TCP 에코 서버 프로그램 (TCP_echoserver_demo.py)

```
1  # TCP_echoserver_demo.py
2
3  from socket import *
4
5  port = 2500
6  BUFSIZE = 1024  # 한번에 수신할 최대 바이트 수
7
8  sock = socket(AF_INET, SOCK_STREAM)
9  sock.bind(('', port)) # 소켓과 주소 결합
10 sock.listen(1) #최대 대기 클라이언트 수
11 print("Waiting for clients...")
12
13 #클라이언트의 연결 요청을 받는다
14 c_sock, (r_host, r_port) = sock.accept()
15 print('connected by', r_host, r_port)
16
17 while True:
18     #상대방 메시지 수신
19     data = c_sock.recv(BUFSIZE) #accept()로 반환받은 소켓 사용
20     if not data: #연결이 종료되었으면
21         break
22     print("Received message: ", data.decode())
23
24     c_sock.send(data)  #수신 메시지를 다시 전송
25
26 c_sock.close()
```

TCP 에코 클라이언트 프로그램(TCP_client_demo.py)

```python
# TCP_client_demo.py
import socket

port = 2500
address = ("localhost", port)
BUFSIZE = 1024

# 클라이언트 소켓 생성
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 서버와 연결
s.connect(address)
print("Connected")

while True:
    msg = input("Message to send: ")
    # 입력 메시지가 없으면 종료한다
    if not msg: # 프로그램을 종료하려면 입력없이 Enter
        break

    s.send(msg.encode()) #send a message to server
    r_msg = s.recv(BUFSIZE) #receive message from server

    #바이트를 문자열로 변환하여 출력
    print("Received message: %s" %r_msg.decode())

s.close()
```

예외 처리 에코 서버(TCP_echoserver.py)

```python
#TCP_echoserver.py
# 송수신 예외 처리를 한 에코 서버 프로그램
from socket import *

port = 2500
BUFSIZE = 1024

sock = socket(AF_INET, SOCK_STREAM)
sock.bind(('', port))
sock.listen(5) #최대 대기 클라이언트 수
print("Waiting for clients...")

c_sock, (r_host, r_port) = sock.accept()
print('connected by', r_host, r_port)

while True:
    # 수신 예외 처리
    try:
        data = c_sock.recv(BUFSIZE)
        if not data: #연결 해제됨
            c_sock.close()
            print('연결이 정상적으로 종료되었습니다')
            break
    except:
        print("연결이 강제로 종료되었습니다")
        c_sock.close() #소켓을 닫는다
        break #무한 루프 종료
    else:
        print(data.decode())

    # 송신 예외 처리
    try:
        c_sock.send(data)
    except: #연결 종료로 인한 예외 발생
        print("연결이 종료되었습니다")
        c_sock.close() #소켓을 닫는다
        break #무한 루프 종료
```

예외 처리 클라이언트(TCP_client.py)

```python
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#서버 주소 입력
svrIP = input(("Server IP(default: 127.0.0.1): "))
if svrIP == '':
    svrIP = '127.0.0.1' #기본 주소

#포트 번호 입력
port = input('port(default: 2500): ')
if port == '':
    port = 2500 #기본 포트
else:
    port = int(port)

sock.connect((svrIP, port))
print('Connected to ' + svrIP)

while True:
    msg = input("Sending message: ")
    if msg == 'q':  #'q'를 입력하면 프로그램 종료
        break

    if not msg:     #송신 데이터가 없으면 다시 진행
        continue

    try:            # 데이터 송신 예외 처리
        sock.send(msg.encode()) #메시지 전송
    except: #연결이 종료됨
        print("연결이 종료되었습니다")
        break

    try:            # 데이터 수신 예외 처리
        msg = sock.recv(1024)
        if not msg: # 연결이 정상 종료됨
            print("연결이 종료되었습니다")
            break
        print(f'Received message: {msg.decode()}')   # 데이터 수신
    except: #연결이 강제 종료됨
        print("연결이 종료되었습니다")
        break

sock.close()
```

