```
P="hello everyone"
lst1= []
plaintext= []
for i in range(97,123):
    lst1.append(chr(i))
print(lst1)
k=[]
for j in lst1:
    k.append(lst1.index(j))
print(k)
for i in P:
    if i in lst1:
        print(lst1.index(i))
        plaintext.append(lst1.index(i))
print(plaintext)
cipher=[x+1 for x in plaintext]
print(cipher)
for m in cipher:
    if m in k:
        print(lst1[m],end="")
```

IDLE Shell 3.11.0 - C:/Users/dhanu/cnsxp1.py (3.11.0)

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/cnsexp1.py ==========
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
'y', 'z']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
7
4
11
11
14
4
21
4
17
24
14
13
4
[7, 4, 11, 11, 14, 4, 21, 4, 17, 24, 14, 13, 4]
[8, 5, 12, 12, 15, 5, 22, 5, 18, 25, 15, 14, 5]
ifmmpfwfszpof
>>>
```

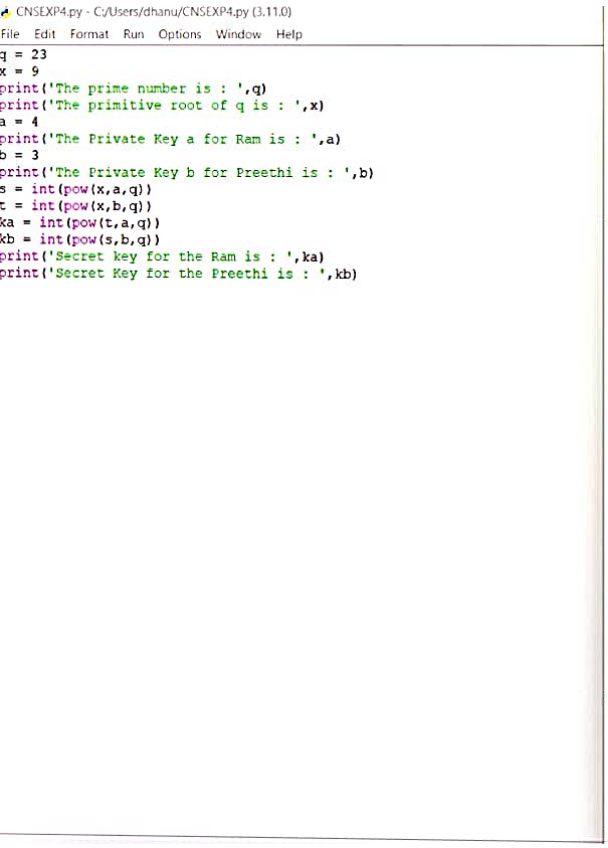
File Edit Format Run Options Window Help

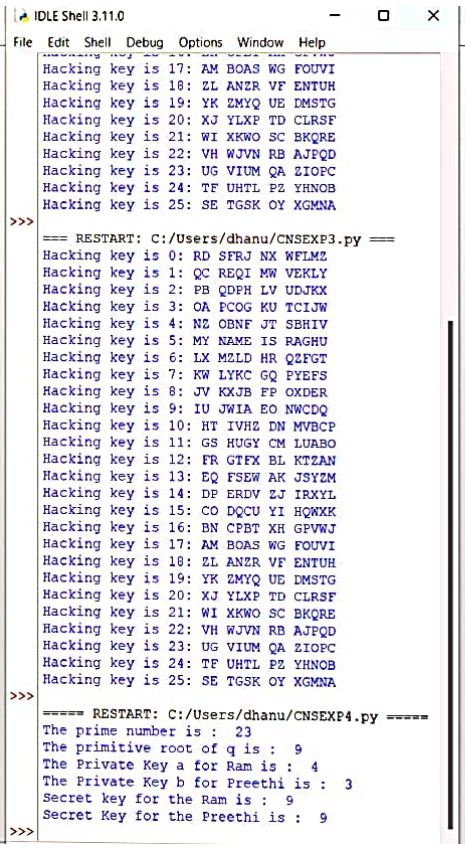
```
def decrypt():
    encrypted_message = input("Enter the message i.e to be decrypted: ").strip()
    letters="abcdefghijklmnopqrstuvwxyz"
    k = int(input("Enter the key to decrypt: "))
    decrypted_message = ""
    for ch in encrypted_message:
        if ch in letters:
            position = letters.find(ch)
            new_pos = (position - k) % 26
            new_char = letters[new_pos]
            decrypted_message += new_char
        else:
            decrypted_message += ch
    print("Your decrypted message is:\n")
    print(decrypted_message)
decrypt()
```

IDLE Shell 3.11.0 - C:/Users/dhanu/cnsxp1.py (3.11.0)

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/cnsexp1.py ==========
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
'y', 'z']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]
7
4
11
11
14
4
21
4
17
24
14
13
4
[7, 4, 11, 11, 14, 4, 21, 4, 17, 24, 14, 13, 4]
[8, 5, 12, 12, 15, 5, 22, 5, 18, 25, 15, 14, 5]
ifmmpfwfszpof
>>>
========== RESTART: C:/Users/dhanu/CNSexp.py ==========
Enter the message i.e to be decrypted: phww
Enter the key to decrypt: 3
Your decrypted message is:

mett
>>>
```


