INSTITUTE OF COMPUTER TECHNOLOGY

B-TECH COMPUTER SCIENCE ENGINEERING 2025-26

SUBJECT: CRYPTOGRAPHY

NAME: Rahul Prajapati

ENRLL. NO: 23162171020

BRANCH: CYBER SECURITY

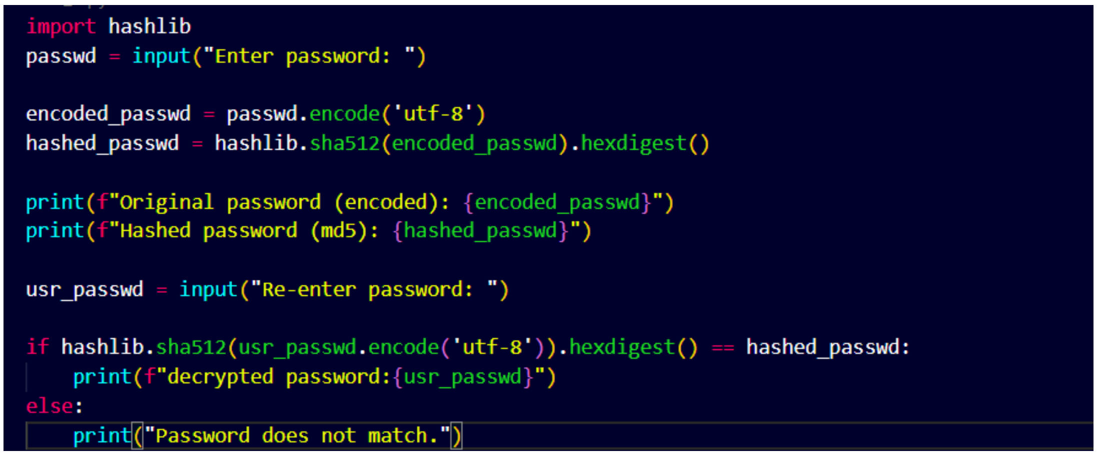
BATCH: 52

PRACTICAL_2

Aim:-To compare various encryption methods used by classmates by exchanging ciphertexts and attempting decryption using both their original decryption logic and third-party tools, in order to evaluate the effectiveness and reversibility of different encryption techniques.

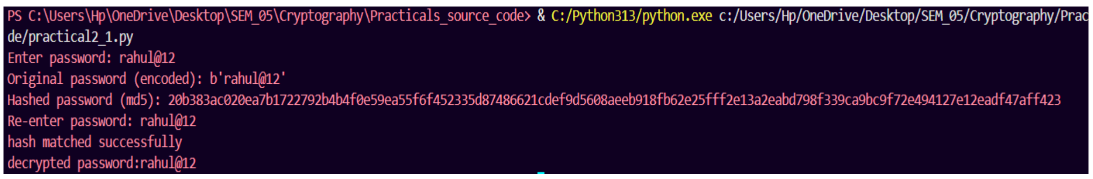
Method_1:

Code:



```python
import hashlib
passwd = input("Enter password: ")

encoded_passwd = passwd.encode('utf-8')
hashed_passwd = hashlib.sha512(encoded_passwd).hexdigest()

print(f"Original password (encoded): {encoded_passwd}")
print(f"Hashed password (md5): {hashed_passwd}")

usr_passwd = input("Re-enter password: ")

if hashlib.sha512(usr_passwd.encode('utf-8')).hexdigest() == hashed_passwd:
    print(f"decrypted password:{usr_passwd}")
else:
    print("Password does not match.")
```

