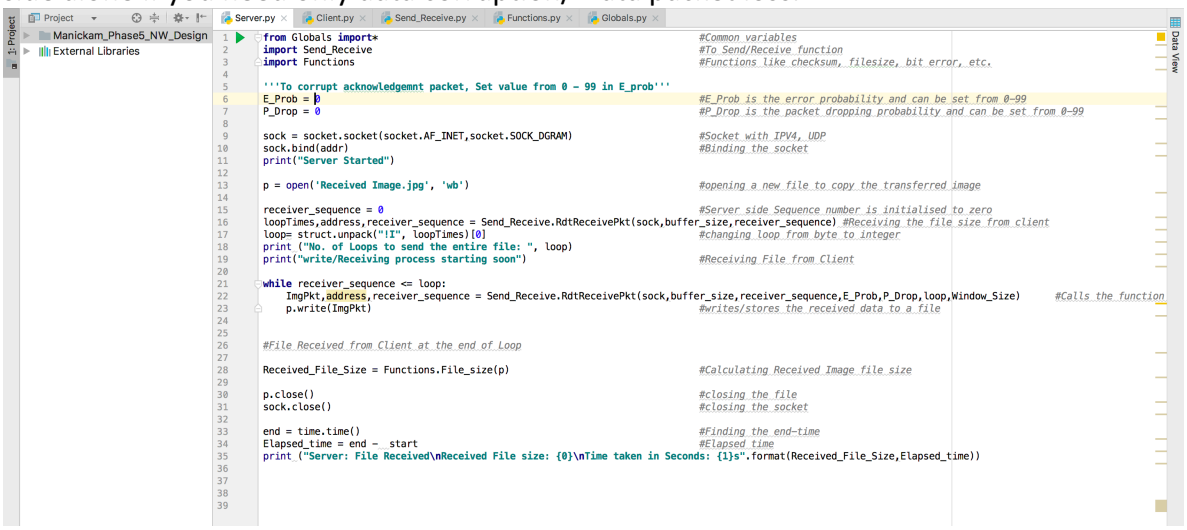
EECE 5830: NETWORK DESIGN
PROJECT PHASE 5: GO-BACK-N PROTOCOL OVER AN UNRELIABLE UDP CHANNEL
AUTHOR: ASWIN KUMAR MANICKAM, 01624359.

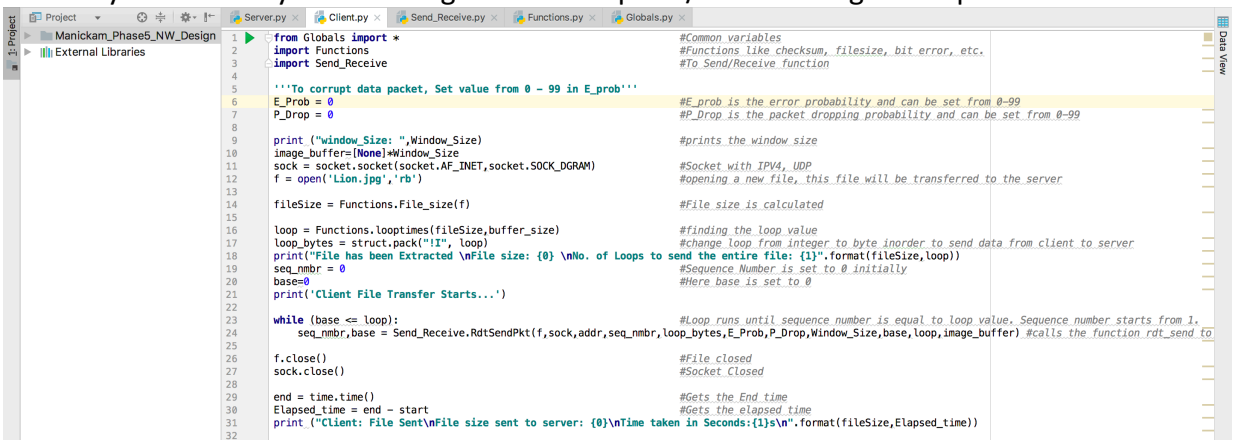
PROGRAM:

- Server.py has the main program for the server to receive a file, Set E_prob OR P_Drop in server side alone if you need only data corruption/ Data packet loss.



```
from Globals import*                                    #Common variables
import Send_Receive                                     #To Send/Receive function
import Functions                                        #Functions like checksum, filesize, bit error, etc.

'''To corrupt acknowledgemnt packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0                                              #E_Prob is the error probability and can be set from 0-99
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be set from 0-99

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)  #Socket with IPV4, UDP
sock.bind(addr)                                         #Binding the socket
print("Server Started")

p = open('Received Image.jpg', 'wb')                    #opening a new file to copy the transferred image

receiver_sequence = 0                                   #Server side Sequence number is initialised to zero
loopTimes,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence) #Receiving the file size from client
loop= struct.unpack("!I", loopTimes)[0]                 #changing loop from byte to integer
print ("No. of Loops to send the entire file: ", loop)
print("write/Receiving process starting soon")          #Receiving File from Client

while receiver_sequence <= loop:
    ImgPkt,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop,loop,Window_Size)      #Calls the function
    p.write(ImgPkt)                                     #writes/stores the received data to a file


#File Received from Client at the end of Loop

Received_File_Size = Functions.File_size(p)             #Calculating Received Image file size

p.close()                                               #closing the file
sock.close()                                            #closing the socket

end = time.time()                                       #Finding the end-time
Elapsed_time = end -  start                             #Elapsed time
print ("Server: File Received\nReceived File size: {0}\nTime taken in Seconds: {1}s".format(Received_File_Size,Elapsed_time))
```

- Client.py has the main program for the client to send a file, Set E_prob or P_Drop in client side alone if you need only Acknowledgement corruption/ Acknowledgement packet loss.

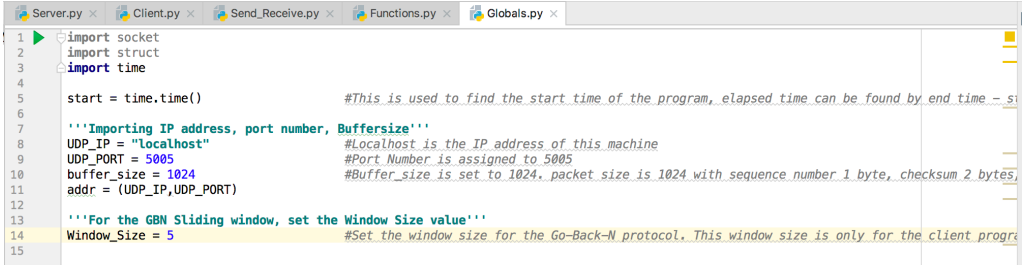


```
from Globals import *                                   #Common variables
import Functions                                        #Functions like checksum, filesize, bit error, etc.
import Send_Receive                                     #To Send/Receive function

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0                                              #E_prob is the error probability and can be set from 0-99
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be set from 0-99

print ("window_Size: ",Window_Size)                     #prints the window size
image_buffer=[None]*Window_Size
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)  #Socket with IPV4, UDP
f = open('Lion.jpg','rb')                               #opening a new file, this file will be transferred to the server

fileSize = Functions.File_size(f)                       #File size is calculated

loop = Functions.looptimes(fileSize,buffer_size)        #finding the loop value
loop_bytes = struct.pack("!I", loop)                    #change loop from integer to byte inorder to send data from client to server
print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}".format(fileSize,loop))
seq_nmbr = 0                                            #Sequence Number is set to 0 initially
base=0                                                  #Here base is set to 0
print('Client File Transfer Starts...')

while (base <= loop):                                   #Loop runs until sequence number is equal to loop value. Sequence number starts from 1.
    seq_nmbr,base = Send_Receive.RdtSendPkt(f,sock,addr,seq_nmbr,loop_bytes,E_Prob,P_Drop,Window_Size,base,loop,image_buffer) #calls the function rdt_send to

f.close()                                               #File closed
sock.close()                                            #Socket Closed

end = time.time()                                       #Gets the End time
Elapsed_time = end - start                              #Gets the elapsed time
print ("Client: File Sent\nFile size sent to server: {0}\nTime taken in Seconds:{1}s\n".format(fileSize,Elapsed_time))
```

- Set E_prob or P_Drop value from 0 to 99.

```
'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0                                              #E_prob is the error probability and can be set from 0-99
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be set from 0-9
```

- Globals.py has the common variables for both server and client.

