

2022년 IoT기반 스마트 솔루션 개발자 양성과정



# Programming : Python

## 15-TCP/IP

담당 교수 : 윤 종 이

010-9577-1696

[ojo1696@naver.com](mailto:ojo1696@naver.com)

<https://cafe.naver.com/yoons2022>



충북대학교 공동훈련센터

# OSI [Open Systems Interconnection]

- ISO에서 기본 참조 모델 제정
  - 개방형 시스템간 상호 접속 가능
  - 표준화된 네트워크 구조 제공
  - 이기종간 상호 접속을 위한 가이드라인
- 세부기능
  - 시스템 간의 통신을 위한 표준 제공과 통신을 방해하는 기술적인 문제들을 제거
  - 단일 시스템 간의 정보 교환을 하기 위한 상호 접속점을 정의
  - 제품들 간의 번거로운 변환 없이 통신할 수 있는 능력을 향상
  - OSI 참조 모델 표준이 모든 요구를 만족시키지 못할 경우, 다른 방법을 사용하는 것에 대한 충분한 이유를 제공



# OSI 7 Layer vs TCP/IP Layer

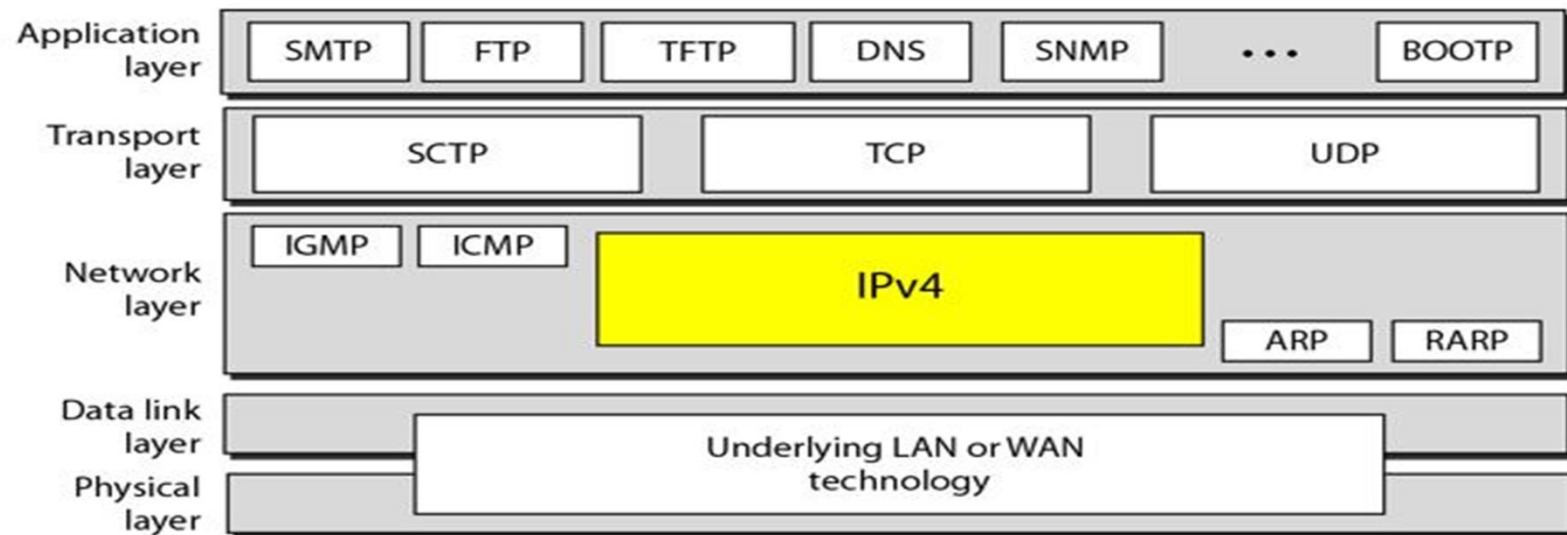
응용 계층	DHCP, FTP, DNS, HTTP, POP, SMTP ....	응용계층
표현 계층		
세션 계층		
전송 계층	TCP UDP Segment	전송계층
네트워크 계층	IP Address : IPv4 IPv6 Datagram	인터넷 계층
데이터링크 계층	MAC Address Frame	네트워크 접근 계층
물리 계층	Ethernet cable, wire...	

