ACN LAB - 01
Socket Programming for Client-Server Communication Using Python

Chaitanya Talware (MIS No: 712422005)
Yogesh Toshniwal (MIS No: 712422021)

1 Introduction

Socket programming enables communication between programs over a network using the client-server model. In this assignment, we implement a client-server system using Python's socket module. The server listens for client requests and provides mathematical operations such as square, square root, and factorial. The client sends a request with the operation and number, and the server computes and returns the result.

The available operations are:

- `sqrt`: Calculate the square root of a number.
- `square`: Calculate the square of a number.
- `factorial`: Calculate the factorial of a number.

2 Source Code

2.1 Client Code

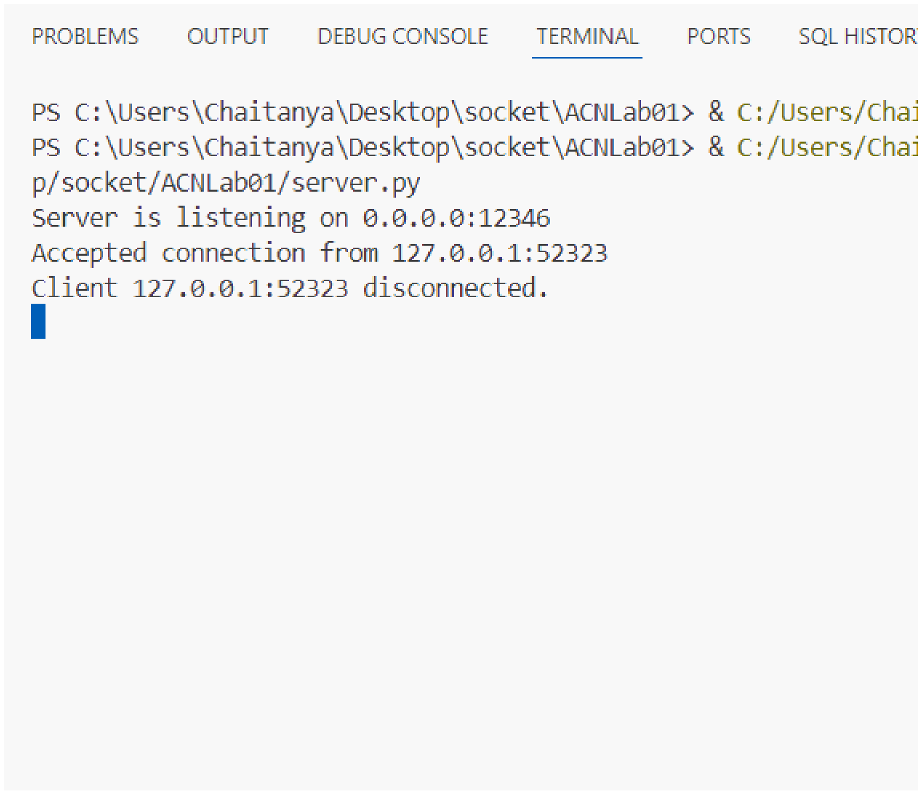
```python
import socket

def main():
    client_socket = socket.socket(socket.AF_INET,
        socket.SOCK_STREAM)
    host = 'localhost'
    port = 12346
    client_socket.connect((host, port))

    print("Available operations: 'sqrt', 'square',
        'factorial' (or 1, 2, 3)")
    print("Enter 'q' to quit.")

    while True:
        operation = input("Enter the operation you want to perform (q to quit): ")

        if operation == 'q':
            client_socket.send(b'q')  # Send quit command to server
            print('Exiting...')
            break

        number = int(input("Enter the number: "))
        data = "{} {}".format(operation, number)
        client_socket.send(data.encode('utf-8'))

        result = client_socket.recv(1024).decode('utf-8')
        print("Result: {}".format(result))

    client_socket.close()

if __name__ == "__main__":
    main()
```

