



COMP2322 COMPUTER NETWORKING

Lab 4 Report: Socket Programming

Author

Wang Yuqi



Lecturer

Dr. LOU Wei

Questions

Question 1

Q: What is the output when running this python program? Screen capture the executing result.



The screenshot shows a code editor with a file named `GoogleClient.py` open. The code is a Python script that attempts to create a socket and connect to Google's web server. The code is as follows:

```
3
4 try:
5     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     print ("socket successfully created")
7 except socket.error as err:
8     print ("socket creation failed with error %s" %(err))
9
10 # default port for socket
11 port = 80
12
13 try:
14     # find the IP of the Google's web server
15     host_ip = socket.gethostbyname('www.google.com')
16 except socket.gaierror:
17     # this means could not resolve the host
18     print ("there was an error resolving the host")
19     sys.exit()
20
21 # connecting to the server
22 s.connect((host_ip, port))
23 print ("the socket has successfully connected to google")
```

Below the code editor, there is a terminal window showing the output of the program. The terminal output is as follows:

```
(def-311) (base) → labs /opt/homebrew/anaconda3/envs/def-311/bin/python "/Users/tony
socket successfully created
the socket has successfully connected to google
(def-311) (base) → labs
```

Question 2

lab4 >  TCPClient.py > ...

```
1  # import the socket library
2  import socket
3  # create a socket object
4  clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STR
5  # define the server's name and port on which you want to con
6  serverName = '127.0.0.1'
7  serverPort = 12345
8  # connect to the server
9  clientSocket.connect((serverName, serverPort))
10 # receive data from the server and decode to get the string.
11 sentence = clientSocket.recv(1024).decode()
12 print("from server:", sentence)
13 # close the connection
14 clientSocket.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS COMMENTS


```
/Users/tonywang/.zprofile:15: no such file or directory: /usr/local/bin/brew
/Users/tonywang/.zshrc:source:120: no such file or directory: /opt/homebrew/opt/powerle
● (def-311) (base) → labs python ./lab4/TCPClient.py
from server: thank you for connecting
❖ (def-311) (base) → labs □
```

```
ogleClient.py  TCPServer.py ×  TCPClient.py  q1.png  | ▶ ▼ □ ...
lab4 > TCPServer.py > ...
1  # import the socket library
2  import socket
3  # create a socket object
4  serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5  print ("socket successfully created")
6  # reserve a port=12345 on your computer
7  serverPort = 12345
8  # bind to the port
9  # we have not typed any ip in the ip field, instead we have
10 # this makes the server listen to requests coming from other
11 serverSocket.bind(('', serverPort))
12 print ("socket binded to %s" %(serverPort))
13 # put the socket into listening mode
14 serverSocket.listen(5)
15 print ("socket is listening")
16
17 # a forever loop until we interrupt it or an error occurs
18 while True:
19     # establish connection with client.
20     connectionSocket, addr = serverSocket.accept()
21     print ('got connection from', addr )
22     # send a message to the client, using encode() to send bytes
23     sentence='thank you for connecting'
24     connectionSocket.send(sentence.encode())
25     # close the connection with the client
26     connectionSocket.close()
27     break

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  COMMENTS
● (def-311) (base) → labs python ./lab4/TCPServer.py
socket successfully created
socket binded to 12345
socket is listening
got connection from ('127.0.0.1', 60994)
✦ (def-311) (base) → labs
```

(See Next Page)

Question 3



The screenshot shows a code editor with several tabs: `TCPServer.py`, `q3-server.png`, `TCPClient.py` (active), and `_socket.py`. The `TCPClient.py` tab shows the following Python code:

```
lab4 > TCPClient.py > ...  
3 # create a socket object  
4 clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
5 clientSocket.bind(('', 40134))  
6 # define the server's name and port on which you want to connect  
7 serverName = '127.0.0.1'  
8 serverPort = 12345  
9 # connect to the server  
10 clientSocket.connect((serverName, serverPort))  
11  
12 srcIP, srcPort = clientSocket.getsockname()  
13 destIP, destPort = clientSocket.getpeername()  
14  
15 print(f'Tuple: (SrcIP: {srcIP}, SrcPort: {srcPort}, DestIP: {destIP}, DestPort: {destPort})')  
16  
17 # receive data from the server and decode to get the string.  
18 sentence = clientSocket.recv(1024).decode()  
19 print("from server:", sentence)  
20 # close the connection  
21 clientSocket.close()
```

At the bottom of the editor, there is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS  
● (def-311) (base) → labs python ./lab4/TCPClient.py  
Tuple: (SrcIP: 127.0.0.1, SrcPort: 40134, DestIP: 127.0.0.1, DestPort: 12345)  
from server: thank you for connecting  
❖ (def-311) (base) → labs
```

```
TCPServer.py x TCPClient.py _socket.py q2-client.png
lab4 > TCPServer.py > ... ~/Code/prac/polyu_comp_
2 import socket
3 # create a socket object
4 serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STR
5 print ("socket successfully created")
6 # reserve a port=12345 on your computer
7 serverPort = 12345
8 # bind to the port
9 # we have not typed any ip in the ip field, instead we have
10 # this makes the server listen to requests coming from other
11 serverSocket.bind('', serverPort))
12 print ("socket binded to %s" %(serverPort))
13 # put the socket into listening mode
14 serverSocket.listen(5)
15 print ("socket is listening")
16
17 # a forever loop until we interrupt it or an error occurs
18 while True:
19     # establish connection with client.
20     connectionSocket, addr = serverSocket.accept()
21     print ('got connection from', addr )
22     # send a message to the client, using encode() to send b
23     sentence='thank you for connecting'
24     connectionSocket.send(sentence.encode())
25     # close the connection with the client
26     connectionSocket.close()
27 break
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

```
• (def-311) (base) → labs python ./lab4/TCPServer.py
socket successfully created
socket binded to 12345
socket is listening
got connection from ('127.0.0.1', 40134)
❖ (def-311) (base) → labs
```

(See Next Page)

Modified Code for TCPClient.py

```
# import the socket library
import socket
# create a socket object
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientSocket.bind(('', 40134))
# define the server's name and port on which you want to connect
serverName = '127.0.0.1'
serverPort = 12345
# connect to the server
clientSocket.connect((serverName, serverPort))

