**LAB Assignment 2**

**Theory:**

The Playfair cipher is a digraph substitution cipher that encrypts pairs of letters using a 5x5 grid constructed from a keyword. Each pair of letters is encrypted based on their positions in the grid. If the letters are in the same row, each letter is replaced by the letter to its immediate right. If they are in the same column, each letter is replaced by the letter directly below. If they form a rectangle, each letter is replaced by the letter in its own row but in the column of the other letter of the pair.

Example: For plaintext **"HELLO"** with the keyword **"KEYWORD"**, the grid might look like:  
  
K E Y W O

R D A B C

F G H I L

M N P Q S

T U V X Z

"MO" will become "SK" after encryption. To decrypt, we reverse the process using the same grid.

**Code:**

**Playfair Cipher:**

def playfair\_cipher(plaintext, key, mode):

alphabet = 'abcdefghiklmnopqrstuvwxyz'

key = key.lower().replace(' ', '').replace('j', 'i')

key\_square = ''

for letter in key + alphabet:

if letter not in key\_square:

key\_square += letter

plaintext = plaintext.lower().replace(' ', '').replace('j', 'i')

if len(plaintext) % 2 == 1:

plaintext += 'x'

digraphs = [plaintext[i:i+2] for i in range(0, len(plaintext), 2)]

def encrypt(digraph):

a, b = digraph

row\_a, col\_a = divmod(key\_square.index(a), 5)

row\_b, col\_b = divmod(key\_square.index(b), 5)

if row\_a == row\_b:

col\_a = (col\_a + 1) % 5

col\_b = (col\_b + 1) % 5

elif col\_a == col\_b:

row\_a = (row\_a + 1) % 5

row\_b = (row\_b + 1) % 5

else:

col\_a, col\_b = col\_b, col\_a

return key\_square[row\_a\*5+col\_a] + key\_square[row\_b\*5+col\_b]

def decrypt(digraph):

a, b = digraph

row\_a, col\_a = divmod(key\_square.index(a), 5)

row\_b, col\_b = divmod(key\_square.index(b), 5)

if row\_a == row\_b:

col\_a = (col\_a - 1) % 5

col\_b = (col\_b - 1) % 5

elif col\_a == col\_b:

row\_a = (row\_a - 1) % 5

row\_b = (row\_b - 1) % 5

else:

col\_a, col\_b = col\_b, col\_a

return key\_square[row\_a\*5+col\_a] + key\_square[row\_b\*5+col\_b]

result = ''

for digraph in digraphs:

if mode == 'encrypt':

result += encrypt(digraph)

elif mode == 'decrypt':

result += decrypt(digraph)

return result

plaintext = input("Enter your text: ")

key = input("Enter your key: ")

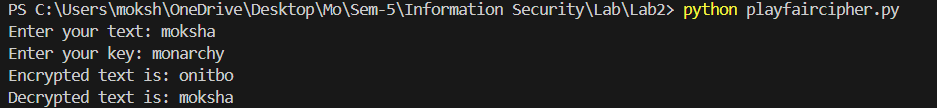
ciphertext = playfair\_cipher(plaintext, key, 'encrypt')

print("Encrypted text is:", ciphertext)

decrypted\_text = playfair\_cipher(ciphertext, key, 'decrypt')

print("Decrypted text is:", decrypted\_text)

**Output:**



**Diagonal Playfair Cipher (my modification):**

**Theory:**

The Diagonal Playfair cipher is a variation of the traditional Playfair cipher, where diagonal movements within a key square are used for encryption. The key square is created by arranging a keyword followed by the remaining letters of the alphabet, with 'j' replaced by 'i'. The plaintext is prepared by converting it to lowercase, removing spaces, and ensuring an even number of characters. Digraphs (pairs of letters) are then encrypted by shifting diagonally within the key square. Decryption reverses this process, making the cipher more complex than the standard Playfair cipher while maintaining its fundamental principles.

**Code:**

def diagonal\_playfair\_cipher(plaintext, key, mode):

    alphabet = 'abcdefghiklmnopqrstuvwxyz'

    key = key.lower().replace(' ', '').replace('j', 'i')

    key\_square = ''

    for letter in key + alphabet:

        if letter not in key\_square:

            key\_square += letter

    plaintext = plaintext.lower().replace(' ', '').replace('j', 'i')

    if len(plaintext) % 2 == 1:

        plaintext += 'x'

    digraphs = [plaintext[i:i+2] for i in range(0, len(plaintext), 2)]

    def find\_position(letter):

        index = key\_square.index(letter)

        return divmod(index, 5)

    def encrypt(digraph):

        a, b = digraph

        row\_a, col\_a = find\_position(a)

        row\_b, col\_b = find\_position(b)

        if row\_a == row\_b:

            row\_a = (row\_a - 1) % 5

            col\_a = (col\_a + 1) % 5

            row\_b = (row\_b - 1) % 5

            col\_b = (col\_b + 1) % 5

        elif col\_a == col\_b:

            row\_a = (row\_a + 1) % 5

            col\_a = (col\_a + 1) % 5

            row\_b = (row\_b + 1) % 5

            col\_b = (col\_b + 1) % 5

        else:

            row\_a, col\_a = (row\_a - 1) % 5, (col\_a + 1) % 5

            row\_b, col\_b = (row\_b - 1) % 5, (col\_b + 1) % 5

        return key\_square[row\_a\*5 + col\_a] + key\_square[row\_b\*5 + col\_b]

    def decrypt(digraph):

        a, b = digraph

        row\_a, col\_a = find\_position(a)

        row\_b, col\_b = find\_position(b)

        if row\_a == row\_b:

            row\_a = (row\_a + 1) % 5

            col\_a = (col\_a - 1) % 5

            row\_b = (row\_b + 1) % 5

            col\_b = (col\_b - 1) % 5

        elif col\_a == col\_b:

            row\_a = (row\_a - 1) % 5

            col\_a = (col\_a - 1) % 5

            row\_b = (row\_b - 1) % 5

            col\_b = (col\_b - 1) % 5

        else:

            row\_a, col\_a = (row\_a + 1) % 5, (col\_a - 1) % 5

            row\_b, col\_b = (row\_b + 1) % 5, (col\_b - 1) % 5

        return key\_square[row\_a\*5 + col\_a] + key\_square[row\_b\*5 + col\_b]

    result = ''

    for digraph in digraphs:

        if mode == 'encrypt':

            result += encrypt(digraph)

        elif mode == 'decrypt':

            result += decrypt(digraph)

    return result

plaintext = 'Moksha Dave'

key = ''

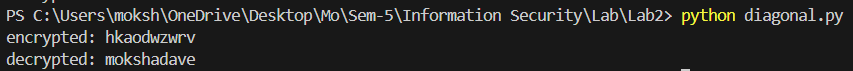
ciphertext = diagonal\_playfair\_cipher(plaintext, key, 'encrypt')

print("encrypted:",ciphertext)

decrypted\_text = diagonal\_playfair\_cipher(ciphertext, key, 'decrypt')

print("decrypted:",decrypted\_text)

**Output:**

****