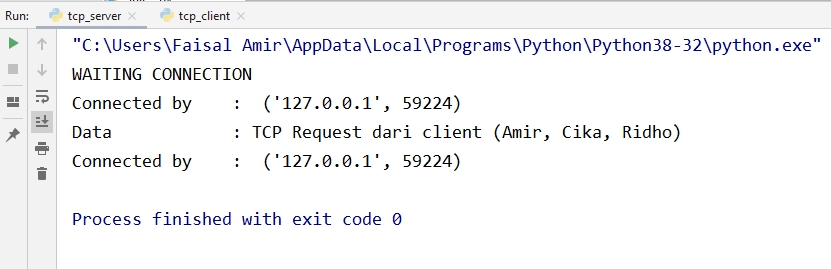
Analisis Config File

|  |  |
| --- | --- |
| CODE | FUNGSI (PENJELASAN) |
| BASE\_CONFIG\_IP\_ADDRESS = **'127.0.0.1'** BASE\_CONFIG\_PORT = 6080 BASE\_CONFIG\_BUFFER = 1110  BASE\_MESSAGE\_RESPONSE = **'RECEIVED'** BASE\_MESSAGE\_CONNECTING = **'WAITING CONNECTION'** BASE\_MESSAGE\_REQUEST = **'REQUEST'** MESSAGE\_REQUEST = **'Request dari client (Amir, Cika, Ridho)'** | Deklarasi konfigurasi ip address, port, buffer, dan message |

Analisis TCP Server

|  |  |
| --- | --- |
| CODE | FUNGSI (PENJELASAN) |
| **import** socket **from** com.frogobox.base.config **import** \* | Import function untuk memanggil fungsi socket |
| *# definisikan alamat IP binding yang akan digunakan* TCP\_SERVER\_IP = BASE\_CONFIG\_IP\_ADDRESS  *# definisikan port number binding yang akan digunakan* TCP\_SERVER\_PORT = BASE\_CONFIG\_PORT  *# definisikan ukuran buffer untuk mengirimkan pesan* TCP\_SERVER\_BUFFER = BASE\_CONFIG\_BUFFER | Deklarasi IP Address, Port, Buffer dan Address. |
| print(BASE\_MESSAGE\_CONNECTING) | Menampilkan proses menunggu koneksi |
| *# buat socket bertipe TCP* **with** socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) **as** tcpServer:  *# lakukan bind* tcpServer.bind((TCP\_SERVER\_IP, TCP\_SERVER\_PORT))   *# server akan listen menunggu hingga ada koneksi dari client* tcpServer.listen()   *# menerima koneksi* connection, address = tcpServer.accept()   **with** connection:   *# lakukan loop forever* **while True**:   *# menampilkan koneksi berupa IP dan port client yang terhubung menggunakan print* print(**'Connected by \t: '**, address)   *# menerima data berdasarkan ukuran buffer* messageFromClient = connection.recv(TCP\_SERVER\_BUFFER)   **if not** messageFromClient:  **break** *# menampilkan pesan yang diterima oleh server menggunakan print,* print(**'Data \t\t\t:'**, str(messageFromClient, **'utf-8'**))   *# mengirim kembali data yang diterima dari client kepada client* connection.send(messageFromClient) | Mengaktifkan socket yang telah di inisialisasi menjadi tcp server. Kemudian mengaitkan socket pada IP dan port yang sudah di deklarasikan pada file config.py  Kemudian code tcpServer.listen() menunggu sampai ada request dari client yang siap di jalankan.  Jika client merequest dengan benar makan koneksi di setujui.  Dengan koneksi yang di setujui tersebut maka akan melakukan perulangan secara terus menerus dengan menjalakan perintah menampilakn koneksi IP Adress dan pesan yang dikirim dari client.  Terakhir connection.send() berfungsi untuk mengirimkan kembali ke client |
| *# tutup koneksi* tcpServer.close() | Koneksi di tutup |

