



# CS425

## Assignment IV - Socket Programming

Course Instructor :

Dr. Amitangshu Pal

Submitted By :

Nishant Roshan  
200643

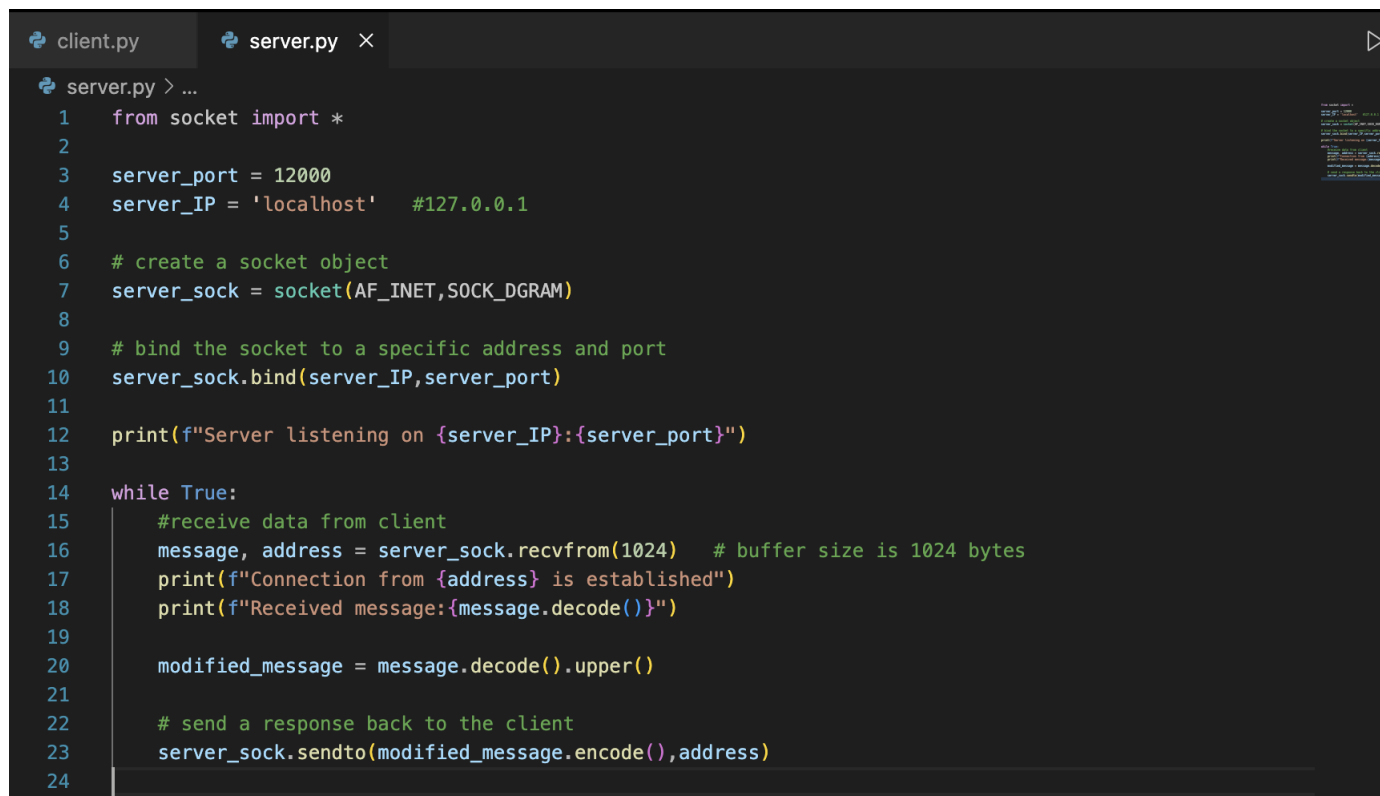
Department of Computer Science and Engineering  
Indian Institute of Technology, Kanpur  
April 19, 2023

## Question

Please write a socket programming for UDP client server application with 1 server and 1 client. You can reuse the code shown in the class and can do some small changes as needed.

