Report

Simulating Encryption, Hashing and Non-Repudiation

AES Encryption and Decryption

This section of the code showcases the process of encrypting a user-provided message using the AES (Advanced Encryption Standard) algorithm. Key points to note include:

- Encryption Key: A 16-byte random key is generated to be used for AES-128 encryption.
- Encryption: The user's input message is encrypted using AES in EAX mode with the generated key.
- Nonce: The nonce (a random value used only once) is part of the output, enhancing security.
- Ciphertext: The encrypted message is displayed as ciphertext.
- Encryption Key: The randomly generated key is printed to the console.

Hash Generation:

This part of the code focuses on generating a hash value from a user-provided message. Important details are as follows:

- Hashing Algorithm: The SHA-256 algorithm is employed for hash generation.
- Encoding: The user's input message is encoded before hashing.
- Hash Value: The resulting hash value is presented in hexadecimal format.

Digital Signature:

The final section involves generating and verifying digital signatures using RSA encryption. Key highlights are:

- RSA Key Pair: An RSA key pair is generated with a key size of 2048 bits, which is a common choice for digital signatures.
- Signing: The sender, referred to as Alice, signs a predefined message using her private key. The message is first hashed using SHA-256.

- Public Key Export: Alice's public key is saved in a 'PUB.SKT' file.
- Verification: The receiver, referred to as Bob, verifies the digital signature of the message using Alice's public key.
- Verification Result: Bob's verification result is displayed, indicating whether the digital signature is valid (authentic) or not.

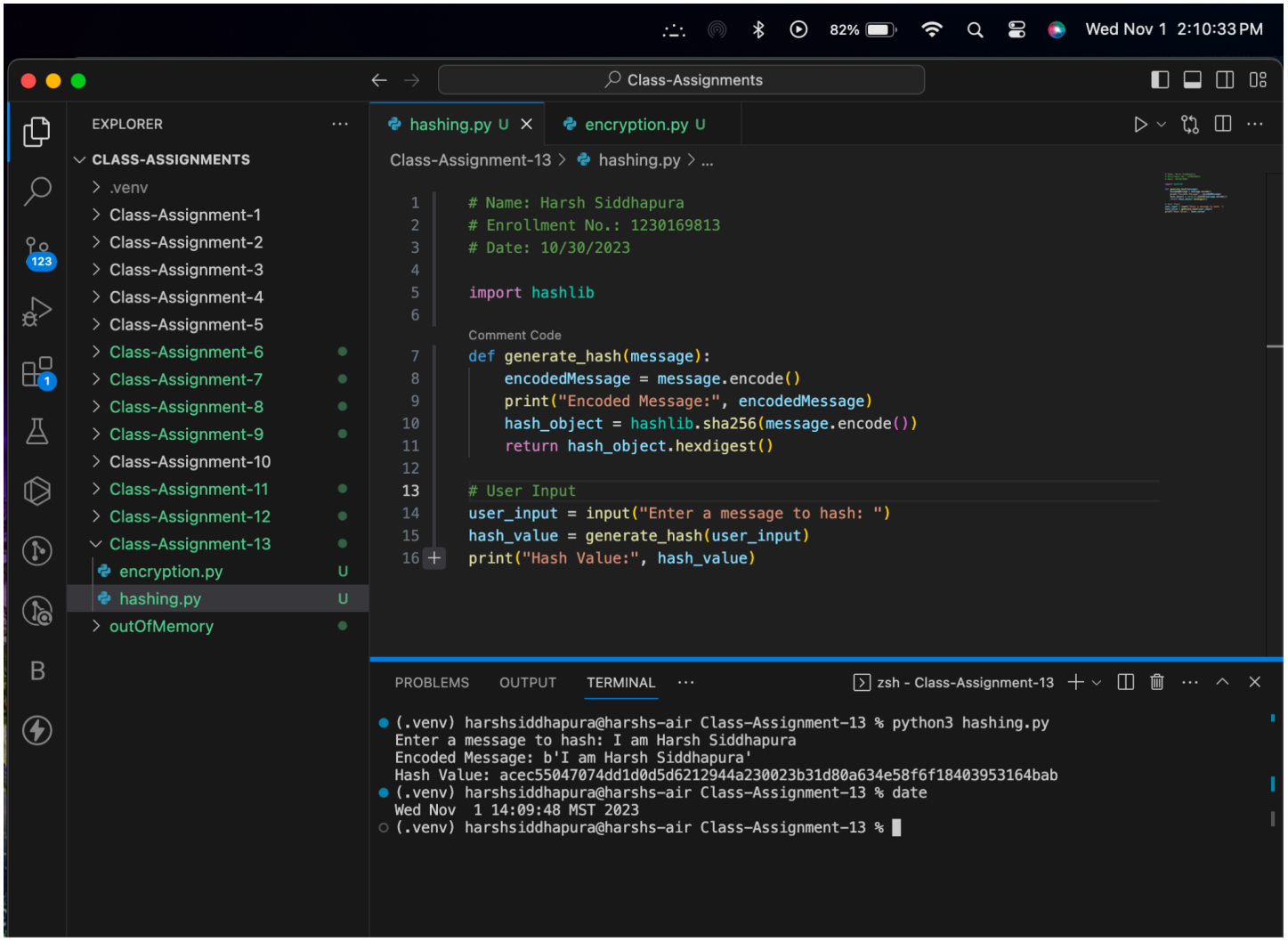
Three fundamental cryptographic operations: AES encryption, hashing using SHA-256, and digital signature generation and verification with RSA. These cryptographic techniques play a pivotal role in modern security practices, serving to ensure data confidentiality, integrity, and authenticity. Below are the key takeaways and implications of these operations:

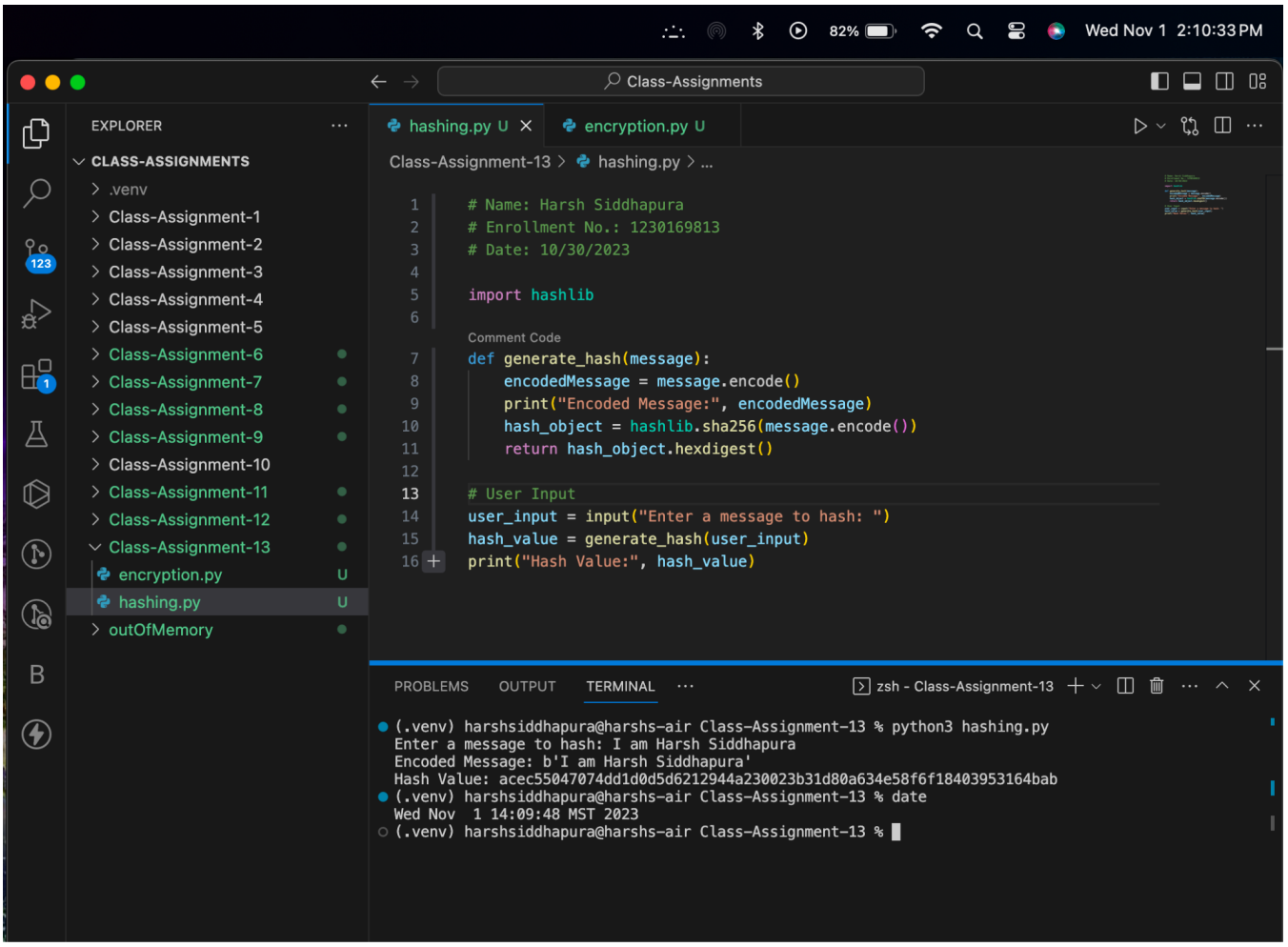
- **AES Encryption:** The use of AES encryption is crucial for maintaining data confidentiality. By encrypting a message with a randomly generated key, the data becomes unintelligible to unauthorized parties. The inclusion of a nonce enhances security by preventing attackers from using replay attacks. This technique is vital for securing data in transit and at rest, such as in secure communication channels or protected data storage.
- **SHA-256 Hashing:** Hash functions are essential for verifying data integrity. SHA-256, a widely used hashing algorithm, takes an input message, encodes it, and produces a fixed-length hash value. This hash value serves as a unique fingerprint for the original data. Any alterations in the input message will result in a substantially different hash value, making it easy to detect tampering. This is crucial for ensuring data hasn't been altered during transmission or storage.
- **RSA Digital Signatures:** Digital signatures are a cornerstone of data authenticity and verification. By signing a message with a private key and subsequently verifying it with a corresponding public key, data recipients can trust that the message is genuine and hasn't been tampered with. The use of RSA, a widely used public-key cryptography algorithm, ensures the security of the digital signature process.