OSI 표준 모델

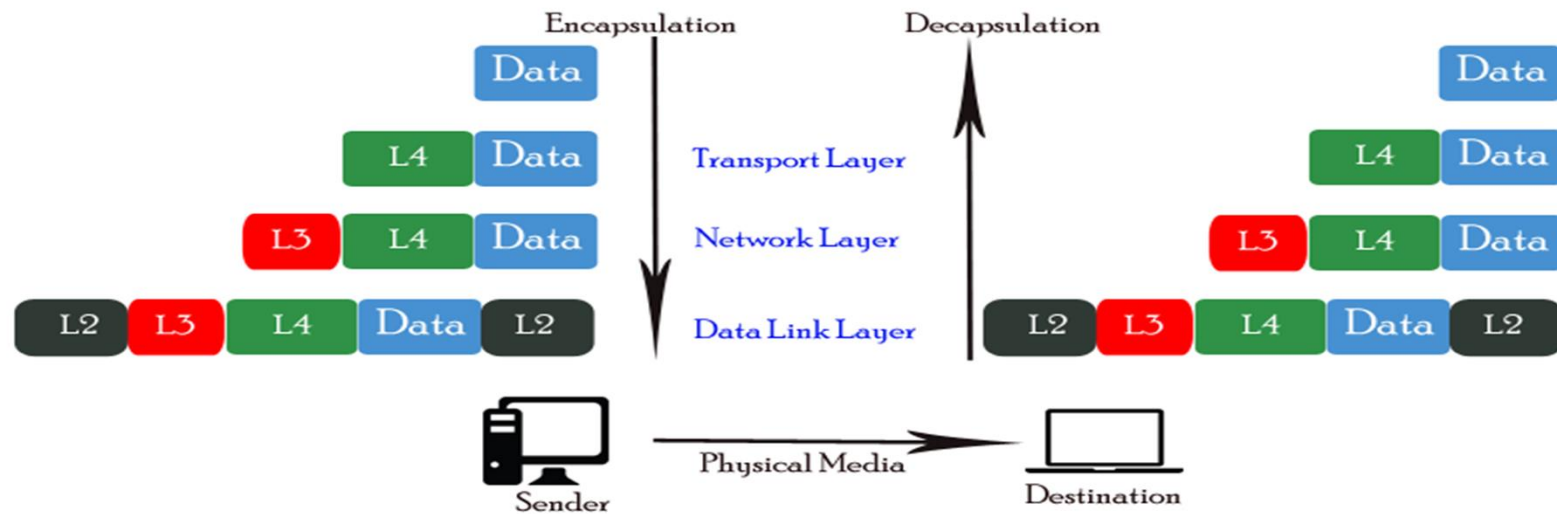
TCP/IP 모델



# TCP/IP Protocol Suite



# TCP/IP Protocol Stack



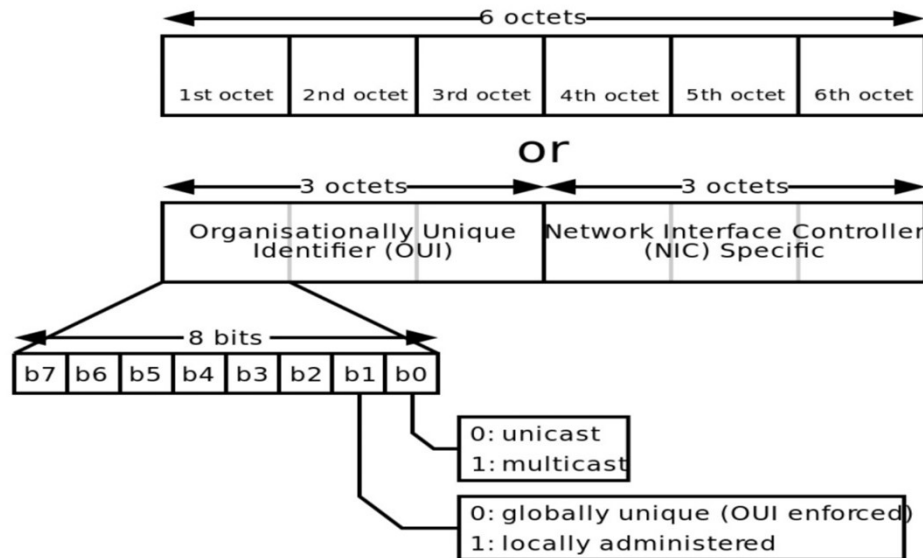
# TCP/IP Layer

Application Layer	응용 프로그램	HTTP, FTP, SSH, SMTP, POP
Transport Layer	데이터 전송	TCP/UDP
Internet Layer	데이터 경로, 주소	IP
Network Access Layer	물리적 연결, 신호변환	MAC



# MAC Address

- Media Access Control Address
- 네트워크 인터페이스 컨트롤러(NIC)의 제조업체가 할당
- 16진수 6자리



# IP Address

- Internet Protocol address : 인터넷규약주소
- 컴퓨터 네트워크에서 장치들이 서로를 인식하고 통신을 하기 위해서 사용하는 특수한 번호
- 이 번호를 이용하여 발신자를 대신하여 메시지가 전송되고 수신자를 향하여 예정된 목적지로 전달
- IP 할당
  - Static IP
  - Dynamic IP : DHCP로부터 할당
- IPv4
  - 192.168.0.21
- IPv6