## Solution:

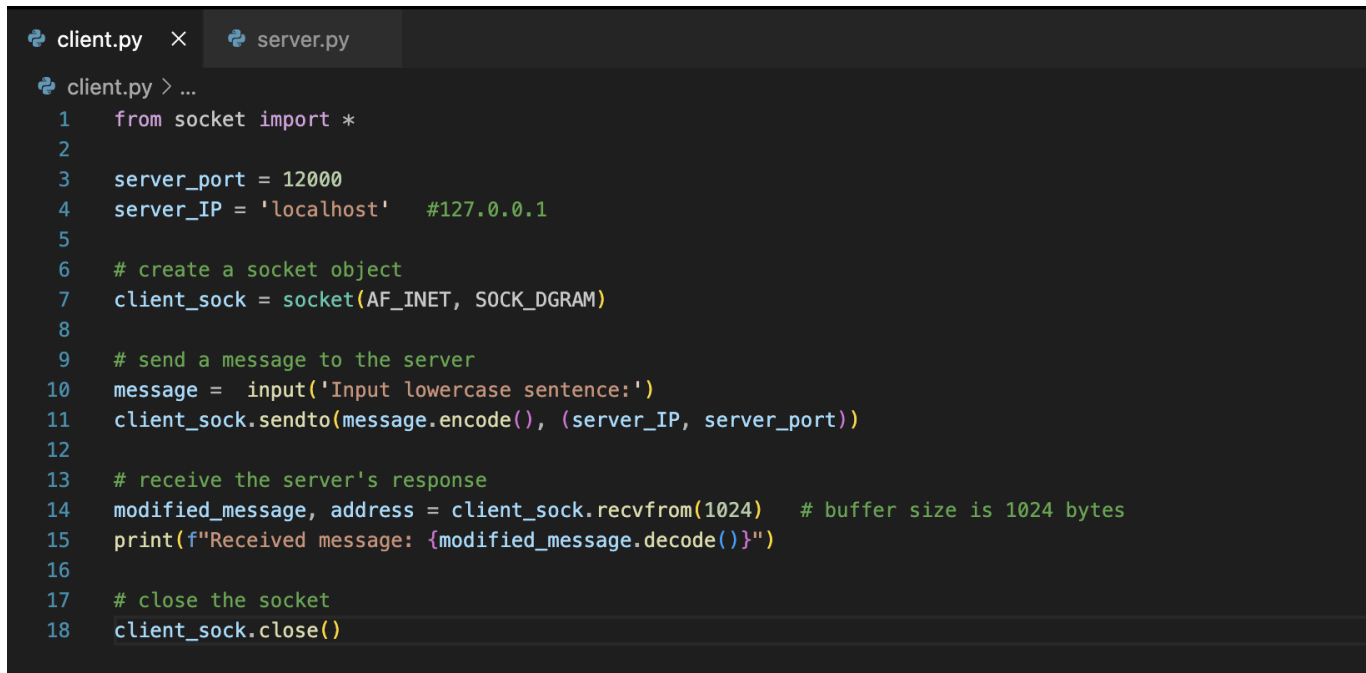
Ans: The following is the server and client code:



```
client.py  server.py ×
server.py > ...
1  from socket import *
2
3  server_port = 12000
4  server_IP = 'localhost'  #127.0.0.1
5
6  # create a socket object
7  server_sock = socket(AF_INET,SOCK_DGRAM)
8
9  # bind the socket to a specific address and port
10 server_sock.bind(server_IP,server_port)
11
12 print(f"Server listening on {server_IP}:{server_port}")
13
14 while True:
15     #receive data from client
16     message, address = server_sock.recvfrom(1024)  # buffer size is 1024 bytes
17     print(f"Connection from {address} is established")
18     print(f"Received message:{message.decode()}")
19
20     modified_message = message.decode().upper()
21
22     # send a response back to the client
23     server_sock.sendto(modified_message.encode(),address)
24
```

Figure 1:

In the above code, the server listens on port 12000 for incoming messages from the client. When it receives a message, it prints the message to the console and sends a response back to the client with the message in uppercase of the message received.



```
client.py × server.py
client.py > ...
1  from socket import *
2
3  server_port = 12000
4  server_IP = 'localhost'  #127.0.0.1
5
6  # create a socket object
7  client_sock = socket(AF_INET, SOCK_DGRAM)
8
9  # send a message to the server
10 message = input('Input lowercase sentence:')
11 client_sock.sendto(message.encode(), (server_IP, server_port))
12
13 # receive the server's response
14 modified_message, address = client_sock.recvfrom(1024)  # buffer size is 1024 bytes
15 print(f"Received message: {modified_message.decode()}")
16
17 # close the socket
18 client_sock.close()
```

Figure 2:

In the above code, the client sends to the server the message it receives from the command line and then waits to receive a response. When it receives a response from the server, it prints the message to the console.

*Thank You*