Sender: Generates a random number, then encrypt the lucky number by a secret key with the message digest and sent to the receiver.

Receiver: decrypt the received message, generate a new message digest, and compare it with the original one to ensure no issues.

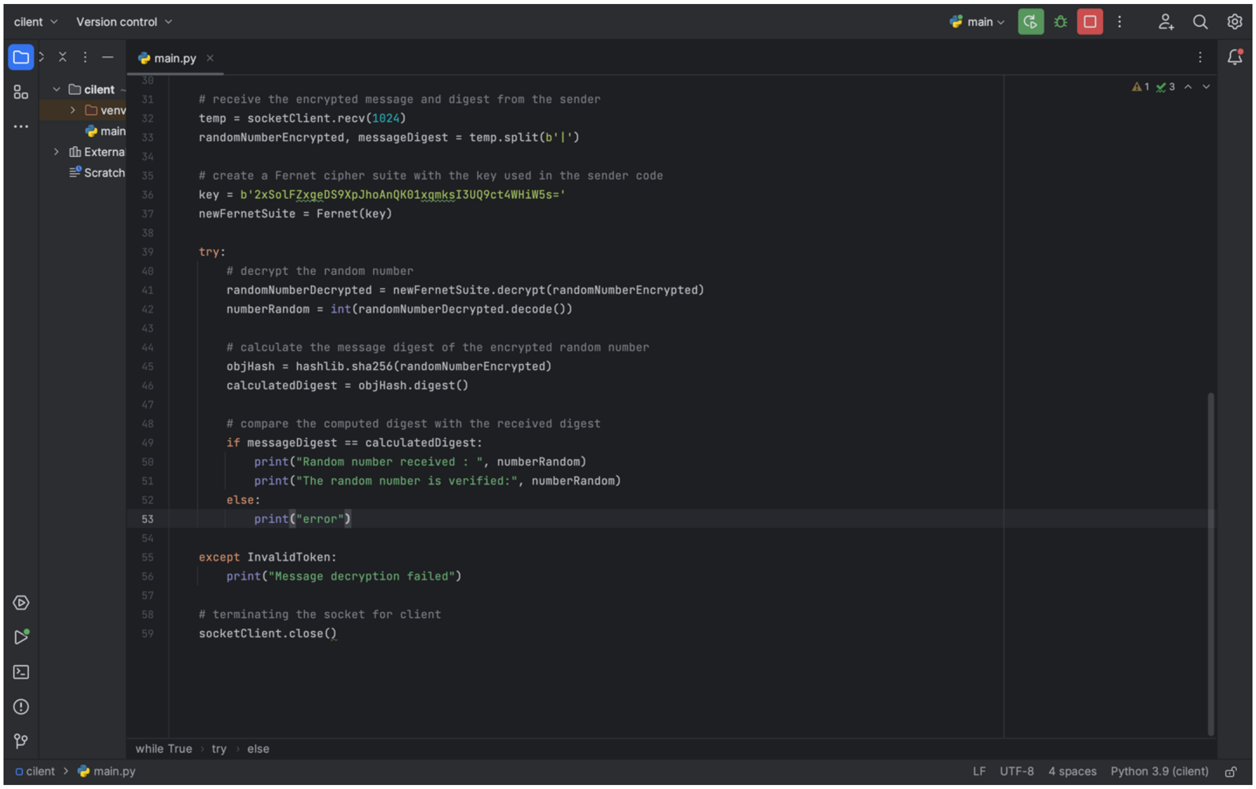
⇨ The code:

```
# import socket
#
# hostname = socket.gethostname()
# print("Hostname:", hostname)
import socket
import hashlib
from cryptography.fernet import Fernet, InvalidToken

# Create socket object
newSocketObject = socket.socket()

# get the local machine name
hostName = ""

portID = 46816
```

