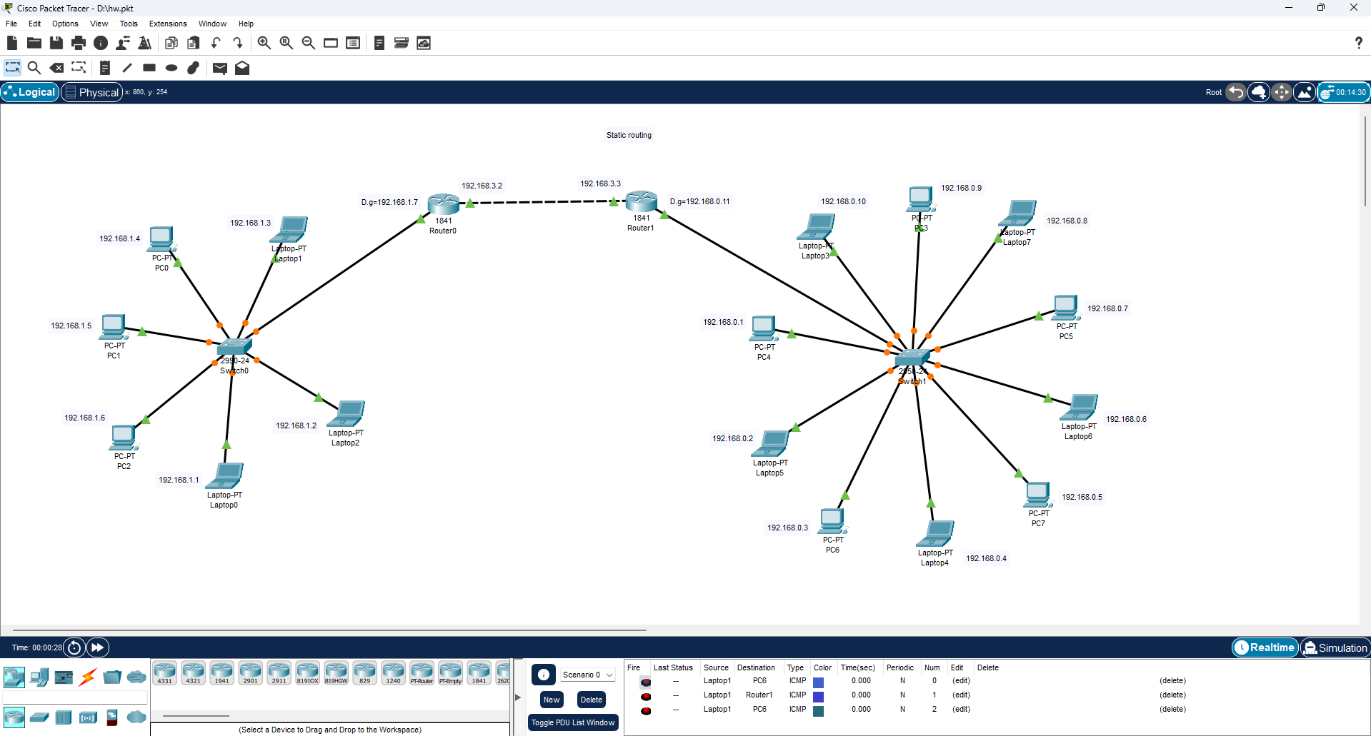
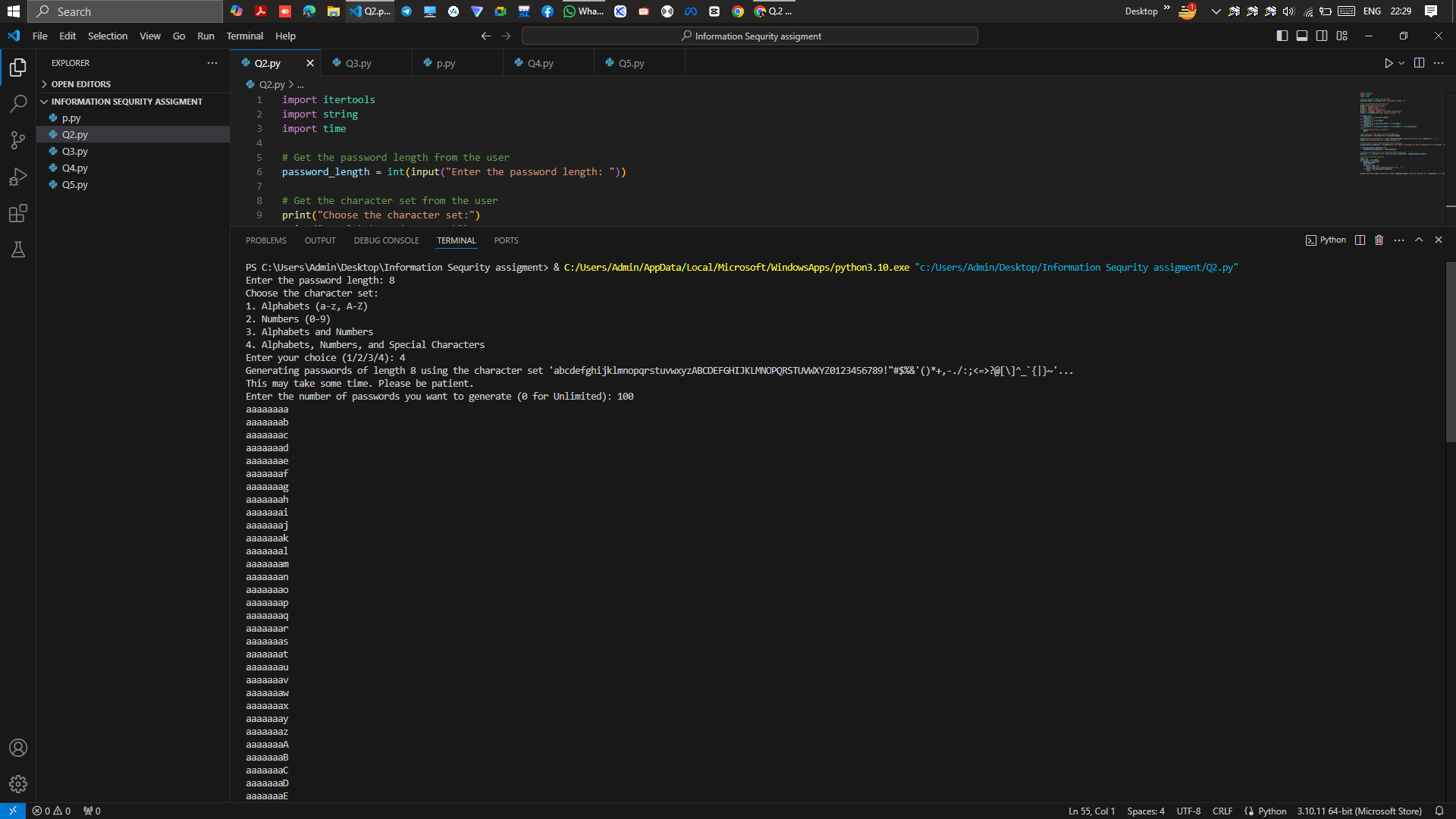
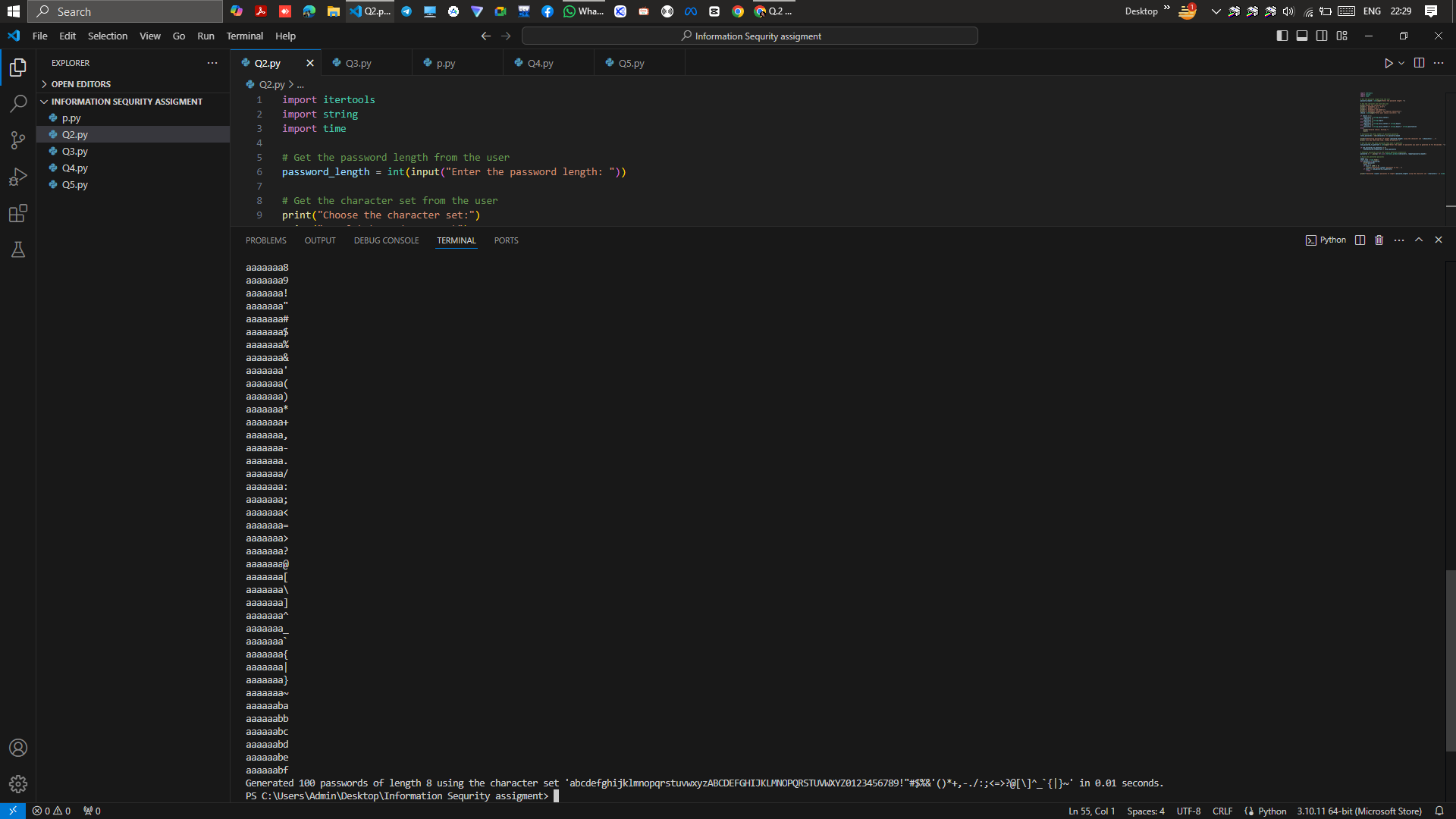
**Q.1 Make 2 LAN where first LAN connected with 6 devices and second LAN connected with 10 devices and two LAN’s are connected by two router(Remarks: try to avoid the extra IP losses).**

