

# **LAPORAN TUGAS PRAKTIKUM PEMROGRAMAN JARINGAN**

**Mata Kuliah: Bahasa Pemrograman Jaringan Komputer**



**Di Susun Oleh:**

**Krisnayanty Kesia Patanda (231401008)**

**Dosen Pengampu: Ucok, S.Kom.,MT**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS INDONESIA TIMUR**

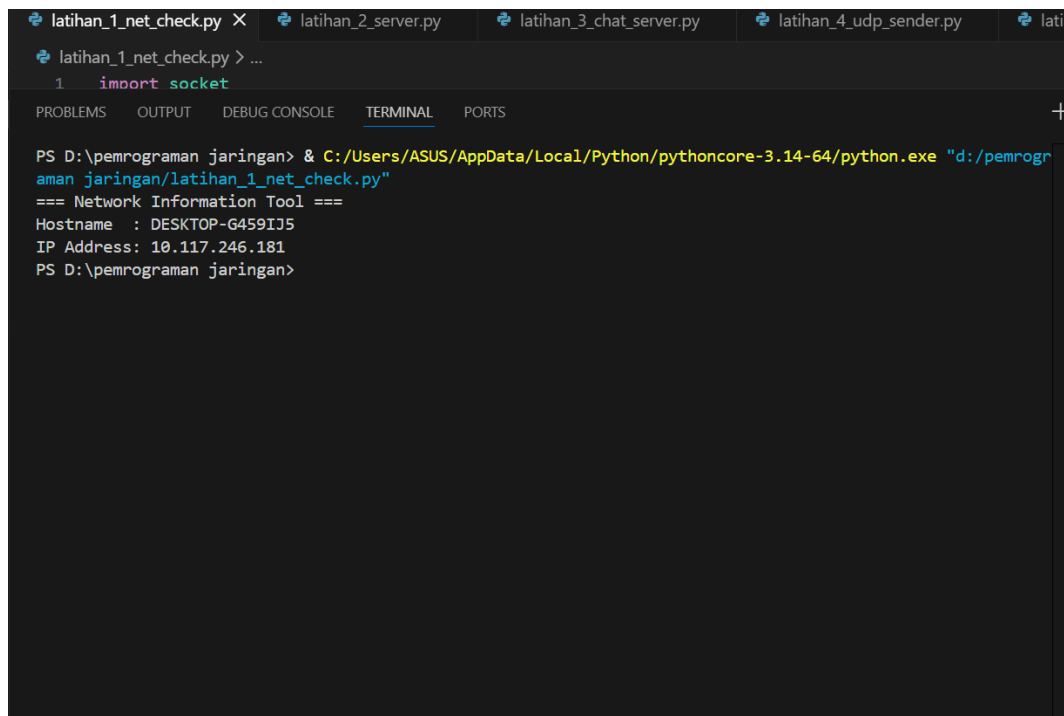
**MAKASSAR**

**2026**

## BAB 1 KONSEP DASAR PEMROGRAMAN JARINGAN

Program pengecekan informasi jaringan menggunakan Python bertujuan untuk menampilkan hostname dan IP address komputer. Program ini memanfaatkan module socket untuk mengambil data jaringan dari sistem operasi, di mana `gethostname()` digunakan untuk mengetahui nama komputer dan `gethostbyname()` untuk memperoleh alamat IP. Hasil informasi kemudian ditampilkan ke layar menggunakan fungsi `print()`. Program ini melatih pemahaman dasar tentang pemrograman jaringan, penggunaan library bawaan Python, serta pengenalan konsep hostname dan IP address dalam jaringan komputer.

### Hasil:



The screenshot shows a Python IDE with several tabs at the top: `latihan_1_net_check.py`, `latihan_2_server.py`, `latihan_3_chat_server.py`, `latihan_4_udp_sender.py`, and `latihan_5_udp_receiver.py`. The active tab is `latihan_1_net_check.py`, which contains the following code:

```
1 import socket
```

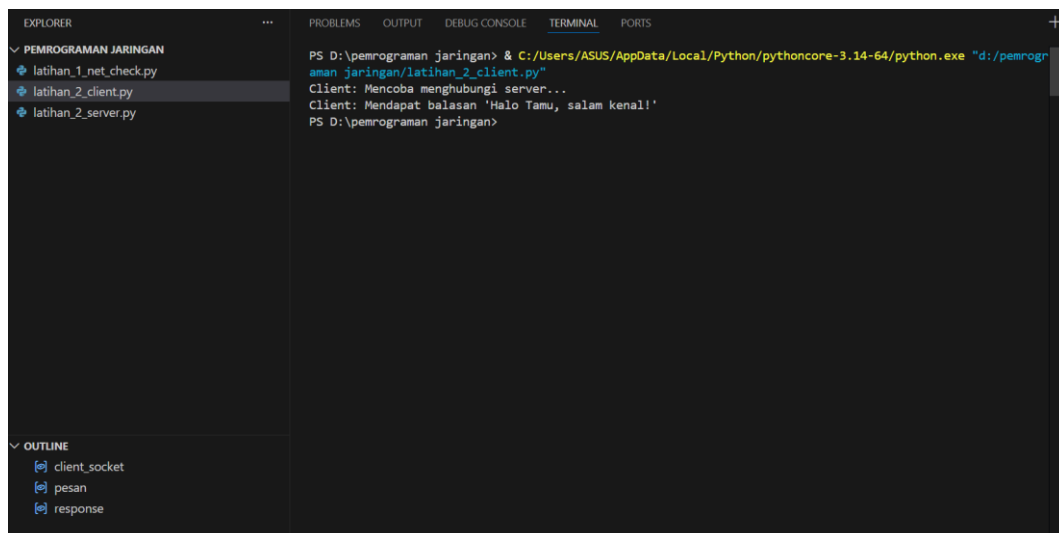
Below the code editor, the `TERMINAL` pane is active, displaying the command prompt output:

```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_1_net_check.py"
=== Network Information Tool ===
Hostname : DESKTOP-G459IJ5
IP Address: 10.117.246.181
PS D:\pemrograman jaringan>
```

## BAB 2 SOCKET API DASAR

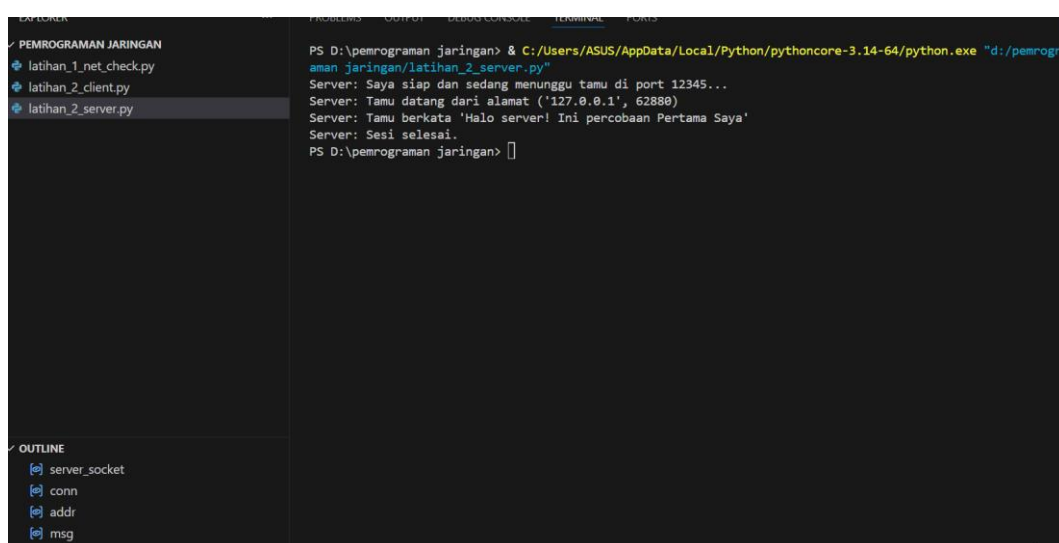
Pada percobaan ini, komunikasi jaringan dilakukan antara client dan server yang berjalan pada satu komputer menggunakan alamat localhost . Server dijalankan terlebih dahulu untuk membuka port dan menunggu koneksi masuk, kemudian client melakukan koneksi ke server dan mengirimkan pesan percobaan. Dari hasil eksekusi terlihat bahwa server berhasil menerima pesan dari client, menampilkan alamat client, serta mengirimkan balasan yang diterima kembali oleh client. Hal ini membuktikan bahwa proses koneksi socket, pengiriman pesan, penerimaan respon, dan penutupan koneksi berjalan dengan baik sesuai konsep dasar pemrograman jaringan menggunakan Python.

### Hasil:



```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_2_client.py"
Client: Mencoba menghubungi server...
Client: Mendapat balasan 'Halo Tamu, salam kenal!'
PS D:\pemrograman jaringan>
```

The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure under 'PEMROGRAMAN JARINGAN', including files like 'latihan\_1\_net\_check.py', 'latihan\_2\_client.py', and 'latihan\_2\_server.py'. The Outline pane shows 'client\_socket', 'pesan', and 'response'. The Terminal pane on the right shows the execution of the client script, which connects to the server and receives a response.



