VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

BCSE308P-COMPUTER NETWORKS LAB

NAME : PATEL SWAPNILKUMAR C.

REG.NO : 22BAI1308

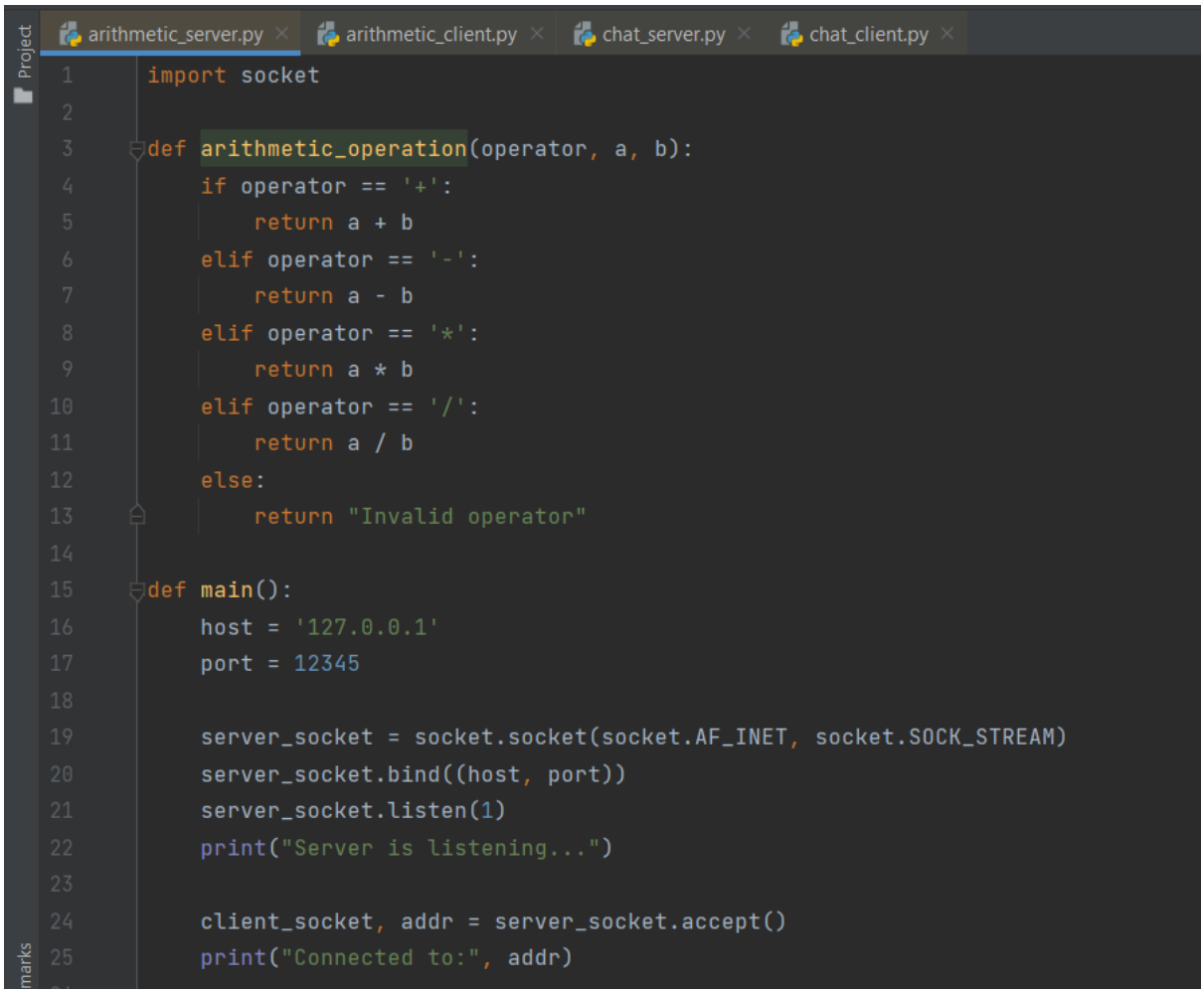
SEM : FALL 23-24

TOPIC : EXPERIMENT- 3&4

(SOCKET PROGRAMMING)

