

GROUP B

COMPUTER NETWORKS

- Taazim Toktogulov
- RAKHIMOV RUSTAMBEK ISMOILBEK UGLI



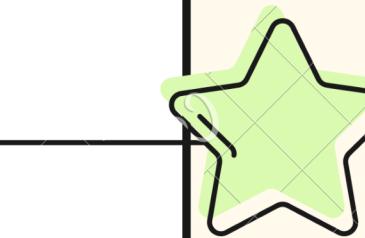
Ads



ooo

Cisco packet

Python



TASKS

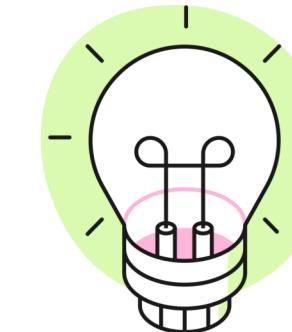
A CLIENT CAN SEND AND VISUALIZE THE IMAGE, CSV AND JSON FILES IN OTHER CLIENT SIDE



• A CLIENT CAN BROADCAST ANY FILE TO MULTIPLE CLIENTS.



• A MENU FORM SELECTING CHOICES





INTRODUCTION

Broadcast to client1 and client 2

```
# CLIENT 1  
# importing necessary libraries  
import socket  
import struct  
  
# setting up the IP/PORT and buffer size  
serverip = "224.1.1.1"  
serverport = 6464  
bufferSize = 2048
```

```
# Setting up the connection by connecting our code to the socket  
ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM,  
socket.IPPROTO_UDP)  
ClientSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  
ClientSocket.bind(("0.0.0.0", serverport)) # Binding the socket to the ip and port
```

```
# Code below is used to enable UDP multicast  
var = struct.pack("4sl", socket.inet_aton(serverip), socket.INADDR_ANY)  
ClientSocket.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP,  
var)
```

```
# Listen for messages from the server/sender and decoding it  
msgFromServer = ClientSocket.recvfrom(bufferSize)  
msg = "Message from Server {}".format(msgFromServer[0].decode())
```

```
# Printing the message received for user view  
print(msg)
```

```
# CLIENT 2  
# importing necessary libraries  
import socket  
import struct
```

```
# setting up the IP/PORT and buffer size  
serverip = "224.1.1.1"  
serverport = 6464  
bufferSize = 2048
```

```
# Setting up the connection by connecting our code to the socket  
ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)  
ClientSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  
ClientSocket.bind(("0.0.0.0", serverport)) # Binding the socket to the ip and port
```

```
# Code below is used to enable UDP multicast  
var = struct.pack("4sl", socket.inet_aton(serverip), socket.INADDR_ANY)  
ClientSocket.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, var)
```

```
# Listen for messages from the server/sender and decode it  
msgFromServer = ClientSocket.recvfrom(bufferSize)  
msg = "Message from Server {}".format(msgFromServer[0].decode())
```

```
# Printing the message received for user view  
print(msg)
```

server

```
# SERVER
# IMPORTING THE SOCKET LIBRARY
REQUIRED FOR THE SENDER/SERVER
import socket

# SETTING THE IP AND PORT USED FOR
CREATING A SOCKET
Addr = '224.1.1.1'
port = 6464

# 2-hop restriction in network
ttl = 2

# Setting up the socket for a multicast that
# allows sending and receiving a broadcast.
sock = socket.socket(socket.AF_INET,
                      socket.SOCK_DGRAM,
                      socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP,
                socket.IP_MULTICAST_TTL,
                ttl)
# The code below sends a broadcast to client 1
# and client 2
sock.sendto(b"This is a message that was
broadcasted!", (Addr, port))
```