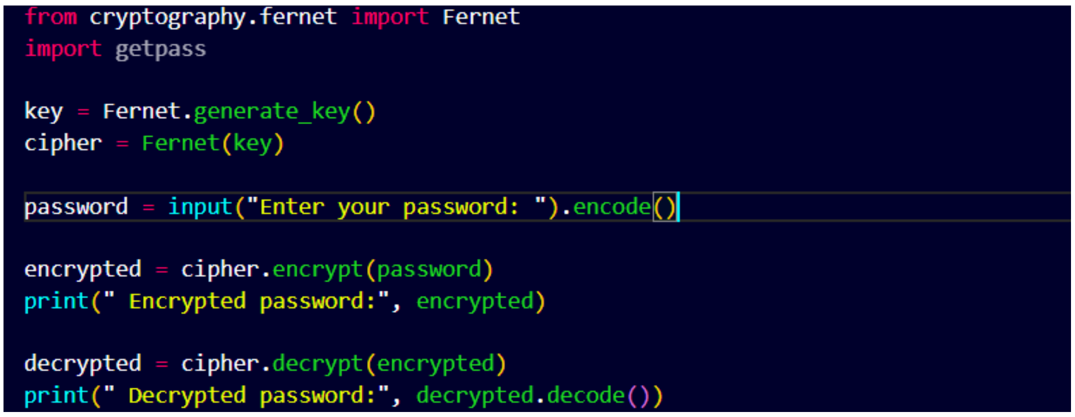
Output:



```
PS C:\Users\Hp\OneDrive\Desktop\SEM_05\Cryptography\Practicals_source_code> & C:/Python313/python.exe c:/Users/Hp/OneDrive/Desktop/SEM_05/Cryptography/Prac
de/practical2_1.py
Enter password: rahul@12
Original password (encoded): b'rahul@12'
Hashed password (md5): 20b383ac020ea7b1722792b4b4f0e59ea55f6f452335d87486621cdef9d5608aeeb918fb62e25fff2e13a2eabd798f339ca9bc9f72e494127e12eadf47aff423
Re-enter password: rahul@12
hash matched successfully
decrypted password:rahul@12
```

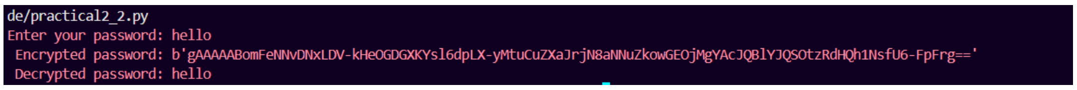
Method_2:

Code:-



```
from cryptography.fernet import Fernet
import getpass

key = Fernet.generate_key()
cipher = Fernet(key)

password = input("Enter your password: ").encode()

encrypted = cipher.encrypt(password)
print(" Encrypted password:", encrypted)

decrypted = cipher.decrypt(encrypted)
print(" Decrypted password:", decrypted.decode())
```

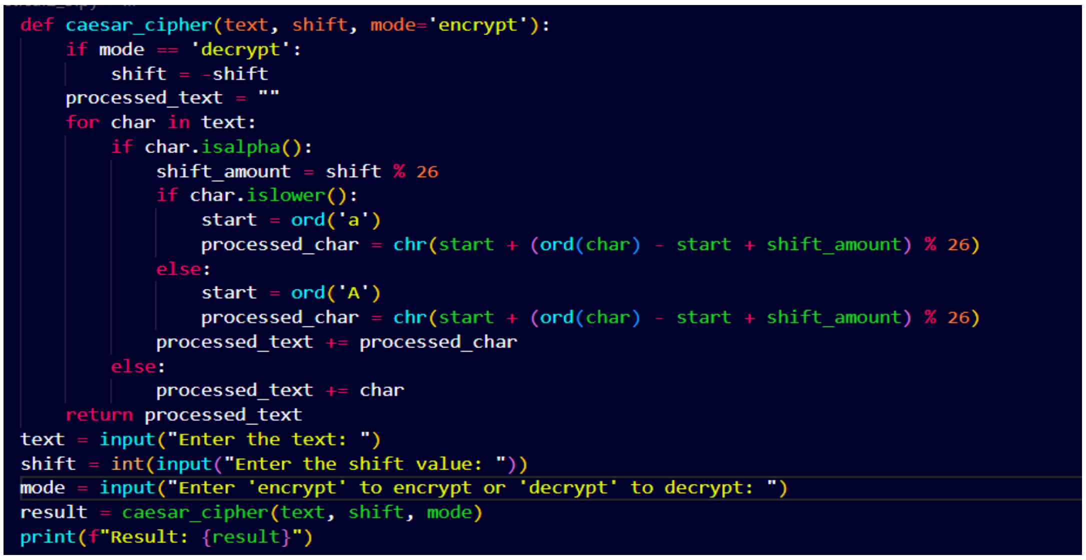
Output:-



```
de/practical2_2.py
Enter your password: hello
 Encrypted password: b'gAAAAABomFeNNvDNxLDV-kHeOGDGXKYsl6dpLX-yMtuCuZXaJrjN8aNNuZkowGEOjMgYAcJQBlYJQSOtzRdHQh1NsfU6-FpFrg=='
 Decrypted password: hello
```