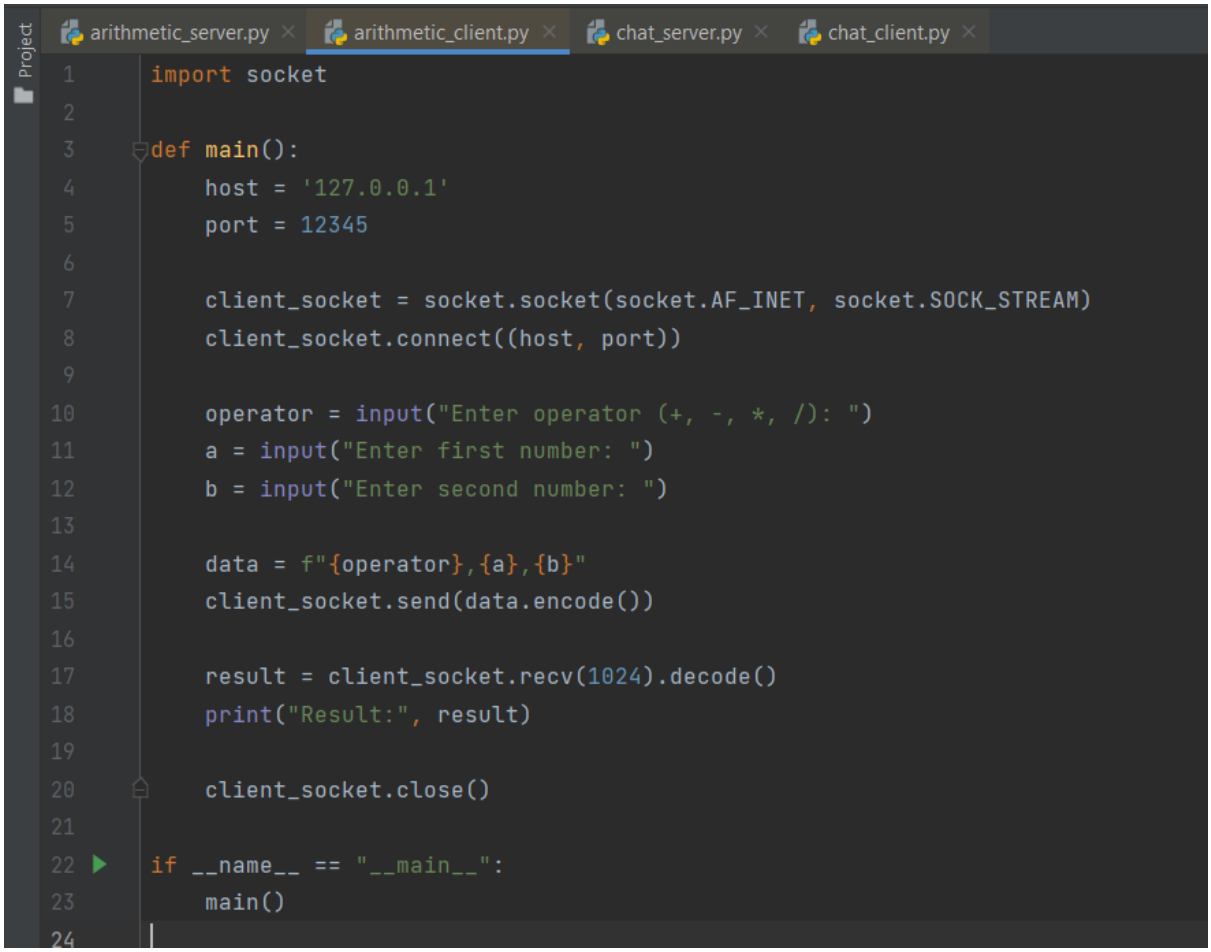
- **Single client-Single server with arithmetic operations.**

Server (arithmetic_server.py):

