

Instructions for 327 Assignment 5:

By: Dylan Swanson and Teresa Nguyen

Submitted by: Dylan Swanson

Compiling and running the project:

Step 1: Open the project. Ensure that you are in the root directory of the project folder.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● dswanson@Dylans-MBP 327Assignment5 % ls
  echo_client.py  echo_server.py  venv
● dswanson@Dylans-MBP 327Assignment5 % pwd
  /Users/dswanson/Downloads/327Assignment5
○ dswanson@Dylans-MBP 327Assignment5 %
```

Step 2: Run the server-side script first. This is to ensure that the client can connect to an existing server.

Run the command:

python3 echo_server.py

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
○ dswanson@Dylans-MBP 327Assignment5 % python3 echo_server.py
```

Step 3: Enter the IP address that you would like the server to be accessible at and the Port number in which you would like the server to be open on.

In this case we will use:

IP address = localhost or (127.0.0.1)

Port Number = 3000

```
dswanson@Dylans-MBP 327Assignment5 % python3 echo_server.py
Enter the IP address for the server to bind to (or 'localhost'): localhost
Enter the port number for the server to bind to: 3000
Server started on localhost:3000, waiting for connections...
```

Step 4: Now your server is running. Next you should start your client-side script.

Run the command:

```
python3 echo_client.py
```

```
dswanson@Dylans-MBP 327Assignment5 % python3 echo_client.py
```

Step 5: Enter the server IP address and port number, then a message you would like to send to the server. Since we ran the server on *localhost:3000* we will use *localhost* for the IP address and *3000* for the port number.

In this case we will use:

IP address = localhost

Port number = 3000

Message = Hello World!

```
dswanson@Dylans-MBP 327Assignment5 % python3 echo_client.py
Enter the server IP address: localhost
Enter the server port number: 3000
Enter the message to send (or 'exit' to quit): Hello World!
```

Step 6: If everything is configured properly then you should be able to see the response from the server. The response will be your same message but in all CAPITAL letters.

```
Enter the message to send (or 'exit' to quit): Hello World!
Server response: HELLO WORLD!
Enter the message to send (or 'exit' to quit): s
```

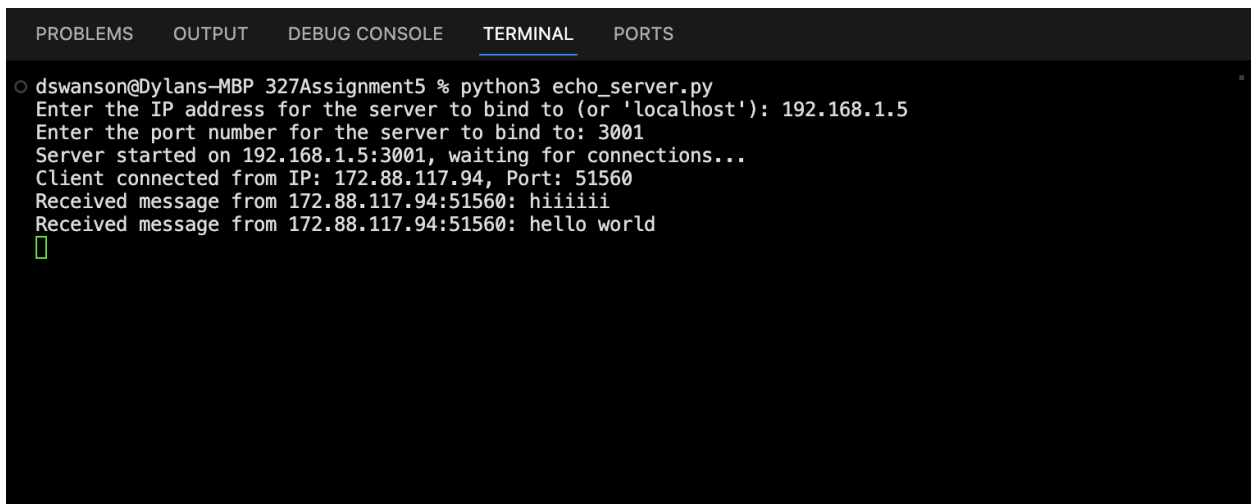
Step 7: Complete. You are now able to use the Echo client-server application!

Sending a message with partners IP

Since my partner and I were on different networks we needed to first set up *port forwarding* on our home routers. Since I have a spectrum router this was fairly easy to do through the app. To perform this, I had to select my device, set the external and internal route, then set the protocol to TCP.

Once *port forwarding* was configured I just needed to run my server at <my-private-ip>:3001 and share my public IP address and the open port with my partner. In this case my public ip address was 97.90.35.195 and my port was 3001.

I could then see that my partner was connected and able to send me messages.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ dswanson@Dylans-MBP 327Assignment5 % python3 echo_server.py
Enter the IP address for the server to bind to (or 'localhost'): 192.168.1.5
Enter the port number for the server to bind to: 3001
Server started on 192.168.1.5:3001, waiting for connections...
Client connected from IP: 172.88.117.94, Port: 51560
Received message from 172.88.117.94:51560: hiiiii
Received message from 172.88.117.94:51560: hello world
█
```

You can see that my partner was connected to my server with their public ip 172.88.117.94 and was able to send messages.

We performed the same steps so that I could communicate with my partner's server as well.

The screenshot displays the PyCharm IDE interface. The top toolbar includes icons for running, debugging, and other development actions. The main editor window shows the file `echo_server.py` with the following Python code:

```
1 server_port = int(input("Enter the port number for the server to bind to: "))
2
3 # Create a TCP socket
4 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5
6 # Bind the socket to the provided IP address and port
7 server_socket.bind((server_ip, server_port))
8
9 # Listen for incoming connections
10 server_socket.listen(1)
11 print(f"Server started on {server_ip}:{server_port}, waiting for connections...")
12
13 while True:
14     # Accept a client connection
15     client_socket, client_address = server_socket.accept()
16     client_ip, client_port = client_address # Unpack the client address tuple
17
18     print(f"Client connected from IP: {client_ip}, Port: {client_port}")
19
20 echo_server()
```

The bottom panel shows the output of the `echo_server` process:

```
Run: echo_server
/Users/teresanguyen/PycharmProjects/327Assignment5/venv/bin/python /Users/teresanguyen/PycharmProjects/327_Assignment5/echo_server.py
Enter the IP address for the server to bind to (or 'localhost'): 192.168.0.119
Enter the port number for the server to bind to: 3001
Server started on 192.168.0.119:3001, waiting for connections...
Client connected from IP: 97.90.35.195, Port: 49219
Received message from 97.90.35.195:49219: Heyyyyyy
Received message from 97.90.35.195:49219: thank god this works
Received message from 97.90.35.195:49219: 10101asdfsadf
Connection with 97.90.35.195:49219 closed.
```