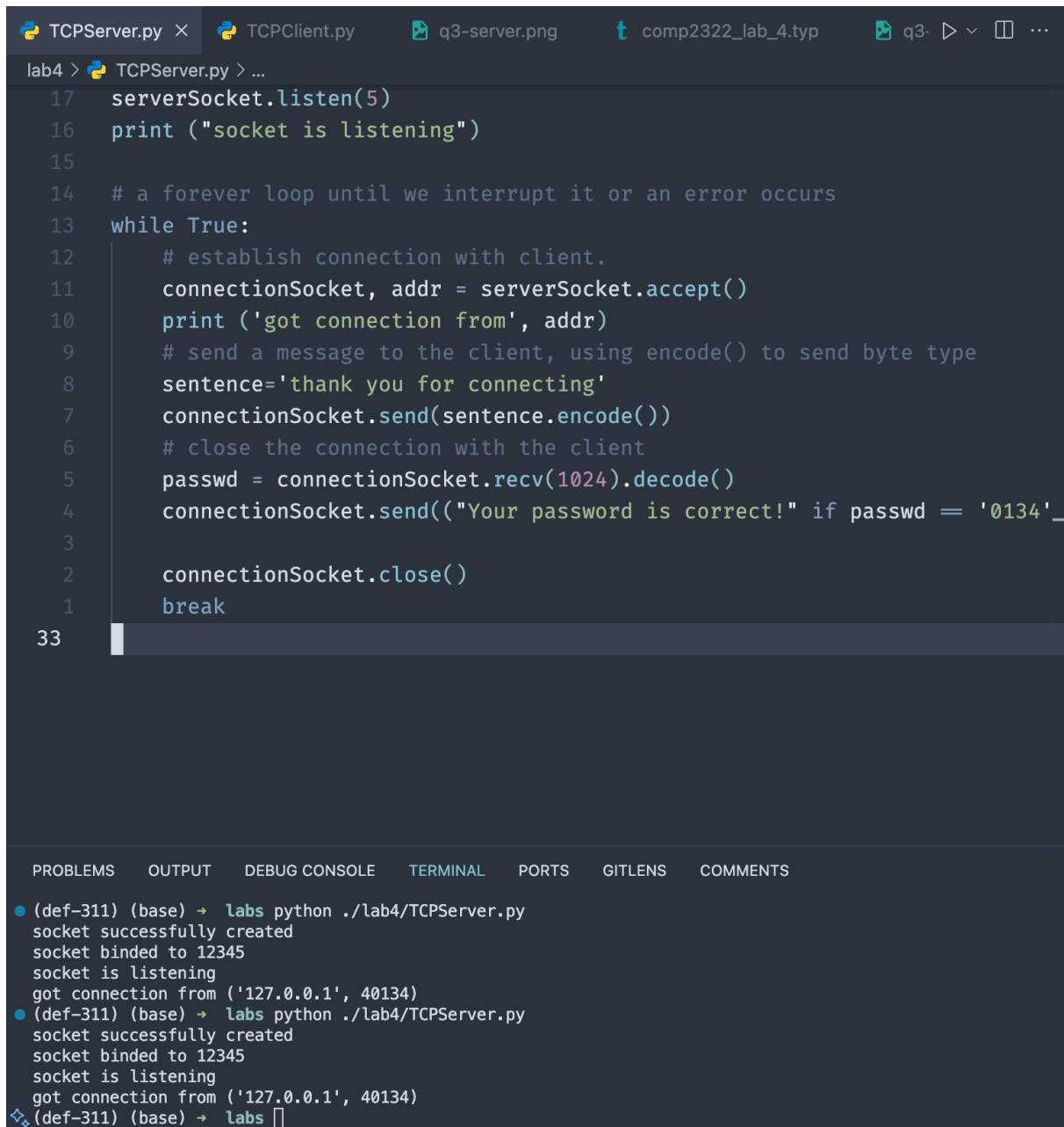
srcIP, srcPort = clientSocket.getsockname()
destIP, destPort = clientSocket.getpeername()

print(f'Tuple: (SrcIP: {srcIP}, SrcPort: {srcPort}, DestIP: {destIP},
DestPort: {destPort})')

# receive data from the server and decode to get the string.
sentence = clientSocket.recv(1024).decode()
print ("from server:", sentence)
# close the connection

clientSocket.close()
```

Question 4



The screenshot shows a code editor with two tabs: `TCPClient.py` and `TCPServer.py`. The `TCPServer.py` tab is active, displaying a Python script for a TCP server. The script listens on port 12345 and responds to client connections with a 'thank you' message and a password check. The terminal output shows the server running successfully and receiving a connection from `127.0.0.1` on port `40134`.

```
lab4 > TCPServer.py > ...
17 serverSocket.listen(5)
16 print ("socket is listening")
15
14 # a forever loop until we interrupt it or an error occurs
13 while True:
12     # establish connection with client.
11     connectionSocket, addr = serverSocket.accept()
10     print ('got connection from', addr)
9     # send a message to the client, using encode() to send byte type
8     sentence='thank you for connecting'
7     connectionSocket.send(sentence.encode())
6     # close the connection with the client
5     passwd = connectionSocket.recv(1024).decode()
4     connectionSocket.send(("Your password is correct!" if passwd == '0134'_)
3
2     connectionSocket.close()
1     break
33
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