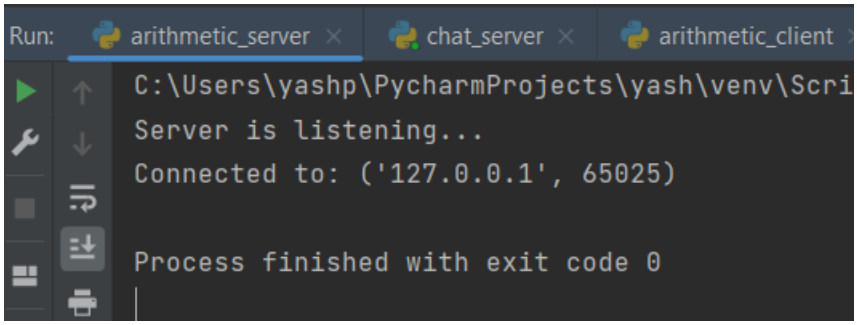


```
26
27      while True:
28          data = client_socket.recv(1024).decode()
29          if not data:
30              break
31
32          operator, a, b = data.split(',')
33          result = arithmetic_operation(operator, float(a), float(b))
34          client_socket.send(str(result).encode())
35
36      client_socket.close()
37
38  if __name__ == "__main__":
39      main()
40
```

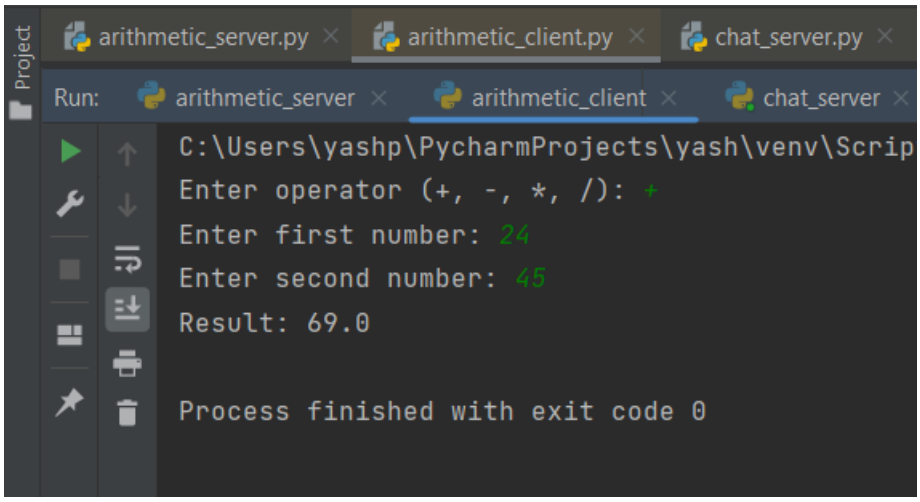
Client (arithmetic_client.py):



SERVER INPUT :

