# 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#

**ГЕНЕРАЦИЯ ЦИФРОВОЙ ПОДПИСИ**



## 

* **ГОСТ Р 34.10-94**

**Код программы:**

alphavit = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,

            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,

            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,

            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,

            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,

            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,

            'ю': 31, 'я': 32

            }

def ciphergostd(clearText):

    array = []

    flag = False

    for s in range(50, 1000):

        for i in range(2, s):

            if s % i == 0:

                flag = True

                break

        if flag == False:

            array.append(s)

        flag = False

    p = 31

    print("p = ", p)

    q = 5

    print("q = ", q)

    a = 2

    print("a =", a)

    array2 = []

    flag2 = False

    for s in range(2, q):

        for i in range(2, s):

            if s % i == 0:

                flag2 = True

                break

        if flag2 == False:

            array2.append(s)

        flag2 = False

    x = 3

    print("x = ", x)

    y = a\*\*x % p

    k = 4

    print("k = ", k)

    r = (a\*\*k % p) % q

    msg = clearText

    msg\_list = list(msg)

    alpha\_code\_msg = list()

    for i in range(len(msg\_list)):

        alpha\_code\_msg.append(int(alphavit.get(msg\_list[i])))

    print("Длина исходного сообщения {} символов".format(len(alpha\_code\_msg)))

    hash\_code\_msg = hash\_value(p, alpha\_code\_msg)

    print("Хэш сообщения:= {}".format(hash\_code\_msg))

    s = (x\*r+k\*hash\_code\_msg) % q

    print("Цифровая подпись = ", r % (2\*\*256), ",", s % (2\*\*256))

    v = (hash\_code\_msg\*\*(q-2)) % q

    z1 = s\*v % q

    z2 = ((q-r)\*v) % q

    u = (((a\*\*z1)\*(y\*\*z2)) % p) % q

    print(r, " = ", u)

    if u == r:

        print("r = u, следовательно:")

        print("Подпись верна\n")

    else:

        print("Подпись неверна")

def hash\_value(n, alpha\_code):

    i = 0

    hash = 1

    while i < len(alpha\_code):

        hash = (((hash-1) + int(alpha\_code[i]))\*\*2) % n

        i += 1

    return hash

def main():

    print('ГОСТ Р 34.10-94:')

    message = input("Введите сообщение: ")

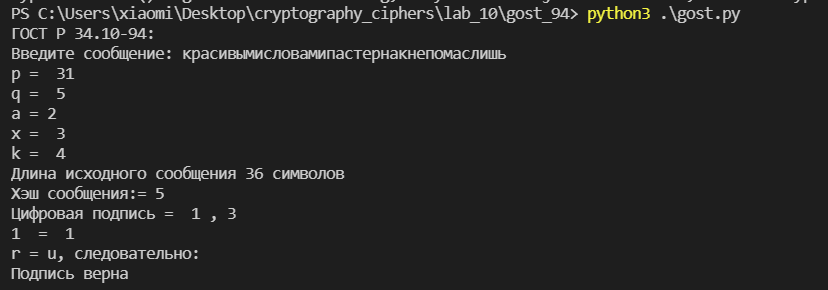
    ciphergostd(message)

if \_\_name\_\_ == "\_\_main\_\_":

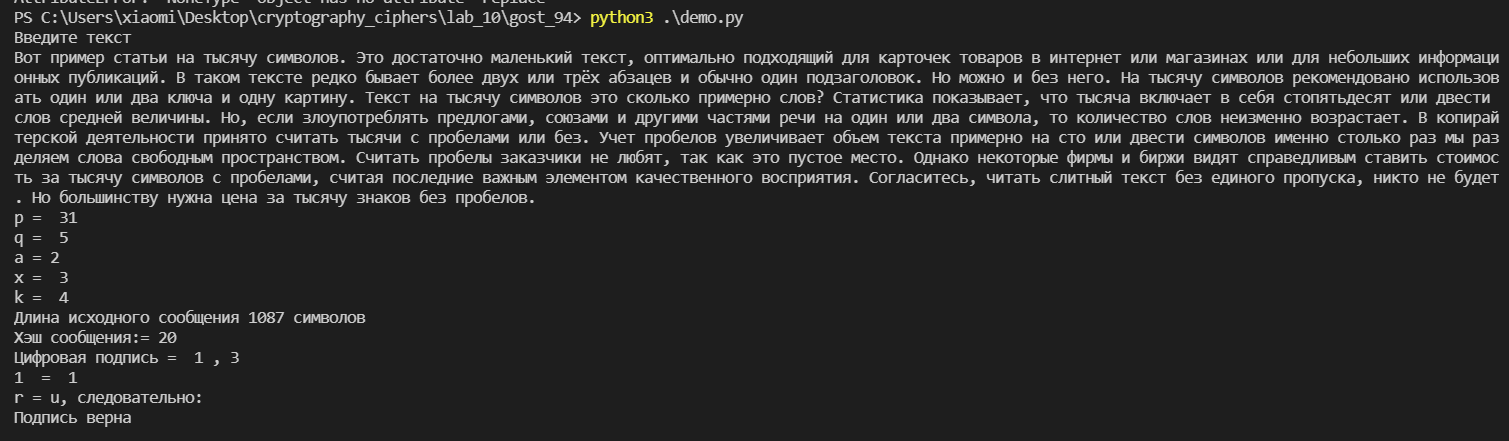
    main()

**Тестирование:**

**Фраза по варианту**



**Текст на 1000 символов**

****

* **ГОСТ Р 34.10-2012**

**Программа**

import random

import collections

alphabet\_lower = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,

            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,

            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,

            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,

            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,

            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,

            'ю': 31, 'я': 32

            }

class Point:

    def \_\_init\_\_(self, x\_init, y\_init):

        self.x = x\_init

        self.y = y\_init

    def shift(self, x, y):

        self.x += x

        self.y += y

    def \_\_repr\_\_(self):

        return "".join(["( x=", str(self.x), ", y=", str(self.y), ")"])

x\_1 = 0

y\_1 = 0

EllipticCurve = collections.namedtuple(

    'EllipticCurve', 'name p q\_mod a b q g n h')

curve = EllipticCurve(

    'secp256k1',

    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,

    q\_mod=0xfffffffffefffffffffcfffffffffffcfffffffffffffffffffffffefffffc2f,

    a=7,

    b=11,

    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,

       0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),

    q=(0xA0434D9E47F3C86235477C7B1AE6AE5D3442D49B1943C2B752A68E2A47E247C7,

       0x893ABA425419BC27A3B6C7E693A24C696F794C2ED877A1593CBEE53B037368D7),

    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,

    h=1,

)

def ciphergostd(clearText):

    msg = clearText

    msg\_list = list(msg)

    alpha\_code\_msg = list()

    for i in range(len(msg\_list)):

        alpha\_code\_msg.append(int(alphabet\_lower.get(msg\_list[i])))

    print("Длина исходного сообщения {} символов".format(len(alpha\_code\_msg)))

    print("Q mod", int(curve.q\_mod))

    print("P mod", int(curve.p))

    hash\_code\_msg = hash\_value(curve.p, alpha\_code\_msg)

    print("Хэш сообщения:={}".format(hash\_code\_msg))

    e = hash\_code\_msg % curve.q\_mod

    print("E={}".format(e))

    k = random.randint(1, curve.q\_mod)

    print("K={}".format(k))

    d = 10

    print("D={}".format(d))

    x, y = scalar\_mult(k, curve.g)

    point\_c = Point(x, y)

    print("Point\_C={}".format(point\_c))

    r = point\_c.x % curve.q\_mod

    print("R={}".format(r))

    s = (r\*curve.p + k\*e) % curve.q\_mod

    print("S={}".format(s))

    v = inverse\_mod(e, curve.p)

    print("V={}".format(v))

    z1 = (s\*v) % curve.q\_mod

    z2 = ((curve.p-r)\*v) % curve.q\_mod

    x\_1, y\_1 = scalar\_mult(d, curve.g)

    print("Point\_Q=( x={}, y={} )".format(x\_1, y\_1))

    point\_c\_new = Point(x, y)

    x, y = point\_add(scalar\_mult(z1, curve.g),

                     scalar\_mult(z2, curve.q))

    r\_1 = point\_c\_new.x % curve.q\_mod

    print("R\_new={}".format(r\_1))

    if r == r\_1:

        print("Подпись прошла проверку!\n")

    else:

        print("Ошибка проверки!")

def hash\_value(mod, alpha\_code\_msg):

    i = 0

    hashing\_value = 1

    while i < len(alpha\_code\_msg):

        hashing\_value = (

            ((hashing\_value-1) + int(alpha\_code\_msg[i]))\*\*2) % curve.p

        i += 1

    return hashing\_value

def is\_on\_curve(point):

    if point is None:

        return True

    x, y = point

    return (y \* y - x \* x \* x - curve.a \* x - curve.b) % curve.p == 0

def point\_neg(point):

    if point is None:

        return None

    x, y = point

    result = (x, -y % curve.p)

    return result

def inverse\_mod(k, p):

    if k == 0:

        raise ZeroDivisionError('деление на 0')

    if k < 0:

        return p - inverse\_mod(-k, p)

    s, old\_s = 0, 1

    t, old\_t = 1, 0

    r, old\_r = p, k

    while r != 0:

        quotient = old\_r // r

        old\_r, r = r, old\_r - quotient \* r

        old\_s, s = s, old\_s - quotient \* s

        old\_t, t = t, old\_t - quotient \* t

    gcd, x, y = old\_r, old\_s, old\_t

    assert gcd == 1

    assert (k \* x) % p == 1

    return x % p

def point\_add(point1, point2):

    if point1 is None:

        return point2

    if point2 is None:

        return point1

    x1, y1 = point1

    x2, y2 = point2

    if x1 == x2 and y1 != y2:

        return None

    if x1 == x2:

        m = (3 \* x1 \* x1 + curve.a) \* inverse\_mod(2 \* y1, curve.p)

    else:

        m = (y1 - y2) \* inverse\_mod(x1 - x2, curve.p)

    x3 = m \* m - x1 - x2

    y3 = y1 + m \* (x3 - x1)

    result = (x3 % curve.p,

              -y3 % curve.p)

    return result

def scalar\_mult(k, point):

    if k % curve.n == 0 or point is None:

        return None

    if k < 0:

        return scalar\_mult(-k, point\_neg(point))

    result = None

    addend = point

    while k:

        if k & 1:

            result = point\_add(result, addend)

        addend = point\_add(addend, addend)

        k >>= 1

    return result

def main():

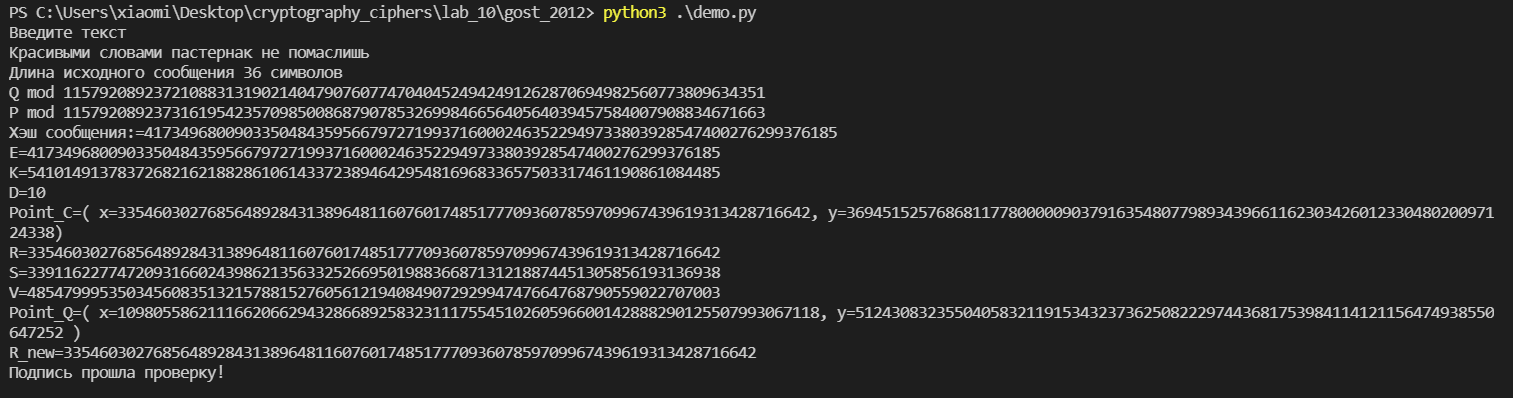
    print('ГОСТ Р 34.10-2012:')

if \_\_name\_\_ == "\_\_main\_\_":

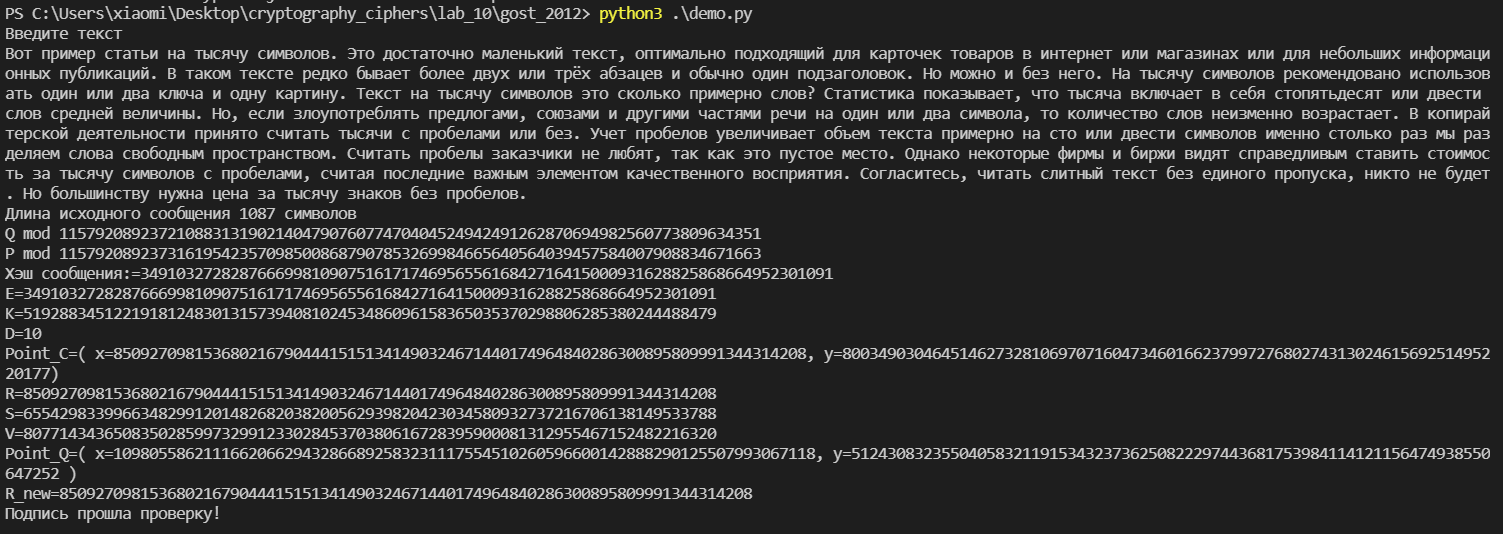
    main()

**Тестирование**

**Фраза по варианту**

****

**Проверка текста на 1000 символов**

****