Design File

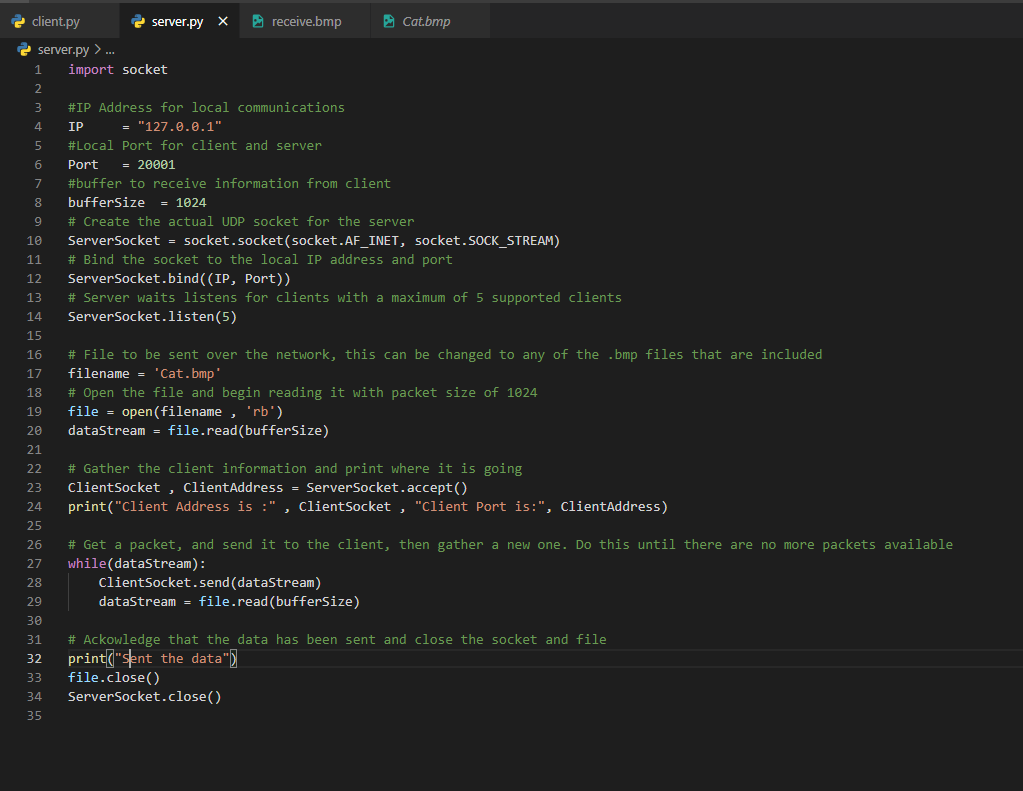
Phase 2: Implementing RDT 1.0 over a reliable UDP channel

**Authors:**   
Kyle Purcell, Kyle Luong, Edward Guenette

**Purpose of the Project:**   
Programming Project Phase 2 provides students with the knowledge to transfer a bmp file between a UDP client process and a UDP server process. The students will provide reliable data transfer service using the RDT 1.0 protocol.

