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 В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ

4. Шифр Тритемия

Шифр Тритемия предполагал использование алфавитной таблицы. Он использовал эту таблицу для многоалфавитного зашифрования самым простым из возможных способов: первая буква текста шифруется первым алфавитом, вторая буква — вторым и т. д. В этой таблице не было отдельного алфавита открытого текста, для этой цели служил алфавит первой строки. Таким образом, открытый текст, начинающийся со слов HUNC CAVETO VIRUM ..., приобретал вид HXPF GFBMCZ FUEIB ... .

Преимущество этого метода шифрования по сравнению с методом Альберти состоит в том, что с каждой буквой задействуется новый алфавит. Альберти менял алфавиты лишь по-сле трех или четырех слов. Поэтому его шифртекст состоял из отрезков, каждый из которых обладал закономерностями открытого текста, которые помогали вскрыть криптограмму. Побуквенное зашифрование не дает такого преимущества. Шифр Тритемия является также первым нетривиальным примером периодического шифра. Так называется многоалфавитный шифр, правило зашифрования которого состоит в использовании периодически повторяющейся последовательности простых замен.

Код программы:

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

# функция шифровки
def trithemius_decode(input):
    decode: str = ""
    k = 0
    for position, symbol in enumerate(input):
        index = (alphabet.find(symbol) + k) % len(alphabet)
        decode += alphabet[index]
        k -= 1
    return decode

# функция дешифровки
def trithemius_encode(input):
    encode = ""
    k = 0
    for position, symbol in enumerate(input):
        index = (alphabet.find(symbol) + k) % len(alphabet)
        encode += alphabet[index]
        k += 1
    return encode
```

```
# вывод результатов работы программы
print(f'''
Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_long())))}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-
1/lab02_4_trithemius.py

Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
всжпгмхщчщтцфоичюгэыхпгыюьммтбимбелвхып

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
впфтфнтлшъькюицьпгмдтлизаеыижкярцкюфсзучщзышвщыъоыхяоюэяиьгкмглпмотйогпбухччн
аърмзшъхютяхжйжряёолаярпдемтшлшцоэфшцыпусъьвхладвюжыкгаомюымъофьчъчлгцээюмзгз
цвагщрдёпхйвувнтсшлтъьпьссщюсмфущзчёдюяяюрузлйфуйъёзпиапнхьпкзябвшюджжэвыялйн
впмхыпухфчршъхоучюцхвжмбешлхмыфсриндспузчмушчръсэсрябьёеегфбиэьпъндйплпнйизух
игмцэуюеэуезяеллсёовиртовхяцеюыътчныщэсндбеядвугзйлейгпнпуотжешъиьуэяцщпааэуь
ршчэлвкофрнтьувыезсужбкряпафсрдёгехйфэямыпжкиедзхчошучлььссьфъучяэмяеулсёйлёо
тёуоммсхышшэьсоъогнвдщщвёыщждммррояйгрнокшьмушхеобгряьеънаёщияекжкиедгхнтицфй
тыяаэъяэыкщжищжкёйюкгззнжрсузхпщйьюмтбвзьфюгеязшгмамоинйежвцйсгхчыэфьючэпдбел
юичкхмцньхзсртсейсжфстцфнтцвзьёкщзёзжъяжкясбемкъмжёлъчкерщауъвдтгеюгьижиьорн
имкжёйчщрътчныщэсщючбвдзйжецёкнюжмьтгликтжнцьчыпобтаувшгсдзйимонюмсудсрчэсщат
эляйюаятчяпэцвшбсджёлдлягвцмхщъифхлчкбюаедёгъгьйлинузмгнбмссйрхъчъёрбцяьлъуье
ьящбэщннмоявёёомжбущыймфяйавяъявёзтяшлхмыфсриндспузшчщшлршнышшбеелюиччиимзввм
иёдгктбуефочръаэыысйшёзшьёжиюкямрпсквцнещяуыщошнашгцдпьеиоблъншойзтоэмцйршъйо
тррцьюуавдгейигщвкокжйтппзешйлъыбхщыоэымйыбёзшкёмбиьугмаивхяцеюырчкцыппфшбгвх
ъвёълсё
```

