**SCRPTING IN PYTHON**

Student Name: DIVAM GUPTA

Network communication is achieved by sending and receiving data on both sides of the connection. At this time, the endpoint that is in charge of the communication service on both sides is called the endpoint, and the object called the socket is in charge of the service.

So, Python **socket programming**[1] is writing code that manages both sides of a network so that they can communicate with each other. What you write in socket programming is basically a server-client model. The reason why we can download websites from Naver or Google and connect to online chatting and games is because the socket controls the network.

Of course, network programming is much more complicated than this. Separate learning about the operation structure of a complex network called OSI 7 layer is required.

It is too difficult to understand these things at once, so it is recommended to learn empirically while coding directly at first.[1]

**Server Client**

Network communication requires server-side code and client-side code. When doing socket programming, note that it is different from code in a sequential way from top to bottom in the source code. In particular, the part that sends a request to the server and exchanges data will not work if one side is not correct.

Binary server file....

activate using the specific port...

It is server side . Originally, two computers are required, but for practice, one computer is also possible. It is like practicing on the local host before programming on the server.Building the server is another task. So network programming has a lot more to do. Network programming is about manipulating multiple computers, not just one computer.

In the code above, the socket module is first imported. The socket module is included in the Python basic package, so there is no need to install it separately.

Create a default socket for the server with socket.socket().[1]

The following socket.gethostname() returns the current computer name. This is the name identified on the network. bind() binds the computer name and port number with the TCP protocol. It says the computer name, but actually opens the port number to the IP address. Port numbers are used to identify TCP services.

Multiple port numbers are used for one server. For example, web service http usually uses port number 80. Port numbers should not overlap because they are used by servers to distinguish between types of services. If you type netstat -an in the command prompt of Windows, the server is created and Listening – the status of listening is checked.

Also, you can see the pre-occupied port number in netstat. An error occurs if the socket attempts to use a port number that is already in use.[2]

TCP 127.0.0.1:999 0.0.0.0:0 LISTENING

**Given Data**

[BASIC]

saveDir=./tmp/savedGames

score=./tmp/scores.txt

[NETWORKING]

serverIP=127.0.0.1

port=7823

[FILTER\_SCORE]

scoreTimeFrom=Nov 10 07:00

scoreTimeTo=Nov 16 07:00

listening sets the number of connections. Depending on the client, there may be cases where you need to do various things with the server. In such cases, limit the number of connections possible to prevent too many clients from connecting at the same time.

**Client Side**

import socket

import configparser

import time

import os

import pprint

from configparser import ConfigParser

def main\_loop....

def menu.....

game = main\_menu()

The client side also creates sockets. It is good to write the meaning of the variable as client. A common industry practice is to use the variable s for servers and c for clients, but the words have been omitted for brevity.

Since this example uses localhost, connect to the same computer name as the server in the connect method, but connect using IP or domain name for remote access.

It is also important that the IP and port number match. Even if the address number of the server the client is trying to connect to is correct, if the service is different, the connection will not be made.[3]

**References**

[1] Radhika, K. A., Raksha, B. L., Sujatha, B. R., Pruthviraj, U., & Gangadharan, K. V. (2018, August). IoT based Joystick controlled pibot using socket communication. In *2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)* (pp. 121-125). IEEE.

[2] Thaker, N., & Shukla, A. (2020). Python as multi paradigm programming language. *International Journal of Computer Applications*, *177*(31), 38-42.

Hunt, J. (2019). Sockets in Python. In *Advanced Guide to Python 3 Programming* (pp. 457-470). Springer, Cham.

[3] Cynthia, J., Mohankumar, T., Arjun, T., & Naveenkumar, C. (2021, December). Development of Python Based UI Application for Tele-Operated Vehicles. In *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)* (pp. 383-388). IEEE.