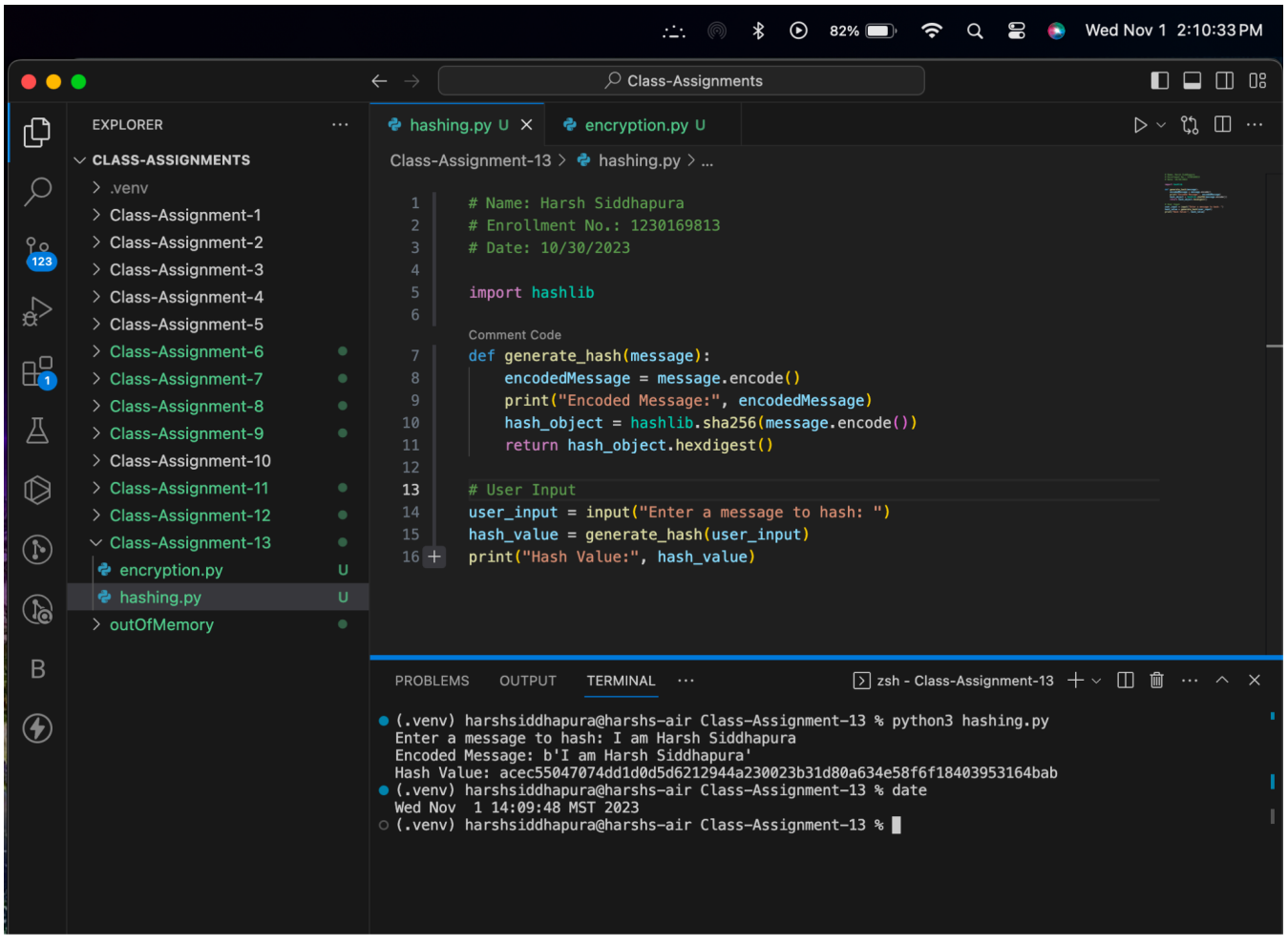
This is especially important for secure communication, where parties need to be certain of the message source.

