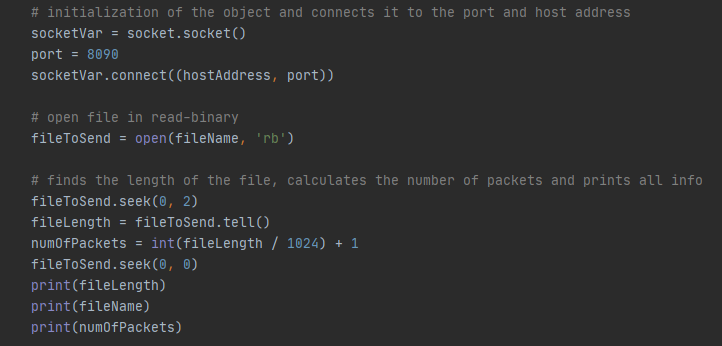
Phase 2: Implement RDT 1.0 over a reliable UDP channel

By: Julie Dawley, Mohammad Musawer, Ricardo Candanedo

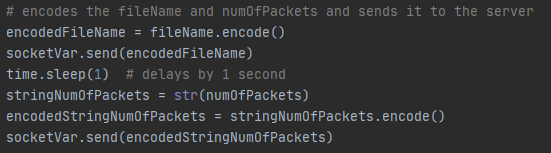
**Purpose:**

Implementing RDT 1.0 using a UDP channel and transferring an image file between a UDP client and UDP server processes.

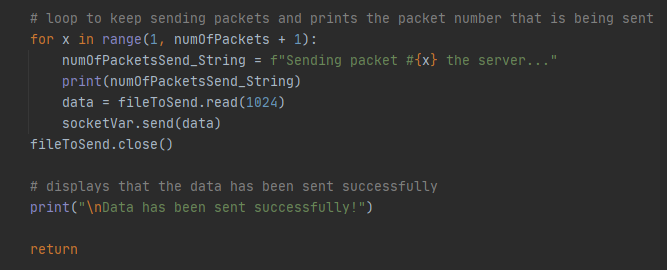
**client.py**



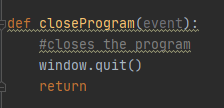
The code block above initializes a socket object port number for the client to establish a connection to the server. In addition, the program opens the file that needs to be transferred in “read-binary” format and then uses the seek() function to read the file from the beginning to the end. This code block also returns the file length and calculates the number of packets that the file needs to send to the server. Finally, this block of code will display the file length, file name, and number of packets to the user.



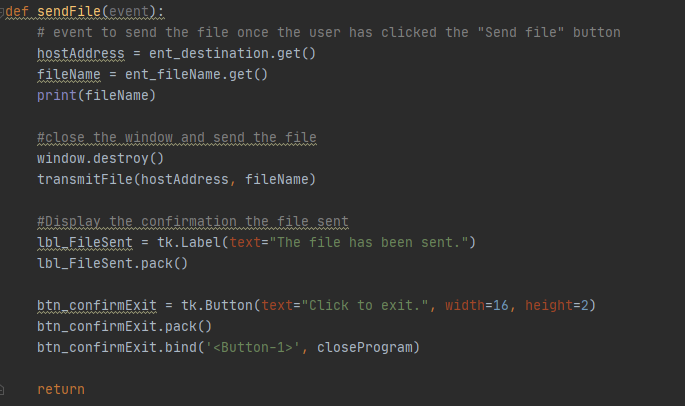
This code block encodes the file name and the number of packets then sends it through the socket connection. There is also an added delay command that delays the process for 1 second.



