import random  
import string  
  
class TextEncoding:  
  
 @staticmethod  
 def determine\_alphabet(text):  
 if text[0] in string.ascii\_lowercase:  
 return string.ascii\_lowercase + string.digits  
 else:  
 return "абвгдеёжзийклмнопрстуфхцчшщъыьэюя" + string.digits  
  
 @staticmethod  
 def generate\_key(size, alphabet):  
 return "".join(random.choice(alphabet) for \_ in range(size))  
  
 @staticmethod  
 def to\_hex(coding):  
 return " ".join(hex(ord(character))[2:] for character in coding)  
  
 @staticmethod  
 def encode\_string(text, key):  
 return "".join(chr(ord(char) ^ ord(key\_char)) for char, key\_char in zip(text, key))  
  
 @staticmethod  
 def find\_possible\_keys(text, fragment):  
 key\_length = len(fragment)  
 possible\_keys = []  
  
 for index in range(len(text) - key\_length + 1):  
 key = [chr(ord(char) ^ ord(key\_char)) for char, key\_char in zip(text[index:index + key\_length], fragment)]  
 presumed\_plaintext = TextEncoding.encode\_string(text, key)  
  
 if fragment in presumed\_plaintext:  
 possible\_keys.append(''.join(key))  
  
 return possible\_keys  
  
  
plaintext = input("Введите открытый текст: ")  
alphabet = TextEncoding.determine\_alphabet(plaintext)  
key = TextEncoding.generate\_key(len(plaintext), alphabet)  
  
print(f"Ключ: {key}", f"Ключ в 16 бит: {TextEncoding.to\_hex(key)}", sep='\n')  
  
ciphertext = TextEncoding.encode\_string(plaintext, key)  
print(f"Зашифрованный текст: {ciphertext}", f"Зашифрованный текст в 16 бит: {TextEncoding.to\_hex(ciphertext)}",  
 sep='\n')  
  
decrypted\_text = TextEncoding.encode\_string(ciphertext, key)  
print("Расшифрованный текст:", decrypted\_text)  
  
known\_fragment = input("Введите фрагмент открытого текста: ")  
possible\_keys = TextEncoding.find\_possible\_keys(ciphertext, known\_fragment)  
print("Возможные ключи для шифротекста:", possible\_keys)