CNSEXP4.py - C:/Users/dhanu/CNSEXP4.py (3.11.0)
File Edit Format Run Options Window Help
q = 23
x = 9
print('The prime number is : ',q)
print('The primitive root of q is : ',x)
a = 4
print('The Private Key a for Ram is : ',a)
b = 3
print('The Private Key b for Preethi is : ',b)
s = int(pow(x,a,q))
t = int(pow(x,b,q))
ka = int(pow(t,a,q))
kb = int(pow(s,b,q))
print('Secret key for the Ram is : ',ka)
print('Secret Key for the Preethi is : ',kb)


IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Hacking key is 17: AM BOAS WG FOUVI
Hacking key is 18: ZL ANZR VF ENTUH
Hacking key is 19: YK ZMYQ UE DMSTG
Hacking key is 20: XJ YLXP TD CLRSF
Hacking key is 21: WI XKWO SC BKQRE
Hacking key is 22: VH WJVN RB AJPQD
Hacking key is 23: UG VIUM QA ZIOPC
Hacking key is 24: TF UHTL PZ YHNOB
Hacking key is 25: SE TGSK OY XGMNA
>>>
=== RESTART: C:/Users/dhanu/CNSEXP3.py ===
Hacking key is 0: RD SFRJ NX WFLMZ
Hacking key is 1: QC REQI MW VEKLY
Hacking key is 2: PB QDPH LV UDJKX
Hacking key is 3: OA PCOG KU TCIJW
Hacking key is 4: NZ OBNF JT SBHIV
Hacking key is 5: MY NAME IS RAGHU
Hacking key is 6: LX MZLD HR QZFGT
Hacking key is 7: KW LYKC GQ PYEFS
Hacking key is 8: JV KXJB FP OXDER
Hacking key is 9: IU JWIA EO NWCDQ
Hacking key is 10: HT IVHZ DN MVBCP
Hacking key is 11: GS HUGY CM LUABO
Hacking key is 12: FR GTFX BL KTZAN
Hacking key is 13: EQ FSEW AK JSYZM
Hacking key is 14: DP ERDV ZJ IRXYL
Hacking key is 15: CO DQCU YI HQWXK
Hacking key is 16: BN CPBT XH GPVWJ
Hacking key is 17: AM BOAS WG FOUVI
Hacking key is 18: ZL ANZR VF ENTUH
Hacking key is 19: YK ZMYQ UE DMSTG
Hacking key is 20: XJ YLXP TD CLRSF
Hacking key is 21: WI XKWO SC BKQRE
Hacking key is 22: VH WJVN RB AJPQD
Hacking key is 23: UG VIUM QA ZIOPC
Hacking key is 24: TF UHTL PZ YHNOB
Hacking key is 25: SE TGSK OY XGMNA
>>>
===== RESTART: C:/Users/dhanu/CNSEXP4.py =====
The prime number is : 23
The primitive root of q is : 9
The Private Key a for Ram is : 4
The Private Key b for Preethi is : 3
Secret key for the Ram is : 9
Secret Key for the Preethi is : 9
>>>

CNSEXP5.py - C:/Users/dhanu/CNSEXP5.py (3.11.0)

File Edit Format Run Options Window Help

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
def is_coprime(a, b):
    return gcd(a, b) == 1
def is_valid_affine(a, b):
    return is_coprime(a, 26) and b >= 0 and b < 26
def encrypt_affine(msg, a, b):
    ciphertext = ''
    for c in msg:
        if c.isalpha():
            idx = ord(c.upper()) - ord('A')
            idx = (a * idx + b) % 26
            ciphertext += chr(idx + ord('A'))
        else:
            ciphertext += c
    return ciphertext
msg =input("Enter the plain text: ")
 = 5
 = 7
f is_valid_affine(a, b):
    ciphertext = encrypt_affine(msg, a, b)
    print("plaintext:", msg)
    print("Ciphertext:", ciphertext)
lse:
    print("Invalid values of a and/or b.")
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.
1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSE
XP5.py ==========
Enter the plain text: MEET
plaintext: MEET
Ciphertext: PBBY
>>>
```

File Edit Format Run Options Window Help

```
matrix = [['M', 'F', 'H', 'I', 'J', 'K'],
          ['U', 'N', 'O', 'P', 'Q', ' '],
          ['Z', 'V', 'W', 'X', 'Y', ' '],
          ['E', 'L', 'A', 'R', 'G', ' '],
          ['D', 'S', 'T', 'B', 'C', ' ']]
def playfair_encode(message):
    message = message.upper()
    message = message.replace('J', 'I')
    message = message.replace(' ', '')
    if len(message) % 2 != 0:
        message += 'X'
    ciphertext = ''
    for i in range(0, len(message), 2):
        a = message[i]
        b = message[i+1]
        a_row, a_col = 0, 0
        b_row, b_col = 0 , 0
        for row in range(len(matrix)):
            if a in matrix[row]:
                a_row = row
                a_col = matrix[row].index(a)
            if b in matrix[row]:
                b_row = row
                b_col = matrix[row].index(b)
        if a_row == b_row:
            ciphertext += matrix[a_row][(a_col+1)%6]
            ciphertext += matrix[b_row][(b_col+1)%6]
        elif a_col == b_col:
            ciphertext += matrix[(a_row+1)%5][a_col]
            ciphertext += matrix[(b_row+1)%5][b_col]
        else:
            ciphertext += matrix[a_row][b_col]
            ciphertext += matrix[b_row][a_col]
    return ciphertext
message = 'Must see you over Cadogan West. Coming at once.'
ciphertext = playfair_encode(message)
print(ciphertext)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.
1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSE
XP5.py ==========
Enter the plain text: MEET
plaintext: MEET
Ciphertext: PBBY
>>>
======== RESTART: C:/Users/dhanu/CNSEXP6.py =======
UZTBDLGZPNNWLGTGTU ROVLDDHTQFJQLTHPODGIZ
>>>
```

CNSEXP7.py - C:/Users/dhanu/CNSEXP7.py (3.11.0)

File Edit Format Run Options Window Help

```
pt=str(input("ENTER THE PLAIN TEXT : "))
cipher=""
letter="abcdefghijklmnopqrstuvwxyz"
common=max(set(pt),key=pt.count)
print("COMMON LETTER : "+common)
if common in letter:
    common=letter.find(common)
key=common-6
#print("key = "+common+" - g = "+key)
if (key<0):
    key=26-key
for i in pt:
    if i in letter:
        pos=letter.find(i)
        new_pos=(pos+key)%26
        new_char=letter[new_pos]
        cipher+=new_char
print("CIPHER TEXT : "+cipher)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSEXP7.py ==========
ENTER THE PLAIN TEXT : MEET ME AFTER
COMMON LETTER : E
```

Ln: 11 Col: 0

File Edit Format Run Options Window Help

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
key = str(input("enter the key: "))
plaintext = str(input("enter the string: "))
ciphertext = ""
for i in range(len(plaintext)):
    index = alphabet.index(plaintext[i])
    key_index = i % len(key)
    key_char = key[key_index]
    key_alphabet_index = alphabet.index(key_char)
    cipher_index = (index + key_alphabet_index) % 26
    ciphertext += alphabet[cipher_index]
print("cipher text is: ",ciphertext)
```

IDLE Shell 3.11.0

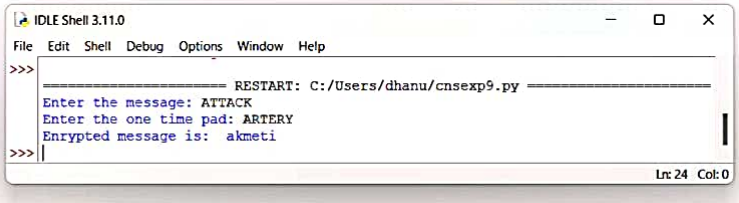
File Edit Shell Debug Options Window Help

```
enter the string: DISCOVERED
Traceback (most recent call last):
  File "C:/Users/dhanu/CNSEXP8.py", line 6, in <module>
    index = alphabet.index(plaintext[i])
ValueError: substring not found
>>>
```

Ln: 19 Col: 0

File Edit Format Run Options Window Help

```
def vernam(plain_text,key):
    plain_text=plain_text.replace(" ","")
    key=key.replace(" ","")
    plain_text=plain_text.lower()
    key=key.lower()
    if(len(plain_text)!=len(key)):
        print("Lengths are different")
    else:
        cipher_text=""
        for i in range(len(plain_text)):
            k1=ord(plain_text[i])-97
            k2=ord(key[i])-97
            s=chr((k1+k2)%26+97)
            cipher_text+=s
        print("Enrypted message is: ",cipher_text)
plain_text=input("Enter the message: ")
key=input("Enter the one time pad: ")
vernam(plain_text,key)
```



cnsexp9.py - C:/Users/dhanu/cnsexp9.py (3.11.0)

File Edit Format Run Options Window Help

```
def vernam(plain_text,key):
    plain_text=plain_text.replace(" ","")
    key=key.replace(" ","")
    plain_text=plain_text.lower()
    key=key.lower()
    if(len(plain_text)!=len(key)):
        print("Lengths are different")
    else:
        cipher_text=""
        for i in range(len(plain_text)):
            k1=ord(plain_text[i])-97
            k2=ord(key[i])-97
            s=chr((k1+k2)%26+97)
            cipher_text+=s
        print("Enrypted message is: ",cipher_text)
plain_text=input("Enter the message: ")
key=input("Enter the one time pad: ")
vernam(plain_text,key)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

Minimize

```
>>>
========== RESTART: C:/Users/dhanu/cnsexp9.py ==========
Enter the message: AKMETI
Enter the one time pad: ARTERY
Enrypted message is:  abfikg
>>>
```

Ln: 29 Col: 0

CNSEXP9.1.py - C:/Users/dhanu/CNSEXP9.1.py (3.11.0)

File Edit Format Run Options Window Help

```
def vernam(cipher_text,key):
    cipher_text=cipher_text.lower()
    key=key.lower()
    cipher_text=cipher_text.replace(" ","")
    key=key.replace(" ","")
    plain_text=""
    for i in range(len(cipher_text)):
        k1=ord(cipher_text[i])-97
        k2=ord(key[i])-97
        s=chr((((k1-k2)+26)%26)+97)
        plain_text+=s
    print("Decrypted message is: ",plain_text)
plain_text=input("Enter the message to be decrypted: ")
key=input("Enter the one time pad: ")
vernam(plain_text,key)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>

