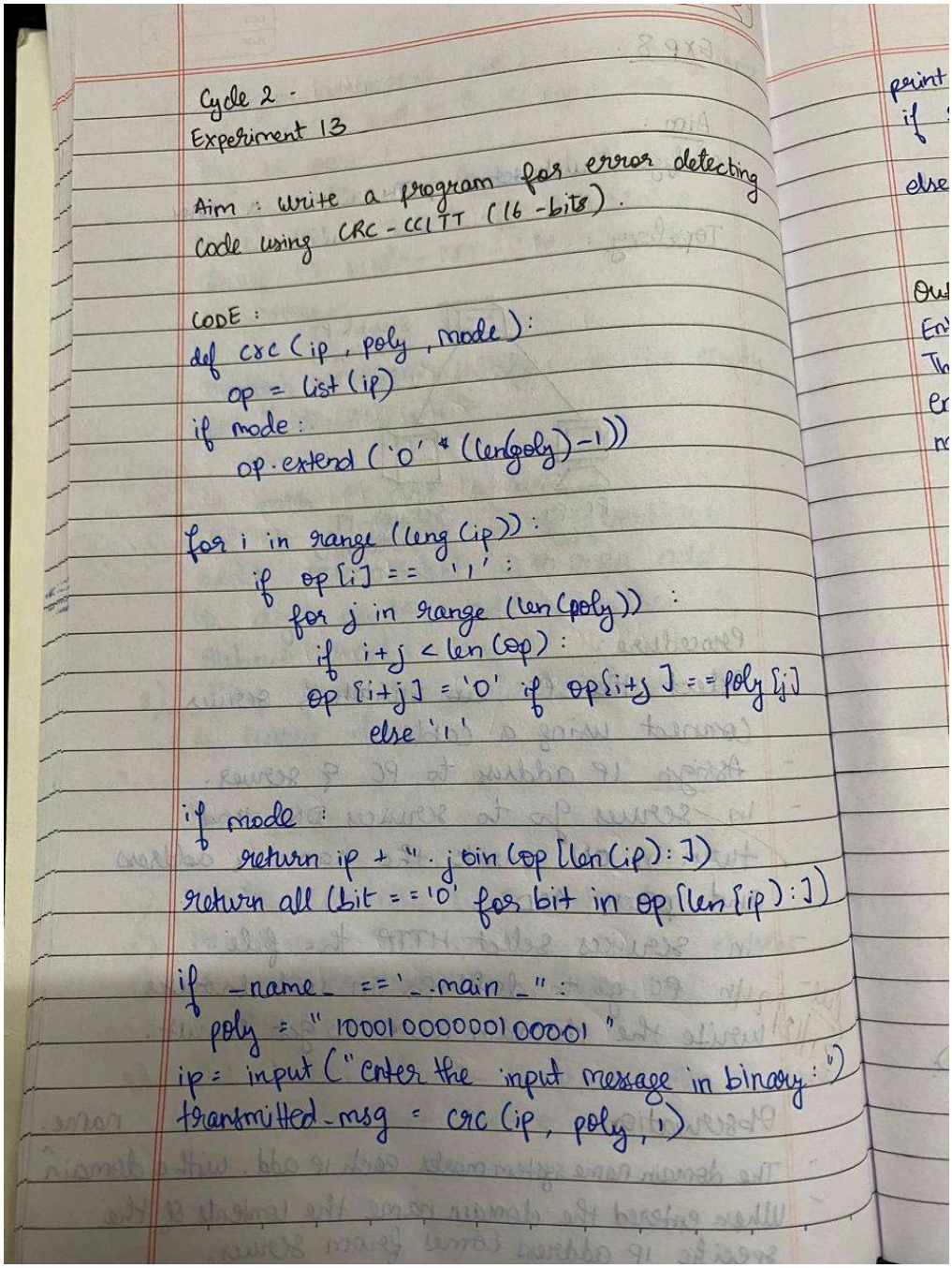
Cycle 2 - 1/1/25
Experiment 13



Cycle 2 -
Experiment 13

Aim : Write a program for error detecting code using CRC - CCITT (16 - bits).

