МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ЛАБОРАТОРНЫЕ РАБОТЫ ПО ПРЕДМЕТУ

«Программирование криптографических алгоритмов»

Выполнил:

Барышников С.С.
гр. 191-351

Преподаватель:

Бутакова Н.Г.

Москва 2021 г.

Содержание

Аннотация

Среда программирования: Visual Studio Code

Язык программирования: Python 3

Процедуры для запуска программы: $ python3 <имя_файла>.py

Пословица-тест: Время, приливы и отливы не ждут человека.

Текст для проверки работы: Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

Интерфейс: #в разработке#

Постоянный модуль

Код модуля base.py используемый для предотвращения дублирования кода, используется во всех последующих программах:

```
import re

alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

dict = {'.': 'тчк', ',': 'зпт'}


def replace_all_to(input_text, dict):
    input_text = input_text.replace(' ', '')
    for i, j in dict.items():
        input_text = input_text.replace(i, j)
    return input_text


def replace_all_from(input_text, dict):
    for i, j in dict.items():
        input_text = input_text.replace(j, i)
    return input_text


def file_to_string(name):
    with open(name) as f:
        input_short_text = " ".join([l.rstrip() for l in f]) + ' '
    return input_short_text.lower()


def input_for_cipher_short():
    return replace_all_to(file_to_string('short.txt'), dict)


def input_for_cipher_long():
    return replace_all_to(file_to_string('long.txt'), dict)


def output_from_decrypted(decrypted_text):
    return replace_all_from(decrypted_text, dict)
```

Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ

8. Матричный шифр

Шифр Хилла — полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике. Изобретён американским математиком Лестером Хиллом в 1929 году. Это был первый шифр, который позволил на практике (хотя и с трудом) одновременно оперировать более чем с тремя символами. Шифр Хилла не нашёл практического применения в криптографии из-за слабой устойчивости ко взлому и отсутствия описания алгоритмов генерации прямых и обратных матриц большого размера.

Код программы:

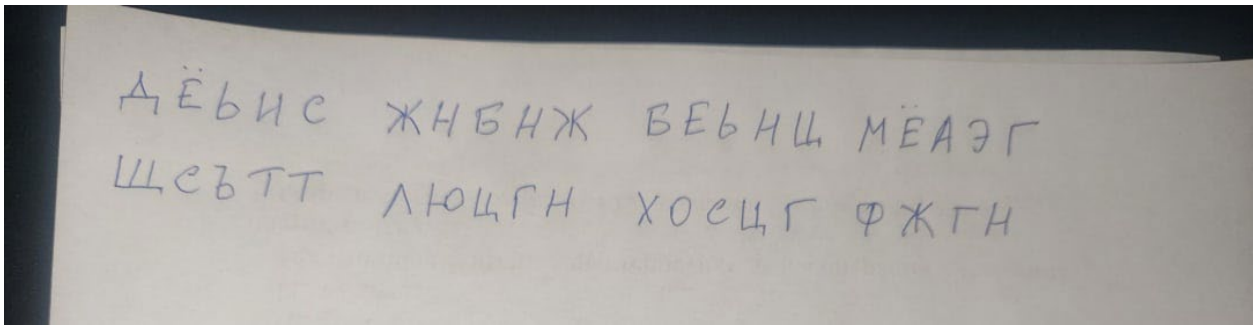
```
from base import alphabet, input_for_cipher_short, input_for_cipher_long, out
put_from_decrypted
import numpy as np
from egcd import egcd

inp = input('Введите матрицу в строку через пробел: ')
inp = inp.split(' ')

# 3 10 20 20 19 17 23 78 17

key = np.matrix([[int(inp[0]), int(inp[1]), int(inp[2])], [int(inp[3]), int(
    inp[4]), int(inp[5])], [int(inp[6]), int(inp[7]), int(inp[8])]])
letter_to_index = dict(zip(alphabet, range(len(alphabet))))
index_to_letter = dict(zip(range(len(alphabet)), alphabet))


def matrix_mod_inv(matrix, modulus):
    det = int(np.round(np.linalg.det(matrix)))
    det_inv = egcd(det, modulus)[1] % modulus
    matrix_modulus_inv = (
        det_inv * np.round(det * np.linalg.inv(matrix)).astype(int) % modulus
    )
    return matrix_modulus_inv


def matrix_encode(message, K):
    encrypted = ""
    message_in_numbers = []
    for letter in message:
        message_in_numbers.append(letter_to_index[letter])

    split_P = [
        message_in_numbers[i: i + int(K.shape[0])]
        for i in range(0, len(message_in_numbers), int(K.shape[0]))
    ]
    for P in split_P:
```

```
        P = np.transpose(np.asarray(P))[:, np.newaxis]
        while P.shape[0] != K.shape[0]:
            P = np.append(P, letter_to_index[" "])[:, np.newaxis]
        numbers = np.dot(K, P) % len(alphabet)
        n = numbers.shape[0]
        for idx in range(n):
            number = int(numbers[idx, 0])
            encrypted += index_to_letter[number]
    return encrypted


def matrix_decode(cipher, Kinv):
    decrypted = ""
    cipher_in_numbers = []
    for letter in cipher:
        cipher_in_numbers.append(letter_to_index[letter])
    split_C = [
        cipher_in_numbers[i: i + int(Kinv.shape[0])]
        for i in range(0, len(cipher_in_numbers), int(Kinv.shape[0]))
    ]
    for C in split_C:
        C = np.transpose(np.asarray(C))[:, np.newaxis]
        numbers = np.dot(Kinv, C) % len(alphabet)
        n = numbers.shape[0]

        for idx in range(n):
            number = int(numbers[idx, 0])
            decrypted += index_to_letter[number]
    return decrypted
# вывод результатов работы программы
print(f'''
Матричный шифр:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_short(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_short(), key), matrix_mod_inv(key, len(alphabet)))).repl
ace(' ', '')}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_long(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_long(), key), matrix_mod_inv(key, len(alphabet)))).repla
ce(' ', '')}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab03_8_matrix.py
Введите матрицу в строку через пробел: 3 10 20 20 19 17 23 78 17

Матричный шифр:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
дёьисжнбнжбеьнцмёаэгщсъттлюцгнхосцгфжгн

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
щвичнкфящёжщрээншусзуйдътюцбёэъяшщктыёжщктийдмпжбзярюмсигфохжртичаужбвфэовккв
кыкзчвзяжтиьудюяшгмювжаепыэсофрдюейхьёзпфрдчвзйсоюбёъштифчыхвлицхилщндкыбоктт
жжгнщъябллфпыдчикъллвусфвпъбёъётстрэуижлиатйчлихчрозюпдмзмлёъзпжюцбёввижгнйдм
фшжтлйбншедщжгнншусзуйдътюцбёдёьбллющрюяшядмдбмйихюяшбээзмлфрдяпйшцпнфшшвкающ
мдуйюьсщигфъарыгхзчэдкхтщтяшцтиоььфчлчнкфящзфэхосжгнёжщюжръчасожцповусштдбнив
щусзйдзшцпвфшьйотрцачиаъшцъйёгфрдэъпчияъжпшеьёттскиенаеэёгсюъвуефъацжщофабюяз
ърцпяшлнкчтшюистщоъшхгчкгёядьчтйёьчеажщъёяъьявшзмлфрдяпйэдкхтщщплгйяоььъжхлжщ
цэоеяшъбьоиыфэцъаптъхйжгнзыгвядгобомиььхеёжкрмзпчичнкыфмшэомюцмээусзлеэзнкфме
дщкфрдимежгнщцгмфвтэяхосймфыгщмнцьвпёдцщэобзчожтвнокдызпмвщвгщзыюжрътюцбёпчръ
щръмяюбёмллаьпяыогнобрчгюяшажиьгдмжзнкёчичыдгъяьиммсшмюцфммрэнохюёжюпплкяыщбя
ьъиохжиъожфцшщъщьачижччачижгншвквхшёсошязцезрэлцъсшдыоемъсимзецарцётьнцфжуэфг
ъмммвщебжюммвфудпулкктиэхосйбмтэячтфтщогземюцлиьвъжщцкбвжяьбнфжччхлтмъбёючжъя
ётгьеобэобеежржгнщпгхвдкнёыъчдбнмюцъммэуилэчфпыщчишюззмлбасзнкэучцпозбнжяымещ
кщтжвпуипоэяйщерджбйозюьщштщрэжюусзчйдчшрозжншрэнъенэъягнх

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходя
щийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпублик
аций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.но
можноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднука
ртину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотыс
ячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблят
ьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеи
зменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиили
без.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностоль
коразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,
таккакэтопустоеместо.однаконекоторыефирмыибиржividятсправедливымставитьстоимо
стьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговоспр
иятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольши
нствунужнаценазатысячузнаковбезпробелов.
```

Карточка:



9. Шифр Плейфера

Шифр Плейфера или квадрат Плейфера — ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 году английским физиком Чарльзом Уитстоном, но названа именем лорда Лайона Плейфера, который внёс большой вклад в продвижение использования данной системы шифрования в государственной службе. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в шифре подстановки и в более сложных системах шифрования Виженера. Таким образом, шифр Плейфера более устойчив к взлому по сравнению с шифром простой замены, так как усложняется его частотный анализ. Он может быть проведён, но не для символов, а для биграмм. Так как возможных биграмм больше, чем символов, анализ значительно более трудоёмок и требует большего объёма зашифрованного текста.