========== RESTART: C:/Users/dhanu/CNSEXP9.1.py ==========
======
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
```

Ln: 8 Col: 0

```
import string
main=string.ascii_lowercase
def conversion(plain_text,key):
    index=0
    cipher_text=""
    plain_text=plain_text.lower()
    key=key.lower()
    for c in plain_text:
        if c in main:
            off=ord(key[index])-ord('a')
            encrypt_num=(ord(c)-ord('a')+off)%26
            encrypt=chr(encrypt_num+ord('a'))
            cipher_text+=encrypt
            index=(index+1)%len(key)
        else:
            cipher_text+=c
    print("plain text: ",plain_text)
    print("cipher text: ",cipher_text)
plain_text=input("Enter the message: ")
key=input("Enter the key: ")
conversion(plain_text,key)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSEXP9.1.py ==========
======
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
========== RESTART: C:/Users/dhanu/CNSEXP11.py ==========
Enter the message: hello everyone
Enter the key: 4
plain text:  hello everyone
cipher text:  olssv lclyfvul
>>>
```

```
import string
main=string.ascii_lowercase
def conversion(cipher_text,key):
    index=0
    plain_text=""
    cipher_text=cipher_text.lower()
    key=key.lower()
    for c in cipher_text:
        if c in main:
            off=ord(key[index])-ord('a')
            positive_off=26-off
            decrypt=chr((ord(c)-ord('a')+positive_off)%26
            plain_text+=decrypt
            index=(index+1)%len(key)
        else:
            plain_text+=c
    print("cipher text: ",cipher_text)
    print("plain text (message): ",plain_text)
cipher_text=input("Enter the message to be decrypted: ")
key=input("Enter the key for decryption: ")
conversion(cipher_text,key)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSEXP9.1.py ==========
======
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
========== RESTART: C:/Users/dhanu/CNSEXP11.py ==========
Enter the message: hello everyone
Enter the key: 4
plain text:  hello everyone
cipher text:  olssv lclyfvul
>>>
========== RESTART: C:/Users/dhanu/CNSEXP12.py ==========
Enter the message to be decrypted: olssv lclyfvul
Enter the key for decryption: 4
cipher text:  olssv lclyfvul
plain text (message):  hello everyone
>>>
```

CNSEXP13.py - C:/Users/dhanu/CNSEXP13.py (3.11.0)

File Edit Format Run Options Window Help

```
ct=str(input("ENTER THE PLAIN TEXT : "))
a=int(input("ENTER a : "))
b=int(input("ENTER b : "))
letter="abcdefghijklmnopqrstuvwxyz"
dec=""
for x in ct:
    en=0
    if x in letter:
        pos=letter.find(x)
        en=((a*pos)+b)%26
        dec+=letter[en]
print("CIPHER TEXT : "+dec)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSEXP9.1.py ==========
======
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
========== RESTART: C:/Users/dhanu/CNSEXP11.py ==========
Enter the message: hello everyone
Enter the key: 4
plain text:  hello everyone
cipher text:  olssv lclyfvul
>>>
========== RESTART: C:/Users/dhanu/CNSEXP12.py ==========
Enter the message to be decrypted: olssv lclyfvul
Enter the key for decryption: 4
cipher text:  olssv lclyfvul
plain text (message):  hello everyone
>>>
========== RESTART: C:/Users/dhanu/CNSEXP13.py ==========
ENTER THE PLAIN TEXT : meet me  after the lunch
ENTER a : 46
ENTER b : \40
CIPHER TEXT : uqqeuqokeqqeyqayocy
>>>
```

```
cipher=""
letter="abcdefghijklmnopqrstuvwxyz"
common=max(set(pt),key=pt.count)
print("COMMON LETTER : "+common)
if common in letter:
    common=letter.find(common)
key=common-6
if (key<0):
    key=26-key
for i in pt:
    if i in letter:
        pos=letter.find(i)
        new_pos=(pos+key)%26
        new_char=letter[new_pos]
        cipher+=new_char
print("CIPHER TEXT : "+cipher)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/dhanu/CNSEXP9.1.py ==========
======
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
========== RESTART: C:/Users/dhanu/CNSEXP11.py ==========
Enter the message: hello everyone
Enter the key: 4
plain text:  hello everyone
cipher text:  olssv lclyfvul
>>>
========== RESTART: C:/Users/dhanu/CNSEXP12.py ==========
Enter the message to be decrypted: olssv lclyfvul
Enter the key for decryption: 4
cipher text:  olssv lclyfvul
plain text (message):  hello everyone
>>>
========== RESTART: C:/Users/dhanu/CNSEXP13.py ==========
ENTER THE PLAIN TEXT : meet me  after the lunch
ENTER a : 46
ENTER b : \40
CIPHER TEXT : uqqeuqokeqqeyqayocy
>>>
========== RESTART: C:/Users/dhanu/CNSEXP14.py ==========
ENTER THE PLAIN TEXT : meet
COMMON LETTER : e
CIPHER TEXT : oggv
>>>
```

```
import math
def row(s,key):
    temp=[]
    for i in key:
        if i not in temp:
            temp.append(i)
    k=""
    for i in temp:
        k+=i
    print("The key used for encryption is: ",k)
    b=math.ceil(len(s)/len(k))
    if(b<len(k)):
        b=b+(len(k)-b)
    arr=[['_' for i in range(len(k))]
         for j in range(b)]
    i=0
    j=0
    for h in range(len(s)):
        arr[i][j]=s[h]
        j+=1
        if(j>len(k)-1):
            j=0
            i+=1
    print("The message matrix is: ")
    for i in arr:
        print(i)
    cipher_text=""
    kk=sorted(k)
    for i in kk:
        h=k.index(i)
        for j in range(len(arr)):
            cipher_text+=arr[j][h]
    print("The cipher text is: ",cipher_text)
msg=input("Enter the message: ")
key=input("Enter the key in alphabets: ")
row(msg,key)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Enter the message to be decrypted: akmeti
Enter the one time pad: artery
Decrypted message is:  attack
>>>
================== RESTART: C:/Users/dhanu/CNSEXP11.py ==================
Enter the message: hello everyone
Enter the key: 4
plain text:  hello everyone
cipher text:  olssv lclyfvul
>>>
================== RESTART: C:/Users/dhanu/CNSEXP12.py ==================
Enter the message to be decrypted: olssv lclyfvul
Enter the key for decryption: 4
cipher text:  olssv lclyfvul
plain text (message):  hello everyone
>>>
================== RESTART: C:/Users/dhanu/CNSEXP13.py ==================
ENTER THE PLAIN TEXT : meet me  after the lunch
ENTER a : 46
ENTER b : \40
CIPHER TEXT : uqqeuqokeqqeyqayocy
>>>
================== RESTART: C:/Users/dhanu/CNSEXP14.py ==================
ENTER THE PLAIN TEXT : meet
COMMON LETTER : e
CIPHER TEXT : oggv
>>>
================== RESTART: C:/Users/dhanu/cnsexp15.py ==================
Enter the message: welcome everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['w', 'e', 'l']
['c', 'o', 'm']
['e', ' ', 'e']
['v', 'e', 'r']
['y', 'o', 'n']
['e', '_', '_']
The cipher text is:  eo eo_wcevyelmern_
>>>
```

```
import math
def row(s,key):
    temp=[]
    for i in key:
        if i not in temp:
            temp.append(i)
    k=""
    for i in temp:
        k+=i
    print("The key used for encryption is: ",k)
    arr=[['' for i in range(len(k))]
         for j in range(int(len(s)/len(k)))]
    kk=sorted(k)
    d=0
    for i in kk:
        h=k.index(i)
        for j in range(len(k)):
            arr[j][h]=s[d]
            d+=1
    print("The message matrix is: ")
    for i in arr:
        print(i)
    plain_text=""
    for i in arr:
        for j in i:
            plain_text+=j
    print("The plain text is: ",plain_text)
msg=input("Enter the message to be decrypted: ")
key=input("Enter the key in alphabets: ")
row(msg,key)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Enter the key for decryption: 4
cipher text:  olssv lclyfvul
plain text (message):  hello everyone
>>>
========== RESTART: C:/Users/dhanu/CNSEXP13.py ==========
ENTER THE PLAIN TEXT : meet me  after the lunch
ENTER a : 46
ENTER b : \40
CIPHER TEXT : uqqeuqokeqqeyqayocy
>>>
========== RESTART: C:/Users/dhanu/CNSEXP14.py ==========
ENTER THE PLAIN TEXT : meet
COMMON LETTER : e
CIPHER TEXT : oggv
>>>
========== RESTART: C:/Users/dhanu/cnsexp15.py ==========
Enter the message: welcome everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['w', 'e', 'l']
['c', 'o', 'm']
['e', ' ', 'e']
['v', 'e', 'r']
['y', 'o', 'n']
['e', '_', '_']
The cipher text is:  eo eo_wcevyelmern_
>>>
========== RESTART: C:/Users/dhanu/CNSEXP16.py ==========
Enter the message to be decrypted: hello everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['l', 'h', 'e']
['o', 'e', 'v']
[' ', 'l', 'e']
['', '', '']
['', '', '']
The plain text is:  lheoev le
>>>
```

```
    arr=[]
    i=0
    while(i<n-1):
        arr.append(i)
        i+=1
    while(i>0):
        arr.append(i)
        i-=1
    return(arr)
def railfence(s,n):
    s=s.lower()
    L=sequence(n)
    print("The raw sequence of indices: ",L)
    temp=L
    while(len(s)>len(L)):
        L=L+temp
    for i in range(len(L)-len(s)):
        L.pop()
    print("The row indices of the characters in the given
    print("Transformed message for encryption: ",s)
    num=0
    cipher_text=""
    while(num<n):
        for i in range(L.count(num)):
            cipher_text=cipher_text+s[L.index(num)]
            L[L.index(num)]=n
        num+=1
    print("The cipher text is: ",cipher_text)
plain_text=input("Enter the string to be encrypted: ")
n=int(input("Enter the number of rails: "))
railfence(plain_text,n)
```

File Edit Shell Debug Options Window Help