CODE :

```
def crc(ip, poly, mode):
    op = list(ip)
if mode:
    op.extend('0' * (len(poly)-1))

for i in range(len(ip)):
    if op[i] == '1':
        for j in range(len(poly)):
            if i+j < len(op):
                op[i+j] = '0' if op[i+j] == poly[j]
                    else '1'

if mode:
    return ip + ''.join(op[len(ip):])
return all(bit == '0' for bit in op[len(ip):])

if __name__ == '__main__':
    poly = "10001000000100001"
ip = input("Enter the input message in binary: ")
transmitted_msg = crc(ip, poly, 1)
```

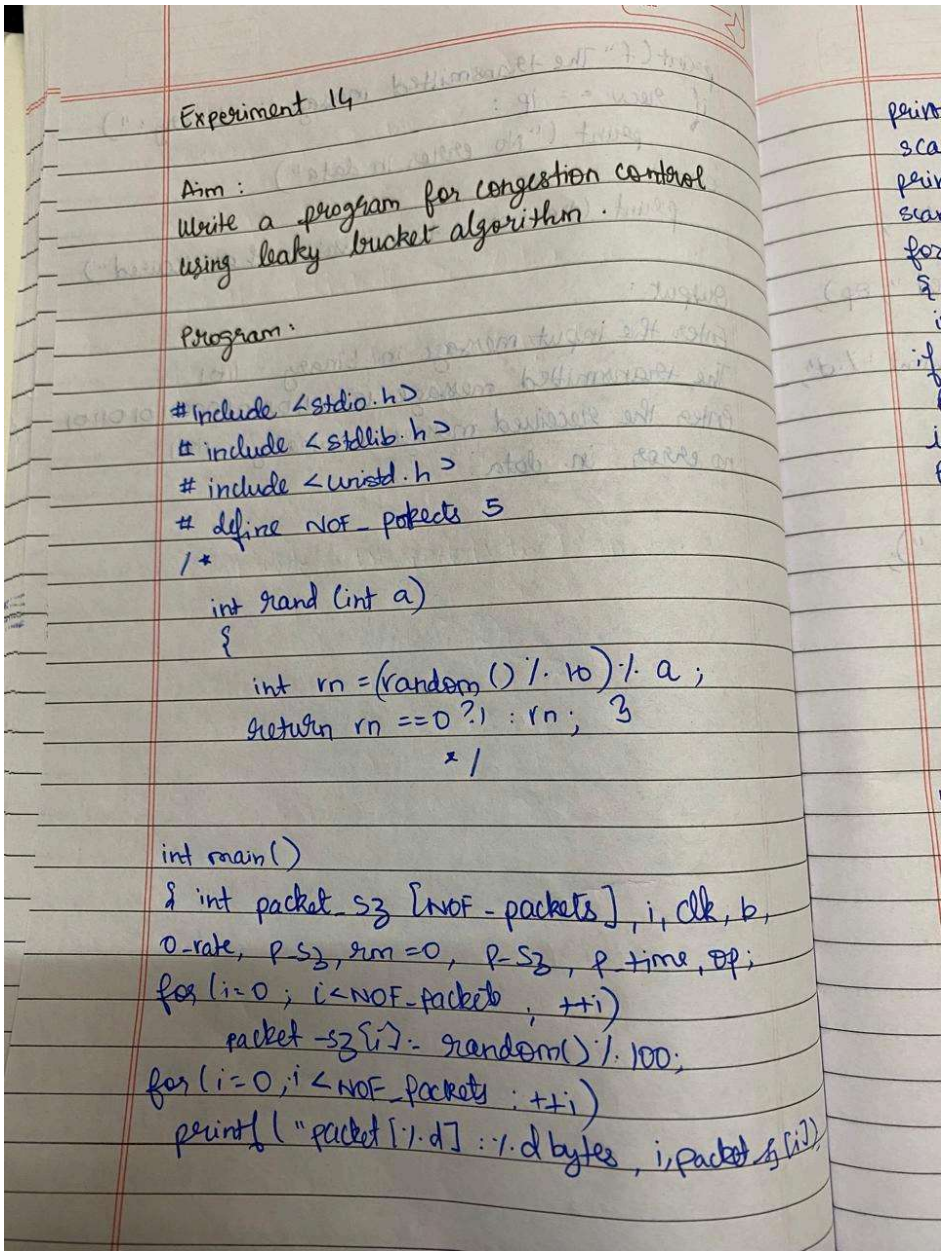
```
print(f"The transmitted message in binary : ")
if recv == ip :
    print ("No error in data")
else :
    print ("error in transmission has occured")
```

Output :

