COMPUTER NETWORKS LAB

COURSE CODE:UE19CS255

NAME:PRIYA MOHATA SRN:PES2UG19CS301

SECTION:E DATE:21/02/2021

EXPERIMENT : Simple Client-Server Application using Network Socket Programming

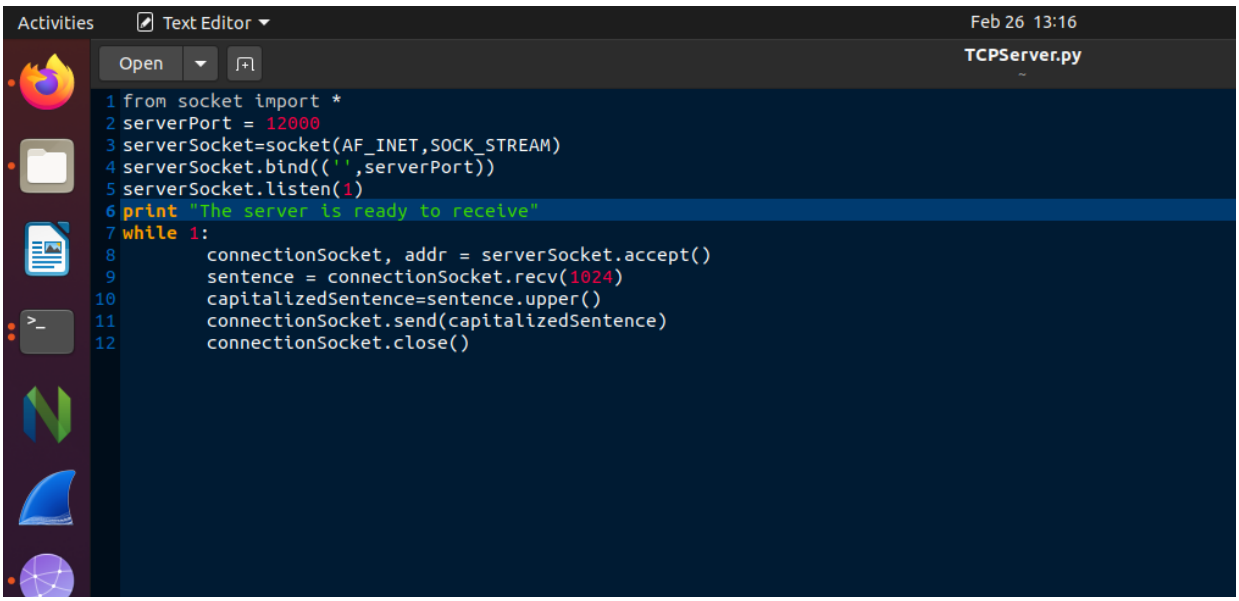
TASK 1: SOCKET PROGRAMMING

1. Create an application that will
    1. Convert lowercase letters to uppercase
        - e.g. [a...z] to [A...Z]
        - code will not change any special characters, e.g. &*!
    2. If the character is in uppercase, the program must not alter
2. Create Socket API both for client and server.
3. Must take the server address and port from the Command Line Interface (CLI).

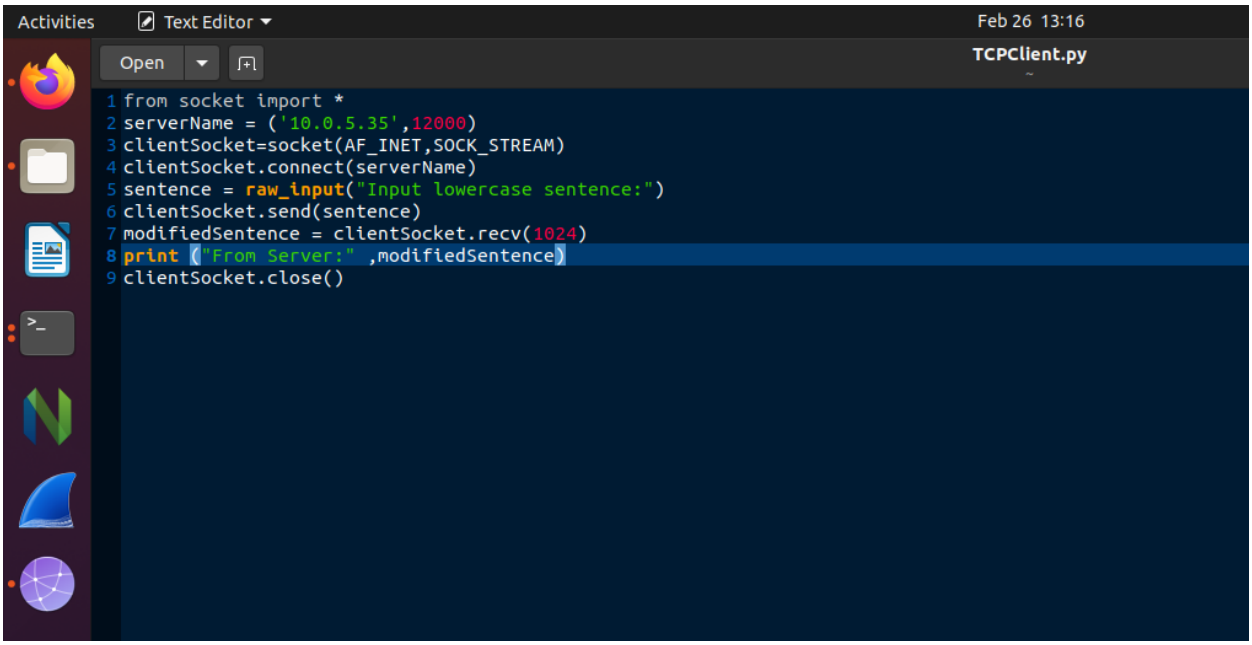
1.1 TCP CONNECTION IN THE SAME SYSTEM

- A TCP connection can be made between two machines with the help of a socket interface using the socket library on Python3.
- To create a TCP socket interface, the type of socket needs to be set as SOCK_STREAM.
- The type of addresses needs to be set as AF_INET which corresponds to IPv4.
- Once the server socket application is created, it needs to be hosted and hence needs to bind to a host IP and port number using the bind()function.
- Similarly, the client socket application needs to connect to a host using the IP address and port number.
- The socket can now listen for incoming connections as well as send messages to connected host machines.