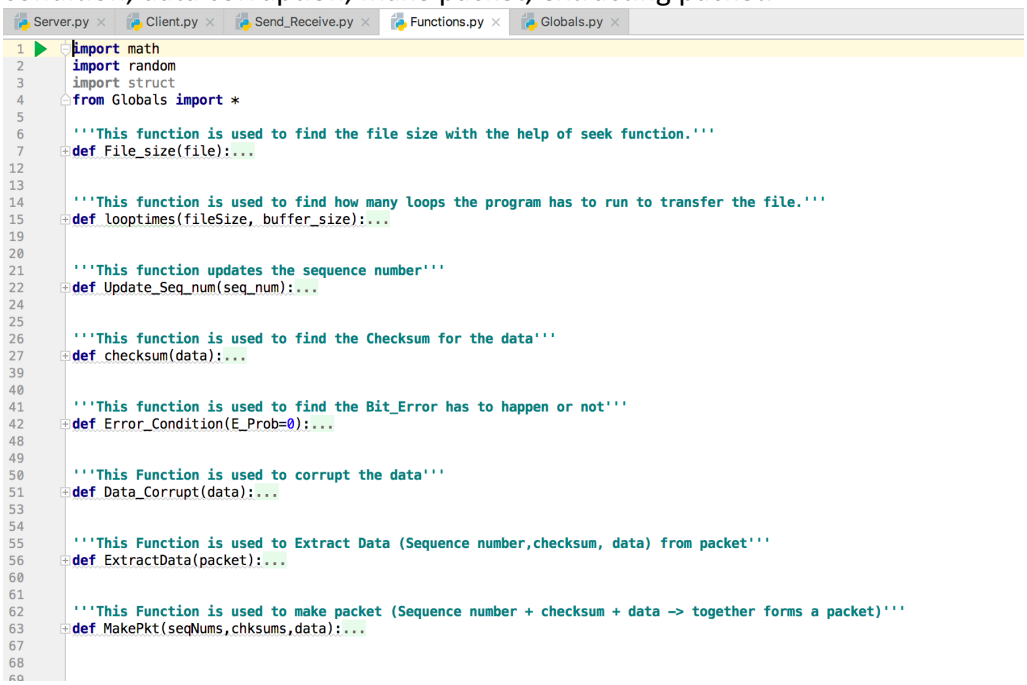


```
import socket
import struct
import time

start = time.time()                 #This is used to find the start time of the program, elapsed time can be found by end time - st

'''Importing IP address, port number, Buffersize'''
UDP_IP = "localhost"                #Localhost is the IP address of this machine
UDP_PORT = 5005                     #Port Number is assigned to 5005
buffer_size = 1024                  #Buffer_size is set to 1024. packet size is 1024 with sequence number 1 byte, checksum 2 bytes,
addr = (UDP_IP,UDP_PORT)

'''For the GBN Sliding window, set the Window Size value'''
Window_Size = 5                     #Set the window size for the Go-Back-N protocol. This window size is only for the client progra
```

- Also, we can set the Window Size in the Global.py. The window size refers to the sliding window size for the client side. The receiver side window size is always 1.

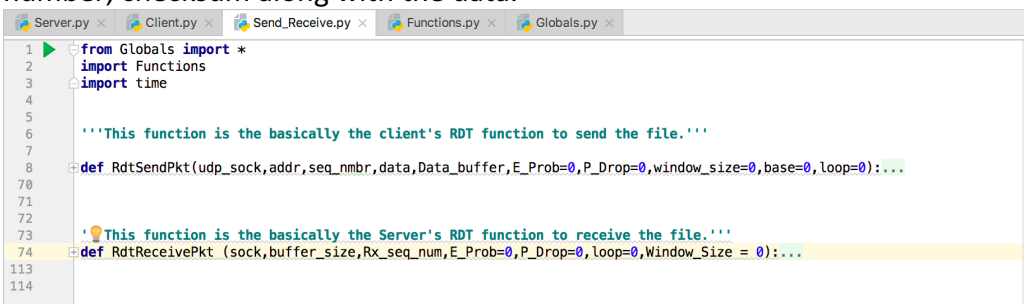
```
'''For the GBN Sliding window, set the Window Size value'''
Window_Size = 5                    #Set the window size for the Go-Back-N protocol. This window size is only for the client progra
```

- Functions.py has the functions for checksum, file size, number of time loop has to run, Bit-error condition, data corruption, make packet, extracting packet.



*Functions used in this project are simple and code is well commented.

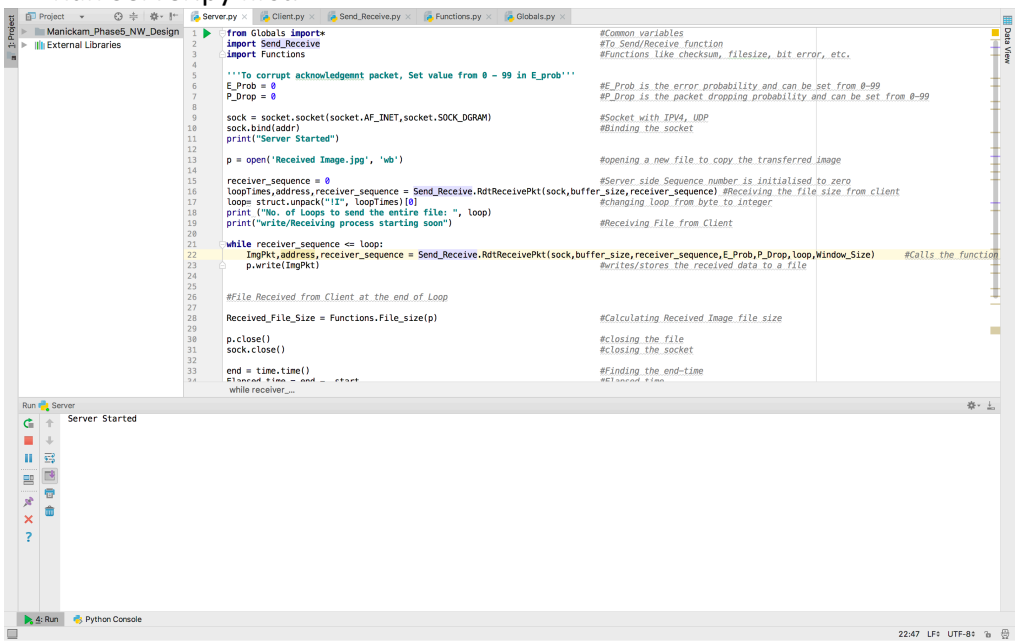
- Send_Receive.py has the RDT protocol for server and client to send/receive file with sequence number, checksum along with the data.



