

Assignment 2

Introduction to web science

OSCAR:
Abdullah Elkindy
Fiorela Ciroku
Stela Nebiaj

1. IP Packet

Consider the IPv4 packet that is received as: 4500 062A 42A1 8001 4210 XXXX C0A8 0001 C0A8 0003. Consider XXXX to be the check sum field that needs to be sent with the packet. Please provide a step-by-step process for calculating the "Check Sum".

To calculate the "Check Sum" first we have to convert the numbers to binary and start adding them all. In case the sum is more than 16 bits, we drop the 1 on the left and add 1 to the number that we have. XXXX should be 2ECD.

| | |
|------|----------------------|
| 4500 | 0100 0101 0000 0000 |
| 062A | +0000 0110 0010 1010 |
| 4B2A | 0100 1011 0010 1010 |

| | |
|------|----------------------|
| 4B2A | 0100 1011 0010 1010 |
| 42A1 | +0100 0010 1010 0001 |
| 8DCB | 1000 1101 1100 1011 |

| | |
|-------|-----------------------|
| 8DCB | 1000 1101 1100 1011 |
| 8001 | +1000 0000 0000 0001 |
| 10DCC | 1 0000 1101 1100 1100 |

| | |
|-------|-----------------------|
| 10DCC | 1 0000 1101 1100 1100 |
| New | 0000 1101 1100 1101 |

| | |
|------|----------------------|
| DCD | 0000 1101 1100 1101 |
| 4210 | +0100 0010 0001 0000 |
| 4FDD | 0100 1111 1101 1101 |

| | |
|------|----------------------|
| 4FDD | 0100 1111 1101 1101 |
| XXXX | +0000 0000 0000 0000 |
| 4FDD | 0100 1111 1101 1101 |

| | |
|-------|-----------------------|
| 4FDD | 0100 1111 1101 1101 |
| C0A8 | +1100 0000 1010 1000 |
| 11085 | 1 0001 0000 1000 0101 |

| | |
|----------|-----------------------|
| 11085 | 1 0001 0000 1000 0101 |
| New | 0001 0000 1000 0110 |
| 1086 | 0001 0000 1000 0110 |
| 0001 | +0000 0000 0000 0001 |
| 1087 | 0001 0000 1000 0111 |
| 1087 | 0001 0000 1000 0111 |
| C0A8 | +1100 0000 1010 1000 |
| D12F | 1101 0001 0010 1111 |
| D12F | 1101 0001 0010 1111 |
| 0003 | +0000 0000 0000 0011 |
| D132 | 1101 0001 0011 0010 |
| checksum | 0010 1110 1100 1101 |
| 2ECD | 0010 1110 1100 1101 |

2. Routing Algorithm

You have seen how routing tables can be used to see how the packets are transferred across different networks. Using the routing tables below of Router 1, 2 and 3.

Draw the network. Find the shortest path of sending information from 67.68.2.10 network to 25.30.3.13 network.

Router 1

| Destination | Next Hop | Interface |
|-------------|-------------|-----------|
| 67.0.0.0 | 67.68.3.1 | eth 0 |
| 62.0.0.0 | 62.4.31.7 | eth 1 |
| 88.0.0.0 | 88.4.32.6 | eth 2 |
| 141.0.0.0 | 141.30.20.1 | eth 3 |
| 26.0.0.0 | 141.71.26.3 | eth 3 |
| 150.0.0.0 | 141.71.26.3 | eth 3 |
| 205.0.0.0 | 141.71.26.3 | eth 3 |
| 25.0.0.0 | 88.6.32.1 | eth 2 |
| 121.0.0.0 | 88.6.32.1 | eth 2 |

