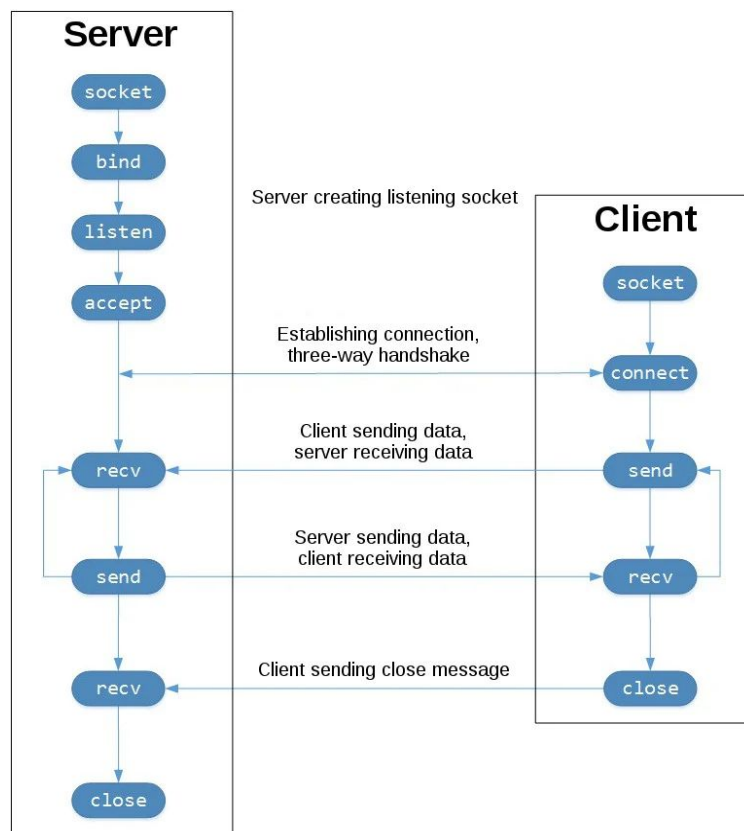


SOCKET PROGRAMMING

AIM: To implement socket programming and establish a connection between client and server.

THEORY: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server.



Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The `socket` library provides specific classes for handling the common transports as well as a generic interface for handling the rest.

The primary socket API functions and methods in this module are:

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `connect_ex()`
- `send()`
- `recv()`
- `close()`

IMPLEMENTATION: To write Internet servers, we use the `socket` function available in the `socket` module to create a socket object. A socket object is then used to call other functions to set up a socket server.

Now call `bind(hostname, port)` function to specify a *port* for your service on the given host. Next, call the *accept* method of the returned object. This method waits until a client connects to the port you specified, and then returns a *connection* object that represents the connection to that client.

Server.py

```
import socket                                # Import socket module

s = socket.socket()                          # Create a socket object
host = socket.gethostname()                  # Get local machine name
port = 12345                                # Reserve a port for your service.
s.bind((host, port))                         # Bind to the port

s.listen(5)                                  # Now wait for client connection.
while True:
    c, addr = s.accept()                     # Establish connection with client.
    print ("Got connection from", addr)
    c.send(bytes("Thank you for connecting", "utf-8"))
    c.close()                                # Close the connection
```

The `socket.connect(hostname, port)` opens a TCP connection to *hostname* on the *port*. Once you have a socket open, you can read from it like any IO object. When done, remember to close it, as you would close a file.

The following code is a very simple client that connects to a given host and port, reads any available data from the socket, and then exits

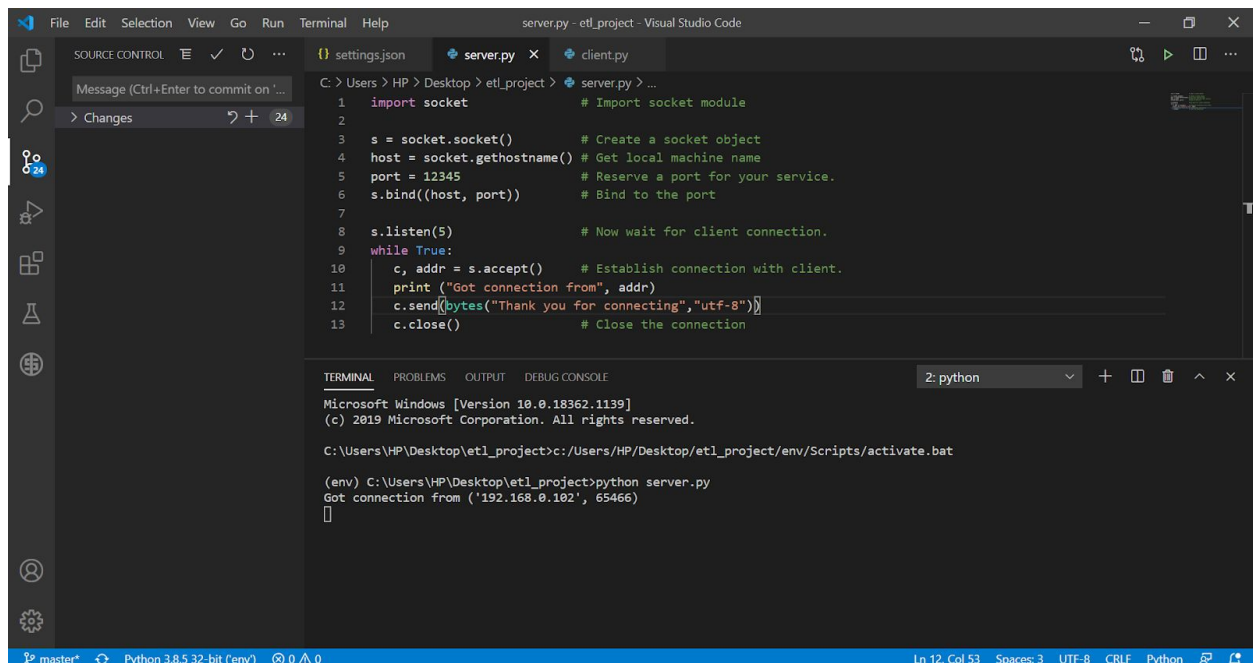
Client.py

```
import socket                                # Import socket module

s = socket.socket()                          # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345                                # Reserve a port for your service.

s.connect((host, port))
print (s.recv(1024))
s.close()                                    # Close the socket when done
```

OUTPUT:



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a source control panel, and a main editor area. The editor displays a file named `server.py` with the following Python code:

```
1 import socket                                # Import socket module
2
3 s = socket.socket()                          # Create a socket object
4 host = socket.gethostname() # Get local machine name
5 port = 12345                                # Reserve a port for your service.
6 s.bind((host, port))                        # Bind to the port
7
8 s.listen(5)                                  # Now wait for client connection.
9 while True:
10     c, addr = s.accept() # Establish connection with client.
11     print ("Got connection from", addr)
12     c.send(bytes("Thank you for connecting","utf-8"))
13     c.close()                                # Close the connection
```

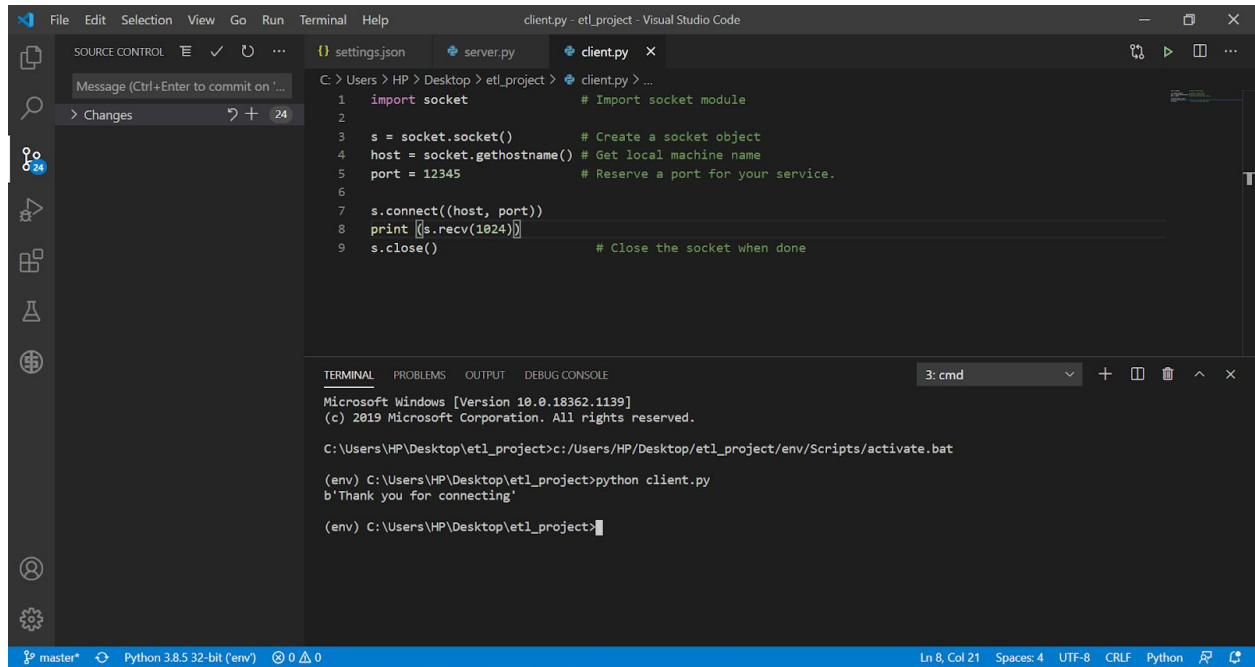
Below the editor is a terminal window titled "TERMINAL" with the following output:

```
Microsoft Windows [Version 10.0.18362.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\etl_project>c:\Users\HP\Desktop\etl_project\env\Scripts\activate.bat

(env) C:\Users\HP\Desktop\etl_project>python server.py
Got connection from ('192.168.0.102', 65466)
[]
```

The status bar at the bottom indicates the current file is `server.py` at line 12, column 53, using Python 3.8.5 32-bit (env).



The screenshot displays the Visual Studio Code interface with a project named 'etl_project'. The editor shows a file named 'client.py' with the following Python code:

```
1 import socket          # Import socket module
2
3 s = socket.socket()     # Create a socket object
4 host = socket.gethostname() # Get local machine name
5 port = 12345           # Reserve a port for your service.
6
7 s.connect((host, port))
8 print (s.recv(1024))
9 s.close()              # Close the socket when done
```

The terminal window at the bottom shows the execution of the script. It starts with the command prompt, followed by activating the Python environment and running the script. The output shows the connection and the received data.

```
Microsoft Windows [Version 10.0.18362.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\etl_project>c:/Users/HP/Desktop/etl_project/env/Scripts/activate.bat

(env) C:\Users\HP\Desktop\etl_project>python client.py
b'Thank you for connecting'

(env) C:\Users\HP\Desktop\etl_project>
```

The status bar at the bottom indicates the current file is 'client.py' at line 8, column 21, using Python 3.8.5 32-bit (env).

CONCLUSION: 1.From this experiment,I learnt about the concepts of socket programming in brief.

2.I also successfully implemented socket programming to establish a connection between client and server.

REFERENCES: [1] https://www.tutorialspoint.com/python/python_networking.htm

[2] <https://realpython.com/python-sockets/>

[3] <https://www.geeksforgeeks.org/socket-programming-cc/>

[4] <https://www.youtube.com/watch?v=T0rYSFPA0A>