2.2 Server Code

```python
import socket
import math

def calculate_sqrt(number):
    return math.sqrt(number)

def calculate_square(number):
    return number ** 2

def calculate_factorial(number):
    if number == 0:
        return 1
    return number * calculate_factorial(number - 1)

def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    host = '0.0.0.0'
    port = 12346
    server_socket.bind((host, port))

    server_socket.listen(5)
    print("Server is listening on {}:{}".format(host, port))

    while True:
        client_socket, addr = server_socket.accept()
        print("Accepted connection from {}:{}".format(addr[0], addr[1]))

        while True:
            data = client_socket.recv(1024).decode('utf-8')
            if data == 'q':
                print("Client {}:{} disconnected.".format(addr[0], addr[1]))
                break

            if data == '':
                print("Disconnected - from {}:{}".format(addr[0], addr[1]))
                break

            operation, number = data.split()
            number = float(number)

            if operation == 'sqrt' or operation == '1':
                result = calculate_sqrt(number)
            elif operation == 'square' or operation == '2':
                result = calculate_square(number)
            elif operation == 'factorial' or operation == '3':
                result = calculate_factorial(int(number))
            else:
                result = "Invalid operation"

            client_socket.send(str(result).encode('utf-8'))

        client_socket.close()  # Close the client socket after exiting the loop

        \section{Output}
        \subsection{Server Output}
        \subsection{Client Output}
if __name__ == "__main__":
    main()
```

