---------------------------------------------------------------------------

**Name : Atharva Paliwal**

**Roll No : 40 [5B]**

---------------------------------------------------------------------------

**\*\*\* EXPERIMENT NO: 01 \*\*\***

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**AIM- Write a Programme to implement transposition and one time pad cipher**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**CODE-**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**TRANSPOSITION CIPHER**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Atharva Paliwal

def transposition(msg,rows,cols): #Encryption and Decryption Function

k=0 #index of text message

arr = [[0]\*cols for j in range(rows)] #matrix initialisation

for i in range(rows):

for j in range(cols):

if k>=len(msg): #if matrix size exceeds length of message

arr[i][j]=' ' #giving spaces

continue

arr[i][j]=msg[k]

k=k+1

print('Matrix :') #printing matrix

for i in range(rows):

print(\*arr[i])

newmsg=''

for i in range(cols): #adding message column-wise

for j in range(rows):

newmsg=newmsg+arr[j][i]

return newmsg

msg=input('Enter Your Message : ')

rows, cols =map(int,input('Enter size of row and column : ').split())

encrmsg=transposition (msg,rows,cols)

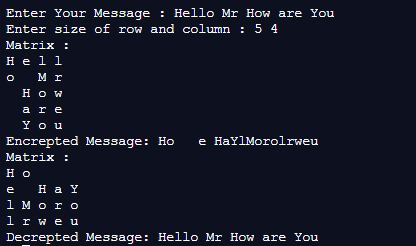
print('Encrepted Message:',encrmsg)

decrprmsg=transposition (encrmsg,cols,rows)

print('Decrepted Message:',decrprmsg)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**OUTPUT-**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ONE-TIME PAD CIPHER**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#function for finding index of required text

def find(bit\_map,k):

for i in range(len(bit\_map)):

if k==bit\_map[i]:

return i

#function for encryption

def encrpt(msg,key):

#Initialising a list of alphabets,numbers and space

bit\_map=list(chr(i)for i in range(65,91))+list(chr(i)for i in range(97,123))+list(str(i)for i in range(0,10))+[' ']

#increasing key length if key length is smaller

key=key\*(len(msg)//len(key)+1)

encr\_msg=''

for i in range(len(msg)):

#modulo addition of message with key

encr\_msg=encr\_msg+

bit\_map[(find(bit\_map,msg[i])+find(bit\_map,key[i]))%63]

return encr\_msg

#function for decryption

def decrpt(msg,key):

bit\_map=list(chr(i)for i in range(65,91))+list(chr(i)for i in range(97,123))+list(str(i)for i in range(0,10))+[' ']

key=key\*(len(msg)//len(key)+1)

decr\_msg=''

for i in range(len(msg)):

#subtraction of encrypted msg and key

k=find(bit\_map,msg[i])-find(bit\_map,key[i])

k=k+[0,63][k<0] #if difference is negative adding 63

decr\_msg=decr\_msg+bit\_map[k]

return decr\_msg

#Driver Code

msg=input('Enter Your Message : ')

key=input('Enter the Key : ')

encr\_msg=encrpt(msg,key)

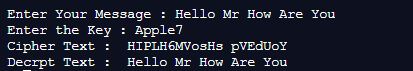
print('Cipher Text : ',encr\_msg)

decr\_msg=decrpt(encr\_msg,key)

print('Decrpt Text : ',decr\_msg)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**OUTPUT-**



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*