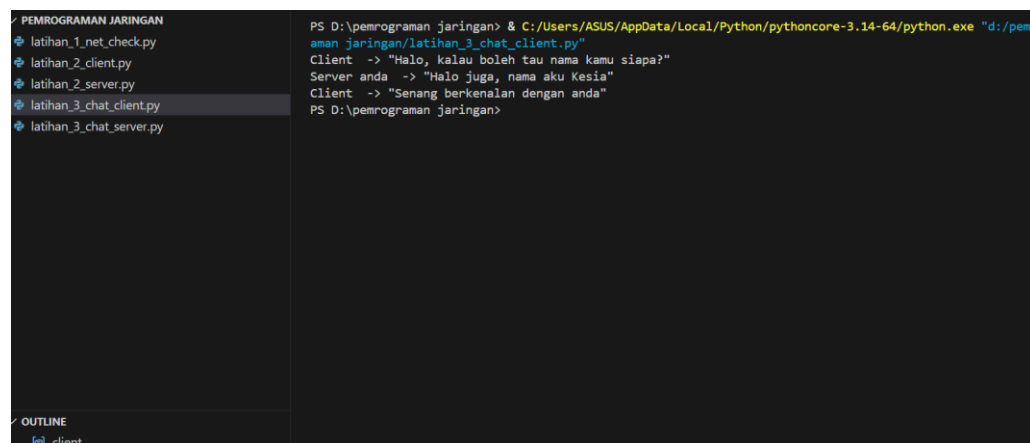
```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_2_server.py"
Server: Saya siap dan sedang menunggu tamu di port 12345...
Server: Tamu datang dari alamat ('127.0.0.1', 62880)
Server: Tamu berkata 'Halo server! Ini percobaan Pertama Saya'
Server: Sesi selesai.
PS D:\pemrograman jaringan>
```

The screenshot shows the VS Code interface with the Explorer pane on the left displaying the project structure under 'PEMROGRAMAN JARINGAN', including files like 'latihan\_1\_net\_check.py', 'latihan\_2\_client.py', and 'latihan\_2\_server.py'. The Outline pane shows 'server\_socket', 'conn', 'addr', and 'msg'. The Terminal pane on the right shows the execution of the server script, which listens on port 12345, receives a connection from 127.0.0.1, and processes the message.

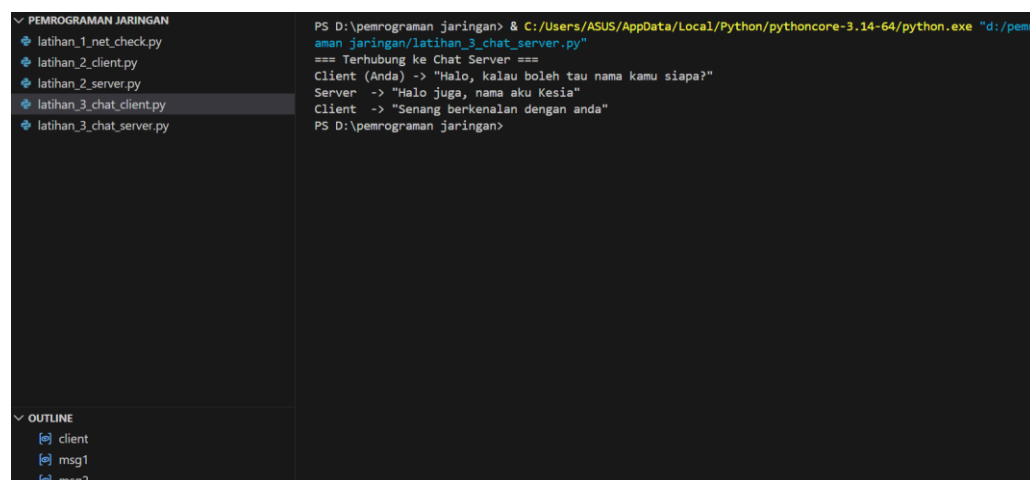
## BAB 3 PROTOKOL TCP (APLIKASI CHAT)

Pada materi ini menampilkan implementasi aplikasi chat berbasis protokol TCP menggunakan bahasa pemrograman Python, di mana komunikasi dilakukan antara satu server dan satu client melalui koneksi jaringan yang andal. Server terlebih dahulu dijalankan untuk membuka port dan menunggu koneksi masuk dari client, kemudian client melakukan koneksi ke server untuk memulai sesi percakapan. Setelah koneksi TCP berhasil terbentuk, client mengirimkan pesan pertama yang diterima oleh server, lalu server membalas pesan tersebut sehingga terjadi pertukaran pesan secara dua arah. Seluruh proses komunikasi ini berlangsung dalam satu sesi koneksi yang sama, sehingga setiap pesan dapat dikirim dan diterima secara berurutan tanpa kehilangan data. Penggunaan protokol TCP pada aplikasi chat ini memastikan keandalan komunikasi, menjaga urutan pesan, serta menjamin bahwa data yang dikirim sampai ke tujuan dengan benar, sesuai dengan prinsip dasar pemrograman jaringan aplikasi chat.

### Hasil:



```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_3_chat_client.py"
Client -> "Halo, kalau boleh tau nama kamu siapa?"
Server anda -> "Halo juga, nama aku Kesia"
Client -> "Senang berkenalan dengan anda"
PS D:\pemrograman jaringan>
```

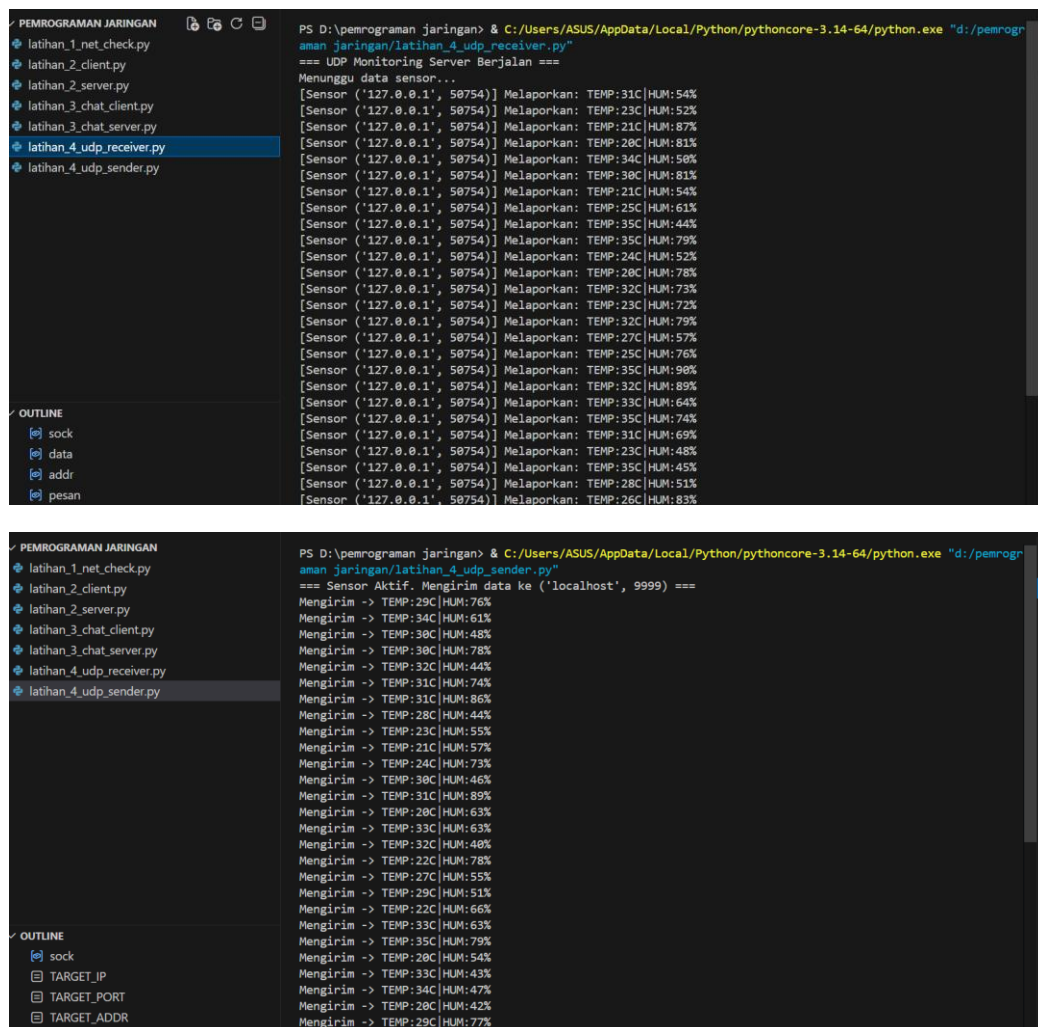


```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_3_chat_server.py"
=== Terhubung ke Chat Server ===
Client (Anda) -> "Halo, kalau boleh tau nama kamu siapa?"
Server -> "Halo juga, nama aku Kesia"
Client -> "Senang berkenalan dengan anda"
PS D:\pemrograman jaringan>
```

## BAB 4 PROTOKOL UDP (STREAMING & BROADCASTING)

Program ini menunjukkan penerapan komunikasi jaringan menggunakan protokol UDP pada simulasi pengiriman data sensor. Program sender bertindak sebagai sensor yang secara terus-menerus mengirimkan data suhu (TEMP) dan kelembapan (HUM) ke alamat localhost dan port tertentu tanpa membangun koneksi terlebih dahulu, sedangkan program receiver berfungsi sebagai server monitoring yang menerima dan menampilkan data sensor tersebut. Karena menggunakan protokol UDP, pengiriman data berlangsung cepat dan sederhana tanpa mekanisme pengecekan koneksi, sehingga sesuai untuk aplikasi monitoring sensor yang membutuhkan kecepatan pengiriman data secara real-time.

