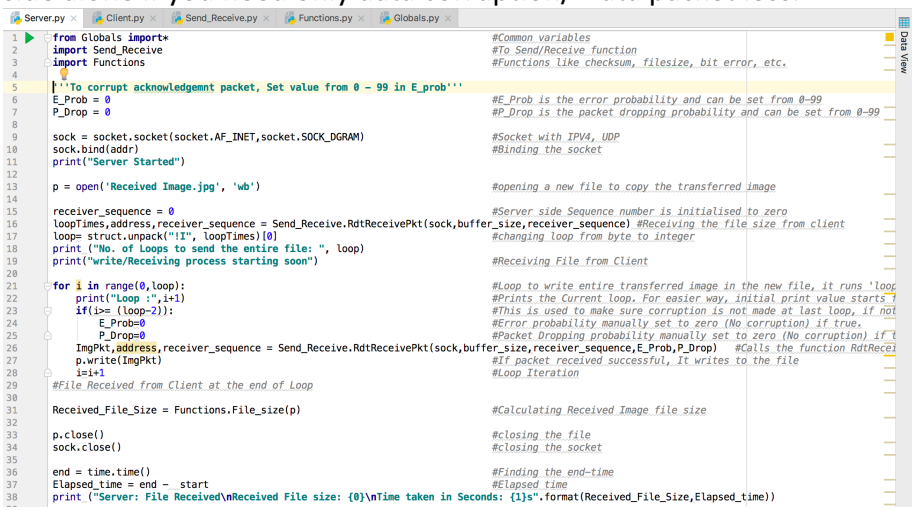
EECE 5830: NETWORK DESIGN

PROJECT PHASE 4: IMPLEMENT RDT 3.0 OVER AN UNRELIABLE UDP CHANNEL WITH BIT-ERRORS AND LOSS

AUTHOR: ASWIN KUMAR MANICKAM, 01624359.

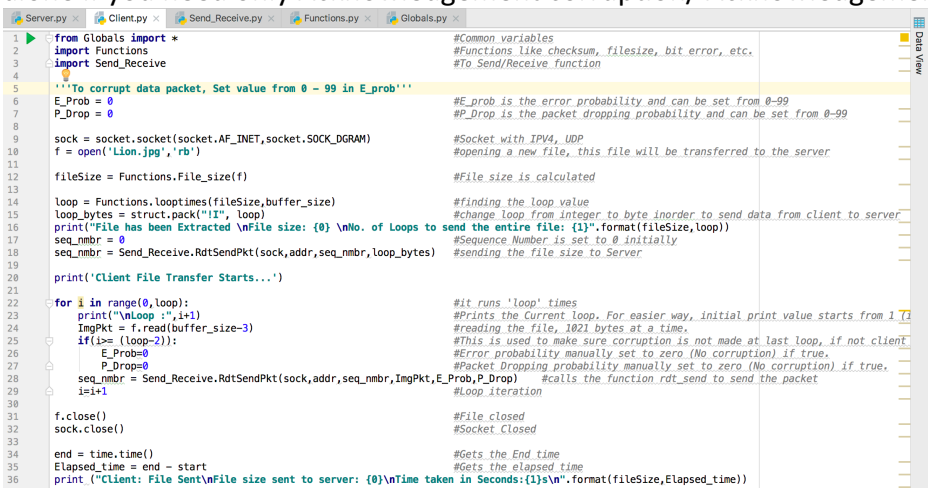
PROGRAM:

- 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

for i in range(0,loop):                                 #Loop to write entire transferred image in the new file, it runs 'loop
    print("Loop :",i+1)                                 #Prints the Current loop. For easier way, initial print value starts f
    if(i>= (loop-2)):                                   #This is used to make sure corruption is not made at last loop, if not
        E_Prob=0                                        #Error probability manually set to zero (No corruption) if true.
        P_Drop=0                                        #Packet Dropping probability manually set to zero (No corruption) if t
    ImgPkt,address,receiver_sequence = Send_Receive.RdtReceivePkt(sock,buffer_size,receiver_sequence,E_Prob,P_Drop)   #Calls the function RdtRecei
    p.write(ImgPkt)                                     #If packet received successful, It writes to the file
    i=i+1                                               #Loop Iteration
#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))
```

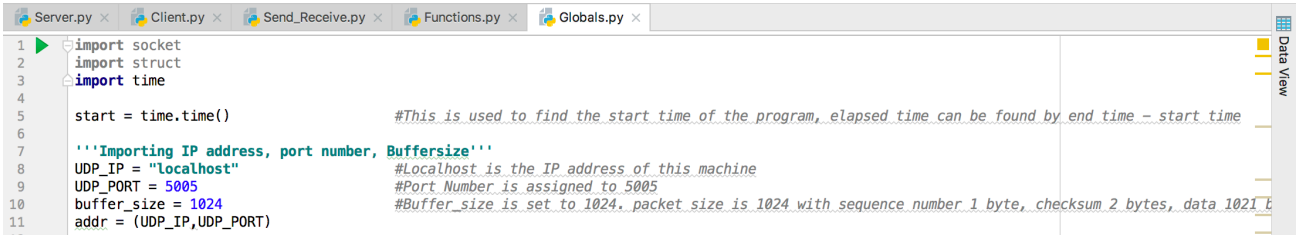
- Client.py has the main program for the client to send a file, Set E_prob or P_Drop in client side alone if you need only Acknowledgement corruption/ Acknowledgement packet loss.



```
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

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
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)   #sending the file size to Server

print('Client File Transfer Starts...')

for i in range(0,loop):                                 #it runs 'loop' times
    print("\nLoop :",i+1)                                #Prints the Current loop. For easier way, initial print value starts from 1 (i
    ImgPkt = f.read(buffer_size-3)                      #reading the file, 1021 bytes at a time.
    if(i>= (loop-2)):                                   #This is used to make sure corruption is not made at last loop, if not client
        E_Prob=0                                        #Error probability manually set to zero (No corruption) if true.
        P_Drop=0                                        #Packet Dropping probability manually set to zero (No corruption) if true.
    seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,ImgPkt,E_Prob,P_Drop)    #calls the function rdt_send to send the packet
    i=i+1                                               #Loop iteration

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.

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

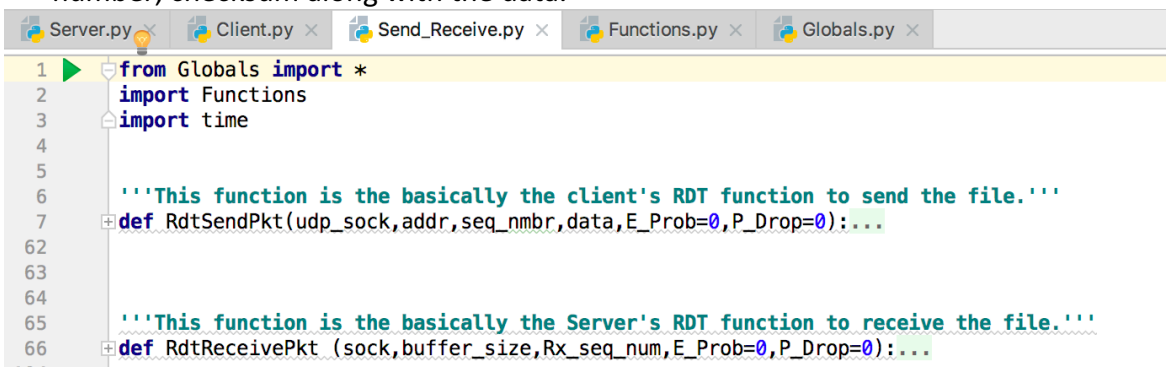


```
import socket
import struct
import time

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

'''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, data 1021 b
addr = (UDP_IP,UDP_PORT)
```

- 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.

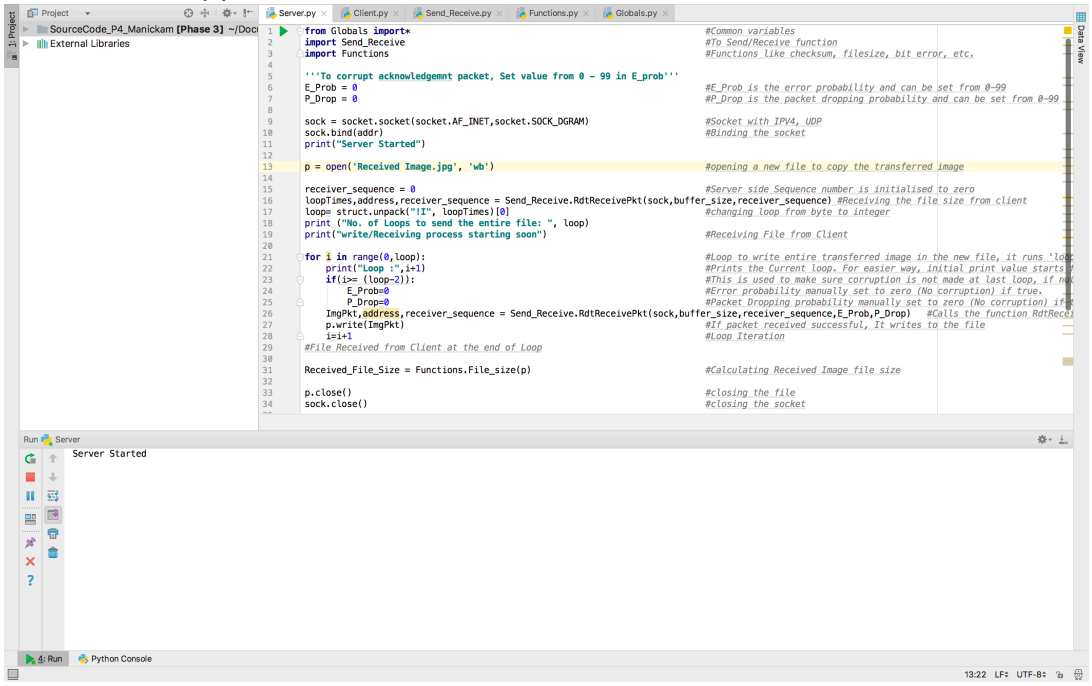
- Send_Receive.py has the RDT protocol for server and client to send/receive file with sequence number, checksum along with the data.



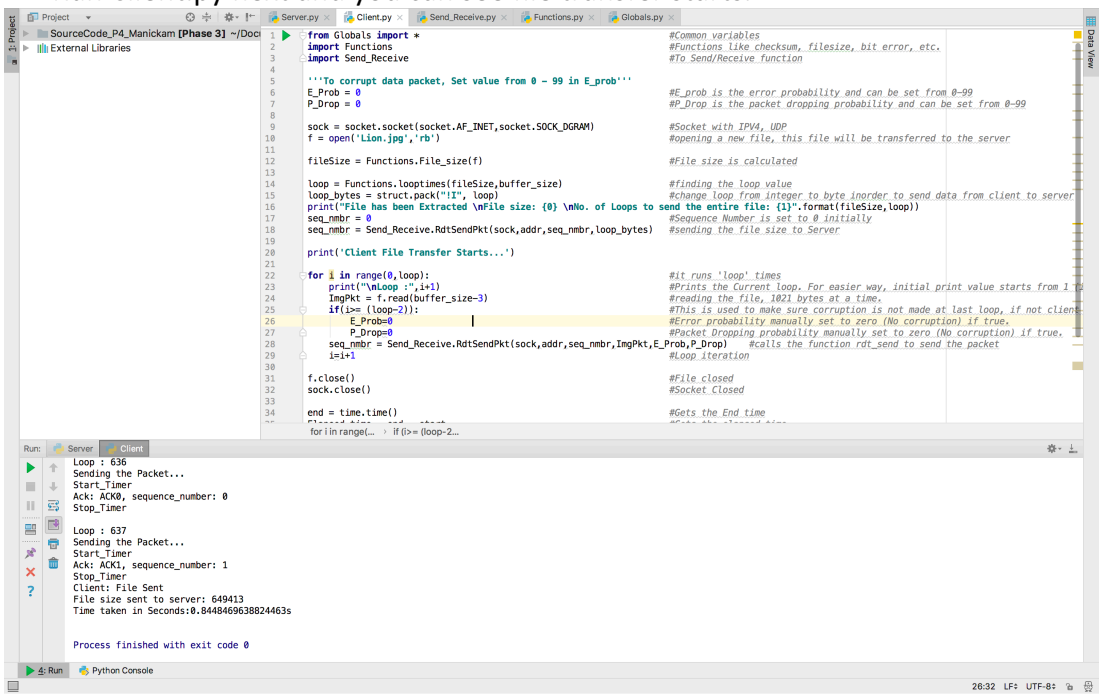
*RDT protocol is designed as per the project/protocol requirements.

PROGRAM EXECUTION:

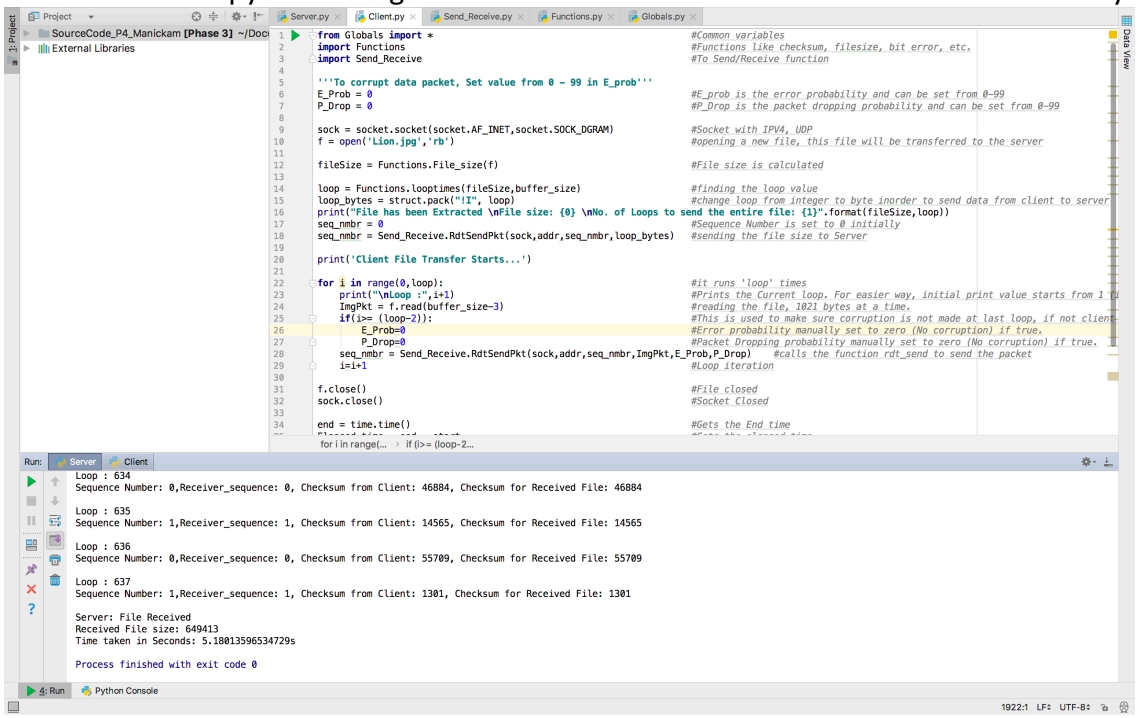
- Run Server.py first.



