Don Bosco Institute of Technology, Mumbai 400070
Department of Information Technology
Experiment No. : 3

Name: Mohammad Zaid Ansari Roll No.: 03 Date: 14/09/22

Title : Playfair Cipher Implementation

Problem Definition : Implement Playfair cipher and illustrate encoding and decoding process on user entered sentence.

Pre-requisite : Any programming knowledge – C, C++, Java, Python and concept of Playfair cipher.

Theory :

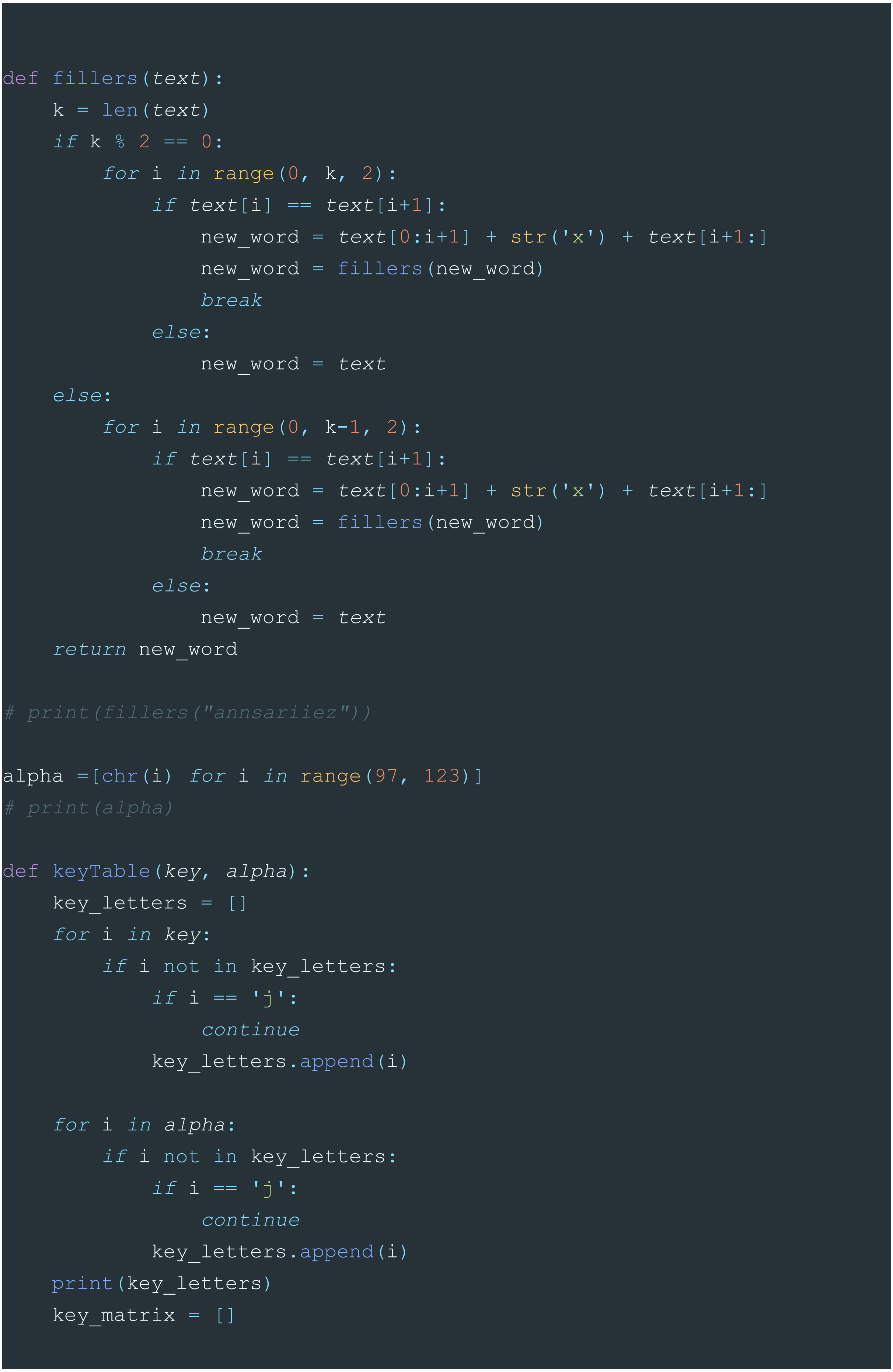
The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet.
It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment**.**

Procedure/ Algorithm :