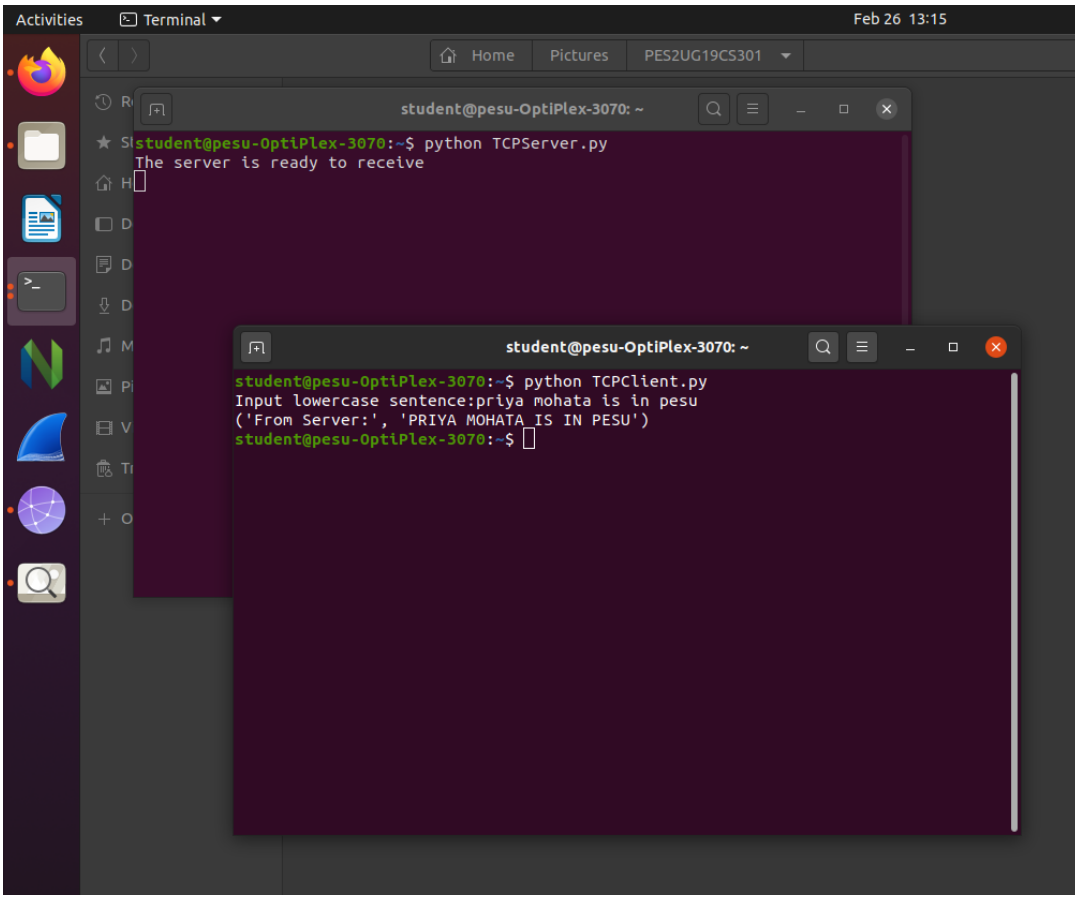
1.1.1 TCP SERVER



1.1.2 TCP CLIENT

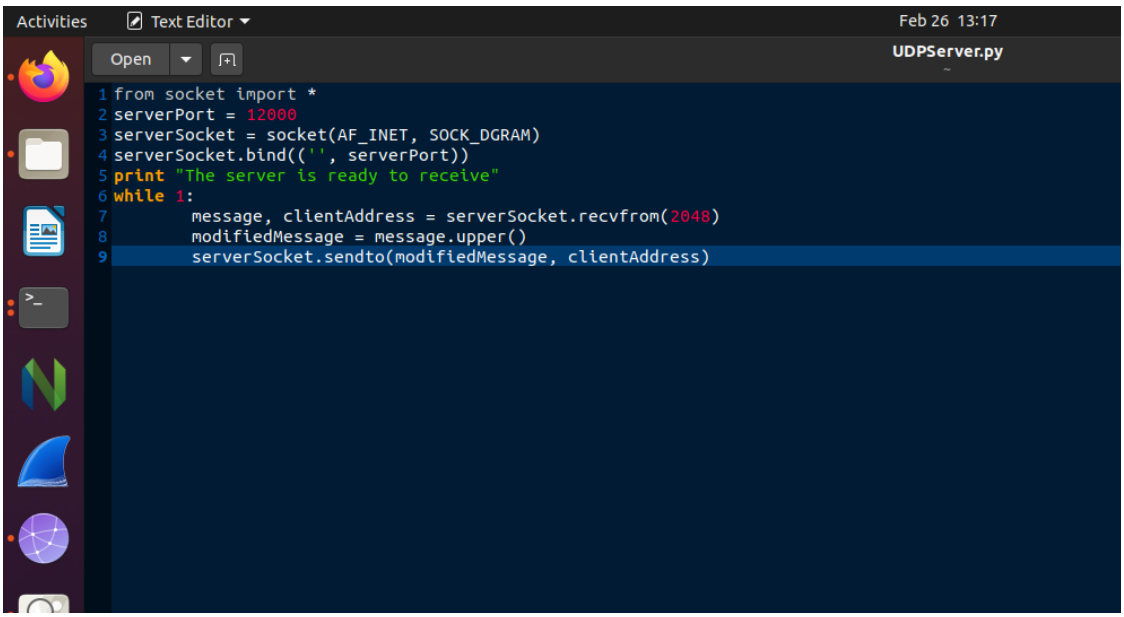


1.1.3 TCP CONNECTION BETWEEN TCP SERVER AND TCP CLIENT

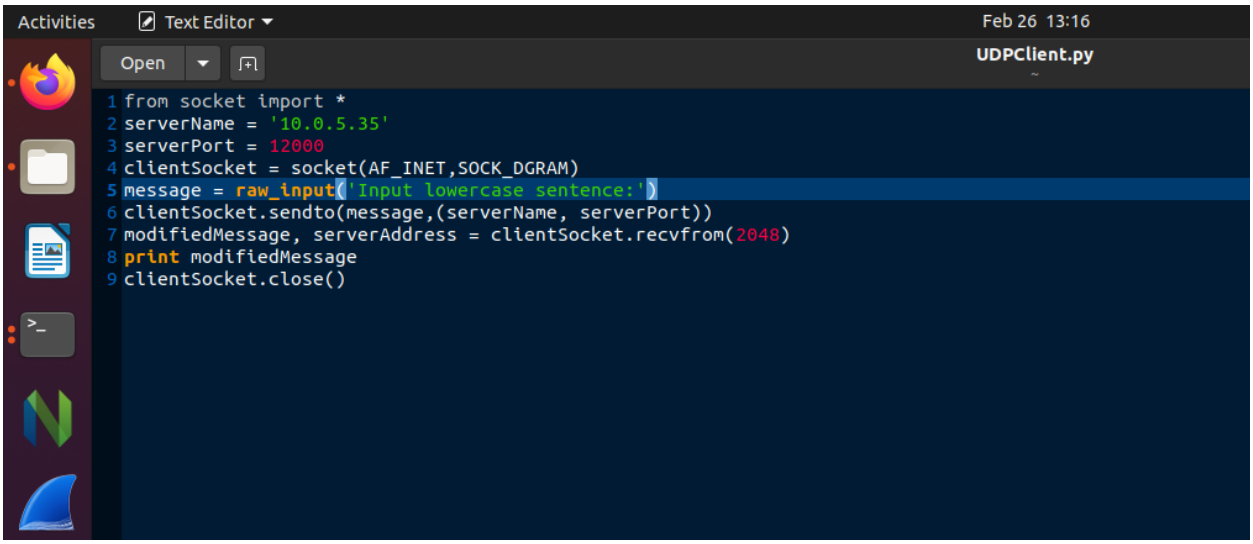


1.2 UDP CONNECTION IN THE SAME SYSTEM

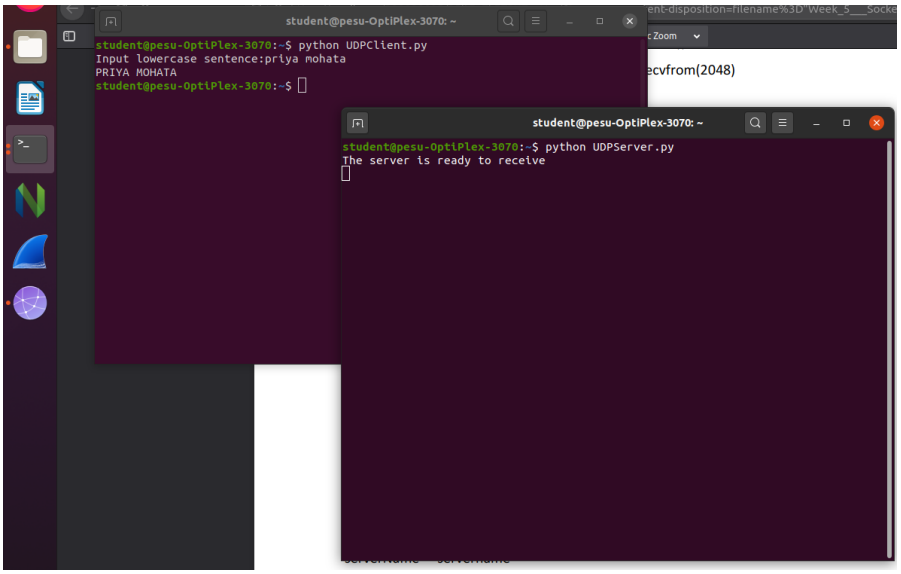
1.2.1 UDP SERVER



1.2.2 UDP CLIENT

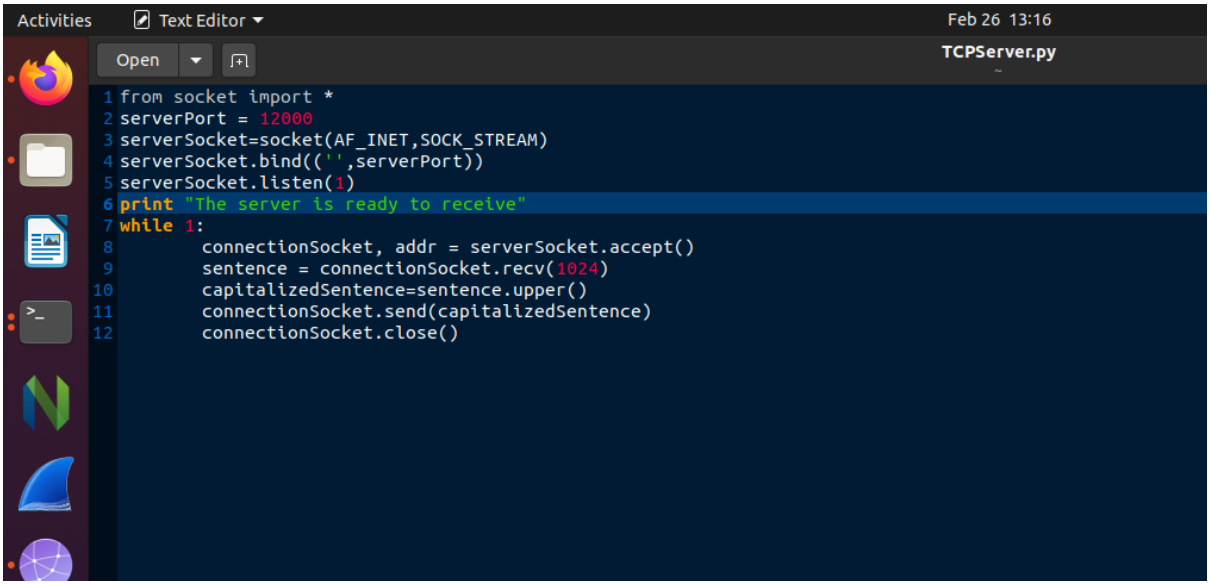