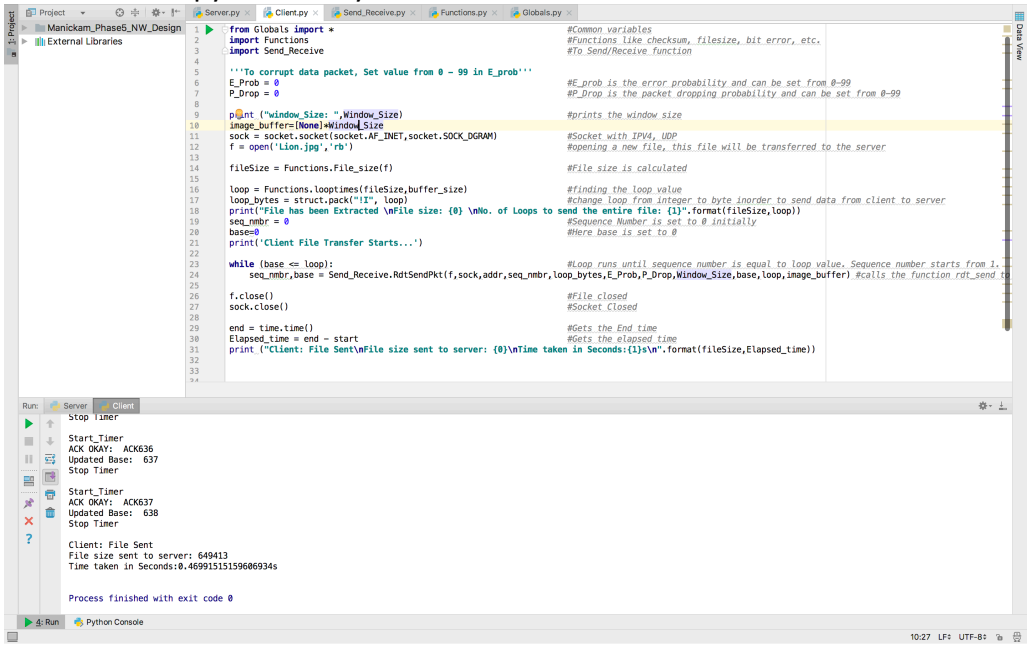
*RDT protocol is designed as per the project/protocol requirements.

PROGRAM EXECUTION:

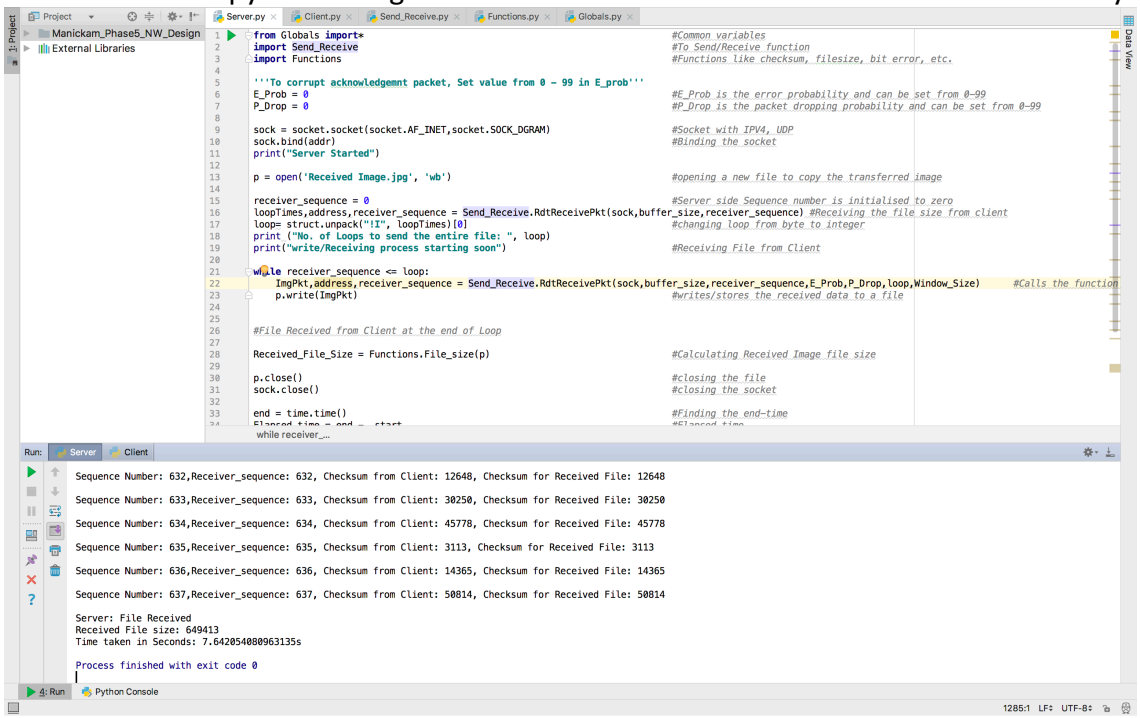
- Set window Size, (Above screenshots) I took window size as 5.
- Run Server.py first.



- Run Client.py next and you can see file transfer starts.