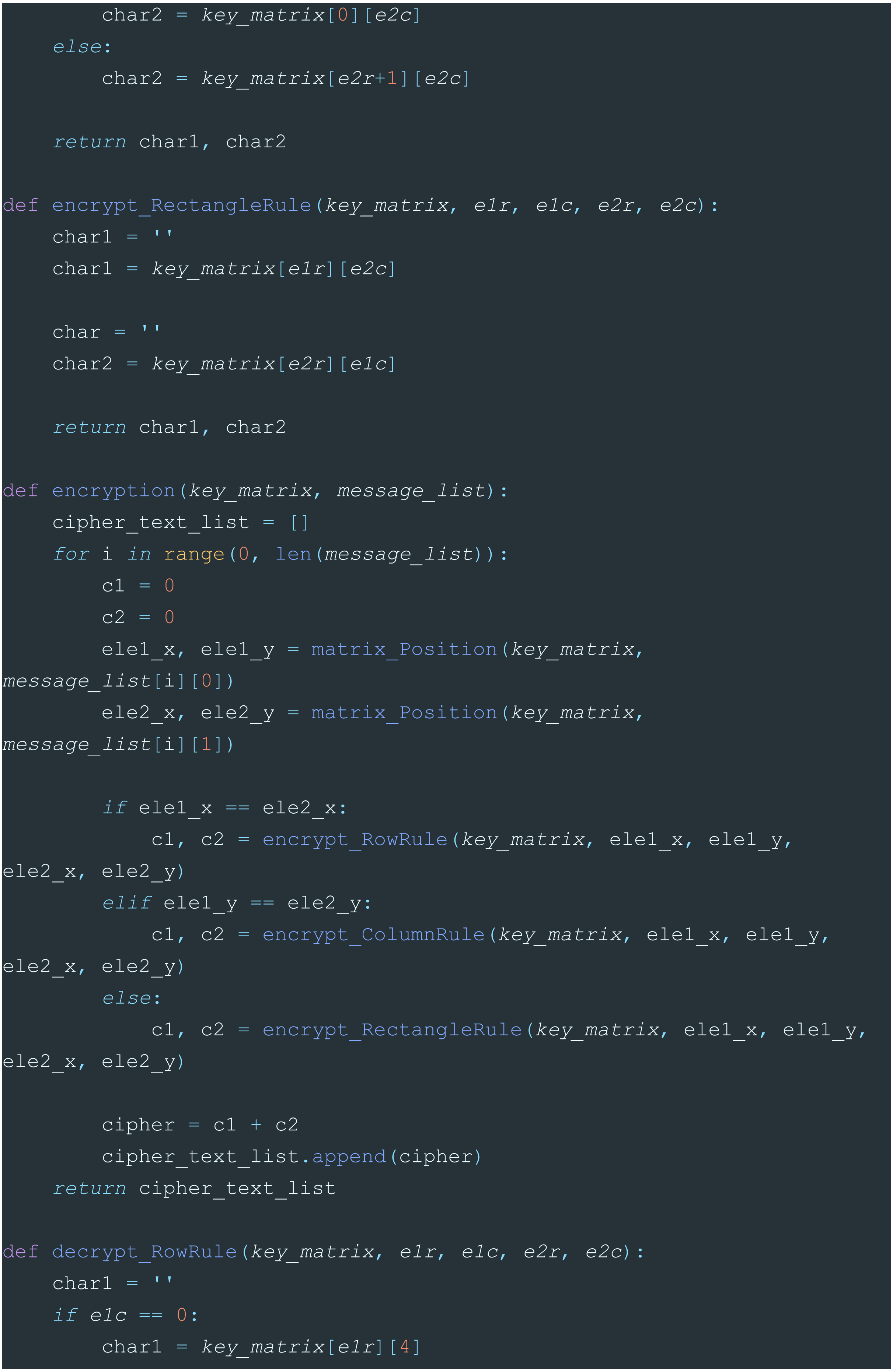
- Take Plain/cipher text and key as input
- Convert them to lower case and remove all spaces
- Take the text and make a group of two alphabets(diagraphs) and make sure same alphabets should not be together
- Apply row column and rectangle rule to the diagraphs