### Hasil:



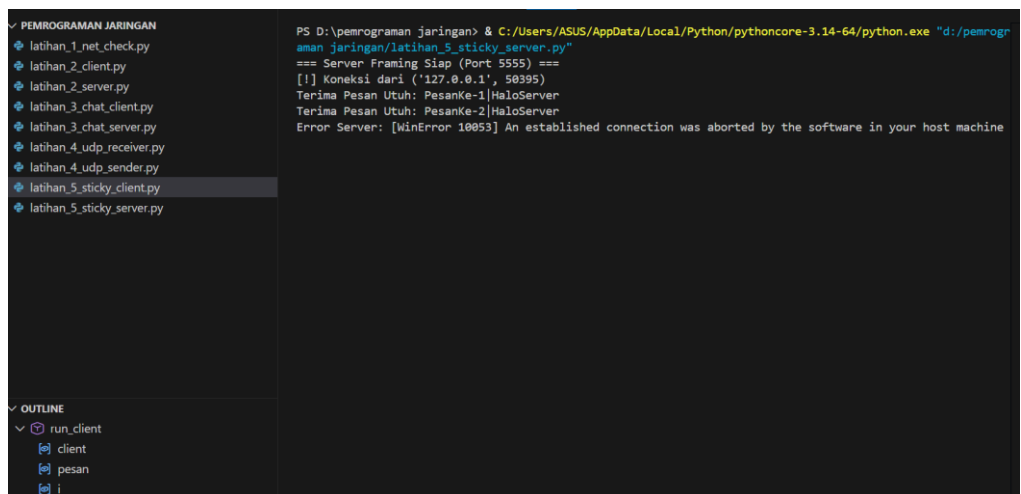
```
PS D:\pemrograman jaringan> C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_4_udp_receiver.py"
=== UDP Monitoring Server Berjalan ===
Menunggu data sensor...
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:31C|HUM:54%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:23C|HUM:52%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:21C|HUM:87%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:20C|HUM:81%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:34C|HUM:50%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:30C|HUM:81%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:21C|HUM:54%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:25C|HUM:61%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:35C|HUM:44%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:35C|HUM:79%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:24C|HUM:52%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:20C|HUM:78%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:32C|HUM:73%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:23C|HUM:72%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:32C|HUM:79%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:27C|HUM:57%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:25C|HUM:76%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:35C|HUM:90%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:32C|HUM:89%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:33C|HUM:64%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:35C|HUM:74%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:31C|HUM:69%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:23C|HUM:48%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:35C|HUM:45%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:28C|HUM:51%
[Sensor ('127.0.0.1', 50754)] Melaporkan: TEMP:26C|HUM:83%

PS D:\pemrograman jaringan> C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_4_udp_sender.py"
=== Sensor Aktif. Mengirim data ke ('localhost', 9999) ===
Mengirim -> TEMP:29C|HUM:76%
Mengirim -> TEMP:34C|HUM:61%
Mengirim -> TEMP:30C|HUM:48%
Mengirim -> TEMP:30C|HUM:78%
Mengirim -> TEMP:32C|HUM:44%
Mengirim -> TEMP:31C|HUM:74%
Mengirim -> TEMP:31C|HUM:86%
Mengirim -> TEMP:28C|HUM:44%
Mengirim -> TEMP:23C|HUM:55%
Mengirim -> TEMP:21C|HUM:57%
Mengirim -> TEMP:24C|HUM:73%
Mengirim -> TEMP:30C|HUM:46%
Mengirim -> TEMP:31C|HUM:46%
Mengirim -> TEMP:20C|HUM:63%
Mengirim -> TEMP:33C|HUM:63%
Mengirim -> TEMP:32C|HUM:40%
Mengirim -> TEMP:22C|HUM:78%
Mengirim -> TEMP:27C|HUM:55%
Mengirim -> TEMP:29C|HUM:51%
Mengirim -> TEMP:22C|HUM:66%
Mengirim -> TEMP:33C|HUM:63%
Mengirim -> TEMP:35C|HUM:79%
Mengirim -> TEMP:20C|HUM:54%
Mengirim -> TEMP:33C|HUM:43%
Mengirim -> TEMP:34C|HUM:47%
Mengirim -> TEMP:20C|HUM:42%
Mengirim -> TEMP:29C|HUM:77%
```

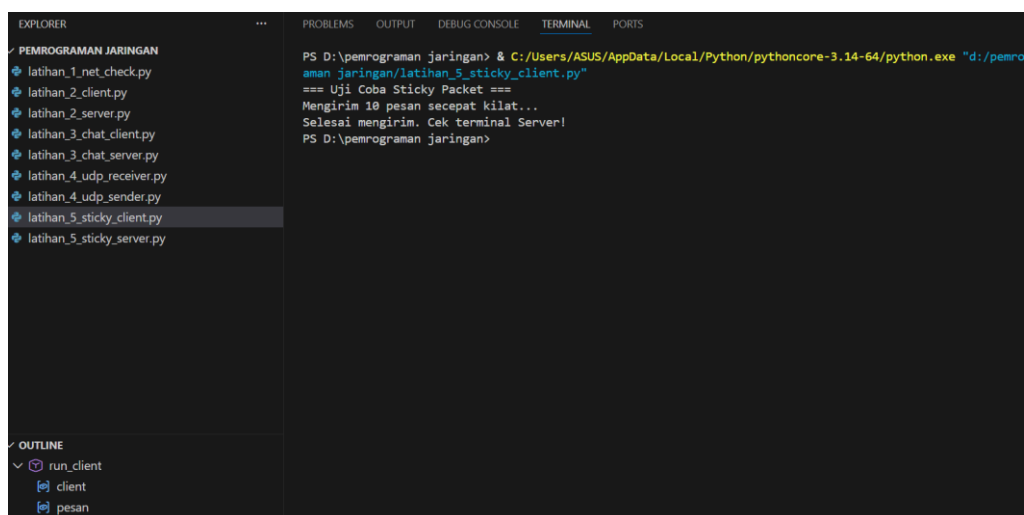
## BAB 5 ERROR HANDLING & FRAMING DAT

Materi ini menjelaskan konsep sticky packet (packet lengket) pada komunikasi TCP melalui simulasi client–server menggunakan Python, di mana server melakukan framing data untuk memastikan pesan diterima secara utuh. Client mengirimkan beberapa pesan secara cepat ke server melalui koneksi TCP yang bersifat stream, sehingga data dapat tiba secara beruntun tanpa batas pesan yang jelas. Di sisi server, mekanisme framing digunakan untuk memisahkan dan menyusun kembali data yang diterima agar setiap pesan dapat dikenali dengan benar, seperti terlihat dari keluaran “Terima Pesan Utuh”. Proses ini menunjukkan bahwa pada TCP tidak ada pemisahan paket secara otomatis seperti pada UDP, sehingga programmer perlu menambahkan logika khusus untuk menangani penggabungan atau pemotongan data agar komunikasi client–server tetap akurat dan stabil.

### Hasil:



```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_5_sticky_server.py"
=== Server Framing Siap (Port 5555) ===
[!] Koneksi dari ('127.0.0.1', 50395)
Terima Pesan Utuh: PesanKe-1|HaloServer
Terima Pesan Utuh: PesanKe-2|HaloServer
Error Server: [WinError 10053] An established connection was aborted by the software in your host machine
```

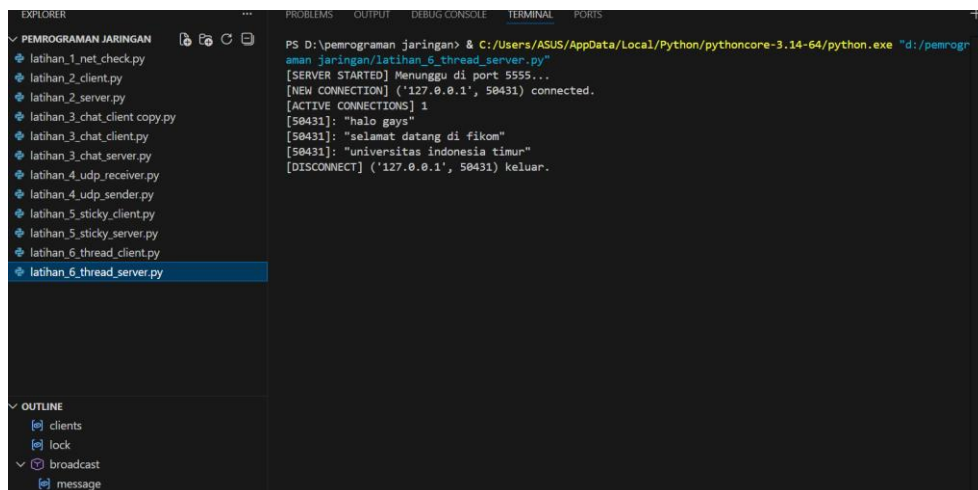


```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_5_sticky_client.py"
=== Uji Coba Sticky Packet ===
Mengirim 10 pesan secepat kilat...
Selesai mengirim. Cek terminal Server!
PS D:\pemrograman jaringan>
```