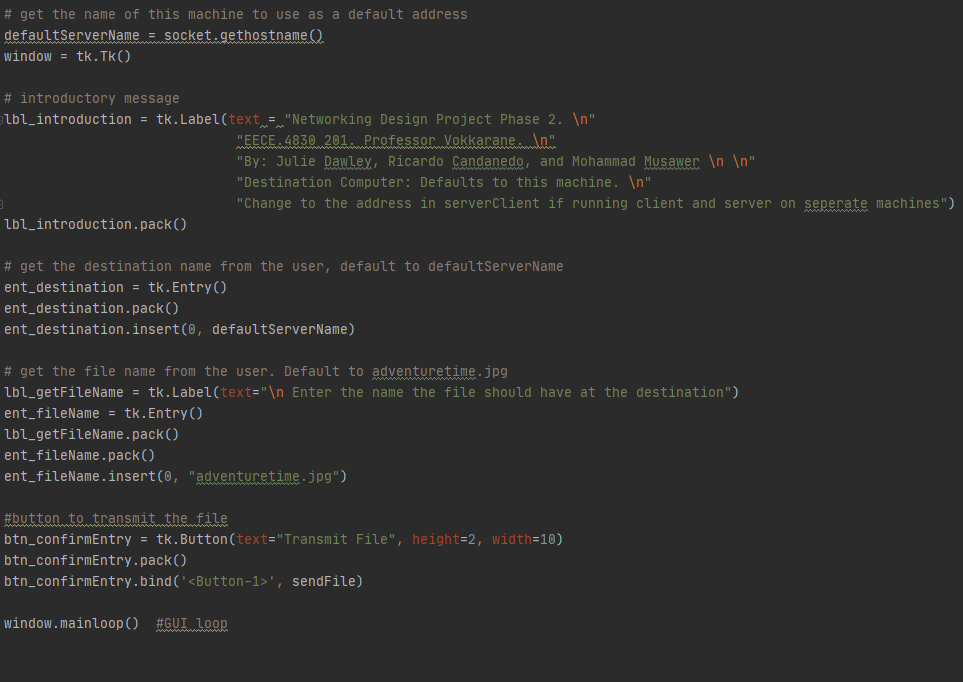
For loop added to send packets to the server and display the packets as it is being sent. The data is sent by reading 1024 bytes at a time.



Closes the final confirmation window, causing the program to terminate.

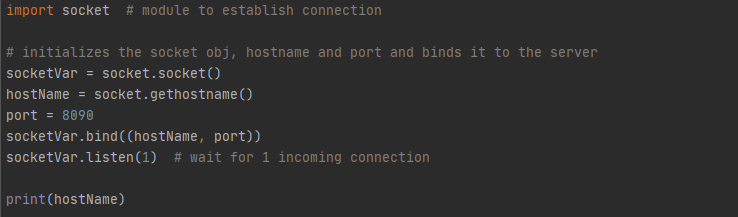


Takes the information inputted by the user (or the defaults if no changes were made), closes the window, then sends the file to the server by executing transmitFile. Then displays a window saying the file was sent and a button to exit.

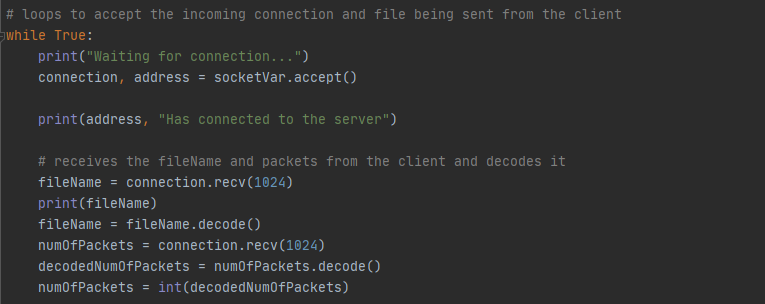


Display the opening window containing an introductory message, a textbox for the user to enter the destination computer or leave defaulted to the current computer, a box to enter the file to send or leave defaulted to adventuretime.jpg, and a button to transmit the file.

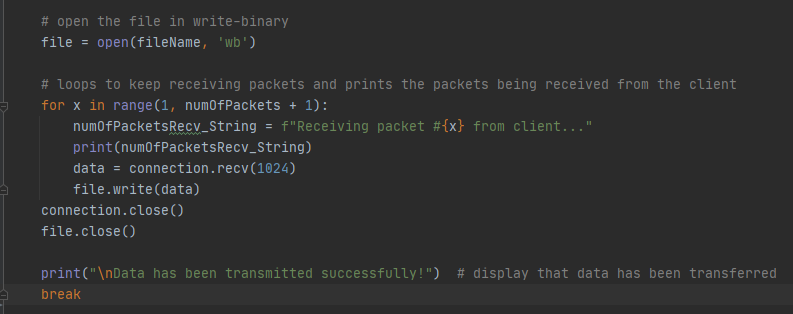
**server.py**

****

This block of code first imports the socket module to allow for UDP sockets within the file and begins to initialize variables and objects such as socketVar, hostName and port. After initialization, the host name and port get binded to the server and waits for an incoming connection from the UDP client.