Results :

```
def message(text):
    text=text.lower()
    text=text.replace(" ","")
    return text

def diagraph(text):
    diagraph = []
    j = 0
    for i in range(2,len(text), 2):
        diagraph.append(text[j:i])
        j = i
    diagraph.append(text[j:])
    return diagraph
```



```python
def fillers(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 = fillers(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 = fillers(new_word)
                break
            else:
                new_word = text
    return new_word

# print(fillers("annsariiez"))

alpha =[chr(i) for i in range(97, 123)]
# print(alpha)

def keyTable(key, alpha):
    key_letters = []
    for i in key:
        if i not in key_letters:
            if i == 'j':
                continue
            key_letters.append(i)

    for i in alpha:
        if i not in key_letters:
            if i == 'j':
                continue
            key_letters.append(i)
    print(key_letters)
    key_matrix = []
    for i in range(5):
        key_matrix.append('')

    key_matrix[0] = key_letters[0:5]
    key_matrix[1] = key_letters[5:10]
    key_matrix[2] = key_letters[10:15]
    key_matrix[3] = key_letters[15:20]
    key_matrix[4] = key_letters[20:25]

    return key_matrix

# print(keyTable("monarchy", alpha))

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

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

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

    return char1, char2

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

    char2 = ''
    if e2r == 4:
```



```
        char2 = key_matrix[0][e2c]
    else:
        char2 = key_matrix[e2r+1][e2c]

    return char1, char2

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

    char = ''
    char2 = key_matrix[e2r][e1c]

    return char1, char2

def encryption(key_matrix, message_list):
    cipher_text_list = []
    for i in range(0, len(message_list)):
        c1 = 0
        c2 = 0
        ele1_x, ele1_y = matrix_Position(key_matrix,
message_list[i][0])
        ele2_x, ele2_y = matrix_Position(key_matrix,
message_list[i][1])

        if ele1_x == ele2_x:
            c1, c2 = encrypt_RowRule(key_matrix, ele1_x, ele1_y,
ele2_x, ele2_y)
        elif ele1_y == ele2_y:
            c1, c2 = encrypt_ColumnRule(key_matrix, ele1_x, ele1_y,
ele2_x, ele2_y)
        else:
            c1, c2 = encrypt_RectangleRule(key_matrix, ele1_x, ele1_y,
ele2_x, ele2_y)

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

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

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

    return char1, char2

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

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

    return char1, char2

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

    char = ''
    char2 = key_matrix[e2r][e1c]

    return char1, char2

def decryption(key_matrix, message_list):
    cipher_text_list = []
    for i in range(0, len(message_list)):
        c1 = 0
        c2 = 0
        ele1_x, ele1_y = matrix_Position(key_matrix,
message_list[i][0])
```

```
        ele2_x, ele2_y = matrix_Position(key_matrix,
message_list[i][1])

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

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

def encrypt_process():
    plain_text = input("Please enter your message: ")
    plain_text = message(plain_text)
    plain_text_list = diagraph(fillers(plain_text))
    if len(plain_text_list[-1]) != 2:
        plain_text_list[-1] = plain_text_list[-1] + 'z'

    key = input("Please enter your key: ")
    key = message(key)

    key_matrix = keyTable(key, alpha)

    print("Plain Text:", plain_text)
    CipherList = encryption(key_matrix, plain_text_list)

    CipherText = ""

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

def decrypt_process():
    cipher_text = input("Please enter your message: ")
    cipher_text = message(cipher_text)
    cipher_text_list = diagraph(fillers(cipher_text))
    if len(cipher_text_list[-1]) != 2:
        cipher_text_list[-1] = cipher_text_list[-1] + 'z'

    key = input("Please enter your key: ")
    key = message(key)

    key_matrix = keyTable(key, alpha)

    print("Cipher Text:", cipher_text)
    PlainList = decryption(key_matrix, cipher_text_list)

    PlainText = ""

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

choice = 0
while(choice!=3):
    choice = int(input("Press\n1 to encrypt\n2 to decrypt\n3 to
exit\n"))
    if choice == 1:
        encrypt_process()
    if choice == 2:
        decrypt_process()
```

Output:

```
Press
1 to encrypt
2 to decrypt
3 to exit
1
Please enter your message: Mass bunk tomorrow
Please enter your key: itgroup
['i', 't', 'g', 'r', 'o', 'u', 'p', 'a', 'b', 'c', 'd', 'e', 'f', 'h', 'k', 'l', 'm', 'n', 'q', 's', 'v', 'w', 'x', 'y', 'z']
Plain Text: massbunktomorrow
CipherText: npnzqcaleotsiooixv
Press
1 to encrypt
2 to decrypt
3 to exit
2
Please enter your message: npnzqcaleotsiooixv
Please enter your key: itgroup
['i', 't', 'g', 'r', 'o', 'u', 'p', 'a', 'b', 'c', 'd', 'e', 'f', 'h', 'k', 'l', 'm', 'n', 'q', 's', 'v', 'w', 'x', 'y', 'z']
Cipher Text: npnzqcaleotsiooixv
PlainText: masxsbunktomorrowz
Press
1 to encrypt
2 to decrypt
3 to exit
3
Bye
```

References :

1) https://www.educba.com/types-of-cipher/
2)https://www.geeksforgeeks.org/playfair-cipher-with-examples/
3)https://www.youtube.com/watch?v=hirmJGZXFQg&t=4s

Lab practice (optional) :

L1. Implement Vigenere cipher.

Questions (Short, Long, MCQs) (optional) :

S1: Monoalphabetic substitution v/s Polyalphabetic sustitution.
L1: Encrypt given sentence using Playfair cipher.
L2: Encrypt given sentence using Vigenere cipher.