1.2.3 UDP CONNECTION BETWEEN UDP SERVER AND UDP CLIENT

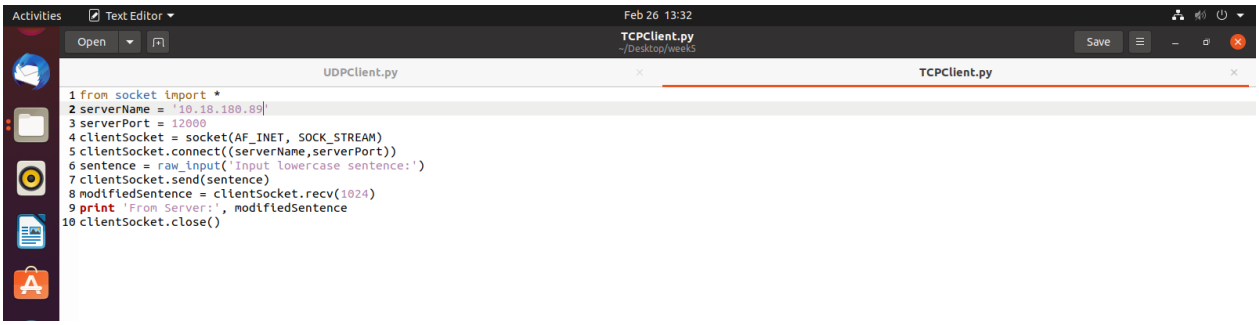

student@pesu-OptiPlex-3070: ~
student@pesu-OptiPlex-3070:~$ python UDPClient.py
Input lowercase sentence:priya mohata
PRIYA MOHATA
student@pesu-OptiPlex-3070:~$
ecvfrom(2048)
student@pesu-OptiPlex-3070: ~
student@pesu-OptiPlex-3070:~$ python UDPServer.py
The server is ready to receive