Enter the input message in binary : 1101
The transmitted message is : 1101110100011010110101
Enter the received message in binary : 1101
no error in data.

```
Enter the input message in binary: 1101
The transmitted message is: 11011101000110101101
Enter the received message in binary: 1101
No error in data
```

Experiment 14



Experiment 14

Aim:
Write a program for congestion control using leaky bucket algorithm.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define NOF_packets 5
/*
  int rand(int a)
  {
     int rn = (random() % 10) % a;
     return rn == 0 ? 1 : rn;
  }
*/

int main()
{ int packet_sz[NOF_packets], i, clk, b,
o_rate, p_sz, rm = 0, p_sz, p_time, op;
for(i=0; i<NOF_packets; ++i)
    packet_sz[i] = random() % 100;
for(i=0; i<NOF_Packets; ++i)
 printf("packet[%d] : %d bytes", i, packet_sz[i]);
```

```
printf("Enter the output rate ");
scanf("%d", &o_rate);
printf("Enter the bucket size :");
scanf("%d", &b_size);
for(i=0; i<NOF_packets ; ++i)
{
 if(packet_sz[i] + p_sz_rm) > b_size)
if(packet_sz[i] > b_size)
 printf("Incoming packet size (%d bytes)
is greater that bucket capacity (%d bytes)-
PACKET REJECTED", packet_sz[i], b_size);
 else
printf("Bucket capacity exceeded -
      PACKETS REJECTED");
else
{ p_sz_rm += packet_sz[i];
printf("Incoming packet size : %d",
      packet_sz[i]);
printf("Bytes remaining to transmit : %d",
      p_sz_rm);

while(p_sz_rm > 0)

{ sleep(1);
    if(p_sz_rm)
    {
```

```
if (p_sz_rm <= o_rate)
  op = p_sz_rm, p_sz_rm = 0;
else
  op = o_rate, p_sz_rm = o_rate;
printf("Packet of size %d transmitted", op);
printf("\n Bytes remaining to transmit: %d
      p_sz_rm);
  }
  else
  {
  printf("No packets to transmit !! ");
  }
  }
  }
  }
```

```
packet[0]:83 bytes
packet[1]:86 bytes
packet[2]:77 bytes
packet[3]:15 bytes
packet[4]:93 bytes
Enter the Output rate:30
Enter the Bucket Size:85


Incoming Packet size: 83
Bytes remaining to Transmit: 83
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 53
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 23
Packet of size 23 Transmitted----Bytes Remaining to Transmit: 0

Incoming packet size (86bytes) is Greater than bucket capacity (85bytes)-PACKET REJECTED

Incoming Packet size: 77
Bytes remaining to Transmit: 77
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 47
Packet of size 30 Transmitted----Bytes Remaining to Transmit: 17
Packet of size 17 Transmitted----Bytes Remaining to Transmit: 0