## IPv6 Address Format

**X:X:X:X:X:X:X:X**  
where X = 0000 ... FFFF (hex)

- 2001:0DB8:0000:0000:0008:8000:0000:417A
- 2001:DB8:0:0:8:8000:0:417A
- 2001:DB8::8:8000:0:417A
- 2001:DB8:0:0:8:8000::417A
- 2001:db8::8:8000:417A





# Raspberry Pi3 MAC/IP Address

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.27 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 fe80::b39c:fbf3:7992:f924 prefixlen 64 scopeid 0x20<link>  
    ether b8:27:eb:af:b2:9e txqueuelen 1000 (Ethernet)  
    RX packets 162414 bytes 65029474 (62.0 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 787910 bytes 649059413 (618.9 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 4731 bytes 443687 (433.2 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 4731 bytes 443687 (433.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@raspberrypi:~ $
```



# TCP & UDP

- 전송 제어 프로토콜 (TCP: Transfer Control Protocol)
  - 기능이 많음
  - 연결성 지원
  - 신뢰할 수 있는 프로토콜
- 사용자 데이터그램 프로토콜 (UDP: User Datagram Protocol)
  - TCP와 유사한 주소지정 방법
  - IP와 연결 해주는 래퍼(Wrapper) 프로토콜
  - 연결을 수립하지 않음
  - 신뢰할 수 없음
  - 데이터가 손실될 수 있음



# TCP/UDP Application

- TCP 애플리케이션
  - TCP가 제공하는 신뢰성과 연결성 서비스에 의한 부하가 존재
  - 파일의 일부만 손실 되더라도 의미없는 데이터가 되는 경우
  - HTTP, FTP, SMTP
- UDP 애플리케이션
  - 애플리케이션 자체에서 UDP의 부족한 기능 보충가능
  - 적당한 수준의 신뢰성과 연결성
  - 데이터의 일부가 손실되는 것이 중요치 않은 프로토콜
  - 비디오, 멀티스트리밍



# TCP/UDP

특성/설명	UDP	TCP
일반 설명	단순하면 빠름, 애플리케이션과, 네트워크 계층의 인터페이스 역할이 중심	애플리케이션이 네트워크 계층 문제(한계)를 신경 쓰지 않고 안정적인 데이터 교환기능 제공
프로토콜 연결 수립	비연결형	연결형
애플리케이션의 데이터 입력 인터페이스	메시지 기반, 별도의 패키지로 송신	스트림기반, 특정한 구조 없이 데이터를 송신
신뢰성과 승인	신뢰성과 승인 기능이 없는 최선노력 전송 방식	데이터 전송을 신뢰할 수 있도록 모든 메시지에 승인이 있음
재전송	수행하지 않음 (상위계층에서 제어)	모든 데이터 전송을 관리, 손실된 데이터는 자동으로 재전송
데이터 흐름 관리 기능	없음	슬라이딩 윈도우를 이용한 흐름제어, 윈도우 크기를 적절히 조정, 혼잡 회피 알고리즘 사용
부하	매우 낮음	UDP보다 큼
전송 속도	매우 빠름	UDP보다 느림
적합한 데이터 양	소형에서 중형(최대 수백 옥텟)	소형에서 초대형 까지 가능 (기가바이트 단위)
프로토콜을 사용하는 애플리케이션의 유형	데이터의 완전성보다 전달 속도 중시, 소량의 데이터를 송신하거나, 멀티/브로드캐스트 중심의 애플리케이션	데이터의 손실이 없어야 하는 애플리케이션 과 프로토콜에 적합
유명 애플리케이션과 프로토콜	DNS, RIP 등	FTP, SMTP, DNS, HTTP, BGP등



# Port

- 전송계층에서 어떤 프로세스로 보낼지 식별
  - 수신계층에서 네트워크 계층에서 수신한 데이터를 어떤 프로세스로 보낼지 식별
  - IANA(Internet Assigned Numbers Authority) 에서 관리
  - 16비트 필드로 0~65535까지 값을 가짐
- 
- well-known port : 0 ~ 1023
  - registered port : 1024 ~ 49151
  - dynamic port : 49152 ~ 65535



# well-known port

Port Number	Transport Protocol	Service Name	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS)	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217



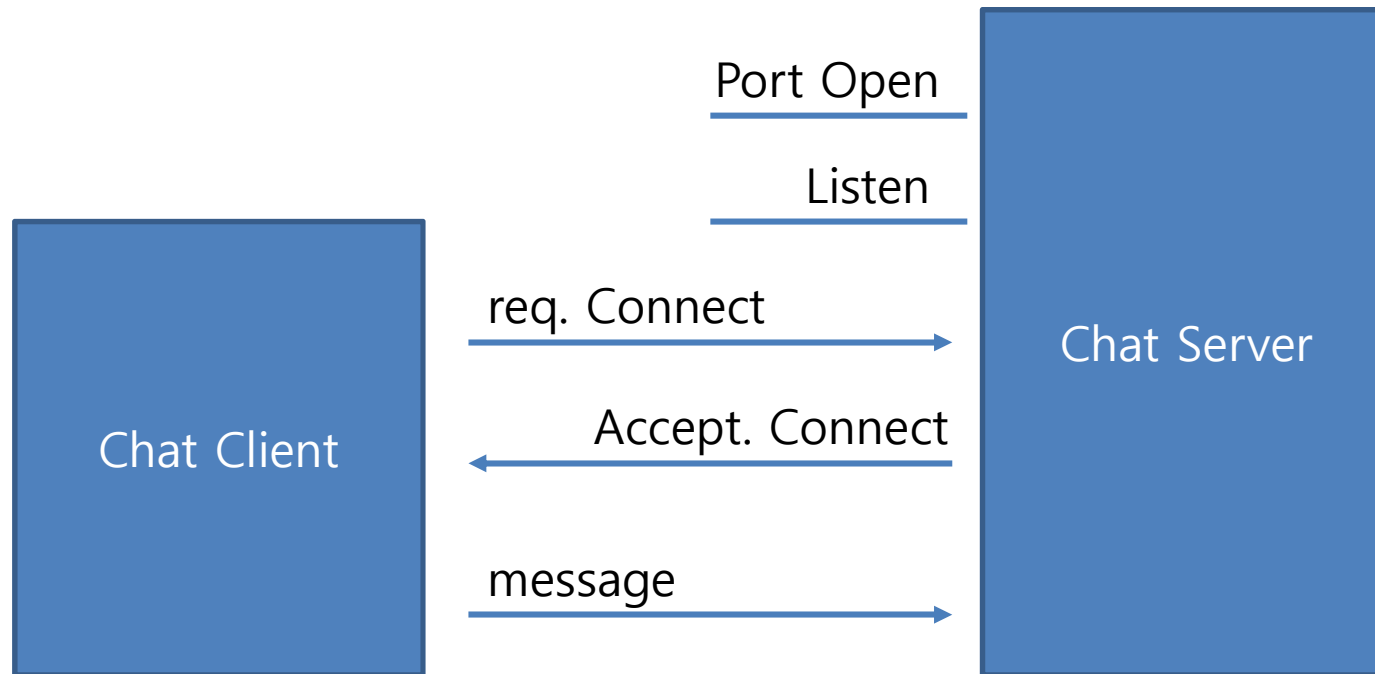
충북대학교 공동훈련센터

# socket

- TCP : socket.SOCK\_STREAM
- UDP: socket.SOCK\_DGRAM
- server
  - socket( )
  - bind( )
  - accept( )
- client
  - connect( )
- TX : send( )
- RX : recv( )



# TCP Server/Client





# Ex1 : TCP Server

## Thonny Editor

- New
- Save As 'Ex\_TCP\_Server.py'
- run

Local host : 127.0.0.1