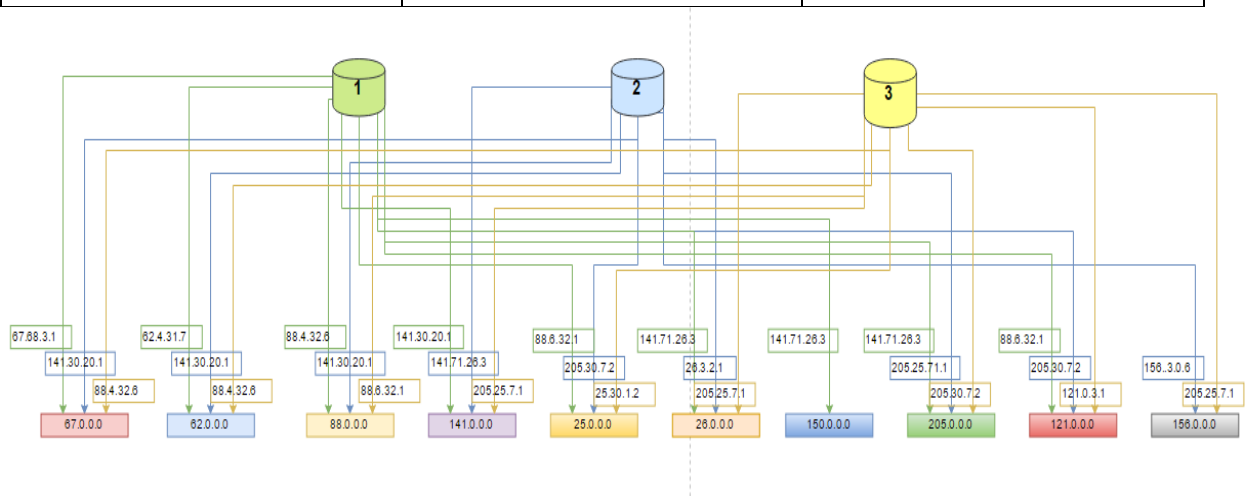
Router 2

| Destination | Next Hop | Interface |
|-------------|-------------|-----------|
| 141.0.0.0 | 141.71.26.3 | eth 3 |
| 205.0.0.0 | 205.25.71.1 | eth 0 |
| 26.0.0.0 | 26.3.2.1 | eth 2 |
| 156.0.0.0 | 156.3.0.6 | eth 1 |
| 67.0.0.0 | 141.30.20.1 | eth 3 |
| 62.0.0.0 | 141.30.20.1 | eth 3 |
| 88.0.0.0 | 141.30.20.1 | eth 3 |
| 25.0.0.0 | 205.30.7.2 | eth 0 |

| | | |
|-----------|------------|-------|
| 121.0.0.0 | 205.30.7.2 | eth 0 |
|-----------|------------|-------|

Router 3

| Destination | Next Hop | Interface |
|-------------|------------|-----------|
| 205.0.0.0 | 205.30.7.2 | eth 0 |
| 88.0.0.0 | 88.6.32.1 | eth 1 |
| 25.0.0.0 | 25.30.1.2 | eth 2 |
| 121.0.0.0 | 121.0.3.1 | eth 3 |
| 156.0.0.0 | 205.25.7.1 | eth 0 |
| 26.0.0.0 | 205.25.7.1 | eth 0 |
| 141.0.0.0 | 205.25.7.1 | eth 0 |
| 67.0.0.0 | 88.4.32.6 | eth 1 |
| 62.0.0.0 | 88.4.32.6 | eth 1 |



We think that a path to go from 67.68.2.10 to 25.30.3.13 is this:

67.68.2.10 goes to Router 1 using 67.68.3.1. From there the router checks the destination address and knows that this package has to go to a network 25.0.0.0. The route to go that way is using 88.6.32.1. This route can take us to Router 2, where the router decides that the package should go towards 25.0.0.0 using 205.30.7.2. Now we have come to Router 3. This router checks and decides that the package should take 25.30.1.2 to go to the 25.0.0.0 network. There it can find the right address of destination.

3. Sliding Window Protocol

Sliding window algorithm, which allows a sender to have more than one unacknowledged packet "in flight" at a time, improves network throughput. Let us consider you have 2 Wide Area Networks. One with a bandwidth of 10 Mbps (Delay of 20 ms) and the other with 1 Mbps (Delay of 30 ms). If a packet is considered to be of size 10kb. Calculate the window size and number of packets necessary for Sliding Window Protocol.