Incoming Packet size: 15
Bytes remaining to Transmit: 15
Packet of size 15 Transmitted----Bytes Remaining to Transmit: 0
```

Experiment 15

Experiment 15

Aim:
Using TCP/IP Sockets, write a client server program to make client sending the file name and the server to send back the contents of the requested file if present

Solution:
Client TCP.py

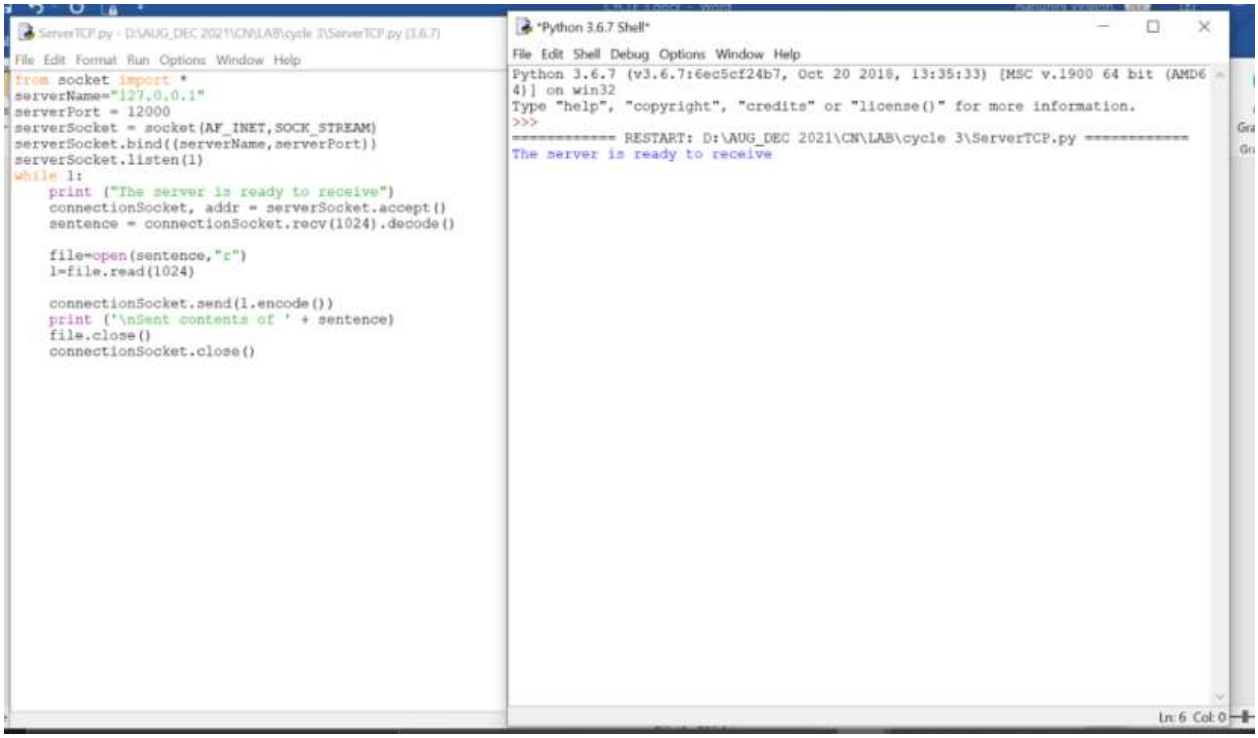
```
from socket import *
servername = '127.0.0.1'
serverport = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((servername, serverport))
sentence = input("Enter file name:")

clientSocket.send(sentence.encode())
file contents = clientSocket.recv(1024).decode()
print('From server')
print(file contents)
clientSocket.close()
```

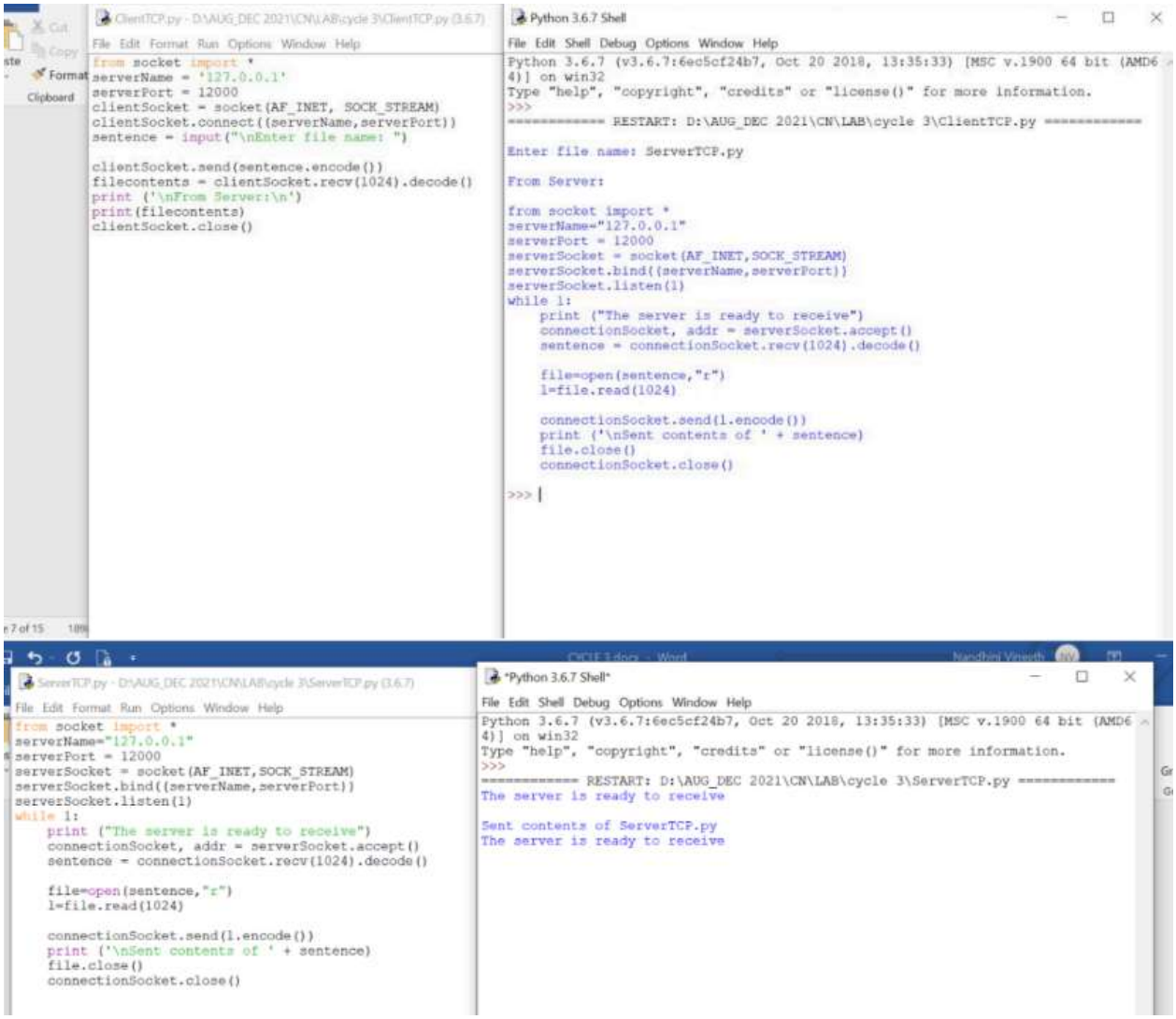
serverTCP.py

