Network Fundamental: Project A

14459933 Eui Ri Kim

Project A brief

The goal of this project is to create server and client codes to communicate with each other through a TCP connection. First, if you look at the server program, define the server socket and initiating it first. If you look at Screen clipping 1, there are variables (AF\_INET, SOCK\_STREAM) declared to designate Serverport first, and to operate ServerSocket as TCP. socketBind socket address assignment, socketListen can successfully prepare to receive server socket by waiting for connection request. And by using the while statement, the file is split and the socket transmission to send continues to occur so that all the files are received.

텍스트이(가) 표시된 사진

자동 생성된 설명

In this server programming, we made it possible to perform concurrent connection requests using multi-threading. First, import Threading to load the thread library and execute the threading.thread() method. In it, specify the target as encodingTargetClient and put clientSocket as the argument. This causes one thread to execute encodingTargetClient(clientSocket).텍스트이(가) 표시된 사진

자동 생성된 설명 When entering, defFlieName(clientSocket) is executed first, and the received from clientSocket.recv(4096) is decomposed with decode(), and each character is entered in the filename using message.split. Then, give open to the f variable to open the file with the name corresponding to filename. It reads this and returns it using a variable called outputData.

텍스트이(가) 표시된 사진

자동 생성된 설명

Then, the header is sent using sendHeader(clientSocket) to the file. The header has HTTP/1.1 200 ok and content-Type = ‘Content-type: text/html’. The HTTP header is 200 ok for version and success, and the content-Type tells us that this text is in text/html format.

텍스트이(가) 표시된 사진

자동 생성된 설명

If an Error occurs, sendErr(clientSocket) is executed. It sends ‘HTTP/1.1 404 Not Found’, which is referred to as an error message.

텍스트이(가) 표시된 사진

자동 생성된 설명

If all of these are executed, one deamonThread is terminated, and the mainThread is continuously executed again. This ensures that serverSocket.close() does not end until arbitrarily specified.

텍스트이(가) 표시된 사진

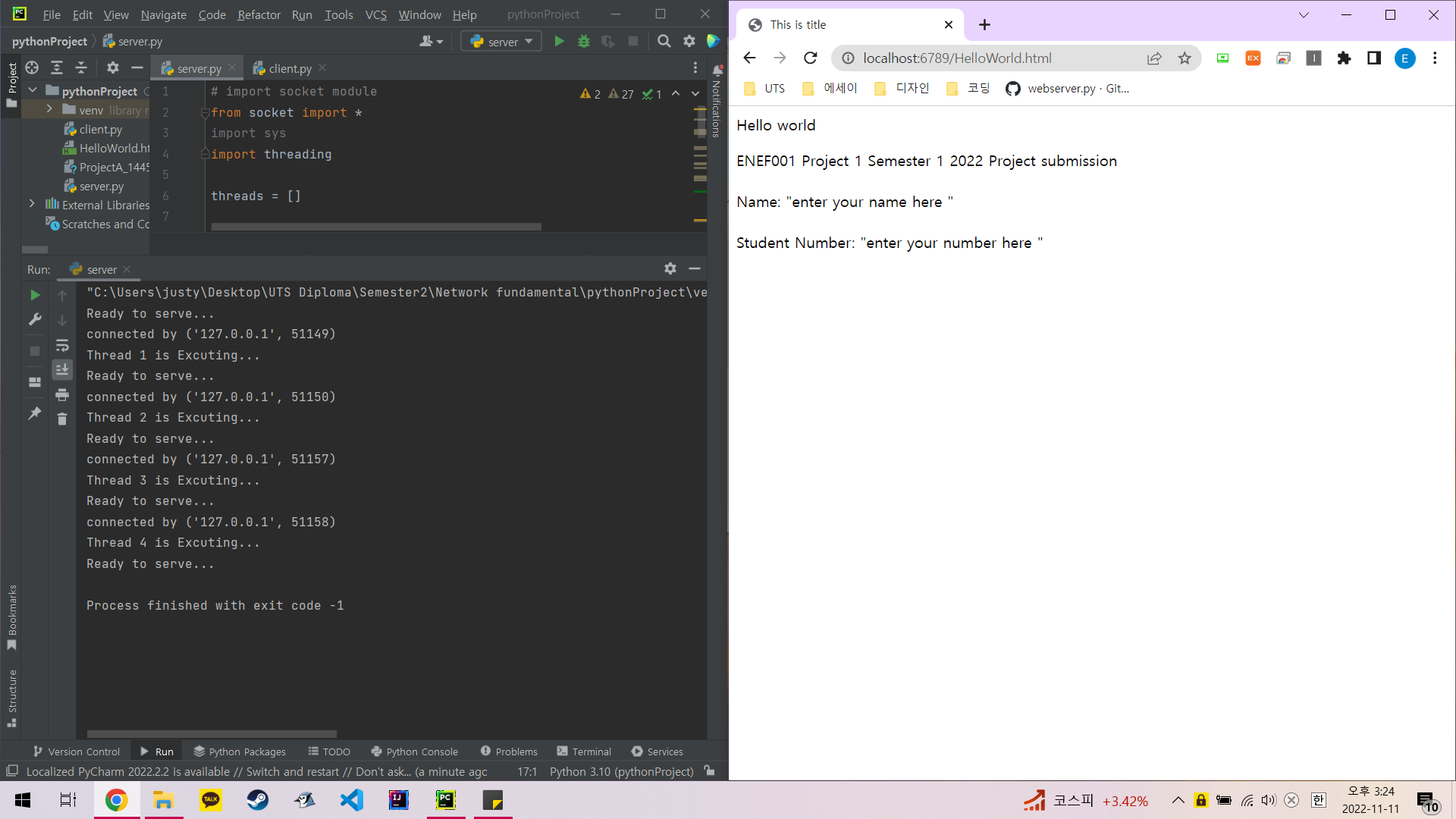
자동 생성된 설명

The following is about the Client. First, import the socket, and call the current IP address with the host gethostname() method. ServerPort is input, and filename is also input. After that, after defining the client socket as a TCP connection, encode() getFile = ‘GET /FileName HTTP/1.1’ to send it to the server, receive the clientSocket again, and print the message returned from the server. And close the socket.

Result of client.py: From Server: HTTP/1.1 200 OK

텍스트, 스크린샷, 모니터, 전자기기이(가) 표시된 사진

자동 생성된 설명

If I connect with chrome browser, it express like this.  


If I do not write the available port number as 8080:

텍스트이(가) 표시된 사진

자동 생성된 설명