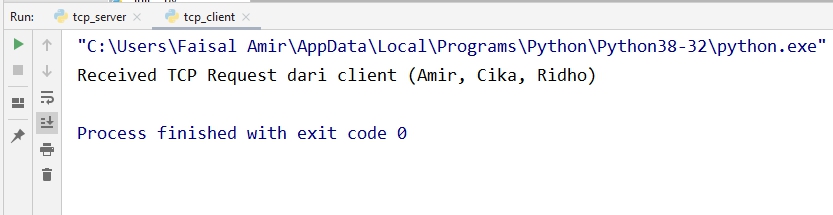
Screenshoot



Analisis TCP Client

|  |  |
| --- | --- |
| CODE | FUNGSI (PENJELASAN) |
| *# import library socket karena akan menggunakan IPC socket* **import** socket **from** com.frogobox.base.config **import** \* | Import function untuk memanggil fungsi socket |
| *# definisikan tujuan IP server* TCP\_SERVER\_IP = BASE\_CONFIG\_IP\_ADDRESS  *# definisikan port dari server yang akan terhubung* TCP\_SERVICE\_PORT = BASE\_CONFIG\_PORT  *# definisikan ukuran buffer untuk mengirimkan pesan* TCP\_SERVER\_BUFFER = BASE\_CONFIG\_BUFFER  *# definisikan pesan yang akan disampaikan* TCP\_MESSAGE = **'TCP '** + MESSAGE\_REQUEST | Deklarasi IP, Port, buffer serta message yang akan di gunakan untuk menghubungkan konesi ke server |
| *# buat socket TCP* tcpClient = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) | Declarasi tcp Client dengan menggunakan Socket |
| *# lakukan koneksi ke server dengan parameter IP dan Port yang telah didefinisikan* tcpClient.connect((TCP\_SERVER\_IP, TCP\_SERVICE\_PORT)) | Membuat koneksi TCP client |
| *# kirim pesan ke server, pesan bebas, dan ditambahkan nama anggota kelompok* tcpClient.send(TCP\_MESSAGE.encode()) | TCP client mengirimkan pesan dengan cara mengencode terlebih dahulu |
| *# terima pesan dari server* messageFromServer = tcpClient.recv(TCP\_SERVER\_BUFFER)  *# tampilkan pesan/reply dari server* print(**'Received'**, messageFromServer.decode()) | Mendapatkan pesan balikan dari server dan kemudian menampilkannya kembali |
| *# tutup koneksi* tcpClient.close() | Menutup koneksi tcp client |

