***Python中socket模块常用函数和socket对象方法：***

sk = socket.socket([family[, type[, proto]]]) 创建一个套接字，famlily时地址族，type是套接字类型，可以设置为SOCK\_STREAM和SOCK\_DGRAM，proto一般不用填，默认为0。

**sk.bind(address)** 将套接字绑定到地址，address的格式取决于地址族，在AF\_INET模式下，以tuple(host, port)的形式表示地址。

**sk.listen(backlog)** 开始监听传入。backlog表示在拒绝接听之前，可以挂起的最大连接数量。比如backlog=5，表示内核已经接到了连接请求，但服务器还没有调用accept()进行处理的连接个数最大是5。这个值不能无限大，因为要在内核中维护连接队列。

sk.setblocking(bool) 是否阻塞（默认True），如果设置为False，那么accept和recv时一旦无数据，则报错。

**sk.accept()** 接受连接并返回(conn, address)，其中conn是新的套接字对象，可以用来接收和发送数据，address是连接客户端的地址。接收TCP客户的连接（阻塞式）等待连接的到来。

**sk.connect(address)** 连接到address处的套接字。一般address的格式是tuple(hostname, port)，如果连接出错，返回socket.error错误。

sk.connect\_ex(address) 同上，但是此函数有返回值，连接成功返回0，失败时返回编码，例如：10061。

**sk.recv(bufsize[, flag])** 接收套接字的数据。数据以字符串形式返回，bufsize指定最多可以接收的数量。flag提供相关消息的其他信息，通常可以忽略。

sk.recvfrom(bufsize[, flag]) 与recv()类似，但返回值是(data, address)。其中data是包含接收数据的字符串，address是发送数据的套接字地址。

**sk.send(string[, flag])** 将string中的数据发送到连接的套接字。返回值是发送的字节数量，该数量可能小于string的字节大小，即可能未将内容全部发送。

sk.sendall(string[, flag]) 将string中的数据发送到套接字。但是在返回之前会尝试发出所有的数据，成功返回None，失败抛出异常。内容通过递归调用send，将所有内容全部发送出去。

sk.sendto(string[, flag], address) 将数据发送到套接字，address形式为tuple(ipaddr, port)，指定远程地址。返回值是发送的字节数，该函数主要用于UDP协议。

sk.settimeout(timeout) 设置套接字操作的超时时间，timeout是一个浮点数，单位是秒。值为None表示没有超时时间。通常，超时时间应该在刚创建套接字时设置，因为它们可能用于连接的操作（如client连接最多等待5s）。

sk.getpeername() 返回套接字的远程地址。返回值是tuple(ipaddr, port)。

sk.getsockname() 返回自己套接字的地址。返回值是tuple(ipaddr, port)。

sk.fileno() 套接字的文件描述符

sk.close() 关闭套接字。

**简单通信实例：**

**服务端程序：**

serversocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

serversocket.bind((‘172.18.71.158’, 9999))

serversocket.listen(5)

while True:

clientsocket, addr = serversocket.accept()

print(“链接地址：%s”, % str(addr))

clientsocket.send(“欢迎”.encode(“utf-8”))

clientsocket.close()

**客户端程序：**

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect((‘172.18.71.158’, 9999))

msg = s.recv(1024)

s.close()

print(msg.decode(“utf-8”))

缺陷：单线程程序无法同时连接多个客户端。

解决方案：在服务端导入socketserver模块，使得每建立一个连接，就自动创建一个新的线程。或者参照以下代码写以支持多线程：

serversocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

serversocket.bind((‘172.18.71.158’, 9999))

serversocket.listen(5)

def tcplink(clientsocket, addr):

print(‘Accept new connect from %s: %s…’ % addr)

clientsocket.send(b‘Welcome’)

while True:

data = clientsocket.recv(1024)

time.sleep(1)

if not data or data.decode(‘utf-8’) == ‘exit’:

break

clientsocket.send(('Hello, % s' % data.decode('utf-8')).encode('utf-8'))

clientsocket.close()

print(“Connection from %s: %s closed” % addr)

while True:

clientsocket, addr = serversocket.accept()

t = threading.Thread(target=tcplink, args=(clientsocket, addr))

t.start()