```
TCP_Server.py ✕
1  from socket import *
2  host="127.0.0.1"
3  port=12345
4
5  s=socket(AF_INET,SOCK_STREAM)
6  s.bind((host,port))
7
8  s.listen(1)
9  print ("Listening for Connection : ", host)
10
11 while True:
12     conn,addr=s.accept()
13     print ("Connection form ", addr)
14     conn.send(b'Connection OK')
15
16 conn.close()
```

```
Shell
Python 3.9.2 (/usr/bin/python3)
>>> %Run TCP_Server.py
Listening for Connection : 127.0.0.1
```

*class socket.socket(family=AF\_INET, type=SOCK\_STREAM, proto=0, fileno=None)*



충북대학교 공동훈련센터

# Ex1 : TCP Client

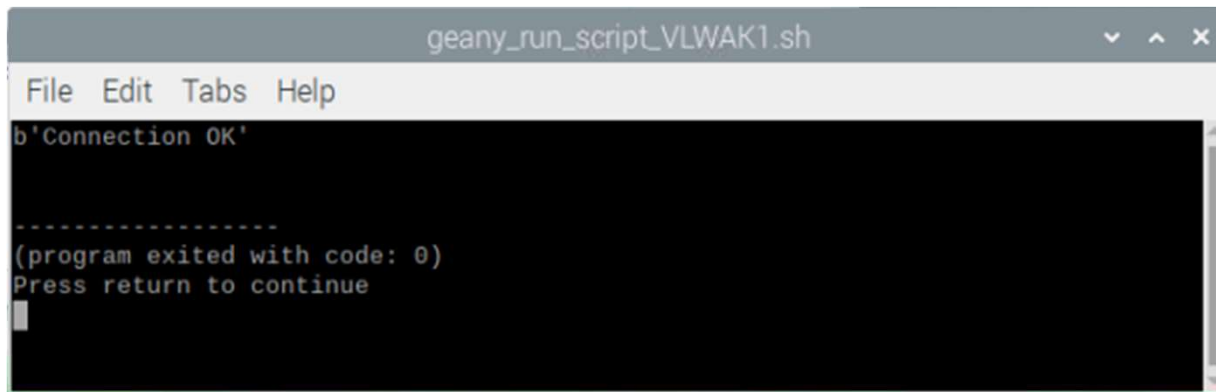
## Geany Editor

- New
- Save As 'Ex\_TCP\_Client.py'
- run

