EXPERIMENT 1

Shashwat Shah
TYBtech Comps B
C22
60004220126

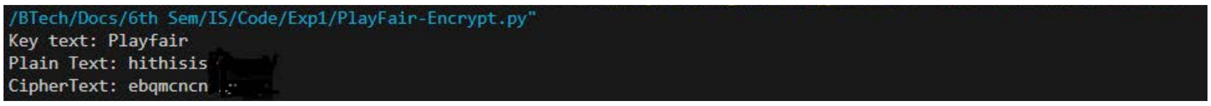
AIM: Study and Implement Playfair Cipher.

ENCRYPTION:

CODE:

```
def toLowerCase(text):
    return text.lower()

def removeSpaces(text):
    newText = ""
    for i in text:
        if i == " ":
            continue
        else:
            newText = newText + i
    return newText

def Diagraph(text):
    Diagraph = []
    group = 0
    for i in range(2, len(text), 2):
        Diagraph.append(text[group:i])

        group = i
    Diagraph.append(text[group:])
    return Diagraph

def FillerLetter(text):
    k = len(text)
    if k % 2 == 0:
        for i in range(0, k, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    else:
        for i in range(0, k-1, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    return new_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):
    key_letters = []
    for i in word:
        if i not in key_letters:
            key_letters.append(i)

    compElements = []
    for i in key_letters:
        if i not in compElements:
            compElements.append(i)
    for i in list1:
        if i not in compElements:
            compElements.append(i)

    matrix = []
    while compElements != []:
        matrix.append(compElements[:5])
        compElements = compElements[5:]

    return matrix


def search(mat, element):
    for i in range(5):
        for j in range(5):
            if(mat[i][j] == element):
                return i, j


def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 4:
        char1 = matr[e1r][0]
    else:
        char1 = matr[e1r][e1c+1]

    char2 = ''
    if e2c == 4:
        char2 = matr[e2r][0]
    else:
        char2 = matr[e2r][e2c+1]

    return char1, char2


def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 4:
        char1 = matr[0][e1c]
    else:
        char1 = matr[e1r+1][e1c]

    char2 = ''
    if e2r == 4:
        char2 = matr[0][e2c]
    else:
        char2 = matr[e2r+1][e2c]

    return char1, char2


def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    char1 = matr[e1r][e2c]

    char2 = ''
    char2 = matr[e2r][e1c]

    return char1, char2


def encryptByPlayfairCipher(Matrix, plainList):
    CipherText = []
    for i in range(0, len(plainList)):
        c1 = 0
        c2 = 0
        ele1_x, ele1_y = search(Matrix, plainList[i][0])
        ele2_x, ele2_y = search(Matrix, plainList[i][1])

        if ele1_x == ele2_x:
            c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
            # Get 2 letter cipherText
        elif ele1_y == ele2_y:
            c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x,
ele2_y)
        else:
            c1, c2 = encrypt_RectangleRule(
                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        cipher = c1 + c2
        CipherText.append(cipher)
    return CipherText


text_Plain = 'Hello world'
text_Plain = removeSpaces(toLowerCase(text_Plain))
PlainTextList = Diagraph(FillerLetter(text_Plain))
if len(PlainTextList[-1]) != 2:
    PlainTextList[-1] = PlainTextList[-1]+'z'

key = "Playfair"
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)
CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""
for i in CipherList:
    CipherText += i
print("CipherText:", CipherText)
```

OUTPUT:



```
/BTech/Docs/6th Sem/IS/Code/Exp1/PlayFair-Encrypt.py"
Key text: Playfair
Plain Text: hithisis [redacted]
CipherText: ebqmcncn
```

DECRYPTION:

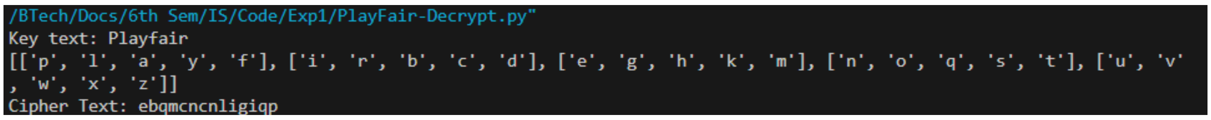
CODE:

```
def toLowerCase(text):
    return text.lower()

def removeSpaces(text):
    newText = ""
    for i in text:
        if i == " ":
            continue
        else:
            newText = newText + i
    return newText

def Diagraph(text):
    Diagraph = []
    group = 0
    for i in range(2, len(text), 2):
        Diagraph.append(text[group:i])
        group = i
    Diagraph.append(text[group:])
    return Diagraph

def FillerLetter(text):
    k = len(text)
    if k % 2 == 0:
        for i in range(0, k, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    else:
        for i in range(0, k-1, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    return new_word

list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):
    key_letters = []
    for i in word:
        if i not in key_letters:
            key_letters.append(i)

    compElements = []
    for i in key_letters:
        if i not in compElements:
            compElements.append(i)
    for i in list1:
        if i not in compElements:
            compElements.append(i)

    matrix = []
    while compElements != []:
        matrix.append(compElements[:5])
        compElements = compElements[5:]

    return matrix

def search(mat, element):
    for i in range(5):
        for j in range(5):
            if(mat[i][j] == element):
                return i, j

def decrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 0:
        char1 = matr[e1r][4]
    else:
        char1 = matr[e1r][e1c-1]

    char2 = ''
    if e2c == 0:
        char2 = matr[e2r][4]
    else:
        char2 = matr[e2r][e2c-1]

    return char1, char2
```

```
def decrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 0:
        char1 = matr[4][e1c]
    else:
        char1 = matr[e1r-1][e1c]

    char2 = ''
    if e2r == 0:
        char2 = matr[4][e2c]
    else:
        char2 = matr[e2r-1][e2c]

    return char1, char2

def decrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    char1 = matr[e1r][e2c]

    char2 = ''
    char2 = matr[e2r][e1c]

    return char1, char2

def decryptByPlayfairCipher(Matrix, cipherList):
    PlainText = []
    for i in range(0, len(cipherList)):
        c1 = 0
        c2 = 0
        ele1_x, ele1_y = search(Matrix, cipherList[i][0])
        ele2_x, ele2_y = search(Matrix, cipherList[i][1])

        if ele1_x == ele2_x:
            c1, c2 = decrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
        elif ele1_y == ele2_y:
            c1, c2 = decrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x,
ele2_y)
        else:
            c1, c2 = decrypt_RectangleRule(Matrix, ele1_x, ele1_y, ele2_x,
ele2_y)

        plaintext = c1 + c2
        PlainText.append(plaintext)
    return PlainText
```

```python
text_Cipher = 'ebqmcncnligiqp'
text_Cipher = removeSpaces(toLowerCase(text_Cipher))

# Pad the ciphertext if its length is odd
if len(text_Cipher) % 2 != 0:
    text_Cipher += 'x'

CipherTextList = Diagraph(text_Cipher)

key = "Playfair"
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)
print(Matrix)

print("Cipher Text:", text_Cipher)
PlainTextList = decryptByPlayfairCipher(Matrix, CipherTextList)

PlainText = ""
for i in PlainTextList:
    PlainText += i
print("PlainText:", PlainText)
```

OUTPUT:



```
/BTech/Docs/6th Sem/IS/Code/Exp1/PlayFair-Decrypt.py"
Key text: Playfair
[['p', 'l', 'a', 'y', 'f'], ['i', 'r', 'b', 'c', 'd'], ['e', 'g', 'h', 'k', 'm'], ['n', 'o', 'q', 's', 't'], ['u', 'v'
, 'w', 'x', 'z']]
Cipher Text: ebqmcncnligiqp
```