**socket**

Существуют клиентские и серверные сокеты. Серверный сокет прослушивает определенный порт, а клиентский подключается к серверу. После того, как было установлено соединение начинается обмен данными. Например, сервер посылает приветствие и предложение ввести какую-либо команду. Клиент в свою очередь вводит команду, сервер ее анализирует, выполняет необходимые операции и отдает клиенту результат.

|  |  |
| --- | --- |
| **Сервер** | **Клиент** |
| import socket  with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server:      # слушаем 5757 порт со всех хостов      port = 5757      server.bind(('', port))      # кол-во подклчений в очереди 10      server.listen(10)      print(f'Server start ...')      # принятие подключения      # con - новый сокет, addr - адрес клиента      # этот сокет и будет использоваться для приема и посылке клиенту данных      con, addr = server.accept()        with con:          print(f'Connected by {addr}')          while True:              # чтение данных от клиента              data = con.recv(1024).decode()                con.sendall(f'You are send: {data}'.encode()) | import socket  client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  client.connect(('localhost', 5757))  while True:      out\_data = input().lower()      client.sendall(f'{out\_data}'.encode())      in\_data = client.recv(1024)      print(in\_data.decode())        if out\_data == 'exit':          break  client.close() |

На сервере блокирующие вызовы **server.accept(), con.recv()** !

Сервер обслуживает только одного клиента, а затем закрывается !

**Asyncio + Socket**

|  |  |
| --- | --- |
| Сервер | Клиент |
| import asyncio, socket  async def handle\_client(client):      loop = asyncio.get\_event\_loop()      request = None      while request != 'quit':          request = (await loop.sock\_recv(client, 255)).decode('utf8')          response = f'You are send: {request}' + '\n'          await loop.sock\_sendall(client, response.encode('utf8'))      client.close()  async def run\_server():      server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)      server.bind(('localhost', 5757))      server.listen(5)      server.setblocking(False)      loop = asyncio.get\_event\_loop()      while True:          client, \_ = await loop.sock\_accept(server)          loop.create\_task(handle\_client(client))  asyncio.run(run\_server()) | import socket  client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  client.connect(('localhost', 5757))  while True:      out\_data = input().lower()      client.send(f'{out\_data}'.encode())      in\_data = client.recv(1024)      print(in\_data.decode())        if out\_data == 'exit':          break  client.close() |