TCP编程

TCP服务器编程

TCP服务器编程的流程

1. 使用socket类创建一个套接字对象
2. 使用build((ip,port))方法绑定IP地址和端口号
3. 使用listen()方法开始TCP监听
4. 使用accept()方法等待客户端的连接
5. 使用recv()/send()方法接受/发送数据
6. 使用close()关闭套接字

套接字

所谓套接字 (Socket)，就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象。 一个套接字就是网络上进程通信的一端，提供了应用层进程利用网络协议交换数据的机制。 从所处的地位来讲，套接字上联应用进程，下联网络协议栈，是应用程序通过网络协议进行通信的接口，是应用程序与网络协议栈进行交互的接口 。

TCP编写服务端代码实现

```
from socket import socket,AF_INET,SOCK_STREAM
# AF_INEI 用于Internet之间的进程通信
# SOCK_STREAM 表示的是TCP协议编程


# 创建一个socket对象
server_socket=socket(AF_INET,SOCK_STREAM)
# 绑定IP地址和端口
ip="127.0.0.1" # local
port=1234 # 只需要数字在端口号的范围之中即可
server_socket.bind((ip,port))
# 使用listen()开始监听
server_socket.listen(5)
print("服务器已经启动")
# 等待客户端的连接
client_socket,client_addr=server_socket.accept() # 系列解包赋值
# 接受来自客户端的数据
data=client_socket.recv(1024)
print("客户端发送来的数据",data.decode("utf-8"))
# 关闭socket
server_socket.close()
```

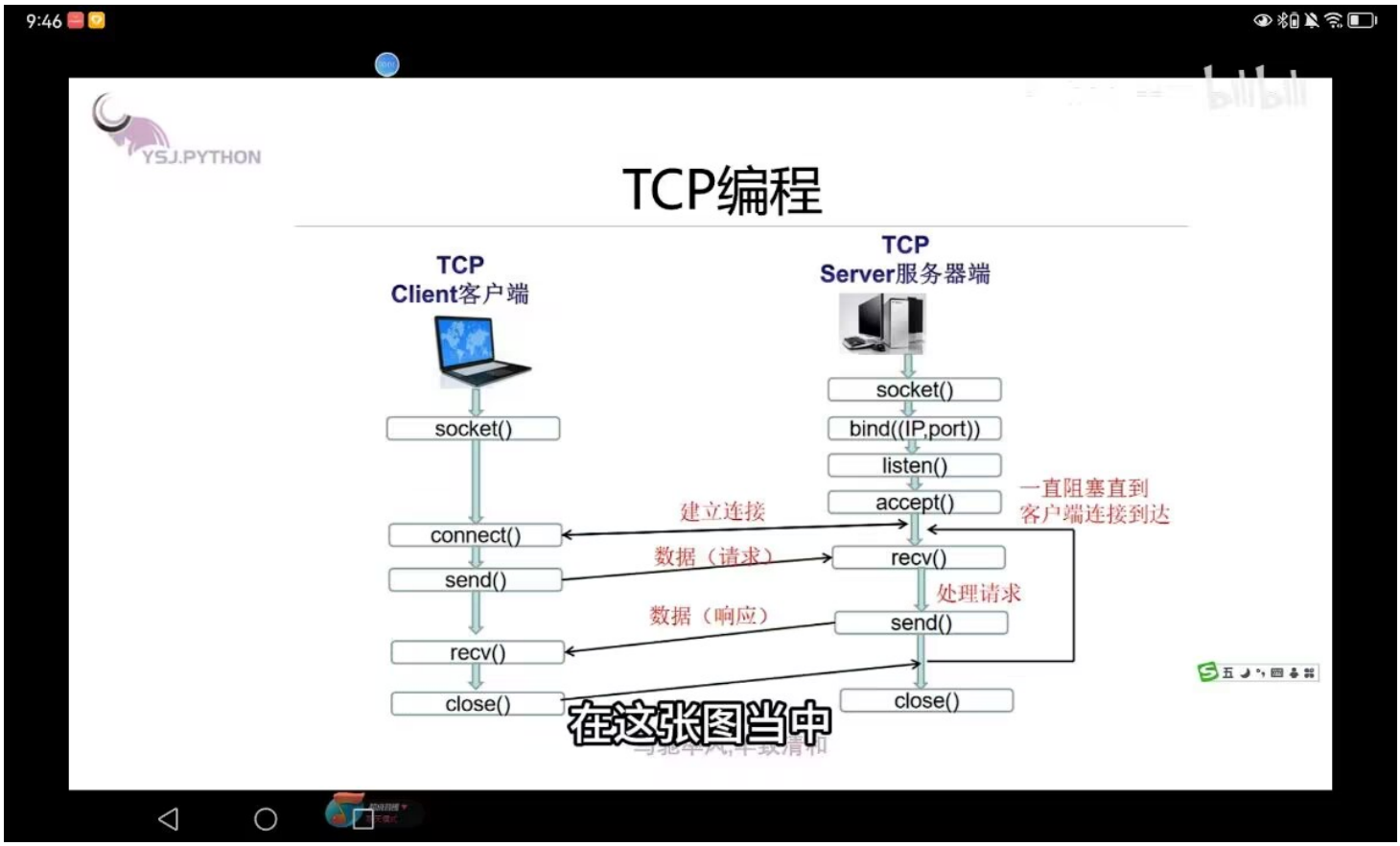
TCP客户端编程

TCP客户端编程的流程

1. 使用socket类创建一个套接字对象
2. 使用connect((host,port))设置连接主机的IP和主机设置的端口号
3. 使用recv()/send()方法接受/发送数据
4. 使用close()关闭套接字

TCP编程的图解

如图:



TCP客户端与服务端之间的多次通信

1. 多次通信的图解(注意一个项目中只可以写一个服务端或者客户端)

多次通信服务端和客户端实现的代码

服务端

```
from socket import socket ,AF_INET,SOCK_STREAM
# 创建一个socket_server对象
socket_sever= socket (AF_INET,SOCK_STREAM)
# 绑定IP地址和端口号
ip='127.0.0.1'
port=9989
socket_sever.bind((ip,port))
# 开始TCP监听
socket_sever.listen(5)
print("服务器开始接受")
# 开始接受数据
client_socket,client_addr=socket_sever.accept()
# 处理接受的数据
# data=client_socket.recv(1024)
# print('接受到的数据为',data.decode('utf-8')) # 最后将解码得到的数据打印出来
info=client_socket.recv(1024).decode('utf-8')
# print("客户端发送的数据为",data.decode("utf-8"))

while info!='bye' :
    if info!='' :
        print('客户端发送的数据是',info)

    # 准备发送的数据
    data=input("请输入要发送的数据:")

    # 服务端回复客户端
    client_socket.send(data.encode("utf-8"))
    if data=='bye':
        break
    info=client_socket.recv(1024).decode("utf-8") # while循环的步骤

# 关闭socket对象
socket_sever.close()
```

客户端代码

```
import socket
# 创建一个客户端对象
client_socket=socket.socket()
# IP地址和主机端口写入并连接
ip='127.0.0.1'
port=9989
client_socket.connect((ip,port))
print('-------与服务器连接成功---------')
# 使用recv方法或者send方法来发送数据
# 接受服务器发送的数据
info=''
while info!='bye':
    send_data=input('请输入你要发送的数据')
    client_socket.send(send_data.encode("utf-8"))
    # 条件判断
    if send_data=="bye":
        break
    # 接受服务器端的响应
    info=client_socket.recv(1024).decode("utf-8")
    print('服务器端发送的数据为',info)

# 关闭客户端的套接字
client_socket.close()
```