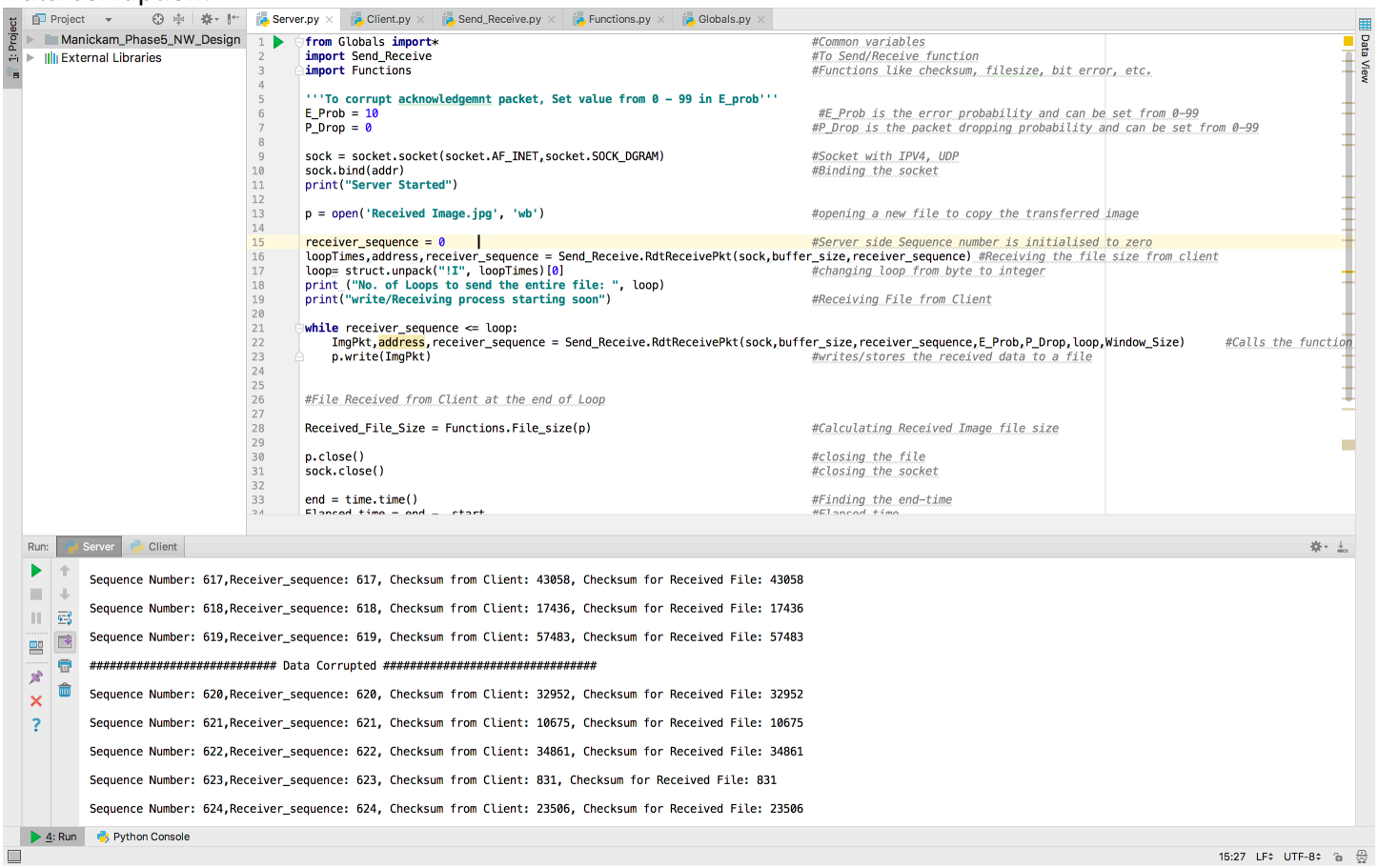


- Check Server.py console again to make sure data has been transferred successfully.



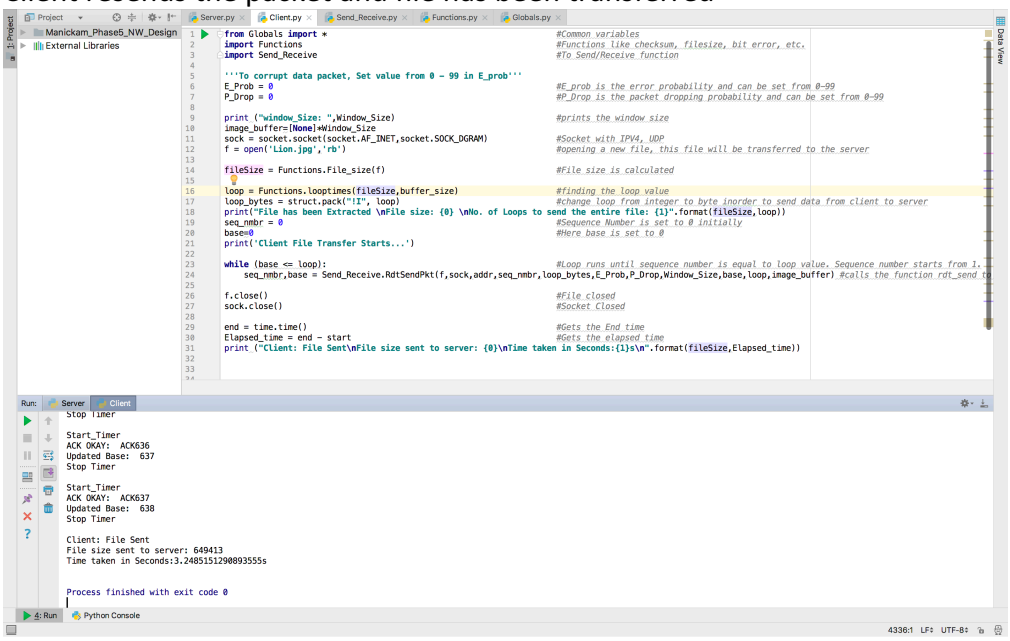
CORRUPTION (E_Prob is set to 10, that is 10% chance of corrupting the data [Sample Output]):

Data Corruption:



You can see Data Corrupted intentionally in the Server

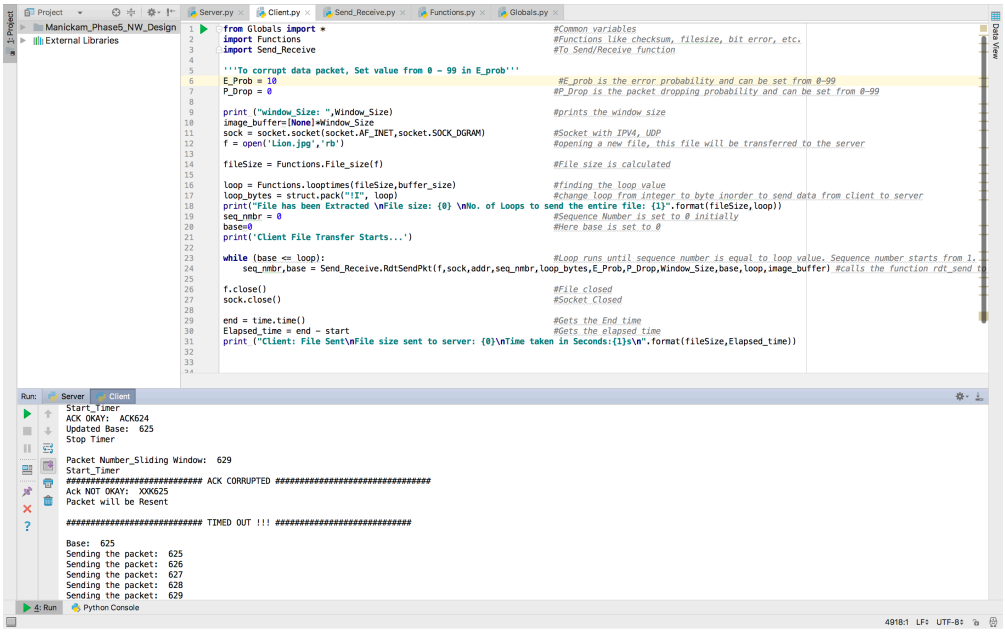
Client resends the packet and file has been transferred



```
from Globals import *                                   #Common variables
import Functions                                        #Functions like checksum, filesize, bit error, etc.
import Send_Receive                                     #To Send/Receive function

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 0                                              #E_prob is the error probability and can be set from 0-99
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be set from 0-99

print ("Window_Size: ",Window_Size)                     #prints the window size
image_buffer=[None]*Window_Size
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)  #Socket with IPV4, UDP
f = open('Lion.jpg','rb')                               #opening a new file, this file will be transferred to the server

fileSize = Functions.File_size(f)                       #File size is calculated

loop = Functions.looptimes(fileSize,buffer_size)        #finding the loop value
loop_bytes = struct.pack("!I", loop)                    #change loop from integer to byte inorder to send data from client to server
print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}".format(fileSize,loop))
seq_nmbr = 0                                            #Sequence Number is set to 0 initially
base=0                                                  #Here base is set to 0
print('Client File Transfer Starts...')

while (base <= loop):                                   #Loop runs until sequence number is equal to loop value. Sequence number starts from 1.
    seq_nmbr,base = Send_Receive.RdtSendPkt(f,sock,addr,seq_nmbr,loop_bytes,E_Prob,P_Drop,Window_Size,base,loop,image_buffer) #calls the function rdt_send to

f.close()                                               #File closed
sock.close()                                            #Socket Closed

end = time.time()                                       #Gets the End time
Elapsed_time = end - start                              #Gets the elapsed time
print ("Client: File Sent\nFile size sent to server: {0}\nTime taken in Seconds:{1}s\n".format(fileSize,Elapsed_time))
```

```
Stop Timer

Start_Timer
ACK OKAY:  ACK636
Updated Base:  637
Stop Timer

Start_Timer
ACK OKAY:  ACK637
Updated Base:  638
Stop Timer

Client: File Sent
File size sent to server: 649413
Time taken in Seconds:3.2485151290893555s


Process finished with exit code 0
```