The screenshot shows a PyCharm interface with three terminal panes. The top pane contains the Python code for both the server and client sides. The bottom-left pane (Server) shows the command-line output for the server script, indicating it finished with exit code 0. The bottom-right pane (Client1) shows the command-line output for the first client, which received the broadcast message "Message from Server This is a message that was broadcasted!". The bottom-middle pane (Client2) shows the command-line output for the second client, which also received the same broadcast message.

```
Client.py
Client2.py
Server.py

# SERVER
# IMPORTING THE SOCKET LIBRARY
REQUIRED FOR THE SENDER/SERVER
import socket

# SETTING THE IP AND PORT USED FOR
CREATING A SOCKET
Addr = '224.1.1.1'
port = 6464

# 2-hop restriction in network
ttl = 2

# Setting up the socket for a multicast that
# allows sending and receiving a broadcast.
sock = socket.socket(socket.AF_INET,
                      socket.SOCK_DGRAM,
                      socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP,
                socket.IP_MULTICAST_TTL,
                ttl)
# The code below sends a broadcast to client 1
# and client 2
sock.sendto(b"This is a message that was
broadcasted!", (Addr, port))

# SERVER
import socket
import struct
serverip = "224.1.1.1"
serverport = 6464
bufferSize = 2048

ClientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
ClientSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
ClientSocket.bind(('', serverport))
mreq = struct.pack("=ll", socket.inet_aton(serverip), socket.INADDR_ANY)
ClientSocket.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

msgFromServer = ClientSocket.recvfrom(bufferSize)

msg = "Message from Server {}".format(msgFromServer[0].decode())

print(msg)
```

```
Server:
C:/Users/nrakh/AppData/Local/Programs/Python/Python39/python.exe
Process finished with exit code 0

Client1:
C:/Users/nrakh/AppData/Local/Programs/Python/Python39/python
Message from Server This is a message that was broadcasted!
Process finished with exit code 0

Client2:
C:/Users/nrakh/AppData/Local/Programs/Python/Python39/python
Message from Server This is a message that was broadcasted!
Process finished with exit code 0
```

SENDING IMAGE, CSV FILE OR JSON FILE

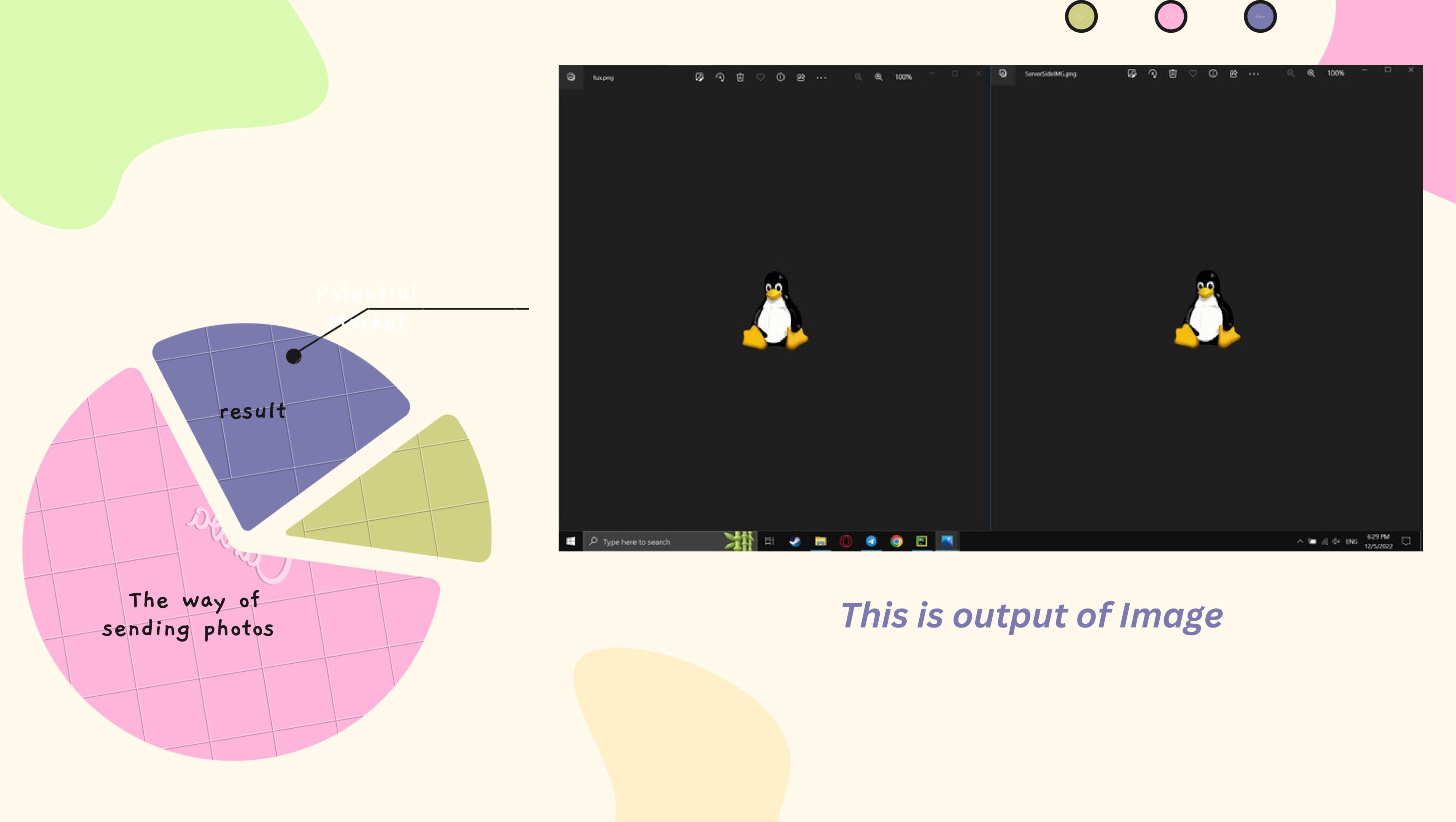
```
UNIV UDPClient.py Current File ▾ ▾ ▾ ▾ ▾ ▾ ▾
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