Since you now understand the concept of Window Size for Sliding Window Protocol and how to calculate it, consider a window size of 3 packets and you have 7 packets to send. Draw the process of Selective Repeat Sliding Window Protocol where in the 3rd packet from the sender is lost while transmission. Show diagrammatically how the system reacts when a packet is not received and how it recuperates from that scenario.

Wide Area Network A

It takes the window size (W), 20 ms to each from A to B.

Our network's Bandwidth is 10Mbps so it means it can withstand 10Mb in one second.

20ms \rightarrow W

1 s \rightarrow 10Mb

$W = 20m * 10M = 200kb$

This means we will send N of packets = $W/10k = 200kb/10kb = 20$ packets from A to B in one window.

Wide Area Network B

It takes the window size (W), 30 ms to each from B to A.

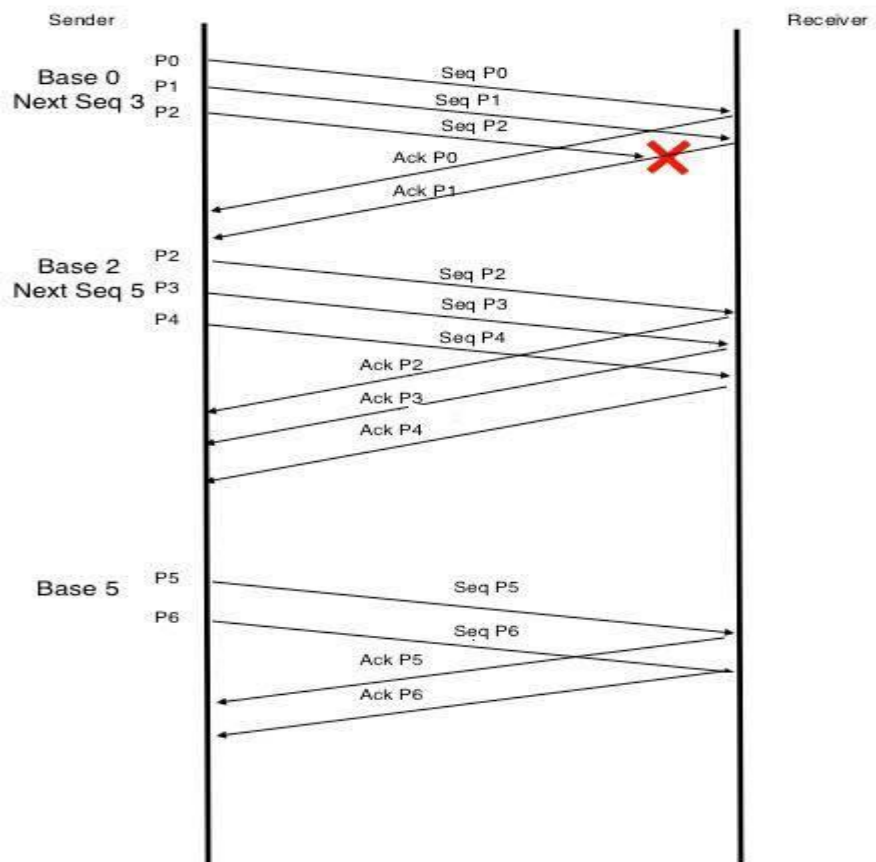
Our network's Bandwidth is 1Mbps so it means it can withstand 1Mb in one second.

30ms \rightarrow W

1 s \rightarrow 1Mb

$W = 30m * 1M = 30kb$

This means we will send N of packets = $W/10k = 30kb/10kb = 3$ packets from B to A in one window.



When packet 2 is lost, The sender waits for sometime and when it receives no ack for 2 it simply resends it.

4. TCP Client Server

Use the information from the <https://docs.python.org/3/howto/sockets.html> documentation and create a simple TCP Server that listens to a Client. Note: Please use port 8080 for communication on localhost for client server communication.

Given below are the following points that your client and server must perform:

The Client side asks the user to input their name, age & matrikelnummer which is then sent to the server all together.