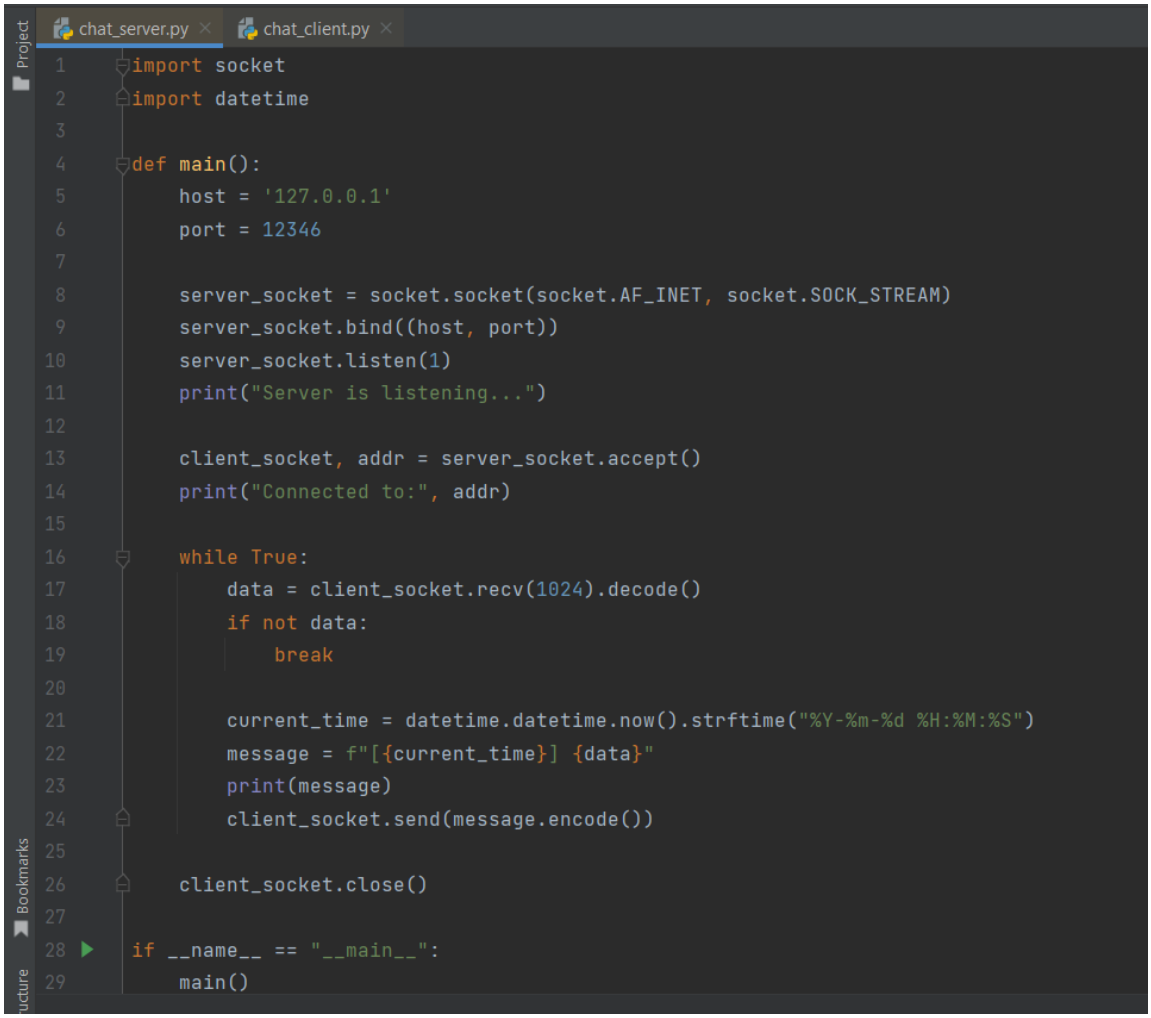


CLIENT OUTPUT :



1. **TCP Chat Application**

Server (chat_server.py):