```
def switch1(user_choice):  
    if user_choice == "IMG":  
        g = 0  
        type_send = str.encode("IMG")      # Encoding the data type  
        UDPClientSocket.sendto(type_send, serverAddressPort)      # Sending the data type  
        time.sleep(1)      # Waiting for the server  
  
        file = open('tux.png', 'rb')  
        image = file.read(1024)  
        while image:  
            start = time.time()  
            bytesToSend = str.encode(str(image))          # Encoding the data line by line  
            UDPClientSocket.sendto(bytesToSend, serverAddressPort)      # Sending the data line by line  
            time.sleep(1)      # Timer used to calculate RTT  
            msgFromServer = UDPClientSocket.recvfrom(bufferSize)      # Catching server response  
            msg = "Message from Server {}".format(msgFromServer[0])      # Viewing server response  
  
            if msg == "Message from Server b'Pong!':      # If ACK was received:  
                print("RTT: ", time.time() - start)      # Timer used to calculate RTT  
                print("ACK from server received.")      # Inform the user that the ACK was received.
```

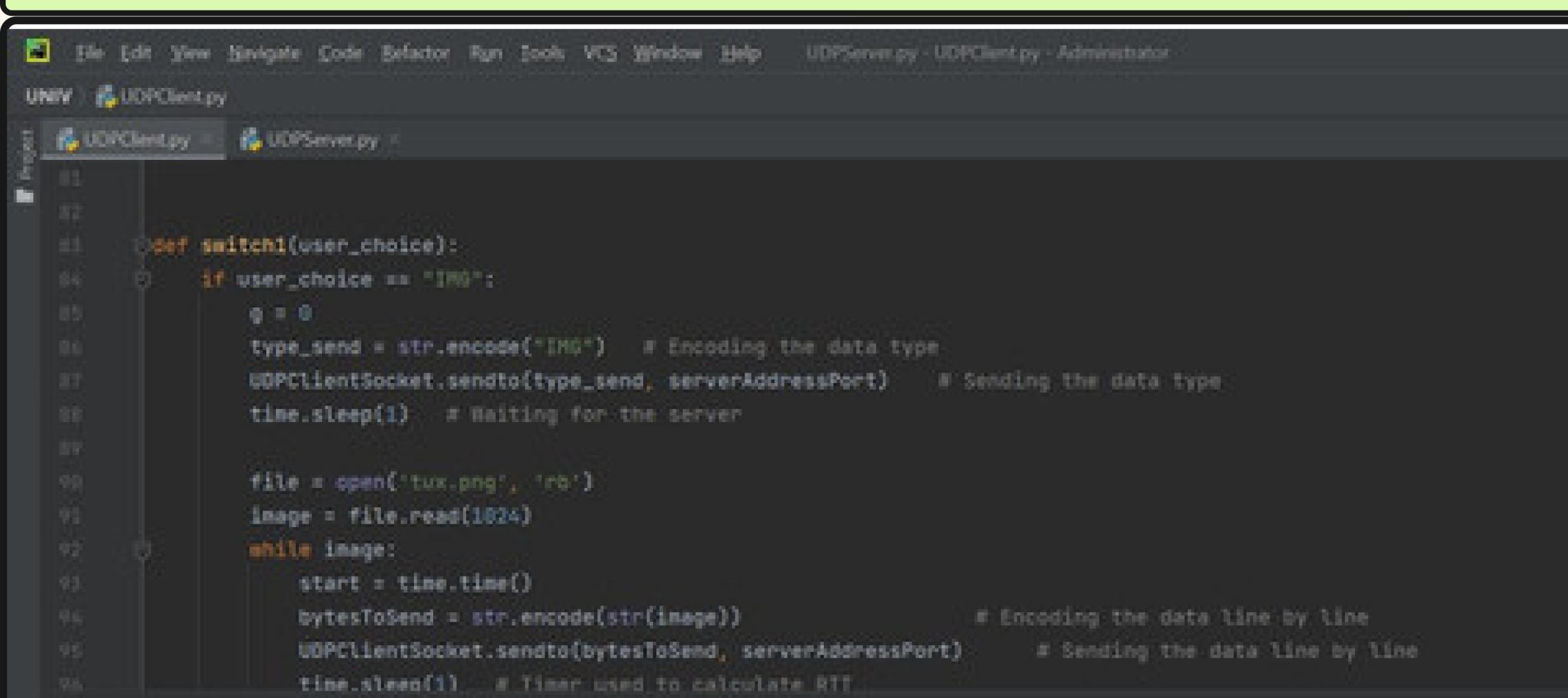
```
Run UDPClient.py
▶ -IMG -> Image
▶ -CSV -> Csv File
▶ -JSON -> Json File
What would you like to send?: CSV
RTT: 1.011275291442871
ACK from server received.
RTT: 1.0057628154754839
ACK from server received.
RTT: 1.0044441223144531
ACK from server received.
RTT: 1.010587453842163
```

