



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **BCSE308P-COMPUTER NETWORKS LAB**

**NAME : PATEL SWAPNILKUMAR C.**

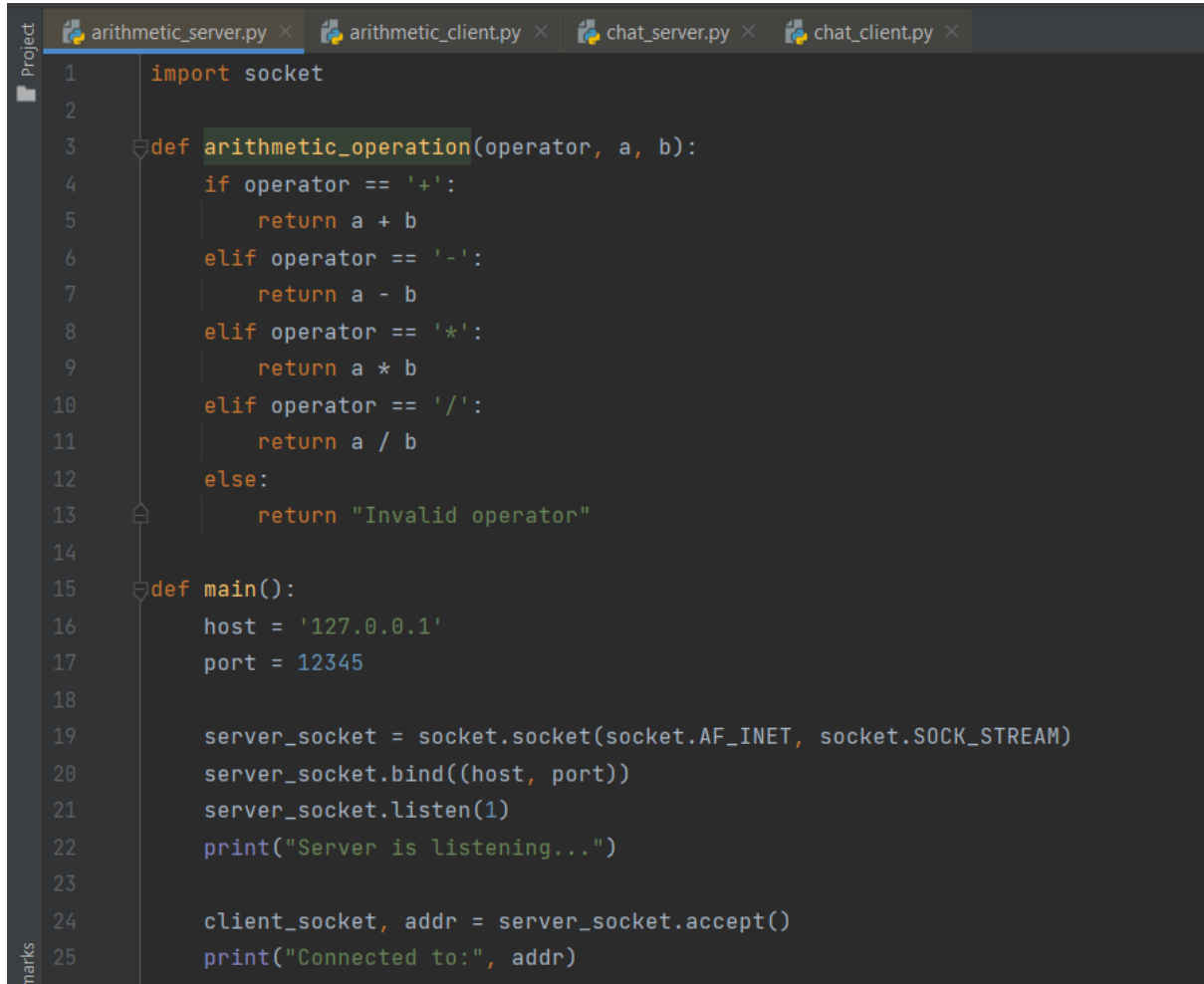
**REG.NO : 22BAI1308**

**SEM : FALL 23-24**

**TOPIC : EXPERIMENT- 3&4  
(SOCKET PROGRAMMING)**

- Single client-Single server with arithmetic operations.