Develop a protocol for sending these three information and subsequently receiving each of the information in three different lines as mentioned in the below format. Provide reasons for the protocol you implemented.

Format the output in a readable format as: Name: Korok Sengupta; Age: 29; Matrikelnummer: 21223ert56.

Server code

```
#Import socket module
import socket
#Create a socket object
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#Get local machine name
host = socket.gethostname()
port = 8080

#Bind to the port
sock.bind((host, port))

#Now wait for client connection.
sock.listen(5)

#Establish connection with client.
connection, client_addr = sock.accept()

print ('Got connection from', client_addr)
while 1:
    data= connection.recv(1024)
    if not data: break
    print('Received data: ',data)
    connection.send(data)
    connection.close()
```

Provide a snapshot of the results along with the code. \\

Client Code

```
#Import socket module
import socket
#Create a socket object
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```

#Get local machine name
host = socket.gethostname()
port = 8080

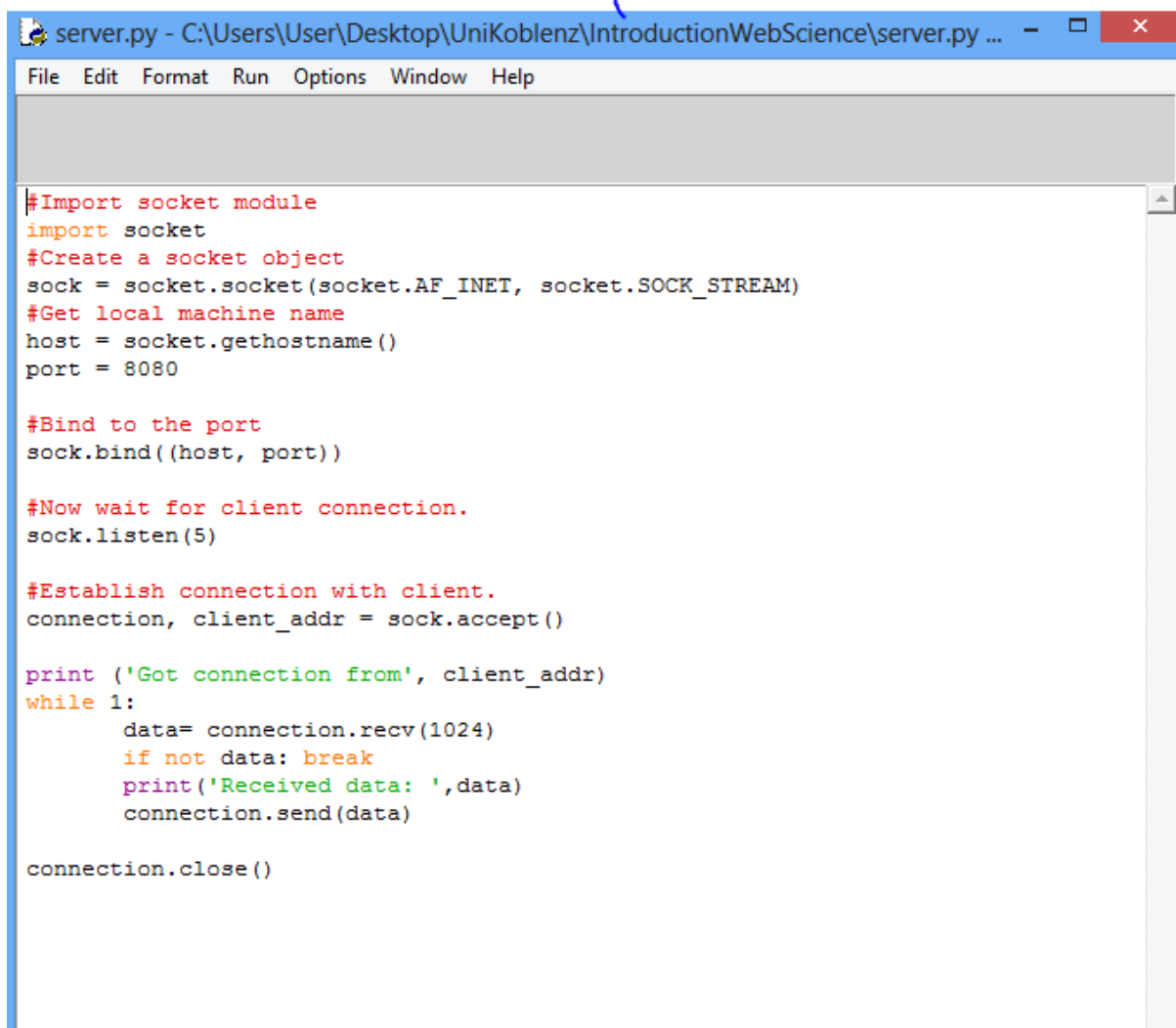
sock.connect((host, port))
name= input("Give your name.")
age= input("Give your age.")
matrikelnummer = input("Give your matrikelnummer.")
message = "Name: "+name+"\n Age: "+age+"\n Matrikelnummer: "+matrikelnummer

sock.send(message.encode('utf-8'))
data = sock.recv(1024)
sock.close
print('Received data: ',data)

\makefooter

\end{document}