```
Ex_TCP_Client.py x
1  from socket import *
2  host="127.0.0.1"
3  port=12345
4
5  s=socket(AF_INET,SOCK_STREAM)
6  s.connect((host,port))
7  print (s.recv(1024))
8  s.close()
9
```

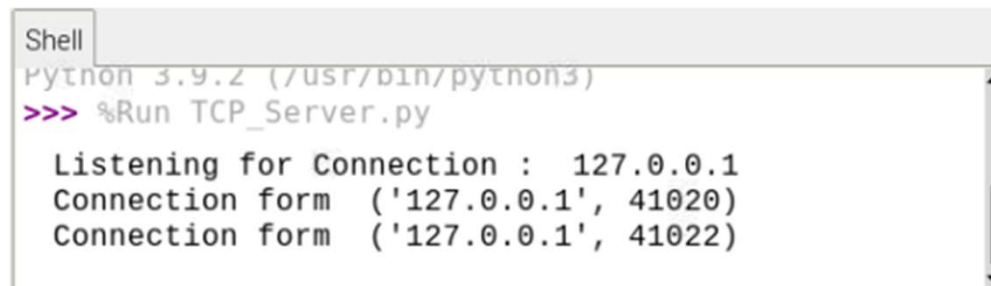


# Ex1 : Run



```
geany_run_script_VLWAK1.sh
File Edit Tabs Help
b'Connection OK'

-----
(program exited with code: 0)
Press return to continue
```



```
Shell
python 3.9.2 (/usr/bin/python3)
>>> %Run TCP_Server.py

Listening for Connection : 127.0.0.1
Connection form ('127.0.0.1', 41020)
Connection form ('127.0.0.1', 41022)
```



## Ex2 : TCP ChatServer

```
TCP_ChatServer.py ✕
1  from socket import *
2  host="127.0.0.1"
3  port=12345
4
5  s=socket(AF_INET,SOCK_STREAM)
6  s.bind((host,port))
7
8  s.listen(1)
9  print ("Listening for Connection : ", host)
10 conn=None
11
12 while True:
13     if conn is None:
14         print ('Waiting : ')
15         conn,addr=s.accept()
16         print ("Connection form ", addr)
17     else:
18         print (conn.recv(1024))
19         conn.send(b'OK')
20
21 conn.close()
22
```

자기 IP

```
Shell
Python 3.9.2 (/usr/bin/python3)
>>> %Run TCP_ChatServer.py
Listening for Connection : 127.0.0.1
Waiting :
```



## Ex2 : TCP ChatClient

```
TCP_ChatClient.py ✕
1  from socket import *
2  host="127.0.0.1" #socket.gethostname()
3  port=12345
4
5  s=socket(AF_INET, SOCK_STREAM)
6  s.connect((host, port))
7  print ('Connect to ', host)
8
9  while True:
10     message=input("Message to Server : ")
11     s.send(message.encode('utf-8'))
12     print (s.recv(1024))
13
14  s.close()
15
```

접속할 IP



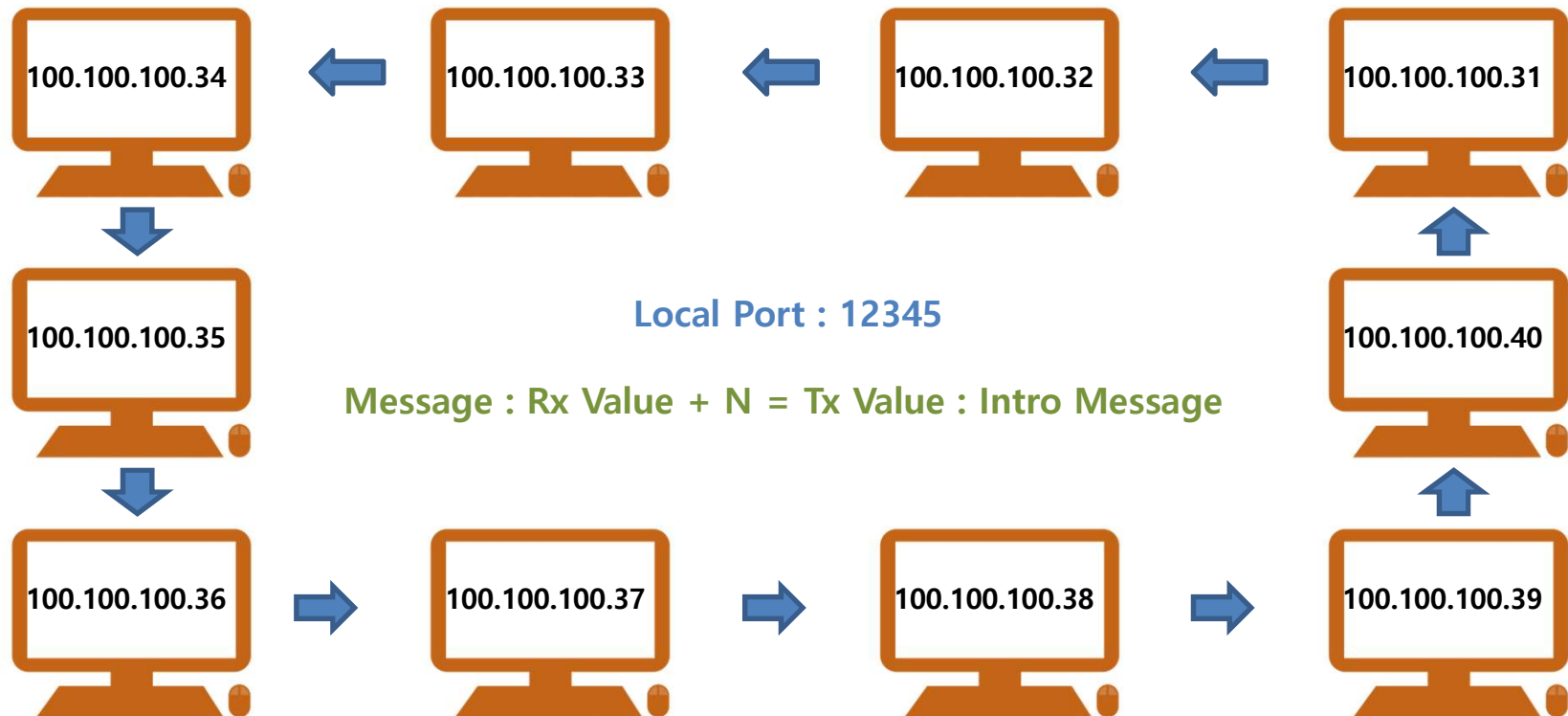
## Ex2 : Run