****

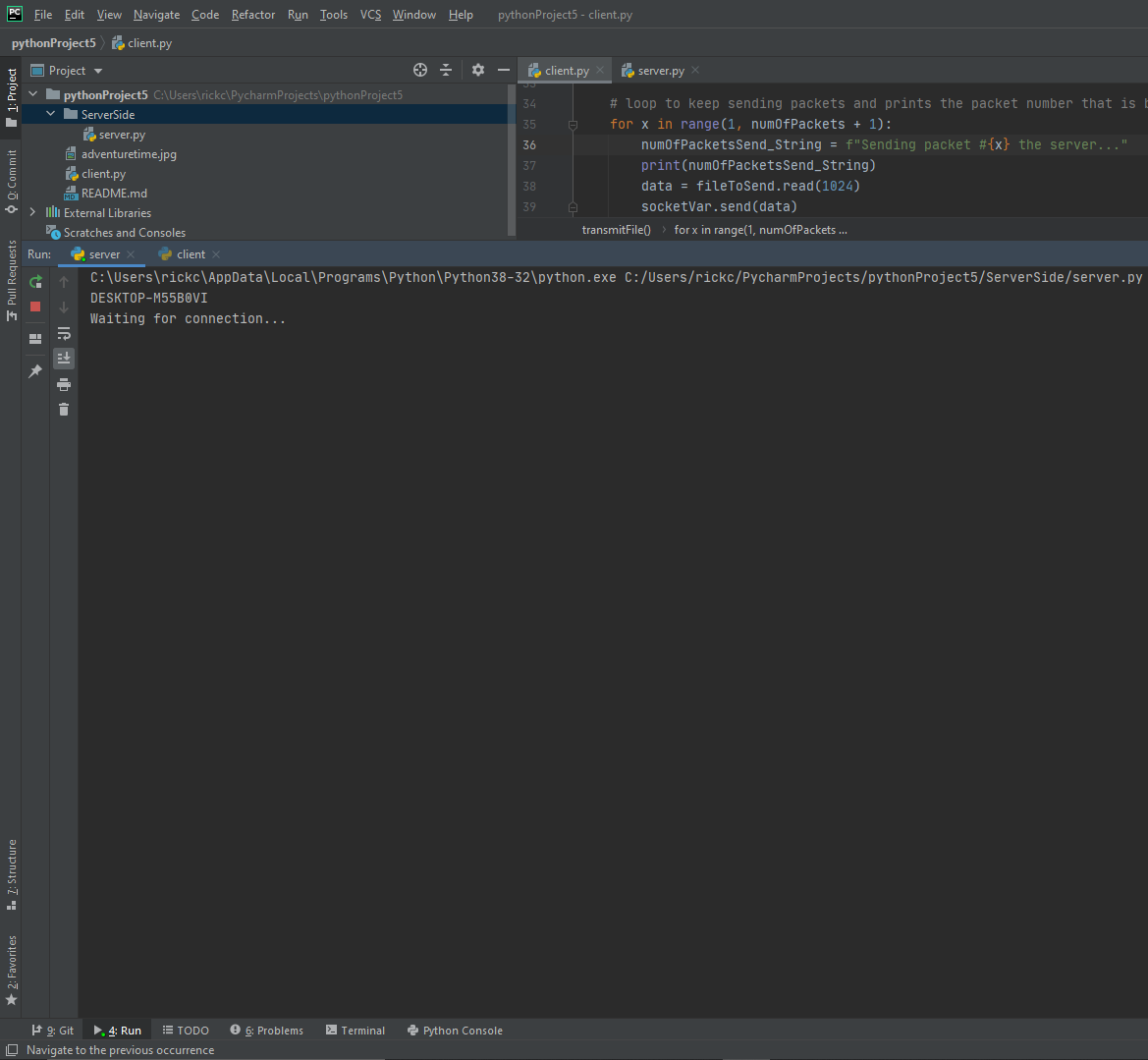
During the loop, the server awaits a connection from the client and tells the user when the server has connected successfully. After, the file name and number of packets that were sent from the client are received and then decodes them as well.