Client (chat_client.py):

```python
import socket

def main():
    host = '127.0.0.1'
    port = 12346

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))

    while True:
        message = input("You: ")
        client_socket.send(message.encode())
        if message.lower() == 'bye':
            break

        response = client_socket.recv(1024).decode()
        print("Server:", response)

    client_socket.close()

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

CLIENT SIDE OUTPUT :

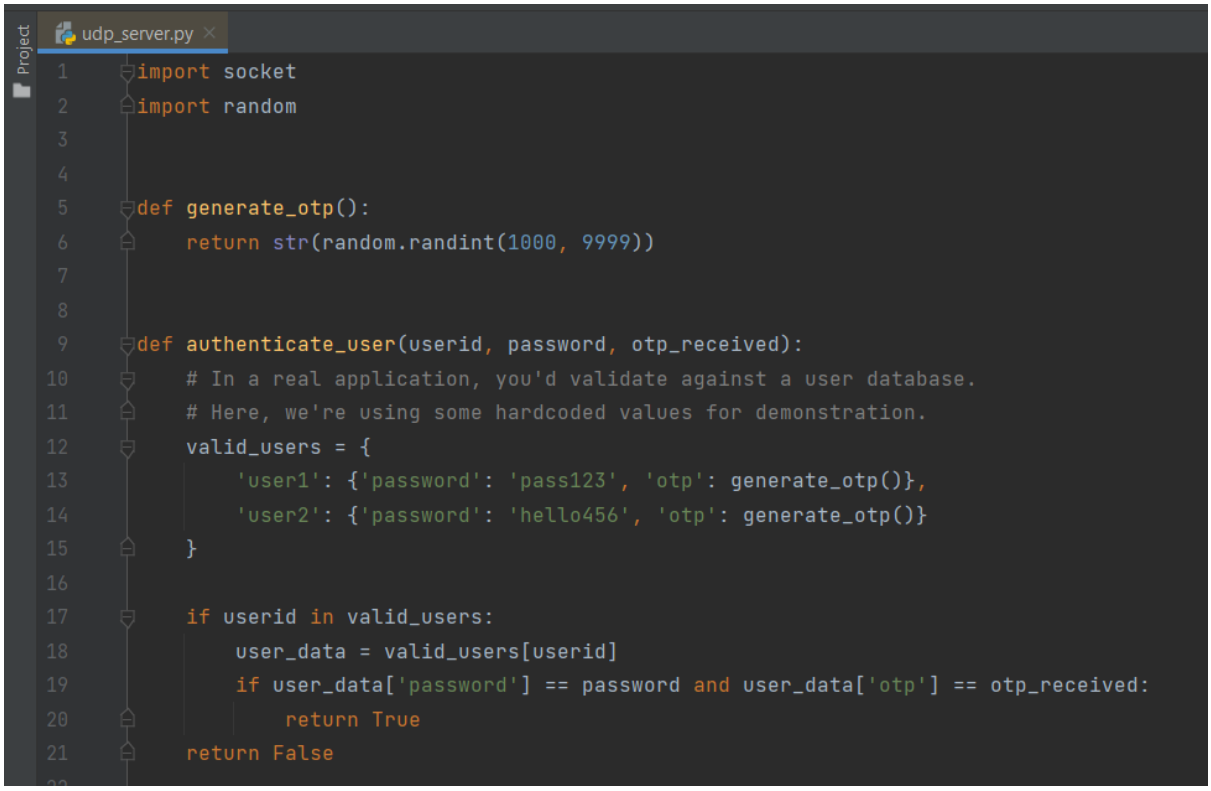
```
Run: chat_server × chat_client ×
C:\Users\yashp\PycharmProjects\yash\venv\Scripts\python.exe C:\User
You: Hi I am Swapnil 22BAI1308
Server: [2023-08-27 21:40:25] Hi I am Swapnil 22BAI1308
You: I completed my dinner
Server: [2023-08-27 21:40:57] I completed my dinner
You:
```

SERVER SIDE OUTPUT :

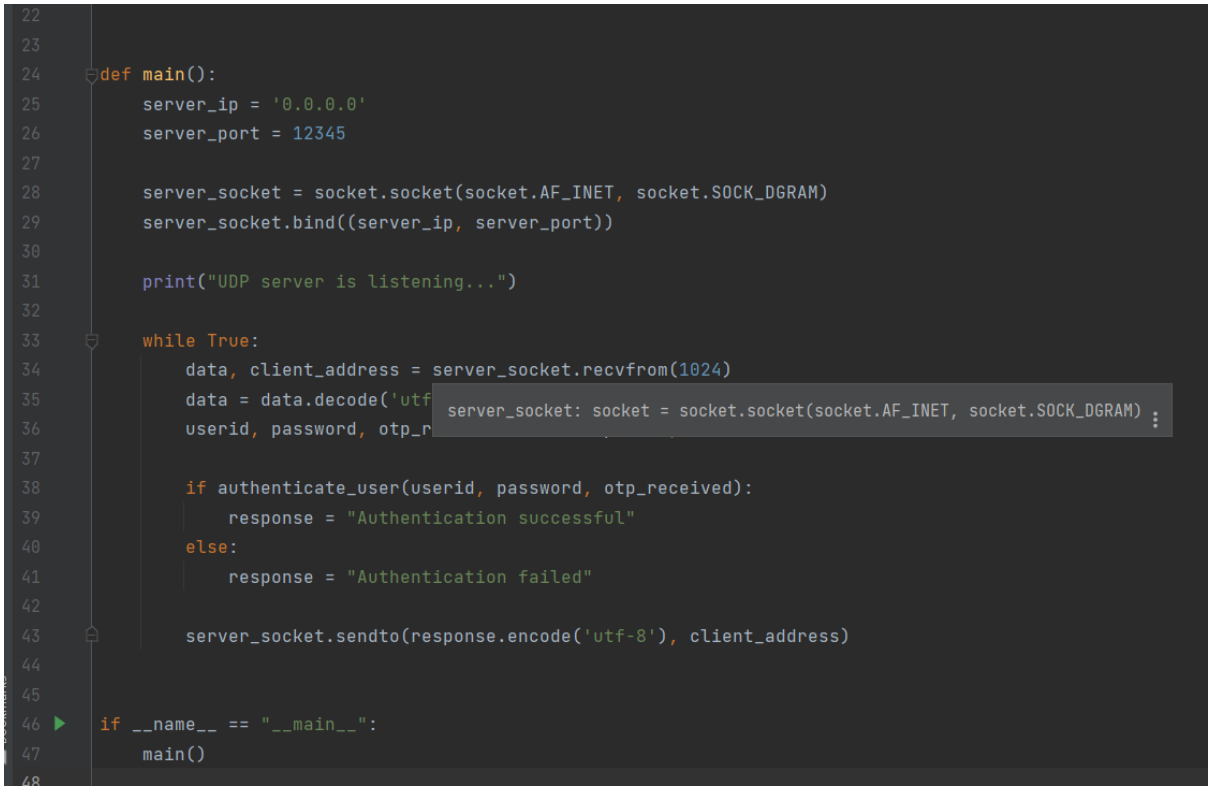
```
Run: chat_server × chat_client ×
C:\Users\yashp\PycharmProjects\yash\venv\Scripts\python.ex
Server is listening...
Connected to: ('127.0.0.1', 65026)
[2023-08-27 21:40:25] Hi I am Swapnil 22BAI1308
[2023-08-27 21:40:57] I completed my dinner
```

2. UDP – OTP Checking

Server (udp_server.py):