```
from Socket import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_NET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while(1):
    print("The server is ready to recieve")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()

file = open(sentence, "r")
file = open(sentence, "r")
l = file.read(4024)
connectionSocket.send(l.encode())
print('Sent contents of ' + sentence)
file.close()
connectionSocket.close()
```

Output:

The server is ready to recieve
Enter file name : serverTCP.py
Sent contents of serverTCP.py


ServerTCP.py - D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerTCP.py (3.6.7)
File Edit Format Run Options Window Help
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,"r")
l=file.read(1024)
connectionSocket.send(l.encode())
print ('\nSent contents of ' + sentence)
file.close()
connectionSocket.close()
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerTCP.py ==========
The server is ready to receive
Ln: 6 Col: 0


ClientTCP.py - D:\AUG_DEC 2021\CN\LAB\cycle 3\ClientTCP.py (3.6.7)
File Edit Format Run Options Window Help
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input("\nEnter file name: ")
clientSocket.send(sentence.encode())
filecontents = clientSocket.recv(1024).decode()
print ('\nFrom Server:\n')
print(filecontents)
clientSocket.close()
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ClientTCP.py ==========
Enter file name: ServerTCP.py
From Server:
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,"r")
l=file.read(1024)
connectionSocket.send(l.encode())
print ('\nSent contents of ' + sentence)
file.close()
connectionSocket.close()
>>>
ServerTCP.py - D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerTCP.py (3.6.7)
File Edit Format Run Options Window Help
from socket import *
serverName="127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((serverName,serverPort))
serverSocket.listen(1)
while 1:
print ("The server is ready to receive")
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file=open(sentence,"r")
l=file.read(1024)
connectionSocket.send(l.encode())
print ('\nSent contents of ' + sentence)
file.close()
connectionSocket.close()
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerTCP.py ==========
The server is ready to receive
Sent contents of ServerTCP.py
The server is ready to receive

Experiment 16

Experiment 16

Aim :
Using UDP sockets, write a client server program to make client sending the file name and the server to send back the contents of the required file if present.

Solution :
ClientUDP.py