```
Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходя
щийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпублик
аций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.но
можноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднука
ртину.текстнатысячусимволовэтосколькопримернословстатистикапоказывает,чтотыс
ячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблят
ьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеи
зменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиили
без.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименносто
лькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,
таккакэтопустоеместо.однаконекоторыефирмыибиржевидятсправедливымставитьстоимо
стьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосп
риятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольши
нствунужнаценазатысячузнаковбезпробелов.
```

Исполняемый файл: https://github.com/lSenichl/mospolytech-education-crypt-dev-2021-1/blob/main/lab02_4_trithemius.py

5. Шифр Белазо

В 1553 Джованни Баттиста Белазо предложил использовать для многоалфавитного шифра буквенный, легко запо-минаемый ключ, который он назвал паролем. Паролем могло служить слово или фраза. Пароль периодически записывался над открытым текстом. Буква пароля, расположенная над буквой текста, указывала на алфавит таблицы, который исполь-зовался для зашифрования этой буквы. Например, это мог быть алфавит из таблицы Тритемия, первой буквой которого являлась буква пароля. Однако Белазо, как и Тритемий, использовал в качестве алфавитов шифра обычные алфавиты.

Код программы:

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

key = str(input('Введите ключ: '))

# функция шифровки
def bellaso_decode(input, key):
    decrypted = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
            offset += -1
        elif (alphabet.find(input[ix])) > (len(alphabet) - (alphabet.find(key
[((ix + offset) % len(key))]))) - 1):
            output = alphabet[(alphabet.find(
                input[ix]) - (alphabet.find(key[((ix + offset) % len(key))]))
) % 33]
        else:
            output = alphabet[alphabet.find(
                input[ix]) - (alphabet.find(key[((ix + offset) % len(key))]))
]
        decrypted += output
    return decrypted

# функция дешифровки
def bellaso_encode(input, key):
    encoded = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
            offset += -1
        elif (alphabet.find(input[ix])) > (len(alphabet) - (alphabet.find(key
[((ix + offset) % len(key))]))) - 1):
```

```
            output = alphabet[(alphabet.find(
                input[ix]) + (alphabet.find(key[((ix + offset) % len(key))]))
) % 33]
        else:
            output = alphabet[alphabet.find(
                input[ix]) + (alphabet.find(key[((ix + offset) % len(key))]))
]
        encoded += output
    return encoded


# вывод результатов работы программы
print(f'''
Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{bellaso_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_short(), key), key))}

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

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab02_5_bellaso.py
Введите ключ: ключик

Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
мьгдзтъюнзсцунщачэцфатцпспсйапцъаьукэги

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
мържщучроиыкэзжеиэёээоььушаёфщмюхвёэщпмиыкэъхеччкчгеехухрьуьэунйчъэфкчфжшънём
ащпэрсфочэвиыэъхьуэщнюзчммфлйнышррафучлбчрушлуафуочэенлщчъпсаущтёщчквжёцшёбнк
йцуцюнсфэгищыкхъкйнхьюгзнохъятккпюяёфпппакюуцфрзоакмёчяпмфмшдвшъмысшъъвяинщчм
щчхэгиеччщтлёслпульлщэгиеиэёээоььушаёфщмьгвччпщвёккшъжишщцзёёккэзмысшучжыккхч
```

```
ьоиущплкукыюжеьэвцрьуьэщюйдьйгсисчмъйёкзэъпвчцжцмжщучроечьцъа?иыкэфпйсхкымвит
ёнюьытъюхйчэёээоимхчьоипэнпьййьюмжзэжпгизэучжыкпьюжифщмэоьмшпхаьфувфлтывхщмяш
эпэйарцщянёыыпмйцыжъьгыфщнлкаръээмхркчфжыщюнфкаакьюэдсыпгжеищофлафуонюисчмъйч
ръэюмвчцуггиымщэйёкшпфёдншшъаёрыкэрчнээгищущъфочтэпьпвчфорэйнцжщмиыуъьжезэщэх
аыкэзртъйвфпжщцлрйчхуучжшнтэгикапэыоёйпцъаккпцфхаккпюмшгпчюгвъэкыоахпыщмеиьэъ
жгсомрпйсьушаёфщмфкьцшщэрёфжхъочрчёьюямпцкгдъцщнюикщлъведчъьмиыыкщпйкщчюхвъву
ююйеъыъяьфётличрвуцженцимэйръэююукхирёшюьюмьхпьюмйахщплчущшриёыщыжглсычжжшсы
сфаамйээнзимппйакёчэрчкуэзпйчучъпйеткющизвюэждкщцъаишыщмггичууннйъвуюющшщьчгыщ