## BAB 6 CONCURRENCY PART I-THREADING

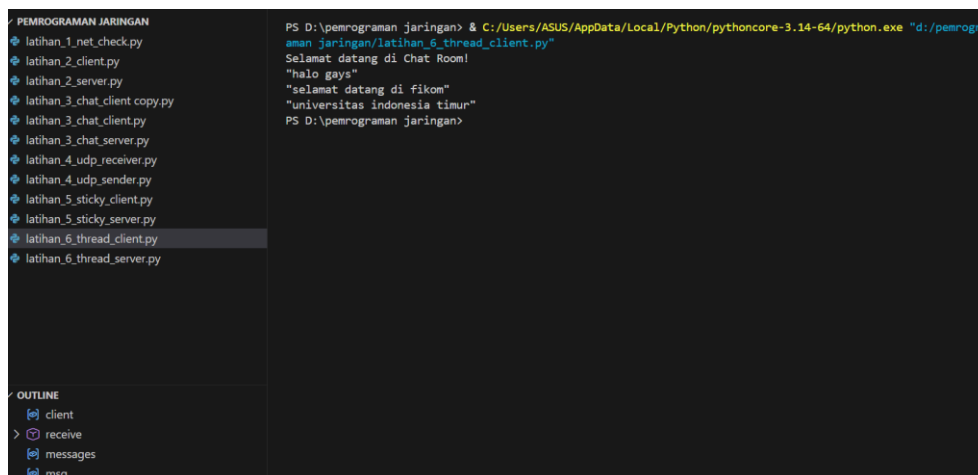
Materi ini menunjukkan implementasi program chat client–server berbasis TCP dengan multithreading menggunakan Python, di mana server mampu menangani beberapa koneksi klien secara bersamaan. Server dijalankan pada port 5555 dan menampilkan status seperti server dimulai, koneksi baru dari klien, jumlah koneksi aktif, pesan yang diterima, hingga notifikasi klien keluar. Setiap klien yang terhubung dilayani oleh thread terpisah sehingga komunikasi dapat berlangsung tanpa saling mengganggu. Di sisi klien, program menampilkan pesan sambutan “Selamat datang di Chat Room!” dan memungkinkan pengguna mengirim beberapa pesan ke server. Konsep ini menunjukkan penggunaan socket, thread, dan mekanisme komunikasi dua arah (send–receive) yang umum digunakan pada aplikasi jaringan real-time seperti chat room.

### Hasil:



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays a file tree for 'PEMROGRAMAN JARINGAN' with files like 'latihan\_1\_net\_check.py', 'latihan\_2\_client.py', 'latihan\_2\_server.py', 'latihan\_3\_chat\_client.py', 'latihan\_3\_chat\_client copy.py', 'latihan\_3\_chat\_server.py', 'latihan\_4\_udp\_receiver.py', 'latihan\_4\_udp\_sender.py', 'latihan\_5\_sticky\_client.py', 'latihan\_5\_sticky\_server.py', 'latihan\_6\_thread\_client.py', and 'latihan\_6\_thread\_server.py'. The 'latihan\_6\_thread\_server.py' file is selected. The bottom-left pane shows the Outline view with sections for 'clients', 'lock', 'broadcast', and 'message'. The main editor pane shows the terminal output for the command 'python.exe "d:/pemrograman jaringan/latihan\_6\_thread\_server.py"'. The output shows the server starting on port 5555, receiving a new connection from '127.0.0.1' on port 50431, and displaying active connections. It then shows messages received from the client: 'halo gays', 'selamat datang di fikom', and 'universitas indonesia timur', followed by a disconnect message.

```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_6_thread_server.py"
[SERVER STARTED] Menunggu di port 5555...
[NEW CONNECTION] ('127.0.0.1', 50431) connected.
[ACTIVE CONNECTIONS] 1
[50431]: "halo gays"
[50431]: "selamat datang di fikom"
[50431]: "universitas indonesia timur"
[DISCONNECT] ('127.0.0.1', 50431) keluar.
```



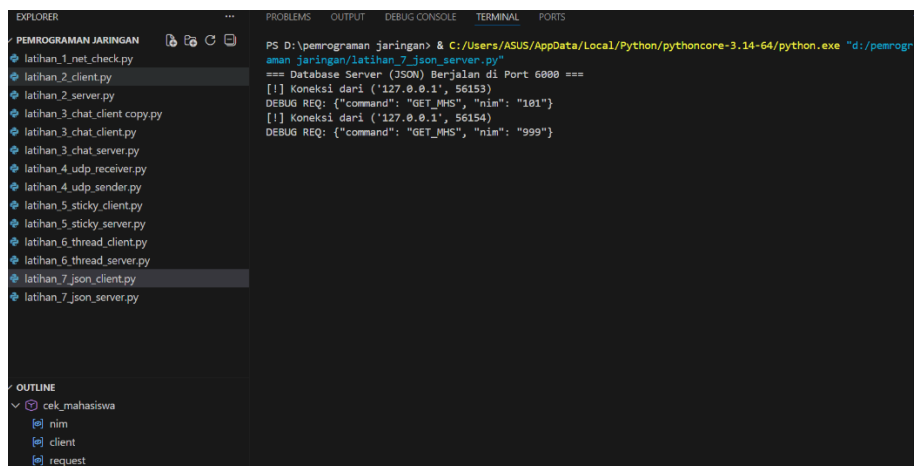
The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane displays the same file tree as the previous screenshot, with 'latihan\_6\_thread\_client.py' selected. The bottom-left pane shows the Outline view with sections for 'client', 'receive', 'messages', and 'msg'. The main editor pane shows the terminal output for the command 'python.exe "d:/pemrograman jaringan/latihan\_6\_thread\_client.py"'. The output shows the client sending messages to the server: 'Selamat datang di Chat Room!', 'halo gays', 'selamat datang di fikom', and 'universitas indonesia timur', followed by a disconnect message.

```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_6_thread_client.py"
Selamat datang di Chat Room!
"halo gays"
"selamat datang di fikom"
"universitas indonesia timur"
PS D:\pemrograman jaringan>
```

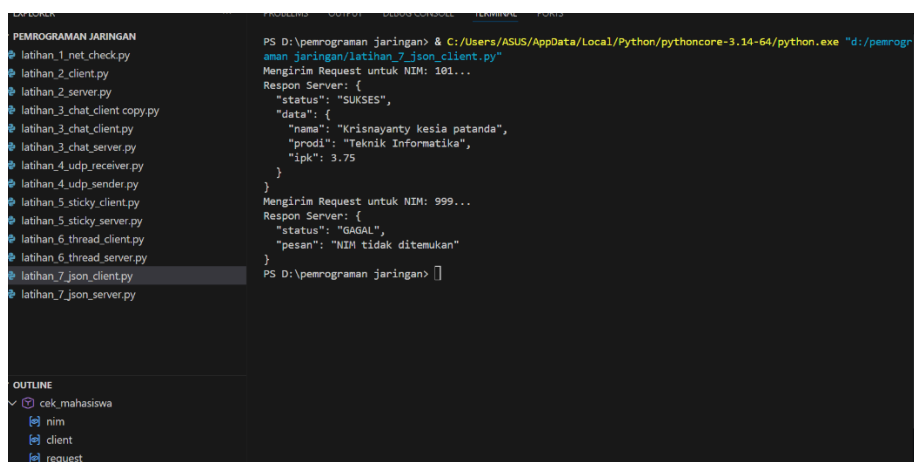
## BAB 7 SERIALISASI DATA (JSON & PICKEL)

Pada materi ini ditunjukkan proses pengujian layanan server berbasis JSON pada pemrograman jaringan yang dijalankan melalui terminal. Server dijalankan pada port 6000 dan menampilkan informasi setiap koneksi yang masuk dari client, termasuk alamat IP dan port client serta data request JSON yang diterima. Client kemudian mengirimkan permintaan pengecekan data mahasiswa berdasarkan NIM, dan hasil respon server ditampilkan langsung di sisi client dalam bentuk JSON. Dari hasil tersebut terlihat dua kondisi, yaitu saat NIM valid server mengembalikan status SUKSES beserta data mahasiswa, dan saat NIM tidak valid server mengembalikan status GAGAL dengan pesan kesalahan. Tampilan ini memperlihatkan alur koneksi, proses request–response, serta cara server dan client saling berinteraksi secara real time menggunakan socket dan format JSON.

### Hasil:



```
PS D:\pemrograman_jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman_jaringan/latihan_7_json_server.py"
=== Database Server (JSON) Berjalan di Port 6000 ===
[!] Koneksi dari ('127.0.0.1', 56153)
DEBUG REQ: {"command": "GET_MHS", "nim": "101"}
[!] Koneksi dari ('127.0.0.1', 56154)
DEBUG REQ: {"command": "GET_MHS", "nim": "999"}
```



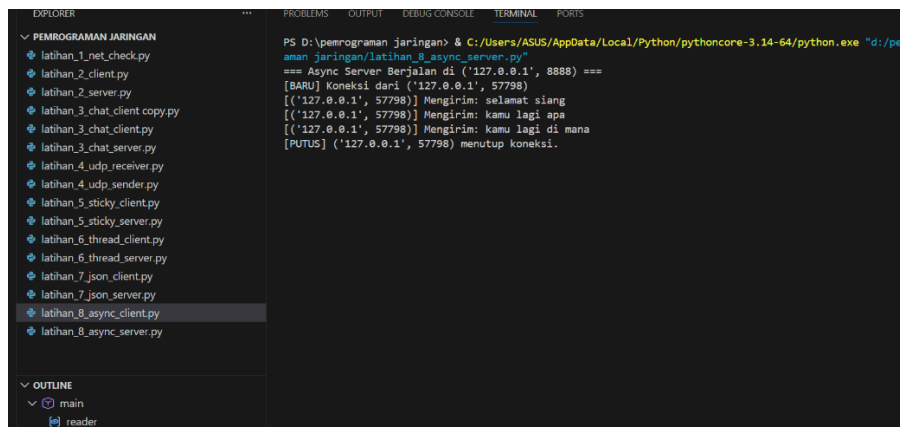
```
PS D:\pemrograman_jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman_jaringan/latihan_7_json_client.py"
Mengirim Request untuk NIM: 101...
Respon Server: {
  "status": "SUKSES",
  "data": {
    "nama": "Krisnayanty kesia patanda",
    "prodi": "Teknik Informatika",
    "ipk": 3.75
  }
}
Mengirim Request untuk NIM: 999...
Respon Server: {
  "status": "GAGAL",
  "pesan": "NIM tidak ditemukan"
}
PS D:\pemrograman_jaringan>
```