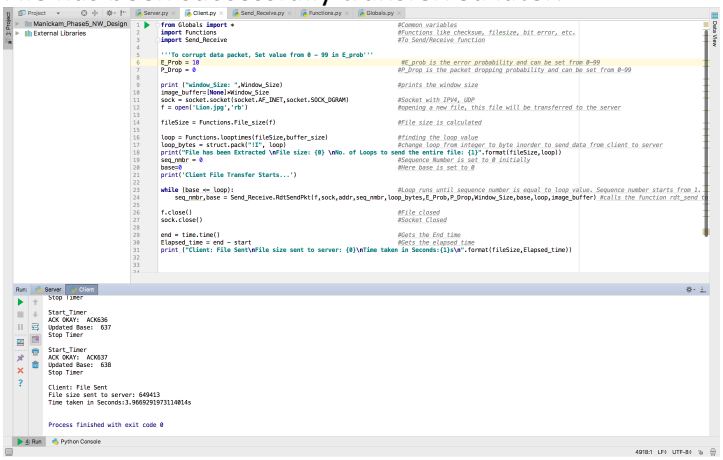
ACK CORRUPTION:



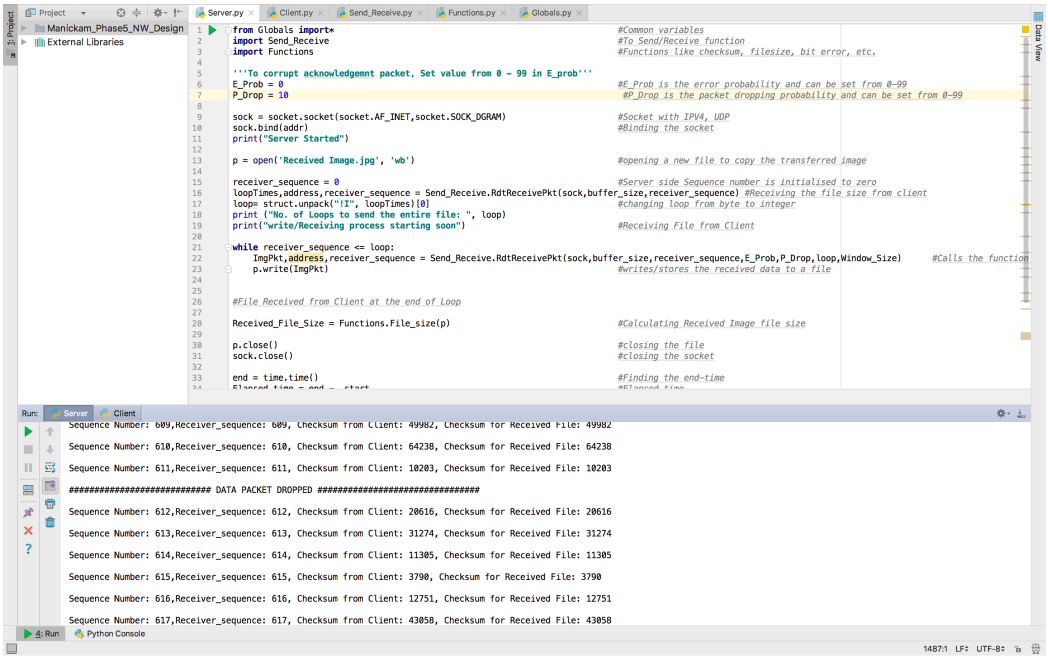
```
from Globals import *                                   #Common variables
import Functions                                        #Functions like checksum, filesize, bit error, etc.
import Send_Receive                                     #To Send/Receive function

'''To corrupt data packet, Set value from 0 - 99 in E_prob'''
E_Prob = 10                                             #E_prob is the error probability and can be set from 0-99
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be set from 0-99

print ("Window_Size: ",Window_Size)                     #prints the window size
image_buffer=[None]*Window_Size
sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)  #Socket with IPV4, UDP
f = open('Lion.jpg','rb')                               #opening a new file, this file will be transferred to the server

fileSize = Functions.File_size(f)                       #File size is calculated

loop = Functions.looptimes(fileSize,buffer_size)        #finding the loop value
loop_bytes = struct.pack("!I", loop)                    #change loop from integer to byte inorder to send data from client to server
print("File has been Extracted \nFile size: {0} \nNo. of Loops to send the entire file: {1}".format(fileSize,loop))
seq_nmbr = 0                                            #Sequence Number is set to 0 initially
base=0                                                  #Here base is set to 0
print('Client File Transfer Starts...')

while (base <= loop):                                   #Loop runs until sequence number is equal to loop value. Sequence number starts from 1.
    seq_nmbr,base = Send_Receive.RdtSendPkt(f,sock,addr,seq_nmbr,loop_bytes,E_Prob,P_Drop,Window_Size,base,loop,image_buffer) #calls the function rdt_send to

f.close()                                               #File closed
sock.close()                                            #Socket Closed

end = time.time()                                       #Gets the End time
Elapsed_time = end - start                              #Gets the elapsed time
print ("Client: File Sent\nFile size sent to server: {0}\nTime taken in Seconds:{1}s\n".format(fileSize,Elapsed_time))
```

```
Start_Timer
ACK OKAY:  ACK624
Updated Base:  625
Stop Timer

Packet Number_Sliding Window:  629
Start_Timer
############################ ACK CORRUPTED ################################
Ack NOT OKAY:  XXX625
Packet will be Resent

############################ TIMED OUT !!! ############################

Base:  625
Sending the packet:  625
Sending the packet:  626
Sending the packet:  627
Sending the packet:  628
Sending the packet:  629
```

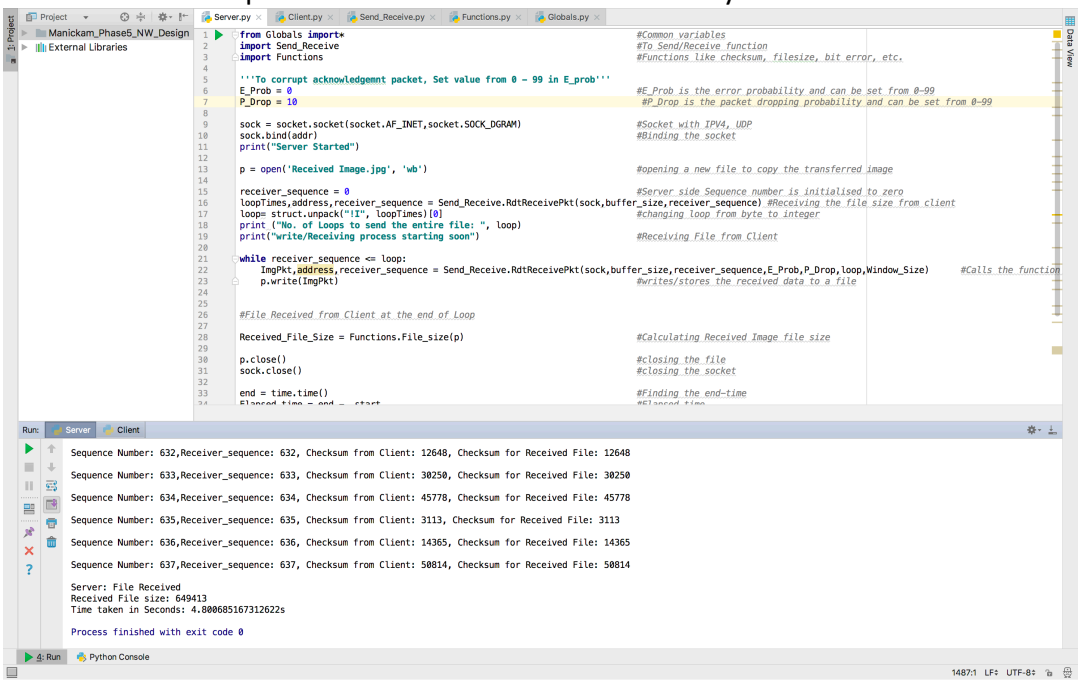
You can see ACK Corrupted intentionally in the client (above screenshot)
File has been successfully transferred later.



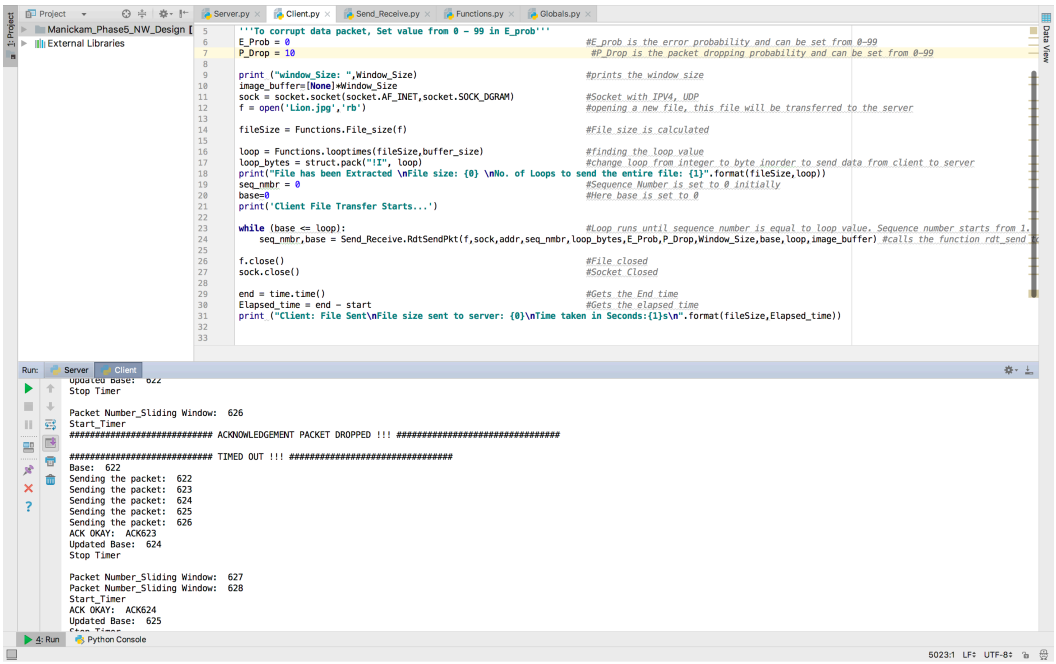
PACKET LOSS (P_Drop is set to 10, that is 10% chance of dropping the packet [Sample Output]):
Data Packet Loss:



You can see Data packet lost intentionally in the Server
Client resends the packet and data transfers successfully later in the server.



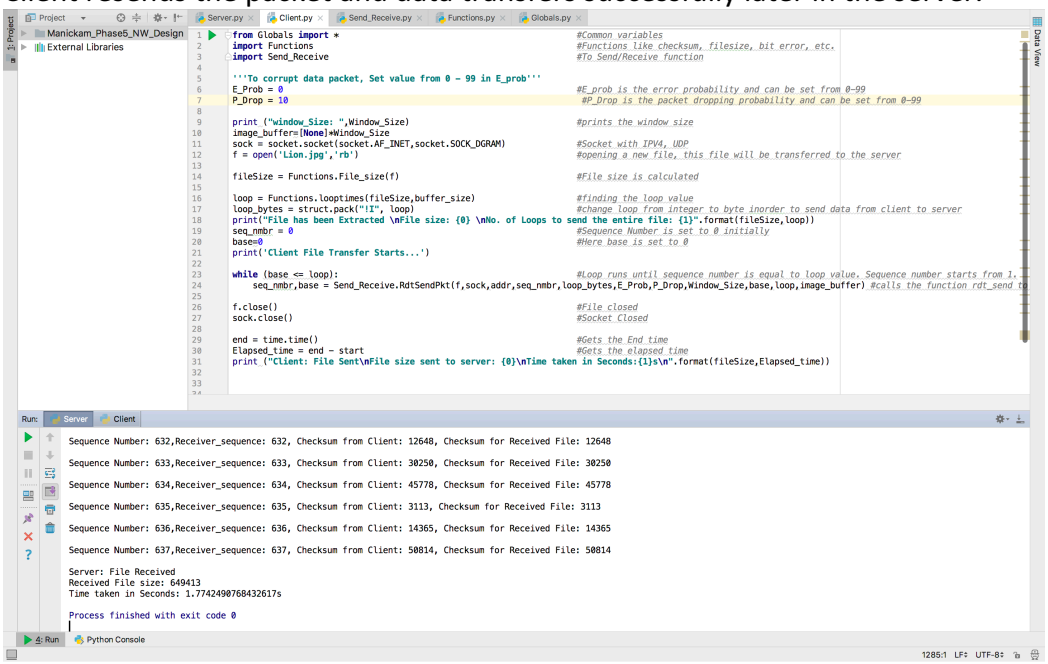
ACK Packet Loss:



You can see ACK packet dropped intentionally in the client, but here cumulative ack works.

