
    # Create a socket object
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Bind the socket to the port
    port = 8080
    sock.bind(('', port))
    # Listen for incoming connections
    sock.listen(1)
    # Accept a connection
    conn, addr = sock.accept()
    # Send a message
    msg = "Hello, World!"
    conn.send(msg.encode('utf-8'))
    # Receive a message
    data = conn.recv(1024)
    # Decode the message
    decodedMessage = decodeMessage(data, decodeMessage)
    # Print the message
    print(decodedMessage)
    # Close the connection
    conn.close()

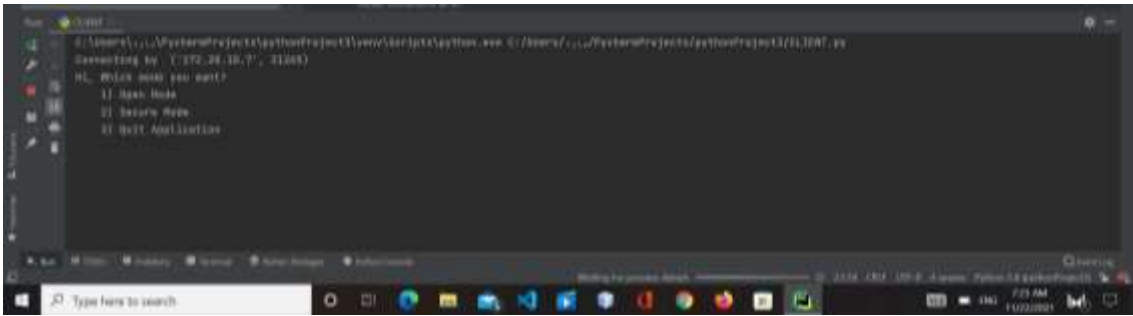
```

5. Snapshots of the application outputs.

First, the server is in standby mode, waiting for the connection from the client

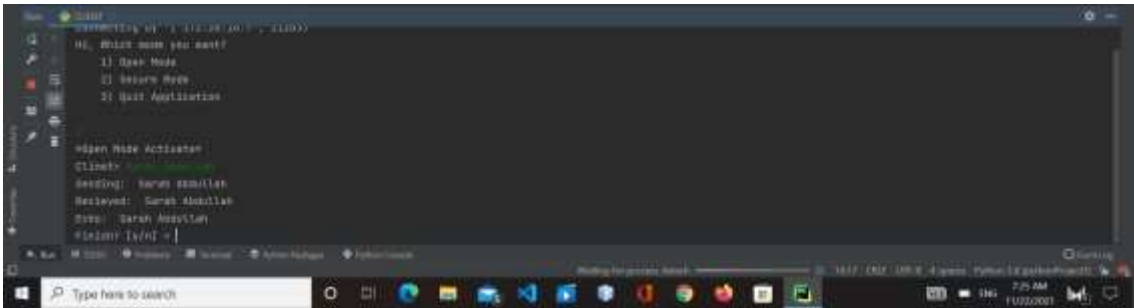
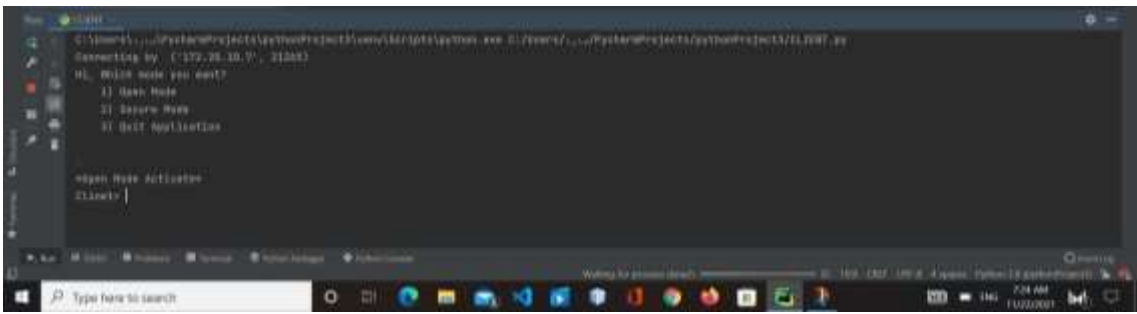


When the client connects, the connection is established then allowed to receive and send messages. Now, the client connects to the server from another host, and it Start then choosing mood from menu



1) Open mode

which mean is open mood, he will write words until the client stops 1When the client chooses the number writing



The user will be asked every time if he has finished or not, while he has not finished, he will be allowed to send like here

```
Python 3.10.12 Shell
C:\Users\Sara\PythonProjects\pythonProject3\venv\Scripts\python.exe C:/Users/Sara/PythonProjects/pythonProject3/server.py
Server running on: ('192.20.30.9', 31300)
Waiting for connection
Connection established ('172.30.10.2', 97549)
.....
Received: Sarah Abdullah
Send: Sarah Abdullah
.....
```

In client when it is write 'n' which mean does not finish, to resend again

```
Python 3.10.12 Shell
>open Mode: Antivirus
Client> Sarah Abdullah
Sending: Sarah Abdullah
Received: Sarah Abdullah
Data: Sarah Abdullah
Pikantur [q/n] > .
.....
Client> Sarah Abdullah
Sending: Sarah Abdullah
Received: Sarah Abdullah
Data: Sarah Abdullah
Pikantur [q/n] > |
.....
```

When it's over, write 'y' to close the connection