## BAB 8 ASYNCHRONOUS I/O (CONCURRENCY PART II)

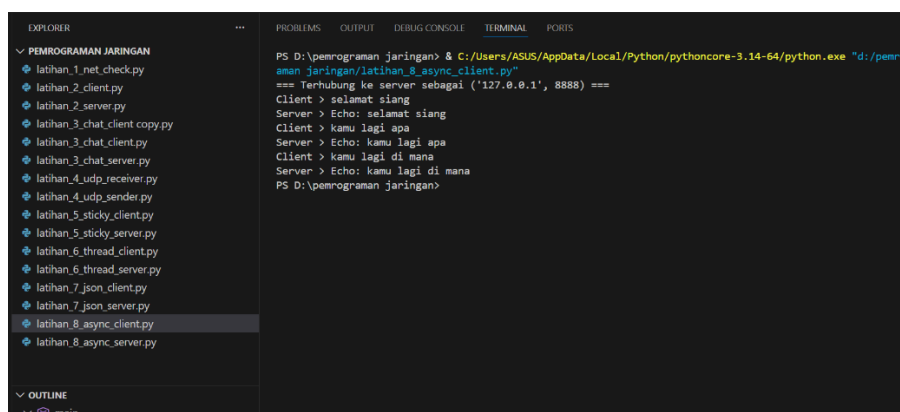
Program ini menunjukkan penerapan komunikasi client–server asynchronous (asyncio) pada pemrograman jaringan Python, di mana server berjalan non-blocking pada alamat 127.0.0.1 port 8888 dan mampu menangani koneksi client tanpa harus menunggu proses lain selesai. Pada gambar terlihat server async berhasil berjalan, menerima koneksi dari client, lalu membaca pesan yang dikirim seperti “*selamat siang*”, “*kamu lagi apa*”, dan “*kamu lagi di mana*”, kemudian memprosesnya secara berurutan dalam satu koneksi. Di sisi client, pengguna mengirim pesan melalui terminal dan langsung menerima balasan dari server berupa *Echo*, yang menandakan server membaca data melalui reader dan mengirimkan kembali respons menggunakan writer. Setelah komunikasi selesai, client menutup koneksi dan server mendeteksi kondisi tersebut lalu menampilkan status *memutus koneksi*. Alur ini membuktikan bahwa model async memungkinkan komunikasi jaringan berjalan efisien, responsif, dan terstruktur tanpa menggunakan banyak thread.

### HASIL:



The screenshot shows the VS Code interface with the Explorer pane on the left displaying a file tree for 'PEMROGRAMAN JARINGAN'. The file 'latihan\_8\_async\_server.py' is selected. The Terminal pane on the right shows the command prompt output for running the server. The output indicates the server is running on 127.0.0.1:8888 and receives three messages from a client: 'selamat siang', 'kamu lagi apa', and 'kamu lagi di mana'. The server responds with 'Echo: selamat siang', 'Echo: kamu lagi apa', and 'Echo: kamu lagi di mana' respectively. Finally, the client closes the connection, and the server prints 'memutus koneksi'.

```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_8_async_server.py"
=== Async Server Berjalan di ('127.0.0.1', 8888) ===
[BARU] Koneksi dari ('127.0.0.1', 57798)
[('127.0.0.1', 57798)] Mengirim: selamat siang
[('127.0.0.1', 57798)] Mengirim: kamu lagi apa
[('127.0.0.1', 57798)] Mengirim: kamu lagi di mana
[PUTUS] ('127.0.0.1', 57798) menutup koneksi.
```



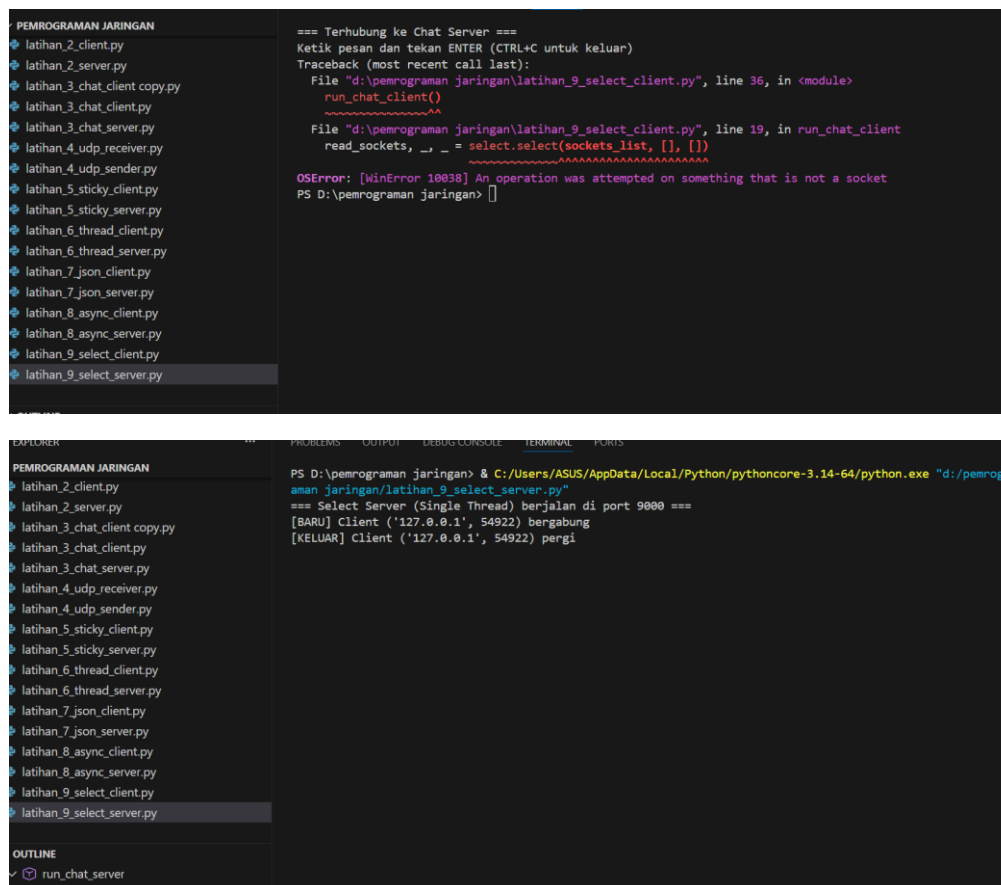
The screenshot shows the VS Code interface with the Explorer pane on the left displaying a file tree for 'PEMROGRAMAN JARINGAN'. The file 'latihan\_8\_async\_client.py' is selected. The Terminal pane on the right shows the command prompt output for running the client. The output indicates the client connects to the server at 127.0.0.1:8888 and sends three messages: 'selamat siang', 'kamu lagi apa', and 'kamu lagi di mana'. The server responds with 'Echo: selamat siang', 'Echo: kamu lagi apa', and 'Echo: kamu lagi di mana' respectively. Finally, the client closes the connection, and the server prints 'memutus koneksi'.

```
PS D:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "d:/pemrograman jaringan/latihan_8_async_client.py"
=== Terhubung ke server sebagai ('127.0.0.1', 8888) ===
Client > selamat siang
Server > Echo: selamat siang
Client > kamu lagi apa
Server > Echo: kamu lagi apa
Client > kamu lagi di mana
Server > Echo: kamu lagi di mana
PS D:\pemrograman jaringan>
```

## BAB 9 I/O MULTIPLEXING (SELECT&POLL)

Program ini menjelaskan cara kerja chat client–server berbasis select() pada Python yang memungkinkan server memantau banyak koneksi secara efisien dalam satu proses. Pada tampilan server terlihat bahwa socket utama berhasil membuka layanan di port 9000, menerima client baru, lalu mendeteksi saat client tersebut keluar tanpa membuat server berhenti. Sebaliknya, pada sisi client muncul pesan error yang menandakan adanya kesalahan saat pemanggilan select(), yaitu ketika daftar socket yang dipantau berisi objek yang tidak valid sebagai socket. Dari sini dapat dipahami bahwa select() hanya dapat bekerja dengan descriptor socket yang aktif, sehingga pengelolaan daftar socket menjadi bagian penting dalam arsitektur program. Materi ini menekankan konsep I/O multiplexing, di mana satu thread dapat menangani banyak koneksi sekaligus dengan tetap menjaga stabilitas dan kontrol alur komunikasi jaringan.