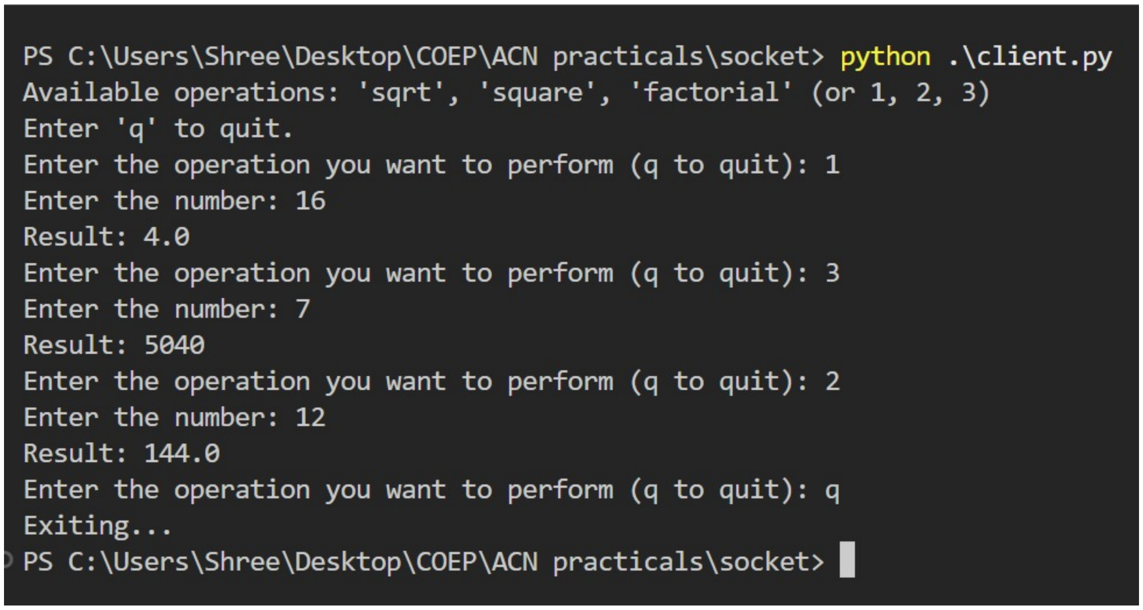
3 Output

3.1 Server Output



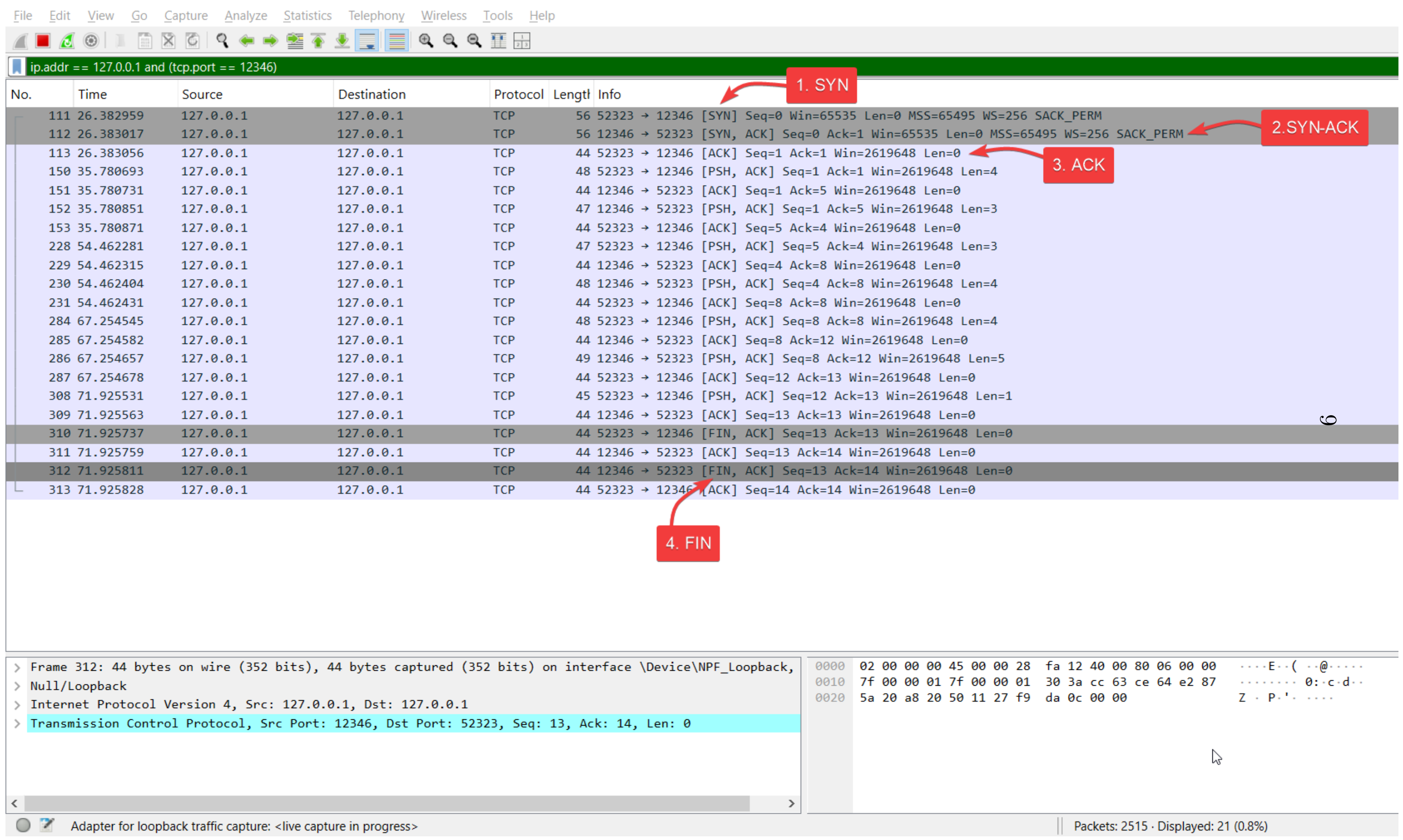
```
PS C:\Users\Chaitanya\Desktop\socket\ACNLab01> & C:/Users/Chai
PS C:\Users\Chaitanya\Desktop\socket\ACNLab01> & C:/Users/Chai
p/socket/ACNLab01/server.py
Server is listening on 0.0.0.0:12346
Accepted connection from 127.0.0.1:52323
Client 127.0.0.1:52323 disconnected.
```