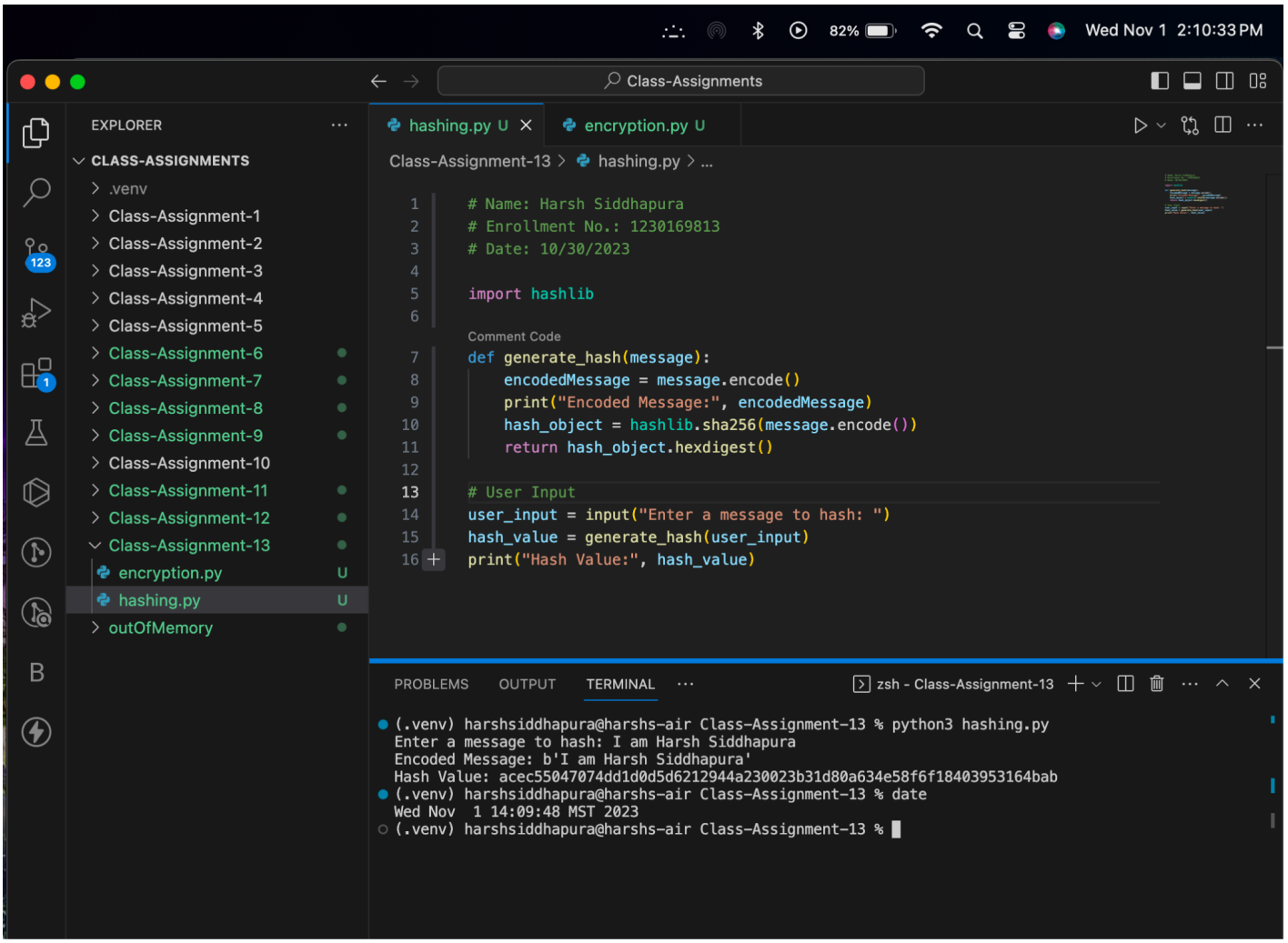
Overall, these cryptographic operations are not isolated practices but are often used together to provide comprehensive security solutions. For example, a secure communication channel might combine AES encryption for confidentiality, SHA-256 hashing for integrity checks, and RSA digital signatures for authenticity assurance. Understanding and effectively implementing these cryptographic techniques is imperative in the realm of cybersecurity, as they safeguard sensitive information, protect against unauthorized access, and ensure the trustworthiness of data and communications. The application of these cryptographic principles extends to a wide range of sectors, from secure messaging and e-commerce to critical infrastructure protection and data privacy compliance.