### HASIL:



The first screenshot shows a Python IDE with a file explorer on the left listing files under 'PEMROGRAMAN JARINGAN'. The main editor displays a traceback for a client script, 'latihan\_9\_select\_client.py', showing an 'OSError: [WinError 10038] An operation was attempted on something that is not a socket' at line 19. The second screenshot shows the same IDE with the terminal window open. The terminal output shows the server script, 'latihan\_9\_select\_server.py', running successfully on port 9000, receiving a client connection from '127.0.0.1' at port 54922, and then the client script exiting.

```
PEMROGRAMAN JARINGAN
latihan_2_client.py
latihan_2_server.py
latihan_3_chat_client copy.py
latihan_3_chat_client.py
latihan_3_chat_server.py
latihan_4_udp_receiver.py
latihan_4_udp_sender.py
latihan_5_sticky_client.py
latihan_5_sticky_server.py
latihan_6_thread_client.py
latihan_6_thread_server.py
latihan_7_json_client.py
latihan_7_json_server.py
latihan_8_async_client.py
latihan_8_async_server.py
latihan_9_select_client.py
latihan_9_select_server.py

=== Terhubung ke Chat Server ===
Ketik pesan dan tekan ENTER (CTRL+C untuk keluar)
Traceback (most recent call last):
  File "d:\pemrograman jaringan\latihan_9_select_client.py", line 36, in <module>
    run_chat_client()
    ~~~~~^
  File "d:\pemrograman jaringan\latihan_9_select_client.py", line 19, in run_chat_client
    read_sockets, _, _ = select.select(sockets_list, [], [])
    ~~~~~^
OSError: [WinError 10038] An operation was attempted on something that is not a socket
PS D:\pemrograman jaringan>

PEMROGRAMAN JARINGAN
latihan_2_client.py
latihan_2_server.py
latihan_3_chat_client copy.py
latihan_3_chat_client.py
latihan_3_chat_server.py
latihan_4_udp_receiver.py
latihan_4_udp_sender.py
latihan_5_sticky_client.py
latihan_5_sticky_server.py
latihan_6_thread_client.py
latihan_6_thread_server.py
latihan_7_json_client.py
latihan_7_json_server.py
latihan_8_async_client.py
latihan_8_async_server.py
latihan_9_select_client.py
latihan_9_select_server.py

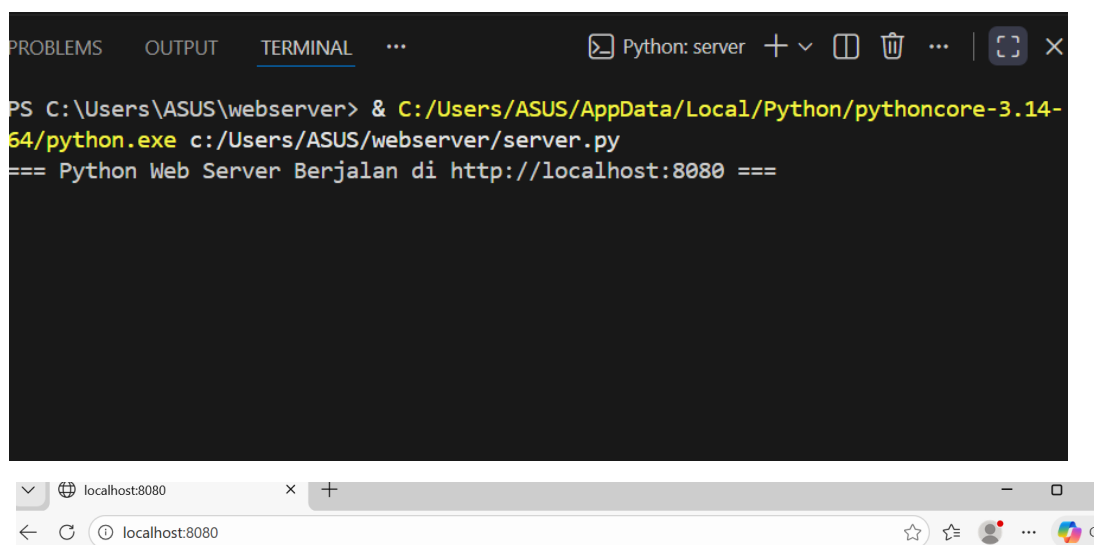
OUTLINE
run_chat_server
latihan_9_select_server.py
```

```
PS D:\pemrograman jaringan> & C:\Users\ASUS\AppData\Local\Python\pythoncore-3.14-64\python.exe "d:\pemrograman jaringan\latihan_9_select_server.py"
=== Select Server (Single Thread) berjalan di port 9000 ===
[BARU] Client ('127.0.0.1', 54922) bergabung
[KELUAR] Client ('127.0.0.1', 54922) pergi
```

## BAB 10 PROTOKOL HTTP & WEB SERVER

Program ini menunjukkan implementasi web server sederhana menggunakan Python Socket yang berjalan secara lokal pada alamat `http://localhost:8080`. Pada terminal terlihat server berhasil dijalankan dan siap menerima permintaan dari browser, sedangkan pada tampilan browser ditunjukkan halaman HTML sederhana dengan pesan “*Web Server Berhasil*” dan keterangan bahwa server berjalan menggunakan Python Socket. Alur kerjanya adalah server membuka socket, melakukan binding ke port 8080, lalu menunggu request HTTP dari client (browser), setelah itu server mengirimkan respons berupa header HTTP dan konten HTML yang kemudian dirender oleh browser. Materi ini memperkenalkan konsep dasar komunikasi HTTP di atas TCP, hubungan antara browser sebagai client dan Python sebagai server, serta membuktikan bahwa tanpa framework tambahan pun Python dapat digunakan untuk membangun web server dasar.

### HASIL:



The image shows two windows side-by-side. The top window is a terminal with a dark background. It shows the command prompt at `C:\Users\ASUS\webserver>` and the command `& C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/ASUS/webserver/server.py` being executed. Below the command, it says `=== Python Web Server Berjalan di http://localhost:8080 ===`. The bottom window is a web browser with a single tab titled `localhost:8080`. The address bar shows `localhost:8080`. The page content displays the text **Web Server Berhasil** in a large, bold font, followed by the text `Ini berjalan menggunakan Python Socket` in a smaller font.

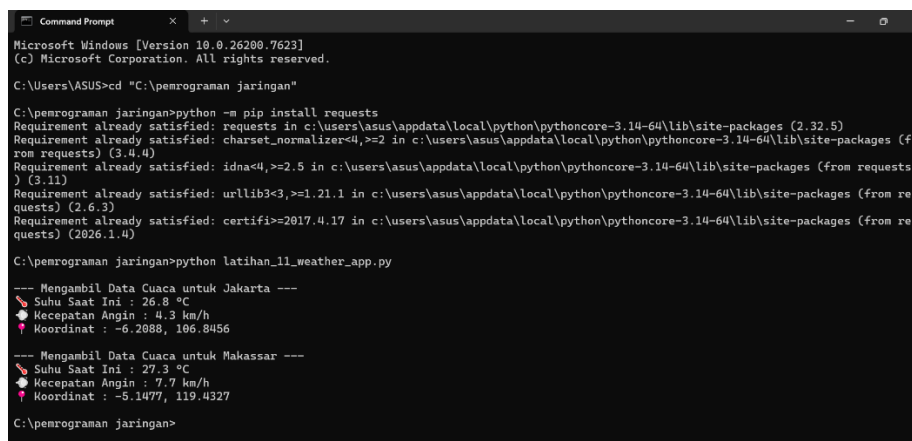
### Web Server Berhasil

Ini berjalan menggunakan Python Socket

## BAB 11 REST API & WEB SERVER

Program ini merupakan aplikasi client jaringan berbasis HTTP menggunakan library requests pada Python untuk mengambil data cuaca secara real-time dari layanan web (API). Pada gambar terlihat proses instalasi library requests yang memastikan dependensi tersedia, lalu program latihan\_11\_weather\_app.py dijalankan melalui terminal dan berhasil menampilkan informasi cuaca untuk beberapa kota seperti Jakarta dan Makassar. Aplikasi ini bekerja dengan mengirim permintaan HTTP ke server cuaca menggunakan koordinat lintang dan bujur, kemudian menerima respons dalam format data yang diproses untuk menampilkan suhu saat ini, kecepatan angin, serta koordinat lokasi. Materi ini memperkenalkan konsep komunikasi client-server berbasis web service, pemanfaatan API eksternal dalam pemrograman jaringan, serta bagaimana Python dapat digunakan sebagai client untuk mengambil dan mengolah data dari internet secara terstruktur.

### HASIL:



```
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>cd "C:\pemrograman jaringan"

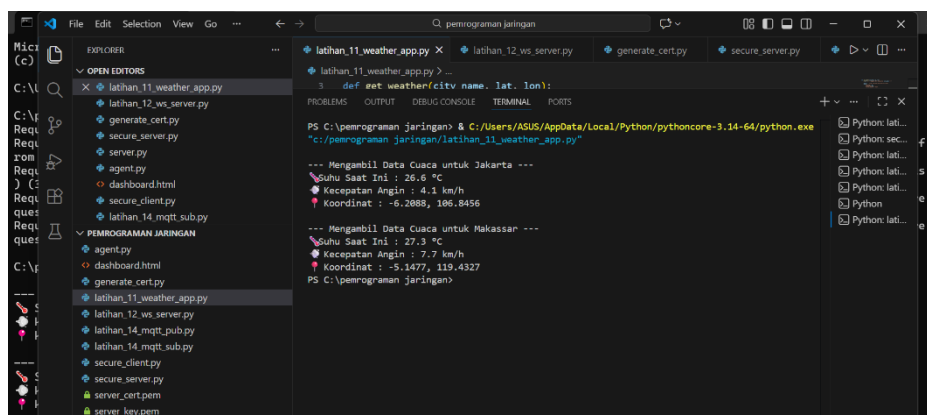
C:\pemrograman jaringan>python -m pip install requests
Requirement already satisfied: requests in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (2.32.5)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from requests) (2.6.2)
Requirement already satisfied: certifi<2017.4.17 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from requests) (2026.1.4)

C:\pemrograman jaringan>python latihan_11_weather_app.py

--- Mengambil Data Cuaca untuk Jakarta ---
Suhu Saat Ini : 26.8 °C
Kecepatan Angin : 4.3 km/h
Koordinat : -6.2088, 106.8456

--- Mengambil Data Cuaca untuk Makassar ---
Suhu Saat Ini : 27.3 °C
Kecepatan Angin : 7.7 km/h
Koordinat : -5.1477, 119.4327

C:\pemrograman jaringan>
```



## BAB 12 REAL-TIME COMMUNICATION (WEBSOCKET)

Program ini merupakan aplikasi pemantauan harga saham BBCA secara real-time yang menggunakan WebSocket sebagai media komunikasi antara server dan client. Pada sisi server (Python), terlihat server berjalan di `ws://localhost:6789` dan secara kontinu melakukan broadcast data harga saham dalam format JSON (berisi simbol saham, harga, dan timestamp) ke client yang terhubung. Setiap kali ada client baru bergabung, server langsung mengirimkan update harga terbaru. Pada sisi client (dashboard HTML), halaman web menampilkan judul *Pantauan Saham BBCA (Real-time)* dan memperbarui harga saham (misalnya Rp 8066) secara otomatis tanpa perlu refresh halaman, menandakan koneksi WebSocket berhasil dengan status *Connected to Server*. Dengan mekanisme ini, perubahan data dari server dapat langsung diterima dan ditampilkan secara instan di browser.