```python
import socket
import random


def generate_otp():
    return str(random.randint(1000, 9999))


def authenticate_user(userid, password, otp_received):
    # In a real application, you'd validate against a user database.
    # Here, we're using some hardcoded values for demonstration.
    valid_users = {
        'user1': {'password': 'pass123', 'otp': generate_otp()},
        'user2': {'password': 'hello456', 'otp': generate_otp()}
    }

    if userid in valid_users:
        user_data = valid_users[userid]
        if user_data['password'] == password and user_data['otp'] == otp_received:
            return True
    return False
```



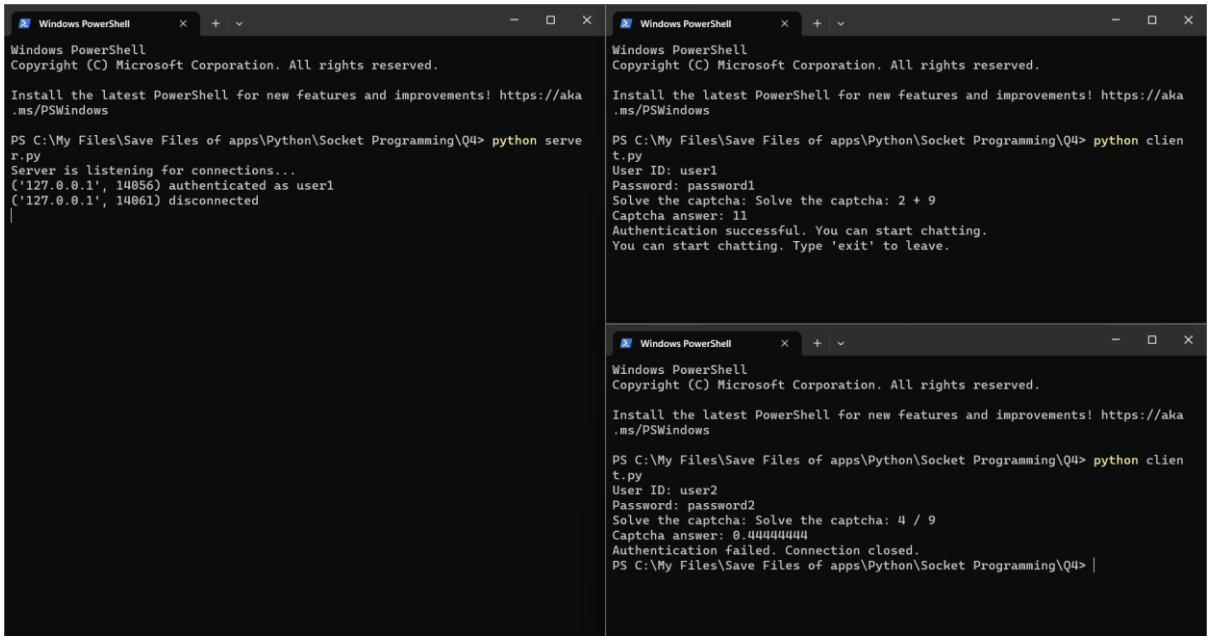
```python


def main():
    server_ip = '0.0.0.0'
    server_port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((server_ip, server_port))

    print("UDP server is listening...")

