**TCP\_server**  
  
import socket

# Create a TCP/IP socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Bind the socket to the address and port

host = '127.0.0.1' # Localhost

port = 65432

server\_socket.bind((host, port))

# Listen for incoming connections

server\_socket.listen(1)

print(f"Server listening on {host}:{port}")

# Accept a connection

connection, client\_address = server\_socket.accept()

print(f"Connection established with {client\_address}")

# Receive and send messages

try:

while True:

# Receive data

data = connection.recv(1024) # Buffer size

if not data:

break

print(f"Received from client: {data.decode()}")

# Send response

response = "Message received: " + data.decode()

connection.sendall(response.encode())

except Exception as e:

print(f"An error occurred: {e}")

finally:

# Close the connection

connection.close()

print("Connection closed.")

**TCP\_Client**

import socket

# Create a TCP/IP socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Connect to the server

host = '127.0.0.1' # Localhost

port = 65432

client\_socket.connect((host, port))

print(f"Connected to server at {host}:{port}")

# Send and receive messages

try:

while True:

# Send a message

message = input("Enter message to send to server (or 'exit' to quit): ")

if message.lower() == 'exit':

break

client\_socket.sendall(message.encode())

# Receive response

response = client\_socket.recv(1024)

print(f"Received from server: {response.decode()}")

except Exception as e:

print(f"An error occurred: {e}")

finally:

# Close the socket

client\_socket.close()

print("Connection closed.")

**UDP\_Server**

import socket

# Create a UDP socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Bind the socket to the address and port

host = '127.0.0.1' # Localhost

port = 65432

server\_socket.bind((host, port))

print(f"UDP Server listening on {host}:{port}")

# Receive and respond to messages

try:

while True:

# Receive data from client

data, client\_address = server\_socket.recvfrom(1024) # Buffer size

print(f"Received from {client\_address}: {data.decode()}")

# Send response back to the client

response = "Message received: " + data.decode()

server\_socket.sendto(response.encode(), client\_address)

except Exception as e:

print(f"An error occurred: {e}")

finally:

# Close the socket

server\_socket.close()

print("Server socket closed.")

**UDP\_client**

import socket

# Create a UDP socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Define the server address and port

server\_host = '127.0.0.1' # Localhost

server\_port = 65432

server\_address = (server\_host, server\_port)

try:

while True:

# Send a message to the server

message = input("Enter message to send to server (or 'exit' to quit): ")

if message.lower() == 'exit':

break

client\_socket.sendto(message.encode(), server\_address)

# Receive response from server

response, server = client\_socket.recvfrom(1024)

print(f"Received from server: {response.decode()}")

except Exception as e:

print(f"An error occurred: {e}")

finally:

# Close the socket

client\_socket.close()

print("Client socket closed.")

**Arithmetic operation using TCP**

**Server**

import socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('127.0.0.1', 65432))

server\_socket.listen(1)

print("TCP Arithmetic Server is listening...")

connection, client\_address = server\_socket.accept()

print(f"Connected to {client\_address}")

try:

while True:

data = connection.recv(1024).decode()

if not data:

break

print(f"Received operation: {data}")

try:

result = eval(data) # Evaluate the arithmetic expression

response = f"Result: {result}"

except Exception as e:

response = f"Error: {e}"

connection.sendall(response.encode())

except Exception as e:

print(f"Error: {e}")

finally:

connection.close()

server\_socket.close()

**client**

import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('127.0.0.1', 65432))

try:

while True:

operation = input("Enter arithmetic operation (e.g., 5+3 or 'exit' to quit): ")

if operation.lower() == 'exit':

break

client\_socket.sendall(operation.encode())

result = client\_socket.recv(1024).decode()

print(result)

finally:

client\_socket.close()

**Arithmetic operation using UDP  
Server**

import socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_socket.bind(('127.0.0.1', 65432))

print("UDP Arithmetic Server is listening...")

try:

while True:

data, client\_address = server\_socket.recvfrom(1024)

print(f"Received from {client\_address}: {data.decode()}")

try:

result = eval(data.decode()) # Evaluate the arithmetic expression

response = f"Result: {result}"

except Exception as e:

response = f"Error: {e}"

server\_socket.sendto(response.encode(), client\_address)

except Exception as e:

print(f"Error: {e}")

finally:

server\_socket.close()