Ans:



**Q.2 Write a python program to generate all possible list of password from a set of alphabet and number**

**Output : **

****

**Code :**

import itertools

import string

def generate\_passwords(length, use\_lowercase=True, use\_uppercase=True, use\_digits=True):

# Define the character set based on user preferences

charset = ''

if use\_lowercase:

charset += string.ascii\_lowercase

if use\_uppercase:

charset += string.ascii\_uppercase

if use\_digits:

charset += string.digits

# Generate all possible combinations

for password in itertools.product(charset, repeat=length):

yield ''.join(password)

def main():

print("Password Generator")

# Get user input

length = int(input("Enter the length of the passwords: "))

use\_lowercase = input("Include lowercase letters? (y/n): ").lower() == 'y'

use\_uppercase = input("Include uppercase letters? (y/n): ").lower() == 'y'

use\_digits = input("Include digits? (y/n): ").lower() == 'y'

# Generate and print passwords

passwords = generate\_passwords(length, use\_lowercase, use\_uppercase, use\_digits)

print("\nGenerated Passwords:")

for i, password in enumerate(passwords, 1):

print(f"{i}. {password}")

if i % 10 == 0:

if input("Press Enter to continue or 'q' to quit: ").lower() == 'q':

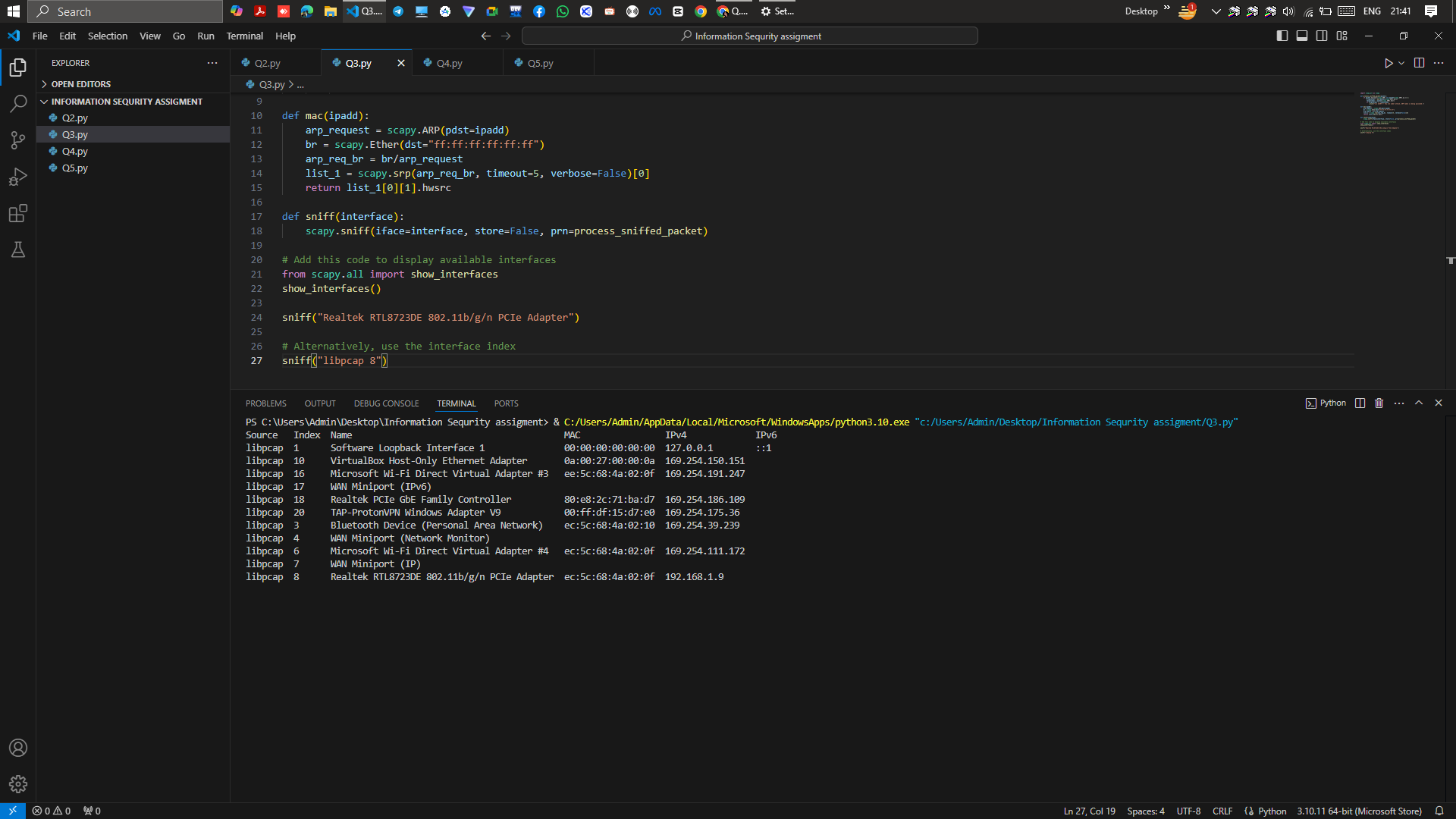
break

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Q.3 Write a Python program to capture the data packets in ARP spoofing.**

**Output:**

****

**CODE 1 :**

import scapy.all as scapy

def process\_sniffed\_packet(packet):

if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:

originalmac = mac(packet[scapy.ARP].psrc)

responsemac = packet[scapy.ARP].hwsrc

if originalmac != responsemac:

print("[\*] ALERT!!! You are under attack, ARP table is being poisoned.")