Server (arithmetic\_server.py):



```
1 import socket
2
3 def arithmetic_operation(operator, a, b):
4     if operator == '+':
5         return a + b
6     elif operator == '-':
7         return a - b
8     elif operator == '*':
9         return a * b
10    elif operator == '/':
11        return a / b
12    else:
13        return "Invalid operator"
14
15 def main():
16     host = '127.0.0.1'
17     port = 12345
18
19     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
20     server_socket.bind((host, port))
21     server_socket.listen(1)
22     print("Server is listening...")
23
24     client_socket, addr = server_socket.accept()
25     print("Connected to:", addr)
```

```

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

```

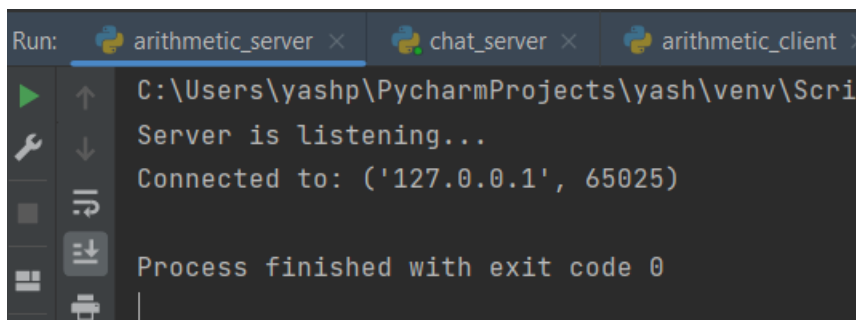
## Client (arithmetic\_client.py):

```

Project
arithmetic_server.py x arithmetic_client.py x chat_server.py x chat_client.py x
1     import socket
2
3     def main():
4         host = '127.0.0.1'
5         port = 12345
6
7         client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8         client_socket.connect((host, port))
9
10        operator = input("Enter operator (+, -, *, /): ")
11        a = input("Enter first number: ")
12        b = input("Enter second number: ")
13
14        data = f"{operator},{a},{b}"
15        client_socket.send(data.encode())
16
17        result = client_socket.recv(1024).decode()
18        print("Result:", result)
19
20        client_socket.close()
21
22 if __name__ == "__main__":
23     main()
24

```

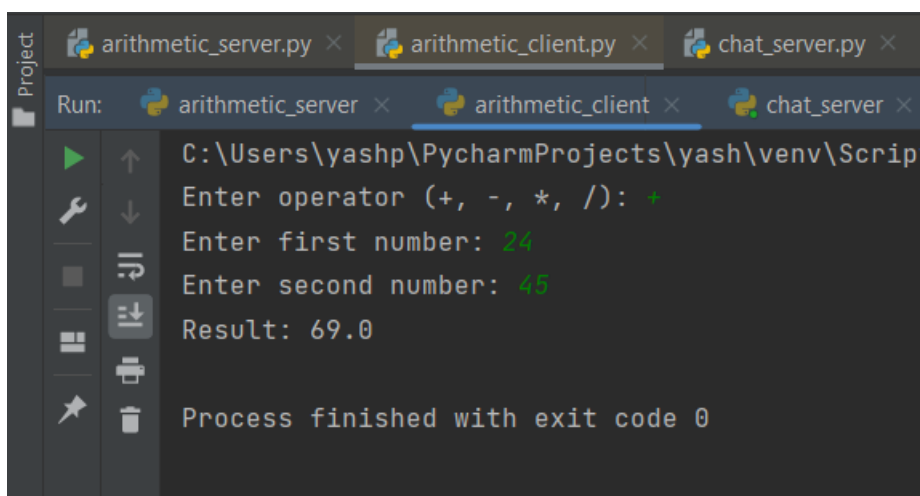
## SERVER INPUT :