```
geany_run_script_EIKCK1.sh
File Edit Tabs Help
Connect to 127.0.0.1
Message to Server : hellow my raspberry
b'OK'
Message to Server : █
```

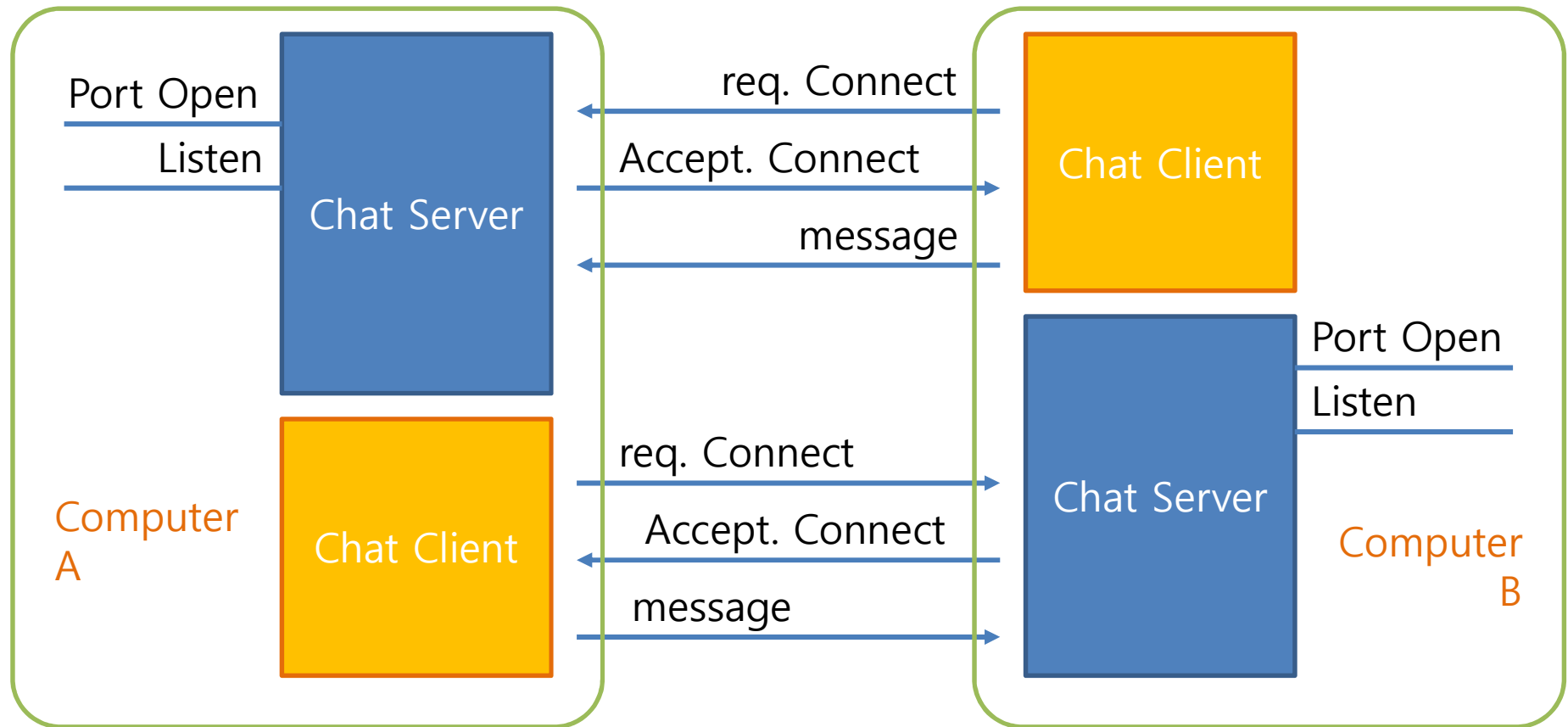
```
Shell
>>> %Run TCP_ChatServer.py
