

TCP

Client TCP Code:

```
from socket import *
import webbrowser

serverName = '127.0.0.1'
serverPort = 11000
fileName = 'HelloWorld.html'

# Create a TCP socket
clientSocket = socket(AF_INET, SOCK_STREAM)

# Connect to the server
clientSocket.connect((serverName, serverPort))

# Send a valid HTTP GET request
request = f"GET /{fileName} HTTP/1.1\r\nHost: {serverName}\r\n\r\n"
clientSocket.send(request.encode())

# Receive the response
response = b""
while True:
    data = clientSocket.recv(2048)
    if not data:
        break
    response += data

# Close the connection
clientSocket.close()

# Decode response and extract HTML body
response_str = response.decode()
header, _, body = response_str.partition("\r\n\r\n")
print(f"Response header: {body}")

# Save HTML to a temporary file
output_file = "received_page.html"
with open(output_file, "w", encoding="utf-8") as f:
    f.write(body)

print(f"Saved HTML content to {output_file}")
webbrowser.open(output_file)
```

Server TCP Code

```
#import socket module
import sys
```

```

from socket import *

serverName = '127.0.0.1'
serverPort = 11000

serverSocket = socket(AF_INET, SOCK_STREAM)

serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
print('The server is ready to receive')

while True:
    connectionSocket, clientAddress = serverSocket.accept()
    try:
        message = connectionSocket.recv(2048).decode()
        filename = message.split()[1]
        f = open(filename[1:])
        outputData = f.readlines()
        f.close()

        connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n".encode())
        for i in range(1, len(outputData)):
            connectionSocket.send(outputData[i].encode())
        connectionSocket.close()
    except IOError:
        connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n".encode())
        connectionSocket.send("<html><body><h1>404 Not
Found</h1></body></html>".encode())
        connectionSocket.close()

```

Printouts:

Client

The client receives the Header from the Server and loads the content into a copy version of the HTML file.

```

C:\Users\olegs\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\olegs\Documents\SDU - Copy\SDU\Semester 5\Date Kommunikation\Journals\20-10-2025 Python Lab (NEED)\TCP Python Server\Client_TCP.py"
Response header: <html lang="en">
<head>
<meta charset="UTF-8">
<title>Todays News</title>

<h1>Hello world! this is the page you got from the server lil bro</h1>
<p>Now let's see if it works</p>

</head>
<body>
</body>
</html>
Saved HTML content to received_page.html

Process finished with exit code 0

```

Then the client opens the copied version locally.



Server

The server is only able to process one client at a time. When the server is running, it will print a message “The server is ready to receive.”

```
C:\Users\olegs\PyCharmMiscProject\.venv\Scripts\python.exe "C:\Users\olegs\Documents\SDU - Copy\SDU\Semester 5\Data Kommunikation\Journals\20-10-2025 Python Lab (NEED)\TCP Python Server\Server_TCP_HTML.py"
The server is ready to receive
```

WIRESHARK

To make Wireshark work with it, it is important to start the server using py -3 Server_TCP_HTML.py command. This will make a HTTP call, which Wireshark will be able to track.

```
PS C:\Users\olegs\Documents\SDU - Copy\SDU\Semester 5\Data Kommunikation\Journals\20-10-2025 Python Lab (NEED)\TCP Python Server> py -3 Server_TCP_HTML.py
The server is ready to receive
```

In the same way, we should start our client. The client will then receive the content from the server.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\olegs\Documents\SDU - Copy\SDU\Semester 5\Data Kommunikation\Journals\20-10-2025 Python Lab (NEED)\TCP Python Server> py -3 Client_TCP.py
Response header: <html lang="en">
<head>
<meta charset="UTF-8">
<title>Todays News</title>
<h1>Hello world! this is the page you got from the server lil bro</h1>
<p>Now let's see if it works</p>

</head>
<body>
</body>
</html>
Saved HTML content to received_page.html
PS C:\Users\olegs\Documents\SDU - Copy\SDU\Semester 5\Data Kommunikation\Journals\20-10-2025 Python Lab (NEED)\TCP Python Server> 
```

1.

The IP for client is:

Source Address: 127.0.0.1

The client port is :

Source Port: 51052

2.

Layer	Size	Key Info
IP	20 bytes	IPv4, proto=6 (TCP), src=127.0.0.1
TCP	20 bytes	Ports, flags, sequence
HTTP	Data: 223 bytes	“HTTP/1.1 200 OK”

In this capture, the packets are sent over the loopback interface (127.0.0.1), so there is no actual Ethernet header.

IP

```
▼ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0xd36 (7478)
  ▶ 010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
    [Stream index: 0]
```

TCP

```
▼ Transmission Control Protocol, Src Port: 11000, Dst Port: 51052, Seq: 243, Ack: 51, Len: 0
  Source Port: 11000
  Destination Port: 51052
  [Stream index: 19]
  [Stream Packet Number: 34]
  ▶ [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 243      (relative sequence number)
  Sequence Number (raw): 3987857335
  [Next Sequence Number: 244      (relative sequence number)]
  Acknowledgment Number: 51      (relative ack number)
  Acknowledgment number (raw): 2420362135
  0101 .... = Header Length: 20 bytes (5)
  ▶ Flags: 0x011 (FIN, ACK)
  Window: 255
  [Calculated window size: 65280]
  [Window size scaling factor: 256]
  Checksum: 0x9528 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  ▶ [Timestamps]
  [Client Contiguous Streams: 1]
  [Server Contiguous Streams: 1]
```

HTTP

```
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  \r\n
  [Request in frame: 588]
  [Time since request: 510.000 microseconds]
  [Request URI: /HelloWorld.html]
  [Full request URI: http://127.0.0.1/HelloWorld.html]
  File Data: 223 bytes
  ▼ Data (223 bytes)
    Data [...]: 3c68746d6c206c616e673d22656e223e0a3c686561643e0a20203c6d65746120636861727365743d22554462d38223e0a20203c7469746c653e546f64
    [Length: 223]
```