def mac(ipadd):

arp\_request = scapy.ARP(pdst=ipadd)

br = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")

arp\_req\_br = br/arp\_request

list\_1 = scapy.srp(arp\_req\_br, timeout=5, verbose=False)[0]

return list\_1[0][1].hwsrc

def sniff(interface):

scapy.sniff(iface=interface, store=False, prn=process\_sniffed\_packet)

# Add this code to display available interfaces

from scapy.all import show\_interfaces

show\_interfaces()

sniff("Realtek RTL8723DE 802.11b/g/n PCIe Adapter")

# Alternatively, use the interface index

sniff("libpcap 8")

**CODE NO 2:** I take this code for understanding

from scapy.all import sniff, Ether, IP, TCP, UDP

from threading import Thread

class PacketSniffer:

def \_\_init\_\_(self):

self.is\_sniffing = False

def start\_sniffing(self):

self.is\_sniffing = True

self.sniffer\_thread = Thread(target=self.sniff\_packets)

self.sniffer\_thread.start()

print("Started sniffing... Press Enter to stop.")

def stop\_sniffing(self):

self.is\_sniffing = False

print("Stopped sniffing.")

def sniff\_packets(self):

sniff(prn=self.process\_packet, stop\_filter=self.should\_stop\_sniffing)

def should\_stop\_sniffing(self, packet):

return not self.is\_sniffing

def process\_packet(self, packet):

packet\_summary = self.get\_packet\_summary(packet)

print(packet\_summary)

def get\_packet\_summary(self, packet):

if Ether in packet:

eth\_layer = f"Ethernet: {packet[Ether].src} -> {packet[Ether].dst}"

else:

eth\_layer = "Ethernet: Unknown"

if IP in packet:

ip\_layer = f"IP: {packet[IP].src} -> {packet[IP].dst}"

else:

ip\_layer = "IP: Unknown"

if TCP in packet:

transport\_layer = f"TCP: {packet[TCP].sport} -> {packet[TCP].dport}"

elif UDP in packet:

transport\_layer = f"UDP: {packet[UDP].sport} -> {packet[UDP].dport}"

else:

transport\_layer = "Transport: Unknown"

return f"{eth\_layer} | {ip\_layer} | {transport\_layer}"

if \_\_name\_\_ == "\_\_main\_\_":

sniffer = PacketSniffer()