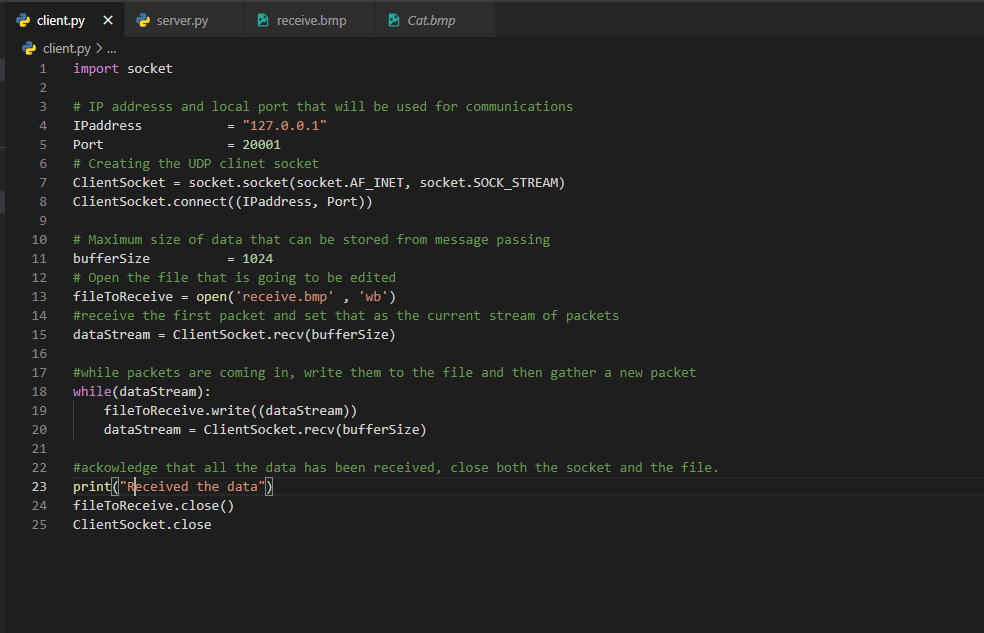
**Code Explanation:**

server.py



This is code from the server side of the UDP protocol. The code is organized in a way where we first import the socket library. Then add both the IP, port number and the buffer size that will be used along with creating and binding the socket that will be used. The server will then listen using the listen() function built into the socket library. The file that will be sent back to the user will then be set as the “filename” (Note: The full path name needs to be given here if the files are not in the desktop). Then the open() and read() functions are used in order to open the file and read it as bytes which will be set to dataStream. Then in the next block, the client information is taken and printed. The next block is where the packets are received, sent to the client and read into the buffer. This will be done until there are not more packets available to be sent. The final block just notifies the client has been sent the data and the close() method will close both the file and server socket.