1.3 TCP CONNECTION in the two systems

1.3.1 TCP SERVER

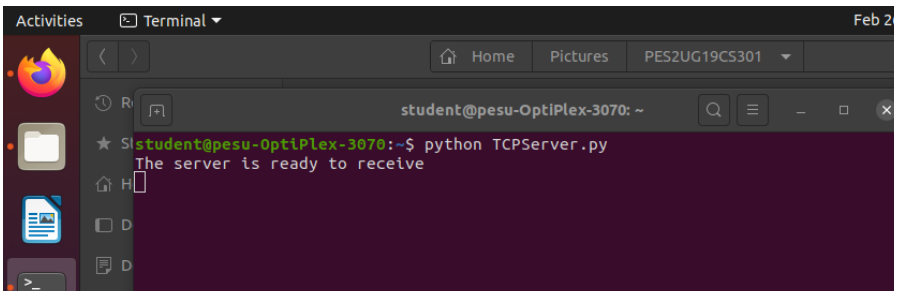


1.3.2 TCP CLIENT

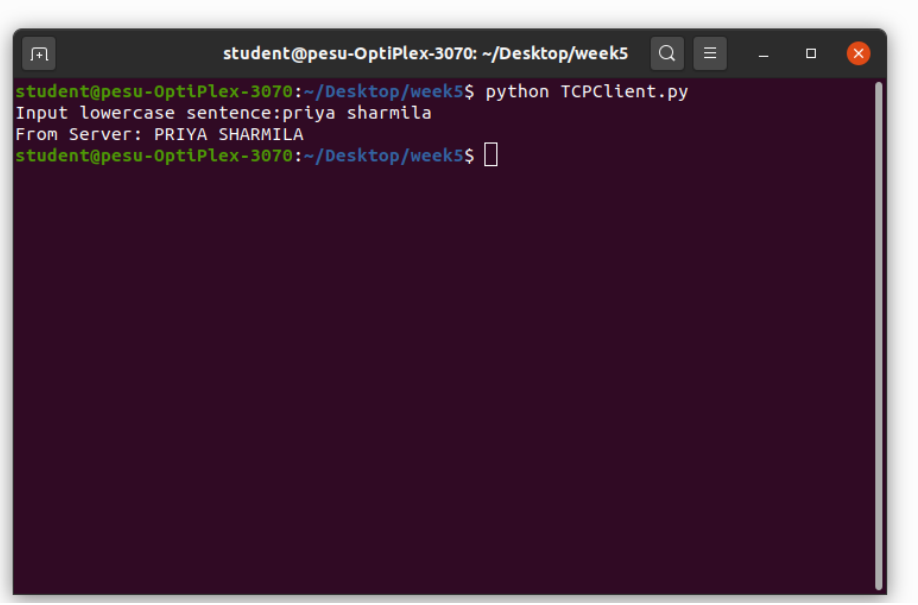


1.3.3 TCP CONNECTION

SERVER

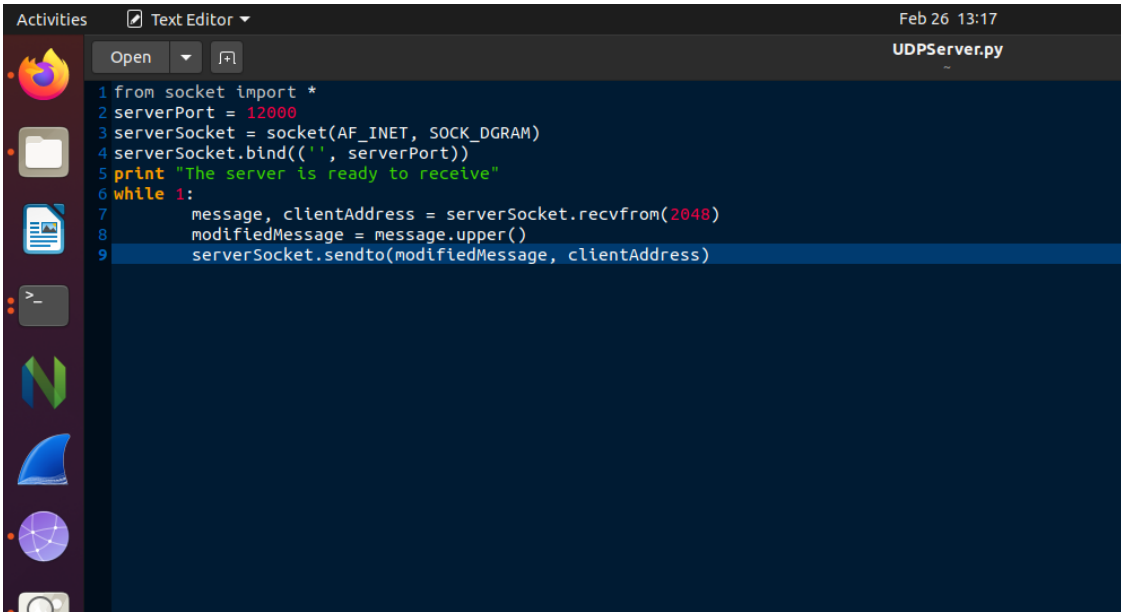


CLIENT

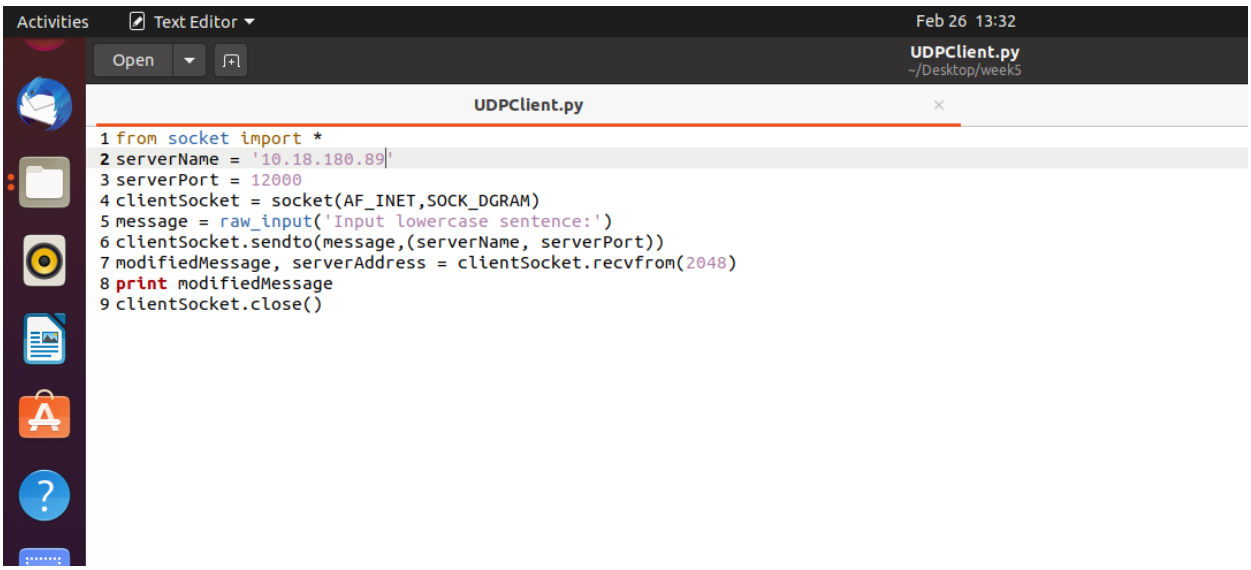

student@pesu-OptiPlex-3070: ~/Desktop/week5
student@pesu-OptiPlex-3070:~/Desktop/week5$ python TCPClient.py
Input lowercase sentence:priya sharmila
From Server: PRIYA SHARMILA
student@pesu-OptiPlex-3070:~/Desktop/week5$