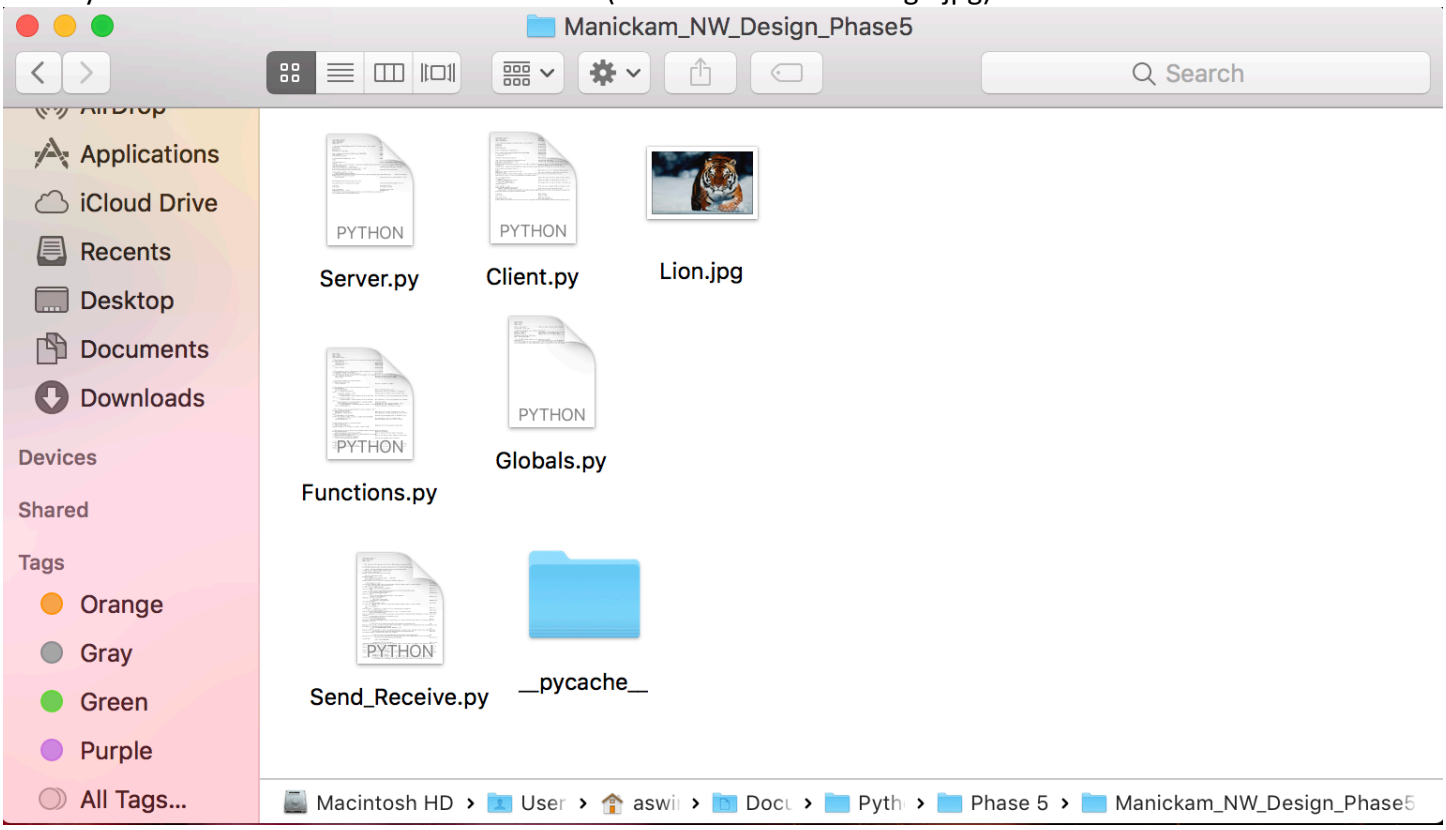
You can see Ack packet lost intentionally in the Server

Client resends the packet and data transfers successfully later in the server.

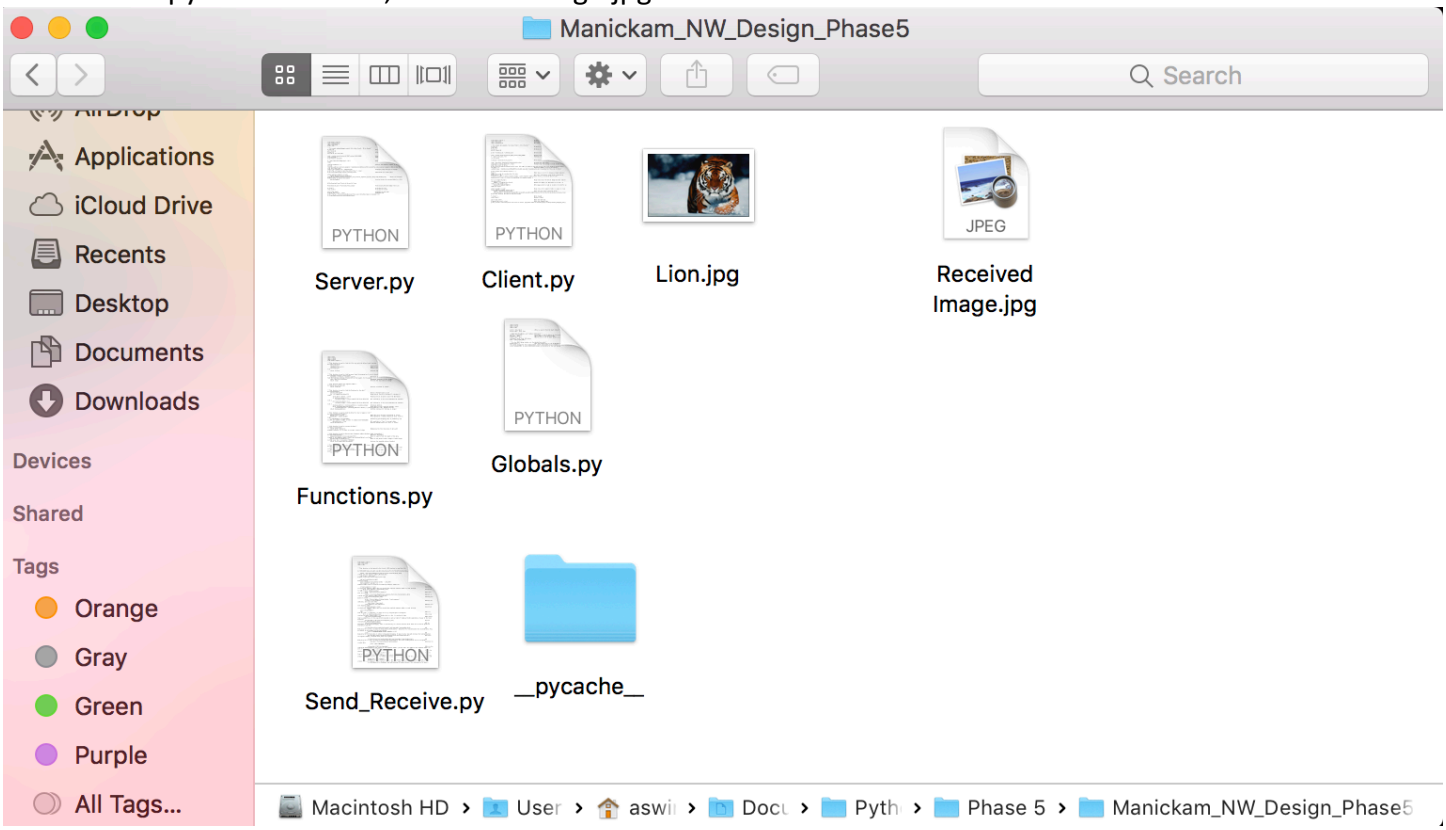


FILE TRANSFER OUTPUT:

Initially the folder looks like below screenshot (Without Received Image.jpg)



After Server.py executed first, Received Image.jpg is created



After Running the Program, New file called Received Image.jpg will be in the folder which shows successful File transfer