```
>>>
========== RESTART: C:/Users/dhanu/CNSEXP14.py ==========
ENTER THE PLAIN TEXT : meet
COMMON LETTER : e
CIPHER TEXT : oggv
>>>
========== RESTART: C:/Users/dhanu/cnsexp15.py ==========
Enter the message: welcome everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['w', 'e', 'l']
['c', 'o', 'm']
['e', ' ', 'e']
['v', 'e', 'r']
['y', 'o', 'n']
['e', '_', '_']
The cipher text is:  eo eo_wcevyelmern_
>>>
========== RESTART: C:/Users/dhanu/CNSEXP16.py ==========
Enter the message to be decrypted: hello everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['l', 'h', 'e']
['o', 'e', 'v']
[' ', 'l', 'e']
['', '', '']
['', '', '']
The plain text is:  lheoev le
>>>
========== RESTART: C:/Users/dhanu/CNSEXP18.py ==========
Enter the string to be encrypted: meet me after the yoga party
Enter the number of rails: 4
The raw sequence of indices:  [0, 1, 2, 3, 2, 1]
The row indices of the characters in the given string:  [0, 1, 2, 3, 2, 1,
0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3]
Transformed message for encryption:  meet me after the yoga party
The cipher text is:  meryaem e  opre atteg ttfhay
>>>
```

```
def sequence(n):
    arr=[]
    i=0
    while(i<n-1):
        arr.append(i)
        i+=1
    while(i>0):
        arr.append(i)
        i-=1
    return(arr)
def railfence(cipher_text,n):
    cipher_text=cipher_text.lower()
    L=sequence(n)
    print("The raw sequence of indices: ",L)
    temp=L
    while(len(cipher_text)>len(L)):
        L=L+temp
    for i in range(len(L)-len(cipher_text)):
        L.pop()
    temp1=sorted(L)
    print("The row indices of the characters in the cipher string: "
    print("The row indices of the characters in the plain string: ",
    print("Transformed message for decryption: ",cipher_text)
    plain_text=""
    for i in L:
        k=temp1.index(i)
        temp1[k]=n
        plain_text+=cipher_text[k]
    print("The cipher text is: ",plain_text)
cipher_text=input("Enter the string to be decrypted: ")
n=int(input("Enter the number of rails: "))
railfence(cipher_text,n)
```

File Edit Shell Debug Options Window Help

```
========== RESTART: C:/Users/dhanu/CNSEXP14.py ==========
ENTER THE PLAIN TEXT : meet
COMMON LETTER : e
CIPHER TEXT : oggv
>>>
========== RESTART: C:/Users/dhanu/cnsexp15.py ==========
Enter the message: welcome everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['w', 'e', 'l']
['c', 'o', 'm']
['e', ' ', 'e']
['v', 'e', 'r']
['y', 'o', 'n']
['e', '_', '_']
The cipher text is:  eo eo_wcevyelmern_
>>>
========== RESTART: C:/Users/dhanu/CNSEXP16.py ==========
Enter the message to be decrypted: hello everyone
Enter the key in alphabets: daddy
The key used for encryption is:  day
The message matrix is:
['l', 'h', 'e']
['o', 'e', 'v']
[' ', 'l', 'e']
['', '', '']
['', '', '']
The plain text is:  lheoev le
>>>
========== RESTART: C:/Users/dhanu/CNSEXP18.py ==========
Enter the string to be encrypted: meet me after the yoga party
Enter the number of rails: 4
The raw sequence of indices:  [0, 1, 2, 3, 2, 1]
The row indices of the characters in the given string:  [0, 1, 2, 3, 2, 1, 0,
1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3]
Transformed message for encryption:  meet me after the yoga party
The cipher text is:  meryaem e  opre atteg ttfhay
>>>
========== RESTART: C:/Users/dhanu/CNSEXP18.1.py ==========
```

```
File  Edit  Format  Run  Options  Window  Help
import math
p = int(input("Enter p: "))
q = int(input("Enter q: "))
n = p*q
print("n: ", n)
phi = (p-1)*(q-1)
print("phi: ",phi)
e = int(input("Enter e: "))
while(e<phi):
    if (math.gcd(e,phi)==1):
        break
    else:
        e+=1
print("e: ", e)
j = 0
while True:
    if ((j * e) % phi == 1):
        d = j
        break
    j += 1
print("d: ", d)
print(f'Public key: {e, n}')
print(f'Private key: {d, n}')
msg = int(input("Enter message: "))
print(f'Original message:{msg}')
C = pow(msg, e)
C = math.fmod(C, n)
print(f'Encrypted message: {C}')
M = pow(C, d)
M = math.fmod(M, n)
```

```
IDLE Shell 3.11.0
File  Edit  Shell  Debug  Options  Window  Help
['o', 'e', 'v']
[' ', 'l', 'e']
['', '', '']
['', '', '']
The plain text is:  lheoev le
>>>
========== RESTART: C:/Users/dhanu/CNSEXP18.py ==========
Enter the string to be encrypted: meet me after the yoga party
Enter the number of rails: 4
The raw sequence of indices:  [0, 1, 2, 3, 2, 1]
The row indices of the characters in the given string:  [0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3]
Transformed message for encryption:  meet me after the yoga party
The cipher text is:  meryaem e  opre atteg ttfhay
>>>
========== RESTART: C:/Users/dhanu/CNSEXP18.1.py ==========
Enter the string to be decrypted: math gr etefe teo aate artpy
Enter the number of rails: 3
The raw sequence of indices:  [0, 1, 2, 1]
The row indices of the characters in the cipher string:  [0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1]
The row indices of the characters in the plain string:  [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2]
Transformed message for decryption:  math gr etefe teo aate artpy
The cipher text is:  m eeat etfaeh rt etog parayt
>>>
========== RESTART: C:/Users/dhanu/CNSEXP19.py ==========
Enter p: 7
Enter q: 11
n:  77
phi:  60
Enter e: 7
e:  7
d:  43
Public key: (7, 77)
Private key: (43, 77)
Enter message: 9
Original message:9
Encrypted message: 37.0
>>>
```

