Aim - To Create an encryption cipher.(BlahBlah Cipher)

Code -

Encryption -

```
#Importing modules
import random

#This Cipher will work on 25(or less) ASCII characters at a time

#Accepting text and key from user
text=input("Enter the message to be encrypted : ")
key=list(input("Enter a 7-digit key : "))
#In case user has given a key>7 as an input, numbers after the 7th digit will be discarded
key=key[0:7]
dkey=key
print("The text to be encrypted is : {}".format(text));
print("The key is : {}".format(key))

#Required for encryption
#Pre-defined matrix(Stage 1)
predefined=[[1,2,3,5,7],[11,4,6,10,14],[13,8,9,15,21],[17,12,18,13,14],[19,16,14,15,7]]
print("Predefined matrix is : {}".format(predefined))

#For shuffling the row elements(Stage 2), position matrix will be sent to the receiver
position=[0,1,2,3,4]
random.shuffle(position)
print("Positions for shuffling row-elements : {}".format(position))

#Encryption
#Adding dummy characters, if the length of the string is less than 25
dum_ch=input("Enter a character to be treated as dummy character : ")
length=len(text)
if(length<25):
    #Number of dummy characters to be added
    padding=(25-length)
    dummy=(dum_ch * padding)
    text=text+dummy
    length=len(text)
    print("Input : {}".format(text))

#Encryption
#The mid value of the key is the number of times the encryption cycle will be repeated
repeat=0
v=int(key[3])
dv=v
'''print("v ",v)
```

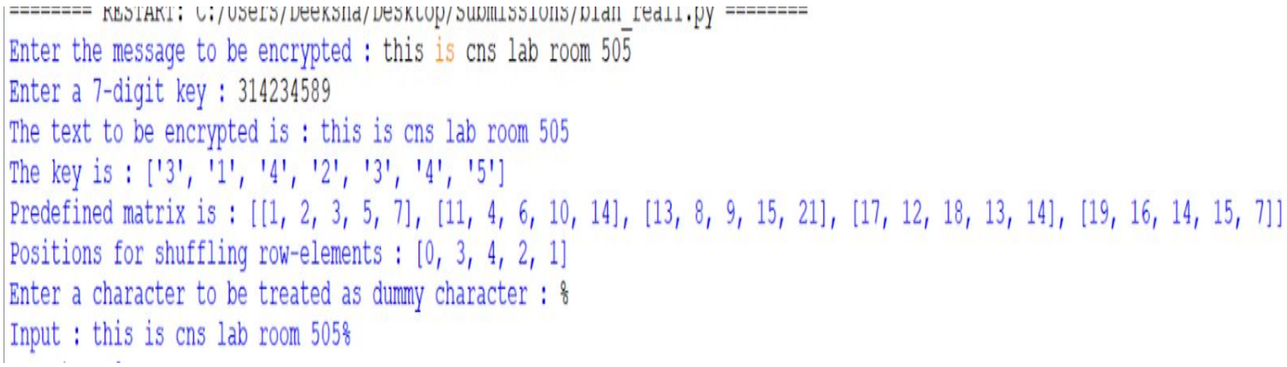
```
print("dv ",dv)'''

#Stage 1 - Adding the ASCII value of the characters to the values in the predefined matrix,
encrypts each element of the matrix,
#independent of the neighbouring values
while repeat!=v:
    print("Repeat : ",repeat)

    if repeat!=0:
        text=encrypt

    #Encryption Matrix - Writing the text in matrix form
    count=0
    cipher=[]
    for i in range(int(length/5)):
        insert=list(text[count*5:(count*5)+5])
        cipher.insert(count,insert)
        count=count+1
    print("Text Matrix {}".format(cipher))

    '''for i in range(5):
        for j in range(5):