### HASIL:

```
:\\pemrograman jaringan>python latihan_12_ws_server.py
== WebSocket Server running on ws://localhost:6789 ==
NEW] Client bergabung.
BROADCAST] {"symbol": "BBCA", "price": 8125, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8240, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8328, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8002, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8436, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8375, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8303, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8108, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8182, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8066, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8450, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8148, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8139, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8333, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8263, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8124, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8362, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8265, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8136, "timestamp": "Live"} -> ke 1 clients
BROADCAST] {"symbol": "BBCA", "price": 8288, "timestamp": "Live"} -> ke 1 clients
```



### Pantauan Saham BBCA (Real-time)

**Rp 8066**

Connected to Server 

## BAB 13 KEAMANAN JARINGAN (NETWORK SECURITY)

Materi menunjukkan proses implementasi komunikasi client-server yang aman menggunakan SSL/TLS dengan Python. Dimulai dari instalasi library cryptography, pembuatan sertifikat digital dan private key melalui file generate\_cert.py, lalu menjalankan secure\_server.py sebagai server yang mendengarkan koneksi pada port tertentu. Server berhasil melakukan SSL handshake dengan klien, menandakan proses enkripsi TLS berjalan dengan baik menggunakan cipher modern (TLS 1.3). Selanjutnya, klien (secure\_client.py) berhasil terhubung dan mengirim pesan yang terenkripsi, kemudian server menampilkan pesan tersebut dalam bentuk terdekripsi. Hal ini membuktikan bahwa data yang dikirimkan melalui jaringan terlindungi dari penyadapan dan komunikasi berlangsung secara aman.

## HASIL:

The screenshot displays a Windows IDE interface. On the left, the 'EXPLORER' pane shows the project structure for 'PEMROGRAMAN JARINGAN'. The files listed are 'dashboard.html', 'generate\_cert.py', 'latihan\_11\_weather\_app.py', 'latihan\_12\_ws\_server.py', 'secure\_client.py', 'secure\_server.py', 'server\_cert.pem', and 'server\_key.pem'. The 'generate\_cert.py' file is currently selected. On the right, the 'TERMINAL' pane shows the execution of the command 'python generate\_cert.py'. The output indicates that a certificate was successfully generated for the 'server\_cert.pem' and 'server\_key.pem' files, with a validity period of 365 days.

```
Command Prompt - python s x Command Prompt x + v
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>cd "C:\pemrograman jaringan"

C:\pemrograman jaringan>python secure_client.py
Menghubungkan ke Secure Server...
Terhubung dengan Cipher: ('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
Balasan Server: Pesan Anda aman bersama Kami.

C:\pemrograman jaringan>
```

Activate Windows  
Go to Settings to activate Windows.

```
Command Prompt - python s x Command Prompt x + v
Microsoft Windows [Version 10.0.26200.7623]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>cd "C:\pemrograman jaringan"

C:\pemrograman jaringan>python -m pip install cryptography
Requirement already satisfied: cryptography in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (46.0.3)
Requirement already satisfied: cffi>=2.0.0 in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from cryptography) (2.0.0)
Requirement already satisfied: pycparser in c:\users\asus\appdata\local\python\pythoncore-3.14-64\lib\site-packages (from cffi>=2.0.0->cryptography) (3.0)

C:\pemrograman jaringan>python generate_cert.py
C:\pemrograman jaringan\generate_cert.py:32: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
    datetime.datetime.utcnow()
C:\pemrograman jaringan\generate_cert.py:34: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
    datetime.datetime.utcnow() + datetime.timedelta(days=365)
Sertifikat berhasil dibuat: server_cert.pem & server_key.pem

C:\pemrograman jaringan>python secure_server.py
Secure Server listening on port 10023...
[NEW] Koneksi TCP dari ('127.0.0.1', 62882)
[SECURE] SSL Handshake sukses dengan ('127.0.0.1', 62882)
Pesan (Decrypted): Halo, ini pesan rahasia CIA.
```

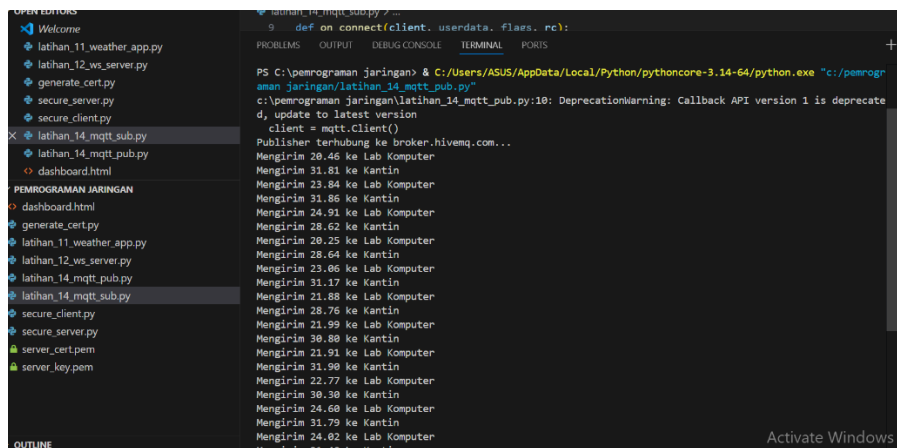
Activate Windows  
Go to Settings to activate Windows.



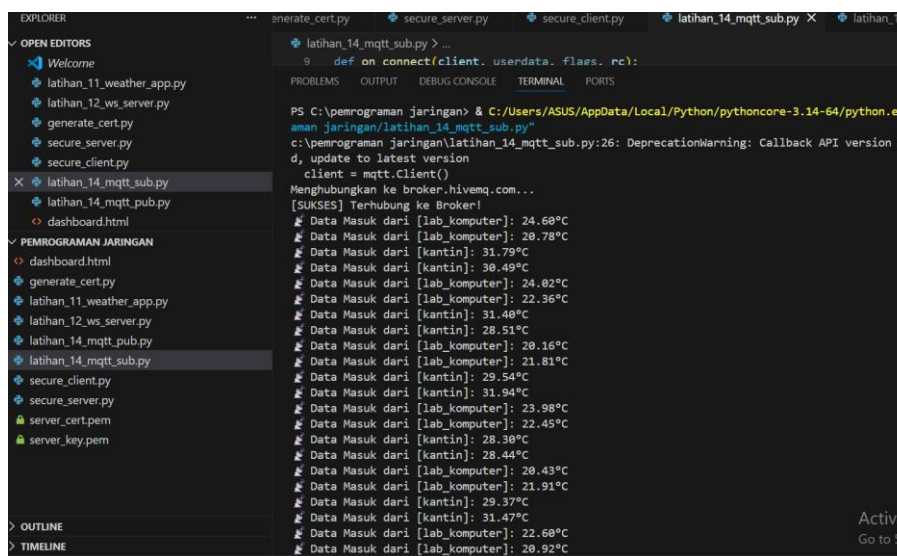
## BAB 14 ARSITEKRUT SISTEM TERDISTRIBUSI & IoT (MQTT)

Materi ini menampilkan penerapan komunikasi data menggunakan protokol MQTT dengan metode publish-subscribe berbasis Python. Program latihan\_14\_mqtt\_pub.py berperan sebagai publisher yang mengirimkan data suhu secara berkala ke broker publik HiveMQ, dengan topik berbeda untuk lokasi seperti lab\_komputer dan kantin. Sementara itu, program latihan\_14\_mqtt\_sub.py bertindak sebagai subscriber yang berhasil terhubung ke broker dan menerima data yang dipublikasikan sesuai topik yang di-subscribe, lalu menampilkannya di terminal. Alur ini menunjukkan bagaimana MQTT memungkinkan pertukaran data secara real-time, ringan, dan efisien, sehingga sangat cocok digunakan pada sistem IoT seperti pemantauan suhu di berbagai lokasi.