```
    if len(msg)%2!=0:
        msg=msg[:]+'X'
    print("CIPHER TEXT:",end=' ')
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
        loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
            print("{}{}".format(my_matrix[(loc[0]+1)%5][loc[1]],my_matrix[(loc1[0]+1
        elif loc[0]==loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1]+1)%5],my_matrix[loc1[0]][(
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]
        i=i+2

def decrypt():  #decryption
    msg=str(input("ENTER CIPHER TEXT:"))
    msg=msg.upper()
    msg=msg.replace(" ", "")
    print("PLAIN TEXT:",end=' ')
    i=0
    while i<len(msg):
        loc=list()
        loc=locindex(msg[i])
        loc1=list()
        loc1=locindex(msg[i+1])
        if loc[1]==loc1[1]:
            print("{}{}".format(my_matrix[(loc[0]-1)%5][loc[1]],my_matrix[(loc1[0]-1
        elif loc[0]==loc1[0]:
            print("{}{}".format(my_matrix[loc[0]][(loc[1]-1)%5],my_matrix[loc1[0]][(
        else:
            print("{}{}".format(my_matrix[loc[0]][loc1[1]],my_matrix[loc1[0]][loc[1]
        i=i+2

while(1):
    choice=int(input("\n 1.Encryption \n 2.Decryption: \n 3.EXIT"))
    if choice==1:
        encrypt()
    elif choice==2:
        decrypt()
    elif choice==3:
        exit()
    else:
        print("Choose correct choice")
```

File Edit Shell Debug Options Window Help

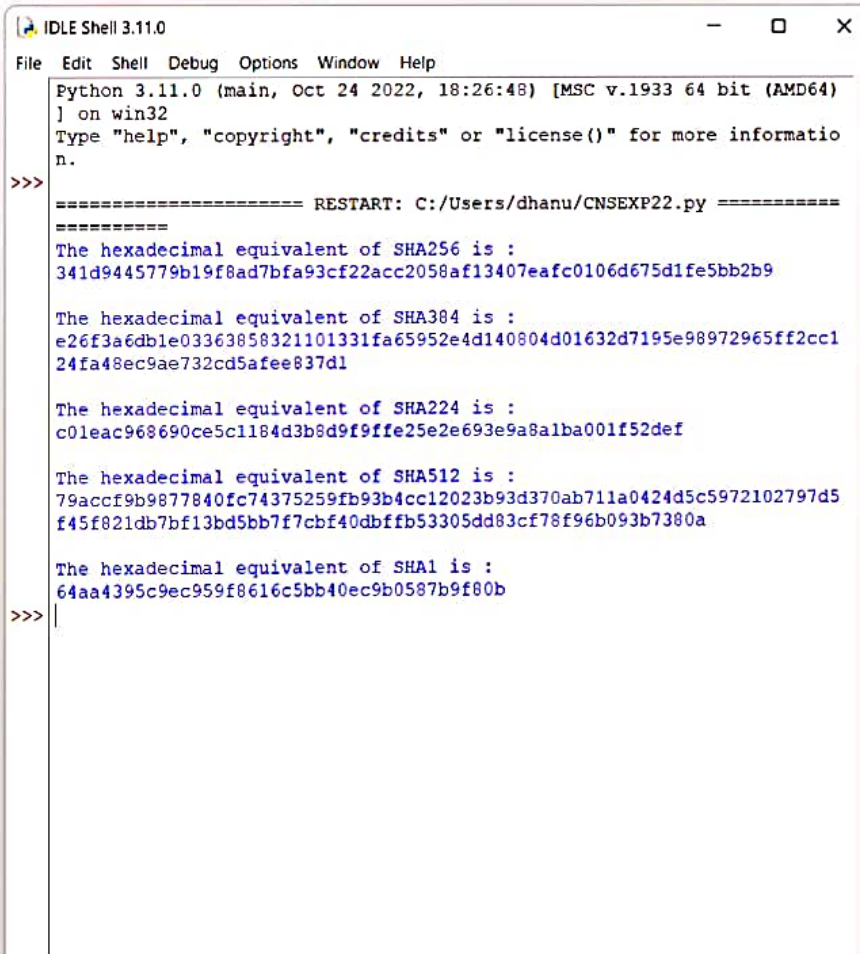
```
1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3, 2, 1, 0, 1, 2, 3]
Transformed message for encryption:  meet me after the yoga party
The cipher text is:  meryaem e  opre atteg ttfhay

>>>

================== RESTART: C:/Users/dhanu/CNSEXP18.1.py ==================
Enter the string to be decrypted: math gr etefe teo aate artpy
Enter the number of rails: 3
The raw sequence of indices:  [0, 1, 2, 1]
The row indices of the characters in the cipher string:  [0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2
, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1]
The row indices of the characters in the plain string:  [0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2]
Transformed message for decryption:  math gr etefe teo aate artpy
The cipher text is:  m eeat etfaeh rt etog parayt