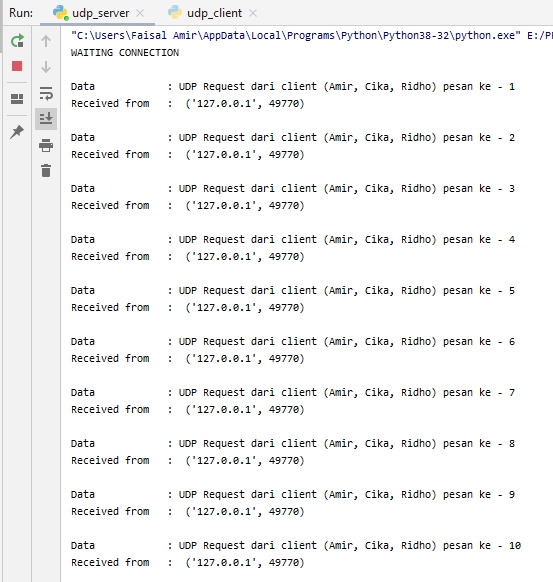
Screenshoot



Analisis UDP Server

|  |  |
| --- | --- |
| CODE | FUNGSI (PENJELASAN) |
| *# import library socket karena akan menggunakan IPC socket* **import** socket **from** com.frogobox.base.config **import** \* | Import function untuk memanggil fungsi socket |
| *# Berikan tampilan (print bahwa socket berhasil dibuat)* print(**"Socket Has Been Created"**) print(BASE\_MESSAGE\_CONNECTING) print() | Tampilan untuk bukti bahwa server telah di jalankan |
| *# definisikan alamat IP bind dari server* UDP\_SERVER\_IP = BASE\_CONFIG\_IP\_ADDRESS  *# definisikan port number untuk bind dari server* UDP\_SERVER\_PORT = BASE\_CONFIG\_PORT UDP\_SERVER\_BUFFER = BASE\_CONFIG\_BUFFER | Deklarasi IP, Port, dan Buffer untuk server |
| *# buat socket bertipe UDP* **with** socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) **as** udpServer:  *# lakukan bind* udpServer.bind((UDP\_SERVER\_IP, UDP\_SERVER\_PORT))   *# Berikan tampilan (print) bahwa socket bind ke alamat IP dengan port yang telah didapatkan   # loop forever* **while True**:  *# terima pesan dari client* messageFromClient, address = udpServer.recvfrom(UDP\_SERVER\_BUFFER)   **if not** messageFromClient:  **break** *# menampilkan hasil pesan dari client* print(**'Data \t\t\t:'**, messageFromClient.decode())   *# print (addr)* print(**'Received from \t: '**, address)  print() | Membuat socket bertipe UDP, kemudian socket tersebut di kaitkan pada ip dan port yang sudah di deklarasikan.  Setelah itu dilakukan perulangan terus menerus untuk mendapatkan mengecek apakah ada pesan dari klien atau tidak, jika ada maka di tampilkan. Jika tidak maka server tetap menyala terus menerus |
| udpServer.close() | Koneksi server udp di tutup |

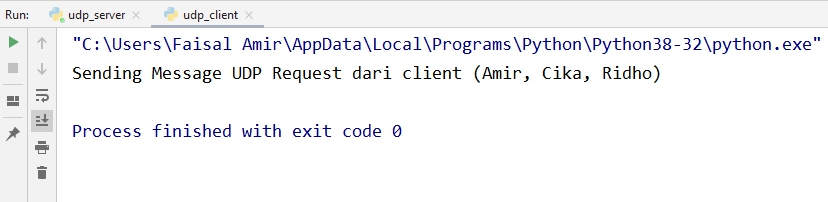
Screenshoot



Analisis UDP Client

|  |  |
| --- | --- |
| CODE | FUNGSI (PENJELASAN) |
| *# import library socket karena akan menggunakan IPC socket* **import** socket **from** com.frogobox.base.config **import** \* | Import function untuk memanggil fungsi socket |
| *# definisikan target IP server yang akan dituju* UDP\_SERVER\_IP = BASE\_CONFIG\_IP\_ADDRESS  *# definisikan target port number server yang akan dituju* UDP\_SERVER\_PORT = BASE\_CONFIG\_PORT UDP\_MESSAGE = **'UDP '** + MESSAGE\_REQUEST | Deklarasi IP, Port dan Message dari klien |
| print(**'Sending Message '** + UDP\_MESSAGE) | Menampilkan apa yang sedang dilakukan oleh client yaitu mengirim pesan |
| *# buat socket bertipe UDP* **with** socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) **as** udpClient:  udpClient.connect((UDP\_SERVER\_IP, UDP\_SERVER\_PORT))  **for** x **in** range(10):  udpClient.sendto(str(UDP\_MESSAGE + **' pesan ke - '** + str(x + 1)).encode(), (UDP\_SERVER\_IP, UDP\_SERVER\_PORT)) | Mendeklarasikan udp client dengan menggunakan socket yang dikaitkan pada ip dan port udp server yang sudah di buat.  Kemudian mengirimkan pesan sebanyak 10 kali dengan perulangan. Sesuai ip dan port server yang tepat |
| udpClient.close() | Menutup koneksi udp client |

Screenshoot



Telkom University / IFX - 43 – GAB

Friskadini Ismayanti (1301198496)

Muhammad Faisal Amir (1301198497)

Ridho Maulana Cahyudi (1301198515)