TCP 프로세싱 서버 프로그램(TCP_processs_server.py)

```python
#TCP_processs_server.py
#숫자를 받아 영어 단어를 송신한다
import socket

#숫자에 대한 영어 사전
table = {'1':'one', '2': 'two', '3': 'three', '4': 'four',\
'5':'five', '6': 'six', '7': 'seven', '8': 'eight',\
'9': 'nine', '10': 'ten'}

# 서버 소켓을 생성하고 연결을 기다린다
s=socket.socket() #AF_INET, SOCK_STREAM
address = ("", 2500)
s.bind(address)
s.listen(1)
print('Waiting...')

c_socket, c_addr = s.accept()   #클라이언트 연결 수락
print("Connection from ", c_addr)

#메시지 처리
while True:
    data = c_socket.recv(1024).decode() #요청 수신
    try:
        resp = table[data] #데이터를 key로 사용하여 value를 가져온다
    except:
        c_socket.send('Try again'.encode()) #오류가 있을 때
    else:
        c_socket.send(resp.encode()) #변환 값을 전송
```

TCP 서버 프로그램(MyTCPServer.py)

```python
class TCPServer:
    # TCP 소켓을 생성하고 연결 대기
    def __init__(self, port):
        import socket
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.sock.bind(('', port)) #소켓과 주소의 결합
        self.sock.listen(1) #연결 대기
        self.connected = False

    # 소켓을 닫는다
    def disconnect(self):
        self.connected = False
        self.c_sock.close()

    #클라이언트 연결 요청 수락
    def accept(self): #연결 수락
        self.c_sock, self.c_addr = self.sock.accept()
        self.connected = True
        return self.c_sock, self.c_addr # 소켓과 주소 반환

    # 메시지 전송
    def send(self, msg):
        if self.connected:
            self.message = msg
            self.c_sock.send(self.message.encode())
            return True # 전송 성공 반환
        else:
            return False # 전송 실패 반환

    # 메시지 수신
    def receive(self, r_sock=None):
        if not r_sock:
            r_sock = self.c_sock # 연결 소켓 사용
        try:
            data = r_sock.recv(1024)
            if not data:
                self.disconect()
                return None
            return data.decode() # 수신 데이터 반환
        except:
            self.disconnect()
            return None
```

TCP 사용자정의 프로그램(MyTCPServer_ex01.py)

```python
#MyTCPServer_ex01.py
#사용자 정의 모듈을 이용한 에코 서버 프로그램

import MyTCPServer as ms#사용자 정의 모듈을 불러온다

server = ms.TCPServer(2500)
print("Waiting for connection")

while True:
    # 연결 요청 or 데이터 수신
    # 미연결 상태이면 연결 수락
    if not server.connected:
        server.accept()

    # 연결 상태이면 메시지를 수신하여 출력하고 다시 전송한다
    else:
        msg = server.receive() # 문자열 반환하므로 decode할 필요없음
        # 연결이 되어 있지 않으면 None이 반환됨
        if msg:
            print('수신메시지: ', msg) # 문자열을 바이트로 변환
            server.send(msg)
        else:
            print("Disconnected")
            break
server.disconnect()
```

파일 전송 프로그램(TCP_sendfile.py)

```python
import socket

port = 2500
s_sock = socket.socket()
host = ""
s_sock.bind((host, port))
s_sock.listen(1)

print('Waiting for connection....')

c_sock, addr = s_sock.accept() #클라이언트와 연결
print('Connected from', addr)

# 상대방의 준비완료를 기다린다
msg = c_sock.recv(1024) #클라이언트로부터 준비 완료 수신
print(msg.decode())

#경로를 포함한 파일 이름을 입력
filename = input('File name to send(c:/test/sample.bin): ')
filename = filename.strip('"')
print(f"Sending '{filename}'")
c_sock.sendall(filename.encode()) #파일 이름 전송

#파일을 읽기 모드로 열고 sendfile() 함수로 파일 내용 전송
with open(filename, 'rb') as f:
    c_sock.sendfile(f,0) #파일 내용 전송

print('Sending complete')
c_sock.close()
```

