1. TCP协议是什么？

TCP协议，传输控制协议，是一种面向连接的、可靠的、基于字节流的传输层通信协议。

2. TCP特点是什么？

面向连接：TCP通信需要经过创建连接、数据传送、终止连接三个步骤。

可靠传输：1）TCP采用发送应答机制。2）超时重传。3）错误校验。4) 流量控制和阻塞管理

流式报文：使用的是字节流，udp使用的是数据报

传输层协议

3. 在TCP客户端使用connect的作用是什么？

连接服务器，socket.connect(address) 参数address是元组格式(ip,port)

4. 在TCP服务器中listen的作用是什么？

使用socket创建的套接字默认的属性是主动的，使用listen可以将其变为被动，这样完成之后，下一步

就可以接受connect请求了。

listen中的参数表示挂起的最大连接数量。

5. 简述tcp客户端创建的流程。

1.创建tcp的socket

2.使用connect连接到服务器端

3.发送消息(send)/接受消息(recv)

4.关闭socket

6. recv函数的参数表示什么含义？

recv()函数用来接收TCP连接的数据，参数表示本次接收的最大长度。

7. 简述TCP服务器创建的流程、

1.创建tcp的socket套接字。

2.绑定ip和port。

3.listen事套接字变为被动套接字。

4.accept取出一个客户端连接用于服务。

5.recv/send接受发送消息。

6.关闭服务于客户端的socket。

7.关闭tcp的socket套接字。

8. accept的返回值分别代表的什么？

client\_socket, client\_addr = tcp\_socket.accept() 从等待的服务区，取出一个客户端连接

，为他服务,返回值时一个元组。

返回值：为客户端服务的套接字-->客户端套接字，客户端地址

9. 完成TCP客户端的编写。

服务器的ip和端口号需手动输入

要发送的信息需要手动输入

接收服务器返回的数据，并打印。

import socket

# 创建socket套接字

tcp\_client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# 获取用户输入服务器的ip和端口

server\_ip = input("请输入服务器的ip:")

server\_port = int(input("请输入服务器的port:"))

# 连接服务器

tcp\_client\_socket.connect((server\_ip, server\_port))

# 提示用户输入要发送的消息

send\_data = input("请输入要发送的消息:")

# 发送消息

tcp\_client\_socket.send(send\_data.encode())

# 接收服务器发送来的消息

recvdata = tcp\_client\_socket.recv(1024)

print("接收到的数据为:%s" % recvdata.decode())

# 关闭套接字

tcp\_client\_socket.close()

10. 完成TCP服务的的编写。

import socket

# 创建socket

tcp\_server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# 绑定固定端口

tcp\_server\_socket.bind(('', 10011))

# 将套接字变成被动套接字

tcp\_server\_socket.listen(128)

5

while True:

# 取出连接

client\_socket, client\_addr = tcp\_server\_socket.accept()

print("客户端:%s上线了" % (str(client\_addr)))

while True:

# 接收发来的消息

recv\_data = client\_socket.recv(4096)

if recv\_data:

print("接收到来自:%s的消息:%s" % (str(client\_addr), recv\_data.decode()))

# 发送一条消息给客户端

client\_socket.send("收到!".encode())

else:

print("客户端%s下线了" % str(client\_addr))

break

client\_socket.close()

# 关闭套接字

tcp\_server\_socket.close()