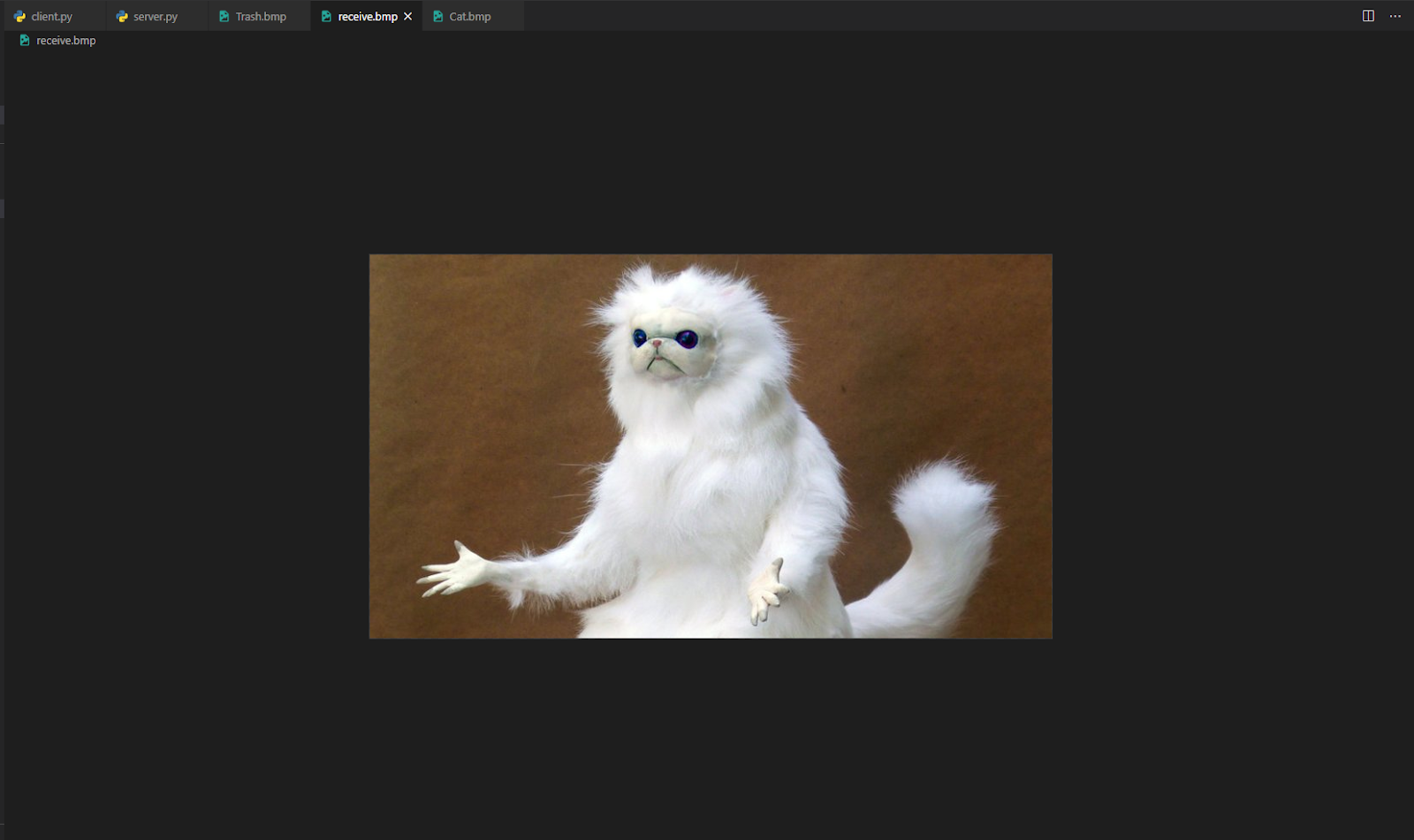
client.py



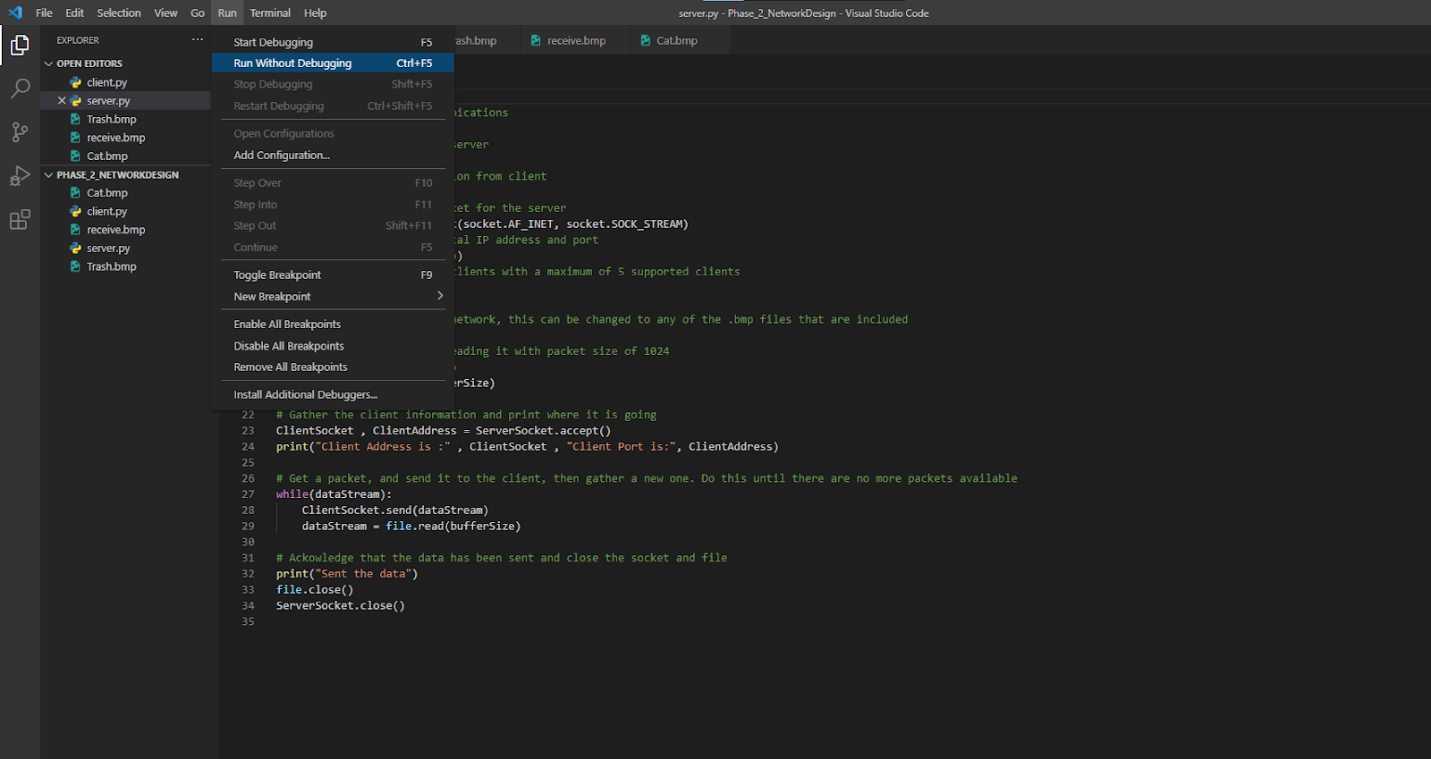
This file defines the client process that we implemented during this phase of the project. After importing the socket library from python, the IP address and Port are defined and the socket is created from lines 3 to 8. Next, the maximum packet size is set to 1024 bits. After establishing that, we needed to open the image in fileToRecieve with the open() function (line 13). Once the file was open, we could start receiving packets from the server. After initiating the transfer from the server, a while loop was formed so that as long as the datastream is active, the client will continue to write the received packets to the fileToRecieve file. Finally, we printed the received data, closed the file, and closed the socket, ending the server and client communication.