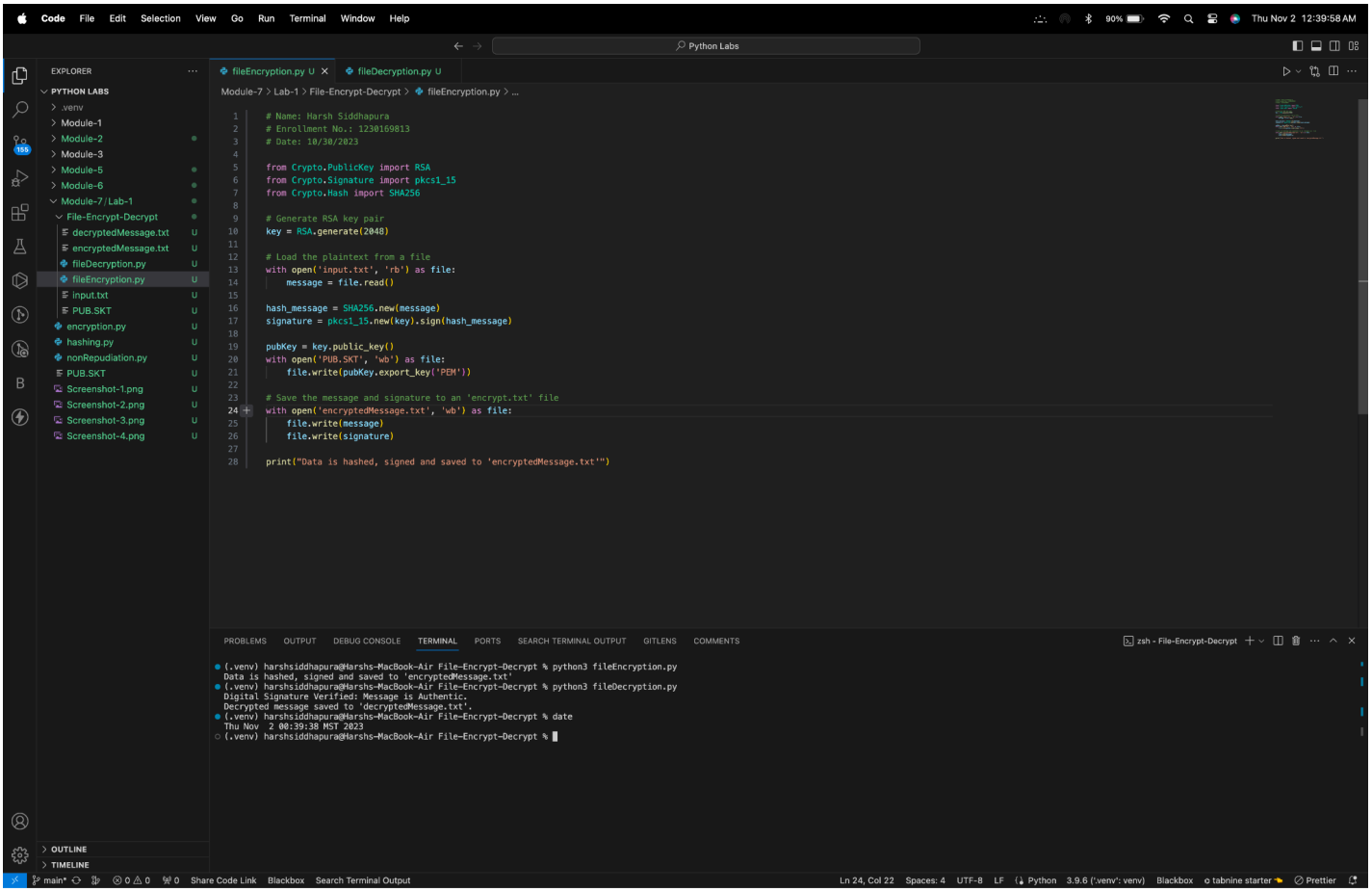
Screenshot

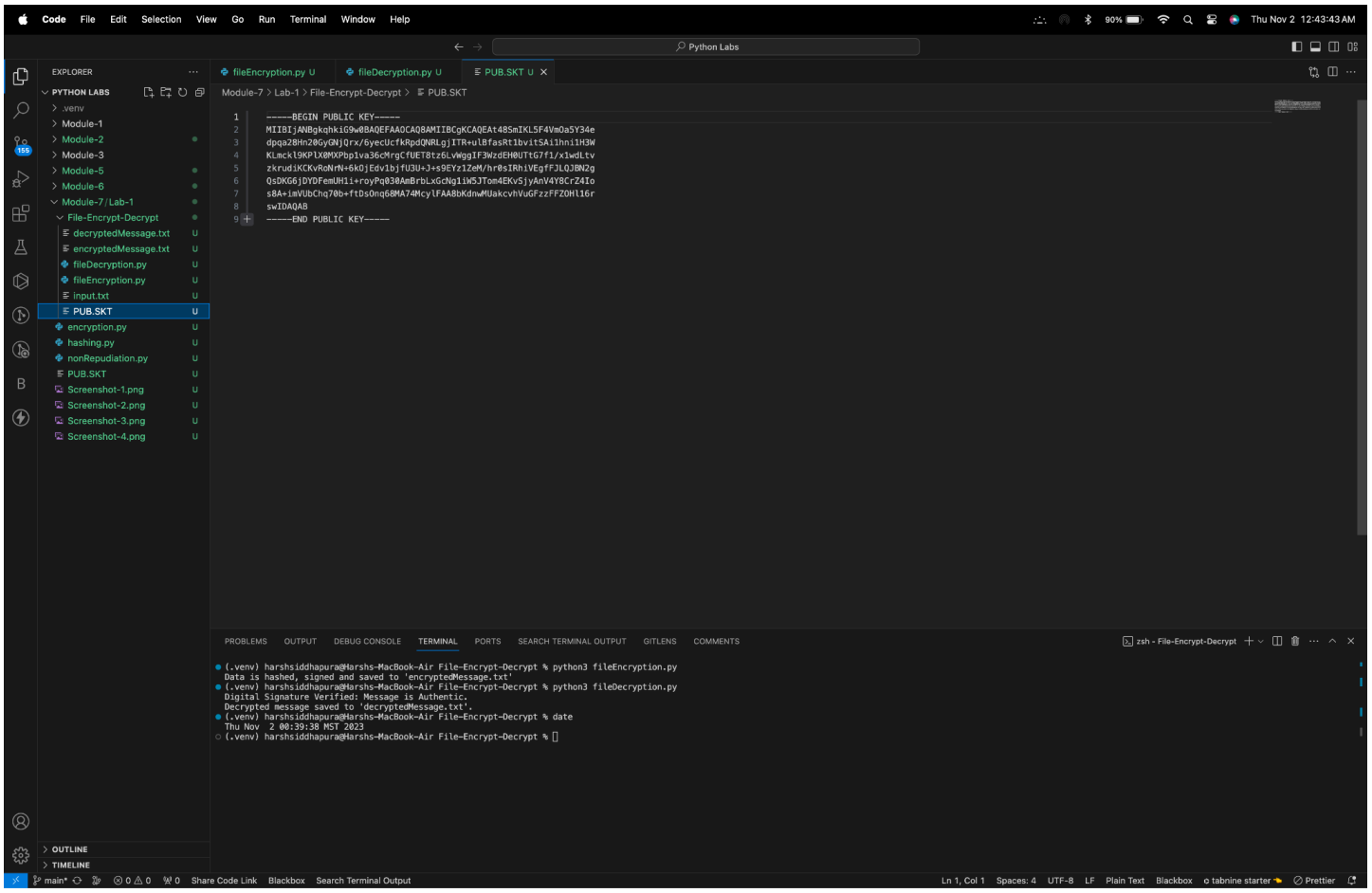


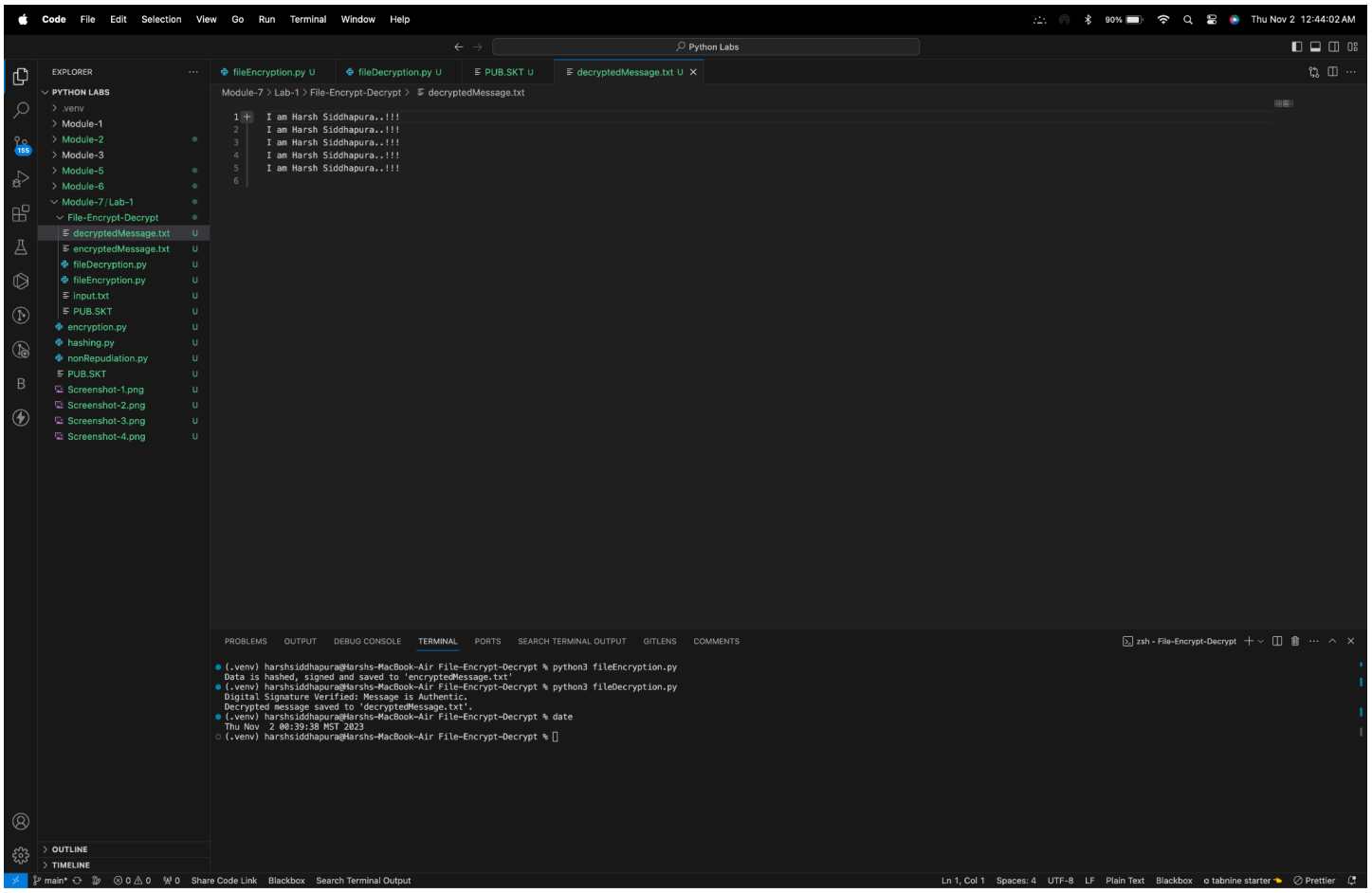

Wed Nov 1 2:10:33 PM
Class-Assignments
EXPLORER
CLASS-ASSIGNMENTS
.venv
Class-Assignment-1
Class-Assignment-2
Class-Assignment-3
Class-Assignment-4
Class-Assignment-5
Class-Assignment-6
Class-Assignment-7
Class-Assignment-8
Class-Assignment-9
Class-Assignment-10
Class-Assignment-11
Class-Assignment-12
Class-Assignment-13
encryption.py U
hashing.py U
outOfMemory
hashing.py U
encryption.py U
Class-Assignment-13 > hashing.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
import hashlib
Comment Code
def generate_hash(message):
encodedMessage = message.encode()
print("Encoded Message:", encodedMessage)
hash_object = hashlib.sha256(message.encode())
return hash_object.hexdigest()
# User Input
user_input = input("Enter a message to hash: ")
hash_value = generate_hash(user_input)
print("Hash Value:", hash_value)
PROBLEMS
OUTPUT
TERMINAL
zsh - Class-Assignment-13
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % python3 hashing.py
Enter a message to hash: I am Harsh Siddhapura
Encoded Message: b'I am Harsh Siddhapura'
Hash Value: acec55047074dd1d0d5d6212944a230023b31d80a634e58f6f18403953164bab
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % date
Wed Nov 1 14:09:48 MST 2023
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 %


Wed Nov 1 2:10:33 PM
Class-Assignments
EXPLORER
CLASS-ASSIGNMENTS
.venv
Class-Assignment-1
Class-Assignment-2
Class-Assignment-3
Class-Assignment-4
Class-Assignment-5
Class-Assignment-6
Class-Assignment-7
Class-Assignment-8
Class-Assignment-9
Class-Assignment-10
Class-Assignment-11
Class-Assignment-12
Class-Assignment-13
encryption.py U
hashing.py U
outOfMemory
hashing.py U
encryption.py U
Class-Assignment-13 > hashing.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
import hashlib
Comment Code
def generate_hash(message):
encodedMessage = message.encode()
print("Encoded Message:", encodedMessage)
hash_object = hashlib.sha256(message.encode())
return hash_object.hexdigest()
# User Input
user_input = input("Enter a message to hash: ")
hash_value = generate_hash(user_input)
print("Hash Value:", hash_value)
PROBLEMS
OUTPUT
TERMINAL
zsh - Class-Assignment-13
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % python3 hashing.py
Enter a message to hash: I am Harsh Siddhapura
Encoded Message: b'I am Harsh Siddhapura'
Hash Value: acec55047074dd1d0d5d6212944a230023b31d80a634e58f6f18403953164bab
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % date
Wed Nov 1 14:09:48 MST 2023