파일 수신 프로그램(TCP_receivefile.py)

```python
import socket, os

s_sock = socket.socket()
host = "localhost"
port = 2500

s_sock.connect((host, port)) #서버와 연결

s_sock.send("I am ready".encode()) #준비 완료 메시지

# 파일 이름 수신
fn = s_sock.recv(1024).decode() # 경로를 포함한 파일이름 수신
filename = os.path.basename(fn) # 기본 파일이름 추출

# 파일 생성, 수신 내용 저장. 관리자모드에서 Permission denied 발생
w_file = 'c:/Users/leejo/'+ filename
with open(w_file, 'wb') as f: #저장 파일 열기
    print('file opened')
    print('receiving file...')
    while True:
        data = s_sock.recv(8192) #파일 내용 수신
        if not data: #내용이 없으면 종료
            break
        f.write(data)#내용을 파일에 쓰기

print(f'Download complete in {w_file}')
s_sock.close()
print('Connection closed')
```

동영상 송신 프로그램(video_server.py)

```python
import socket, cv2, pickle, struct, imutils

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
TCP_PORT = 9000
server_addr = ('', TCP_PORT)

server_socket.bind(server_addr)  # 주소와 포트번호 바인드

server_socket.listen(5)   # 접속 대기
print("접속 대기:", server_addr)

# 클라이언트 연결
while True:
    client_socket, addr = server_socket.accept()
    print(addr, '와 연결됨')

    #클라이언트와 연결되면 웹카메라에서 동영상 프레임 획득
    if client_socket:
        vid = cv2.VideoCapture(0) # 동영상 객체 생성. WebCam=0

        while (vid.isOpened()):
            # 프레임 획득
            img, frame = vid.read() # retvalue, image_frame
            # 프레임 크기 조절
            frame = imutils.resize(frame, width=640)
            # 프레임을 바이트 스트림으로 변환
            frame_bytes = pickle.dumps(frame)
            #프레임 길이와 프레임 데이터를 결합하여 전송 프레임 구성
            # frame 길이(unsigned 8bytes) + frame
            message = struct.pack("Q", len(frame_bytes)) + frame_bytes
            # [길이 + 프레임] 전송
            client_socket.sendall(message)
            cv2.imshow('Server Video', frame) # 전송 영상 표시

            # 전송 화면을 닫으면 종료
            key = cv2.waitKey(1) & 0xFF
            if key == ord('q'):
                client_socket.close()
```

동영상 수신 프로그램(video_client.py)

```python
import socket, cv2, pickle, struct

client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
HOST_IP = 'localhost' # 서버 IP 주소
TCP_PORT = 9000

# 서버와 연결
client_socket.connect((HOST_IP,TCP_PORT))
#빈 프레임 준비
data = b""
#프레임 길이(8바이트) 저장 변수 준비
payload_size = struct.calcsize("Q") # 길이정보를 unsigned 8bytes로 표시

while True:
    # 전체 수신 프레임의 길이가 길이 영역(8바이트)보다 커질 때까지
    # 데이터를 계속 읽는다
    while len(data) < payload_size:
        #최대 4K 데이터 수신
        packet = client_socket.recv(4*1024) # 4K
        if not packet: break # 연결 종료?
        data += packet

    # 프레임 길이 추출
    packed_msg_size = data[:payload_size]
    # 프레임 추출
    data = data[payload_size:]
    # 프레임 길이를 8 bytes 정수로 변환
    msg_size = struct.unpack("Q",packed_msg_size)[0]

    # 길이 만큼의 frame을 계속 수신한다
    while len(data) < msg_size:
        data += client_socket.recv(4*1024)

    # 한 프레임 크기를 잘라낸다
    frame_data = data[:msg_size]
    # 나머지는 다음 프레임(next frame)
    data  = data[msg_size:]

    # 동영상 프레임 표시
    frame = pickle.loads(frame_data) # 바이트 스트림을 프레임으로 변환
    cv2.imshow("Client Video",frame)
    key = cv2.waitKey(1) & 0xFF
    if key  == ord('q'):
        break
client_socket.close()
```