            print(ord(cipher[i][j]), end=" ")'''

    for i in range(5):
        for j in range(5):
            numb=ord(cipher[i][j])
            #print(numb)
            cipher[i][j]=numb-predefined[i][j]

    print("Stage 1 Encryption : {}".format(cipher))

    #Stage 2 - Shuffling the row elements according to a position matrix
    new=[]
    row_shuff=[]

    for i in range(5):
        new=[]
        temp=cipher[i]
        for j in range(5):
            x=temp[position[j]]
            new.insert(j,x)
        list(new)
        row_shuff.insert(i,new)
    cipher=row_shuff
    print("Stage 2 Encryption : {}".format(cipher))

    #Stage 3 -
    print("Stage 3 Encryption : ")
    #Phase 1 - Transpose the matrix
    y=[]
    column=[]
    m=0
    for i in range(5):
        y=[]
        for j in range(5):
            m=row_shuff[j][i]
            y.insert(j,m)
        list(y)
        column.insert(i,y)

    cipher=column
    print("Transpose of matrix : {}".format(cipher))

    #Phase 2 - Reverse the Row elements
    subs=[]
    final=[]
    for i in range(5):
        subs=[]
        subs=cipher[i]
        subs.reverse()
        final.insert(i,subs)

    cipher=final
    print("Reverse of rows : {}".format(cipher))

    #Phase 3 - Adding key elements to the matrix elements
    #Reducing the 7 digit key to a 5 digit key by removing the element at the position specified by
the first and last element
    #of the list
    if repeat==0:
        for i in range(7):
            key[i]=int(key[i])

        #If the position specified by the first/last element does not exist, the first/last element will
be removed
        try:
            a=key[0]
            key.pop(a)
        except IndexError:
            dkey.pop(0)

        try:
            b=key[5]
            key.pop(b)
        except IndexError:
            key.pop(5)

        print("5 Digit Key : {}".format(key))

    #Adding key values
    for i in range(5):
        value=key[i]
        for j in range(5):
            cipher[i][j]=cipher[i][j]+value

    print("Adding Key : {}".format(cipher))

    #Encrypted text to be sent