Listening for Connection : 127.0.0.1
Waiting :
Connection form ('127.0.0.1', 41024)
b'hellow my raspberry'
```



# 실습

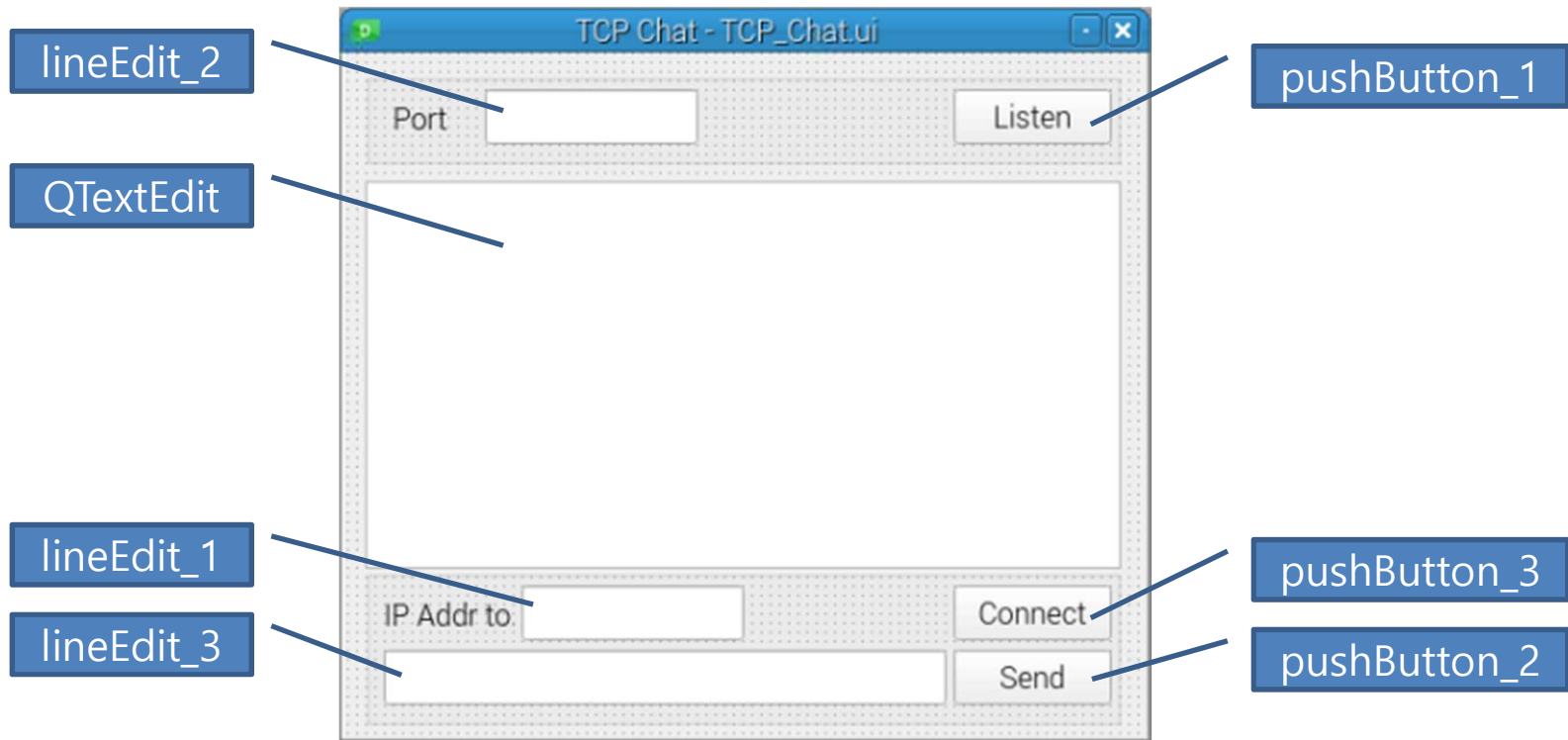


# TCP Chat Server/Client





# Chat UI



# Define