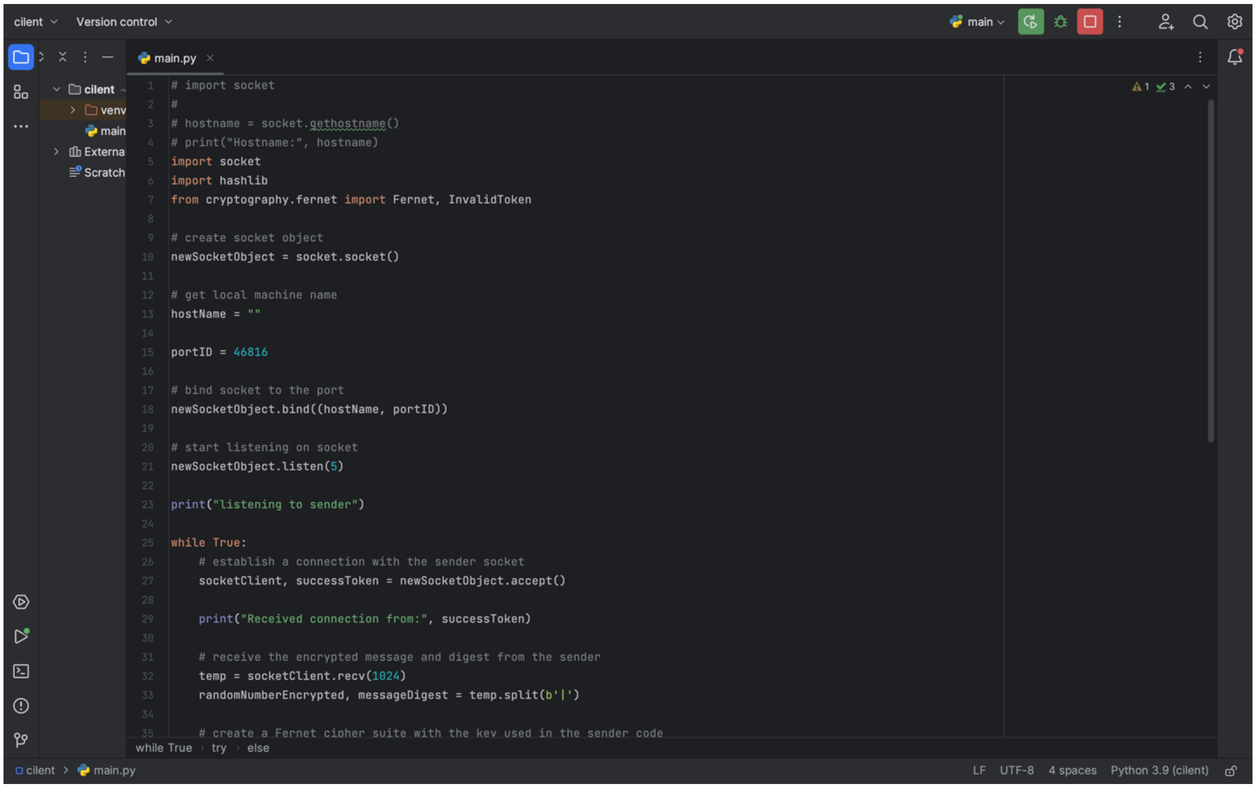
```
# Bind socket to the port
newSocketObject.bind((hostName, portID))

# start listening on the socket
newSocketObject.listen(5)

print("listening to sender")

while True:
    # establish a connection with the sender socket
    socketClient, successToken =
newSocketObject.accept()

    print("Received connection from:", successToken)

    # receive the encrypted message and digest from
the sender
    temp = socketClient.recv(1024)
    randomNumberEncrypted, messageDigest =
temp.split(b'|')

    # create a Fernet cipher suite with the key used in
the sender code
```

```
    key =
b'2xSolFZxgeDS9XpJhoAnQK01xqmksI3UQ9ct4WHi
W5s='
    newFernetSuite = Fernet(key)

    try:
        # decrypt the random number
        randomNumberDecrypted =
newFernetSuite.decrypt(randomNumberEncrypted)
        numberRandom =
int(randomNumberDecrypted.decode())

        # calculate the message digest of the encrypted
random number
        objHash =
hashlib.sha256(randomNumberEncrypted)
        calculatedDigest = objHash.digest()

        # compare the computed digest with the
received digest
        if messageDigest == calculatedDigest:
            print("Random number received : ",
numberRandom)
            print("The random number is verified:",
numberRandom)
        else:
```

```
        print("error")

except InvalidToken:
    print("Message decryption failed")

# Terminating the socket for the client
socketClient.close()
```



```

    # receive the encrypted message and digest from the sender
    temp = socketClient.recv(1024)
    randomNumberEncrypted, messageDigest = temp.split(b'|')

    # create a Fernet cipher suite with the key used in the sender code
    key = b'2xSolFZxgeDS9XpJhoAnQK01xqmksI3UQ9ct4WHiW5s='
    newFernetSuite = Fernet(key)

    try:
        # decrypt the random number
        randomNumberDecrypted = newFernetSuite.decrypt(randomNumberEncrypted)
        numberRandom = int(randomNumberDecrypted.decode())

        # calculate the message digest of the encrypted random number
        objHash = hashlib.sha256(randomNumberEncrypted)
        calculatedDigest = objHash.digest()

        # compare the computed digest with the received digest
        if messageDigest == calculatedDigest:
            print("Random number received : ", numberRandom)
            print("The random number is verified:", numberRandom)
        else:
            print("error")

    except InvalidToken:
        print("Message decryption failed")

    # terminating the socket for client
    socketClient.close()
```



```python
# import socket
#
# hostname = socket.gethostname()
# print("Hostname:", hostname)
import socket
import hashlib
from cryptography.fernet import Fernet, InvalidToken

# create socket object
newSocketObject = socket.socket()

# get local machine name
hostName = ""

portID = 46816

# bind socket to the port
newSocketObject.bind((hostName, portID))

# start listening on socket
newSocketObject.listen(5)

print("listening to sender")

while True:
    # establish a connection with the sender socket
    socketClient, successToken = newSocketObject.accept()

    print("Received connection from:", successToken)

    # receive the encrypted message and digest from the sender
    temp = socketClient.recv(1024)
    randomNumberEncrypted, messageDigest = temp.split(b'|')

    # create a Fernet cipher suite with the key used in the sender code
```

⇨ The explanation:

This Python code generates a socket server that monitors a particular port for incoming connections. The client sends an encrypted random number and a message digest when a connection is made. It computes the message digest of the encrypted random number after decrypting it with the Fernet cipher suite. Then, if the computed and received digest match, it publishes the random number. Otherwise, it validates that the computed digest matches the received digest. The message digest is computed using hashlib, and the socket object is created using the socket module. The Fernet cipher suite is also produced using Fernet from the cryptography module. This code is a straightforward illustration of a socket server that uses message authentication to ensure the integrity of received data.