**Execution Example:**

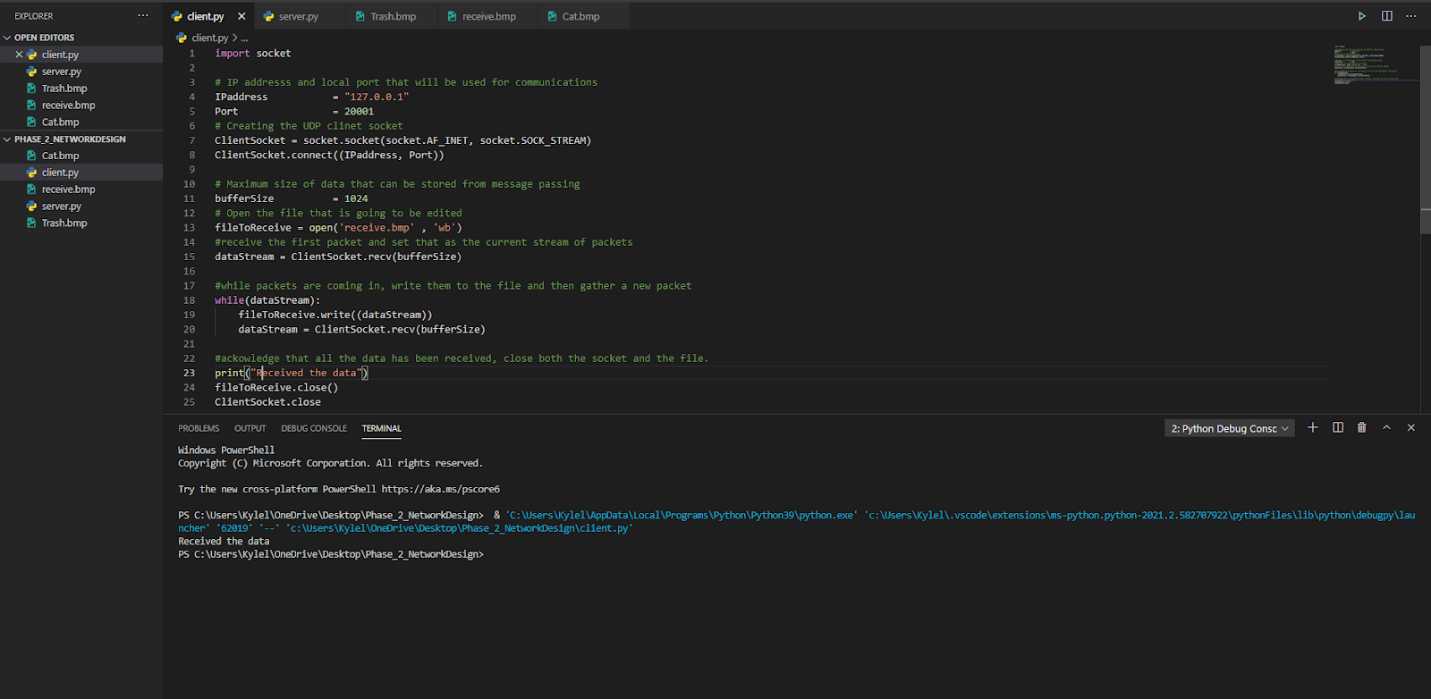
Original image in receive.bmp file

****

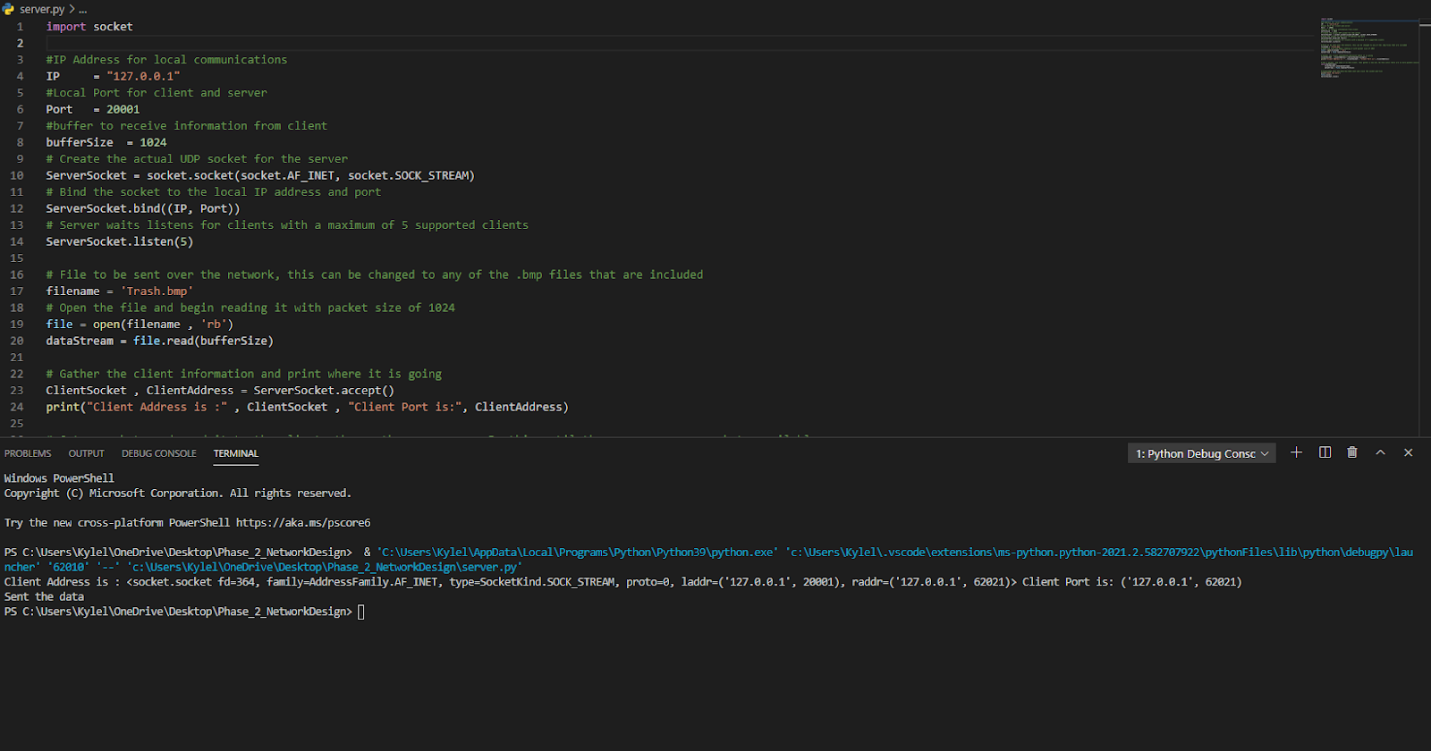
Running the server.py first

