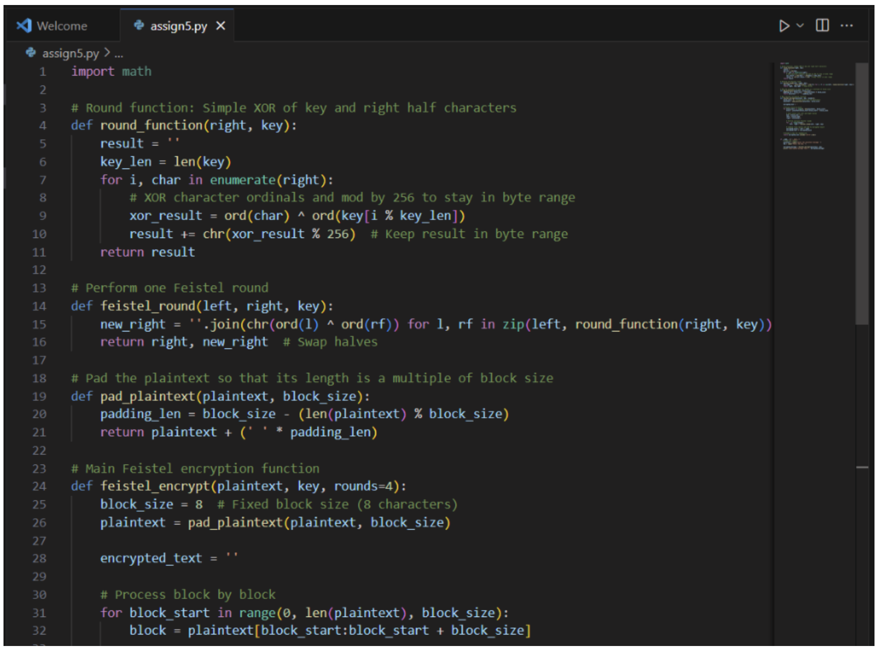
CRYPTOGRAPHY & COMPUTER NETWORK

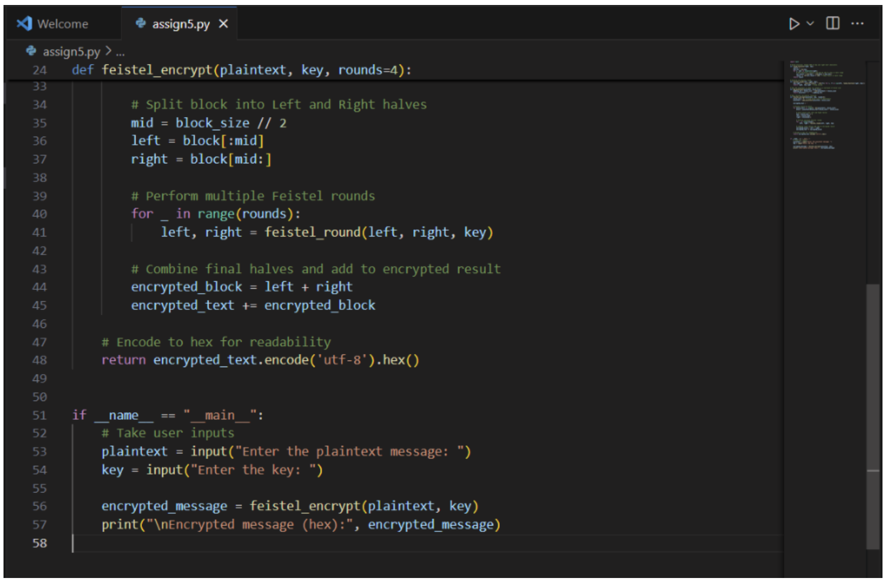
ASSIGNMENT- 5

Name : N. Sanjay
HT No : 2303A51LA4
Batch : 30

- Implement a Feistel cipher encryption method in Python as a backend utility that demonstrates basic cryptographic understanding, string manipulation, and block-based encryption.