**Client**  
import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ('127.0.0.1', 65432)

try:

while True:

operation = input("Enter arithmetic operation (e.g., 5+3 or 'exit' to quit): ")

if operation.lower() == 'exit':

break

client\_socket.sendto(operation.encode(), server\_address)

response, \_ = client\_socket.recvfrom(1024)

print(response.decode())

finally:

client\_socket.close()

**Date & Time**

**TCP\_server**

import socket

from datetime import datetime

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('127.0.0.1', 65432))

server\_socket.listen(1)

print("TCP Date and Time Server is listening...")

connection, client\_address = server\_socket.accept()

print(f"Connected to {client\_address}")

try:

while True:

data = connection.recv(1024).decode()

if not data:

break

if data.lower() == 'datetime':

now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

response = f"Current Date and Time: {now}"

else:

response = "Invalid request. Send 'datetime' to get date and time."

connection.sendall(response.encode())

except Exception as e:

print(f"Error: {e}")

finally:

connection.close()

server\_socket.close()

**TCP\_Client**

import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('127.0.0.1', 65432))

try:

while True:

message = input("Enter 'datetime' to get date and time or 'exit' to quit: ")

if message.lower() == 'exit':

break

client\_socket.sendall(message.encode())

response = client\_socket.recv(1024).decode()

print(response)

finally:

client\_socket.close()

**UDP\_server**

import socket

from datetime import datetime

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_socket.bind(('127.0.0.1', 65432))

print("UDP Date and Time Server is listening...")

try:

while True:

data, client\_address = server\_socket.recvfrom(1024)

print(f"Received from {client\_address}: {data.decode()}")

if data.decode().lower() == 'datetime':

now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

response = f"Current Date and Time: {now}"

else:

response = "Invalid request. Send 'datetime' to get date and time."

server\_socket.sendto(response.encode(), client\_address)

except Exception as e:

print(f"Error: {e}")

finally:

server\_socket.close()

**UDP\_client**  
  
import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ('127.0.0.1', 65432)

try:

while True:

message = input("Enter 'datetime' to get date and time or 'exit' to quit: ")

if message.lower() == 'exit':

break

client\_socket.sendto(message.encode(), server\_address)

response, \_ = client\_socket.recvfrom(1024)

print(response.decode())

finally:

client\_socket.close()

**Multiclient\_communication  
  
Tcp server**

import socket

import threading

def handle\_client(connection, address):

print(f"New connection: {address}")

try:

while True:

data = connection.recv(1024)

if not data:

break

print(f"From {address}: {data.decode()}")

connection.sendall(f"Echo: {data.decode()}".encode())

except Exception as e:

print(f"Error with {address}: {e}")

finally:

connection.close()

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('127.0.0.1', 65432))

server\_socket.listen(5)

print("TCP Multi-client Server is listening...")

try:

while True:

conn, addr = server\_socket.accept()

threading.Thread(target=handle\_client, args=(conn, addr)).start()

except Exception as e:

print(f"Server error: {e}")

finally:

server\_socket.close()

**tcp client**

import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('127.0.0.1', 65432))

try:

while True:

message = input("Enter a message or 'exit' to quit: ")

if message.lower() == 'exit':

break

client\_socket.sendall(message.encode())

response = client\_socket.recv(1024).decode()

print(response)

finally:

client\_socket.close()  
  
**UDP\_server**

import socket

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_socket.bind(('127.0.0.1', 65432))

print("UDP Multi-client Server is listening...")

try:

while True:

data, client\_address = server\_socket.recvfrom(1024)

print(f"From {client\_address}: {data.decode()}")

response = f"Echo: {data.decode()}"

server\_socket.sendto(response.encode(), client\_address)

except Exception as e:

print(f"Error: {e}")

finally:

server\_socket.close()

**UDP\_client**

import socket

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server\_address = ('127.0.0.1', 65432)

try:

while True:

message = input("Enter a message or 'exit' to quit: ")

if message.lower() == 'exit':

break

client\_socket.sendto(message.encode(), server\_address)

response, \_ = client\_socket.recvfrom(1024)

print(response.decode())

finally:

client\_socket.close()