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 %


Wed Nov 1 2:10:33 PM
Class-Assignments
EXPLORER
CLASS-ASSIGNMENTS
.venv
Class-Assignment-1
Class-Assignment-2
Class-Assignment-3
Class-Assignment-4
Class-Assignment-5
Class-Assignment-6
Class-Assignment-7
Class-Assignment-8
Class-Assignment-9
Class-Assignment-10
Class-Assignment-11
Class-Assignment-12
Class-Assignment-13
encryption.py U
hashing.py U
outOfMemory
hashing.py U
encryption.py U
Class-Assignment-13 > hashing.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
import hashlib
Comment Code
def generate_hash(message):
encodedMessage = message.encode()
print("Encoded Message:", encodedMessage)
hash_object = hashlib.sha256(message.encode())
return hash_object.hexdigest()
# User Input
user_input = input("Enter a message to hash: ")
hash_value = generate_hash(user_input)
print("Hash Value:", hash_value)
PROBLEMS
OUTPUT
TERMINAL
zsh - Class-Assignment-13
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % python3 hashing.py
Enter a message to hash: I am Harsh Siddhapura
Encoded Message: b'I am Harsh Siddhapura'
Hash Value: acec55047074dd1d0d5d6212944a230023b31d80a634e58f6f18403953164bab
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 % date
Wed Nov 1 14:09:48 MST 2023
(.venv) harshsiddhapura@harshs-air Class-Assignment-13 %