```
C:\Users\rrrakh\AppData\Local\Programs\Python\Python39\python.exe C:/Users/rrrakh/OneDrive/Desktop/UDPserver.py
UDP server is running...
INFO
```



This is output of Image

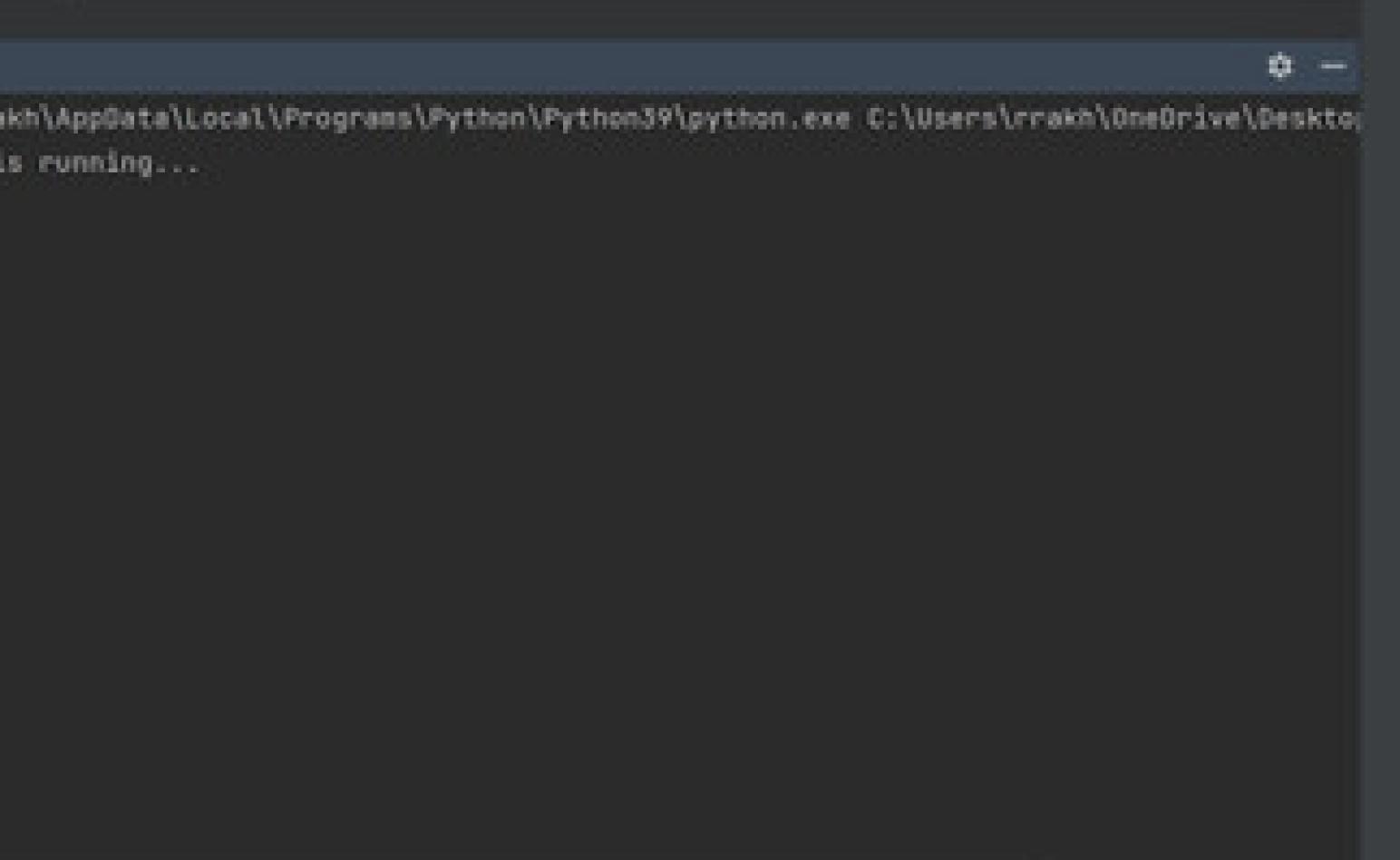
CSV FILE OUTPUT AND IT'S CODE



The screenshot shows a Python IDE interface with two tabs: UDPClient.py and UDPServer.py. The UDPClient.py tab contains the following code:

```
11
12
13     def switch1(user_choice):
14         if user_choice == "IMG":
15             s = 0
16             type_send = str.encode("IMG")      # Encoding the data type
17             UDPClientSocket.sendto(type_send, serverAddressPort)    # Sending the data type
18             time.sleep(1)    # Waiting for the server
19
20             file = open('tvx.png', 'rb')
21             image = file.read(1024)
22             while image:
23                 start = time.time()
24                 bytesToSend = str.encode(str(image))          # Encoding the data line by line
25                 UDPClientSocket.sendto(bytesToSend, serverAddressPort)    # Sending the data line by line
26                 time.sleep(1)    # Timer used to calculate RTT
27
28             file.close()
```

The Run tab shows the command to run UDPClient.py and the user input "What would you like to send?: CSV".



The terminal window displays the output of the UDPClient.py script, which includes the CSV file content and the message "UDP server is running...".

