SOCKET-CLIENT

What is our GOAL for this CLASS?

In this class, we have learned about client side sockets. We created the client side using a python program and learned how to connect client side to server. At last we checked our fully functional chat app on the terminal.

What did we ACHIEVE in the class TODAY?

- Understanding Client socket
- Client Programming
- IP'S address name's conversion into user's name on server side script.
- Fully functional chat app on our command line terminal.

Which CONCEPTS/ CODING BLOCKS did we cover today?

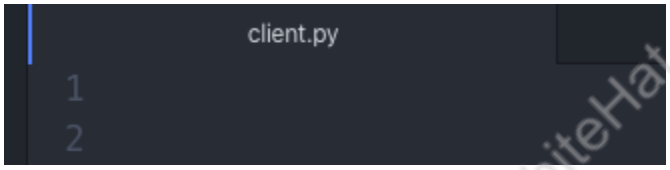
- Client socket
- Connection with server
- Conversion of IP's into Nicknames

How did we DO the activities?

In the last class we created the server socket. Today, we started a client socket to communicate with the server.

Activity 1:

1. Open visual studio code and create one file with name cient.py



2. Import socket

```
import socket
```

3. Create client sockets with the help of socket.socket()function.

```
import socket

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

4. Define IP Address & Port for client side.

```
ip_address = '127.0.0.1'
port = 8000
```

5. Connect your client using function **connect()**.

```
client.connect((ip_address, port))
```

6. Use a **print ()** statement to showcase the client is connected with the server.

```
print("Connected with the server...")
```

7. Our code until now

client.py

```
import socket

nickname = input("Choose your nickname: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ip_address = '127.0.0.1'
port = 8000

client.connect((ip_address, port))

print("Connected with the server...")
```

8. Create a receive function
    - To make the client run indefinitely, add a while loop.

```
print("Connected with the server...")

def receive():
    while True:
```

    - Use **recv()** function to receive message from the server
    - Use **try except** statements, in the mean case if an error occurs,
    - Close the client using **close()** function

```
def receive():
    while True:
        try:
            message = client.recv(2048).decode('utf-8')
        except:
            print("An error occured!")
            client.close()
            break
```

9. Use if -else statement to check the nickname condition.
    - Use **send()** function to send nicknames if available.

```
def receive():
    while True:
        try:
            message = client.recv(2048).decode('utf-8')
            if message == 'NICKNAME':
                client.send(nickname.encode('utf-8'))
            else:
                print(message)
        except:
            print("An error occured!")
            client.close()
            break
```


10. Create the **write()** function and add a **while** loop to make it work indefinitely.

```
def write():
    while True:
```

11. Create message variable to take input from the user using **input()** using string format

```
def write():
    while True:
        message = '{}: {}'.format(nickname, input(''))
        client.send(message.encode('utf-8'))
```

12. Import thread library

```
import socket
from threading import Thread
```

13. Create a new thread for all client connections inside the while loop.

```
receive_thread = Thread(target=receive)
receive_thread.start()
write_thread = Thread(target=write)
write_thread.start()
```

Activity 2: Add nicknames logic to server side

1. Open server.py file
2. Create two variables for nickname and list of clients.

```
list_of_clients = []
nicknames = []
```

3. Add nicknames and a list of client code script to the server.py file.
    - Earlier code in server file looks like this:

```
while True:
    conn, addr = server.accept()
    list_of_clients.append(conn)
    print (addr[0] + " connected")
    new_thread = Thread(target= clientthread,args=(conn,addr))
    new_thread.start()
```

- Add nickname and list of client code

```
while True:
    conn, addr = server.accept()
    conn.send('NICKNAME'.encode('utf-8'))
    nickname = conn.recv(2048).decode('utf-8')
    list_of_clients.append(conn)
    nicknames.append(nickname)
    message = "{} joined!".format(nickname)
    print(message)
    broadcast(message, conn)
    new_thread = Thread(target= clientthread,args=(conn, nickname))
    new_thread.start()
```

4. Go to client thread function
    - Earlier code is like this:

```
def clientthread(conn, addr):
    conn.send("Welcome to this chatroom!".encode('utf-8'))
    while True:
        try:
            message = conn.recv(2048).decode('utf-8')
            if message:
                print ("<" + addr[0] + "> " + message)

                message_to_send = "<" + addr[0] + "> " + message
                broadcast(message_to_send, conn)
            else:
                remove(conn)
        except:
            continue
```

- Add nickname and client list script in client thread function. It will look like this:

```
def clientthread(conn, nickname):
    conn.send("Welcome to this chatroom!".encode('utf-8'))
    while True:
        try:
            message = conn.recv(2048).decode('utf-8')
            if message:
                print(message)
                broadcast(message, conn)
            else:
                remove(conn)
                remove_nickname(nickname)
        except:
            continue
```

5. Create a new function to remove nickname.

```
def remove_nickname(nickname):
    if nickname in nicknames:
        nicknames.remove(nickname)
```

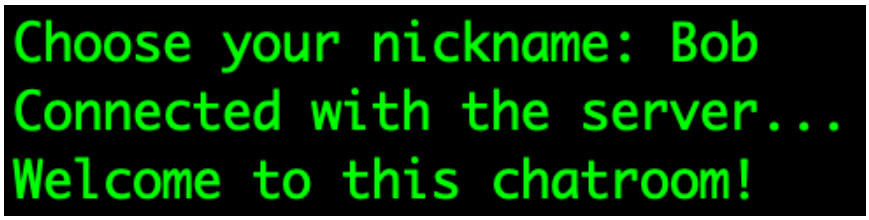
6. Open terminal run server-side file.

```
Server has started...
John joined!
Bob joined!
```

7. Open terminal runs client side for one client.

```
Choose your nickname: John
Connected with the server...
Welcome to this chatroom!
Bob joined!
```

8. Open command terminal run client side for another client



What's NEXT?

In the next class, we will learn about graphical user interfaces based on Tkinter.

Expand Your Knowledge:

Explore the socket's documentation here