Code File Edit Selection View Go Run Terminal Window Help
Thu Nov 2 12:39:58 AM
Python Labs
EXPLORER
PYTHON LABS
fileEncryption.py U
fileDecryption.py U
Module-7 > Lab-1 > File-Encrypt-Decrypt > fileEncryption.py > ...
# Name: Harsh Siddhapura
# Enrollment No.: 1230169813
# Date: 10/30/2023
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256
# Generate RSA key pair
key = RSA.generate(2048)
# Load the plaintext from a file
with open('input.txt', 'rb') as file:
message = file.read()
hash_message = SHA256.new(message)
signature = pkcs1_15.new(key).sign(hash_message)
pubKey = key.public_key()
with open('PUB.SKT', 'wb') as file:
file.write(pubKey.export_key('PEM'))
# Save the message and signature to an 'encrypt.txt' file
with open('encryptedMessage.txt', 'wb') as file:
file.write(message)
file.write(signature)
print("Data is hashed, signed and saved to 'encryptedMessage.txt'")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - File-Encrypt-Decrypt
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileEncryption.py
Data is hashed, signed and saved to 'encryptedMessage.txt'
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileDecryption.py
Digital Signature Verified: Message is Authentic.
Decrypted message saved to 'decryptedMessage.txt'.
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % date
Thu Nov 2 00:39:38 MST 2023
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt %


Code File Edit Selection View Go Run Terminal Window Help
Thu Nov 2 12:43:43 AM
Python Labs
EXPLORER
PYTHON LABS
fileEncryption.py U
fileDecryption.py U
PUB.SKT U
Module-7 > Lab-1 > File-Encrypt-Decrypt > PUB.SKT
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt4B5mIKL5F4VmOa5Y34e
dpqa28Hn20GyGNjQrx/6yecUcfkRpdQNRLgjITR+ulBfasRt1bvitSAi1hni1H3W
KLmckl9KPlX0MXPbp1va36cMrgCfUET8tz6LvWggIF3WzdEH0UTtG7f1/x1wdLtv
zkrudiKCkvRoNrNv6kOjEdv1bjfU3U+Js9EYz1ZeM/kr0sIRh1VEgfFJLQJBWZg
QsDKG6jDYOFemUH1i+royPq030AmBrbLxGcNg1iW5JTom4EKv5jyAnV4Y8CrZ4Io
s8A+imVUbChq70b+ftDsOnq68MA74McylFAA8bKdnwMUakcvhVuGFzzFFZOHl16r
swIDAQAB
-----END PUBLIC KEY-----
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - File-Encrypt-Decrypt
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileEncryption.py
Data is hashed, signed and saved to 'encryptedMessage.txt'
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileDecryption.py
Digital Signature Verified: Message is Authentic.
Decrypted message saved to 'decryptedMessage.txt'.
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % date
Thu Nov 2 00:39:38 MST 2023
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt %
OUTLINE
TIMELINE
main*
Share Code Link Blackbox Search Terminal Output
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Blackbox tabnine starter Prettier


Code File Edit Selection View Go Run Terminal Window Help
Thu Nov 2 12:44:02 AM
Python Labs
EXPLORER
PYTHON LABS
fileEncryption.py U
fileDecryption.py U
PUB.SKT U
decryptedMessage.txt U
Module-7 > Lab-1 > File-Encrypt-Decrypt > decryptedMessage.txt
I am Harsh Siddhapura..!!!
I am Harsh Siddhapura..!!!
I am Harsh Siddhapura..!!!
I am Harsh Siddhapura..!!!
I am Harsh Siddhapura..!!!
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT GITLENS COMMENTS
zsh - File-Encrypt-Decrypt
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileEncryption.py
Data is hashed, signed and saved to 'encryptedMessage.txt'
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % python3 fileDecryption.py
Digital Signature Verified: Message is Authentic.
Decrypted message saved to 'decryptedMessage.txt'.
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt % date
Thu Nov 2 00:39:38 MST 2023
(.venv) harshsiddhapura@Harshs-MacBook-Air File-Encrypt-Decrypt %
OUTLINE
TIMELINE
main*
Share Code Link Blackbox Search Terminal Output
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text Blackbox tabnine starter Prettier