

Криптография с Python

Добро пожаловать!

Справочник?

Данный файл является скорее справочником, чем полноценной книгой, т.к. в привычном плане теории здесь не будет, будет лишь практика. В PDF файле будет разобрано 34 шифра из криптографии, 4 программы для помощи в криптоанализе, 3 программы связанные со стеганографией и 2 дополнительные программы.

Для кого создан этот справочник?

Этот справочник скорее создан для тех людей, кто хочет изучать Криптографию, но знают при этом лишь программирование, либо же наоборот, хотят изучать Программирование, но знают при этом лишь криптографию.

Почему именно Python?

Python идеальный язык для работы со строками, а также различного рода вычислений. Из-за своей лёгкости и понятного синтаксиса у нас будут получаться достаточно красивые и понятные программы.

Версия Python?

Version – 3.6

Какие шифры находятся в этом справочнике?

001 шифр Цезаря	018 шифр замены слогов
002 шифр ROT13	019 шифр Порты
003 шифр Тритемиуса	020 роторное шифрование
004 шифр замены	021 криптографические хеш-функции
005 шифр Атбаш	022 степени шифра Виженера
006 шифр Бэкона	023 шифр Штакетник
007 шифр пар	024 шифр решётки
008 тарабарская грамота	025 шифр с омофонами
009 шифр Полибия	026 шифровальная машина <Турех>
010 шифр Виженера	027 шифр двойной цифри
011 шифр Вернама	028 шифр Плейфера
012 книжный шифр	029 шифр ADFGVX
013 XOR шифрование	030 AES шифрование
014 шифр с ложными символами	031 Великий Шифр
015 шифр Гронсфельда	032 RSA шифрование
016 псевдосимвольный шифр	033 аффинный шифр
017 кодирование	034 шифр Хилла

Какие программы для криптоанализа находятся в этом справочнике?

- 101 BruteForce шифра Цезаря
- 102 Частотный криптоанализ
- 103 BruteForce криптографической хеш-функции по словарю
- 104 BruteForce запароленного zip-архива по словарю

Какие программы со стеганографией находятся в этом справочнике?

- 201 Сообщение в файле на уровне байтов
- 202 Чтение байтов в файле
- 203 Занесение архива в файл

Какие дополнительные программы находятся в этом справочнике?

- 301 Шифровальщик файлов
- 302 Скрипт для создания .onion сайта в сети Tor

Обозначения синтаксиса?

Black – переменные/константы

Gray - комментарии

Purple – строки

Red – операторы / регулярные выражения

Blue – функции / методы / исключения

Green – объявление функции / булевы значения / регулярные выражения

Orange - аргументы

Частые обозначения в справочнике?

// file.py – python скрипт

// terminal – ввод и вывод из под терминала

START_[...]_END – Начало и конец программы/файла/терминала

encryptDecrypt – главная функция с аргументами переключателя и сообщением

startMessage – сообщение для шифрования/расшифрования

cryptMode – переключатель режимов шифрования/расшифрования

E – encrypt (Зашифровать)

D – decrypt (Расшифровать)

Задания в справочнике?

В конце справочника будут указаны различные задачки связанные с криптографией. Надеюсь эти задачки и загадки будут интересны читателю.

**Автор справочника
Коваленко Г. А. #571**

Криптография

001. Шифр Цезаря.

(Шифр Цезаря – является классическим методом шифрования и одним из самых знаменитых. Принцип шифрования заключён в ключе-позиции по алфавиту.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
try: rotKey = int(input("Write the key: "))
except ValueError: print("Only numbers!"); raise SystemExit
def encryptDecrypt(mode, message, key, final = ""):
    for symbol in message:
        if mode == 'E':
            final += chr((ord(symbol) + key - 13)%26 + ord('A'))
        else:
            final += chr((ord(symbol) - key - 13)%26 + ord('A'))
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage, rotKey))
_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt[D]ecrypt: e

Write the message: helloworld

Write the key: 3

Final message: KHOORZRUOG

\$ python file.py

[E]ncrypt[D]ecrypt: d

Write the message: KHOORZRUOG

Write the key: 3

Final message: HELLOWORLD

_End

```
# Переключатель режимов шифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

# Если cryptMode не будет равняться 'E' ИЛИ 'D', тогда вывести ошибку и
закрыть программу.
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение которое мы хотим зашифровать ИЛИ расшифровать.
startMessage = input("Write the message: ").upper()

# Переменная-ключ, если ключ не является числом – выводится ошибка.
try: rotKey = int(input("Write the key: "))
except ValueError: print("Only numbers!"); raise SystemExit

# Основная функция принимающая аргументы: переключатель, сообщение, ключ.
def encryptDecrypt(mode, message, key, final = ""):

    # Посимвольный перебор подаваемого функции сообщения.
    for symbol in message:

        # Шифрование сообщения при помощи таблицы ASCII.
        if mode == 'E':
            final += chr((ord(symbol) + key - 13)%26 + ord('A'))

        # Если переключатель равен символу 'D', тогда делать тоже самое что и с 'E' лишь
        за исключением вычитания переменной key, а не её сложения!
        else:
            final += chr((ord(symbol) - key - 13)%26 + ord('A'))

    # Вернуть получившееся сообщение после шифровки/расшифровки.
    return final

# Вывод получившегося сообщения.
print("Final message:", encryptDecrypt(cryptMode, startMessage, rotKey))
```

002. Шифр ROT13.

(Шифр ROT13 – часть шифра Цезаря с позицией 13. Особенность ROT13 заключена в принципе инволюции, при которой не нужен переключатель режимов шифрования/расшифрования.)

// file.py

Start_

```
message = list(input("Write the message: ").upper())
```

```
for symbol in range(len(message)):
```

```
    message[symbol] = chr(ord(message[symbol])%26+ord('A'))
```

```
print("Final message:", "".join(message))
```

_End

// terminal

Start_

```
$ python file.py
```

```
Write the message: helloworld
```

```
Final message: URYYBJBEYQ
```

```
$ python file.py
```

```
Write the message: URYYBJBEYQ
```

```
Final message: HELLOWORLD
```

_End

```
# Ввод сообщения для шифрования/расшифрования.
```

```
message = list(input("Write the message: ").upper())
```

```
# Перебор индексов каждого символа в списке message.
```

```
for symbol in range(len(message)):
```

```
# Шифрование сообщения при помощи таблицы ASCII.
```

```
message[symbol] = chr(ord(message[symbol])%26+ord('A'))
```

```
# Вывод получившегося сообщения
```

```
print("Final message:", "".join(message))
```

003. Шифр Тритемиуса.

(Шифр Тритемиуса – улучшенный шифр Цезаря в котором ключём является не число, а какая-либо функция. В отличии от шифра Цезаря не подвержен частотному криптоанализу, т.к. шифрует посимвольно, изменяя каждый раз значение в функции ключа.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = list(input("Write the message: ").upper())
for symbol in startMessage:
    if symbol not in [chr(x) for x in range(65,91)]:
        startMessage.remove(symbol)
funcKey = lambda x: x*2
def encryptDecrypt(mode, message, key, final = ""):
    for index, symbol in enumerate(message):
        if mode == 'E':
            temp = ord(symbol) + key(index) - 13
        else:
            temp = ord(symbol) - key(index) - 13
        final += chr(temp%26 + ord('A'))
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage, funcKey))
End_
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: HGPRWGAFBV

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: HGPRWGAFBV

Final message: HELLOWORLD

End_

```

# Переключатель режимов шифрования/расшифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = list(input("Write the message: ").upper())

# Посимвольный перебор сообщения.
for symbol in startMessage:

# Если символ не находится в диапазоне от А до Z, то удалить этот символ.
if symbol not in [chr(x) for x in range(65,91)]:
    startMessage.remove(symbol)

# Ключ-функция, принимающая числовое значение и умножающая его на 2.
funcKey = lambda x: x*2

# Основная функция для шифрования/расшифрования принимающая 4 аргумента
mode, message, key, final.
def encryptDecrypt(mode, message, key, final = ""):

# Посимвольный перебор сообщения. Index – индекс символа в сообщении, Symbol
– символ сообщения.
for index, symbol in enumerate(message):

# Если переключатель равен 'E', тогда присвоить к переменной temp
получившееся числовое значение.
if mode == 'E':
    temp = ord(symbol) + key(index) - 13

# Если переключатель равен символу 'D', тогда делать тоже самое что и с 'E' лишь
за исключением вычитания функции key, а не её сложения!
else:
    temp = ord(symbol) - key(index) - 13

# Присвоить к переменной final получившийся символ.
final += chr(temp%26 + ord('A'))

# Вернуть получившееся сообщение.
return final

```

Вывод получившегося сообщения

```
print("Final message:",encryptDecrypt(cryptMode, startMessage, funcKey))
```


004. Шифр замены.

(Шифр замены – один из самых распространённых методов шифрования. В отличие от шифра Цезаря не имеет конкретного ключа и алгоритма.)

