```
#CaesarCipher

uppercase_letters = ['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']


def caesar_cipher(msg, k):
    cipher_text = ""
    for ch in msg:
        if ch.isalpha():
            cipher_text += uppercase_letters[(uppercase_letters.index(ch) + k) % 26]
        else:
            cipher_text += ch
    print("cipher text:",cipher_text)

    plain_text = ""
    for ch in cipher_text:
        if ch.isalpha():
            plain_text += uppercase_letters[(uppercase_letters.index(ch) - k) % 26]
        else:
            plain_text += ch
    print("plain text:",plain_text)


message = input("enter the message: ").upper()
key = int(input("enter the key: "))
caesar_cipher(message, key)
```

```
#SubstitutionCipher

alphabet = ['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']
key = []

message = input("enter the message: ").lower()
key_word = input("enter the keyword: ")

for i in key_word:
    if i not in key:
        key.append(i)
for i in alphabet:
    if i not in key:
        key.append(i)

ciphertext = ""
for i in message:
    if i.isalpha():
        ciphertext += key[alphabet.index(i)]
    else:
        ciphertext += i
print("cipher text:",ciphertext)

plaintext = ""
for i in ciphertext:
    if i.isalpha():
        plaintext += alphabet[key.index(i)]
    else:
        plaintext += i
print("plain text:",plaintext)
```

```
#DiffieHellmanKeyExchange

import sympy
import random

p = 0
while not(sympy.isprime(p)):
    p = random.randint(100,200)
g = sympy.primitive_root(p)

a = random.randint(1,100)
b = random.randint(1,100)

A = pow(g,a,p)
B = pow(g,b,p)

print(f"(p={p}, g={g}, a={a}, b={b}, A={A}, B={B})")
print("Alice secret key:",pow(B,a,p))
print("Bob secret key:",pow(A,b,p))
```

```
#SHA1

import hashlib

text = input("enter text:\n")
text_bytes = text.encode()

sha1_hash = hashlib.sha1(text_bytes)
digest = sha1_hash.hexdigest()

print("SHA 1 message digest:",digest)
```

```
#HillCipher

import random, math, numpy, sympy

alphabet = ['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']

text = input("enter text: ").upper()
n = 1
while n*n < len(text):
    n += 1
while n*n > len(text):
    text += "X"
P = [[alphabet.index(text[i*n + j]) for j in range(n)] for i in range(n)]

while True:
    K = [[random.randint(1,10) for _ in range(n)] for _ in range(n)]
    det = int(round(numpy.linalg.det(K))) % 26
    if math.gcd(det,26)==1:
        break

X = sympy.Matrix(K)
Ki = numpy.array(X.inv_mod(26)).astype(int)

C = numpy.dot(K,P) % 26
PT = numpy.dot(Ki,C) % 26

cipher = ""
plaintext = ""
for i,j in zip(C,PT):
    for _ in range(n):
     cipher += alphabet[int(i[_])%26]
     plaintext += alphabet[int(j[_])%26]
plaintext = plaintext.rstrip("X")

print("cipher text:", cipher)
print("decrypted text:", plaintext)
```

```
#RSA

import random, math, sympy

def e_value(phin_val):
    while True:
        e_val = random.randint(2,phin_val-1)
        if math.gcd(e_val,phin_val)==1:
            return e_val

def encryption(plain_text,e_val,n_val):
    cipher_text = []
    for ch in plain_text:
        cipher_text.append(pow(ord(ch),e_val,n_val))
    return cipher_text

def decryption(enc_text,d_val,n_val):
    dec_text = ""
    for i in enc_text:
        dec_text += chr(pow(i,d_val,n_val))
    return dec_text

p,q = 0,0
while not(sympy.isprime(p) and sympy.isprime(q)) or p == q:
    p = random.randint(1,100)
    q = random.randint(1,100)

n = p*q
phin = (p-1)*(q-1)
e = e_value(phin)
d = pow(e, -1, phin)

text = input("enter text: ")
ciphertext = encryption(text,e,n)
print("cipher text:",ciphertext)
dectext = decryption(ciphertext,d,n)
print("decrypted text",dectext)
```

```
#DES

from Crypto.Cipher import DES
from Crypto.Util.Padding import pad, unpad

key = b'12345678'
msg = input("enter text: ").encode()
cipher = DES.new(key, DES.MODE_ECB).encrypt(pad(msg, 8))
print("cipher:", cipher.hex())
plain = DES.new(key, DES.MODE_ECB).decrypt(cipher)
print("decrypted:", unpad(plain, 8).decode())
```