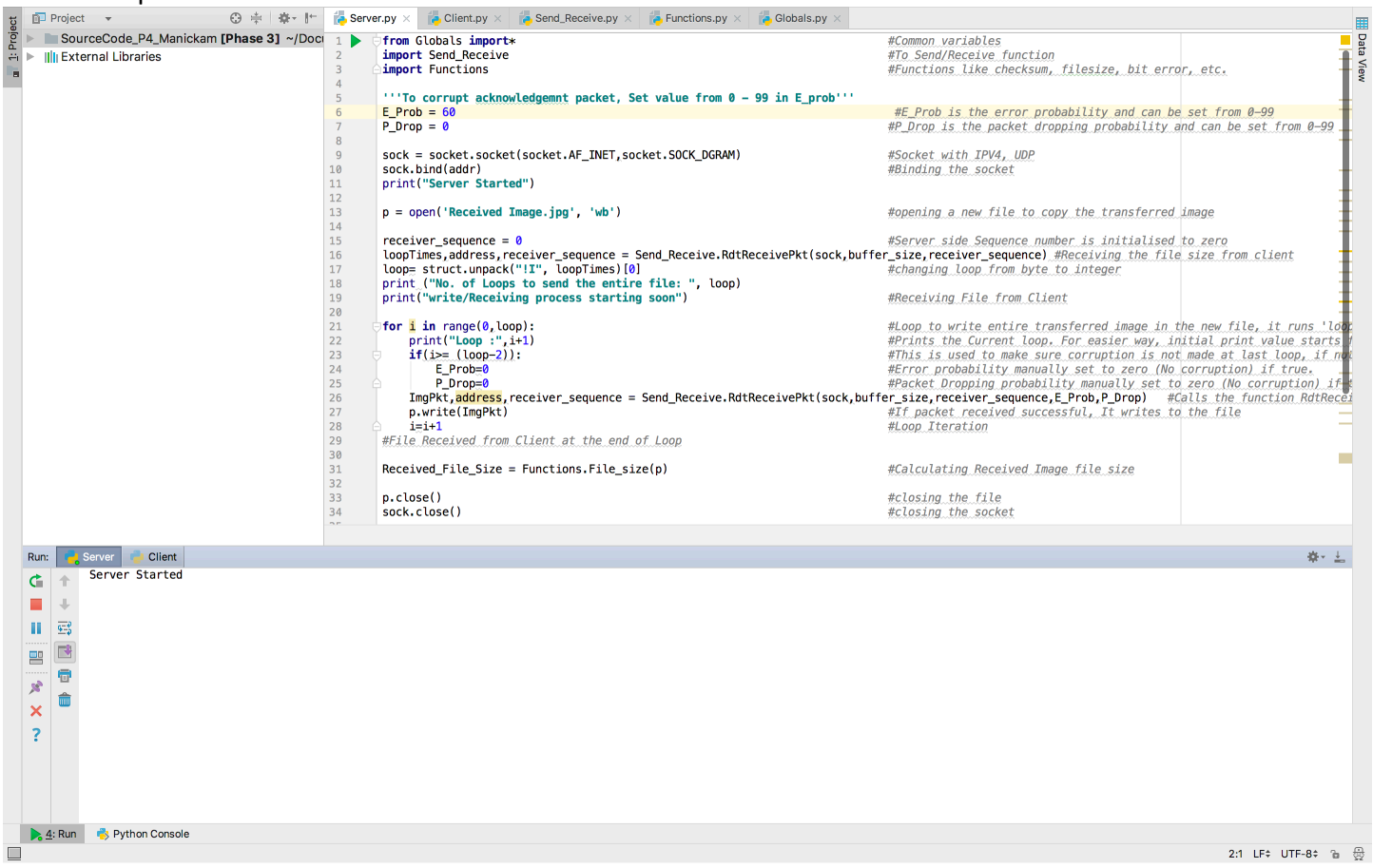
- Run Client.py next and you can see file transfer starts.

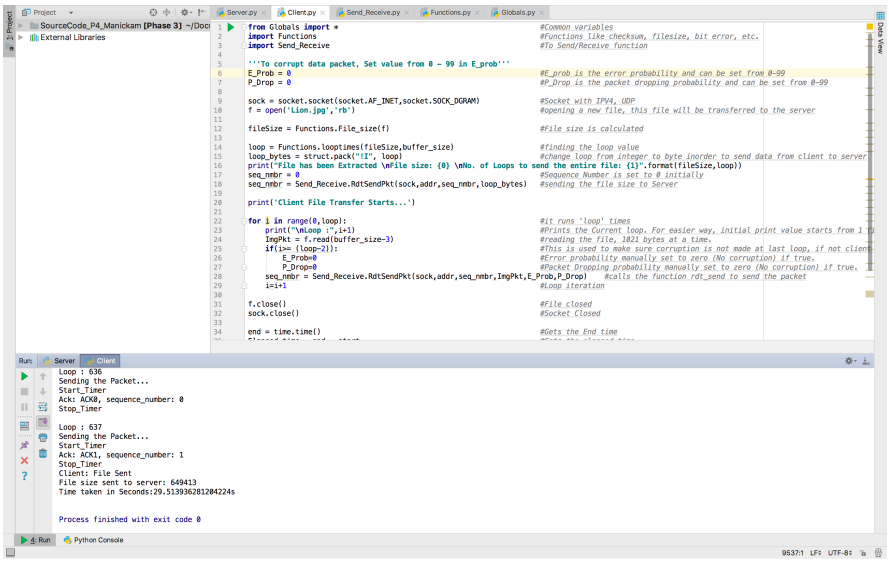


- Check Server.py console again to make sure data has been transferred successfully.



