Caesar Cipher

```python
def a2d(text):
    return [ord(i) for i in text]


def encrpyt(text, key):
    dtext = a2d(text)
    result = []
    for i in dtext:
        if (i ≥ 65 and i ≤ 90):
            result.append((((i - 65) + key) % 26) + 65)
        elif(i ≥ 97 and i ≤ 122):
            result.append((((i - 97) + key) % 26) + 97)
        else:
            result.append(i)
    final = list(map(chr, result))
    return ''.join(final)


def decrypt(text, key):
    dtext = a2d(text)
    result = []
    for i in dtext:
        if (i ≥ 65 and i ≤ 90):
            result.append((((i - 65) - key) % 26) + 65)
        elif(i ≥ 97 and i ≤ 122):
            result.append((((i - 97) - key) % 26) + 97)
        else:
            result.append(i)
    final = list(map(chr, result))
    return ''.join(final)


if __name__ == '__main__':
    text = input("Before encryption Plain text: ")
    key = int(input("Key: "))

    cipher = encrpyt(text, key)
    print("Cipher text : {}".format(cipher))

    plain = decrypt(cipher, key)
    print("After decryption Plain text : {}".format(plain))
```

Output:

```
Before encryption Plain text: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Key: 23
Cipher text : XYZABCDEFGHIJKLMNOPQRSTUVW
After decryption Plain text : ABCDEFGHIJKLMNOPQRSTUVWXYZ

***Repl Closed***
```

Playfair Cipher

playfair_cipher.py

```python
import string
from collections import OrderedDict
import numpy as np
from ordered_set import OrderedSet

def key_text_rule(key):
    for j in range(len(key)):
        for i in range(len(key)):
            if ((i%2==0) and (i+1≠len(key))):
                if ((key[i]) == (key[i+1])):
                    near = i+1
                    key = key[:near] + 'x' + key[near:]
                    break

    if (len(key)%2≠0):
        key = key[:len(key)+1] + 'z'
        return key
    else:
        return key

def matrix_fill(key):
    key = "".join(OrderedDict.fromkeys(key))  ## remove the repeated characters in the string
    str1 = string.ascii_lowercase
    for i in key:
        if i in str1:
            str1 = str1.replace(i,'')
    str1 = str1.replace('j','')
    matrix_elements = key + str1

    list1 = []
    ind = 0
    for i in range(5):
        temp = []
        for j in range(5):
            temp.append(matrix_elements[ind])
            ind+=1
        list1.append(temp)

    return list1

################################################################################

## encryption

def same_row_encrypt(ind1,ind2,ind3,ind4,matrix): ## Same 1st index(i.e i)

    ## loop
    if (ind2==4 or ind4==4):
        if (ind2==4):
            ind2 = 0
            print(matrix[ind1][ind2])
            print(matrix[ind3][ind4+1])

        if (ind4==4):
            ind4 = 0
            print(matrix[ind1][ind2+1])
            print(matrix[ind3][ind4])

    ## not a loop
    else:
        print(matrix[ind1][ind2+1])
        print(matrix[ind3][ind4+1])

def same_col_encrypt(ind1,ind2,ind3,ind4,matrix): ## Same 2nd index(i.e j)

    ## loop
    if (ind1==4 or ind3==4):
        if (ind1==4):
            ind1 = 0
            print(matrix[ind1][ind2])
            print(matrix[ind3+1][ind4])

        if (ind3==4):
            ind3 = 0
            print(matrix[ind1+1][ind2])
            print(matrix[ind3][ind4])

    ## not a loop
    else:
        print(matrix[ind1+1][ind2])
        print(matrix[ind3+1][ind4])

def diff(ind1,ind2,ind3,ind4,matrix):  ## Not in same row and same column
    print(matrix[ind1][ind4])
    print(matrix[ind3][ind2])

def encrypt(i_index, j_index, matrix):
    for ind in range(len(i_index)):
        if ((ind%2==0) and ind≠len(i_index)):

            if (i_index[ind]==i_index[ind+1]): ## same i-value
                same_row_encrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)
```

```
            elif (j_index[ind]==j_index[ind+1]): ## same j-value
                same_col_encrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)

            else:
                diff(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)

## decryption

def same_row_decrypt(ind1,ind2,ind3,ind4,matrix): ## Same 1st index(i.e i)

    print(end='')

    ## loop
    if (ind2==0 or ind4==0):
        if (ind2==0):
            ind2 = 4
            print(matrix[ind1][ind2])
            print(matrix[ind3][ind4-1])

        if (ind4==0):
            ind4 = 4
            print(matrix[ind1][ind2-1])
            print(matrix[ind3][ind4])

    ## not a loop
    else:
        print(matrix[ind1][ind2-1])
        print(matrix[ind3][ind4-1])

def same_col_decrypt(ind1,ind2,ind3,ind4,matrix): ## Same 2nd index(i.e j)
    print(end='')

    ## loop
    if (ind1==0 or ind3==0):
        if (ind1==0):
            ind1 = 4
            print(matrix[ind1][ind2])
            print(matrix[ind3-1][ind4])

        if (ind3==0):
            ind3 = 4
            print(matrix[ind1-1][ind2])
            print(matrix[ind3][ind4])

    ## not a loop
    else:
        print(matrix[ind1-1][ind2])
        print(matrix[ind3-1][ind4])

def decrypt(i_index, j_index, matrix):
    for ind in range(len(i_index)):
        if ((ind%2==0) and ind≠len(i_index)):

            if (i_index[ind]==i_index[ind+1]): ## same i-value
                same_row_decrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)

            elif (j_index[ind]==j_index[ind+1]): ## same j-value
                same_col_decrypt(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)

            else:
                diff(i_index[ind],j_index[ind],i_index[ind+1],j_index[ind+1],matrix)

#################################################################################3

def index_fill(plain_text, matrix,mode):

    i_index = []
    j_index = []

    for k in range(len(plain_text)):
        if (k%2==0) and (k+1≠len(plain_text)):
            word_1 = plain_text[k]
            word_2 = plain_text[k+1]

            ## fiding the letters in the matrix
            for i in range(5):
                for j in range(5):
                    if ((word_1==matrix[i][j])):
                        i_index.append(i)
                        j_index.append(j)

            for i in range(5):
                for j in range(5):
                    if ((word_2==matrix[i][j])):
                        i_index.append(i)
                        j_index.append(j)

    if mode=='encryption':
        print("Cipher text")
        encrypt(i_index, j_index,matrix)

    elif mode=='decryption':
        print("Plain text")
        decrypt(i_index, j_index,matrix)
```

```
190
191  ## encryption
192      key = key_text_rule('monarchy')
193      plain_text = key_text_rule('instruments')
194      matrix = matrix_fill(key)
195
196      index_fill(plain_text, matrix,'encryption')
197
198  ## decryption
199      cipher_text = key_text_rule('gatlmzclrqtx')
200      index_fill(cipher_text, matrix,'decryption')
```

Output

playfair_cipher.py | *REPL* [python]

```
Cipher text
g
a
t
l
m
z
c
l
r
q
t
x
Plain text
i
n
s
t
r
u
m
e
n
t
s
z

***Repl Closed***
```

Git-hub link : https://github.com/PrashanthSingaravelan/winter_semester-2022/tree/main/CSI3002%20Applied%20Cryptography%20and%20Network%20Security/lab%20exercises