Este programa recebe mensagens de entrada e as ecoa de volta ao remetente.

- server.py

Realiza a importação da biblioteca socket

```
import socket
```

Logo após, utiliza duas variáveis auxiliares para determinar o host e a porta do endereço do servidor a ser utilizado.

```
HOST = 'localhost'
PORT = 50000
```

Agora é criado a TCP / IP socket. Na função socket() são passados dois parâmetros, o primeiro é a família de protocolo, que no caso é a família IPv4 e o segundo parâmetro é o tipo de protocolo, que no caso é do tipo TCP.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Então a função bind() é utilizada para associar o socket com o endereço do servidor. Nesse caso, o endereço é localhost, referindo-se ao servidor atual, e o número da porta é 50000.

```
s.bind((HOST, PORT))
```

Em seguida é chamada a função listen() que coloca o soquete do servidor em modo de escuta e espera por uma conexão de entrada.

```
s.listen()
```

Enquanto aguarda a resposta, é exibida a mensagem abaixo:

```
print('Aguardando conexão de um cliente')
```

A função accept() retorna uma conexão aberta entre o servidor e o cliente, junto com o endereço do cliente. A conexão é na verdade um soquete diferente em outra porta (atribuída pelo kernel).

```
connection, address = s.accept()
```

Após a conexão ser bem sucedida é exibida na tela uma mensagem com o endereço do cliente.

```
print('Conectado em', address)
```

Os dados são lidos da conexão com a função recv().

```
while True:
  data = connection.recv(1024)
```

Quando a comunicação com um cliente é concluída, a conexão precisa ser finalizada usando close().

```
  if not data:
      print('Fechando a conexão')
      connection.close()
      break
```

E também é necessário transmitir esses dados através da função sendall()

```
  connection.sendall(data)
```

Dessa forma, podemos finalizar o socket.

```
s.close()
```

- client.py

Realiza a importação da biblioteca socket

```
import socket
```

O programa do cliente configura o soquete de forma diferente ao do servidor. Em vez de se conectar a uma porta e ouvir, ele conecta o soquete diretamente ao endereço remoto. Assim, logo abaixo é utilizado novamente duas variáveis auxiliares para determinar o host e a porta do endereço do servidor a conectado.

```
HOST = '127.0.0.1'
PORT = 50000
```

Assim como no servidor é criado a TCP / IP socket, com os mesmos parâmetros, o primeiro é a família de protocolo, que no caso é a família IPv4 e o segundo parâmetro é o tipo de protocolo, que no caso é do tipo TCP.

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Como citado anteriormente, o soquete é diretamente conectado ao endereço remoto, isso ocorre através da função connect().

```
s.connect((HOST, PORT))
```

Após a conexão ter sido estabelecida, os dados podem ser enviados pelo soquete com a função sendall(), a qual envia uma string com a mensagem *“Olá mundo”*.

```
s.sendall(str.encode('Olá Mundo'))
```

E esses mesmos dados podem ser recebidos pela função recv(), assim como no servidor.

```
data = s.recv(1024)
```

Quando toda a mensagem é enviada e uma cópia é recebida, é exibida uma outra mensagem com os dados decodificados (data.decode()).

```
print('Conexão cliente servidor realizada com sucesso! \nMessage:',
data.decode())
```

Assim, o soquete é finalizado.

```
s.close()
```