****

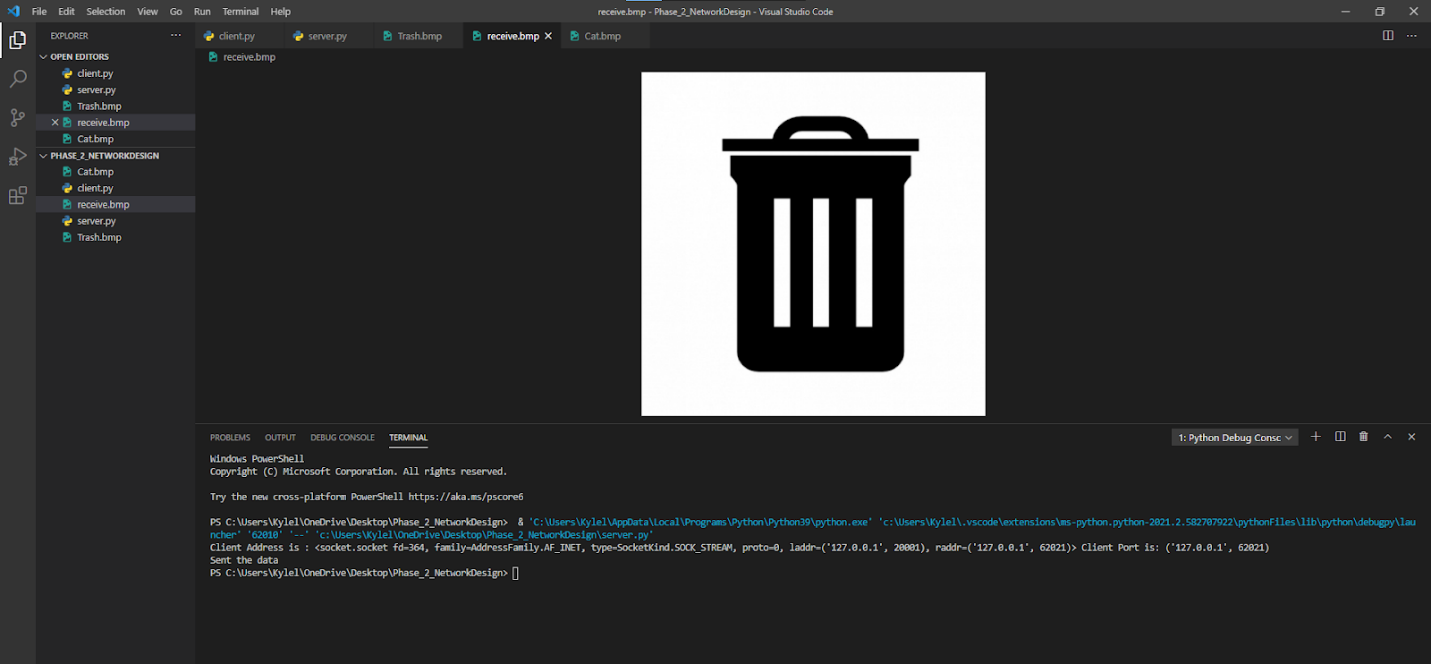
Run the client.py second

****

Terminal after running both files

****

Receive.bmp image after executing the code

****