```



```

server.py - C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\server.py ...
File Edit Format Run Options Window Help

#Import socket module
import socket
#Create a socket object
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#Get local machine name
host = socket.gethostname()
port = 8080

#Bind to the port
sock.bind((host, port))

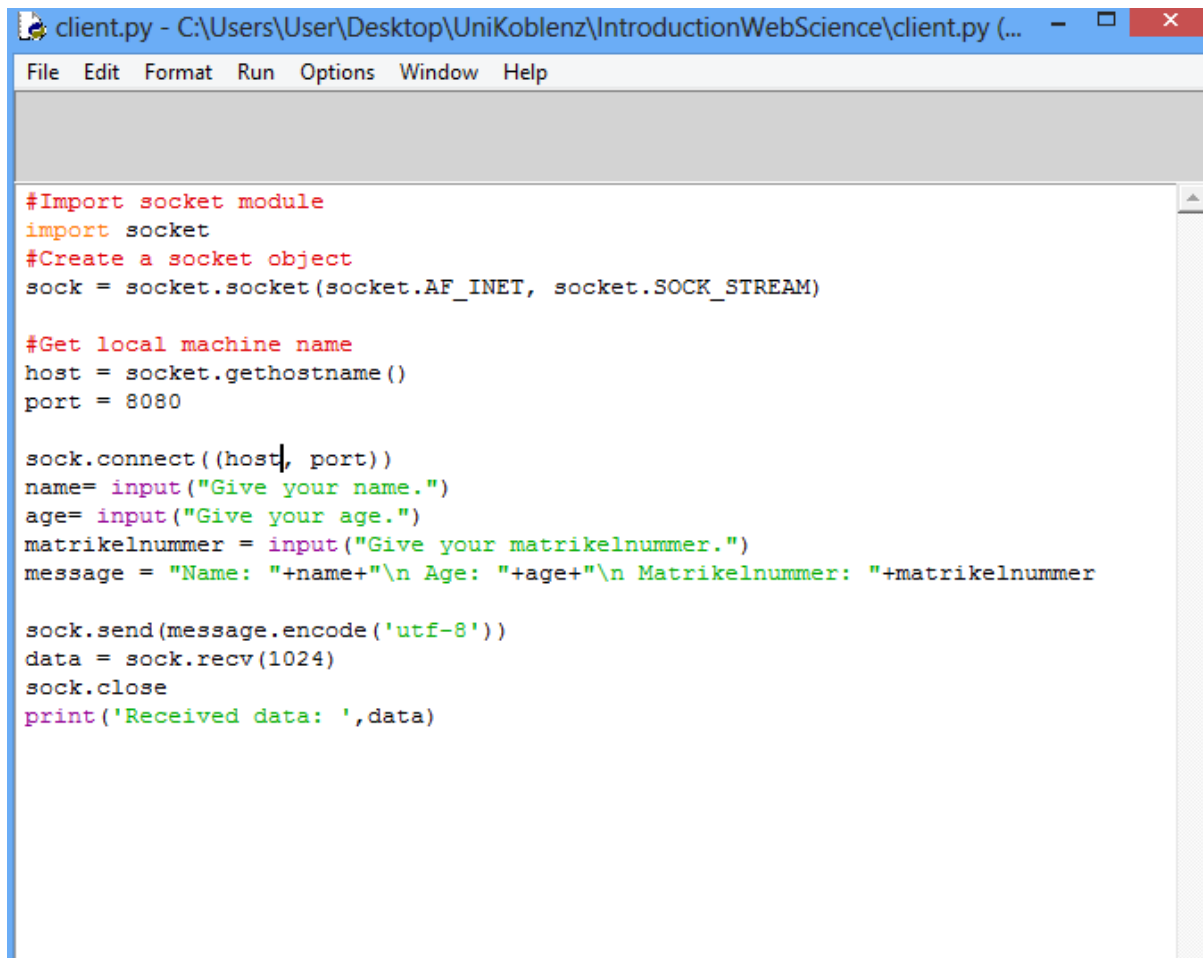
#Now wait for client connection.
sock.listen(5)

#Establish connection with client.
connection, client_addr = sock.accept()

print ('Got connection from', client_addr)
while 1:
    data= connection.recv(1024)
    if not data: break
    print('Received data: ',data)
    connection.send(data)

connection.close()