1.4 UDP CONNECTION in two systems

1.4.1 UDP SERVER

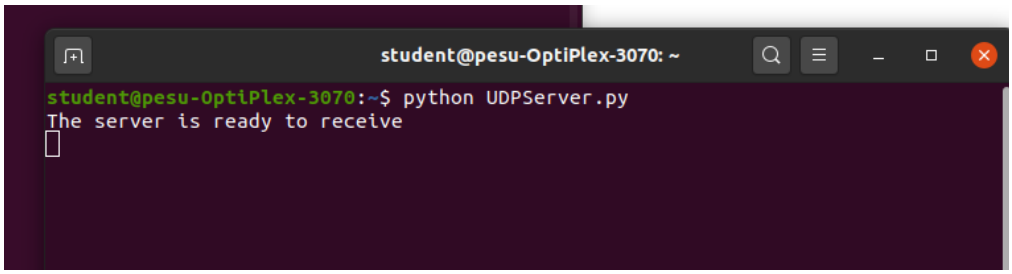


1.4.2 UDP CLIENT

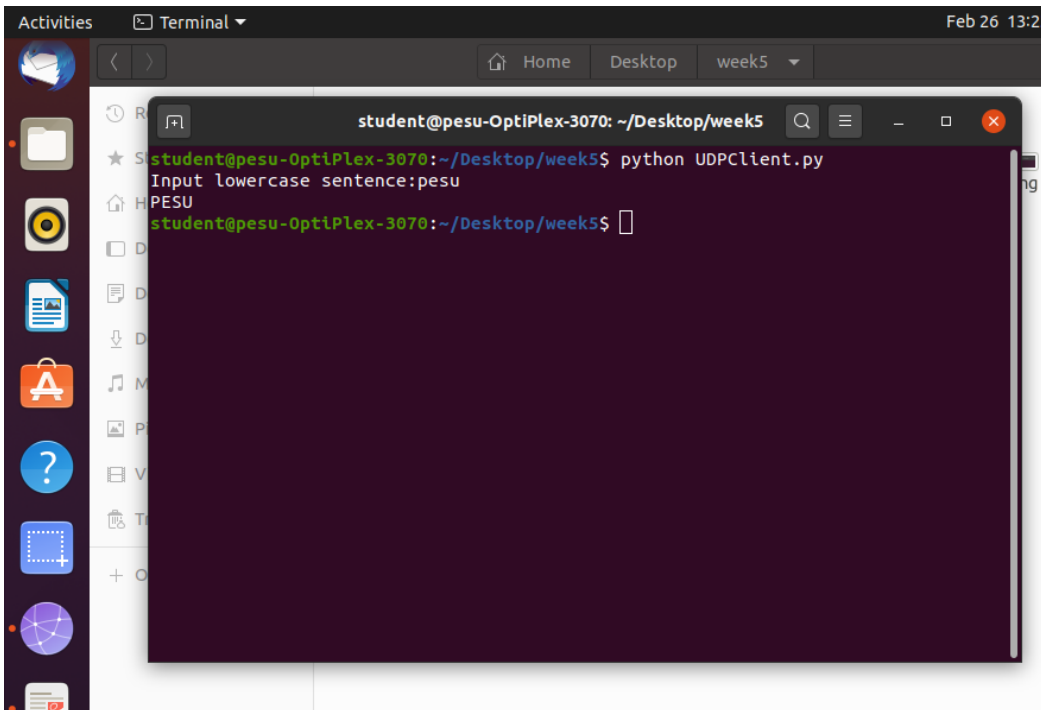


1.4.3 UDP CONNECTION

SERVER



CLIENT


Activities
Terminal
Feb 26 13:2
Home
Desktop
week5
student@pesu-OptiPlex-3070: ~/Desktop/week5
student@pesu-OptiPlex-3070:~/Desktop/week5$ python UDPClient.py
Input lowercase sentence:pesu
PESU
student@pesu-OptiPlex-3070:~/Desktop/week5$

QUESTIONS RELATED TO TASK 1 :

QUESTION 1 : Suppose you run TCPClient before you run TCPServer. What happens? Why?

ANSWER 1: This will lead to a **ConnectionRefusedError,** since the server socket application we are trying to connect to has not been initiated and is not listening for connections on the given port number. Hence, any connection requests sent by a client
machine at that IP and port number immediately fail since the connection gets refused.

```
student@pesu-OptiPlex-3070:~/Desktop/week5$ python TCPClient.py
Traceback (most recent call last):
  File "TCPClient.py", line 5, in <module>
    clientSocket.connect((serverName,serverPort))
  File "/usr/lib/python2.7/socket.py", line 228, in meth
    return getattr(self._sock,name)(*args)
socket.error: [Errno 111] Connection refused
student@pesu-OptiPlex-3070:~/Desktop/week5$
```

QUESTION 2: Suppose you run UDPClient before you run UDPServer. What happens? Why?

