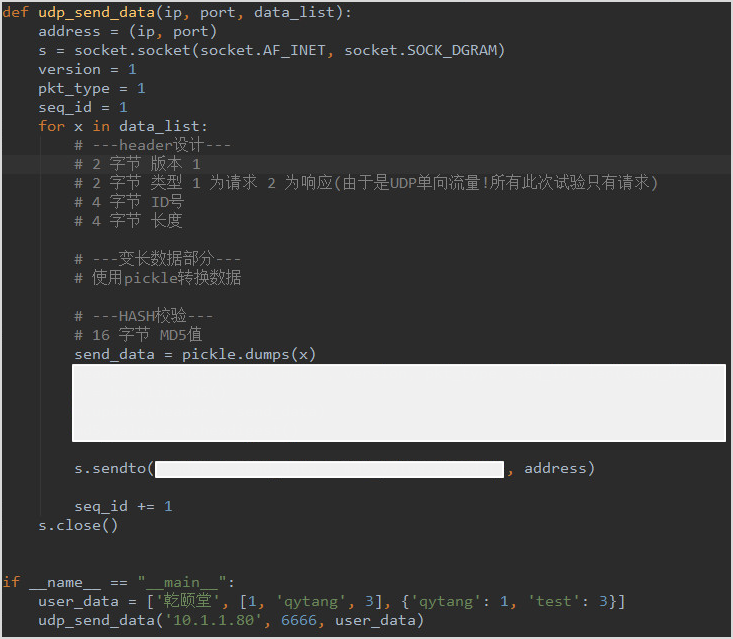
**经典网络协议 第三天作业**

1. **设计一个自己的UDP协议!用于传输各种Python数据**

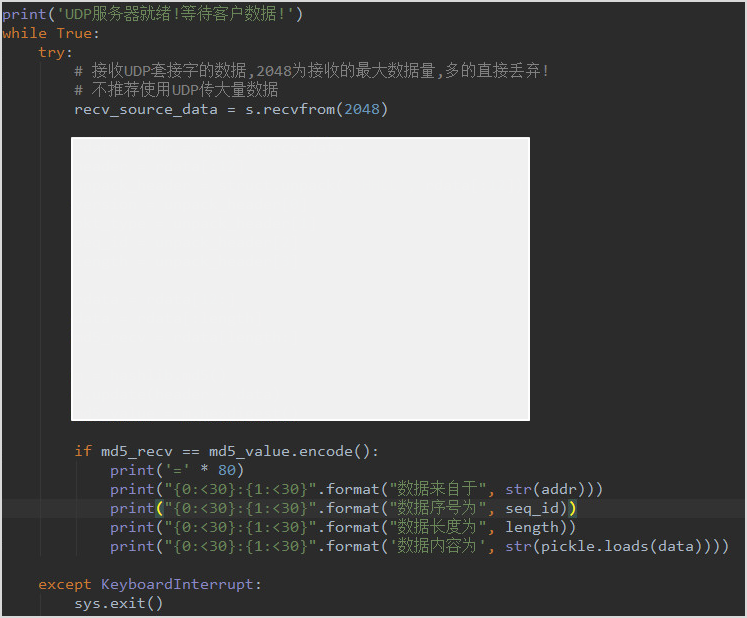
**协议字段设计**

# ---header设计---  
# 2 字节 版本 1  
# 2 字节 类型 1 为请求 2 为响应(由于是UDP单向流量!所有此次试验只有请求)  
# 4 字节 ID号  
# 4 字节 长度  
  
# ---变长数据部分---  
# 使用pickle转换数据  
  
# ---HASH校验---  
# 16 字节 MD5值

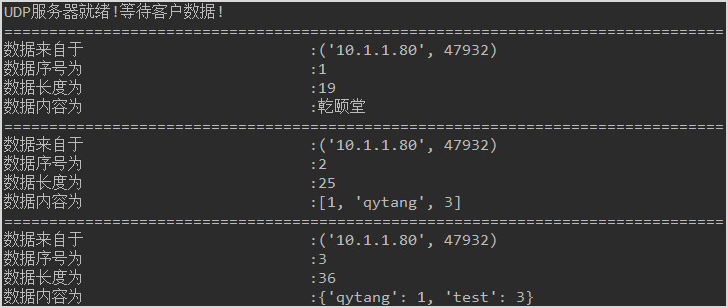
**UDP客户端代码:**



**UDP服务器端代码:**



**UDP服务器端显示结果:**



代码：

客户端：

import pickle  
import sys  
import socket  
import pickle  
import hashlib  
import struct  
print('UDP 服务器就绪！等待客户端数据！')  
address = ('0.0.0.0', 6666)  
s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
s.bind(address)  
while True:  
  
 try:  
 recv\_source\_data = s.recvfrom(2048)  
 data\_values,addr = recv\_source\_data  
 print(data\_values)  
 head\_values = struct.unpack('>hhii',data\_values[:12])  
 seq\_id = head\_values[2]  
 length = head\_values[3]  
 data = data\_values[12:(12+length)]  
 md5\_recv= data\_values[(12+length):]  
 m = hashlib.md5()  
 m.update(data)  
 md5\_value = m.hexdigest()  
  
  
 if md5\_recv == md5\_value.encode():  
 print('='\*80)  
 print('{0:<30}:{1:<30}'.format('数据来源于',str(addr)))  
 print('{0:<30}:{1:<30}'.format('数据来源于',seq\_id))  
 print('{0:<30}:{1:<30}'.format('数据来源于',length))  
 print('{0:<30}:{1:<30}'.format('数据来源于',str(pickle.loads(data))))  
  
 except KeyboardInterrupt:  
 sys.exit()  
s.close()

服务端

import socket  
import pickle  
import struct  
import hashlib  
def udp\_send\_data(ip,port,data\_list):  
 address = (ip,port)  
 s = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)  
 version = 1  
 pkt\_type = 1  
 seq\_id = 1  
  
 for x in data\_list:  
 print(x)  
 send\_data = pickle.dumps(x)  
 m = hashlib.md5()  
 m.update(send\_data)  
 md5\_value = m.hexdigest()  
 pkg\_head = struct.pack('>hhii',version,pkt\_type,seq\_id,len(send\_data))  
 pkg= pkg\_head + send\_data+md5\_value.encode()  
 s.sendto(pkg,address)  
  
 seq\_id +=1  
  
 s.close()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 user\_data = ['乾颐堂',[1,'qytang',3],{'qytang':1,'test':3}]  
 udp\_send\_data('192.168.77.4',6666,user\_data)

运行结果：