```
1 import sys
2 from socket import *
3 myIP='127.0.0.1'
4 port=12345
5
6 Status=False
7
8 from threading import Thread
9
10 from PyQt5.QtWidgets import *
11 from PyQt5 import uic
12
13 uiDialog='TCP_Chat.ui'
14 bListen=False
15
```



# class MyDialog

```
16 class MyDialog(QDialog):
17     def __init__(self):
18         QDialog.__init__(self, None)
19         uic.loadUi(uiDialog, self)
20
21         self.lineEdit_1.setText('')
22         self.lineEdit_2.setText(str(port))
23         self.pushButton_1.setFocus()
24         self.pushButton_1.clicked.connect(self.Server_Listen)
25         self.pushButton_2.clicked.connect(self.Send)
26         self.pushButton_3.clicked.connect(self.Connect)
27         self.server=None
28         self.acceptor=None
29
30         self.client=None
31
32     def Server_Listen(self):
33         if self.pushButton_1.text()=='Listen':
34             self.pushButton_1.setText('Close')
35
36             self.server=socket(AF_INET,SOCK_STREAM)
37             self.server.bind((myIP,port))
38             self.bListen=True
39             self.listenThread = Thread(target=self.listen, args=(self.server,))
40             self.listenThread.start()
41             self.textEdit.append('Start Listening')
42
43         else:
44             self.pushButton_1.setText('Listen')
45
46             self.server=None
47             bListen=False
48             self.textEdit.append('Stop Listening')
```

```
50     def listen(self, server):
51         while True:
52             self.server.listen(1)
53             if self.acceptor is None:
54                 self.acceptor,addr=server.accept()
55                 strAddr=' '.join(map(str,addr))
56                 msg='Connect : ' + strAddr
57                 self.textEdit.append(msg)
58             else:
59                 rcv='[rcv] ' + str(self.acceptor.recv(1024),encoding='utf-8')
60                 self.textEdit.append(rcv)
61                 self.acceptor.send(b'Server OK')
62                 #if not bListen:
63                     # break
64
65         #self.listenThread.stop()
66         self.textEdit.append('Stop Server ')
67
68     def Connect(self):
69         if self.pushButton_3.text()=='Connect':
70             self.pushButton_3.setText('Close')
71             self.client=socket(AF_INET,SOCK_STREAM)
72             toIP=self.lineEdit_1.text()
73             self.client.connect((toIP,port))
74
75         else:
76             self.pushButton_3.setText('Connect')
77             self.client=None
78
79     def Send(self):
80         msg=str(self.lineEdit_3.text())
81         send='[send] ' + msg
82         self.client.send(msg.encode('utf-8'))
83         self.textEdit.append(send)
84         self.lineEdit_3.setText('')
```

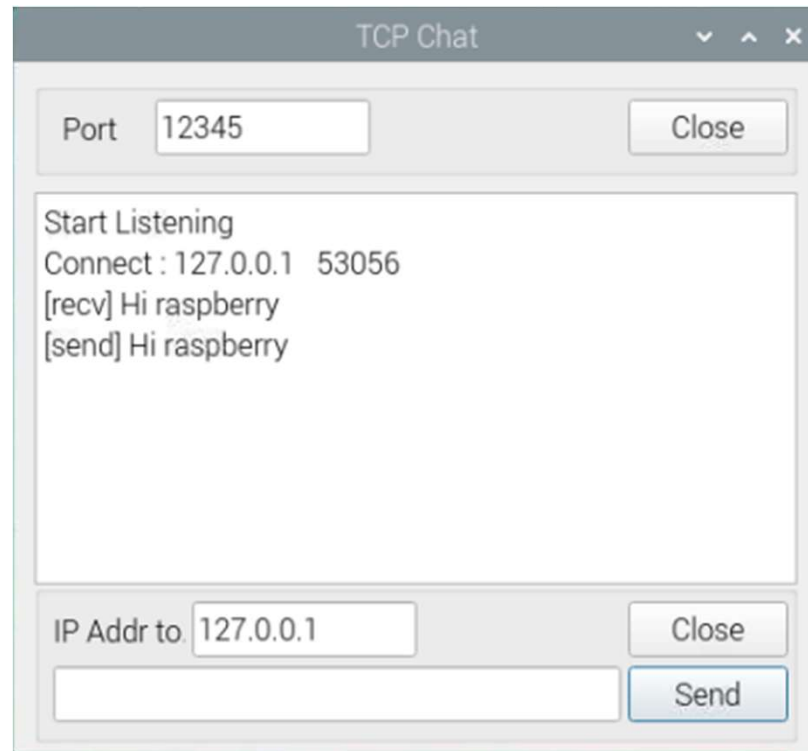


# main

```
86 if __name__ == '__main__':  
87     app=QApplication(sys.argv)  
88     form=MyDialog()  
89     form.show()  
90     app.exec()
```



# Run



# my host

- Host Name
  - `socket.gethostname( )`
- Host IP address
  - `socket.gethostbyname(socket.gethostname( ))`
- `sudo killall python3`

