Аннотация

Среда программирования: Visual Studio Code

Язык программирования: Python 3

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

Пословица-тест: Красивыми словами пастернак не помаслишь

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#

Блок F: ПОТОЧНЫЕ ШИФРЫ

- **A5/1**

Код программы:

```
# -*- coding:utf-8 -*-
from demo import alphabet, input_for_cipher_short, output_from_decrypted
import re
import copy
reg_x_length = 19
reg_y_length = 22
reg_z_length = 23

key_one = ""
reg_x = []
reg_y = []
reg_z = []


def loading_registers(key):
    i = 0
    while(i < reg_x_length):
        reg_x.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = reg_x_length
    while(j < reg_y_length):
        reg_y.insert(j, int(key[p]))
        p = p + 1
        j = j + 1
    k = reg_y_length + reg_x_length
    r = 0
    while(r < reg_z_length):
        reg_z.insert(r, int(key[k]))
        k = k + 1
        r = r + 1


def set_key(key):
    if(len(key) == 64 and re.match("^([01])+", key)):
        key_one = key
        loading_registers(key)
        return True
    return False


def get_key():
    return key_one
```

```
def to_binary(plain):
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 12):
            binary = "0" + binary
            s = s + binary
            j = j + 1
    binary_values = []
    k = 0
    while(k < len(s)):
        binary_values.insert(k, int(s[k]))
        k = k + 1
    return binary_values


def get_majority(x, y, z):
    if(x + y + z > 1):
        return 1
    else:
        return 0


def get_keystream(length):
    reg_x_temp = copy.deepcopy(reg_x)
    reg_y_temp = copy.deepcopy(reg_y)
    reg_z_temp = copy.deepcopy(reg_z)
    keystream = []
    i = 0
    while i < length:
        majority = get_majority(reg_x_temp[8], reg_y_temp[10], reg_z_temp[10])
        if reg_x_temp[8] == majority:
            new = reg_x_temp[13] ^ reg_x_temp[16] ^ reg_x_temp[17] ^
reg_x_temp[18]
            reg_x_temp_two = copy.deepcopy(reg_x_temp)
            j = 1
            while(j < len(reg_x_temp)):
                reg_x_temp[j] = reg_x_temp_two[j-1]
                j = j + 1
            reg_x_temp[0] = new

        if reg_y_temp[10] == majority:
            new_one = reg_y_temp[20] ^ reg_y_temp[21]
            reg_y_temp_two = copy.deepcopy(reg_y_temp)
            k = 1
            while(k < len(reg_y_temp)):
                reg_y_temp[k] = reg_y_temp_two[k-1]
                k = k + 1
```

```
            reg_y_temp[0] = new_one

        if reg_z_temp[10] == majority:
            new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ reg_z_temp[21] ^
reg_z_temp[22]
            reg_z_temp_two = copy.deepcopy(reg_z_temp)
            m = 1
            while(m < len(reg_z_temp)):
                reg_z_temp[m] = reg_z_temp_two[m-1]
                m = m + 1
            reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^ reg_z_temp[22])
        i = i + 1
    return keystream


def convert_binary_to_str(binary):
    s = ""
    length = len(binary) - 12
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+12], 2))
        i = i + 12
    return str(s)


def encode(plain):
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s


def decode(cipher):
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i, int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))
```

```
def user_input_key():
    tha_key = str(input('Введите 64-bit ключ: '))
    if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
        return tha_key
    else:
        while(len(tha_key) != 64 and not re.match("^([01])+", tha_key)):
            if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                return tha_key
            tha_key = str(input('Введите 64-bit ключ: '))
    return tha_key

# 0101001000011010110001110001100100101001000000110111111010110111


key = str(user_input_key())
set_key(key)


print(f'''
Зашифрованный текст:
{encode(input_for_cipher_short())}
Расшифрованный текст:
{output_from_decrypted(decode(encode(
    input_for_cipher_short())))}
''')
```

Тестирование:


Расшифрованный текст:
Вотпримерстатьинатысячусимволов.Этодостаточноманенькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидляне6ольшихинформационныхпубликаций.Втакомтексте6ываетболеедвухилитрёхабзацевиобычноодинподзаголовок.Номожноибезнего.Натысячусимволоврекомендованоиспользоватьодинилидваключачиоднукартину.Текстнатысячусимволовэтоскольконримернослов?Статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.Но,еслиз
лоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословвнеизменноевозрастает.Вкопирайтерскойдеятельностипринятос-читатьтыся-ичпробеламиилибез.Учетпробеловувеличиваетобъемтекстапримернонастолилдвест
исимволовименностолькоразмыразделяемсловавсвободнымпространством.Считатьпробелызаказчикинелобят,таккакэтопустоеместо.Однаконекоторыефирмыбираживидятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэл
ементомкачественноговосприятия.Согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.Нобольшинствунужнаценазатысячузнаковбезпробелов.