I. Pengenalan socket pada Python

Socket adalah titik akhir dari saluran komunikasi dua arah. Setiap socket dalam Python dapat berkomunikasi mengirimkan data baik dalam satu arus proses; dua atau lebih proses yang berlainan dalam mesin yang sama; atau antar proses pemrograman di mesin yang berseberangan.

Socket pun bisa diimplementasikan pada banyak tipe channel sekaligus, seperti Unix domain sockets, UDP, TCP, dan sejenisnya. Perpustakaan socket memuat banyak kategori khusus untuk menangani beragam tipe komunikasi, serta sistem antarmuka general bagi tugas-tugas lain.

Istilah Socket dan Artinya
Domain Grup protokol yang berperan dalam mekanisme pengiriman data. Nilai-nilai dalam kelompok ini berupa konstanta seperti AF_INET, PF_UNIX, PF_INET, PF_X25 dan lain-lain.
type Jenis komunikasi antara dua titik akhir *tunnel*, biasanya ada SOCK_STREAM yang digunakan untuk protokol berorientasi koneksi; dan SOCK_DGRAM untuk protokol tanpa koneksi.
protocol Biasanya berupa data nol. Protocol dapat digunakan untuk mengidentifikasi varian protokol dalam beragam domain dan jenis.
hostname Merupakan penanda antarmuka sebuah jaringan yang terdiri atas: · String: dapat berupa nama host, alamat quad bertitik, atau alamat IPV6 dalam notasi titik dua atau satu (*dot*). · String "<broadcast>": berfungsi mengidentifikasi alamat INADDR_BROADCAST. · Zero-length string: mengidentifikasi INADDR_ANY. · Integer: ditafsirkan sebagai alamat biner dalam urutan byte host.
port Setiap server dapat menangkap sinyal klien saat memanggil satu atau beberapa port sekaligus. Port dapat berupa nomor port Fixnum, string yang berisi nomor port, atau nama layanan.

II. Pembuatan socket pada Python

s = socket.socket (socket_family, socket_type, protocol=0)

Keterangan parameter :

socket_family: gunakan antara AF_UNIX atau AF_INET menunjukkan bahwa kita akan menggunakan protokol seperti IPv4.

socket_type: gunakan **SOCK_STREAM** atau **SOCK_DGRAM**. menunjukkan bahwa kita akan menggunakan protokol **TCP** atau **UDP**.

protocol: biasanya bagian ini dibiarkan kosong atau bernilai default

Metode Server Socket Programming

No	Metode Server Socket dan Definisinya
1	**s.bind()** Metode ini dipakai untuk mengikat alamat (nama host, pasangan nomor port) ke soket.
2	**s.listen()** Metode ini digunakan untuk menyiapkan dan memulai TCP listener.
3	**s.accept()** Metode ini secara pasif menerima koneksi dari klien TCP, lalu menunggu sampai koneksi tiba (memblokir).

Metode Programming Client Socket

No	Metode Client Socket dan Definisinya
1	**s.connect()** Metode ini memulai koneksi server TCP secara aktif.

Metode General Socket

No	Metode General Socket dan Definisinya
1	**s.recv()** Metode yang digunakan untuk menerima pesan TCP.
2	**s.send()** Metode yang digunakan untuk mentransmisikan pesan TCP.
3	**s.recvfrom()** Metode yang digunakan untuk menerima pesan UDP.
4	**s.sendto()** Metode yang digunakan untuk mentransmisikan pesan UDP.
5	**s.close()** Metode untuk menutup socket.
6	**socket.gethostname()** Metode untuk mengembalikan nama host.