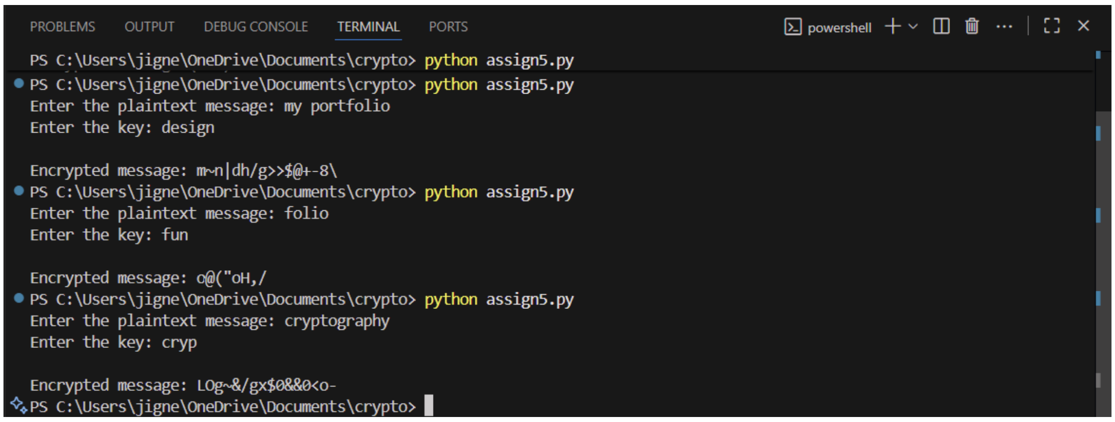


```python
import math

# Round function: Simple XOR of key and right half characters
def round_function(right, key):
    result = ''
    key_len = len(key)
    for i, char in enumerate(right):
        # XOR character ordinals and mod by 256 to stay in byte range
        xor_result = ord(char) ^ ord(key[i % key_len])
        result += chr(xor_result % 256)  # Keep result in byte range
    return result

# Perform one Feistel round
def feistel_round(left, right, key):
    new_right = ''.join(chr(ord(l) ^ ord(rf)) for l, rf in zip(left, round_function(right, key))
    return right, new_right  # Swap halves

# Pad the plaintext so that its length is a multiple of block size
def pad_plaintext(plaintext, block_size):
    padding_len = block_size - (len(plaintext) % block_size)
    return plaintext + (' ' * padding_len)

# Main Feistel encryption function
def feistel_encrypt(plaintext, key, rounds=4):
    block_size = 8  # Fixed block size (8 characters)
    plaintext = pad_plaintext(plaintext, block_size)

    encrypted_text = ''

    # Process block by block
    for block_start in range(0, len(plaintext), block_size):
        block = plaintext[block_start:block_start + block_size]
```



```python
def feistel_encrypt(plaintext, key, rounds=4):

        # Split block into Left and Right halves
        mid = block_size // 2
        left = block[:mid]
        right = block[mid:]

        # Perform multiple Feistel rounds
        for _ in range(rounds):
            left, right = feistel_round(left, right, key)

        # Combine final halves and add to encrypted result
        encrypted_block = left + right
        encrypted_text += encrypted_block

    # Encode to hex for readability
    return encrypted_text.encode('utf-8').hex()


if __name__ == "__main__":
    # Take user inputs
    plaintext = input("Enter the plaintext message: ")
    key = input("Enter the key: ")

    encrypted_message = feistel_encrypt(plaintext, key)
    print("\nEncrypted message (hex):", encrypted_message)

```

OUTPUT :



```
PS C:\Users\jigne\OneDrive\Documents\crypto> python assign5.py
PS C:\Users\jigne\OneDrive\Documents\crypto> python assign5.py
Enter the plaintext message: my portfolio
Enter the key: design

Encrypted message: m~n|dh/g>>$@+-8\
PS C:\Users\jigne\OneDrive\Documents\crypto> python assign5.py
Enter the plaintext message: folio
Enter the key: fun

Encrypted message: o@("oH,/
PS C:\Users\jigne\OneDrive\Documents\crypto> python assign5.py
Enter the plaintext message: cryptography
Enter the key: cryp

Encrypted message: LOg~&/gx$0&&0<o-
PS C:\Users\jigne\OneDrive\Documents\crypto>
```