упнююцёчийьхпшюмдукврпйкпшщмъчмщэнзсйэфэйахьъбгиьуюгиетъюхаыкэзпгсэшжзйнхьюяь
рпофлёлщъьмжььхлёжышуцрёцплявьыэвцлёйщцзцацьэнсеьсшлфьцктлртъйвяёеихщняьръыъя
ьфщмюхв

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

Исполняемый файл: https://github.com/lSenichl/mospolytech-education-crypt-dev-2021-1/blob/main/lab01_2_caesar.py

6. Шифр Вижинера

Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова.

Код программы:

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

key = str(input('Введите ключ: '))

# функция шифровки
def vigenere_encode(input, key):
    enc_string = ''
    string_length = len(input)

    expanded_key = key
    expanded_key_length = len(expanded_key)

    while expanded_key_length < string_length:
        expanded_key = expanded_key + key
        expanded_key_length = len(expanded_key)

    key_position = 0

    for letter in input:
        if letter in alphabet:
            position = alphabet.find(letter)

            key_character = expanded_key[key_position]
            key_character_position = alphabet.find(key_character)
            key_position = key_position + 1

            new_position = position + key_character_position
            if new_position >= 33:
                new_position = new_position - 33
            new_character = alphabet[new_position]
            enc_string = enc_string + new_character
        else:
            enc_string = enc_string + letter
    return(enc_string)
```

```
# функция дешифровки
def vigenere_decode(input, key):
    dec_string = ''
    string_length = len(input)

    expanded_key = key
    expanded_key_length = len(expanded_key)

    while expanded_key_length < string_length:
        expanded_key = expanded_key + key
        expanded_key_length = len(expanded_key)

    key_position = 0

    for letter in input:
        if letter in alphabet:
            position = alphabet.find(letter)

            key_character = expanded_key[key_position]
            key_character_position = alphabet.find(key_character)
            key_position = key_position + 1

            new_position = position - key_character_position
            if new_position >= 33:
                new_position = new_position + 33
            new_character = alphabet[new_position]
            dec_string = dec_string + new_character
        else:
            dec_string = dec_string + letter
    return(dec_string)
# вывод результатов работы программы
print(f'''
Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{vigenere_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(vigenere_decode(vigenere_encode(
    input_for_cipher_short(), key), key))}

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

Расшифрованный текст:
{output_from_decrypted(vigenere_decode(vigenere_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab02_6_vigenere.py
Введите ключ: ключик

Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
мьгдзтъюнзсцунщачэцфатцпспсйапцъаьукэги

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
мържщучроиыкэзжеиэёээоььушаёфщмюхвёэщпмиыкэъхеччкчгеехухрьуьэунйчъэфкчфжшънём
ащпэрсфочэвиыэъхьуэщнюзчммфлйнышррафучлбчрушлуафуочэенлщчъпсаущтёщчквжёцшёбнк
йцуцюнсфэгищыкхъкйнхьюгзнохъятккпюяёфпппакюуцфрзоакмёчяпмфмшдвшъмысшъъвяинщчм
щчхэгиеччщтлёслпульлщэгиеиэёээоььушаёфщмьгвччпщвёккшъжишщцзёёккэзмысшучжыккхч
ьоиущплкукыюжеьэвцрьуьэщюйдьйгсисчмъйёкзэъпвчцжцмжщучроечьцъа?иыкэфпйсхкымвит
ёнюьытъюхйчэёээоимхчьоипэнпьййьюмжзэжпгизэучжыкпьюжифщмэоьмшпхаьфувфлтывхщмяш
эпэйарцщянёыыпмйцыжъьгыфщнлкаръээмхркчфжыщюнфкаакьюэдсыпгжеищофлафуонюисчмъйч
ръэюмвчцуггиымщэйёкшпфёднншъаёрыкэрчнээгищущъфочтэпьпвчфорэйнцжщмиыуъьжезэщэх
аыкэзртъйвфпжщцлрйчхуучжшнтэгикапэыоёйпцъаккпцфхаккпюмшгпчюгвъэкыоахпыщмеиьэъ
жгсомрпйсьушаёфщмфкьцшщэрёфжхъочрчёьюямпцкгдъцщнюикщлъведчъьмиыыкщпйкщчюхвъву
ююйеъыъяьфётличрвуцженцимэйръэюювукхирёшюьюмьхпьюмйахщплчущшриёыщыжглсычжжшсы
сфаамйээнзимппйакёчэрчкуэзпйчучъпйеткющизвюэждкщцъаишыщмггичууннйъвуююцшщьчгыц
упнююцёчийьхпшюмдукврпйкпшщмъчмщэнзсйэфэйахьъбгиьуюгиетъюхаыкэзпгсэшжзйнхьюяь
рпофлёлщъьмжььхлёжышуцрёцплявьыэвцлёйщцзцацьэнсеьсшлфьцктлртъйвяёеихщняьръыъя
ьфщмюхв

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

Исполняемый файл: https://github.com/lSenichl/mospolytech-education-crypt-dev-2021-1/blob/main/lab01_3_square.py