****

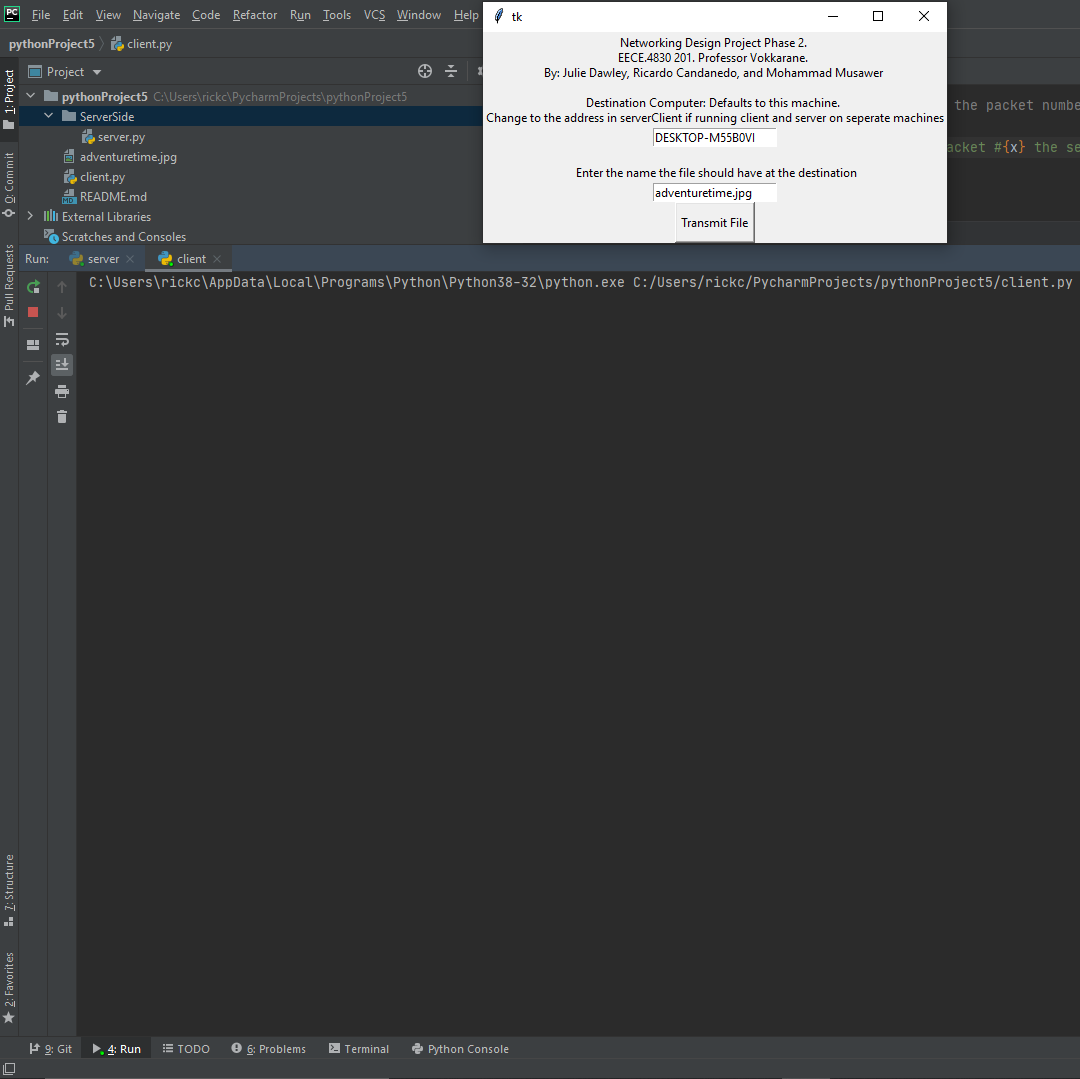
The file then opens and the packets are then received one by one in packets within the for loop. Each time one of the packets is received the file then writes that data. After each one gets written, the image sent from the client is fully processed and is the same as it was prior to being sent.