```
Python 3.10.12 Shell
Sending: Sarah Abdullah
Received: Sarah Abdullah
Data: Sarah Abdullah
Pikantur [q/n] > .
.....
Client> Sarah Abdullah
Sending: Sarah Abdullah
Received: Sarah Abdullah
Data: Sarah Abdullah
Pikantur [q/n] > .
.....
Process finished with exit code 0
```

On the server side, it receives messages and send them like this then close connection in server

```
Python 3.10.12 Shell
Received: Sarah Abdullah
Send: Sarah Abdullah
.....
Received: Sarah Abdullah
Send: Sarah Abdullah
.....
Class connection
Process finished with exit code 0
```

2) Secure mode

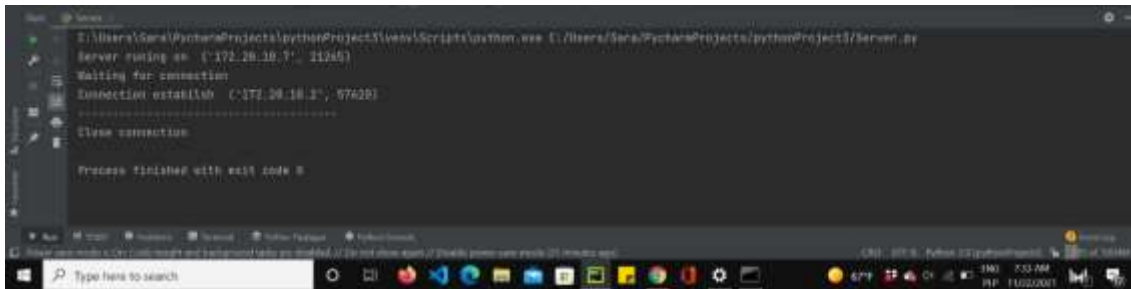
If the client wants to send messages in encrypted mode, he will choose from the list number '2', which means sending encrypted messages

When he chooses option number '3' from the list, the connection will be closed application



```
C:\Users\Gara\PythonProjects\pythonProject5\env\Scripts\python.exe C:/Users/Gara/PythonProjects/pythonProject5/Client.py
Connecting to ('172.28.18.7', 51245)
Hi, What mode you want?
1) Scan Mode
2) Secure Mode
3) Exit Application

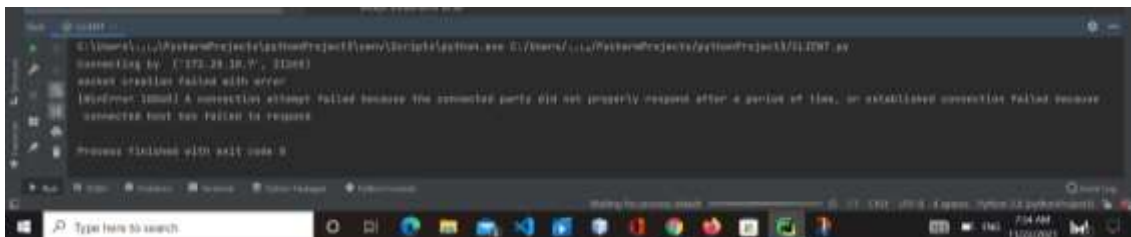
>Exit Application<
Process finished with exit code 0
```



```
C:\Users\Gara\PythonProjects\pythonProject5\env\Scripts\python.exe C:/Users/Gara/PythonProjects/pythonProject5/Server.py
Server running on ('172.28.18.7', 51245)
Waiting for connection
Connection established ('172.28.18.2', 57428)
.....
Close connection
Process finished with exit code 0
```

4) ERORR

When the server is down, and the client will try to connect



```
C:\Users\Gara\PythonProjects\pythonProject5\env\Scripts\python.exe C:/Users/Gara/PythonProjects/pythonProject5/Client.py
Connecting to ('172.28.18.7', 51245)
socket.gaierror: [Errno 1] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond
Process finished with exit code 0
```

6. Problems and solutions:

Problem 1: We try to understand the server and client flow process, and how to make a socket to server and client.

Solution: By watching Socket Programming Tutorial on YouTube and sites dedicated to explaining this concept.

Problem 2: Find a specific encryption algorithm and how to integrate it into the code, does the client send one word or several? How is the server in down mode? When the client and server code are not on the same host, how do they communicate?

Solution: Through the question of the teacher of the course and research on these concepts in many sites and a question from specialists.

Problem 3: Lack of time and a lot of projects and exams in this semester

Solution: by developing a regular time management plan and ending all assignments on time

Problem 4: How to setup connection between two hosts, what must change to establish a connection between them.

Solution: we search in many sites, and we found where the Ip of any device and we just change the Ip of server in code.

References:

- [1] <https://www.jetbrains.com/pycharm/download/#section=windows>
- [2] <https://www.youtube.com/watch?v=6DtinPYTZBY>
- [3] <https://www.youtube.com/watch?v=jgaQAIP4toU>
- [4] <https://www.lifewire.com/introduction-to-client-server-networks-817420>
- [5] 8th edition Jim Kurose, Keith Ross Person, 2020
- [6] <https://realpython.com/python-sockets/>