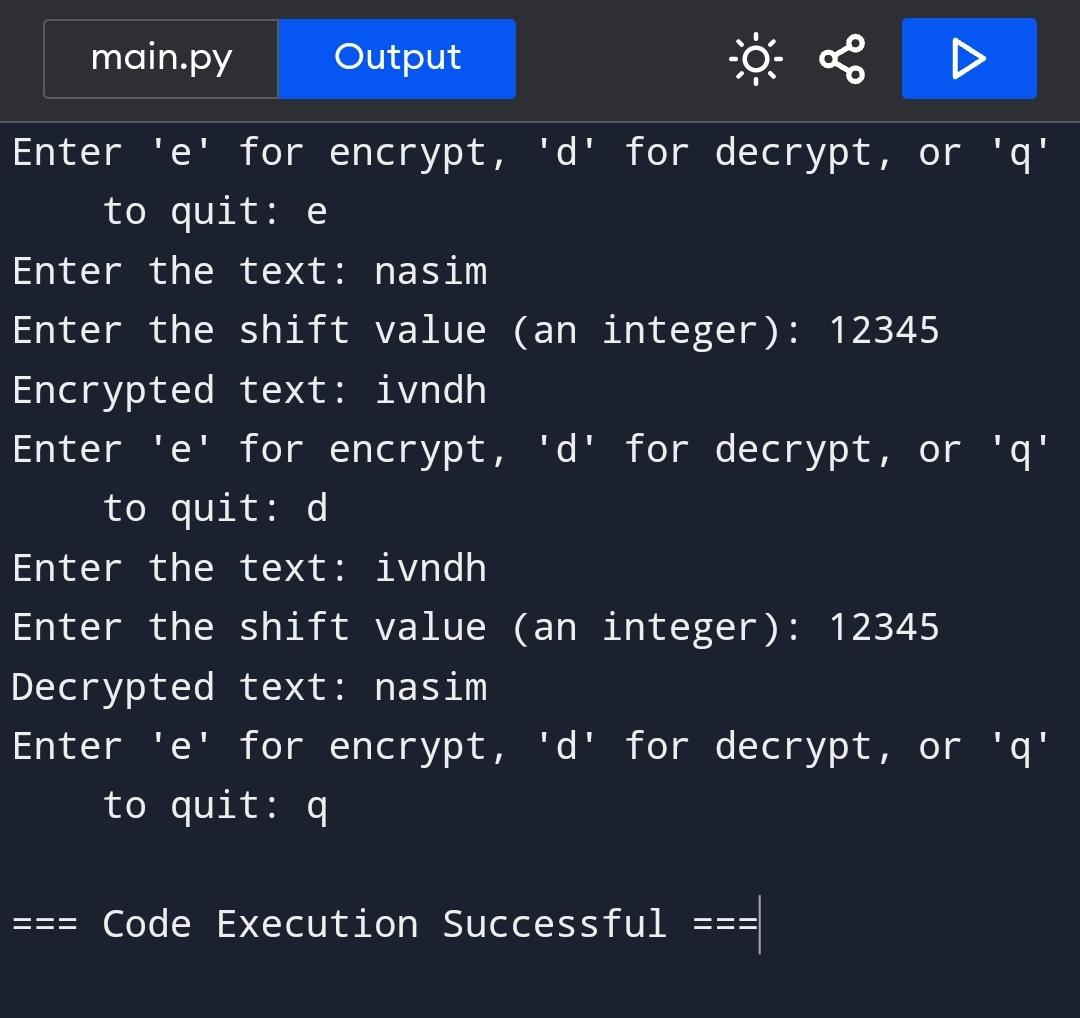
sniffer.start\_sniffing()

input("Press Enter to stop...")

sniffer.stop\_sniffing()

**Q.4 Write a Python program to encrypt or decrypt data by shift cipher, where the shift key(number) is user entered.**

**Output :**

****

**Code :**

def shift\_cipher(text, shift, mode='encrypt'):

result = ""

for char in text:

if char.isalpha():

ascii\_offset = 65 if char.isupper() else 97

if mode == 'encrypt':

shifted = (ord(char) - ascii\_offset + shift) % 26

else: # decrypt

shifted = (ord(char) - ascii\_offset - shift) % 26

result += chr(shifted + ascii\_offset)

else:

result += char

return result

def main():

while True:

mode = input("Enter 'e' for encrypt, 'd' for decrypt, or 'q' to quit: ").lower()

if mode == 'q':

break

elif mode not in ['e', 'd']:

print("Invalid mode. Please try again.")

continue

text = input("Enter the text: ")

while True:

try:

shift = int(input("Enter the shift value (an integer): "))

break

except ValueError:

print("Invalid shift value. Please enter an integer.")

if mode == 'e':

result = shift\_cipher(text, shift, 'encrypt')

print(f"Encrypted text: {result}")

else:

result = shift\_cipher(text, shift, 'decrypt')

