➢ Hill Cipher Encryption

Note: **3-dimension is not working properly because there are so many ways to find adjacent and I didn't able to figure out the correct way**

Though I tried but that doesn't work

Code:

```
import math
import numpy


def convert_cipher(text1, keyarray):
  text='abcdefghijklmnopqrstuvwxyz'
  ll1=[]
  for i in text1:
    ll1.append(text.index(i))
  num=numpy.array(ll1)
  result=numpy.dot(keyarray, num)
  return result%26


print('\nStep: 1')
plain=input('Enter your plain text: ')
key=input('Enter your key: ')
text='abcdefghijklmnopqrstuvwxyz'
cipher=''
leng=math.floor(math.sqrt(len(key)))
if (math.sqrt(len(key))-leng) == 0:
  l1=[]
  l3=[]
  for i in key:
    l1.append(i)
  for i in l1:
    l3.append(text.index(i))
```

```
        keyarr=numpy.array(l1)
        nkeyarr=numpy.array(l3)
        newarr = keyarr.reshape(leng, leng)
        nnewarr = nkeyarr.reshape(leng, leng)
    else:
        print('Your key should be perfect square of some number')
    print('\nStep: 2\n')
    print(newarr)
    if not len(plain)%leng == 0:
        a1=leng-(len(plain)%leng)
        for i in range(a1):
            plain+='x'
    print('\nStep: 3\n')
    l2 = [(plain[i:i+leng]) for i in range(0, len(plain), leng)]
    print(l2)

    for i in l2:
        cipherno=convert_cipher(i, nnewarr)
        for j in cipherno:
            cipher+=text[j]
    print('\nStep: 4')
    print(f'\nCipher Text: {cipher}')
```

OUTPUT:

```
In [21]: runfile('C:/Users/Admin/study material/sem5/Practicals/Cryptography/Practical-6/
Hill_cipher_encryption.py', wdir='C:/Users/Admin/study material/sem5/Practicals/Cryptography/
Practical-6')

Step: 1

Enter your plain text: exam

Enter your key: hill

Step: 2

[['h' 'i']
 ['l' 'l']]

Step: 3

['ex', 'am']

Step: 4

Cipher Text: elsc

In [22]:
```

➢ Hill Cipher Decryption

Code:

```
def encryption():
import math
import numpy


def convert_cipher(text1, keyarray):
  text='abcdefghijklmnopqrstuvwxyz'
  ll1=[]
  for i in text1:
     ll1.append(text.index(i))
  num=numpy.array(ll1)
  result=numpy.dot(keyarray, num)
  return result%26


def get_inverse(mat):
   np=numpy.linalg.det(mat)
   np=int(np)
```

```
    if np<0:
        np=np%26
    for i in range(0, 26):
        if (np*i)%26==1:
            return i
    return 1


def two_dim(mat):
    ll1=[]
    ll1.append(mat[1][1])
    ll1.append(-mat[0][1])
    ll1.append(-mat[1][0])
    ll1.append(mat[0][0])
    ll2=numpy.array(ll1)
    newarr = ll2.reshape(2, 2)
    return newarr%26

def three_dim(mat):
    ll1=[]
    ll1.append(mat[0][0]*( (mat[1][1]*mat[2][2]) - (mat[1][2]*mat[2][1]) ))
    ll1.append(-mat[0][1]*( (mat[1][0]*mat[2][2]) - (mat[1][2]*mat[2][0]) ))
    ll1.append(mat[0][2]*( (mat[1][0]*mat[2][1]) - (mat[1][1]*mat[2][0]) ))
    ll1.append(-mat[1][0]*( (mat[0][1]*mat[2][2]) - (mat[0][2]*mat[2][1]) ))
    ll1.append(mat[1][1]*( (mat[0][0]*mat[2][2]) - (mat[0][2]*mat[2][0]) ))
    ll1.append(-mat[1][2]*( (mat[0][0]*mat[2][1]) - (mat[0][1]*mat[2][0]) ))
    ll1.append(mat[2][0]*( (mat[0][1]*mat[1][2]) - (mat[0][2]*mat[1][1]) ))
    ll1.append(-mat[2][1]*( (mat[0][0]*mat[1][2]) - (mat[0][2]*mat[2][0]) ))
    ll1.append(mat[2][2]*( (mat[0][0]*mat[1][1]) - (mat[0][1]*mat[1][0]) ))
    ll2=numpy.array(ll1)
```

```
    newarr = ll2.reshape(3, 3)
    return newarr%26


print('\nStep: 1')
plain=input('Enter your cipher text: ')
key=input('Enter your key: ')
text='abcdefghijklmnopqrstuvwxyz'
cipher=''
leng=math.floor(math.sqrt(len(key)))
if (math.sqrt(len(key))-leng) == 0:
    l1=[]
    l3=[]
    for i in key:
        l1.append(i)
    for i in l1:
        l3.append(text.index(i))
    keyarr=numpy.array(l1)
    nkeyarr=numpy.array(l3)
    newarr = keyarr.reshape(leng, leng)
    nnewarr = nkeyarr.reshape(leng, leng)

else:
    print('Your key should be perfect square of some number')
print('\nStep: 2\n')
print(newarr)
if not len(plain)%leng == 0:
    a1=leng-(len(plain)%leng)
    for i in range(a1):
        plain+='x'
```

```
print('\nmatrix: \n')
print(nnewarr)
print('\nStep: 3\n')
l2 = [(plain[i:i+leng]) for i in range(0, len(plain), leng)]
print(l2)

if len(nnewarr)==2:
    nnewarr1=two_dim(nnewarr)
    inverse=get_inverse(nnewarr)
    nnewarr2=inverse*nnewarr1

elif len(nnewarr)==3:
    nnewarr1=three_dim(nnewarr)
    inverse=get_inverse(nnewarr)
    nnewarr2=inverse*nnewarr1

else:
    print('Key out of bound')
for i in l2:
    cipherno=convert_cipher(i, nnewarr2)
    for j in cipherno:
        cipher+=text[j]
print('\nStep: 4')
print(f'\nPlain Text: {cipher}')
```

OUTPUT:

Tejas Tripathi

```
In [23]: runfile('C:/Users/Admin/study material/sem5/Practicals/Cryptography/Practical-6/
Hill_cipher_decryption.py', wdir='C:/Users/Admin/study material/sem5/Practicals/Cryptography/
Practical-6')

Step: 1

Enter your cipher text: elsc

Enter your key: hill

Step: 2

[['h' 'i']
 ['l' 'l']]

matrix:

[[ 7  8]
 [11 11]]

Step: 3

['el', 'sc']

Step: 4

Plain Text: exam

In [24]:
```