ANSWER 2: No error will be obtained since UDP does not require a prior connection to be set up between the host machines for data transfer to begin. It is a connectionless protocol which transfers packets of data to a destination IP and port number without verifying the existence
of the connection.

```
student@pesu-OptiPlex-3070:~/Desktop/week5$ python UDPClient.py
Input lowercase sentence:priya sharmila
PRIYA SHARMILA
student@pesu-OptiPlex-3070:~/Desktop/week5$
```

QUESTION 3: What happens if you use different port numbers for the client and server sides?

ANSWER 3: This will lead to a **ConnectionRefusedError** for a **TCP connection**, since the server socket application we are trying to connect to is not listening for requests at the same port number as the one the client socket application is trying to connect with. Hence,the connection between the two socket interfaces is never setup and the connection is downright refused.However, on a **UDP connection**, since no prior connection is required to be established between the host machines for data transfer to take place, no error as such is obtained. Any messages sent by the client are lost since the destination server does not exist

TASK 2:WEB SERVER

Web server will

a) create a connection socket when contacted by a client (browser);

b) receive the HTTP request from this connection;

c) parse the request to determine the specific file being requested;

d) get the requested file from the server's file system;

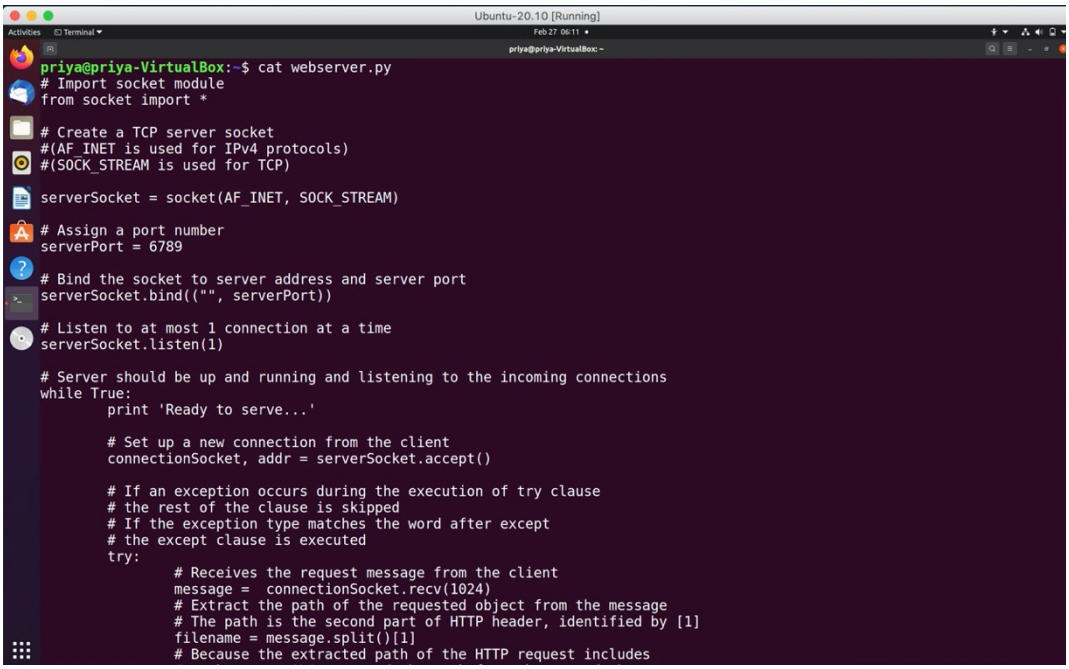
e) create an HTTP response message consisting of the requested file preceded by header

lines; and

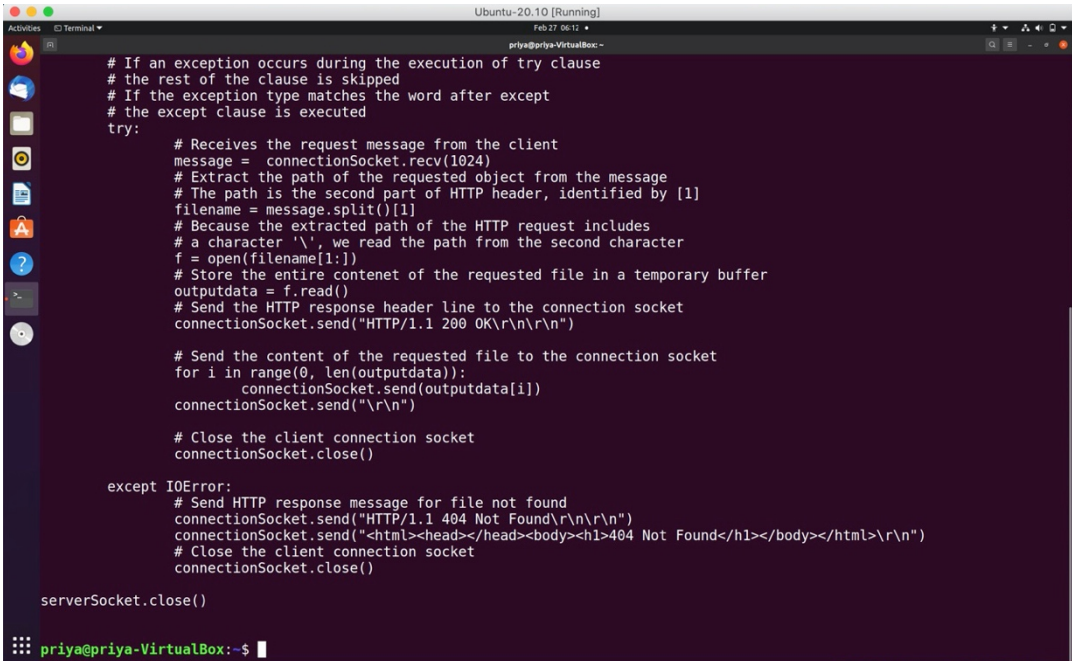
f) send the response over the TCP connection to the requesting browser

If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.

WEBSERVER.py

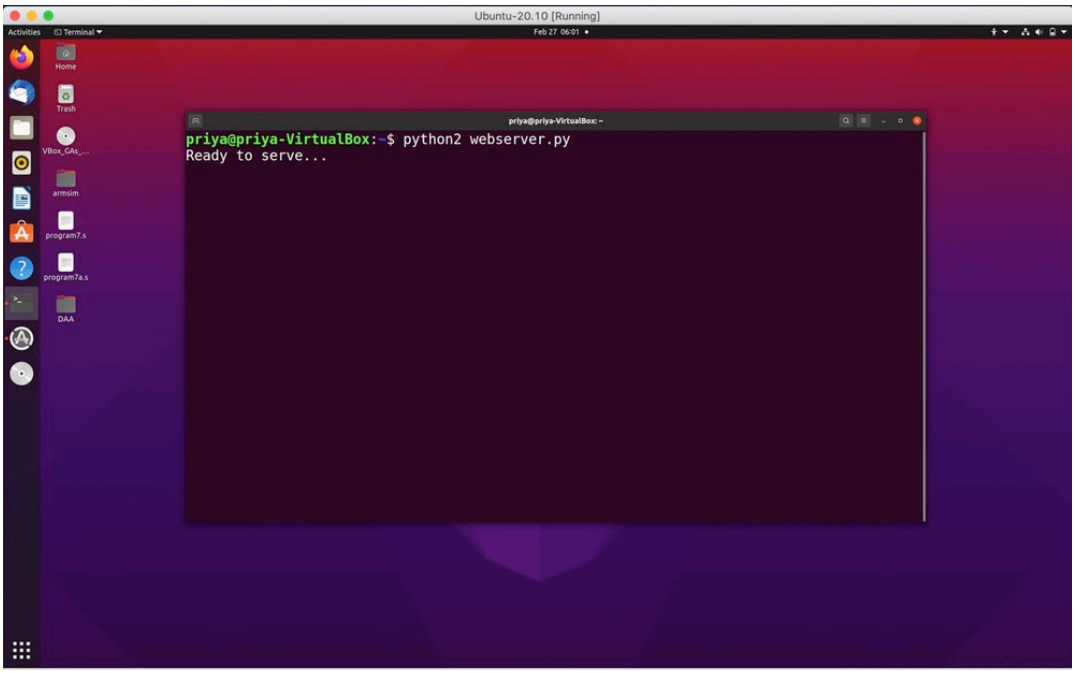


```
priya@priya-VirtualBox:~$ cat webserver.py
# Import socket module
from socket import *

# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = socket(AF_INET, SOCK_STREAM)

# Assign a port number
serverPort = 6789

# Bind the socket to server address and server port
serverSocket.bind(("", serverPort))

# Listen to at most 1 connection at a time
serverSocket.listen(1)

# Server should be up and running and listening to the incoming connections
while True:
        print 'Ready to serve...'

        # Set up a new connection from the client
        connectionSocket, addr = serverSocket.accept()

        # If an exception occurs during the execution of try clause
        # the rest of the clause is skipped
        # If the exception type matches the word after except
        # the except clause is executed
        try:
                # Receives the request message from the client
                message =  connectionSocket.recv(1024)
                # Extract the path of the requested object from the message
                # The path is the second part of HTTP header, identified by [1]
                filename = message.split()[1]
                # Because the extracted path of the HTTP request includes
```



```
        # If an exception occurs during the execution of try clause
        # the rest of the clause is skipped
        # If the exception type matches the word after except
        # the except clause is executed
        try:
                # Receives the request message from the client
                message =  connectionSocket.recv(1024)
                # Extract the path of the requested object from the message
                # The path is the second part of HTTP header, identified by [1]
                filename = message.split()[1]
                # Because the extracted path of the HTTP request includes
                # a character '\', we read the path from the second character
                f = open(filename[1:])
                # Store the entire contenet of the requested file in a temporary buffer
                outputdata = f.read()
                # Send the HTTP response header line to the connection socket
                connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n")

                # Send the content of the requested file to the connection socket
                for i in range(0, len(outputdata)):
                        connectionSocket.send(outputdata[i])
                connectionSocket.send("\r\n")

                # Close the client connection socket
                connectionSocket.close()

        except IOError:
                # Send HTTP response message for file not found
                connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n")
                connectionSocket.send("<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n")
                # Close the client connection socket
                connectionSocket.close()

serverSocket.close()

priya@priya-VirtualBox:~$
```

