Server

The **server** is just a python program that utilizes sockets. Socket programming is a way of connecting two nodes of a network to communicate and send data to earch other.

The data that needs to be sent back and forth between the client and the server is supposed to be encapsulated in a file, for example a JSON (JavascrIPt Object Notation) or a CSV (Comma Seperated Values) file, the current implementation can handle sending and receiving any kind of files.

Let's take a quick look at the implementation:

```
PORT = 5010
SERVER = socket.gethostbyname(socket.gethostname())
BUFFER_SIZE = 4096
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
```

Constant	Usage
PORT	The port that will be used to communicate with the server, so any time one of the clients want to send data to the server, it's supposed to send this data on that port.
SERVER	The IP that the server will be hosted on, since in our case it will be hosted on localhost we need to get the laptop's local IP, that's done by using `socket.gethostbyname` function.
BUFFER_SIZE	The number of bytes that will written to received file.
ADDR	A tuple that holds both `SERVER` and `PORT` values to be used when creating the socket.
FORMAT	The format that will be used when converting the bytes received from the client to string.

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)
```

After that comes defininig a socket object, that will represent the server, using `socket.socket` function that takes two arguments, the first one represents the address family `socket.AF_INET` is the internet address family for IPv4 (Internet Protocol version 4), the second one specifies the socket type `socket.SOCK_STREAM` is the socket type for TCP (Transmission Control Protocol).

```
def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
```

Then comes the definition for `start` function, `server.listen` is used to make the server start listening for incoming request and data on the specified `PORT` and `IP`, and if a request is received a thread is created to handle that request, using threads will allow the server to handle multiple clients in parallel.

```
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")

    connected = True
    while connected:

        msg = conn.recv(BUFFER_SIZE).decode(FORMAT)

        if msg:
            filename = msg.split(" ")[0]
            with open(filename, "wb") as f:
                while True:
                    # read 1024 bytes from the socket (receive)
                    bytes_read = conn.recv(BUFFER_SIZE)
                    if not bytes_read:
                        # nothing is received
                        # file transmitting is done
                        break
                    # write to the file the bytes we just received
                    f.write(bytes_read)
                    f.close()
```

At last a function `handle_client` is defined that's responsible for dealing with the incoming requests, when a request is sent a connection object is received in this function, this connection object have the data sent by the client and by using a method on `conn` object called `recv` the server can receive the sent data successfully, this `recv` method takes one argument that specifies the size of data sent in bytes, since the communication between the client and the server will be in the form of files, the server needs to know the name of that file and store it in `filename` variable`, after that a file is created by using` open` function in "wb" mode that allows us to write data to existing file or create a new file if that file doesn't exist, then the server enters a loop that will continue to receive the data of the file from the client until all of the data is sent.