

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 4

Aim : Alice wants to send some confidential information to Bob over a secure network. Prepare a key matrix for the given key and apply encryption on the plain text (key is your surname & plain text is your name).

Code :

```
import YSL_io

letters = "abcdefghijklmnopqrstuvwxyz"

def to_lower(text):
    return text.lower()

def remove_spaces(text):
    new_text = ""
    for char in text:
        if char != " ":
            new_text += char
    return new_text

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

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 4

```
def fill_letter(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] + "x" + text[i + 1 :]
                new_word = fill_letter(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] + "x" + text[i + 1 :]
                new_word = fill_letter(new_word)
                break
        else:
            new_word = text
    return new_word

def generate_key_table(word, letters):
    key_letters = []
    for char in word:
        if char not in key_letters:
            key_letters.append(char)
    comp_elements = []
    for char in key_letters:
        if char not in comp_elements:
            comp_elements.append(char)
    for char in letters:
        if char not in comp_elements:
            comp_elements.append(char)
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 4

```
matrix = []
while comp_elements:
    matrix.append(comp_elements[:5])
    comp_elements = comp_elements[5:]
return matrix

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

def encrypt_row_rule(matrix, r1, c1, r2, c2):
    char1 = matrix[r1][(c1 + 1) % 5] if c1 != 4 else matrix[r1][0]
    char2 = matrix[r2][(c2 + 1) % 5] if c2 != 4 else matrix[r2][0]
    return char1, char2

def encrypt_column_rule(matrix, r1, c1, r2, c2):
    char1 = matrix[(r1 + 1) % 5][c1] if r1 != 4 else matrix[0][c1]
    char2 = matrix[(r2 + 1) % 5][c2] if r2 != 4 else matrix[0][c2]
    return char1, char2

def encrypt_rectangle_rule(matrix, r1, c1, r2, c2):
    char1 = matrix[r1][c2]
    char2 = matrix[r2][c1]
    return char1, char2

def encrypt_by_playfair_cipher(matrix, plain_list):
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 4

```
cipher_text = []
for pair in plain_list:
    ele1_r, ele1_c = search(matrix, pair[0])
    ele2_r, ele2_c = search(matrix, pair[1])
    if ele1_r == ele2_r:
        char1, char2 = encrypt_row_rule(matrix, ele1_r, ele1_c, ele2_r,
ele2_c)
    elif ele1_c == ele2_c:
        char1, char2 = encrypt_column_rule(matrix, ele1_r, ele1_c,
ele2_r, ele2_c)
    else:
        char1, char2 = encrypt_rectangle_rule(
            matrix, ele1_r, ele1_c, ele2_r, ele2_c
        )
    cipher_text.append(char1 + char2)
return cipher_text

text_plain = YSL_io.inputGRN("\nEnter the plain text : ")
text_plain = remove_spaces(to_lower(text_plain))
plain_text_list = create_digraphs(fill_letter(text_plain))
if len(plain_text_list[-1]) != 2:
    plain_text_list[-1] += "z"

key = YSL_io.inputBLU("\nEnter the key : ")
YSL_io.printBLU("\nKey text : ", key)
YSL_io.printMGNTA('\n\nMatrix : ')
key = to_lower(key)
matrix = generate_key_table(key, letters)