- (def-311) (base) → labs python ./lab4/TCPServer.py
socket successfully created
socket binded to 12345
socket is listening
got connection from ('127.0.0.1', 40134)
- (def-311) (base) → labs python ./lab4/TCPServer.py
socket successfully created
socket binded to 12345
socket is listening
got connection from ('127.0.0.1', 40134)
- ❖ (def-311) (base) → labs


```
TCPClient.py x q3-server.png comp232%
lab4 > TCPClient.py > ...
8 import socket
7 # create a socket object
6 clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 clientSocket.bind('', 40134)
4 # define the server's name and port on which you want to connect
3 serverName = '127.0.0.1'
2 serverPort = 12345
1 # connect to the server
10 clientSocket.connect((serverName, serverPort))
1
2 srcIP, srcPort = clientSocket.getsockname()
3 destIP, destPort = clientSocket.getpeername()
4
5 print(f'Tuple: (SrcIP: {srcIP}, SrcPort: {srcPort}, DestIP: {destIP}, DestPort: {destPort})')
6
7 # receive data from the server and decode to get the string.
8 sentence = clientSocket.recv(1024).decode()
9 print("from server:", sentence)
10 # close the connection
11
12 passwd = input('Input 4 digit password: ')
13 clientSocket.send(passwd.encode())
14
15 auth_result = clientSocket.recv(1034).decode()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS

- (def-311) (base) → labs python ./lab4/TCPClient.py
Tuple: (SrcIP: 127.0.0.1, SrcPort: 40134, DestIP: 127.0.0.1, DestPort: 12345)
from server: thank you for connecting
Input 4 digit password: 0134
from server: Your password is correct!
- (def-311) (base) → labs python ./lab4/TCPClient.py
Tuple: (SrcIP: 127.0.0.1, SrcPort: 40134, DestIP: 127.0.0.1, DestPort: 12345)
from server: thank you for connecting
Input 4 digit password: 1441
from server: Your password is incorrect!
- (def-311) (base) → labs

(See Next Page)

Modified Code for TCPServer.py

```
# import the socket library
import socket
# create a socket object
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

print ("socket successfully created")
# reserve a port=12345 on your computer
serverPort = 12345
# bind to the port
# we have not typed any ip in the ip field, instead we have inputted an
empty string.
# this makes the server listen to requests coming from other computers
on the network
serverSocket.bind('', serverPort)
print ("socket binded to %s" %(serverPort))
# put the socket into listening mode
serverSocket.listen(5)
print ("socket is listening")

# a forever loop until we interrupt it or an error occurs
while True:
    # establish connection with client.
    connectionSocket, addr = serverSocket.accept()
    print ('got connection from', addr)
    # send a message to the client, using encode() to send byte type
    sentence='thank you for connecting'
    connectionSocket.send(sentence.encode())
    # close the connection with the client
    passwd = connectionSocket.recv(1024).decode()
    connectionSocket.send(("Your password is correct!" if passwd ==
'0134' else "Your password is incorrect!").encode())

    connectionSocket.close()
    break
```

(See Next Page)

Modified Code for TCPClient.py

```
# import the socket library
import socket
# create a socket object
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientSocket.bind(('', 40134))
# define the server's name and port on which you want to connect
serverName = '127.0.0.1'
serverPort = 12345
# connect to the server
clientSocket.connect((serverName, serverPort))

srcIP, srcPort = clientSocket.getsockname()
destIP, destPort = clientSocket.getpeername()

print(f'Tuple: (SrcIP: {srcIP}, SrcPort: {srcPort}, DestIP: {destIP},
DestPort: {destPort})')

# receive data from the server and decode to get the string.
sentence = clientSocket.recv(1024).decode()
print ("from server:", sentence)
# close the connection

passwd = input('Input 4 digit password: ')
clientSocket.send(passwd.encode())

auth_result = clientSocket.recv(1034).decode()
print("from server: ", auth_result)

clientSocket.close()
```