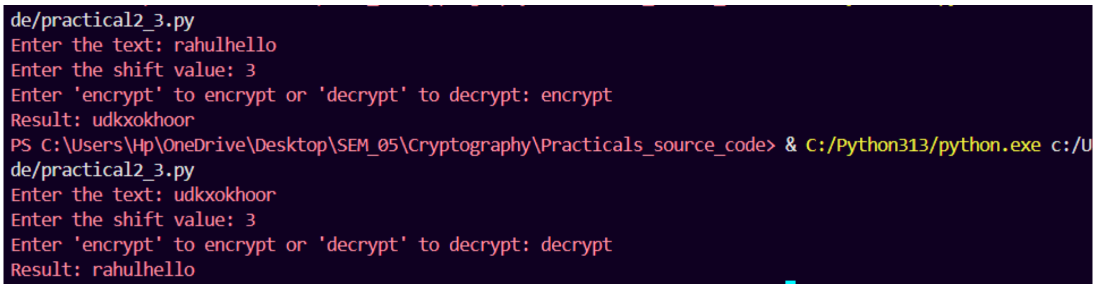
Method_3:

Code:



```
def caesar_cipher(text, shift, mode='encrypt'):
    if mode == 'decrypt':
        shift = -shift
    processed_text = ""
    for char in text:
        if char.isalpha():
            shift_amount = shift % 26
            if char.islower():
                start = ord('a')
                processed_char = chr(start + (ord(char) - start + shift_amount) % 26)
            else:
                start = ord('A')
                processed_char = chr(start + (ord(char) - start + shift_amount) % 26)
            processed_text += processed_char
        else:
            processed_text += char
    return processed_text
text = input("Enter the text: ")
shift = int(input("Enter the shift value: "))
mode = input("Enter 'encrypt' to encrypt or 'decrypt' to decrypt: ")
result = caesar_cipher(text, shift, mode)
print(f"Result: {result}")
```

Output:



```
de/practical2_3.py
Enter the text: rahulhello
Enter the shift value: 3
Enter 'encrypt' to encrypt or 'decrypt' to decrypt: encrypt
Result: udkxokhoor
PS C:\Users\Hp\OneDrive\Desktop\SEM_05\Cryptography\Practicals_source_code> & C:/Python313/python.exe c:/U
de/practical2_3.py
Enter the text: udkxokhoor
Enter the shift value: 3
Enter 'encrypt' to encrypt or 'decrypt' to decrypt: decrypt
Result: rahulhello
```

1. Analysis – Which was stronger and when it failed

Method 1 – SHA512 “encryption”

- Here we take the password, turn it into a fixed scrambled code.
- When we “decrypt” in your code, we’re actually just checking if the scrambled version matches the original scrambled one.
- In your example, it works fine if the password is correct, but fails instantly if it’s even 1 letter wrong.
- Strength-wise, it’s much harder to break than Caesar Cipher, but weaker than Fernet if you actually need to get the original data back.

Method 2 – Fernet encryption

- Proper encryption. You can lock (encrypt) and unlock (decrypt) perfectly if you have the key.
- In your code, it worked every time because the same key was used right away.
- Security is very high, but only if you keep the key safe.

Method 3 – Caesar Cipher

- Very old-school and weak.
- Anyone can break it in seconds using trial and error or an online decoder.
- It still works for both encrypt and decrypt in the code, but not safe at all in real life.

2. Summary – Why this matters in real life

- **Most secure overall:** Method 2 (Fernet) – solid protection and can get your data back.
- **Good for checking passwords but not for actual data retrieval:** Method 1.
- **Fun but useless for real protection:** Method 3.

- In real life, if you choose a weak method like Caesar Cipher, it’s like locking your phone with “1234” anyone can open it.
- If you use Fernet or a strong hash, it’s like having a fingerprint lock plus a PIN.

3. Limitations

Method 1 – SHA512

- **Easy to break?** Not really, but weak passwords still can be guessed with tools.
- **Key problem?** No key, so you can’t “actually” decrypt — only compare.
- **Data type?** Works for text but not great for other data like images unless converted first.
- **Real life?** Good for storing and checking passwords, not for retrieving original stuff.

Method 2 – Fernet

- **Easy to break?** Nope, unless someone gets your key.
- **Key problem?** Lose the key = lose the data forever.
- **Data type?** Works with any type of data.
- **Real life?** Perfect for protecting files, emails, or any sensitive info.

Method 3 – Caesar Cipher

- **Easy to break?** Yep, super easy.
- **Key problem?** The “key” (shift number) is so small it can be guessed in seconds.
- **Data type?** Only works properly with letters.
- **Real life?** Only for games or secret notes between kids.