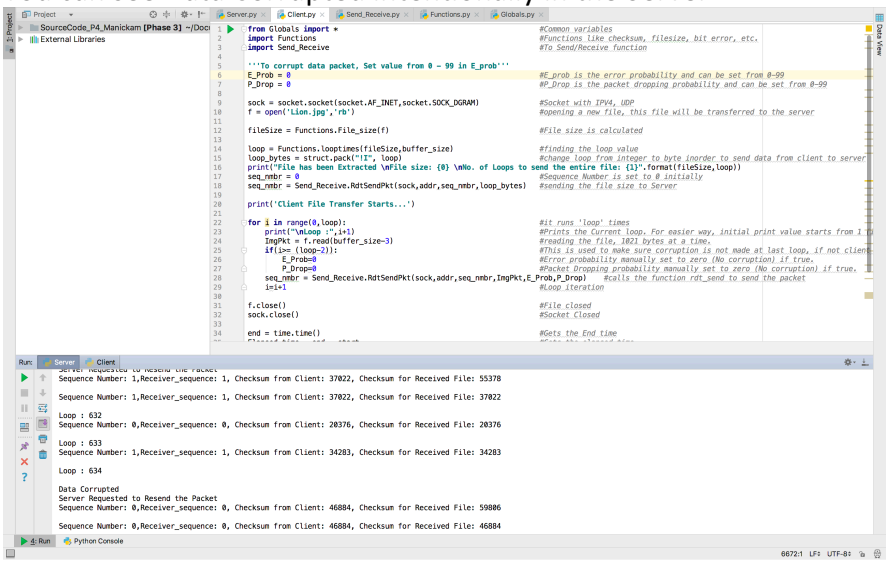
CORRUPTION (E_Prob is set to 60, that is 60% chance of corrupting the data [Sample Output]):
Data Corruption:





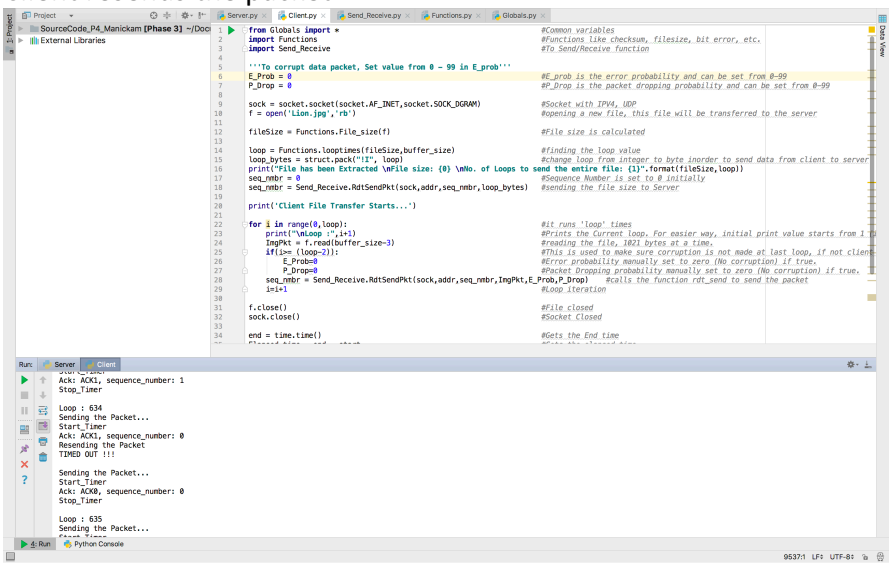
```
Loop : 636
Sending the Packet...
Start_Timer
Ack: ACK0, sequence_number: 0
Stop_Timer

Loop : 637
Sending the Packet...
Start_Timer
Ack: ACK1, sequence_number: 1
Stop_Timer
Client: File Sent
File size sent to server: 649413
Time taken in Seconds:29.513936281204224s


Process finished with exit code 0
```

You can see Data Corrupted intentionally in the Server



```
Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 37022, Checksum for Received File: 55378

Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 37022, Checksum for Received File: 37022

Loop : 632
Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 20376, Checksum for Received File: 20376

Loop : 633
Sequence Number: 1,Receiver_sequence: 1, Checksum from Client: 34283, Checksum for Received File: 34283

Loop : 634

Data Corrupted
Server Requested to Resend the Packet
Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 46884, Checksum for Received File: 59086

Sequence Number: 0,Receiver_sequence: 0, Checksum from Client: 46884, Checksum for Received File: 46884
```

Client resends the packet



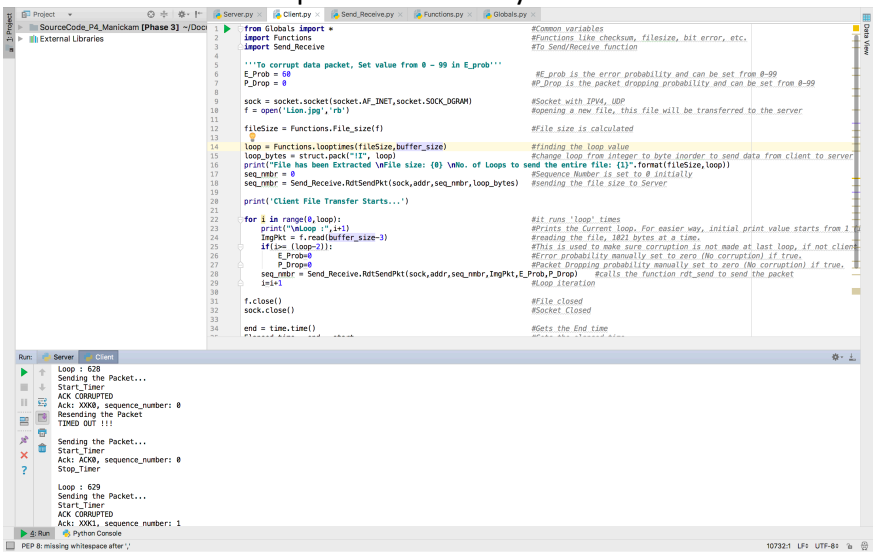
```
Ack: ACK1, sequence_number: 1
Stop_Timer

Loop : 634
Sending the Packet...
Start_Timer
Ack: ACK1, sequence_number: 0
Resending the Packet
TIMED OUT !!!

Sending the Packet...
Start_Timer
Ack: ACK0, sequence_number: 0
Stop_Timer

Loop : 635
Sending the Packet...
```

ACK CORRUPTION:

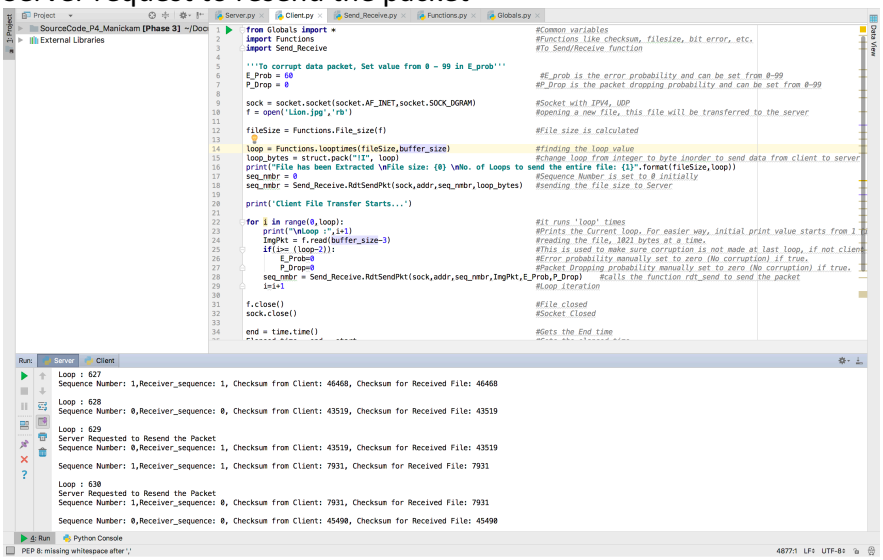
Server.py | Client.py | Send_Receive.py | Functions.py | Globals.py

```
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 = 60                                              #E_prob is the error probability and can be set fro
P_Drop = 0                                              #P_Drop is the packet dropping probability and can be

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 t

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 da
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
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)   #sending the file size to Server

print('Client File Transfer Starts...')

for i in range(0,loop):                                 #it runs 'loop' times
    print("\nLoop :",i+1)                                #Prints the Current loop. For easier way, initial pr
    ImgPkt = f.read(buffer_size-3)                      #reading the file, 1021 bytes at a time.
    if(i>= (loop-2)):                                   #This is used to make sure corruption is not made at
        E_Prob=0                                        #Error probability manually set to zero (No corruptio
        P_Drop=0                                        #Packet Dropping probability manually set to zero (No
    seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,ImgPkt,E_Prob,P_Drop)     #calls the function rdt_send to send
    i=i+1                                               #Loop iteration

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))
```

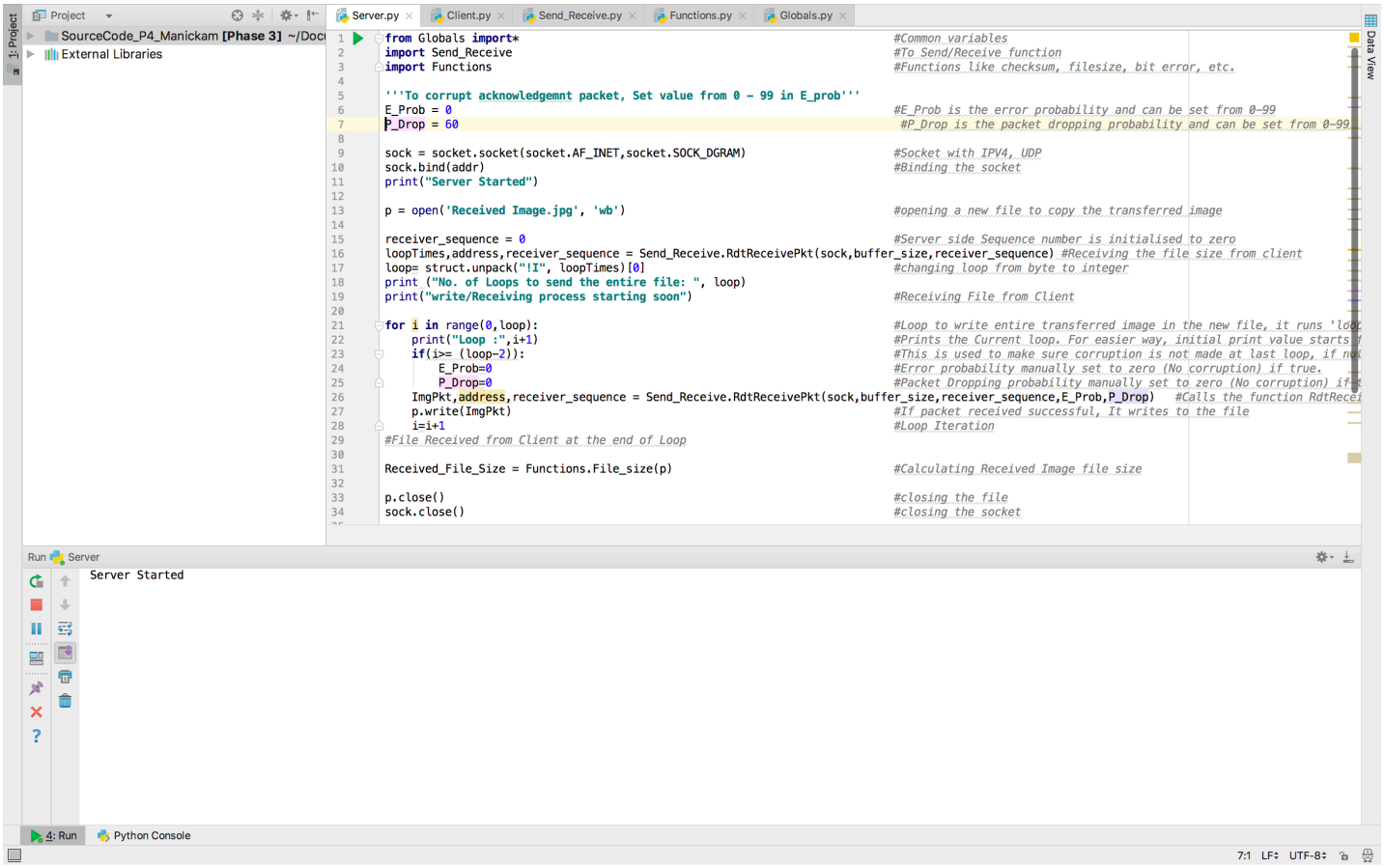
You can see ACK Corrupted intentionally in the client

