Running The UDP Server & Client Together

We've spent the last few lessons writing code for a very basic client and a server. Let's see these in action in this lesson!

WE'LL COVER THE FOLLOWING ^

Connecting the Two

Connecting the Two

We'll run both together in one file called udp.py instead of running them separately. We've written some python code in the main function that allows you to specify which function you want the code to run, the server or the client.

To run the code, you would need to follow these steps:

- 1. Type your code and when you are ready to run the program, click on **Run**. The server code should start up automatically.
- 2. Open another terminal by clicking on +
- 3. Type the command python3 /usercode/udp.py client Note that it can be server in place of client.
- 4. Enter the text in the client window and see the effect.
- 5. If the program is not running to your satisfaction:
 - 1. Kill the running server program by typing the break sequence ctrl+c or command+c in both of the terminal windows.
 - 2. Change the code
 - 3. Click on Run.
 - 4. Type command python3 /usercode/udp.py server and python3 /usercode/udp.py client in the first and second terminal window, respectively. Go back to step 4.

Every time you make a change to the code you must click **run** for the changes to take effect.

```
import argparse, socket
MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram
def server(port):
   s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hostname = '127.0.0.1'
    s.bind((hostname, port))
    print('Listening at {}'.format(s.getsockname()))
    while True:
        data, clientAddress = s.recvfrom(MAX_SIZE_BYTES)
        message = data.decode('ascii')
        upperCaseMessage = message.upper()
        print('The client at {} says {!r}'.format(clientAddress, message))
        data = upperCaseMessage.encode('ascii')
        s.sendto(data, clientAddress)
def client(port):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    message = input('Input lowercase sentence:' )
    data = message.encode('ascii')
    s.sendto(data, ('127.0.0.1', port))
    print('The OS assigned the address {} to me'.format(s.getsockname()))
    data, address = s.recvfrom(MAX_SIZE_BYTES)
    text = data.decode('ascii')
    print('The server {} replied with {!r}'.format(address, text))
if __name__ == '__main__':
   funcs = {'client': client, 'server': server}
    parser = argparse.ArgumentParser(description='UDP client and server')
    parser.add_argument('functions', choices=funcs, help='client or server')
    parser.add_argument('-p', metavar='PORT', type=int, default=3000,
                        help='UDP port (default 3000)')
    args = parser.parse_args()
    function = funcs[args.functions]
    function(args.p)
```

In the next lesson, we're going to look at some possible improvements to our current UDP program.