```
from socket import *
serverName = "127.0.0.1"
serverPort = 12000
clientSocket = socket (AF_INET, SOCK_DGRAM)
sentence = input ("Enter file name")

clientSocket.sendto(bytes(sentence, "utf-8"),
        (serverName, serverPort))

fileContents.serverAddress = clientSocket.recvfrom(2048)
print ("Reply from server")
print (fileContents.decode("utf-8"))
 # for i in fileContents:
        # print (str(i), end = '')
clientSocket.close()
clientSocket.close()
```

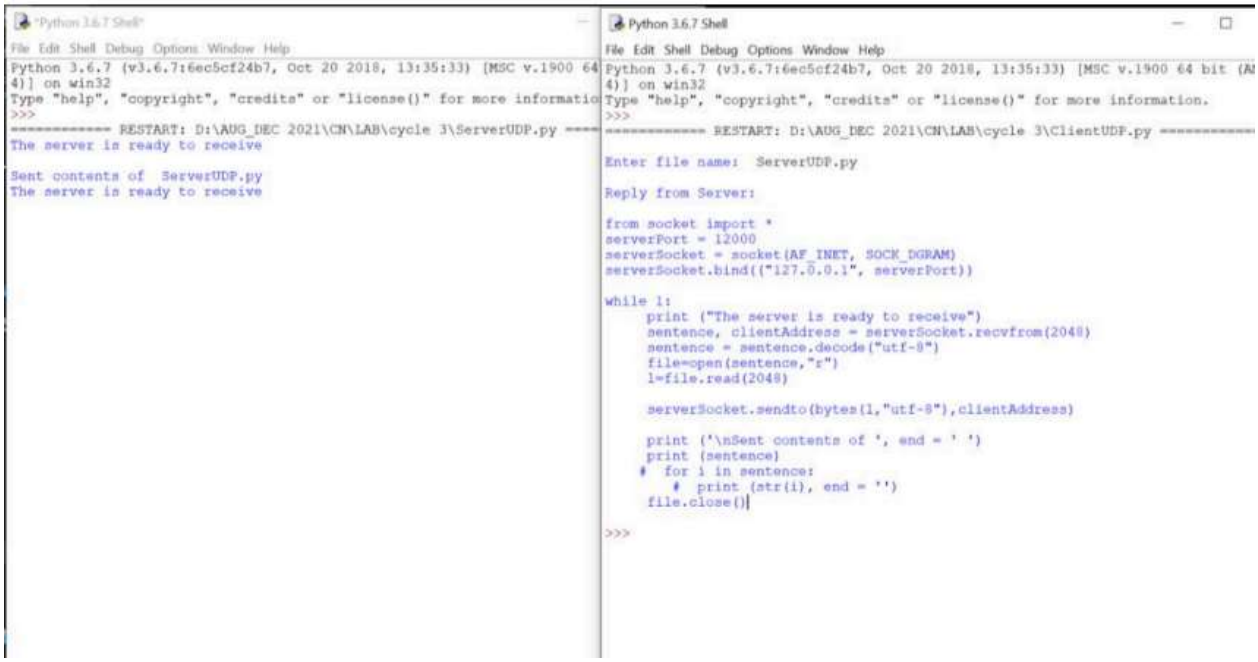
ServerUDP.py

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to recieve")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    con = file.read(2048)

serverSocket.sendto(bytes(con, "utf-8"),
                    clientAddress)
print('Sent contents of ', end = '')
print(sentence)
# for i in sentence: # print(str(i), end="")
    file.close()
```

Output:

The server is ready to recieve
Enter file name : ServerUDP.py

Sent contents of ServerUDP.py
The server is ready to recieve.



"Python 3.6.7 Shell"