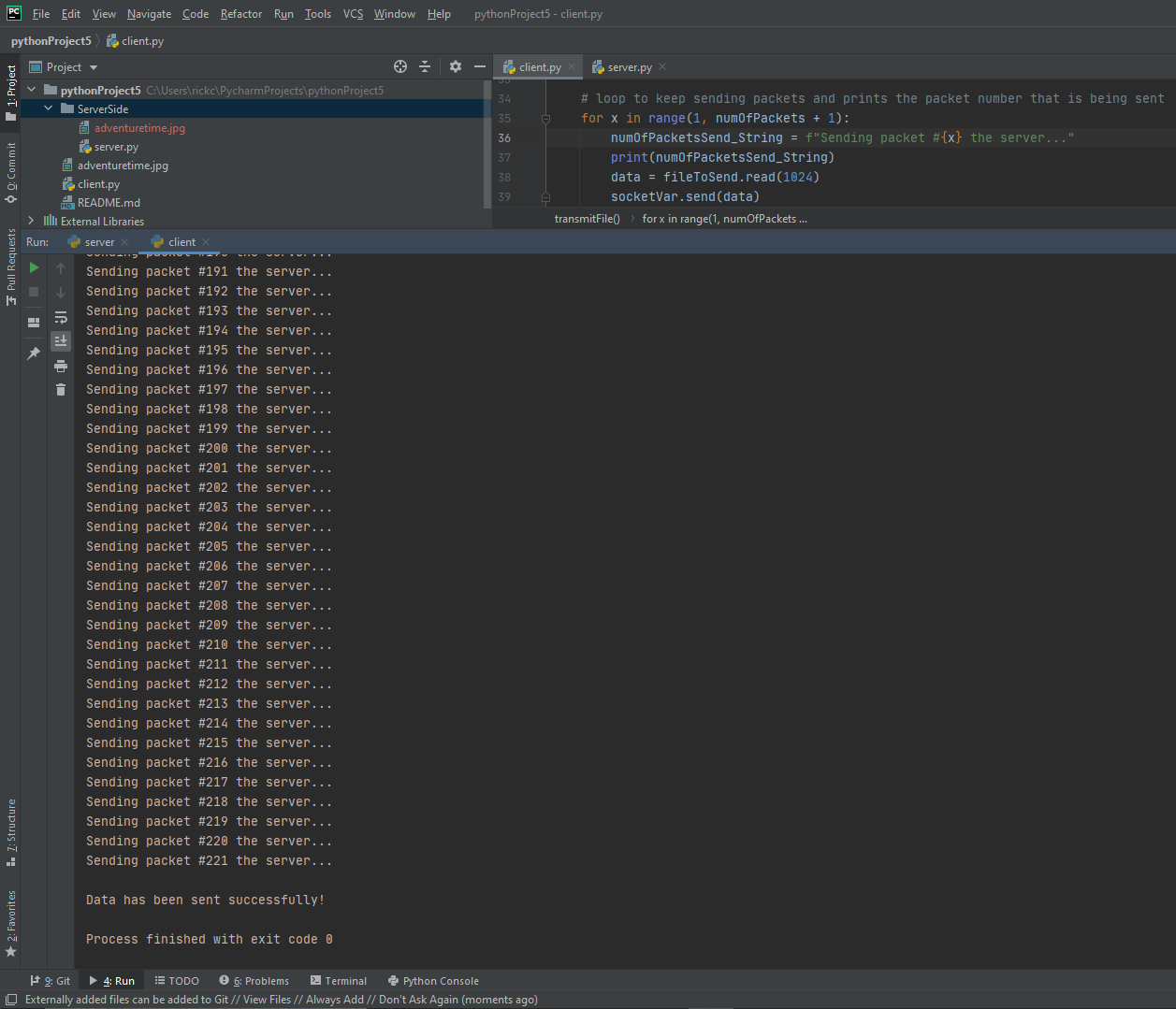
**Code Execution**

Server waiting for connection

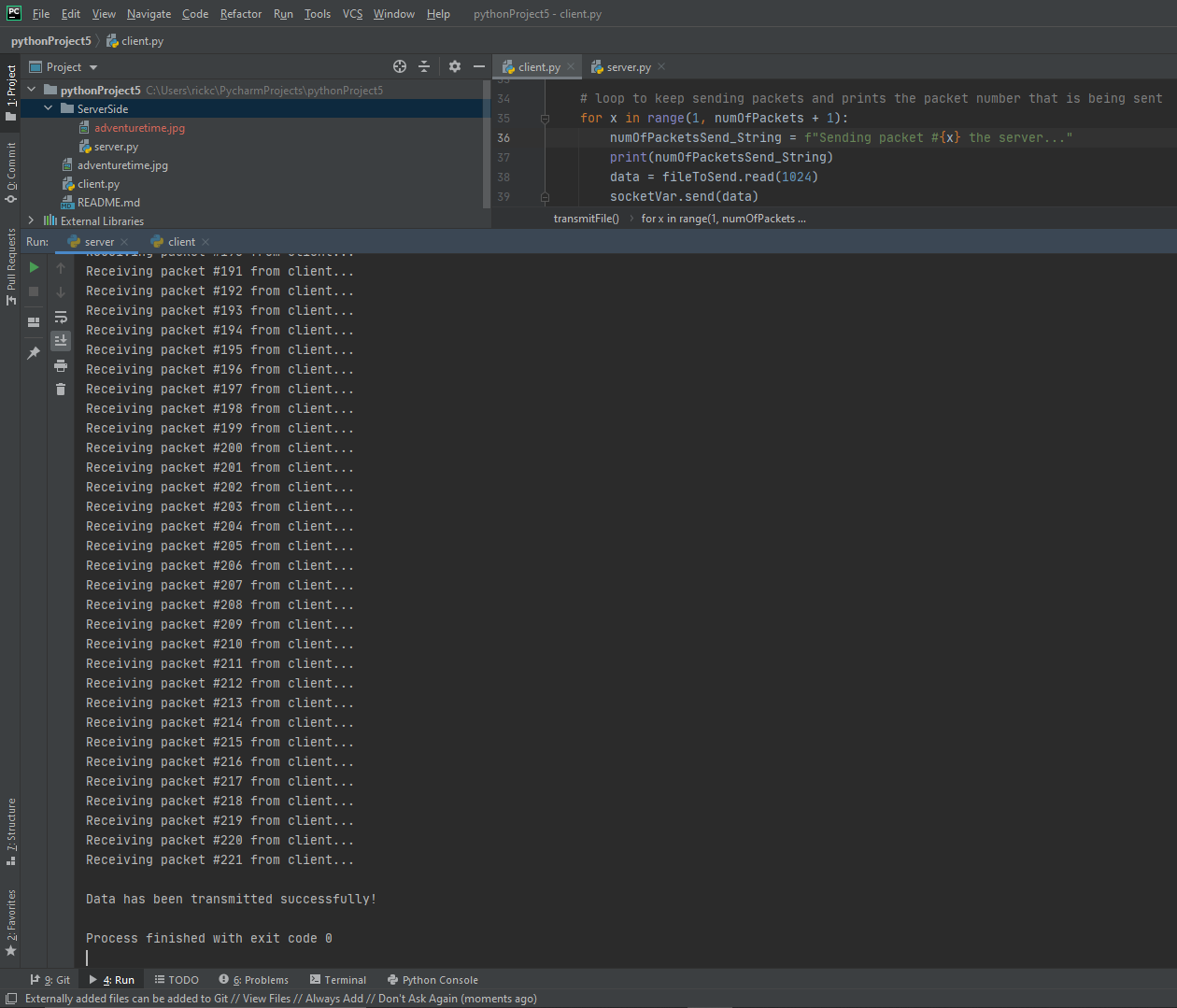


Client window to transfer file (GUI implemented)





After transmit is pressed, each packet is sent to the client one by one



After the file is sent a confirmation window appears with an exit button.