### HASIL:



```
PS C:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:\pemrograman jaringan\latihan_14_mqtt_pub.py"
c:\pemrograman jaringan\latihan_14_mqtt_pub.py:10: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
client = mqtt.Client()
Publisher terhubung ke broker.hivemq.com...
Mengirim 20.46 ke Lab Komputer
Mengirim 31.81 ke Kantin
Mengirim 23.84 ke Lab Komputer
Mengirim 31.86 ke Kantin
Mengirim 24.91 ke Lab Komputer
Mengirim 28.62 ke Kantin
Mengirim 20.25 ke Lab Komputer
Mengirim 28.64 ke Kantin
Mengirim 23.06 ke Lab Komputer
Mengirim 31.17 ke Kantin
Mengirim 21.88 ke Lab Komputer
Mengirim 28.76 ke Kantin
Mengirim 21.99 ke Lab Komputer
Mengirim 30.80 ke Kantin
Mengirim 21.91 ke Lab Komputer
Mengirim 31.90 ke Kantin
Mengirim 22.77 ke Lab Komputer
Mengirim 30.30 ke Kantin
Mengirim 24.60 ke Lab Komputer
Mengirim 31.79 ke Kantin
Mengirim 24.02 ke Lab Komputer
Mengirim 31.48 ke Kantin
```



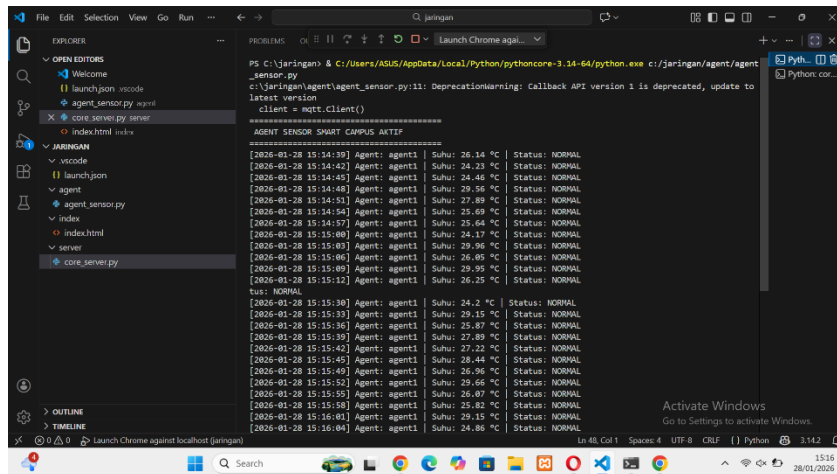
```
PS C:\pemrograman jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:\pemrograman jaringan\latihan_14_mqtt_sub.py"
c:\pemrograman jaringan\latihan_14_mqtt_sub.py:26: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
client = mqtt.Client()
Menghubungkan ke broker.hivemq.com...
[SUKSES] Terhubung ke Broker!
Data Masuk dari [lab komputer]: 24.60°C
Data Masuk dari [lab komputer]: 20.78°C
Data Masuk dari [kantint]: 31.79°C
Data Masuk dari [kantint]: 30.49°C
Data Masuk dari [lab komputer]: 24.02°C
Data Masuk dari [lab komputer]: 22.36°C
Data Masuk dari [kantint]: 31.40°C
Data Masuk dari [kantint]: 28.51°C
Data Masuk dari [lab komputer]: 20.16°C
Data Masuk dari [lab komputer]: 21.81°C
Data Masuk dari [kantint]: 29.54°C
Data Masuk dari [kantint]: 31.94°C
Data Masuk dari [lab komputer]: 23.98°C
Data Masuk dari [lab komputer]: 22.45°C
Data Masuk dari [kantint]: 28.30°C
Data Masuk dari [kantint]: 28.44°C
Data Masuk dari [lab komputer]: 20.43°C
Data Masuk dari [lab komputer]: 21.91°C
Data Masuk dari [kantint]: 29.37°C
Data Masuk dari [kantint]: 31.47°C
Data Masuk dari [lab komputer]: 22.60°C
Data Masuk dari [lab komputer]: 20.92°C
```



## BAB 15 PROYEK AKHIR (CAPSTONE PROJECT)

materi sistem yang ditampilkan merupakan implementasi arsitektur IoT berbasis MQTT yang terdiri dari agent/sensor (publisher) broker MQTT dan core server (subscriber + web server). Program `agent_sensor.py` berperan sebagai agent yang mensimulasikan pembacaan suhu, menentukan status (NORMAL/OK), lalu mengirimkan data secara periodik ke broker MQTT. Selanjutnya `core_server.py` berfungsi sebagai subscriber yang menerima data dari agent melalui broker, menambahkan waktu server, dan menampilkannya di terminal sekaligus menjalankan Flask web server agar data dapat diakses melalui browser.

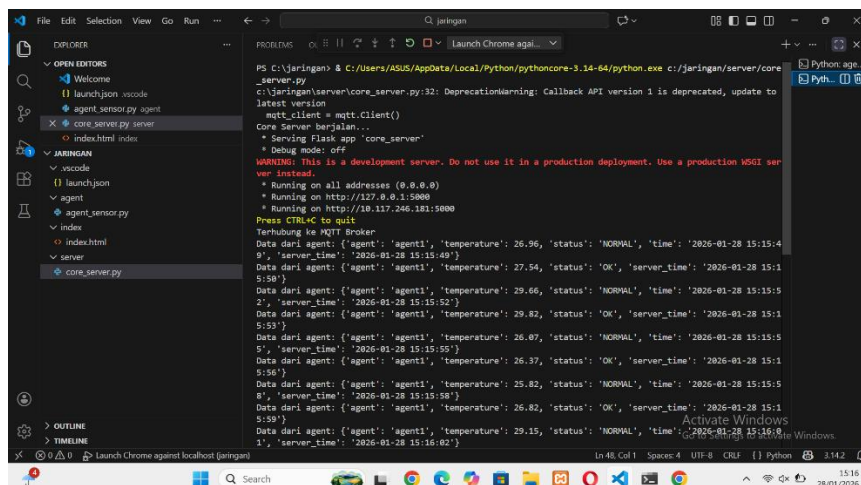
### HASIL:



```
PS C:\jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/jaringan/agent/agent_sensor.py
c:\jaringan\agent\agent_sensor.py:11: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
client = mqtt.Client()

AGENT SENSOR SMART CAMPUS AKTIF

[2026-01-28 15:14:39] Agent: agent1 | Suhu: 26.14 °C | Status: NORMAL
[2026-01-28 15:14:42] Agent: agent1 | Suhu: 24.23 °C | Status: NORMAL
[2026-01-28 15:14:45] Agent: agent1 | Suhu: 24.46 °C | Status: NORMAL
[2026-01-28 15:14:48] Agent: agent1 | Suhu: 29.56 °C | Status: NORMAL
[2026-01-28 15:14:51] Agent: agent1 | Suhu: 27.89 °C | Status: NORMAL
[2026-01-28 15:14:54] Agent: agent1 | Suhu: 25.69 °C | Status: NORMAL
[2026-01-28 15:14:57] Agent: agent1 | Suhu: 25.64 °C | Status: NORMAL
[2026-01-28 15:15:00] Agent: agent1 | Suhu: 24.17 °C | Status: NORMAL
[2026-01-28 15:15:03] Agent: agent1 | Suhu: 29.96 °C | Status: NORMAL
[2026-01-28 15:15:06] Agent: agent1 | Suhu: 26.09 °C | Status: NORMAL
[2026-01-28 15:15:09] Agent: agent1 | Suhu: 29.95 °C | Status: NORMAL
[2026-01-28 15:15:12] Agent: agent1 | Suhu: 26.25 °C | Status: NORMAL
[2026-01-28 15:15:15] Agent: agent1 | Suhu: 24.2 °C | Status: NORMAL
[2026-01-28 15:15:18] Agent: agent1 | Suhu: 29.15 °C | Status: NORMAL
[2026-01-28 15:15:21] Agent: agent1 | Suhu: 25.87 °C | Status: NORMAL
[2026-01-28 15:15:24] Agent: agent1 | Suhu: 27.89 °C | Status: NORMAL
[2026-01-28 15:15:27] Agent: agent1 | Suhu: 27.22 °C | Status: NORMAL
[2026-01-28 15:15:30] Agent: agent1 | Suhu: 28.44 °C | Status: NORMAL
[2026-01-28 15:15:33] Agent: agent1 | Suhu: 26.96 °C | Status: NORMAL
[2026-01-28 15:15:36] Agent: agent1 | Suhu: 29.66 °C | Status: NORMAL
[2026-01-28 15:15:39] Agent: agent1 | Suhu: 26.07 °C | Status: NORMAL
[2026-01-28 15:15:42] Agent: agent1 | Suhu: 25.82 °C | Status: NORMAL
[2026-01-28 15:15:45] Agent: agent1 | Suhu: 29.15 °C | Status: NORMAL
[2026-01-28 15:15:48] Agent: agent1 | Suhu: 24.86 °C | Status: NORMAL
```



```
PS C:\jaringan> & C:/Users/ASUS/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/jaringan/server/core_server.py
c:\jaringan\server\core_server.py:32: DeprecationWarning: Callback API version 1 is deprecated, update to latest version
mqtt_client = mqtt.Client()

Core Server berjalan...
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://10.137.246.181:5000
Press CTRL+C to quit

Terhubung ke MQTT Broker
Data dari agent: {'agent': 'agent1', 'temperature': 26.96, 'status': 'NORMAL', 'time': '2026-01-28 15:15:49'}
Data dari agent: {'agent': 'agent1', 'temperature': 27.54, 'status': 'OK', 'server_time': '2026-01-28 15:15:50'}
Data dari agent: {'agent': 'agent1', 'temperature': 29.66, 'status': 'NORMAL', 'time': '2026-01-28 15:15:52'}
Data dari agent: {'agent': 'agent1', 'temperature': 29.82, 'status': 'OK', 'server_time': '2026-01-28 15:15:53'}
Data dari agent: {'agent': 'agent1', 'temperature': 26.07, 'status': 'NORMAL', 'time': '2026-01-28 15:15:55'}
Data dari agent: {'agent': 'agent1', 'temperature': 26.37, 'status': 'OK', 'server_time': '2026-01-28 15:15:56'}
Data dari agent: {'agent': 'agent1', 'temperature': 25.82, 'status': 'NORMAL', 'time': '2026-01-28 15:15:58'}
Data dari agent: {'agent': 'agent1', 'temperature': 26.82, 'status': 'OK', 'server_time': '2026-01-28 15:15:59'}
Data dari agent: {'agent': 'agent1', 'temperature': 29.15, 'status': 'NORMAL', 'time': '2026-01-28 15:16:02'}
```