```
-CSV -> Csv File
-JSON -> Json File
What would you like to send?: CSV

      #   Name Type 1 ... Speed Generation Legendary
0  1   Bulbasaur  Grass ... 45       1   False
1  2   Ivysaur   Grass ... 60       1   False
2  3   Venusaur  Grass ... 80       1   False
3  3  Venusaur?Mega Venusaur  Grass ... 80       1   False
4  4   Charmander Fire ... 65       1   False

[5 rows x 13 columns]

C:\Users\rrrakh\AppData\Local\Programs\Python\Python39\python.exe C:\Users\rrrakh\OneDrive\Desktop\UDPServer.py
UDP server is running...
```

CSV FILE OUTPUT

Save ⌘ S ServerSideCSV.csv Search (Alt+Q) rustam.nachimov A

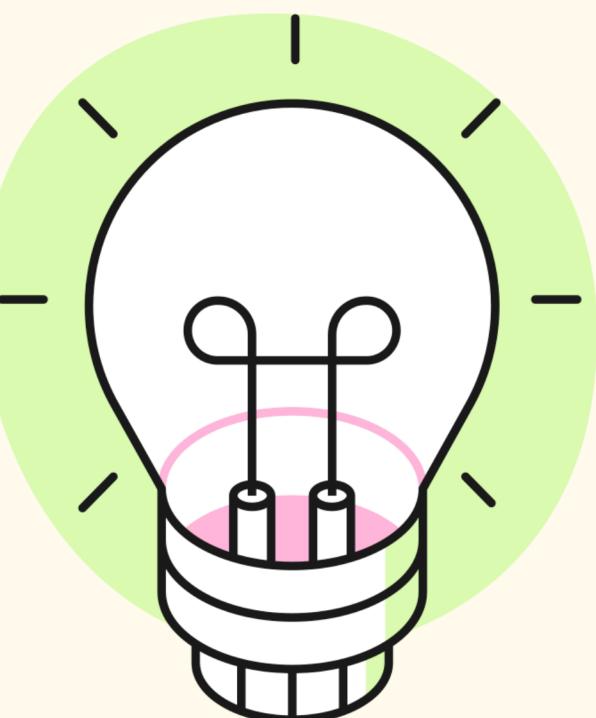
Home Insert Page Layout Formulas Data Review View Help

Cut Copy Format Painter Paste Alignment Number Styles Cells Editing

General Conditional Format as Table Normal Bad Good Neutral AutoSum Fill Sort & Filter Clear

POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary										
1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False										
2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False										
3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False										
3	Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False										
4	Charmeleon	Fire		309	39	52	43	60	50	65	1	False										
5	Charmeleon	Fire		405	58	64	58	80	65	80	1	False										
6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False										
6	Charizard	Fire	Dragon	634	78	130	111	130	85	100	1	False										
6	Charizard	Fire	Flying	634	78	104	78	159	115	100	1	False										



JSON FILE CODE AND OUTPUT

```
Run: UDPClient >
    -IMG -> Image
    -CSV -> Csv File
    -JSON -> Json File
What would you like to send?: JSON
Choose the day: 1
RTT: 1.0091478824415479
ACK from server received.
RTT: 1.006244421005249
ACK from server received.
```

```
C:\Users\rrrakh\AppData\Local\Programs\Python\Python39\python.exe C:\Users\rrrakh\OneDrive\Desktop\UDP Server.py
```



COMPUTER NETWORKS

THANKS PROFESSOR

for a wonderful semester. I appreciated your caring disposition and teaching style. Thanks for your attitude and great job!!!

THANKS, GROUPMATES, AND TEAMMATES

For hard-working and desire!

Good luck on the semester :)