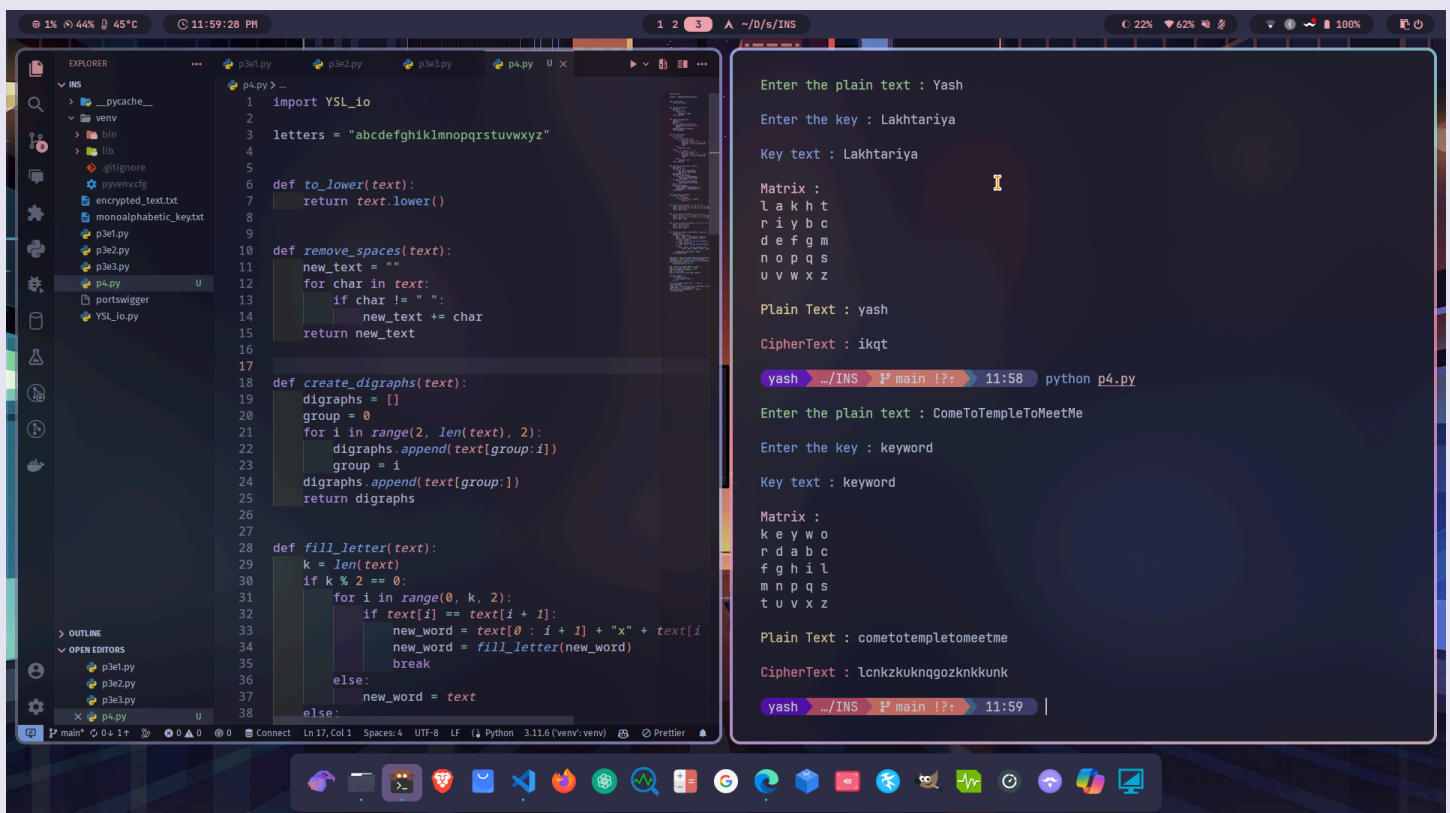
for row in matrix:
    for element in row:
        print(element, end=" ")
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
INS Practical 4

```
print()

YSL_io.printORNG("\nPlain Text : ", end="")
print(text_plain)
cipher_list = encrypt_by_playfair_cipher(matrix, plain_text_list)
cipher_text = "".join(cipher_list)
YSL_io.printRED("\nCipherText : ", end="")
print(cipher_text)
```

Output :



The screenshot shows a code editor with a Python script and its execution output in a terminal. The script implements a Playfair cipher encryption. The terminal output shows the user entering 'Yash' as the plain text and 'Lakhtariya' as the key, resulting in the ciphertext 'ikqt'. A second example shows 'ComeToTempleToMeetMe' as the plain text and 'keyword' as the key, resulting in the ciphertext 'lcnkzkuknqgzkknkkunk'.

```
1 import YSL_io
2
3 letters = "abcdefghijklmnopqrstuvwxyz"
4
5
6 def to_lower(text):
7     return text.lower()
8
9
10 def remove_spaces(text):
11     new_text = ""
12     for char in text:
13         if char != " ":
14             new_text += char
15     return new_text
16
17
18 def create_digraphs(text):
19     digraphs = []
20     group = 0
21     for i in range(2, len(text), 2):
22         digraphs.append(text[group:i])
23         group = i
24     digraphs.append(text[group:])
25     return digraphs
26
27
28 def fill_letter(text):
29     k = len(text)
30     if k % 2 == 0:
31         for i in range(0, k, 2):
32             if text[i] == text[i + 1]:
33                 new_word = text[0 : i + 1] + "x" + text[i + 1:]
34                 new_word = fill_letter(new_word)
35                 break
36             else:
37                 new_word = text
38     else:
```

Enter the plain text : Yash
Enter the key : Lakhtariya
Key text : Lakhtariya
Matrix :
l a k h t
r i y b c
d e f g m
n o p q s
u v w x z
Plain Text : yash
CipherText : ikqt
yash ~/INS P main !? 11:58 python p4.py
Enter the plain text : ComeToTempleToMeetMe
Enter the key : keyword
Key text : keyword
Matrix :
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
Plain Text : cometotempletomeetme
CipherText : lcnkzkuknqgzkknkkunk
yash ~/INS P main !? 11:59