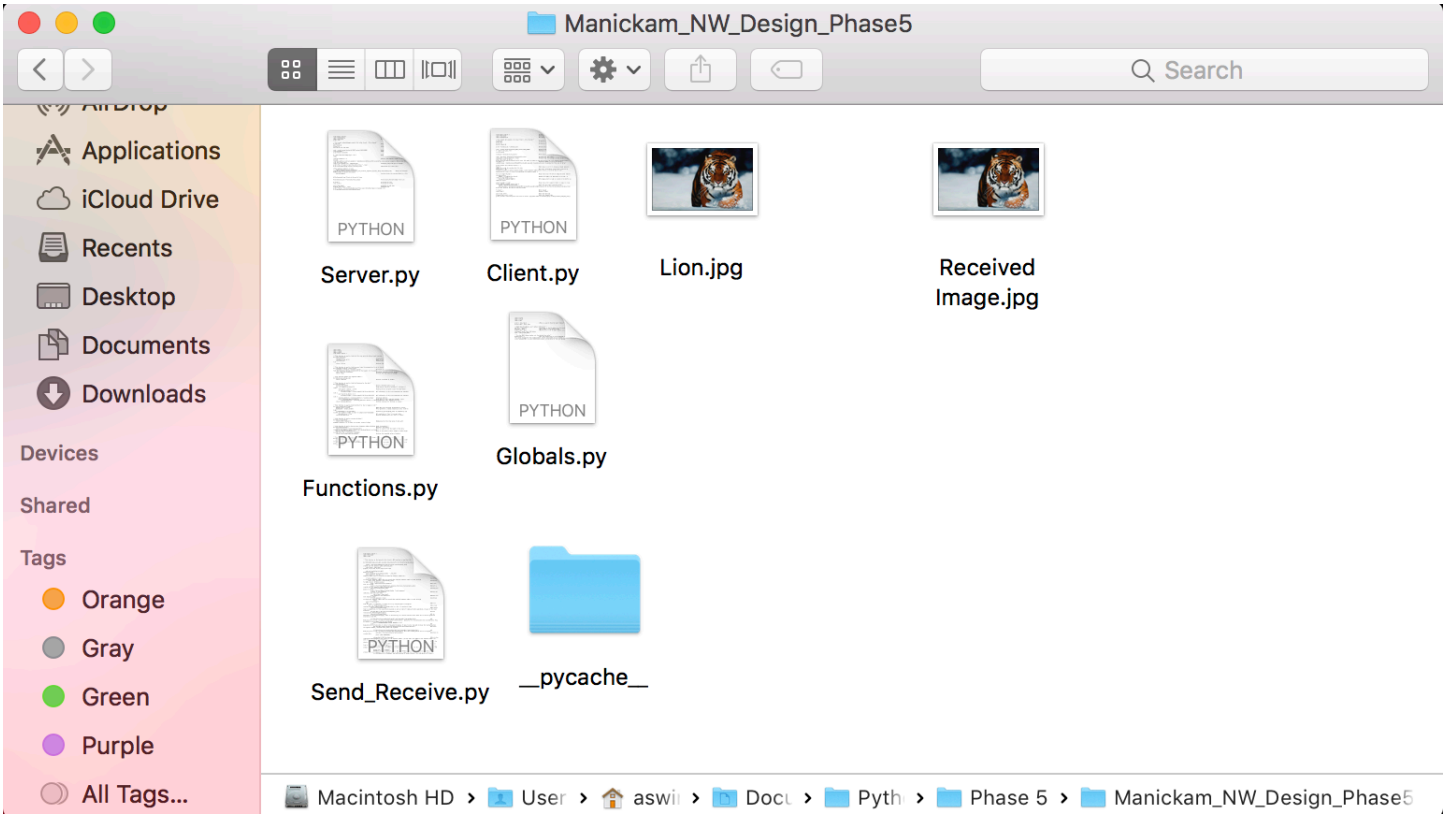
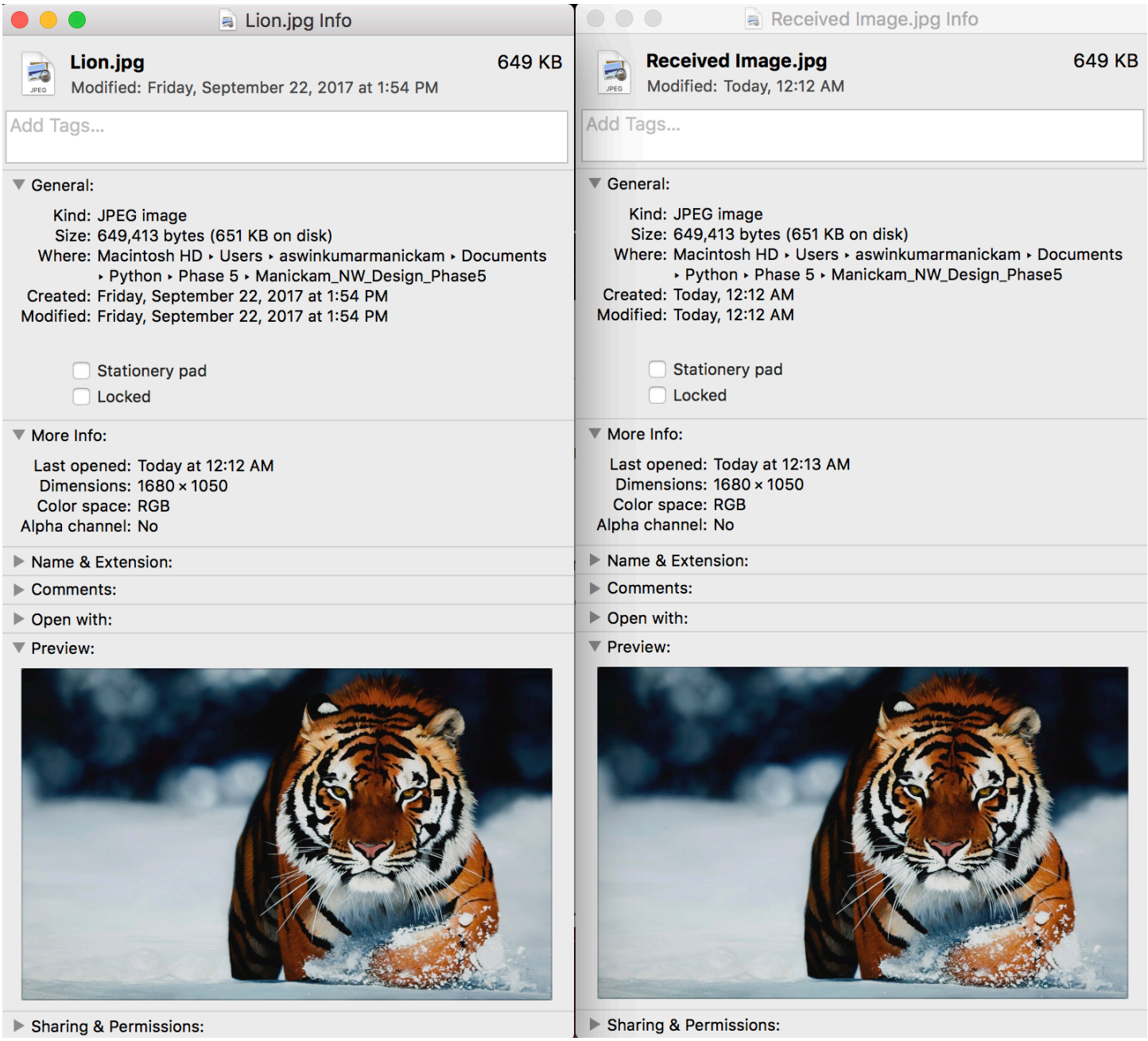


Image file and Received Image is of same size (Reliable Data Transfer)



Result:

Thus Go-Back-N protocol over an unreliable UDP channel has been implemented.