3.2 Client Output



```
PS C:\Users\Shree\Desktop\COEP\ACN practicals\socket> python .\client.py
Available operations: 'sqrt', 'square', 'factorial' (or 1, 2, 3)
Enter 'q' to quit.
Enter the operation you want to perform (q to quit): 1
Enter the number: 16
Result: 4.0
Enter the operation you want to perform (q to quit): 3
Enter the number: 7
Result: 5040
Enter the operation you want to perform (q to quit): 2
Enter the number: 12
Result: 144.0
Enter the operation you want to perform (q to quit): q
Exiting...
PS C:\Users\Shree\Desktop\COEP\ACN practicals\socket>
```

3.3 Wireshark Output


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
1. SYN
2.SYN-ACK
3. ACK
4. FIN

```
ip.addr == 127.0.0.1 and (tcp.port == 12346)

No.   Time        Source      Destination  Protocol Length Info
  111 26.382959   127.0.0.1   127.0.0.1    TCP      56 52323 → 12346 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  112 26.383017   127.0.0.1   127.0.0.1    TCP      56 12346 → 52323 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
  113 26.383056   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
  150 35.780693   127.0.0.1   127.0.0.1    TCP      48 52323 → 12346 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=4
  151 35.780731   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [ACK] Seq=1 Ack=5 Win=2619648 Len=0
  152 35.780851   127.0.0.1   127.0.0.1    TCP      47 12346 → 52323 [PSH, ACK] Seq=1 Ack=5 Win=2619648 Len=3
  153 35.780871   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [ACK] Seq=5 Ack=4 Win=2619648 Len=0
  228 54.462281   127.0.0.1   127.0.0.1    TCP      47 52323 → 12346 [PSH, ACK] Seq=5 Ack=4 Win=2619648 Len=3
  229 54.462315   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [ACK] Seq=4 Ack=8 Win=2619648 Len=0
  230 54.462404   127.0.0.1   127.0.0.1    TCP      48 12346 → 52323 [PSH, ACK] Seq=4 Ack=8 Win=2619648 Len=4
  231 54.462431   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [ACK] Seq=8 Ack=8 Win=2619648 Len=0
  284 67.254545   127.0.0.1   127.0.0.1    TCP      48 52323 → 12346 [PSH, ACK] Seq=8 Ack=8 Win=2619648 Len=4
  285 67.254582   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [ACK] Seq=8 Ack=12 Win=2619648 Len=0
  286 67.254657   127.0.0.1   127.0.0.1    TCP      49 12346 → 52323 [PSH, ACK] Seq=8 Ack=12 Win=2619648 Len=5
  287 67.254678   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [ACK] Seq=12 Ack=13 Win=2619648 Len=0
  308 71.925531   127.0.0.1   127.0.0.1    TCP      45 52323 → 12346 [PSH, ACK] Seq=12 Ack=13 Win=2619648 Len=1
  309 71.925563   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [ACK] Seq=13 Ack=13 Win=2619648 Len=0
  310 71.925737   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [FIN, ACK] Seq=13 Ack=13 Win=2619648 Len=0
  311 71.925759   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [ACK] Seq=13 Ack=14 Win=2619648 Len=0
  312 71.925811   127.0.0.1   127.0.0.1    TCP      44 12346 → 52323 [FIN, ACK] Seq=13 Ack=14 Win=2619648 Len=0
  313 71.925828   127.0.0.1   127.0.0.1    TCP      44 52323 → 12346 [ACK] Seq=14 Ack=14 Win=2619648 Len=0

> Frame 312: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback,
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 12346, Dst Port: 52323, Seq: 13, Ack: 14, Len: 0

0000  02 00 00 00 45 00 00 28  fa 12 40 00 80 06 00 00   ····E··( ··@·····
0010  7f 00 00 01 7f 00 00 01  30 3a cc 63 ce 64 e2 87   ········ 0:·c·d··
0020  5a 20 a8 20 50 11 27 f9  da 0c 00 00               Z · P·'· ····

Adapter for loopback traffic capture: <live capture in progress>        Packets: 2515 · Displayed: 21 (0.8%)
```