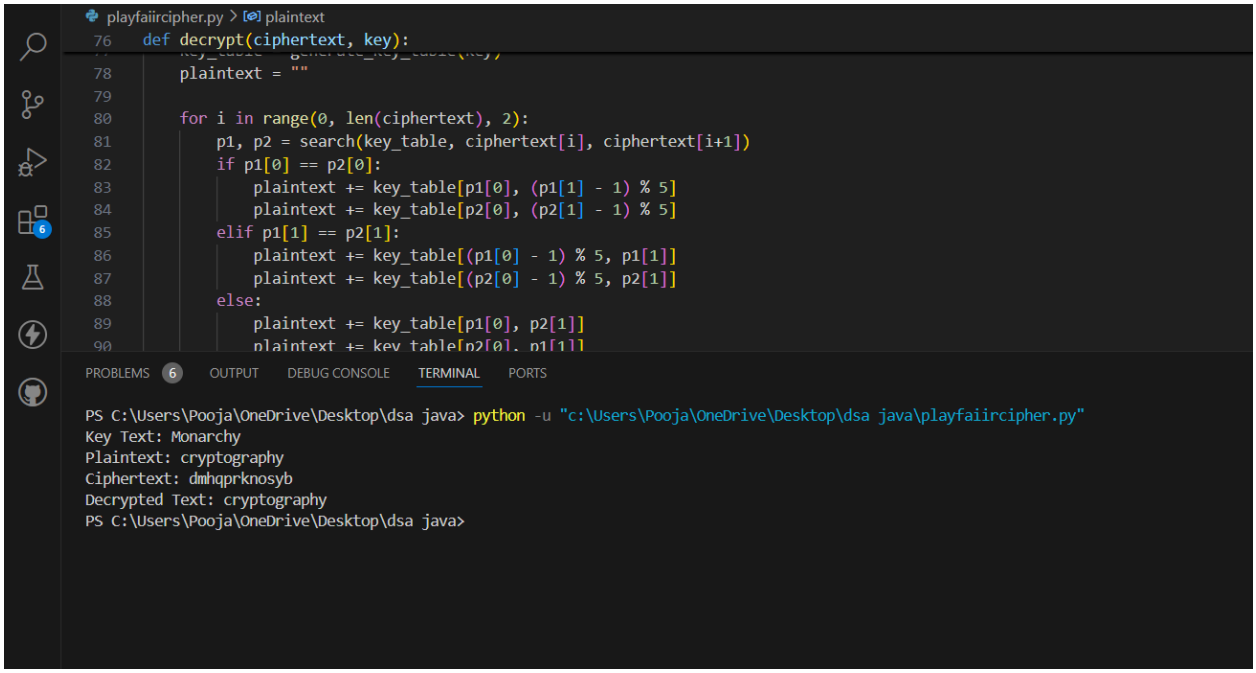
```
# To write a python program to implement Playfair cipher for encryption and decryption
import numpy as np
def to_lower_case(text):
    return text.lower()

def remove_spaces(text):
    return text.replace(" ", "")

def generate_key_table(key):
    key = remove_spaces(to_lower_case(key))
    key = key.replace('j', 'i')
    key = ''.join(dict.fromkeys(key))  # Remove duplicate letters

    alphabet = "abcdefghiklmnopqrstuvwxyz"  # 'j' is excluded
    key_table = [c for c in key if c in alphabet]

    for char in alphabet:
        if char not in key_table:
            key_table.append(char)

    key_table = np.array(key_table).reshape(5, 5)
    return key_table

def search(key_table, a, b):
    if a == 'j':
        a = 'i'
    if b == 'j':
```

```
            b = 'i'

        p1 = p2 = None
        for i in range(5):
            for j in range(5):
                if key_table[i, j] == a:
                    p1 = (i, j)
                elif key_table[i, j] == b:
                    p2 = (i, j)
        return p1, p2


def prepare_text(text, filler='x'):
    text = remove_spaces(to_lower_case(text)).replace('j', 'i')
    new_text = ""
    i = 0
    while i < len(text):
        a = text[i]
        if i + 1 < len(text):
            b = text[i + 1]
        else:
            b = 'z'
        if a != b:
            new_text += a + b
            i += 2
        else:
            new_text += a + filler
            i += 1
```

```
    return new_text

def encrypt(plaintext, key):
    key_table = generate_key_table(key)
    plaintext = prepare_text(plaintext)
    ciphertext = ""

    for i in range(0, len(plaintext), 2):
        p1, p2 = search(key_table, plaintext[i], plaintext[i+1])
        if p1[0] == p2[0]:
            ciphertext += key_table[p1[0], (p1[1] + 1) % 5]
            ciphertext += key_table[p2[0], (p2[1] + 1) % 5]
        elif p1[1] == p2[1]:
            ciphertext += key_table[(p1[0] + 1) % 5, p1[1]]
            ciphertext += key_table[(p2[0] + 1) % 5, p2[1]]
        else:
            ciphertext += key_table[p1[0], p2[1]]
            ciphertext += key_table[p2[0], p1[1]]

    return ciphertext

def decrypt(ciphertext, key):
    key_table = generate_key_table(key)
    plaintext = ""

    for i in range(0, len(ciphertext), 2):
        p1, p2 = search(key_table, ciphertext[i], ciphertext[i+1])
```

```
        if p1[0] == p2[0]:
            plaintext += key_table[p1[0], (p1[1] - 1) % 5]
            plaintext += key_table[p2[0], (p2[1] - 1) % 5]
        elif p1[1] == p2[1]:
            plaintext += key_table[(p1[0] - 1) % 5, p1[1]]
            plaintext += key_table[(p2[0] - 1) % 5, p2[1]]
        else:
            plaintext += key_table[p1[0], p2[1]]
            plaintext += key_table[p2[0], p1[1]]

    return plaintext

# Example usage
key = "Monarchy"
plaintext = "cryptography"
ciphertext = encrypt(plaintext, key)
decrypted_text = decrypt(ciphertext, key)

print("Key Text:", key)
print("Plaintext:", plaintext)
print("Ciphertext:", ciphertext)
print("Decrypted Text:", decrypted_text)
```


playfaiircipher.py > plaintext
def decrypt(ciphertext, key):
plaintext = ""
for i in range(0, len(ciphertext), 2):
p1, p2 = search(key_table, ciphertext[i], ciphertext[i+1])
if p1[0] == p2[0]:
plaintext += key_table[p1[0], (p1[1] - 1) % 5]
plaintext += key_table[p2[0], (p2[1] - 1) % 5]
elif p1[1] == p2[1]:
plaintext += key_table[(p1[0] - 1) % 5, p1[1]]
plaintext += key_table[(p2[0] - 1) % 5, p2[1]]
else:
plaintext += key_table[p1[0], p2[1]]
PROBLEMS 6
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\Pooja\OneDrive\Desktop\dsa java> python -u "c:\Users\Pooja\OneDrive\Desktop\dsa java\playfaiircipher.py"
Key Text: Monarchy
Plaintext: cryptography
Ciphertext: dmhqprknosyb
Decrypted Text: cryptography
PS C:\Users\Pooja\OneDrive\Desktop\dsa java>