```



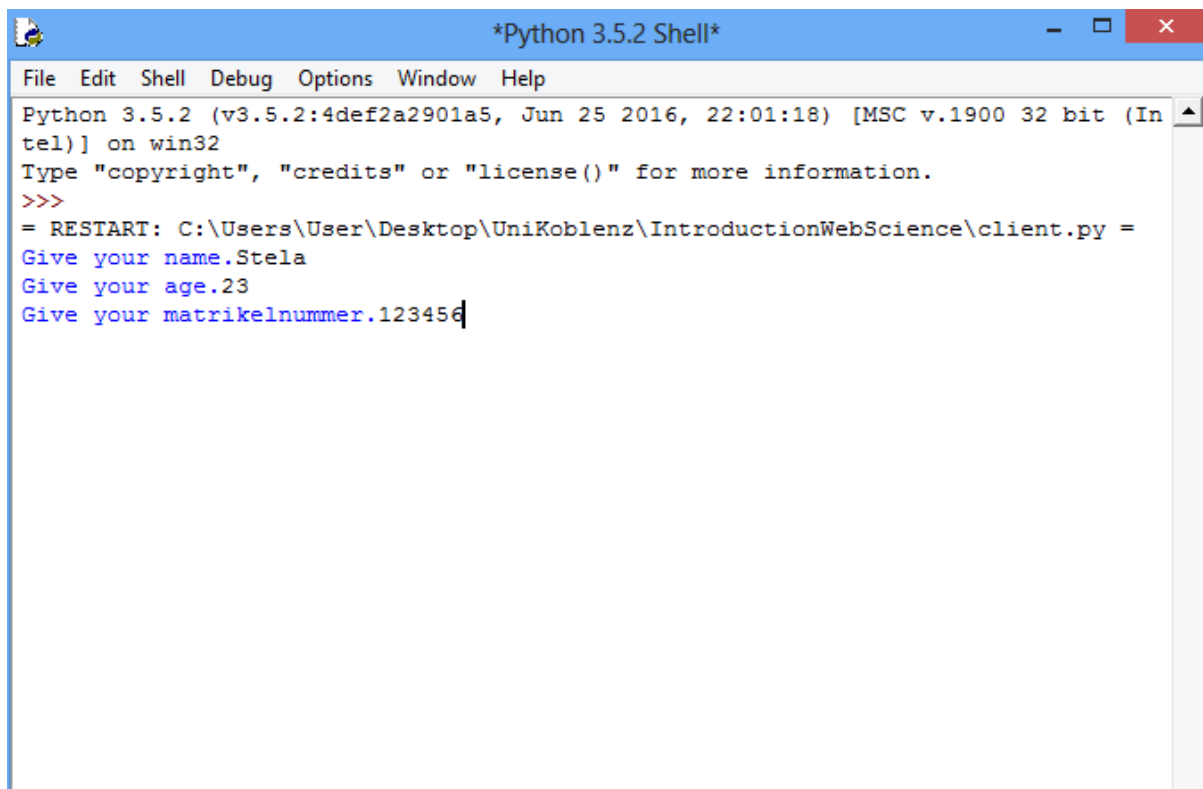
```
client.py - C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\client.py (... - □ ×
File Edit Format Run Options Window Help

#Import socket module
import socket
#Create a socket object
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

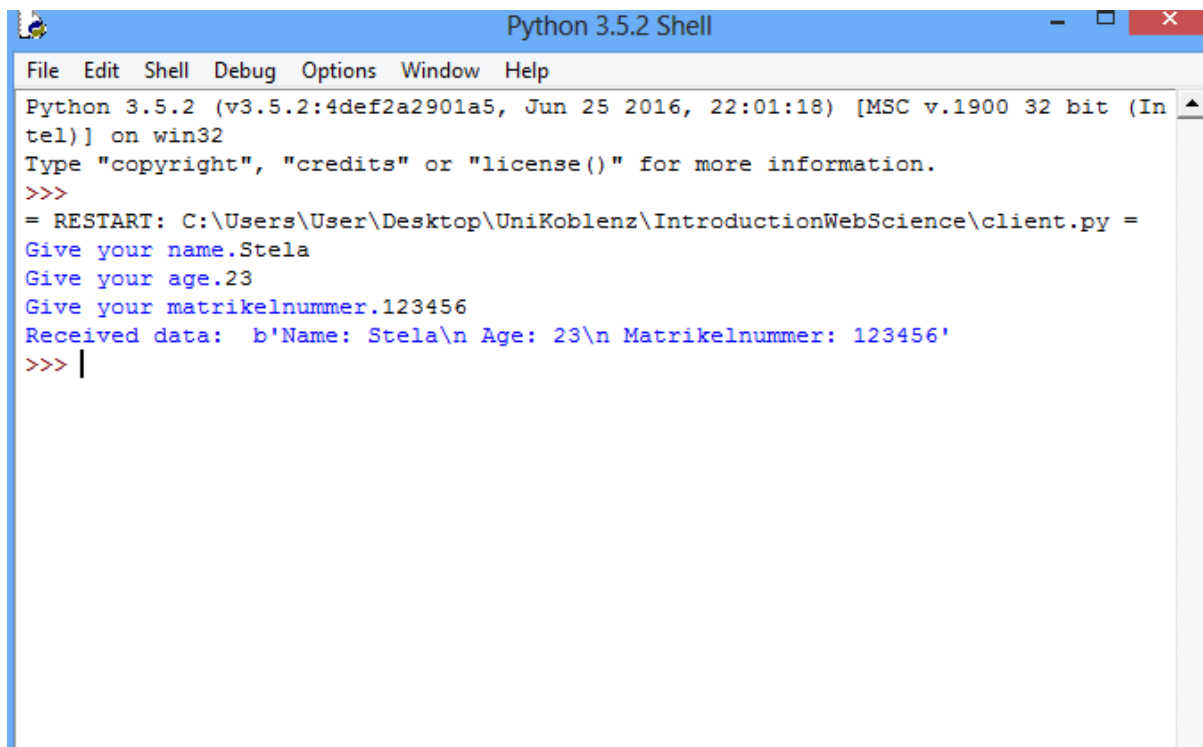
#Get local machine name
host = socket.gethostname()
port = 8080

sock.connect((host, port))
name= input("Give your name.")
age= input("Give your age.")
matrikelnummer = input("Give your matrikelnummer.")
message = "Name: "+name+"\n Age: "+age+"\n Matrikelnummer: "+matrikelnummer

sock.send(message.encode('utf-8'))
data = sock.recv(1024)
sock.close
print('Received data: ',data)
```