    while True:
        data, client_address = server_socket.recvfrom(1024)
        data = data.decode('utf
        userid, password, otp_r
        server_socket: socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

        if authenticate_user(userid, password, otp_received):
            response = "Authentication successful"
        else:
            response = "Authentication failed"

        server_socket.sendto(response.encode('utf-8'), client_address)


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

Client(udp_clients.py):

udp_server.py udp_clients.py

```python
import socket


def main():
    server_ip = '127.0.0.1'
    server_port = 12345
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    userid = input("Enter your userid: ")
    password = input("Enter your password: ")

    client_socket.sendto(f"{userid},{password}".encode('utf-8'), (server_ip, server_port))

    otp_received = input("Enter the OTP received on your device: ")
    client_socket.sendto(otp_received.encode('utf-8'), (server_ip, server_port))

    response, _ = client_socket.recvfrom(1024)
    print(response.decode('utf-8'))


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

OUTPUT (in TERMINALS):



3. HTTP – Arithmetic Captcha Checking.

Server(http_server.py) :

http_server.py | http_client.py

```python
import socket
import threading
import random
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import parse_qs

# User database (insecure for demonstration purposes)
users = {"user1": "password1", "user2": "password2"}

# Generate a random arithmetic captcha
def generate_captcha():
    num1 = random.randint(1, 10)
    num2 = random.randint(1, 10)
    operator = random.choice(["+", "-", "*"])
    captcha = f"What is {num1} {operator} {num2}?"
    answer = str(eval(f"{num1} {operator} {num2}"))
    return captcha, answer

# HTTP request handler
class MyRequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == "/login":
            captcha, answer = generate_captcha()
            self.send_response(200)
            self.send_header("Content-type", "text/html")
            self.end_headers()
            self.wfile.write(f"Captcha: {captcha}".encode())

            # Store captcha answer for later verification
            self.captcha_answer = answer
        else:
            self.send_response(404)
            self.end_headers()
            self.wfile.write(b"Page not found")

    def do_POST(self):
        if self.path == "/authenticate":
            content_length = int(self.headers['Content-Length'])
            post_data = self.rfile.read(content_length).decode("utf-8")
            params = parse_qs(post_data)
            username = params['username'][0]
            password = params['password'][0]
            captcha_response = params['captcha'][0]
```

```
44
45              # Check captcha and credentials
46              if captcha_response == self.captcha_answer and users.get(username) == password:
47                  self.send_response(200)
48                  self.send_header("Content-type", "text/html")
49                  self.end_headers()
50                  self.wfile.write(b"Authentication successful")
51              else:
52                  self.send_response(401)
53                  self.send_header("Content-type", "text/html")
54                  self.end_headers()
55                  self.wfile.write(b"Authentication failed")
56          else:
57              self.send_response(404)
58              self.end_headers()
59              self.wfile.write(b"Page not found")
60
61  def run_http_server():
62      server_address = ('localhost', 8080)
63      httpd = HTTPServer(server_address, MyRequestHandler)
64      print("HTTP server running on http://localhost:8080")
65      httpd.serve_forever()
66
67  # Start HTTP server in a separate thread
68  http_thread = threading.Thread(target=run_http_server)
69  http_thread.start()
70
```

Client(http_Client.py) :

http_server.py | http_client.py

```python
import socket
import requests

# Simulate a client requesting a captcha and authentication
def client_request():
    captcha_response = ""
    with requests.get("http://localhost:8080/login") as response:
        print(response.text)
        captcha_response = input("Enter the captcha answer: ")

    username = input("Enter username: ")
    password = input("Enter password: ")

    payload = {
        'username': username,
        'password': password,
        'captcha': captcha_response
    }

    response = requests.post("http://localhost:8080/authenticate", data=payload)
    print(response.text)

# Simulate multiple clients
for _ in range(3):
    client_request()
```

OUTPUT :