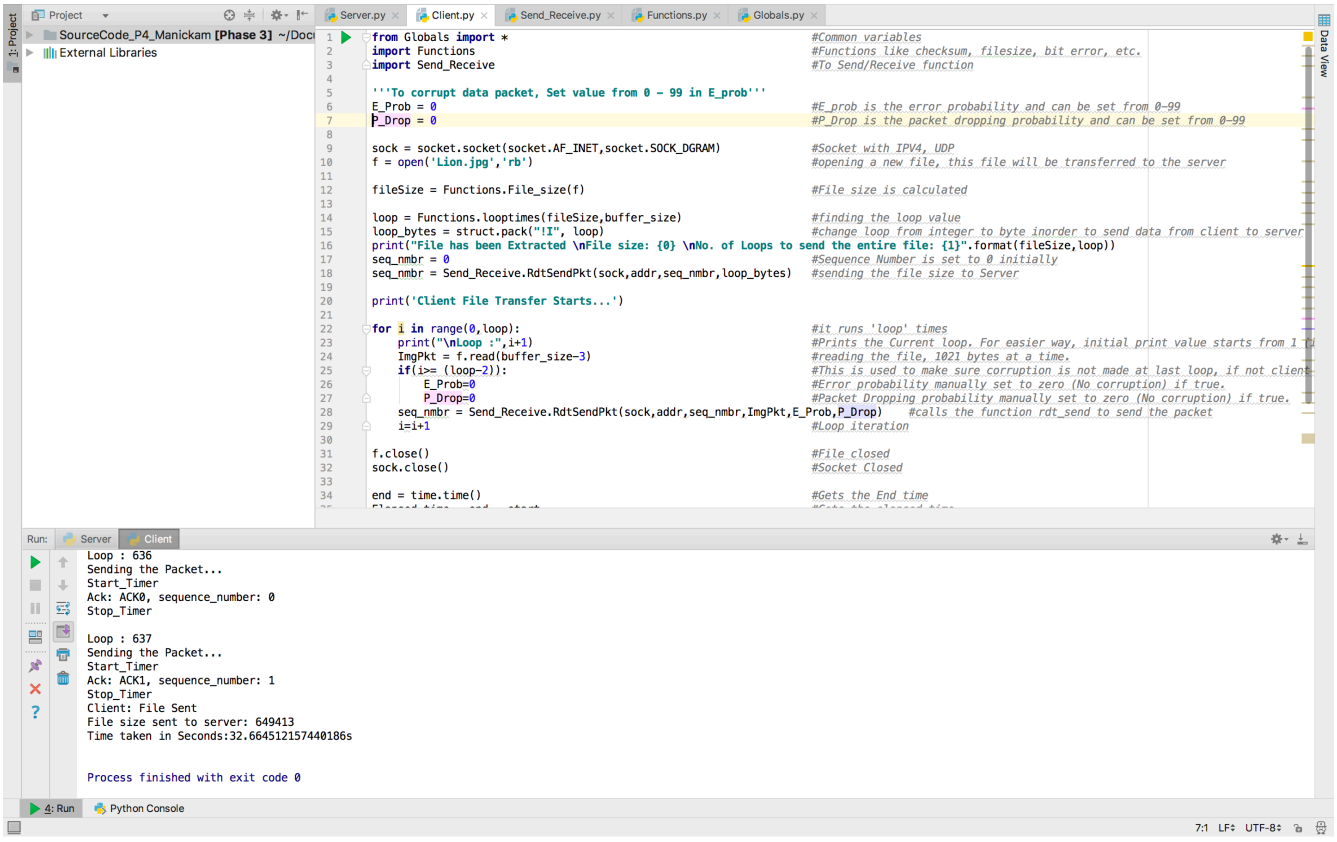


Server request to resend the packet

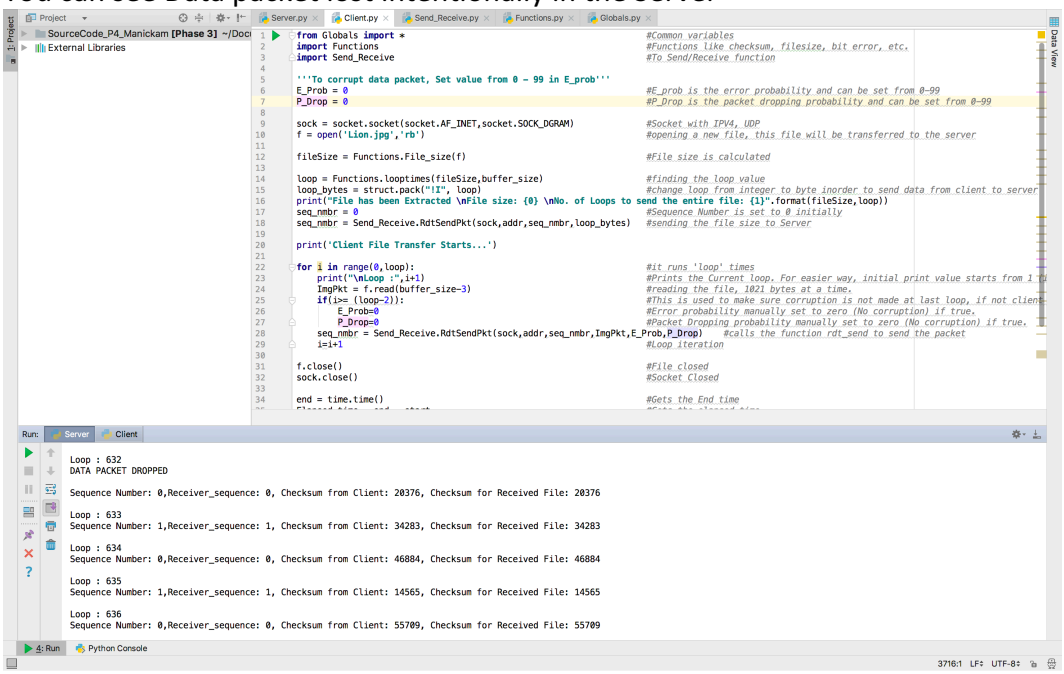


PACKET LOSS (P_Drop is set to 60, that is 60% chance of dropping the packet [Sample Output]):
Data Packet Loss:

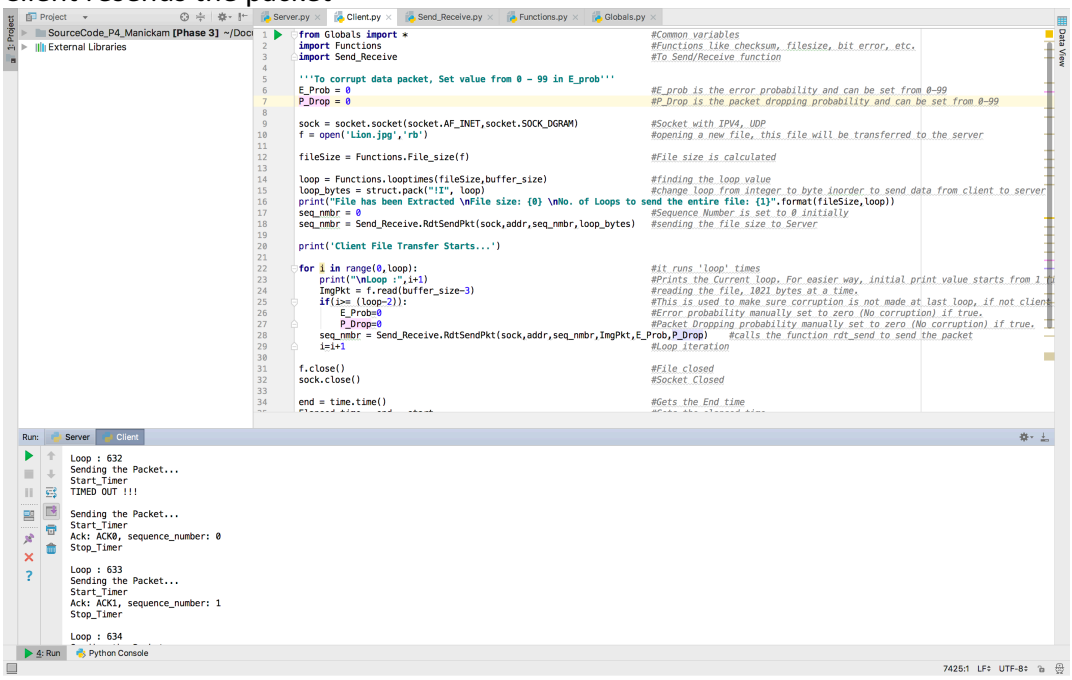




You can see Data packet lost intentionally in the Server



Client resends the packet

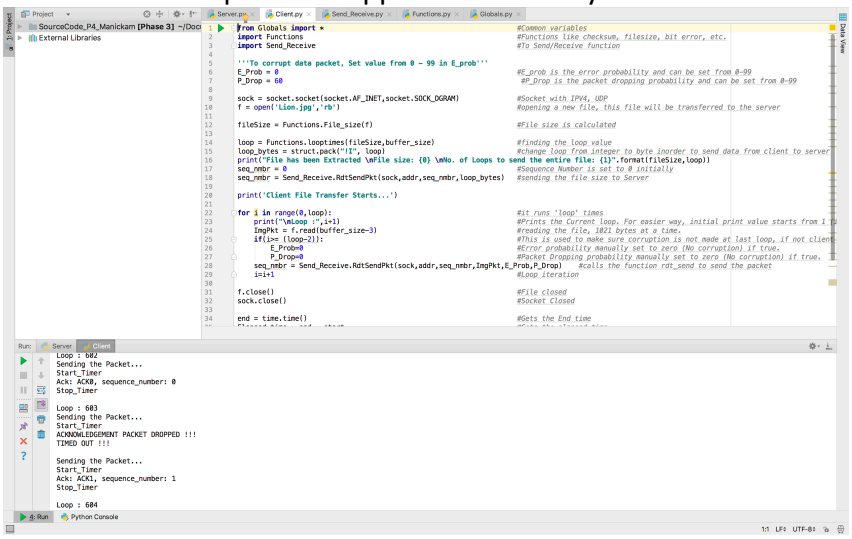


ACK Packet Loss:

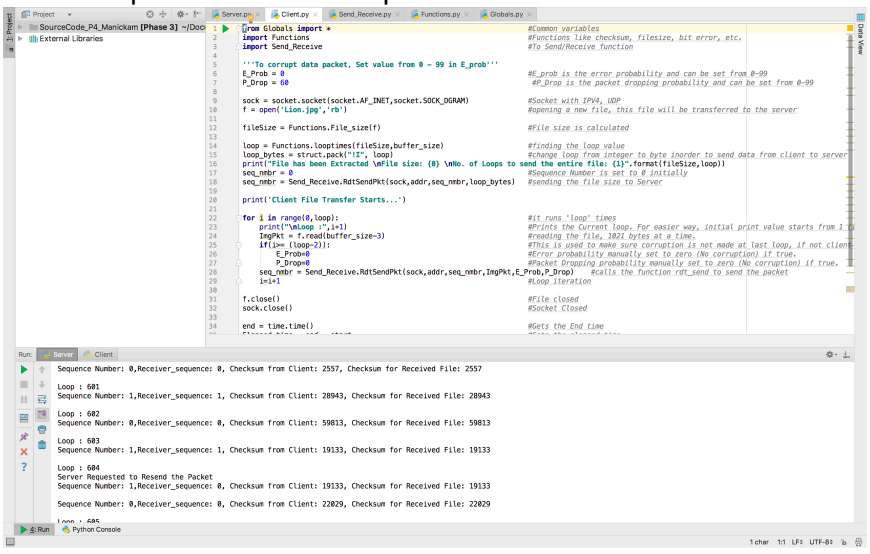
Server.py | Client.py | Send_Receive.py | Functions.py | Globals.py

```python
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
P_Drop = 60                                              #P_Drop is the packet dropping probability and can

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 t

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 da
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
seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,loop_bytes)   #sending the file size to Server

print('Client File Transfer Starts...')

for i in range(0,loop):                                 #it runs 'loop' times
    print("\nLoop :",i+1)                                #Prints the Current loop. For easier way, initial pr
    ImgPkt = f.read(buffer_size-3)                      #reading the file, 1021 bytes at a time.
    if(i>= (loop-2)):                                   #This is used to make sure corruption is not made at
        E_Prob=0                                        #Error probability manually set to zero (No corruptio
        P_Drop=0                                        #Packet Dropping probability manually set to zero (N
    seq_nmbr = Send_Receive.RdtSendPkt(sock,addr,seq_nmbr,ImgPkt,E_Prob,P_Drop)    #calls the function rdt_send to send
    i=i+1                                               #Loop iteration

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))
```

You can see ACK packet dropped intentionally in the client



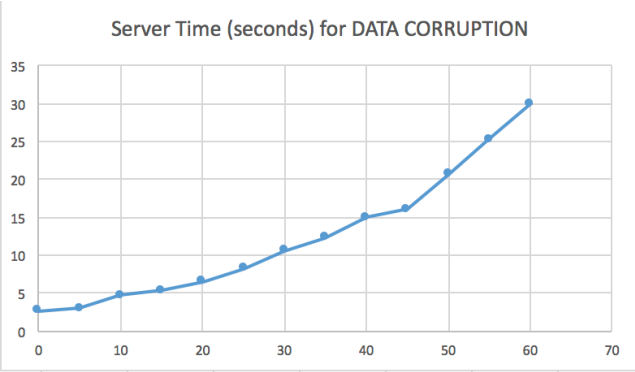
Server request to resend the packet

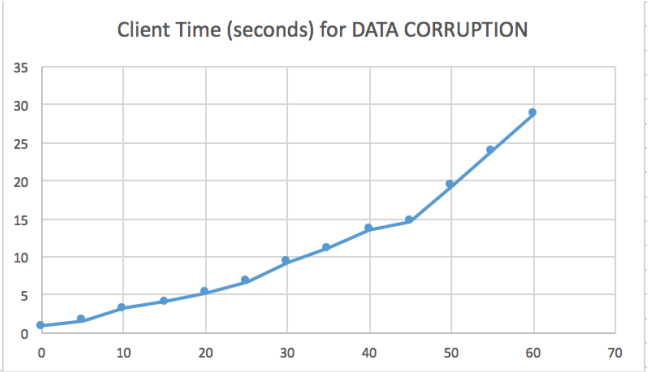


CHARTS FOR COMPLETION TIME FOR TRANSFERRING THE SAME FILE 0% LOSS/ERROR TO 60% LOSS/ERROR IN INCREMENTS OF 5%:

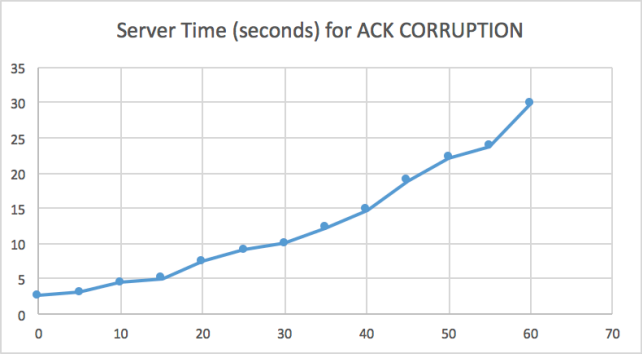
GRAPH -> X-AXIS: PROBABILITY 0 TO 60 %, Y-AXIS: TIME IN SECONDS

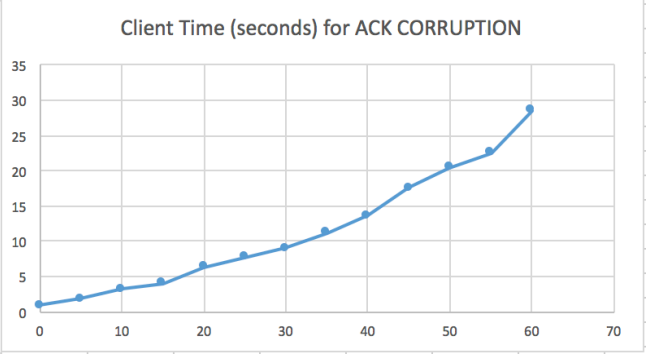
DATA CORRUPTION:



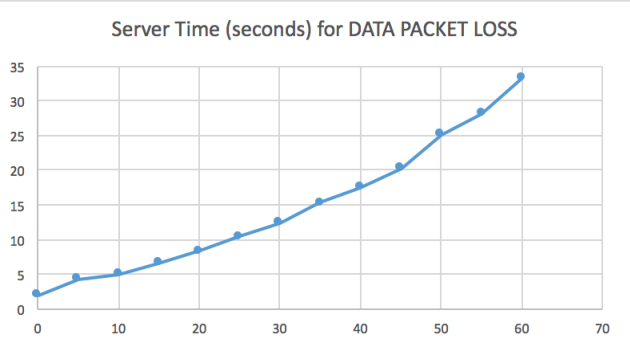


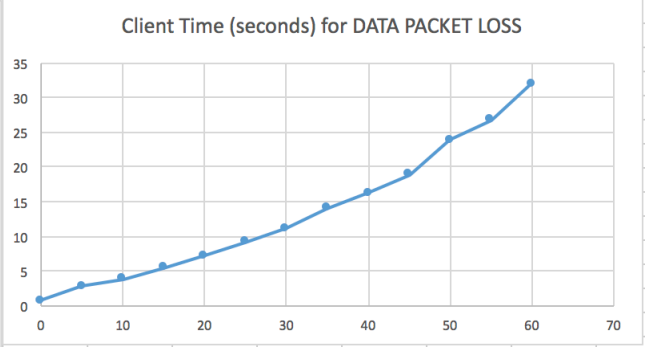
ACKNOWLEDGEMENT CORRUPTION:



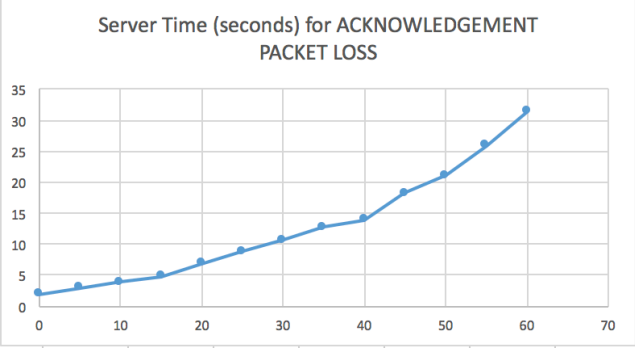


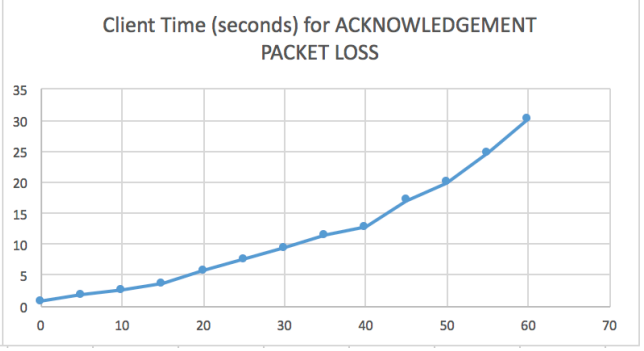
DATA PACKET LOSS:





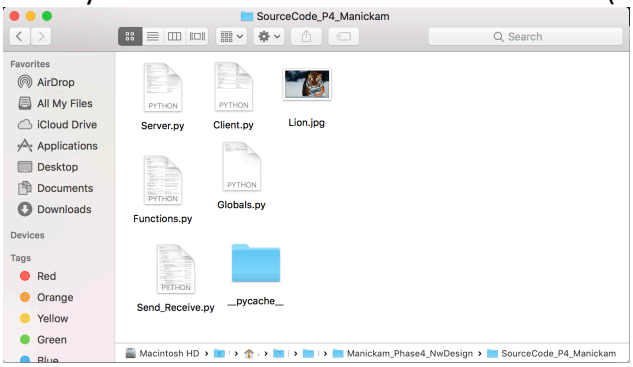
ACKNOWLEDGEMENT PACKET LOSS:



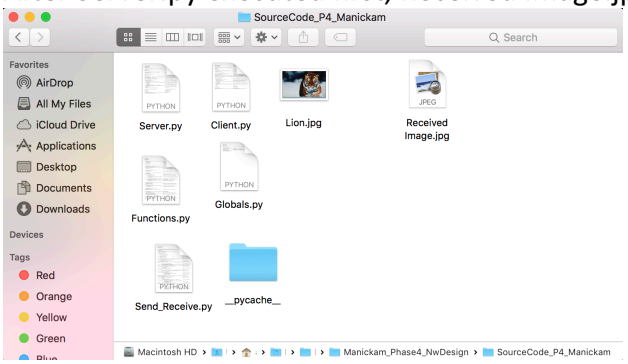


FILE TRANSFER OUTPUT:

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

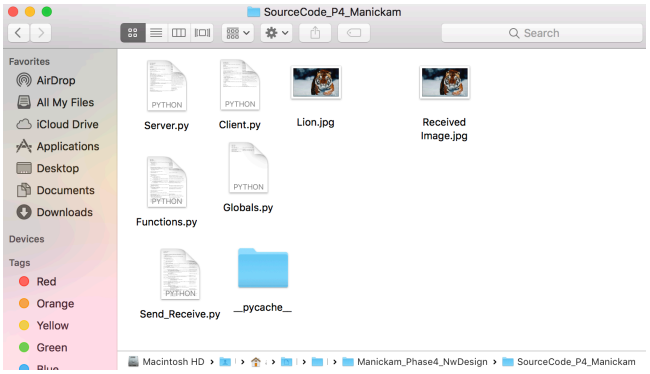
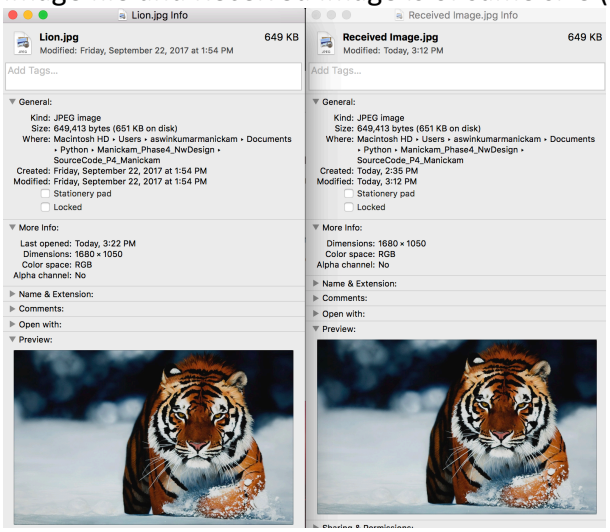


Image file and Received Image is of same size (Reliable Data Transfer)



Result:

Thus RDT 3.0 over an unreliable UDP channel with bit-errors and loss has been implemented.