// file.py

Start_

```
symbolsAlpha = [chr(x) for x in range(65,91)]
symbolsCrypt = ('!','@','#','$','%','^','&','*','(',')','-', '=',
'+','?',':',';','<','>','/','[',']','{','}','|',',','.','~')
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
keys = dict(zip(symbolsAlpha,symbolsCrypt))
def encryptDecrypt(mode, message, final = ""):
    if mode == 'E':
        for symbol in message:
            if symbol in keys: final += keys[symbol]
    else:
        for symbol in message:
            for key in keys:
                if symbol == keys[key]: final += key
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: *%==:}>=\$

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: *%==:}>=\$

Final message: HELLOWORLD

_End

```

# Объявление списка символов от 'A' до 'Z'.
symbolsAlpha = [chr(x) for x in range(65,91)]

# Объявление списка символов для шифровки.
symbolsCrypt = ('!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '=',
'+', '?', ':', ';', '<', '>', '/', '[', ']', '{', '}', '|', ',', '~')

# Переключатель для шифрования/расшифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

# Создание словаря подобного типа → символ_алфавита : символ_шифра.
keys = dict(zip(symbolsAlpha, symbolsCrypt))

# Объявление функции с 3 аргументами.
def encryptDecrypt(mode, message, final = ""):

# Если переключатель равен 'E', тогда сделать посимвольный перебор сообщения
и если символ будет находится в словаре keys (символ_алфавита), то к переменной
final прибавить значение ключа (символ_шифра).
if mode == 'E':
    for symbol in message:
        if symbol in keys: final += keys[symbol]

# Если переключатель равен 'D', тогда сделать посимвольный перебор сообщения
и перебор всех ключей в словаре keys. И если есть совпадение между символом в
сообщении и значением ключа (символ_шифра), то к переменной final прибавить
сам ключ (символ_алфавита).
else:
    for symbol in message:
        for key in keys:
            if symbol == keys[key]: final += key

# Вернуть получившееся сообщение.
return final

# Вывод получившегося сообщения.
print("Final message:", encryptDecrypt(cryptMode, startMessage))

```

005. Шифр Атбаш.

(Шифр Атбаш является одним из самых лёгких методов шифрования. Создаёт зеркальную версию алфавита, где начало - буква 'Z', а конец алфавита буква - 'A'. Не требует переключателя и ключа для шифрования/расшифрования.)

// file.py

Start_

```
message = input("Write the message: ").upper()
alphaDefault = [chr(x) for x in range(65,91)]
alphaReverse = list(alphaDefault); alphaReverse.reverse()
final = ""
for symbolMessage in message:
    for indexAlpha, symbolAlpha in enumerate(alphaDefault):
        if symbolMessage == symbolAlpha:
            final += alphaReverse[indexAlpha]
print("Final message:", final)
```

End_

// terminal

Start_

\$ python file.py

Write the message: helloworld

Final message: SVOOLDLIOW

\$ python file.py

Write the message: SVOOLDLIOW

Final message: HELLOWORLD

End_

Сообщение для шифрования/расшифрования.

```
message = input("Write the message: ").upper()
```

Создание списка символом от 'A' до 'Z'.

```
alphaDefault = [chr(x) for x in range(65,91)]
```

Создание списка символом от 'A' до 'Z' с реверсией.

```
alphaReverse = list(alphaDefault); alphaReverse.reverse()
```

Создание переменной-строки final.

```
final = ""
```

Посимвольный перебор в сообщении.

for symbolMessage **in** message:

Посимвольный перебор в списке алфавита.

for indexAlpha, symbolAlpha **in** enumerate(alphaDefault):

Если символ сообщения будет равен символу алфавита, тогда к переменной final добавить символ из списка реверсированного алфавита по индексу.

if symbolMessage == symbolAlpha:

final += alphaReverse[indexAlpha]

Вывод получившегося сообщения.

print("Final message:", final)

006. Шифр Бэкона.

(Шифр Бэкона является шифром замены, который можно представить в виде двоичного кода, где A = 0, B = 1. Также шифр Бэкона можно использовать в стеганографии, где допустим гласные буквы равны символу A, а согласные равны символу B.)

// file.py

Start_

```
from re import findall
```

```
keysBacon = {
```

```
    'A':"AAAAA",    'B':"AAAAB",    'C':"AAABA",
    'D':"AAABB",    'E':"AABAA",    'F':"AABAB",
    'G':"AABBA",    'H':"AABBB",    'I':"ABAAA",
    'J':"ABAAB",    'K':"ABABA",    'L':"ABABB",
    'M':"ABBAA",    'N':"ABBAB",    'O':"ABBBA",
    'P':"ABBBB",    'Q':"BAAAA",    'R':"BAAAB",
    'S':"BAABA",    'T':"BAABB",    'U':"BABAA",
    'V':"BABAB",    'W':"BABBA",    'X':"BABBB",
    'Y':"BBAAB",    'Z':"BBAAB",    ' ': "BBABA"
```

```
}
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
def regular(text):
```

```
    template = r"[A-Z]{5}"
```

```
    return findall(template, text)
```

```
def encryptDecrypt(mode, message, final = ""):
```

```
    if mode == 'E':
```

```
        for symbol in message:
```

```
            if symbol in keysBacon: final += keysBacon[symbol]
```

```
    else:
```

```
        for symbolsFive in regular(message):
```

```
            for key in keysBacon:
```

```
                if symbolsFive == keysBacon[key]: final += key
```

```
    return final
```

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

End

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message:

AABBBAABAABABBBABABBBABABBAABBBABAABABABBBAAABB

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message:

AABBBAABAABABBBABABBBABABBAABBBABAABABABBBAAABB

Final message: HELLOWORLD

_End

Импортирование функции findall из модуля 'регулярные выражения'.

from re import findall

Словарь вида → символ_алфавита : 5символов_шифра.

keysBacon = {

'A':"AAAAA", 'B':"AAAAB", 'C':"AAABA",
'D':"AAABB", 'E':"AABAA", 'F':"AABAB",
'G':"AABBA", 'H':"AABBB", 'I':"ABAAA",
'J':"ABAAB", 'K':"ABABA", 'L':"ABABB",
'M':"ABBA", 'N':"ABBAB", 'O':"ABBBA",
'P':"ABBBB", 'Q':"BAAAA", 'R':"BAAAB",
'S':"BAABA", 'T':"BAABB", 'U':"BABAA",
'V':"BABAB", 'W':"BABBA", 'X':"BABBB",
'Y':"BBA", 'Z':"BBAAB", ' ': "BBABA"

}

Переключатель шифрования/расшифрования.

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E','D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщение для шифрования/расшифрования.

startMessage = input("Write the message: ").upper()

Функция регулярного выражения, помогающая расшифровывать сообщение для зашифрованное сообщение по 5 символов.

```
def regular(text):  
    template = r"[A-Z]{5}"  
    return findall(template, text)
```

Функция с 3 аргументами.

```
def encryptDecrypt(mode, message, final = ""):
```

Если переключатель равен 'E', то сделать посимвольный перебор сообщения.

```
if mode == 'E':  
    for symbol in message:
```

Если символ сообщения (символ_алфавита) будет находится в ключах keysBacon, то к переменной final прибавить значение ключа (5символов_шифра).

```
if symbol in keysBacon: final += keysBacon[symbol]
```

Если переключатель равен 'D', то сделать перебор сообщения через регулярное выражение.

```
else:  
    for symbolsFive in regular(message):
```

Сделать посимвольный перебор ключей в словаре keysBacon.

```
for key in keysBacon:
```

И если пять символов равны значению ключа (5символов_шифра), тогда к переменной final добавить сам ключ (символ_алфавита).

```
if symbolsFive == keysBacon[key]: final += key
```

Вернуть получившееся сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

007. Шифр пар.

(Шифр пар является также шифром замены, но особенность данного метода шифрования в том, что есть привязка между двумя символами. То-есть, если в тексте попадётся символ А, тогда он заменится на В ($A \rightarrow B$), но это также действует и в обратную сторону ($B \rightarrow A$).)

// **file.py**

Start_[

```
keys = {
    'A':'B','C':'D','E':'F','G':'H','I':'J','K':'L',
    'M':'N','O':'P','Q':'R','S':'T','U':'V','W':'X',
    'Y':'Z'}
message = list(input("Write the message: ").upper())
for symbol in range(len(message)):
    for key in keys:
        if message[symbol] == key:
            message[symbol] = keys[key]
        elif message[symbol] == keys[key]:
            message[symbol] = key
        else: pass
print("Final message:", "".join(message))
_End
```

// **terminal**

Start_[

\$ python file.py

Write the message: helloworld

Final message: GFKKPXPQKC

\$ python file.py

Write the message: GFKKPXPQKC

Final message: HELLOWORLD

_End

Создание словаря с привязками символов.

```
keys = {
    'A':'B','C':'D','E':'F','G':'H','I':'J','K':'L',
    'M':'N','O':'P','Q':'R','S':'T','U':'V','W':'X',
    'Y':'Z'}
```


Сообщение для шифрования/расшифрования.

```
message = list(input("Write the message: ").upper())
```

Посимвольный перебор сообщения.

```
for symbol in range(len(message)):
```

Перебор всех ключей в словаре.

```
for key in keys:
```

Если символ сообщения будет равен ключу, тогда этот символ сообщения будет равен значению ключа.

```
if message[symbol] == key:
```

```
    message[symbol] = keys[key]
```

Если символ сообщения будет равен значению ключа, тогда этот символ сообщения будет равен ключу.

```
elif message[symbol] == keys[key]:
```

```
    message[symbol] = key
```

```
else: pass
```

Вывод получившегося сообщения.

```
print("Final message:", "".join(message))
```

008. Тарабарская грамота.

(Тарабарская грамота является своего рода шифром пар. Отличие заключено в замене только согласных символов.)

```
// file.py
Start_
keys = {
'B':'Z','C':'X','D':'W','F':'V','G':'T',
'H':'S','J':'R','K':'Q','L':'P','M':'N'}
message = list(input("Write the text: ").upper())
for symbol in range(len(message)):
    for key in keys:
        if message[symbol] == key:
            message[symbol] = keys[key]
        elif message[symbol] == keys[key]:
            message[symbol] = key
        else: pass
print("Final message:", "".join(message))
_End
```

// terminal

Start_

\$ python file.py

Write the text: helloworld

Final message: SEPPODOJPW

\$ python file.py

Write the text: SEPPODOJPW

Final message: HELLOWORLD

_End

Создание словаря с привязкой согласных символов.

```
keys = {
'B':'Z','C':'X','D':'W','F':'V','G':'T',
'H':'S','J':'R','K':'Q','L':'P','M':'N'}
```

Сообщение для шифрования/расшифрования.

```
message = list(input("Write the text: ").upper())
```

Посимвольный перебор сообщения.

for symbol **in** range(len(message)):

Перебор всех ключей в словаре.

for key **in** keys:

Если символ сообщения будет равен ключу, тогда символ сообщения будет равен значению ключа.

if message[symbol] == key:

 message[symbol] = keys[key]

Если символ сообщения будет равен значению ключа, тогда символ сообщения будет равен ключу.

elif message[symbol] == keys[key]:

 message[symbol] = key

else: pass

Вывод получившегося сообщения.

print("Final message: ", "".join(message))

009. Шифр Полибия.

(Шифр Полибия является также шифром замены. Шифр Полибия часто представляется в виде квадрата с прочерченными по горизонтали и вертикали числами от 1 до 6.)

// file.py

Start_

```
from re import findall
```

```
keysPolibiy = {
```

```
    'A':'11',    'B':'12',    'C':'13',    'D':'14',
    'E':'15',    'F':'16',    'G':'21',    'H':'22',
    'I':'23',    'J':'24',    'K':'25',    'L':'26',
    'M':'31',    'N':'32',    'O':'33',    'P':'34',
    'Q':'35',    'R':'36',    'S':'41',    'T':'42',
    'U':'43',    'V':'44',    'W':'45',    'X':'46',
    'Y':'51',    'Z':'52',    '0':'53',    '1':'54',
    '2':'55',    '3':'56',    '4':'61',    '5':'62',
    '6':'63',    '7':'64',    '8':'65',    '9':'66'
```

```
}
```

```
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
def regular(text):
```

```
    template = r"[0-9]{2}"
```

```
    return findall(template, text)
```

```
def encryptDecrypt(mode, message, final = []):
```

```
    if mode == 'E':
```

```
        for symbol in message:
```

```
            if symbol in keysPolibiy:
```

```
                final.append(keysPolibiy[symbol])
```

```
    else:
```

```
        for twoNumbers in regular(message):
```

```
            for key in keysPolibiy:
```

```
                if twoNumbers == keysPolibiy[key]:
```

```
                    final.append(key)
```

```
    return ".".join(final)
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

End

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: 22.15.26.26.33.45.33.36.26.14

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: 22.15.26.26.33.45.33.36.26.14

Final message: H.E.L.L.O.W.O.R.L.D

End

Импортирование функции findall из регулярных выражений.

from re import findall

Словарь с видом → символ : два_числа.

keysPolibiy = {

'A': '11',	'B': '12',	'C': '13',	'D': '14',
'E': '15',	'F': '16',	'G': '21',	'H': '22',
'I': '23',	'J': '24',	'K': '25',	'L': '26',
'M': '31',	'N': '32',	'O': '33',	'P': '34',
'Q': '35',	'R': '36',	'S': '41',	'T': '42',
'U': '43',	'V': '44',	'W': '45',	'X': '46',
'Y': '51',	'Z': '52',	'0': '53',	'1': '54',
'2': '55',	'3': '56',	'4': '61',	'5': '62',
'6': '63',	'7': '64',	'8': '65',	'9': '66'

}

Переключатель шифрования/расшифрования

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E', 'D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщение для шифрования/расшифрования

startMessage = input("Write the message: ").upper()

Функция с регулярным выражением, которая выделяет по два числа.

def regular(text):

template = r"[0-9]{2}"

return findall(template, text)

Функция с 3 аргументами.

```
def encryptDecrypt(mode, message, final = []):
```

Если переключатель равен 'E', тогда сделать посимвольный перебор сообщения.

```
if mode == 'E':
```

```
    for symbol in message:
```

Если символ будет найден в ключах, тогда к списку final добавить значение ключа.

```
if symbol in keysPolibiy:
```

```
    final.append(keysPolibiy[symbol])
```

Если переключатель равен 'D', тогда сделать перебор сообщения через регулярное выражение.

```
else:
```

```
    for twoNumbers in regular(message):
```

Перебор всех ключей в словаре.

```
for key in keysPolibiy:
```

Если два числа будут найдены в значениях ключа, тогда к списку final добавить сам ключ.

```
if twoNumbers == keysPolibiy[key]:
```

```
    final.append(key)
```

Вернуть сообщение.

```
return ".".join(final)
```

Вывод получившегося сообщения.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

010. Шифр Виженера.

(Шифр Виженера – самый знаменитый многоалфавитный шифр. Построен на конструкции шифра Цезаря. Содержит в себе 26 шифров Цезаря с разными смещениями по алфавиту.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
oneKey = input("Write the key: ").upper()
def encryptDecrypt(mode, message, key, final = ""):
    key *= len(message) // len(key) + 1
    for index, symbol in enumerate(message):
        if mode == 'E':
            temp = ord(symbol) + ord(key[index])
        else:
            temp = ord(symbol) - ord(key[index])
        final += chr(temp % 26 + ord('A'))
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage, oneKey))
End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Write the key: war

Final message: DECHONKRCZ

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: DECHONKRCZ

Write the key: war

Final message: HELLOWORLD

End

```

# Переключатель шифрования/расшифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

# Ключ-слово.
oneKey = input("Write the key: ").upper()

# Функция с 4 аргументами.
def encryptDecrypt(mode, message, key, final = ""):

# Подгоняем длину ключа под длину сообщения.
key *= len(message) // len(key) + 1

# Посимвольный перебор сообщения.
for index, symbol in enumerate(message):

# Если переключатель будет равен 'E', тогда к переменной temp присвоить число
символа по таблице ASCII + число символа-ключа по таблице ASCII.
if mode == 'E':
    temp = ord(symbol) + ord(key[index])

# Если переключатель будет равен 'D', тогда к переменной temp присвоить число
символа по таблице ASCII - число символа-ключа по таблице ASCII.
else:
    temp = ord(symbol) - ord(key[index])

# К переменной final прибавить получившийся символ.
final += chr(temp % 26 + ord('A'))

# Вернуть сообщение.
return final

# Вывод получившегося сообщения.
print("Final message:", encryptDecrypt(cryptMode, startMessage, oneKey))

```


011. Шифр Вернама.

(Шифр Вернама построен на конструкции шифра Цезаря. Является одним из самых криптостойких методов шифрования, т.к. использует случайные ключи для каждого символа.)

// file.py

Start_

```
from random import randint
```

```
from re import findall
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
def regular(text):
```

```
    template = r"[0-9]+"
```

```
    return findall(template, text)
```

```
def encryptDecrypt(mode, message, final = "", keys = []):
```

```
    if mode == 'E':
```

```
        for symbol in message:
```

```
            key = randint(0,25); keys.append(str(key))
```

```
            final += chr((ord(symbol) + key - 13)%26 + ord('A'))
```

```
        return final, ' '.join(keys)
```

```
    else:
```

```
        keys = input("Write the keys: ")
```

```
        for index, symbol in enumerate(message):
```

```
            final += chr((ord(symbol) - int(regular(keys)[index]) - 13)%26 +
```

```
ord('A'))
```

```
        return final
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

End

// terminal

Start_

```
$ python file.py
```

```
[E]ncrypt|[D]ecrypt: e
```

```
Write the message: helloworld
```

```
Final message: ('SXWZOTBFYA', '11/19/11/14/0/23/13/14/13/23/')
```

```
$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: SXWZOTBFYA
Write the keys: 11/19/11/14/0/23/13/14/13/23/
Final message: HELLOWORLD
]_End
```

```
# Импортирование функции randint из модуля random.
```

```
from random import randint
```

```
# Импортирование функции findall из регулярных выражений.
```

```
from re import findall
```

```
# Переключатель режимов шифрования
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
# Сообщение для шифрования/расшифрования
```

```
startMessage = input("Write the message: ").upper()
```

```
# Функция с регулярным выражением, возвращающая все числа
```

```
def regular(text):
```

```
    template = r"[0-9]+"
```

```
    return findall(template, text)
```

```
# Функция с 4 аргументами.
```

```
def encryptDecrypt(mode, message, final = "", keys = []):
```

```
# Если переключатель равен 'E', тогда сделать посимвольный перебор в сообщении.
```

```
if mode == 'E':
```

```
    for symbol in message:
```

```
# Создание случайного ключа.
```

```
key = randint(0,25); keys.append(str(key))
```

```
# Вычисление символа при помощи таблицы ASCII.
```

```
final += chr((ord(symbol) + key - 13)%26 + ord('A'))
```

```
# Вернуть зашифрованное сообщение и ключи
```

```
return final, '!.join(keys)
```

```
# Если переключатель равен 'D', тогда сделать ввод ключа.
```

```
else:
```

```
    keys = input("Write the keys: ")
```

```
# Посимвольный перебор зашифрованного сообщения
```

```
for index, symbol in enumerate(message):
```

```
# Вычисление символа при помощи таблицы ASCII.
```

```
# regular(keys) возвращает список, [index] даёт нам определённое число из списка.
```

```
final += chr((ord(symbol) - int(regular(keys)[index]) - 13)%26 + ord('A'))
```

```
# Вернуть сообщение.
```

```
return final
```

```
# Вывод получившегося сообщения.
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

012. Книжный шифр.

