I will give you some information and you need to implement an FTP server in Python using only the Python standard library, excluding the ftblib module.

For the FTP server you implemented in Python, it should support the following commands for FTP clients. Attention, I will give you the command name(in uppercase), the description, and the command usage, they will be separated by a slash(/). Here are the commands: USER/Authentication username/USER username, PASS/Authentication password/PASS password, PORT/Specifies an address and port to which the server should connect/PORT xxx,xxx,xxx,xxx,yyy,yyy(xxx represents a segment of the IP address, and yyy represents the port number), EPRT/Specifies an extended address and port to which the server should connect/EPRT|xxx.xxx.xxx.xxx|yyyyy|, STOR/Accept the data and store the data as a file at the server side/STOR filename, RETR/Retrieve a copy of the file/RETR filename, SYST/Return system type/SYST, and SIZE/Return the size of a file/SIZE filename. Each command should be ended with \r\n

In response to the commands, the server will respond with a 3-digit status code, followed by a sentence explaining the status code. Please also implements responses(status code and description)from the FTP server including: 220 CS305 FTP server ready. 331 Username ok, send password. 230 Login successful. 200 Type set to: Binary. 213 xxxx(xxxx represents the size of the file). 200 Active data connection established. 125 Data connection already open, Transfer starting. 226 Transfer complete. 221 Goodbye. 504 Command not implemented for that parameter. 502 Command not implemented. 421 Service not available, closing control connection. 425 Can’t open data connection. 426 Connection closed; transfer aborted. 430 Invalid username or password. 530 Not logged in. 534 Request denied for policy reasons.

In summary, you should implement an FTP server that can:

Handle connection: Listen on port 52305, accept connections, and close connections upon QUIT command. After closing a connection, it should continue to wait for succeeding connections. You should change the welcome message to a SID. For example, my SID is 12116666, the welcome message should be: 220 12116666 ready.

Anonymous logins: Correctly handle USER command. After the user sends the user name, the server should acknowledge that the login is successful. Transfer files: Correctly handle RETR and STOR commands. The server should be able to properly receive and store a file whose name and content are random strings and to properly transfer a file to the client, with its filename and content not modified.

You should also optimize your server so that it can handle the following errors: File errors: File not exist, file not accessible, illegal filename. Command errors: Operations before login, file transmission before connecting, illegal command (format error, command unrecognized, linefeed error). Connection errors: Connection establishment failure(e.g.the address given by EPRT command is unavailable), connection interrupted (e.g. the data or the control connection breaks up when transmitting files), client down.