```
"""
Client script for socket based file downloader
Usage: python3 client.py server_ip server_port target_file

3/8/21
Alex Burling
88866582
"""


import os.path
import socket
import sys
from FileRequest import FileRequest
from FileResponse import FileResponse


#Parses command line arguments and checks validity
def parse_args():

    if len(sys.argv) != 4:
        sys.exit("Usage: python3 client.py server_ip server_port target_file.")

    server_ip = str(sys.argv[1])
    server_port = int(sys.argv[2])
    target_file = str(sys.argv[3])

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

    try:
        server_addr = socket.gethostbyname(server_ip)
    except socket.gaierror:
        sys.exit("MALFORMED IP/HOSTNAME")

    if (os.path.exists(target_file)):
        sys.exit("The requested file already exists locally.")

    return server_ip, server_port, target_file, server_addr


""" Starts socket and attempts to connect to the provided
    server address and port, exits on connection error"""
def start_socket(server_addr, server_port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        sock.connect((server_addr, server_port))
    except ConnectionRefusedError:
        sock.close()
        sys.exit("{}:{} REFUSED CONNECTION.".format(
            server_addr, str(server_port)))

    return sock


""" Recieves FileResponse from provided socket and counts
    number of bytes recieved. Closes socket once full response
    recieved"""
def recieve_response(sock):
    recieved = bytearray(sock.recv(1024))
    packet_len = len(recieved)
    file_response = FileResponse.from_bytearray(recieved)
```

```
    while (not file_response.check_len()):
        recieved = bytearray(sock.recv(4096))
        packet_len += len(recieved)
        file_response.append_data(recieved)
    sock.close()
    return file_response, packet_len


""" Generates a FileRequest from provided filename and
    sends to provided socket"""
def send_request(sock, filename):
    req = FileRequest.from_filename(filename)
    sock.send(req.generate_packet())


""" Writes retrieved data into target file"""
def write_file(file_response, target_file):
    file = open(target_file, 'wb+')
    file.write(file_response.get_data())
    file.close()


""" Main function"""
def run_client():
    server_ip, server_port, target_file, server_addr = parse_args()

    print("REQUESTING '{}' FROM [{}]{}:{}".format(
        target_file, server_ip, server_addr, str(server_port)))

    try:
        sock = start_socket(server_addr, server_port)
        sock.settimeout(1)

        send_request(sock, target_file)

        file_response, bytes = recieve_response(sock)
        print("RESPONSE RECIEVED, RECIEVED {} BYTES".format(bytes))

        file_response.validate()

        write_file(file_response, target_file)
        print("FILE SUCCESSFULLY DOWNLOADED")

    except socket.timeout:
        sock.close()
        print("CONNECTION WITH [{}]{}:{} TIMED OUT".format(
            server_ip, server_addr, str(server_port)))


def main():
    run_client()


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