(Книжный шифр является одним из самых криптостойких методов шифрования. Особенность шифра заключена в ключе-книге, которая может являться любым текстом, начиная от газеты и заканчивая книгой.)

// file.py

Start_

```
from random import choice
```

```
from re import findall
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ")
```

```
bookKey = input("Write the book-key: ")
```

```
def regular(text):
```

```
    template = r"[0-9]+"
```

```
    return findall(template, text)
```

```
def encryptDecrypt(mode, message, key, final = ""):
```

```
    with open(key) as bookKey:
```

```
        book = bookKey.read()
```

```
    if mode == 'E':
```

```
        for symbolMessage in message:
```

```
            listIndexKey = []
```

```
            for indexKey, symbolKey in enumerate(book):
```

```
                if symbolMessage == symbolKey:
```

```
                    listIndexKey.append(indexKey)
```

```
            try: final += str(choice(listIndexKey)) + '/'
```

```
            except IndexError: pass
```

```
    else:
```

```
        for numbers in regular(message):
```

```
            for indexKey, symbolKey in enumerate(book):
```

```
                if numbers == str(indexKey):
```

```
                    final += symbolKey
```

```
    return final
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage, bookKey))
```

End

// book.txt

Start_

O how I faint when I of you do write Knowing a better spirit doth use your name And in the praise thereof spends all his might To make me tongue tied speaking of your fame But since your worth wide as the ocean is The humble as the proudest sail doth bear My saucy bark inferior far to his On your broad main doth wilfully appear Your shallowest help will hold me up afloat Whilst he upon your soundless deep doth ride Or being wracked I am a worthless boat He of tall building and of goodly pride Then if he thrive and I be cast away The worst was this my love was my decay

_End

// terminal

Start_

\$ python file.py

[E]ncrypt[D]ecrypt: e

Write the message: helloworld

Write the book-key: book.txt

Final message: 381/223/489/320/482/4/294/233/377/359/

\$ python file.py

[E]ncrypt[D]ecrypt: d

Write the message: 381/223/489/320/482/4/294/233/377/359/

Write the book-key: book.txt

Final message: helloworld

_End

Импорт функции choice из модуля random.

from random import choice

Импорт функции findall из регулярных выражений.

from re import findall

Переключатель режимов шифрования.

cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()

if cryptMode not in ['E', 'D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщения для шифрования/расшифрования.

startMessage = input("Write the message: ")

Ключ-книга.

```
bookKey = input("Write the book-key: ")
```

Функция с регулярным выражением, возвращающая список чисел.

```
def regular(text):  
    template = r"[0-9]+"  
    return findall(template, text)
```

Функция с 4 аргументами.

```
def encryptDecrypt(mode, message, key, final = ""):
```

Открытие ключа-книги на чтение.

```
with open(key) as bookKey:  
    book = bookKey.read()
```

Если переключатель будет равен 'E', тогда перебрать посимвольно сообщение.

```
if mode == 'E':  
    for symbolMessage in message:
```

Создание или обнуление списка

```
listIndexKey = []
```

Посимвольный перебор книги

```
for indexKey, symbolKey in enumerate(book):
```

Если символ сообщения будет равен символу в книге, тогда к списку listIndexKey добавить индекс этого символа в книге.

```
if symbolMessage == symbolKey:  
    listIndexKey.append(indexKey)
```

Попытка добавить к переменной final ключ символа, если же список пустой – пропустить.

```
try: final += str(choice(listIndexKey)) + '/'  
except IndexError: pass
```

Если переключатель равен 'D', то перебрать зашифрованное сообщение по ключам.

```
else:  
    for numbers in regular(message):
```

Посимвольный перебор книги и если номер зашифрованного сообщения будет равен индексу этого номера в книге, тогда к переменной final добавить этот символ.

```
for indexKey, symbolKey in enumerate(book):  
    if numbers == str(indexKey):  
        final += symbolKey
```

Вернуть сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage, bookKey))
```

013. XOR шифрование.

(XOR шифрование основано на логической операции Иключающее ИЛИ. Данное шифрование не требует переключателя.)

// file.py

Start_

```
message = list(input("Write the message: "))
```

```
key = input("Write the key: ")
```

```
for symbol in range(len(message)):
```

```
    try: message[symbol] = chr(ord(message[symbol]) ^ int(key))
```

```
    except ValueError: message[symbol] = chr(ord(message[symbol]) ^ ord(key))
```

```
print("Final message:", "".join(message))
```

_End

// terminal

Start_

```
$ python file.py
```

```
Write the message: helloworld
```

```
Write the key: 50
```

```
Final message: ZW^^]E]@^V
```

```
$ python file.py
```

```
Write the message: ZW^^]E]@^V
```

```
Write the key: 50
```

```
Final message: helloworld
```

_End

Сообщение для шифрования/расшифрования

```
message = list(input("Write the message: "))
```

Ключ

```
key = input("Write the key: ")
```

Посимвольный перебор сообщения

```
for symbol in range(len(message)):
```

Если key = целое число

```
try: message[symbol] = chr(ord(message[symbol]) ^ int(key))
```


Если key = символ

```
except ValueError: message[symbol] = chr(ord(message[symbol]) ^ ord(key))
```

Вывод получившегося сообщения.

```
print("Final message:", "".join(message))
```

014. Шифр с ложными символами.

(Ложные символы используются для увеличения криптостойкости шифра. Могут быть использованы во множестве методов шифрования.)

// file.py

Start_

```
from random import randint, choice
traps = ('"', '\\', '{', '}', '^', '№', '\n')
symbolAlpha = [chr(x) for x in range(65,91)]
symbolCrypt = ('!', '@', '#', '$', '%', '^', '&', '*', ')', '(', '~', '_', '+', '-', '=', '<', '>', '?', '/', '[', ']', ',', '|', ':', ';', '.')
keys = dict(zip(symbolAlpha, symbolCrypt))
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
def encryptDecrypt(mode, message, final = ""):
    message = list(message)
    if mode == 'E':
        length = len(message) // 4
        for _ in range(length):
            message.insert(randint(0, len(message)), choice(traps))
        for symbol in message:
            if symbol in keys:
                final += keys[symbol]
            else: final += symbol
    else:
        for symbol in message:
            for key in keys:
                if symbol == keys[key]:
                    final += key
    return final
print("Final message:", encryptDecrypt(cryptMode, startMessage))
]_End
```

/ terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: *%__=|"?_№

```
$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: *%__=|"?_$_№
Final message: HELLOWORLD
]_End
```

```
# Импорт двух функций из модуля random.
```

```
from random import randint, choice
```

```
# Кортеж с ложными символами.
```

```
traps = ('"', '\\', '{', '}', ',', '№', '\\')
```

```
# Символы алфавита от 'A' до 'Z'.
```

```
symbolAlpha = [chr(x) for x in range(65,91)]
```

```
# Символы для замены.
```

```
symbolCrypt = ('!', '@', '#', '$', '%', '^', '&', '*', ')', '(', '~',  
'_', '+', '-', '=', '<', '>', '?', '/', '[', ']', ',', '|', ':', ';', ':')
```

```
# Создание словаря вида → символ_алфавита : символ_шифра.
```

```
keys = dict(zip(symbolAlpha, symbolCrypt))
```

```
# Переключатель режимов шифрования.
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E', 'D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
# Сообщение для шифрования/расшифрования.
```

```
startMessage = list(input("Write the message: ").upper())
```

```
# Функция с 3 аргументами.
```

```
def encryptDecrypt(mode, message, final = ""):
```

```
# Если переключатель будет равен 'E', тогда создать переменную length.
```

```
if mode == 'E':
```

```
    length = len(message) // 4
```

```
# Вносить в сообщение рандомные ложные символы в рандомные позиции.
```

```
for _ in range(length):
```

```
    message.insert(randint(0,len(message)),choice(traps))
```

Посимвольный перебор сообщения

for symbol in message:

Если символ находится в ключах, тогда к переменной final добавить значение ключа.

if symbol in keys:

final += keys[symbol]

Иначе к переменной final добавить сам символ.

else: final += symbol

Если переключатель равен значению 'D', тогда сделать посимвольный перебор сообщения и перебор всех ключей.

else:

for symbol in message:

for key in keys:

Если символ будет равен значению ключа, тогда к переменной final добавить ключ.

if symbol == keys[key]:

final += key

Вернуть сообщение.

return final

Вывод получившегося сообщения.

print("Final message:", encryptDecrypt(cryptMode, startMessage))

015. Шифр Гронсфельда.

(Шифр Гронсфельда основан на шифре Виженера. Вместо ключа-слова используются числа. Соответственно вместо 26 возможных символов (шифр Виженера), получаем всего лишь 10 (шифр Гронсфельда).)

// file.py

Start_

```
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
keyNumber = input("Write the keyNumber: ")
def encryptDecrypt(mode, message, key, final = ""):
    key *= len(message) // len(key) + 1
    for index, symbol in enumerate(message):
        if mode == 'E':
            temp = ord(symbol) + int(key[index]) - 13
        else:
            temp = ord(symbol) - int(key[index]) - 13
        final += chr(temp%26 + ord('A'))
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage, keyNumber))
End
```

// terminal

Start_

```
$ python file.py
[E]ncrypt[D]ecrypt: e
Write the message: helloworld
Write the keyNumber: 2018
Final message: JEMTQWPZND
```

```
$ python file.py
[E]ncrypt[D]ecrypt: d
Write the message: JEMTQWPZND
Write the keyNumber: 2018
Final message: HELLOWORLD
```

End

```
# Переключатель режимов шифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

# Ключ-номер.
keyNumber = input("Write the keyNumber: ")

# Функция с 4 аргументами.
def encryptDecrypt(mode, message, key, final = ""):

# Подгоняем длину ключа к длине сообщения.
key *= len(message) // len(key) + 1

# Посимвольный перебор сообщения.
for index, symbol in enumerate(message):

# Если переключатель равен 'E', тогда прибавлять число ключа.
if mode == 'E':
    temp = ord(symbol) + int(key[index]) - 13

# Если переключатель равен 'D', тогда вычитать число ключа.
else:
    temp = ord(symbol) - int(key[index]) - 13

# К переменной final прибавить получившийся символ.
final += chr(temp%26 + ord('A'))

# Вернуть сообщение.
return final

# Вывод получившегося сообщения.
print("Final message:",encryptDecrypt(cryptMode, startMessage, keyNumber))
```

016. Псевдосимвольный шифр.

(Псевдосимвольный шифр основан на шифре Бэкона. Исключительная особенность данного метода шифрования заключена в трёх одинаковых внешне символах, но различающихся в кодировке Unicode. Один символ 'А' взят из латиницы, другой из кириллицы, третий из греческого алфавита.)

// file.py

Start_

```
from re import findall
keysPsevdo = {
    'A':"AAA", 'B':"AAA", 'C':"AAA",
    'D':"AAA", 'E':"AAA", 'F':"AAA",
    'G':"AAA", 'H':"AAA", 'I':"AAA",
    'J':"AAA", 'K':"AAA", 'L':"AAA",
    'M':"AAA", 'N':"AAA", 'O':"AAA",
    'P':"AAA", 'Q':"AAA", 'R':"AAA",
    'S':"AAA", 'T':"AAA", 'U':"AAA",
    'V':"AAA", 'W':"AAA", 'X':"AAA",
    'Y':"AAA", 'Z':"AAA", ' ': "AAA"
}
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
def regular(text):
    template = r"\w{3}"
    return findall(template, text)
def encryptDecrypt(mode, message, final = ""):
    if mode == 'E':
        for symbol in message:
            if symbol in keysPsevdo:
                final += keysPsevdo[symbol]
    else:
        for threeSymbols in regular(message):
            for key in keysPsevdo:
                if threeSymbols == keysPsevdo[key]:
                    final += key
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Final message: HELLOWORLD

End

Импортирование функции из регулярных выражений.

from re import findall

Словарь вида → символ_алфавита : 3символа.

keysPsevdo = {

'A': "AAA", 'B': "AAA", 'C': "AAA",
'D': "AAA", 'E': "AAA", 'F': "AAA",
'G': "AAA", 'H': "AAA", 'I': "AAA",
'J': "AAA", 'K': "AAA", 'L': "AAA",
'M': "AAA", 'N': "AAA", 'O': "AAA",
'P': "AAA", 'Q': "AAA", 'R': "AAA",
'S': "AAA", 'T': "AAA", 'U': "AAA",
'V': "AAA", 'W': "AAA", 'X': "AAA",
'Y': "AAA", 'Z': "AAA", ' ': "AAA"

}

Переключатель режимов шифрования.

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E', 'D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщение для шифрования/расшифрования.

startMessage = input("Write the message: ").upper()

Функция регулярного выражения возвращающая 3 символа.

def regular(text):

template = r"\w{3}"

return findall(template, text)

Функция с 3 аргументами.

```
def encryptDecrypt(mode, message, final = ""):
```

Если переключатель будет равен 'E', тогда сделать посимвольный перебор сообщения.

```
if mode == 'E':
```

```
    for symbol in message:
```

Если символ будет находится в ключах, тогда добавить к переменной final значение ключа.

```
if symbol in keysPsevdo:
```

```
    final += keysPsevdo[symbol]
```

Если переключатель будет равен 'D', тогда сделать перебор через функцию с регулярным выражением и сделать перебор всех ключей в словаре.

```
else:
```

```
    for threeSymbols in regular(message):
```

```
        for key in keysPsevdo:
```

Если три символа равны значению ключа в словаре, тогда к переменной final добавить сам ключ.

```
if threeSymbols == keysPsevdo[key]:
```

```
    final += key
```

Вернуть сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

017. Кодирование.

(Кодирование в криптографии – это замена целых слов на какой-либо шифртекст/слово. Относится к ветке шифров замены.)

// file.py

Start_

```
tupleWord = ('AND','THE','OR','ALL','ANY','WHAT','WHY','YES','NO',  
'ONE','YOU','HE','SHE','USE','IF','ELSE','THIS','THAN','YOUR',  
'ON','HOW','ARE','ME','IT','IS','THAT','WAS','OF','BE','OK')
```

```
tupleCode = ('!','@','#','$','%','^','&','*','(',')','-', '_',  
'+', '=', '/', '?', '<', '>', ';', ':', '{', '}', '[', ']', '~', ',', '.', '"', '|', '\\')
```

```
keys = dict(zip(tupleWord, tupleCode))
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
def encryptDecrypt(mode, message):
```

```
    for key in keys:
```

```
        if mode == 'E':
```

```
            if key in message:
```

```
                message = message.replace(key,keys[key])
```

```
        else:
```

```
            if keys[key] in message:
```

```
                message = message.replace(keys[key],key)
```

```
    return message
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

End

/ terminal

Start_

```
$ python file.py
```

```
[E]ncrypt|[D]ecrypt: e
```

```
Write the message: yes or no
```

```
Final message: * # (
```

```
$ python file.py
```

```
[E]ncrypt|[D]ecrypt: d
```

```
Write the message: * # (
```

```
Final message: YES OR NO
```

End

Кортеж слов.

```
tupleWord = ('AND','THE','OR','ALL','ANY','WHAT','WHY','YES','NO',  
'ONE','YOU','HE','SHE','USE','IF','ELSE','THIS','THAN','YOUR',  
'ON','HOW','ARE','ME','IT','IS','THAT','WAS','OF','BE','OK')
```

Кортеж кодов.

```
tupleCode = ('!','@','#','$','%','^','&','*','(',')','-', '_',  
'+', '=', '/', '?', '<', '>', ';', ':', '{', '}', '[', ']', '~', ',', '.', '"', '|', '\\')
```

Словарь вида → слово : код.

```
keys = dict(zip(tupleWord, tupleCode))
```

Переключатель режимов шифрования.

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

Сообщение для шифрования/расшифрования.

```
startMessage = input("Write the message: ").upper()
```

Функция с 2 аргументами.

```
def encryptDecrypt(mode, message):
```

Перебор всех ключей.

```
for key in keys:
```

Если переключатель будет равен 'E' и ключ будет находится в сообщении, то заменить этот ключ на значение ключа.

```
if mode == 'E':
```

```
    if key in message:
```

```
        message = message.replace(key,keys[key])
```

Если переключатель равен 'D' и значение ключа находится в сообщении, то заменить это значение ключа на сам ключ.

```
else:
```

```
    if keys[key] in message:
```

```
        message = message.replace(keys[key],key)
```

Вернуть сообщение.

```
return message
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

018. Шифр замены слогов.

(Шифр замены слогов используется для увеличения криптостойкости. Может быть использован во множестве методов шифрования.)

// file.py

Start_

```
syllables = ('TH','EE','OO','ING','ED','SS','DE','RE','AR',
'WH','AI','IS','BE','CH','SH','GH','EN','OU','LL','HE','US',
'ST','EV','WO','UI','IN','ER','OR','AT','RD','AL','LE','LD',
'UR','UP','SO','ME','SE','MY','NA','TE','NE','VE','LA','GE',
'ON','GU','RA','AN','AG','SH','CR','FO','OW','PY','WR','CA',
'EA','SP','PR','AS','AU','MA','KE','UT','DO','NT','WA','HU',
'AD','WI','RI','LO','FU','BR','OF','AP','TO','IF','AM','ND',
'LY','TA','KN','FA','TT','LP')
symbols = ('!$', '@@2', '#99', '$$', '^$', '&<<', '**?', ';;{',
'|}', '::#', '-/','++','/~','='],['++','?:','>>&', '//'?',
'&*','!<<','%((,':>','<;','*++','?//','^$$','~''''','!::',
':!','&??','//!','#$$','(:,:)', '{[[','[[',';;<','[[',
'$??','0//','1[[','@]]','[<,:]]','[:[',']:','!','!','@',
#','$','%','^','&','*','(',')','_','_','=','+','{','}',':',
';','''', '<','>','?','/','~','`','|','\\','[',']', '1','2',
'3','4','5','6','7','8','9','0')
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
def encryptDecrypt(mode, message):
    if mode == 'E':
        for syllable in syllables:
            if syllable in message:
                message =
message.replace(syllable,symbols[syllables.index(syllable)])
    else:
        for symbol in symbols:
            if symbol in message:
                message = message.replace(symbol,
syllables[symbols.index(symbol)])
    return message
print("Final message:",encryptDecrypt(cryptMode, startMessage))
_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: !<&**O*++R((:

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: !<&**O*++R((:

Final message: HELLOWORLD

End

Кортеж слогов.

```
syllables = ('TH','EE','OO','ING','ED','SS','DE','RE','AR',  
'WH','AI','IS','BE','CH','SH','GH','EN','OU','LL','HE','US',  
'ST','EV','WO','UI','IN','ER','OR','AT','RD','AL','LE','LD',  
'UR','UP','SO','ME','SE','MY','NA','TE','NE','VE','LA','GE',  
'ON','GU','RA','AN','AG','SH','CR','FO','OW','PY','WR','CA',  
'EA','SP','PR','AS','AU','MA','KE','UT','DO','NT','WA','HU',  
'AD','WI','RI','LO','FU','BR','OF','AP','TO','IF','AM','ND',  
'LY','TA','KN','FA','TT','LP')
```

Кортеж шифров.

```
symbols = ('!$', '@@2', '#99', '$$!', '^ ^$', '&<<', '**?', ';;{',  
'||}', ':::#', '- /', '+ +', '//~', '=]', '[++', '?::', '>>&', '///?',  
'&**', '!<<', '%((', ':>', '<;', '*++', '?//', '^$$', '~""', '!::',  
'::!', '&??', '///', '#$$', '(:, :)', '{[', '[[]', ';;<', '[[]',  
'$??', '0//', '1[[', '@[]', '[<', ':[]', ':[[]', ':', ':!', '@',  
'#', '$', '%', '^', '&', '*', '(', ')', '-', '_', '=', '+', '{', '}', ':',  
';', '"', ',', '<', '>', '?', '/', '~', '\', '|', '\\', '[', ']', '1', '2',  
'3', '4', '5', '6', '7', '8', '9', '0')
```

Переключатель режимов шифрования.

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E', 'D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

Сообщения для шифрования/расшифрования.

```
startMessage = input("Write the message: ").upper()
```

```
# Функция с 2 аргументами.
def encryptDecrypt(mode, message):

    # Если переключатель равен 'E', тогда сделать перебор всех слогов в кортеже.
    if mode == 'E':
        for syllable in syllables:

            # Если слог находится в сообщении, то заменить этот слог на символ из шифров.
            if syllable in message:
                message = message.replace(syllable, symbols[syllables.index(syllable)])

    # Если переключатель равен 'D', тогда сделать перебор всех шифров в кортеже.
    else:
        for symbol in symbols:

            # Если шифр находится в сообщении, то заменить шифр на слог.
            if symbol in message:
                message = message.replace(symbol, syllables[symbols.index(symbol)])

    # Вернуть сообщение.
    return message

# Вывод получившегося сообщения.
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

019. Шифр Порты.

(Шифр порты является биграммным методом шифрования. Смысл заключён в шифровании двух символов одной триадой чисел. Чаще всего шифр Порты представляют в виде таблицы, где по горизонтали и вертикали прочертаны символы алфавита, а в самой таблице находятся триады чисел.)

// file.py

Start_

```
from re import findall
stageOne = ['00'+str(x) for x in range(1,10)]
stageTwo = ['0'+str(x) for x in range(10,100)]
stageThree = [str(x) for x in range(100,676+1)]
N = tuple(stageOne + stageTwo + stageThree)
del stageOne, stageTwo, stageThree
coordinateX = tuple([chr(alpha) for alpha in range(65,91)])
coordinateY = tuple([chr(alpha) for alpha in range(65,91)])
cryptKeys = {x:None for x in N}
keys = tuple([key for key in cryptKeys])
counter = 0
for x in coordinateX:
    for y in coordinateY:
        cryptKeys[keys[counter]] = x + y
        counter += 1
del N, coordinateX, coordinateY, counter, keys
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode not in (E/D)")
    raise SystemExit
startMessage = input("Write the message: ").upper()
def regular(mode, text):
    if mode == 'E': template = r"[A-Z]{2}"
    else: template = r"[0-9]{3}"
    return findall(template, text)
def encryptDecrypt(mode, message, final = []):
    if mode == 'E':
        for symbol in message:
            if symbol not in [chr(x) for x in range(65,91)]:
                message = message.replace(symbol,"")
        if len(message)%2 != 0: message += 'Z'
        for symbols in regular(mode, message):
            for key in cryptKeys:
```



```

        if symbols == cryptKeys[key]:
            final.append(key)
    else:
        for number in regular(mode, message):
            if number in cryptKeys:
                final.append(cryptKeys[number])
    return ".".join(final)
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_

```

$ python file.py
[E]ncrypt[D]ecrypt: e
Write the message: helloworld
Final message: 187.298.387.382.290

```

```

$ python file.py
[E]ncrypt[D]ecrypt: d
Write the message: 187.298.387.382.290
Final message: HE.LL.OW.OR.LD

```

]_End

Импортир функции из регулярных выражений

```

from re import findall

```

Создание списков содержащих числа от 001 до 676.

```

stageOne = ['00'+str(x) for x in range(1,10)]
stageTwo = ['0'+str(x) for x in range(10,100)]
stageThree = [str(x) for x in range(100,676+1)]

```

Ввод всех списков в один кортеж N и удаление предыдущих списков.

```

N = tuple(stageOne + stageTwo + stageThree)
del stageOne, stageTwo, stageThree

```

Создание координат с алфавитом от A-Z.

```

coordinateX = tuple([chr(alpha) for alpha in range(65,91)])
coordinateY = tuple([chr(alpha) for alpha in range(65,91)])

```

```

# Создание ключей.
cryptKeys = {x:None for x in N}
keys = tuple([key for key in cryptKeys])
# Перебор всех возможных пар алфавита.
counter = 0
for x in coordinateX:
    for y in coordinateY:

# Занесение в словарь значений ключа.
cryptKeys[keys[counter]] = x + y
counter += 1

# Удаление ненужных элементов.
del N, coordinateX, coordinateY, counter, keys

# Создание переключателя шифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode not in (E/D)")
    raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

# Регулярное выражение с условием, если зашифровать сообщение, значит
разграничить сообщение по 2 символа, если же расшифровать, значить
разграничить сообщение по 3 числа и вернуть список.
def regular(mode, text):
    if mode == 'E': template = r"[A-Z]{2}"
    else: template = r"[0-9]{3}"
    return findall(template, text)

# Создание главной функции
def encryptDecrypt(mode, message, final = []):

# Если переключатель равен шифрованию, значит перебрать все символы в
сообщении.
if mode == 'E':
    for symbol in message:

```

Если символ не будет находится в диапазоне A-Z, тогда удалить этот символ.

```
if symbol not in [chr(x) for x in range(65,91)]:  
    message = message.replace(symbol, '')
```

Если длина сообщения не делится на 2 без остатка – добавить в конец сообщения символ 'Z'.

```
if len(message)%2 != 0: message += 'Z'
```

Перебор пар символов через регулярное выражение.

```
for symbols in regular(mode, message):  
    for key in cryptKeys:
```

Если символ будет равен значению ключа, значит заменить на сам ключ.

```
if symbols == cryptKeys[key]:  
    final.append(key)
```

Если же переключатель равен расшифровки, тогда сделать перебор трёх чисел через функцию с регулярным выражением.

```
else:  
    for number in regular(mode, message):
```

Если число находится в ключах – заменить на значение ключа.

```
if number in cryptKeys:  
    final.append(cryptKeys[number])
```

Вернуть сообщение.

```
return ".".join(final)
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

020. Роторное шифрование.

(Роторное шифрование построено на принципе шифра Вернама, то-есть с каждым новым поступающим на вход символом ключ будет изменяться по роторам, имитируя случайность. В основе роторного шифрования лежит шифр Цезаря.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
rotors = (
```

```
    (10,24,14,12,23,2,7,15,24,2,7,5,22,6,2,1,22,12,6,9,7,2,11,23,14,2),
```

```
    (1,7,11,26,12,5,11,20,11,7,18,6,17,18,19,1,13,5,2,9,11,13,6,17,26,24),
```

```
    (9,1,21,6,4,19,25,6,17,10,26,1,23,6,1,17,19,17,25,21,3,21,17,1,18,20)
```

```
)
```

```
def encryptDecrypt(mode, message, final = ""):
```

```
    x,y,z = 1,2,3
```

```
    for symbol in message:
```

```
        rotor = rotors[0][x] + rotors[1][y] + rotors[2][z]
```

```
        if mode == 'E':
```

```
            if symbol in [chr(x) for x in range(65,91)]:
```

```
                final += chr((ord(symbol) - 13 + rotor)%26 + ord('A'))
```

```
            else: continue
```

```
        else:
```

```
            final += chr((ord(symbol) - 13 - rotor)%26 + ord('A'))
```

```
        if x != 25: x += 1
```

```
        else:
```

```
            x = 0
```

```
            if y != 25: y += 1
```

```
            else:
```

```
                y = 0
```

```
                if z != 25: z += 1
```

```
                else: z = 0
```

```
    return final
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

End

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: WJOZHUUGEB

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: WJOZHUUGEB

Final message: HELLOWORLD

End

Переключатель режимов шифрования.

cryptMode = **input**("[E]ncrypt|[D]ecrypt: ").**upper**()

if cryptMode **not in** ['E','D']:

print("Error: mode is not Found!"); **raise** **SystemExit**

Сообщение для шифрования/расшифрования.

startMessage = **input**("Write the message: ").**upper**()

Роторные позиции.

rotors = (

 (10,24,14,12,23,2,7,15,24,2,7,5,22,6,2,1,22,12,6,9,7,2,11,23,14,2),

 (1,7,11,26,12,5,11,20,11,7,18,6,17,18,19,1,13,5,2,9,11,13,6,17,26,24),

 (9,1,21,6,4,19,25,6,17,10,26,1,23,6,1,17,19,17,25,21,3,21,17,1,18,20)

)

Главная функция.

def **encryptDecrypt**(**mode**, **message**, **final** = ""):

Начальная позиция роторов.

x,y,z = 1,2,3

Посимвольный перебор сообщения и прибавление всех позиций роторов.

for **symbol** **in** **message**:

rotor = **rotors**[0][**x**] + **rotors**[1][**y**] + **rotors**[2][**z**]

Если переключатель равен режиму шифрования, значит проверить встречается ли символ в диапазоне A-Z. И если это так, то зашифровать.

```
if mode == 'E':  
    if symbol in [chr(x) for x in range(65,91)]:  
        final += chr((ord(symbol) - 13 + rotor)%26 + ord('A'))  
    else: continue
```

Если переключатель равен режиму расшифрования, значит расшифровывать посимвольно сообщение.

```
else:  
    final += chr((ord(symbol) - 13 - rotor)%26 + ord('A'))
```

Если x не равняется 25 (конечному значению в кортеже), значит к x прибавлять единицу. Если же x равен 25 – округлить значение и к y прибавить единицу. Также работает и с другими кортежами y,z.

```
if x != 25: x += 1  
else:  
    x = 0  
    if y != 25: y += 1  
    else:  
        y = 0  
        if z != 25: z += 1  
        else: z = 0
```

Вернуть сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

021. Криптографические хеш-функции.

(Криптографические хеш-функции не предназначены для расшифровки. Они созданы лишь для проверки и подтверждения информации.
(На примере показана криптографическая хеш-функция sha256, но в библиотеке hashlib также есть и другие, наподобие md5, sha512 и т.д.))

// file.py

Start_

```
from hashlib import sha256
def encrypt(string):
    signature = sha256(string.encode()).hexdigest()
    return signature
staticPassword = encrypt("secret")
dynamicPassword = encrypt(input("Write the password: "))
if staticPassword == dynamicPassword:
    print("Password is True!")
else:
    print("Password is False!")
```

_End

// terminal

Start_

```
$ python file.py
Write the password: secret
Password is True!
```

```
$ python file.py
Write the password: hello
Password is False!
```

_End

Импортирование функции sha256 из библиотеки hashlib

```
from hashlib import sha256
```

Создание функции, которая будет возвращать хешированное сообщение.

```
def encrypt(string):
    signature = sha256(string.encode()).hexdigest()
    return signature
```

Пароль, который является статичным.

staticPassword = **encrypt**("secret")

Пароль, который мы пытаемся сравнить со статичным.

dynamicPassword = **encrypt**(**input**("Write the password: "))

Если статичный пароль равен динамическому, значит пароль верный.

if **staticPassword** == **dynamicPassword**:

print("Password is True!")

Иначе пароль не верный.

else:

print("Password is False!")

022. Степени шифра Виженера.

(Многоалфавитные шифры обладают способностью множественного шифрования, то-есть способны шифровать одно и то же сообщение многократно усиливая криптостойкость зашифрованного сообщения.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
try: numberKeys = int(input("How much keys: "))
except: print("Only integer numbers!"); raise SystemExit
listKeys = []
for index in range(numberKeys):
    listKeys.append(input("Write the keyWord["+str(index)+"]: ").upper())
def encryptDecrypt(mode, message, keys):
    for key in listKeys:
        final = ""
        key *= len(message) // len(key) + 1
        for index, symbol in enumerate(message):
            if mode == 'E':
                temp = ord(symbol) + ord(key[index])
            else:
                temp = ord(symbol) - ord(key[index])
            final += chr(temp % 26 + ord('A'))
        message = final
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage, listKeys))
]_End
```

// terminal

Start_

```
$ python file.py
[E]ncrypt|[D]ecrypt: e
Write the message: helloworld
How much keys: 3
Write the keyWord[0]: python
Write the keyWord[1]: good
Write the keyWord[2]: language
Final message: NQFBCXXWVY
```

```
$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: NQFBCXXWVY
How much keys: 3
Write the keyWord[0]: python
Write the keyWord[1]: good
Write the keyWord[2]: language
Final message: HELLOWORLD
```

_End

Переключатель для шифрования.

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
```

Сообщение для шифрования/расшифрования.

```
startMessage = input("Write the message: ").upper()
```

Ключ является целым числом, если введено не целое число — вывод ошибки.

```
try: numberKeys = int(input("How much keys: "))
except: print("Only integer numbers!"); raise SystemExit
```

Создание списка ключей.

```
listKeys = []
```

Добавление в список ключей.

```
for index in range(numberKeys):
    listKeys.append(input("Write the keyWord["+str(index)+"]: ").upper())
```

Создание главной функции.

```
def encryptDecrypt(mode, message, keys):
```

Перебор всех ключей в списке ключей. Создание переменной final. Подгон длины ключа под длину сообщения.

```
for key in listKeys:
```

```
    final = ""
```

```
    key *= len(message) // len(key) + 1
```

Посимвольный перебор сообщения.

```
for index, symbol in enumerate(message):
```

Если переключатель равен шифрованию, значит присваивать к переменной temp сложение.

```
if mode == 'E':
```

```
    temp = ord(symbol) + ord(key[index])
```

Если переключатель равен расшифрованию, значит присваивать к переменной temp вычитание.

```
else:
```

```
    temp = ord(symbol) - ord(key[index])
```

Посимвольное добавление в переменную final зашифрованных символов.

Присвоить переменную final к переменной message.

```
final += chr(temp % 26 + ord('A'))
```

```
message = final
```

Вернуть сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage, listKeys))
```

023. Шифр Штакетник.

(Шифр Штакетник относится к шифрам перестановки и является одним из самых лёгких в своём роде. Принцип шифрования – разделение всех символов на чётные и нечётные позиции, а далее совмещение групп символов, сначала чётные, потом нечётные.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ")
def encryptDecrypt(mode, message, final = ""):
    if mode == 'E':
        encryptList = [
            message[x] for x in range(len(message)) if x%2 == 0],
            message[x] for x in range(len(message)) if x%2 != 0]
        for index in range(len(encryptList)):
            final += "".join(encryptList[index])
    else:
        if len(message)%2 != 0: message += ' '
        length, half = len(message), len(message)//2
        decryptList = [
            message[x] for x in range(half)],
            message[x] for x in range(half,length)]
        for index in range(half):
            final += decryptList[0][index]+decryptList[1][index]
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
_End
```

// terminal

Start_

```
$ python file.py
[E]ncrypt|[D]ecrypt: e
Write the message: helloworld
Final message: hloolelwrđ
```

```
$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: hloolelwrđ
Final message: helloworld
]_End
```

```
# Переключатель режимов шифрования
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования
startMessage = input("Write the message: ")

# Главная функция
def encryptDecrypt(mode, message, final = ""):

    # Если переключатель равен шифрованию, значит создать списки всех чётных и
    # нечётных символов из сообщения.
    if mode == 'E':
        encryptList = [
            message[x] for x in range(len(message)) if x%2 == 0],
            message[x] for x in range(len(message)) if x%2 != 0]
        ]

    # Перебор всех списков и заполнение переменной final символами.
    for index in range(len(encryptList)):
        final += "".join(encryptList[index])

    # Если переключатель равен расшифрованию и если длина сообщения не кратно
    # двум, значит добавить в конец сообщения символ-пробел.
    else:
        if len(message)%2 != 0: message += ' '

    # Создание переменных длины сообщения и половины длины сообщения.
    length, half = len(message), len(message)//2

    # Создание списков с символами от нуля до середины сообщения и от середины до
    # конца сообщения.
    decryptList = [
        message[x] for x in range(half),
```

```
[message[x] for x in range(half,length)]  
]
```

Занесение в переменную final сразу двух символов с одинаковым индексом из двух разных списков.

```
for index in range(half):  
    final += decryptList[0][index]+decryptList[1][index]
```

Вернуть сообщение.

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

024. Шифр решётки.

(Шифр решётки скрывает открытое сообщение в ложных символах.)

// file.py

Start_

```
from random import choice, randint
squade = 10
alphaList = [chr(x) for x in range(65,91)] + [chr(y) for y in range(97,123)]
stringList = [choice(alphaList) for _ in range(squade*squade)]
def getKey(text):
    while True:
        keys = [randint(0,99) for _ in range(len(text))]
        for number in keys:
            switch = False
            if keys.count(number) > 1:
                switch = True; break
        if switch == False:
            return keys
def lattice(index = 0):
    print(end = ' ')
    for string in range(squade):
        print(string, end = ' ')
    for string in range(squade):
        print()
        for column in range(squade):
            if index%squade == 0:
                print(index//squade, end = ' | ')
            print(stringList[index], end = ' | ')
            index += 1
        print()
message = input("Write the message: ")
keyList = getKey(message)
keyList.sort()
print("Keys:",keyList)
for index, symbol in enumerate(message):
    del stringList[keyList[index]]
    stringList.insert(keyList[index],symbol)
lattice()
_End
```

// terminal

Start_1

\$ python file.py

```
Write the message: helloworld
Keys: [0, 9, 16, 19, 29, 31, 38, 39, 81, 86]
 0  1  2  3  4  5  6  7  8  9
0 | h | A | g | R | D | g | b | v | G | e |
1 | V | Q | r | u | s | E | l | K | X | l |
2 | t | Y | f | a | a | L | y | D | L | o |
3 | f | w | T | Y | i | F | F | T | o | r |
4 | x | Z | W | H | r | J | p | q | Q | y |
5 | t | Q | m | V | p | d | y | k | f | D |
6 | u | A | R | c | O | J | q | q | J | L |
7 | y | L | l | d | o | g | P | M | d | A |
8 | A | l | v | b | q | D | d | O | A | w |
9 | Y | f | J | n | A | q | f | i | Y | b |
```

1_End

Импортирование двух функций из модуля random.

from random import choice, randint

Создание переменной, которая является длиной квадрата.

squade = 10

Создание списка символов диапазона A-Z и a-z.

alphaList = [chr(x) for x in range(65,91)] + [chr(y) for y in range(97,123)]

Создание списка с случайно разброшенными символами.

stringList = [choice(alphaList) for _ in range(squade*squade)]

Создание функции getKey (Создание ключа).

def getKey(text):

Создание списка случайных ключей по длине сообщения.

while True:

keys = [randint(0,99) for _ in range(len(text))]

Перебор всех чисел в ключе.

for number in keys:

switch = False

Если находятся одинаковые ключи, то переменная switch становится равной True и цикл for останавливается.

```
if keys.count(number) > 1:  
    switch = True; break
```

Если же переменная switch равна False (в ключах нет одинаковых значений), то вернуть ключи.

```
if switch == False:  
    return keys
```

Создание новой функции lattice (Создание визуальной решётки).

```
def lattice(index = 0):  
    print(end = '  ' )  
    for string in range(squade):  
        print(string, end = '  ' )  
    for string in range(squade):  
        print()  
        for column in range(squade):  
            if index%squade == 0:  
                print(index//squade, end = ' | ' )  
            print(stringList[index], end = ' | ' )  
            index += 1  
    print()
```

Сообщение для шифрования.

```
message = input("Write the message: ")
```

Присваивание списка ключей к переменной и их отсортировка.

```
keyList = getKey(message)  
keyList.sort()
```

Вывод ключей.

```
print("Keys:",keyList)
```

Добавление в stringList символов сообщения.

```
for index, symbol in enumerate(message):  
    del stringList[keyList[index]]  
    stringList.insert(keyList[index],symbol)  
lattice()
```

025. Шифр с омофонами.