The screenshot shows the PyCharm Run console with three tabs: arithmetic\_server, chat\_server, and arithmetic\_client. The arithmetic\_server tab is active. The console output shows the server listening, a connection from 127.0.0.1 on port 65025, and the process finishing with exit code 0.

```
Run: arithmetic_server x chat_server x arithmetic_client x
C:\Users\yashp\PycharmProjects\yash\venv\Scripts>
Server is listening...
Connected to: ('127.0.0.1', 65025)
Process finished with exit code 0
```

## CLIENT OUTPUT :



The screenshot shows the PyCharm Run console with three tabs: arithmetic\_server.py, arithmetic\_client.py, and chat\_server.py. The arithmetic\_client.py tab is active. The console output shows the client entering an operator (+), two numbers (24 and 45), and the result (69.0), followed by the process finishing with exit code 0.

```
Run: arithmetic_server x arithmetic_client x chat_server x
C:\Users\yashp\PycharmProjects\yash\venv\Scripts>
Enter operator (+, -, *, /): +
Enter first number: 24
Enter second number: 45
Result: 69.0
Process finished with exit code 0
```

## **1. TCP Chat Application**

**Server (chat\_server.py):**

```
chat_server.py x chat_client.py x
1 import socket
2 import datetime
3
4 def main():
5     host = '127.0.0.1'
6     port = 12346
7
8     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9     server_socket.bind((host, port))
10    server_socket.listen(1)
11    print("Server is listening...")
12
13    client_socket, addr = server_socket.accept()
14    print("Connected to:", addr)
15
16    while True:
17        data = client_socket.recv(1024).decode()
18        if not data:
19            break
20
21        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
22        message = f"[{current_time}] {data}"
23        print(message)
24        client_socket.send(message.encode())
25
26    client_socket.close()
27
28 if __name__ == "__main__":
29     main()
```

**Client (chat\_client.py):**

```
chat_server.py x chat_client.py x
1 import socket
2
3 def main():
4     host = '127.0.0.1'
5     port = 12346
6
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     client_socket.connect((host, port))
9
10    while True:
11        message = input("You: ")
12        client_socket.send(message.encode())
13        if message.lower() == 'bye':
14            break
15
16        response = client_socket.recv(1024).decode()
17        print("Server:", response)
18
19    client_socket.close()
20
21 if __name__ == "__main__":
22     main()
```

## CLIENT SIDE OUTPUT :

```
Run: chat_server x chat_client x
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 x chat_client x
C:\Users\yashp\PycharmProjects\yash\venv\Scripts\python.exe
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):

```
1 import socket
2 import random
3
4
5 def generate_otp():
6     return str(random.randint(1000, 9999))
7
8
9 def authenticate_user(userid, password, otp_received):
10     # In a real application, you'd validate against a user database.
11     # Here, we're using some hardcoded values for demonstration.
12     valid_users = {
13         'user1': {'password': 'pass123', 'otp': generate_otp()},
14         'user2': {'password': 'hello456', 'otp': generate_otp()}
15     }
16
17     if userid in valid_users:
18         user_data = valid_users[userid]
19         if user_data['password'] == password and user_data['otp'] == otp_received:
20             return True
21     return False
```

```
22
23
24 def main():
25     server_ip = '0.0.0.0'
26     server_port = 12345
27
28     server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
29     server_socket.bind((server_ip, server_port))
30
31     print("UDP server is listening...")
32
33     while True:
34         data, client_address = server_socket.recvfrom(1024)
35         data = data.decode('utf-8')
36         userid, password, otp_r
37
38         if authenticate_user(userid, password, otp_received):
39             response = "Authentication successful"
40         else:
41             response = "Authentication failed"
42
43         server_socket.sendto(response.encode('utf-8'), client_address)
44
45
46 if __name__ == "__main__":
47     main()
48
```

## Client(udp\_clients.py):