III. Komunikasi antar socket

Sisi Server :

1. Gunakan fungsi socket yang tersedia di modul pembuatan objek socket untuk menulis server Internet sederhana (simple server). Objek socket digunakan untuk memanggil fungsi lain yang mengatur kinerja server.
2. Kemudian, jalankan fungsi bind(hostname, port) untuk menentukan port layanan yang tertera di host.
3. Lalu panggil metode accept dari objek yang dikembalikan. Metode ini akan menunggu hingga klien tersambung ke port yang telah ditentukan, lalu mengembalikan objek koneksi yang mewakili sambungan ke klien.

```
import socket

# membuat objek socket server
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# mengikat socket server ke alamat dan port tertentu
server_address = ('localhost', 8000)
server_socket.bind(server_address)

# mengaktifkan socket server agar dapat menerima koneksi
server_socket.listen(1)

# menerima koneksi dari socket client
print('Waiting for a connection...')
client_socket, client_address = server_socket.accept()
print(f'Connected by {client_address}')

# menerima data dari socket client
data = client_socket.recv(1024)
print(f'Received: {data.decode()}')

# mengirim respons ke socket client
message = 'Hello, client!'
client_socket.sendall(message.encode())

# menutup koneksi
client_socket.close()
server_socket.close()
```

Sisi Client :

1. Setelah sukses dengan pemrograman server, lalu lanjutkan dengan menulis pemrograman klien (client) yang sederhana memakai modul socket Python. Kode berikut ini ditujukan untuk membuka koneksi ke port 8000 dan host yang telah ditentukan.
2. Kode socket.connect(hosname, port) membuka koneksi TCP ke hostname pada port. Setelah soket terbuka, Anda dapat membacanya seperti objek IO lainnya. Jika sudah, ingatlah untuk menutupnya sebagaimana Anda akan menutup file biasa.

```
import socket

# membuat objek socket client
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# menghubungkan socket client ke server
server_address = ('localhost', 8000)
client_socket.connect(server_address)

# mengirim data ke server
```

```
message = 'Hello, server!'
client_socket.sendall(message.encode())

# menerima respons dari server
data = client_socket.recv(1024)
print(f'Received: {data.decode()}')

# menutup koneksi
client_socket.close()
```

IV. Socket Option di Python

Socket option adalah parameter yang dapat diatur untuk mengkonfigurasi perilaku socket. Dalam Python, kita dapat mengatur socket option menggunakan metode setsockopt() dari objek socket.

Berikut adalah beberapa socket option yang dapat diatur pada socket Python:

- SO_REUSEADDR: opsi ini digunakan untuk mengaktifkan penggunaan kembali alamat yang sudah digunakan. Dengan mengatur opsi ini ke nilai 1, socket dapat digunakan kembali meskipun alamat tersebut masih digunakan oleh socket yang belum selesai.
- SO_KEEPALIVE: opsi ini digunakan untuk mengaktifkan mekanisme keepalive pada koneksi TCP. Jika koneksi tidak aktif selama waktu yang ditentukan, maka mekanisme ini akan mengirimkan pesan kecil ke tujuan untuk mempertahankan koneksi.
- TCP_NODELAY: opsi ini digunakan untuk menonaktifkan buffering pada data yang dikirimkan melalui koneksi TCP. Dengan mengatur opsi ini ke nilai 1, data akan langsung dikirimkan setiap kali write() dipanggil, tanpa menunggu buffering.
- SO_LINGER: opsi ini digunakan untuk menentukan waktu penundaan saat socket ditutup. Dengan mengatur opsi ini ke nilai (1, t), socket akan ditutup setelah menunggu t detik, bahkan jika masih ada data yang belum terkirim.

```python
import socket

# membuat objek socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# mengatur opsi socket
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)
s.setsockopt(socket.SOL_SOCKET, socket.SO_LINGER, (1, 5))
```

V. Tugas dan Latihan

Analisa Hasil Percobaan tadi yang sudah dibuat Komunikasi antar socket dan Socket Option

1. Bagaimana cara membuat objek socket pada Python?
2. Apa yang dimaksud dengan alamat pada socket?
3. Bagaimana cara mengikat objek socket ke alamat tertentu pada Python?
4. Bagaimana cara menerima koneksi dari klien pada server menggunakan Python?
5. Bagaimana cara mengirim dan menerima pesan menggunakan socket pada Python?
6. Apa yang dimaksud dengan socket option pada Python, dan berikan contohnya?
7. Bagaimana cara mengirim pesan pada socket menggunakan protokol TCP pada Python?
8. Bagaimana cara menerima pesan pada socket menggunakan protokol TCP pada Python?
9. Bagaimana cara mengirim pesan pada socket menggunakan protokol UDP pada Python?
10. Bagaimana cara menerima pesan pada socket menggunakan protokol UDP pada Python?