    encrypt=""
    for i in range(5):
        for j in range(5):
            a=chr(cipher[i][j])
            #print(cipher[i][j]," ",a);
            encrypt=encrypt+a
    print("Encrypted Text : ",encrypt)

    #Incrementing repeat at the completion of a cycle
    repeat=repeat+1
```

Decryption -

```
#Decryption
print()
detext=encrypt
print("Text to be decrypted : ",detext)

#Redefining position matrix, for re-shuffling the row elements
dposition=[]
for i in range(5):
    for j in range(5):
        if position[j]==i:
            break;
    dposition.append(j)

print("Positions for re-shuffling row-elements : {}".format(dposition));

#The mid value determines the number of times the cycle will be repeated
drepeat=0
'''dv=int(key[3])'''

#Decryption Matrix - Writing the text in matrix form
dcount=0;
decipher=[]
```

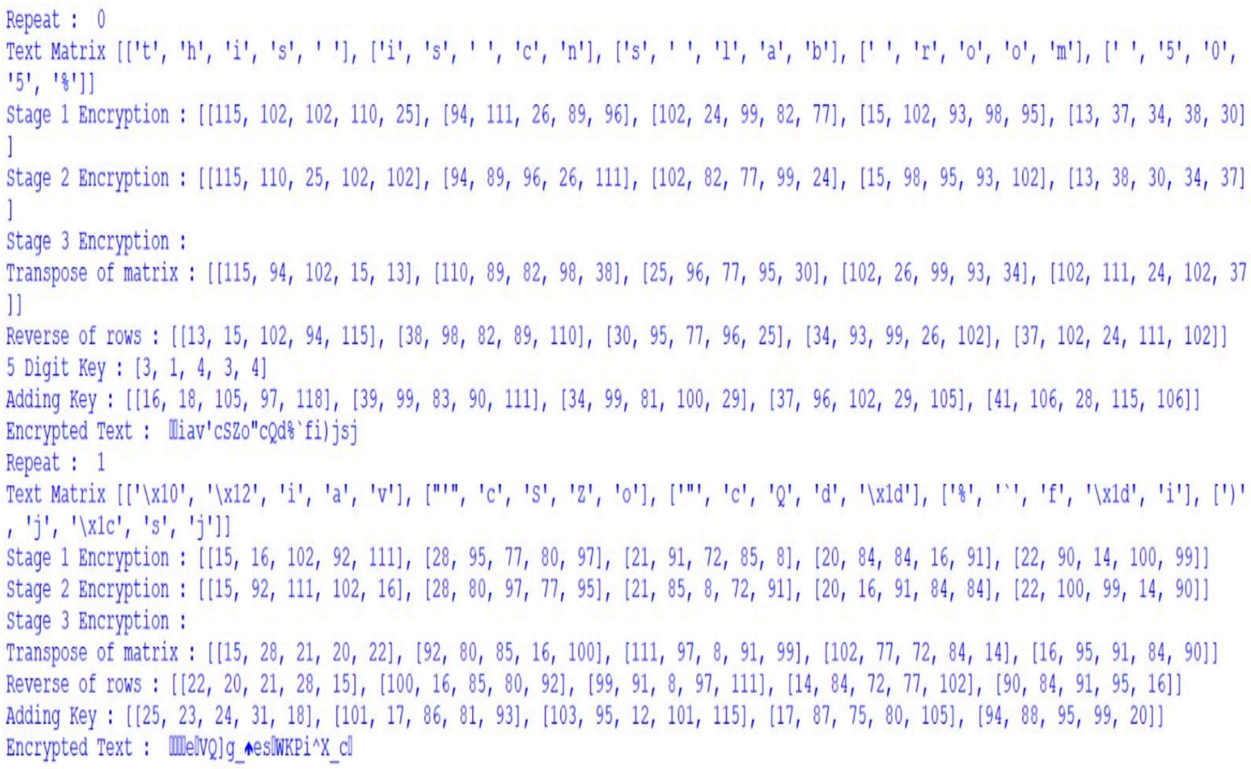
```
for i in range(int(length/5)):
    dinsert=list(detext[dcount*5:(dcount*5)+5])
    decipher.insert(count,dinsert)
    dcount=dcount+1
print("Text Matrix {}".format(decipher))

for i in range(5):
    for j in range(5):
        x=ord(decipher[i][j])
        decipher[i][j]=x

while drepeat!=dv:
    #Stage 1
    print("Stage 1 Decryption : ")

    #Phase 1
    #In case sender and receiver application are different the following code will be used to reduce
the key
    '''#Reducing the 7 digit key to a 5 digit key by removing the element at the position specified
by the first and last element of the list
    print(dkey)
    for i in range(7):
        dkey[i]=int(dkey[i])

    #If the position specified by the first/last element does not exist, the first/last element will be
removed
    try:
        a=dkey[0]
        dkey.pop(a)
    except IndexError:
        dkey.pop(0)

    try:
        b=dkey[5]
        dkey.pop(b)
    except IndexError:
        dkey.pop(5)

    print("5 Digit Key : {}".format(dkey))

    drepeat=1
    dv=int(dkey[3])
    '''

    print("The key is : ",dkey)
    #Subtracting key values
    for i in range(5):
        value=dkey[i]
        for j in range(5):
            decipher[i][j]=decipher[i][j]-value

    print("Subtracting key values : {}".format(decipher))

    #Phase 2 - Reverse the Row elements
    dsubs=[]
    dfinal=[]
    for i in range(5):
        dsubs=[]
        dsubs=decipher[i]
        dsubs.reverse()
        dfinal.insert(i,dsubs)

    cipher=final
    print("Reverse of rows : {}".format(cipher))

    #Phase 3 - Transpose of the matrix
    dy=[]
    dcolumn=[]
    dm=0
    for i in range(5):
        dy=[]
        for j in range(5):
            dm=decipher[j][i]
            dy.insert(j,dm)
        list(dy)
        dcolumn.insert(i,dy)

    decipher=dcolumn
    print("Transpose of matrix : {}".format(decipher))

    #Stage 2 - Re-shuffling the row elements
    dnew=[]
    drow_shuff=[]
    #print(decipher)

    for i in range(5):
        dnew=[]
        dtemp=decipher[i]
        #print(dtemp)
        for j in range(5):
            x=dtemp[dposition[j]]
            #print(x)
            dnew.insert(j,x)
        list(dnew)
        drow_shuff.insert(i,dnew)
        #print(drow_shuff)
    decipher=drow_shuff
    print("Stage 2 Decryption : {}".format(decipher))

    #Stage 3 - Subtracting the ASCII value of the characters to the values in the predefined matrix,
encrypts each element of the matrix,
    #independent of the neighbouring values
    for i in range(5):
        for j in range(5):
            dnumb=decipher[i][j]
            #print(dnumb)
            decipher[i][j]=dnumb+predefined[i][j]

    print("Stage 3 Decryption : {}".format(decipher))
    drepeat=drepeat+1

#Decrypted text
decrypt=""
for i in range(5):
    for j in range(5):
        a=chr(decipher[i][j])
        #print(cipher[i][j]," ",a)
        decrypt=decrypt+a
print("Decrypted Text : ",decrypt)



#Removing the dummy characters added
dtext=decrypt.replace(dum_ch, " ")
dtext.strip();
print("Received Text : {}".format(dtext))
```

Output -



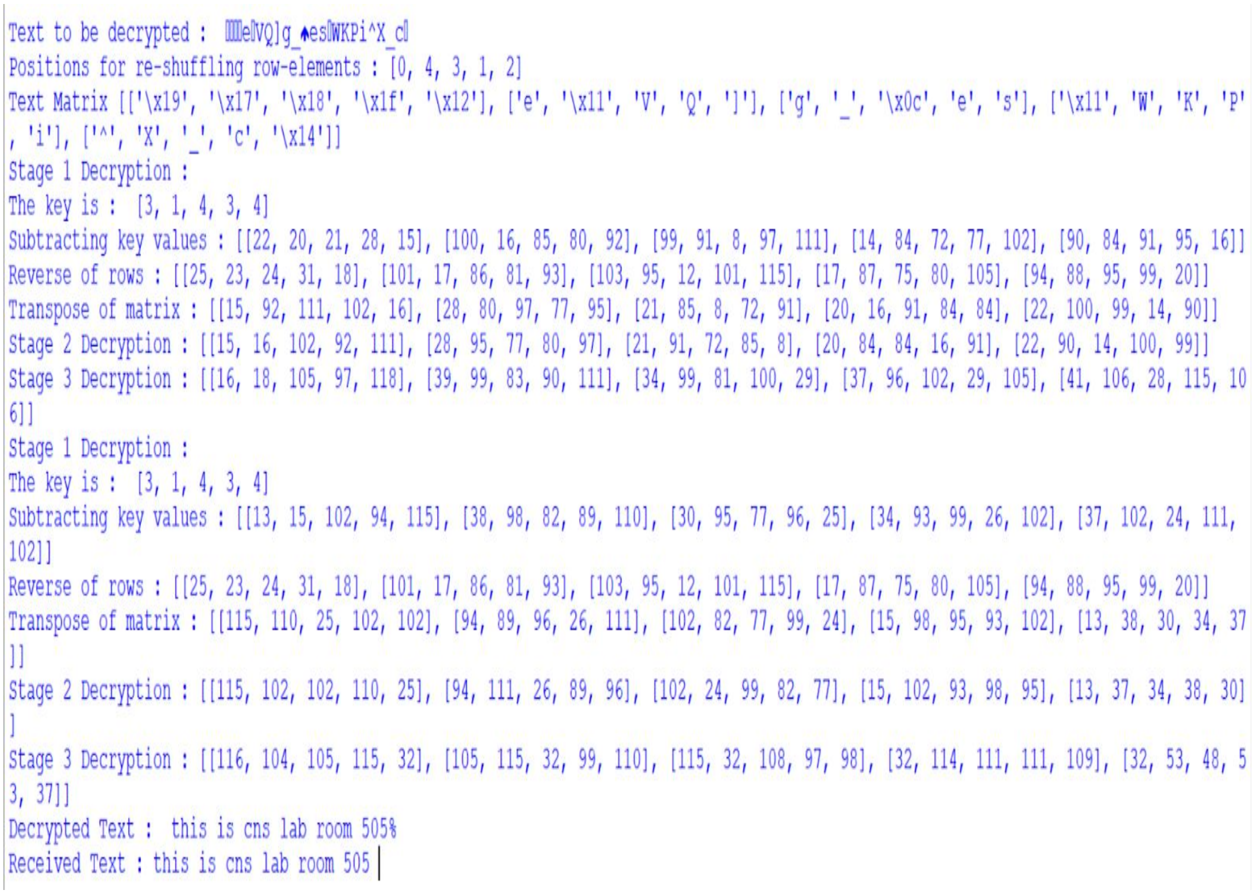
```
======== RESTART: C:/Users/Deeksha/Desktop/Submissions/bian_real1.py ========
Enter the message to be encrypted : this is cns lab room 505
Enter a 7-digit key : 314234589
The text to be encrypted is : this is cns lab room 505
The key is : ['3', '1', '4', '2', '3', '4', '5']
Predefined matrix is : [[1, 2, 3, 5, 7], [11, 4, 6, 10, 14], [13, 8, 9, 15, 21], [17, 12, 18, 13, 14], [19, 16, 14, 15, 7]]
Positions for shuffling row-elements : [0, 3, 4, 2, 1]
Enter a character to be treated as dummy character : %
Input : this is cns lab room 505%
```

Encryption -



```
Repeat :  0
Text Matrix [['t', 'h', 'i', 's', ' '], ['i', 's', ' ', 'c', 'n'], ['s', ' ', 'l', 'a', 'b'], [' ', 'r', 'o', 'o', 'm'], [' ', '5', '0',
'5', '%']]
Stage 1 Encryption : [[115, 102, 102, 110, 25], [94, 111, 26, 89, 96], [102, 24, 99, 82, 77], [15, 102, 93, 98, 95], [13, 37, 34, 38, 30]
]
Stage 2 Encryption : [[115, 110, 25, 102, 102], [94, 89, 96, 26, 111], [102, 82, 77, 99, 24], [15, 98, 95, 93, 102], [13, 38, 30, 34, 37]
]
Stage 3 Encryption :
Transpose of matrix : [[115, 94, 102, 15, 13], [110, 89, 82, 98, 38], [25, 96, 77, 95, 30], [102, 26, 99, 93, 34], [102, 111, 24, 102, 37
]]
Reverse of rows : [[13, 15, 102, 94, 115], [38, 98, 82, 89, 110], [30, 95, 77, 96, 25], [34, 93, 99, 26, 102], [37, 102, 24, 111, 102]]
5 Digit Key : [3, 1, 4, 3, 4]
Adding Key : [[16, 18, 105, 97, 118], [39, 99, 83, 90, 111], [34, 99, 81, 100, 29], [37, 96, 102, 29, 105], [41, 106, 28, 115, 106]]
Encrypted Text :  ▯iav'cSZo"cQd%`fi)jsj
Repeat :  1
Text Matrix [['\x10', '\x12', 'i', 'a', 'v'], ["'", 'c', 'S', 'Z', 'o'], ['"', 'c', 'Q', 'd', '\x1d'], ['%', '`', 'f', '\x1d', 'i'], [')'
, 'j', '\x1c', 's', 'j']]
Stage 1 Encryption : [[15, 16, 102, 92, 111], [28, 95, 77, 80, 97], [21, 91, 72, 85, 8], [20, 84, 84, 16, 91], [22, 90, 14, 100, 99]]
Stage 2 Encryption : [[15, 92, 111, 102, 16], [28, 80, 97, 77, 95], [21, 85, 8, 72, 91], [20, 16, 91, 84, 84], [22, 100, 99, 14, 90]]
Stage 3 Encryption :
Transpose of matrix : [[15, 28, 21, 20, 22], [92, 80, 85, 16, 100], [111, 97, 8, 91, 99], [102, 77, 72, 84, 14], [16, 95, 91, 84, 90]]
Reverse of rows : [[22, 20, 21, 28, 15], [100, 16, 85, 80, 92], [99, 91, 8, 97, 111], [14, 84, 72, 77, 102], [90, 84, 91, 95, 16]]
Adding Key : [[25, 23, 24, 31, 18], [101, 17, 86, 81, 93], [103, 95, 12, 101, 115], [17, 87, 75, 80, 105], [94, 88, 95, 99, 20]]
Encrypted Text :  ▯▯▯e▯VQ]g_♠es▯WKPi^X_c▯
```

Decryption -



```
Text to be decrypted :  ▯▯▯e▯VQ]g_♠es▯WKPi^X_c▯
Positions for re-shuffling row-elements : [0, 4, 3, 1, 2]
Text Matrix [['\x19', '\x17', '\x18', '\x1f', '\x12'], ['e', '\x11', 'V', 'Q', ']'], ['g', '_', '\x0c', 'e', 's'], ['\x11', 'W', 'K', 'P'
, 'i'], ['^', 'X', '_', 'c', '\x14']]
Stage 1 Decryption :
The key is :  [3, 1, 4, 3, 4]
Subtracting key values : [[22, 20, 21, 28, 15], [100, 16, 85, 80, 92], [99, 91, 8, 97, 111], [14, 84, 72, 77, 102], [90, 84, 91, 95, 16]]
Reverse of rows : [[25, 23, 24, 31, 18], [101, 17, 86, 81, 93], [103, 95, 12, 101, 115], [17, 87, 75, 80, 105], [94, 88, 95, 99, 20]]
Transpose of matrix : [[15, 92, 111, 102, 16], [28, 80, 97, 77, 95], [21, 85, 8, 72, 91], [20, 16, 91, 84, 84], [22, 100, 99, 14, 90]]
Stage 2 Decryption : [[15, 16, 102, 92, 111], [28, 95, 77, 80, 97], [21, 91, 72, 85, 8], [20, 84, 84, 16, 91], [22, 90, 14, 100, 99]]
Stage 3 Decryption : [[16, 18, 105, 97, 118], [39, 99, 83, 90, 111], [34, 99, 81, 100, 29], [37, 96, 102, 29, 105], [41, 106, 28, 115, 10
6]]
Stage 1 Decryption :
The key is :  [3, 1, 4, 3, 4]
Subtracting key values : [[13, 15, 102, 94, 115], [38, 98, 82, 89, 110], [30, 95, 77, 96, 25], [34, 93, 99, 26, 102], [37, 102, 24, 111,
102]]
Reverse of rows : [[25, 23, 24, 31, 18], [101, 17, 86, 81, 93], [103, 95, 12, 101, 115], [17, 87, 75, 80, 105], [94, 88, 95, 99, 20]]
Transpose of matrix : [[115, 110, 25, 102, 102], [94, 89, 96, 26, 111], [102, 82, 77, 99, 24], [15, 98, 95, 93, 102], [13, 38, 30, 34, 37
]]
Stage 2 Decryption : [[115, 102, 102, 110, 25], [94, 111, 26, 89, 96], [102, 24, 99, 82, 77], [15, 102, 93, 98, 95], [13, 37, 34, 38, 30]
]
Stage 3 Decryption : [[116, 104, 105, 115, 32], [105, 115, 32, 99, 110], [115, 32, 108, 97, 98], [32, 114, 111, 111, 109], [32, 53, 48, 5
3, 37]]
Decrypted Text :  this is cns lab room 505%
Received Text : this is cns lab room 505
```