```
Project
udp_server.py × udp_clients.py ×
1 import socket
2
3
4 def main():
5     server_ip = '127.0.0.1'
6     server_port = 12345
7     client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
8
9     userid = input("Enter your userid: ")
10    password = input("Enter your password: ")
11
12    client_socket.sendto(f"{userid},{password}".encode('utf-8'), (server_ip, server_port))
13
14    otp_received = input("Enter the OTP received on your device: ")
15    client_socket.sendto(otp_received.encode('utf-8'), (server_ip, server_port))
16
17    response, _ = client_socket.recvfrom(1024)
18    print(response.decode('utf-8'))
19
20
21 if __name__ == "__main__":
22     main()
23
```

## OUTPUT (in TERMINALS):

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

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

PS C:\My Files\Save Files of apps\Python\Socket Programming\Q4> python server.py
Server is listening for connections...
('127.0.0.1', 14056) authenticated as user1
('127.0.0.1', 14061) disconnected

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

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

PS C:\My Files\Save Files of apps\Python\Socket Programming\Q4> python client.py
User ID: user1
Password: password1
Solve the captcha: Solve the captcha: 2 + 9
Captcha answer: 11
Authentication successful. You can start chatting.
You can start chatting. Type 'exit' to leave.

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

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

PS C:\My Files\Save Files of apps\Python\Socket Programming\Q4> python client.py
User ID: user2
Password: password2
Solve the captcha: Solve the captcha: 4 / 9
Captcha answer: 0.44444444
Authentication failed. Connection closed.
PS C:\My Files\Save Files of apps\Python\Socket Programming\Q4> |
```

## 3. HTTP – Arithmetic Captcha Checking.

## Server(http\_server.py) :

```
http_server.py x http_client.py x
1  import socket
2      import threading
3      import random
4      from http.server import BaseHTTPRequestHandler, HTTPServer
5      from urllib.parse import parse_qs
6
7      # User database (insecure for demonstration purposes)
8      users = {"user1": "password1", "user2": "password2"}
9
10     # Generate a random arithmetic captcha
11     def generate_captcha():
12         num1 = random.randint(1, 10)
13         num2 = random.randint(1, 10)
14         operator = random.choice(["+", "-", "*"])
15         captcha = f"What is {num1} {operator} {num2}?"
16         answer = str(eval(f"{num1} {operator} {num2}"))
17         return captcha, answer
18
19     # HTTP request handler
20     class MyRequestHandler(BaseHTTPRequestHandler):
21         def do_GET(self):
22             if self.path == "/login":
23                 captcha, answer = generate_captcha()
24                 self.send_response(200)
25                 self.send_header("Content-type", "text/html")
26                 self.end_headers()
27                 self.wfile.write(f"Captcha: {captcha}".encode())
28
29                 # Store captcha answer for later verification
30                 self.captcha_answer = answer
31             else:
32                 self.send_response(404)
33                 self.end_headers()
34                 self.wfile.write(b"Page not found")
35
36         def do_POST(self):
37             if self.path == "/authenticate":
38                 content_length = int(self.headers['Content-Length'])
39                 post_data = self.rfile.read(content_length).decode("utf-8")
40                 params = parse_qs(post_data)
41                 username = params['username'][0]
42                 password = params['password'][0]
43                 captcha_response = params['captcha'][0]
44
```

```
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 x http_client.py x
1 import socket
2 import requests
3
4 # Simulate a client requesting a captcha and authentication
5 def client_request():
6     captcha_response = ""
7     with requests.get("http://localhost:8080/login") as response:
8         print(response.text)
9         captcha_response = input("Enter the captcha answer: ")
10
11     username = input("Enter username: ")
12     password = input("Enter password: ")
13
14     payload = {
15         'username': username,
16         'password': password,
17         'captcha': captcha_response
18     }
19
20     response = requests.post("http://localhost:8080/authenticate", data=payload)
21     print(response.text)
22
23 # Simulate multiple clients
24 for _ in range(3):
25     client_request()
```

## OUTPUT :

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

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

PS C:\My Files\Save Files of apps\Python\Socket Programming\Q3> python serve
r.py
Server is waiting for a connection...
Connected to: ('127.0.0.1', 13927)
Generated OTP for user1: 673058
PS C:\My Files\Save Files of apps\Python\Socket Programming\Q3> |

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

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

PS C:\My Files\Save Files of apps\Python\Socket Programming\Q3> python cli
ent.py
Connected to the server.
Enter your user ID: user1
Enter your password: password1
valid
Enter the OTP sent to your registered email/mobile: 673058
OTP verified. Access granted.
PS C:\My Files\Save Files of apps\Python\Socket Programming\Q3> |
```