Код программы:

```
from base import alphabet, input_for_cipher_short, input_for_cipher_long, out
put_from_decrypted

alphabet = alphabet.replace(' ', '') + 'abc'

key = str(input('Введите ключ: '))


def playfair_encode(clearText, key):
    text = clearText
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet)):
        bool_buff = False
        for j in range(len(key)):
            if alphabet[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet[i])
    mtx_abt_j = []
    counter = 0
    for j in range(6):
        mtx_abt_i = []
        for i in range(6):
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)
    if len(text) % 2 == 1:
```

```
        text = text + "я"
    enc_text = ""
    for t in range(0, len(text), 2):
        flag = True
        for j_1 in range(6):
            if flag == False:
                break
            for i_1 in range(6):
                if flag == False:
                    break
                if mtx_abt_j[j_1][i_1] == text[t]:
                    for j_2 in range(6):
                        if flag == False:
                            break
                        for i_2 in range(6):
                            if mtx_abt_j[j_2][i_2] == text[t+1]:
                                if j_1 != j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_2] + \
                                        mtx_abt_j[j_2][i_1]
                                elif j_1 == j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][(i_1+1) % 6] + \
                                        mtx_abt_j[j_2][(i_2+1) % 6]
                                elif j_1 != j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[(j_1+1) % 5][i_1] + \
                                        mtx_abt_j[(j_2+1) % 5][i_2]
                                elif j_1 == j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_1] + \
                                        mtx_abt_j[j_1][i_1]
                                flag = False
                                break
    return enc_text


def playfair_decode(clearText, key):
    text = clearText
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet)):
        bool_buff = False
        for j in range(len(key)):
            if alphabet[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet[i])
```

```
    mtx_abt_j = []
    counter = 0
    for j in range(6):
        mtx_abt_i = []
        for i in range(6):
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)
    if len(text) % 2 == 1:
        text = text + "я"
    enc_text = ""
    for t in range(0, len(text), 2):
        flag = True
        for j_1 in range(6):
            if flag == False:
                break
            for i_1 in range(6):
                if flag == False:
                    break
                if mtx_abt_j[j_1][i_1] == text[t]:
                    for j_2 in range(6):
                        if flag == False:
                            break
                        for i_2 in range(6):
                            if mtx_abt_j[j_2][i_2] == text[t+1]:
                                if j_1 != j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_2] + \
                                        mtx_abt_j[j_2][i_1]
                                elif j_1 == j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][(i_1-1) % 6] + \
                                        mtx_abt_j[j_2][(i_2-1) % 6]
                                elif j_1 != j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[(j_1-1) % 5][i_1] + \
                                        mtx_abt_j[(j_2-1) % 5][i_2]
                                elif j_1 == j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_1] + \
                                        mtx_abt_j[j_1][i_1]
                                flag = False
                                break
    return enc_text
# вывод результатов работы программы
print(f'''
Шифр Плейфера:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_short(), key)}
```

```
Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab03_9_playfair.py
Введите ключ: ключик

Шифр Плейфера:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
зцднэйрурскююдакпуюкаэунмбоукгкпгааёсичы

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
аруртюндстодпакмдояогготдтаркпдрилартаптодупкйтёбкнуылчмудчормрупрщдёдббёурпб
щтаашчмбиычвоупкгиораворадюмувуунщдюкёддбмюёещлюкбиьйавпкяхлщкмкфтзвфктнньфро
жбклвфчмсиюаодафтщактувуаеафаьгбдуапкбаеерщлюкусжфбвёвъвдюпаякёутакмрпвмбдпкр
афасикётёпёёулдвнунассикёдояогготдтаркпзцактёнуатгбёучтрпббёргбпатакмкюдмгблю
чидктауъаёсукмоуилудчоумдояогготдтаркпзвупочпкылпртюндузптктпдрилтудочтщдаёрпа
ёёэгбдужрсиупоашгкгаюючкгдугравгшупсьпаеашгщдюкеггущдпчратсаеунзгбккикмаоилёу
жрудпчюмкпорпууважбасбхувбисадёюмрутпврдёиивтседткигосатдувикёетакмкюдмгбтчздп
квёруупафюккгтуарпчраунюмндннрарёовтубаттилаюпрютгёудстафмггbудббёутулттюйbуп
шгдщдоапяоггчтрспабкдёииюквамрилскдурспабкраребккиюдбаупехднудчоодрсдтвуёуёет
уткюкеггущдтчздпкрадтнуёутупкылпсвёёаовмвбкbгйткпгбргпатаёbжтсптуовйурдтёсичо
икодпарспабкэёёавёиклкунючгьрмруодккёаарпротупднгуупсиафемёауёакпупсbащктзакд
лпзюддмастровгабиюдаётубгдщяпупдтптпаёвоашгкстчздпкратрспвакбтджрутикодьспткб
емкдгбзёаёьюдннуупёигкгурднуёуасартртюасчасичосакбтчудпяжрсидщдояпюкумяёудчоп
двнаекмсапрспроочвёрумкиоуёавтедусикёпапкяхкмтуеруъзёвфнувёдояоггрнёеафгввнрс
пабкрасичы

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходя
щийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпублик
аций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.но
```

```
можноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднука
ртину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотыс
ячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблят
ьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеи
зменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиили
без.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностто
лькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,
таккакэтопустоеместо.однаконекоторыефирмыибиржвидятсправедливымставитьстоимо
стьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосп
риятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольши
нствунужнаценазатысячузнаковбезпробелов.
```

Карточка: