```
"""
Server script for socket based file downloader
Usage: python3 server.py listen_port

3/8/21
Alex Burling
88866582
"""


import sys
import socket
from datetime import datetime
from FileRequest import FileRequest
from FileResponse import FileResponse


#Parses command line arguments and checks validity
def parse_args():
    if len(sys.argv) != 2:
        sys.exit("Usage: python3 server.py listen_port")

    port = int(sys.argv[1])
    if port < 1024 or port > 64000:
        sys.exit("PORT NUMBER SHOULD BE >= 1,024 AND <= 64,000")

    return port


""" Starts socket and attempts to bind to the provided
    port, exits on error"""
def start_socket(port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        hostname = socket.gethostname()
        sock.bind((hostname, port))
        sock.listen()
    except:
        sock.close()
        sys.exit("ERROR OPENING SOCKET")

    return sock, hostname


""" Recieves FileRequest from provided socket"""
def recieve_request(sock):
    recieved = bytearray(sock.recv(1024))
    file_request = FileRequest.from_bytearray(recieved)
    while (not file_request.check_len()):
        recieved = bytearray(sock.recv(1024))
        file_request.append_data(recieved)

    return file_request


""" Generates a FileResponse from provided filename and
    sends to provided socket"""
def send_response(sock, filename):
    res = FileResponse.from_filename(filename)
    packet, packet_len = res.generate_packet()
    client_addr = sock.getpeername()
    print("[{}]  {}:{} SENT {} BYTES".format(datetime.now().strftime(
        "%H:%M:%S"), client_addr[0],  client_addr[1], packet_len))
```

```
    sock.send(packet)


""" Main function"""
def run_server():
    port = parse_args()
    sock, hostname = start_socket(port)
    print("Listening on [{}]{}:{}".format(hostname, sock.getsockname()[0], str(port)))

    while True:

        try:
            client_sock, client_address = sock.accept()
            client_sock.settimeout(1)

            print("[{}]  {}:{} CONNECTED".format(datetime.now().strftime(
                "%H:%M:%S"), client_address[0], client_address[1]))

            file_request = recieve_request(client_sock)

            file_request.validate()
            filename = file_request.get_filename()

            print("[{}]  {}:{} GET FILE '{}'".format(datetime.now().strftime(
                "%H:%M:%S"), client_address[0], client_address[1], filename))

            send_response(client_sock, filename)
            client_sock.close()

        except AssertionError: #thrown by FileRequest.validate()

            print("[{}]  {}:{} RECIEVED MALFORMED PACKET".format(datetime.now().strftime(
                "%H:%M:%S"), client_address[0], client_address[1]))
            send_response(client_sock, None)
            client_sock.close()

        except socket.timeout:

            print("[{}]  {}:{} CONNECTION TIMED OUT".format(datetime.now().strftime(
                "%H:%M:%S"), client_address[0], client_address[1]))
            client_sock.close()


def main():
    run_server()


if __name__ == "__main__":
    main()
```