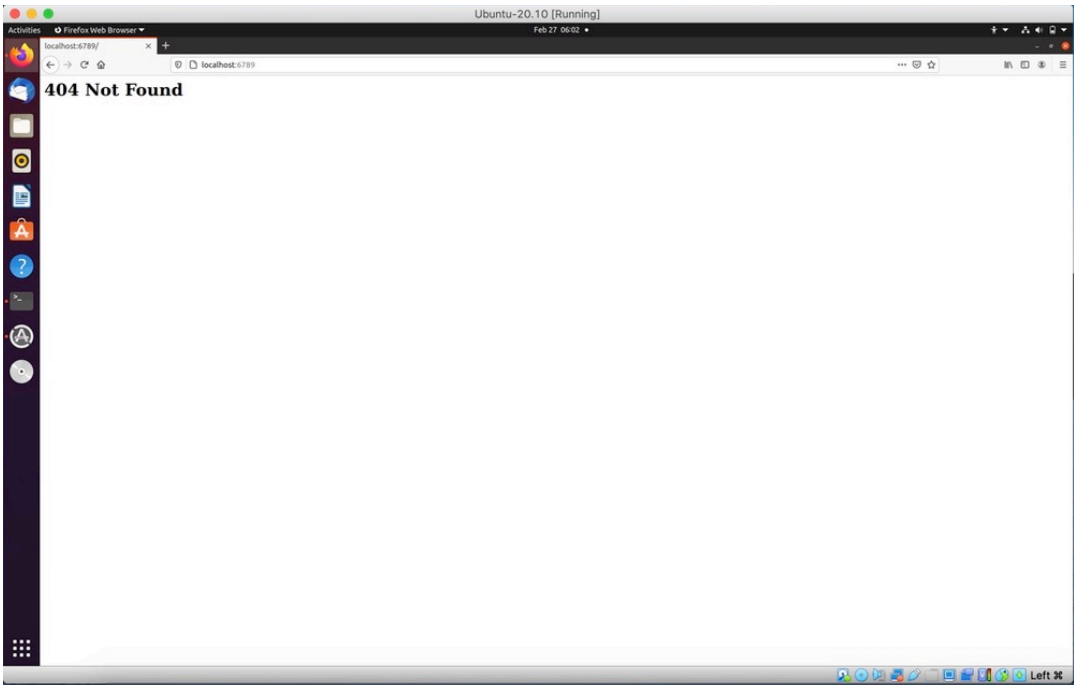
Running the webserver



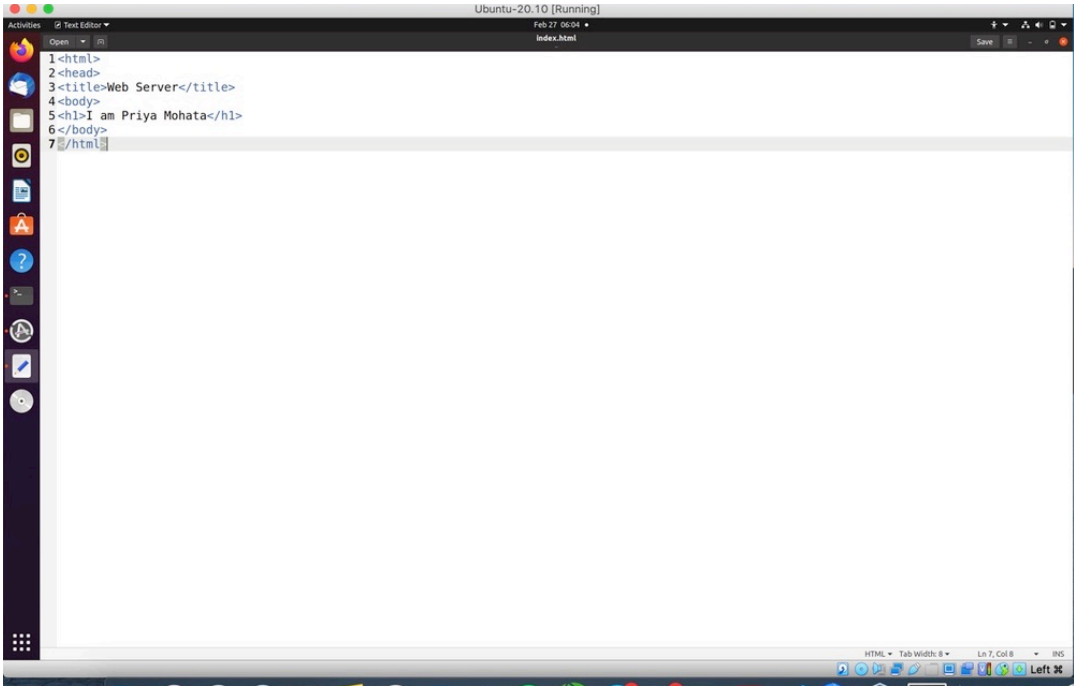
```
priya@priya-VirtualBox:~$ python2 webserver.py
Ready to serve...
```

Accessing a file not present on the server

404 Not Found error is obtained



Creating index.html that will now be accessed



Accessing localhost:6789/index.html


Activities
Firefox Web Browser
Feb 27 06:05
Web Server
localhost:6789/index.html
I am Priya Mohata
Left ⌘

----------Thank you----------