File Edit Shell Debug Options Window Help

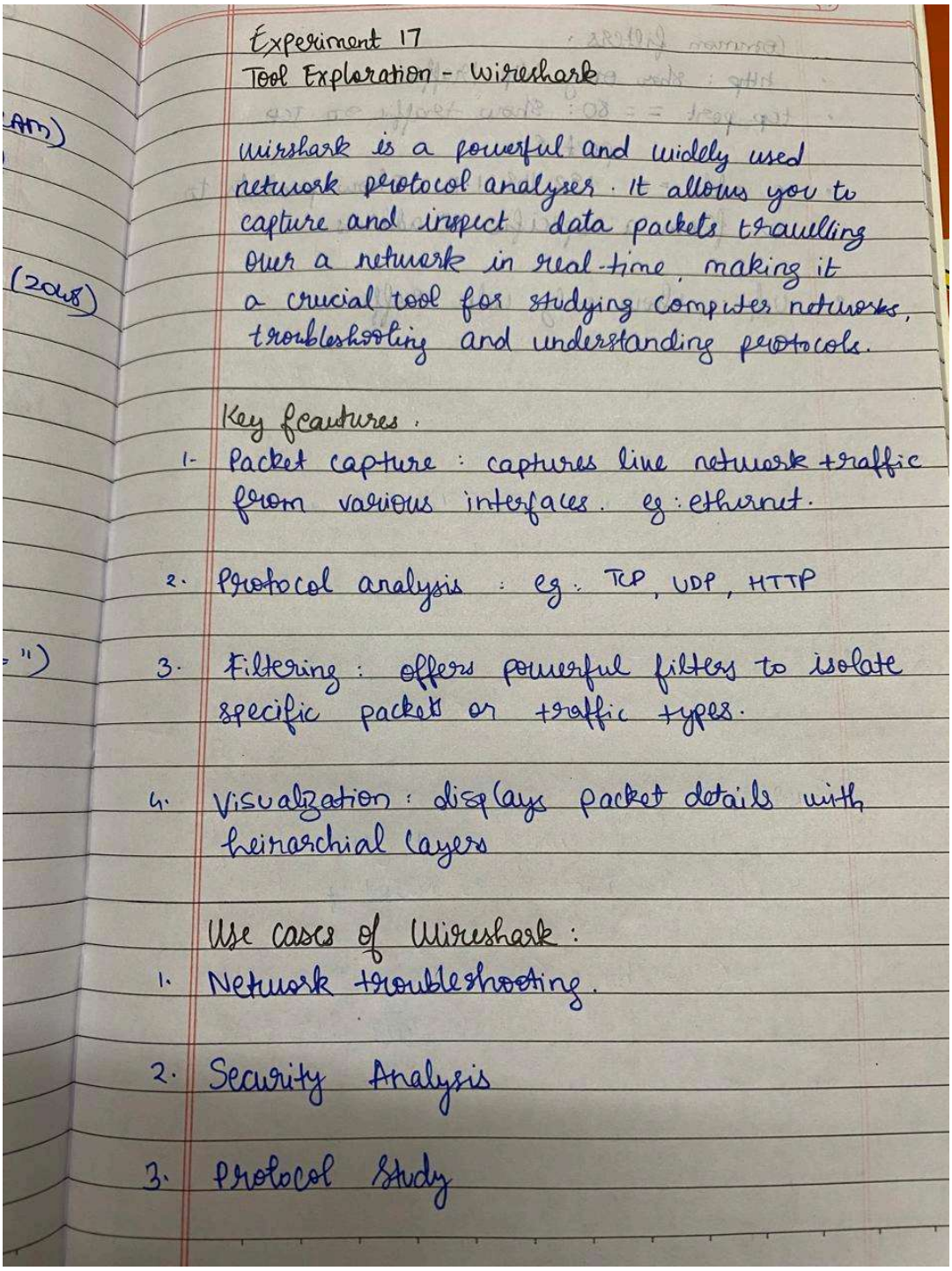
```
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64
4)] on win32
Type "help", "copyright", "credits" or "license()" for more informatio
>>>
========== RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ServerUDP.py ====
The server is ready to receive

Sent contents of  ServerUDP.py
The server is ready to receive
```

Python 3.6.7 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (A
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\AUG_DEC 2021\CN\LAB\cycle 3\ClientUDP.py ==========

Enter file name:  ServerUDP.py

Reply from Server:

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))

while 1:
    print ("The server is ready to receive")
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("utf-8")
    file=open(sentence,"r")
    l=file.read(2048)

    serverSocket.sendto(bytes(l,"utf-8"),clientAddress)

    print ('\nSent contents of ', end = ' ')
    print (sentence)
  #  for i in sentence:
      #  print (str(i), end = '')
    file.close()

>>>
```

Experiment 17



Experiment 17
Tool Exploration - Wireshark

Wirshark is a powerful and widely used network protocol analyser. It allows you to capture and inspect data packets travelling over a network in real-time, making it a crucial tool for studying computer networks, troubleshooting and understanding protocols.

Key features:

1. Packet capture : captures live network traffic from various interfaces. eg: ethernet.
2. Protocol analysis : eg: TCP, UDP, HTTP
3. Filtering : offers powerful filters to isolate specific packet or traffic types.
4. Visualization : displays packet details with heirarchial layers

Use cases of Wireshark:

1. Network troubleshooting.
2. Security Analysis
3. Protocol Study

Common filters:

- http : show only http traffic
- tcp.port == 80 : show traffic on TCP port 80.
- ip.addr == 192.168.1.1 : show packets to or from a specific IP address.
- udp : show only udp traffic.