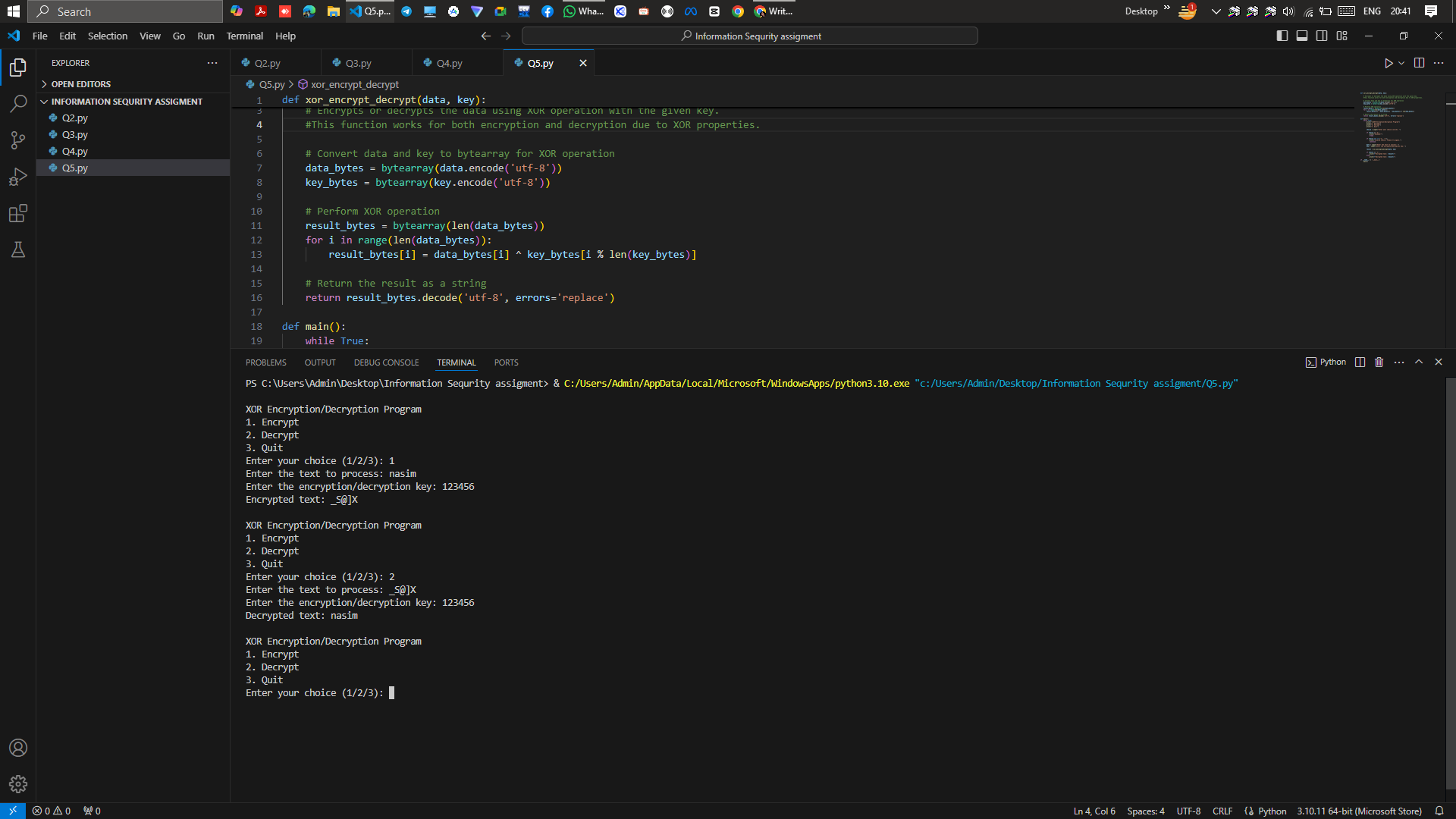
print(f"Decrypted text: {result}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Q.5 Write a Python program to encrypt and decrypt by using XOR operation.**

**Output ;**



**Code :**

def xor\_encrypt\_decrypt(data, key):

Encrypts or decrypts the data using XOR operation with the given key.

This function works for both encryption and decryption due to XOR properties.

# Convert data and key to bytearray for XOR operation

data\_bytes = bytearray(data.encode('utf-8'))

key\_bytes = bytearray(key.encode('utf-8'))

# Perform XOR operation

result\_bytes = bytearray(len(data\_bytes))

for i in range(len(data\_bytes)):

result\_bytes[i] = data\_bytes[i] ^ key\_bytes[i % len(key\_bytes)]

# Return the result as a string

return result\_bytes.decode('utf-8', errors='replace')

def main():

while True:

print("\nXOR Encryption/Decryption Program")

print("1. Encrypt")

print("2. Decrypt")

print("3. Quit")

choice = input("Enter your choice (1/2/3): ")

if choice == '3':

print("Goodbye!")

break

if choice not in ['1', '2']:

print("Invalid choice. Please try again.")

continue

data = input("Enter the text to process: ")

key = input("Enter the encryption/decryption key: ")

result = xor\_encrypt\_decrypt(data, key)

if choice == '1':

print(f"Encrypted text: {result}")

else:

print(f"Decrypted text: {result}")

if \_\_name\_\_ == "\_\_main\_\_":

main()