

# Writing a UDP Client Program

Let's now write a client to go with the server we wrote.

## WE'LL COVER THE FOLLOWING ^

- The Server
- Creating a Client Socket
- Reading Data
- Sending It to the Server
- Receiving the Server's Response
- Decoding & Printing the Capitalized Message

## The Server #

Here's the server code that we have so far for reference.

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

# Setting up a socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 3000
hostname = '127.0.0.1'
s.bind((hostname, port))
print('Listening at {}'.format(s.getsockname()))
while True:
    data, address = 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)
```

## Creating a Client Socket #

Instead of explicitly binding the socket to a given port and IP as we did previously, we can let the OS take care of it. Remember [ephemeral ports](#)? Yes

the OS will bind the socket to a port dynamically. So all we really need is to create a UDP socket (**line 3**).

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

In fact, we can check what address and port the OS assigned to the socket using the following line of code on **line 4**:

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print('The OS assigned the address {} to me'.format(s.getsockname()))
```

You'll always get `(0.0.0.0,0)` for now because we haven't actually used the socket. You'll get the correct answer when we use the socket to send data in the next lesson.

## Reading Data #

Remember that the goal of this client and server was for the client to send a string to the server that it would capitalize and send back? Well, we'll get that string from the user's keyboard using the `Python3` function, `input()`. The function displays whatever prompt we specify, in this case the message 'Input lowercase sentence:' and waits for the user to provide an input from the keyboard (**line 4**). The user can then type a string of their choice and hit enter. The string will be stored in the variable `message`.

Next, we encode the ASCII encoded data to bytes using the `encode()` function (**line 5**).

```
import socket
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
message = input('Input lowercase sentence: ' )
data = message.encode('ascii')
```

## Sending It to the Server #

We now send the message to the server using the `sendto()` function. In addition to the data, this function takes an IP address and a port number (**line 6**). We give it the IP address `127.0.0.1` and the port `3000` which we assigned to the server previously.

Also, notice that the `getsockname()` function will give us a useful answer at this point (**line 7**).

```
import socket

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', 3000))
print('The OS assigned the address {} to me'.format(s.getsockname()))
```

## Receiving the Server's Response #

We next receive the server's response and limit it to the size `MAX_SIZE_BYTES` which we saw previously.

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

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', 3000))
print('The OS assigned the address {} to me'.format(s.getsockname()))
data, address = s.recvfrom(MAX_SIZE_BYTES)
```

## Decoding & Printing the Capitalized Message #

Lastly, we decode and print the capitalized message that the server sent us.

```
import socket

MAX_SIZE_BYTES = 65535 # Mazimum size of a UDP datagram

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
message = input('Input lowercase sentence: ' )
data = message.encode('ascii')
```

```
data = message.encode('ascii')
s.sendto(data, ('127.0.0.1', 3000))
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))
```

---

Now we have both our server and client programs. Let's see them live in action next!