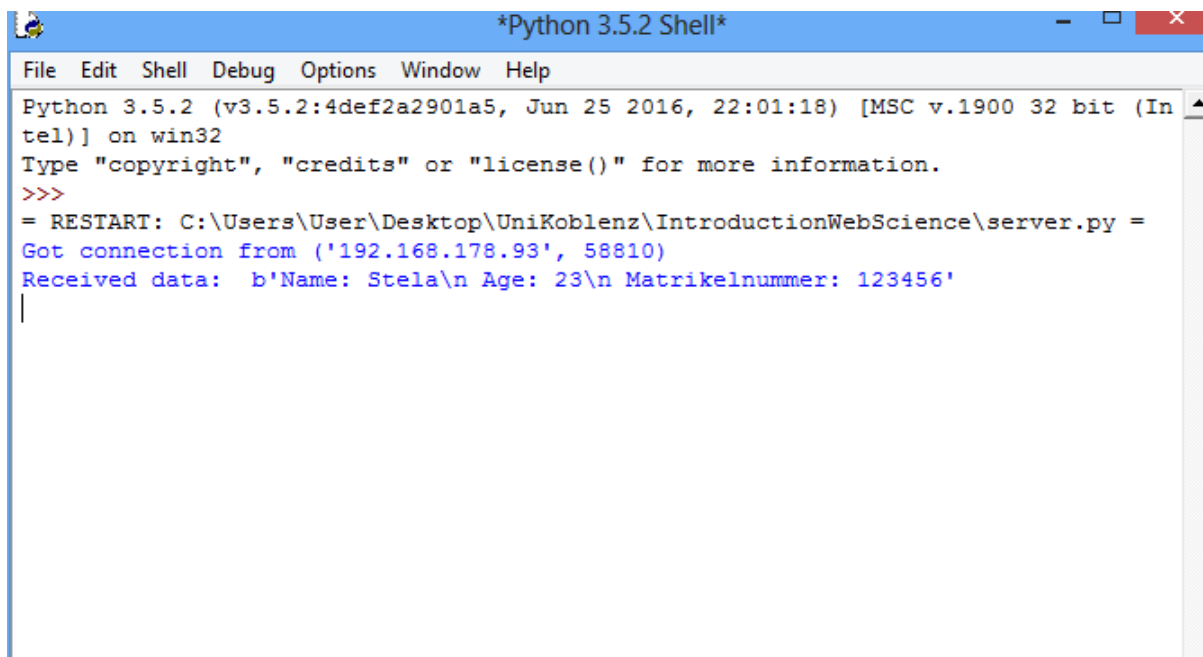


```
*Python 3.5.2 Shell*
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\client.py =
Give your name.Stela
Give your age.23
Give your matrikelnummer.123456
```



A screenshot of a Python 3.5.2 Shell window. The title bar is blue and contains the text "Python 3.5.2 Shell". Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output: "Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". Below the prompt, the text "= RESTART: C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\client.py =" is shown. This is followed by three lines of user input: "Give your name.Stela", "Give your age.23", and "Give your matrikelnummer.123456". The final line of output is "Received data: b'Name: Stela\n Age: 23\n Matrikelnummer: 123456'", followed by another prompt ">>>".

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\client.py =
Give your name.Stela
Give your age.23
Give your matrikelnummer.123456
Received data: b'Name: Stela\n Age: 23\n Matrikelnummer: 123456'
>>> |
```



A screenshot of a Python 3.5.2 Shell window. The title bar is blue and contains the text "*Python 3.5.2 Shell*". Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output: "Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". Below the prompt, the text "= RESTART: C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\server.py =" is shown. This is followed by two lines of output: "Got connection from ('192.168.178.93', 58810)" and "Received data: b'Name: Stela\n Age: 23\n Matrikelnummer: 123456'". The prompt ">>>" is not visible at the bottom of the window.

```
*Python 3.5.2 Shell*
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\User\Desktop\UniKoblenz\IntroductionWebScience\server.py =
Got connection from ('192.168.178.93', 58810)
Received data: b'Name: Stela\n Age: 23\n Matrikelnummer: 123456'
|
```