>>>

==================== RESTART: C:/Users/dhanu/CNSEXP19.py ====================
Enter p: 7
Enter q: 11
n:  77
phi:  60
Enter e: 7
e:  7
d:  43
Public key: (7, 77)
Private key: (43, 77)
Enter message: 9
Original message:9
Encrypted message: 37.0

>>>

========================== RESTART: C:/Users/dhanu/CNSEXP20.py ==========================
Enter key4

 1.Encryption
 2.Decryption:
 3.EXITwe together forever
```

Ln: 98 Col: 0

successfully

Experiment-21

```python
import hashlib
str = "Hello everyone"
result = hashlib.sha256(str.encode())
print("The hexadecimal equivalent of SHA256 is : ")
print(result.hexdigest())
print ("\r")
str = "Hello everyone"
result = hashlib.sha384(str.encode())
print("The hexadecimal equivalent of SHA384 is : ")
print(result.hexdigest())
print ("\r")
str = "Hello everyone"
result = hashlib.sha224(str.encode())
print("The hexadecimal equivalent of SHA224 is : ")
print(result.hexdigest())
print ("\r")
str ="Hello everyone"
result = hashlib.sha512(str.encode())
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())
print ("\r")
str = "Hello everyone"
result = hashlib.sha1(str.encode())
print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```



CNSEXSP23.py - C:/Users/dhanu/CNSEXSP23.py (3.11.0)

File Edit Format Run Options Window Help

```
def hex2bin(s):
        mp = {'0': "0000",
              '1': "0001",
              '2': "0010",
              '3': "0011",
              '4': "0100",
              '5': "0101",
              '6': "0110",
              '7': "0111",
              '8': "1000",
              '9': "1001",
              'A': "1010",
              'B': "1011",
              'C': "1100",
              'D': "1101",
              'E': "1110",
              'F': "1111"}
        bin = ""
        for i in range(len(s)):
                bin = bin + mp[s[i]]
        return bin


def bin2hex(s):
        mp = {"0000": '0',
              "0001": '1',
              "0010": '2',
              "0011": '3',
              "0100": '4',
              "0101": '5',
              "0110": '6',
              "0111": '7',
              "1000": '8',
              "1001": '9',
              "1010": 'A',
              "1011": 'B',
              "1100": 'C',
              "1101": 'D',
              "1110": 'E',
              "1111": 'F'}
        hex = ""
        for i in range(0, len(s), 4):
                ch = ""
                ch = ch + s[i]
                ch = ch + s[i + 1]
                ch = ch + s[i + 2]
                ch = ch + s[i + 3]
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
==========================================
Encryption
After initial permutation 14A7D67818CA18AD
Round  1   18CA18AD   5A78E394   194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
Round  9   308BEE97   10AF9D37   84BB4473DCCC
Round  10   10AF9D37   6CA6CB20   02765708B5BF
Round  11   6CA6CB20   FF3C485F   6D5560AF7CA5
Round  12   FF3C485F   22A5963B   C2C1E96A4BF3
Round  13   22A5963B   387CCDAA   99C31397C91F
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text :  C0B7A8D05F3A829C
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536
```

```
def determinantOfMatrix(mat, n):
    D = 0
    if (n == 1):
        return mat[0][0]
    temp = [[0 for x in range(n)]
            for y in range(n)]
    sign = 1
    for f in range(n):
        getCofactor(mat, temp, 0, f, n)
        D += (sign * mat[0][f] *
              determinantOfMatrix(temp, n - 1))
        sign = -sign
    return D
def isInvertible(mat, n):
    if (determinantOfMatrix(mat, n) != 0):
        return True
    else:
        return False
def multiply_and_convert(key,message):
    res_num = [[0 for x in range(len(message[0]))] for y in range
    for i in range(len(key)):
        for j in range(len(message[0])):
            for k in range(len(message)):
                res_num[i][j]+=key[i][k] * message[k][j]
    res_alpha = [['' for x in range(len(message[0]))] for y in ra
    for i in range(len(key)):
        for j in range(len(message[0])):
            res_alpha[i][j]+=chr((res_num[i][j]%26)+97)
    return(res_alpha)
n=int(input("What will be the order of square matrix: "))
s=input("Enter the key: ")
key=generate_key(n,s)
if (isInvertible(key, len(key))):
    print("Yes it is invertable and can be decrypted")
else:
    print("No it is not invertable and cannot be decrypted")
plain_text=input("Enter the message: ")
message=message_matrix(plain_text,n)
final_message=''
for i in message:
    sub=multiply_and_convert(key,i)
    for j in sub:
        for k in j:
            final_message+=k
print("plain message: ",plain_text)
print("final encrypted message: ",final_message)
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text :  C0B7A8D05F3A829C
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536
>>>
================ RESTART: C:/Users/dhanu/CNSEXP24.py ================
What will be the order of square matrix: 3
Enter the key: GYBNQKURP
The key matrix (3x3) is:
['gyb', 'nqk', 'urp']
[6, 24, 1]
[13, 16, 10]
[20, 17, 15]
Yes it is invertable and can be decrypted
Enter the message: ACT
Converted plain_text for encryption:  act
The column matrices of plain text in numbers are:
[[0], [2], [19]]
plain message:  ACT
final encrypted message:  poh
```