(Омофонический шифр построен на шифре замены. Особенность шифра заключена во множестве шифр-символов под один символ текста. То-есть буква 'А' может быть зашифрована разными шифр-символами: '1' или '?' или 'Z'.)

// file.py

Start_1

```
from random import choice
values = ('1','2','3','4','5','6','7','8','9','0','a','b','c',\
'd','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s',\
't','u','v','w','x','y','z','!','@','\','#','№','$','%','^',\
':','&','?','(',')','-', '_', '+', '=', '~', '[', ']', '{', \
'}', ',', '/', '|', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L', \
'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '<', \
'>', 'A', 'M', 'B', 'C', 'y', 'E', 'T', 'a', 'X', '3')
dictHom = {
    'A':values[0:8],    'B':values[8:10],
    'C':values[10:13], 'D':values[13:17],
    'E':values[17:29], 'F':values[29:31],
    'G':values[31:33], 'H':values[33:39],
    'T':values[39:45], 'J':[values[45]],
    'K':[values[46]],  'L':values[47:51],
    'M':values[51:53], 'N':values[53:59],
    'O':values[59:66], 'P':values[66:68],
    'Q':[values[68]],  'R':values[69:75],
    'S':values[75:81], 'T':values[81:90],
    'U':values[90:93], 'V':[values[93]],
    'W':values[94:96], 'X':[values[96]],
    'Y':values[97:99], 'Z':[values[99]]
}
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ")
def encryptDecrypt(mode, message, final = ""):
    if mode == 'E':
        for symbol in message.upper():
            if symbol in dictHom:
                final += choice(dictHom[symbol])
    else:
        for symbol in message:
```

```

        for key in dictHom:
            if symbol in dictHom[key]:
                final += key

    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_

```

$ python file.py
[E]ncrypt|[D]ecrypt: e
Write the message: helloworld
Final message: \r)-.y,G?d

```

```

$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: \r)-.y,G?d
Final message: HELLOWORLD

```

]_End

Импорт функции из модуля random.

from random import choice

Значения для шифрования.

```

values = ['1','2','3','4','5','6','7','8','9','0','a','b','c',\
'd','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s',\
't','u','v','w','x','y','z','!','@','\','#','№','$',';','%','^',\
':','&','?','(',')','_','+','=','~','[',']','{',\
'}',',','.','/',|','A','B','C','D','E','F','G','H','J','K','L',\
'M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','<',\
'>','A','M','B','C','y','E','T','a','X','3']

```

Словарь с ключами и значениями-списками.

```

dictHom = {
    'A':values[0:8],  'B':values[8:10],
    'C':values[10:13], 'D':values[13:17],
    'E':values[17:29], 'F':values[29:31],
    'G':values[31:33], 'H':values[33:39],
    'T':values[39:45], 'J':[values[45]],
    'K':[values[46]],  'L':values[47:51],

```

```

'M':values[51:53], 'N':values[53:59],
'O':values[59:66], 'P':values[66:68],
'Q':[values[68]], 'R':values[69:75],
'S':values[75:81], 'T':values[81:90],
'U':values[90:93], 'V':[values[93]],
'W':values[94:96], 'X':[values[96]],
'Y':values[97:99], 'Z':[values[99]]
}

# Переключатель шифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования
startMessage = input("Write the message: ")

# Главная функция
def encryptDecrypt(mode, message, final = ""):

    # Если переключатель равен шифрованию, значит сделать посимвольный перебор
    # сообщения и если символ будет находится в ключах, значит к переменной final
    # добавить случайно-выбранное значение ключа из списка.
    if mode == 'E':
        for symbol in message.upper():
            if symbol in dictHom:
                final += choice(dictHom[symbol])

    # Если же переключатель равен расшифрованию, значит сделать посимвольный
    # перебор сообщения и перебор всех ключей. И если символ будет находится в
    # значениях ключа, значит к переменной final добавлять сам ключ.
    else:
        for symbol in message:
            for key in dictHom:
                if symbol in dictHom[key]:
                    final += key

    # Вернуть сообщение.
    return final

# Вывод получившегося сообщения.
print("Final message:", encryptDecrypt(cryptMode, startMessage))

```

026. Шифровальная машина <Турех>.

(Шифровальная машина <Турех> основана на роторном шифровании и шифре пар. Фактически является шифровальной машиной <Enigma> с убранным изъёмом, который заключался в инволюции (рефлекторах). Именно поэтому <Турех> требует переключателя режимов шифрования.)

// file.py

Start_

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
rotors = (
    (10,24,14,12,23,2,7,15,24,2,7,5,22,6,2,1,22,12,6,9,7,2,11,23,14,2),
    (1,7,11,26,12,5,11,20,11,7,18,6,17,18,19,1,13,5,2,9,11,13,6,17,26,24),
    (9,1,21,6,4,19,25,6,17,10,26,1,23,6,1,17,19,17,25,21,3,21,17,1,18,20)
)
switch = {
    'H':'Z', 'S':'N', 'L':'M',
    'P':'Q', 'R':'W', 'X':'Y'
}
def stageOne(message):
    message = list(message)
    for symbol in range(len(message)):
        for key in switch:
            if message[symbol] == key:
                message[symbol] = switch[key]
            elif message[symbol] == switch[key]:
                message[symbol] = key
            else: pass
    return "".join(message)
def stageTwo(mode, message, final = ""):
    X,Y,Z = 2,0,1; x,y,z = 1,2,3
    for symbol in message:
        rotor = rotors[X][x] + rotors[Y][y] + rotors[Z][z]
        if mode == 'E':
            if symbol in [chr(x) for x in range(65,91)]:
                final += chr((ord(symbol) - 13 + rotor)%26 + ord('A'))
            else: continue
        else:
            final += chr((ord(symbol) - 13 - rotor)%26 + ord('A'))
```

```

        if x != 25: x += 1
    else:
        x = 0
        if y != 25: y += 1
    else:
        y = 0
        if z != 25: z += 1
    else: z = 0

    return final
def encryptDecrypt(mode, message):
    if mode == 'E':
        message = stageOne(message)
        message = stageTwo(mode, message)
    else:
        message = stageTwo(mode, message)
        message = stageOne(message)
    return message
print("Final message:", encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_1

```

$ python file.py
[E]ncrypt|[D]ecrypt: e
Write the message: helloworld
Final message: RSFDVJIRJW

```

```

$ python file.py
[E]ncrypt|[D]ecrypt: d
Write the message: RSFDVJIRJW
Final message: HELLOWORLD

```

]_End

```

# Переключатель режимов шифрования.
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E', 'D']:
    print("Error: mode is not Found!"); raise SystemExit

# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

```

Роторы.

```
rotors = (  
    (10,24,14,12,23,2,7,15,24,2,7,5,22,6,2,1,22,12,6,9,7,2,11,23,14,2),  
    (1,7,11,26,12,5,11,20,11,7,18,6,17,18,19,1,13,5,2,9,11,13,6,17,26,24),  
    (9,1,21,6,4,19,25,6,17,10,26,1,23,6,1,17,19,17,25,21,3,21,17,1,18,20)  
)
```

Коммутаторы.

```
switch = {  
    'H':'Z', 'S':'N', 'L':'M',  
    'P':'Q', 'R':'W', 'X':'Y'  
}
```

Функция stageOne выполняет полностью аналогичные действия шифра пар.

```
def stageOne(message):  
    message = list(message)  
    for symbol in range(len(message)):  
        for key in switch:  
            if message[symbol] == key:  
                message[symbol] = switch[key]  
            elif message[symbol] == switch[key]:  
                message[symbol] = key  
            else: pass  
    return "".join(message)
```

Функция stageTwo выполняет полностью аналогичные действия роторного шифрования, за исключением возможности переставлять сами роторы местами.

```
def stageTwo(mode, message, final = ""):  
    X,Y,Z = 2,0,1; x,y,z = 1,2,3  
    for symbol in message:  
        rotor = rotors[X][x] + rotors[Y][y] + rotors[Z][z]  
        if mode == 'E':  
            if symbol in [chr(x) for x in range(65,91)]:  
                final += chr((ord(symbol) - 13 + rotor)%26 + ord('A'))  
            else: continue  
        else:  
            final += chr((ord(symbol) - 13 - rotor)%26 + ord('A'))  
        if x != 25: x += 1  
        else:  
            x = 0  
            if y != 25: y += 1  
            else:  
                y = 0  
                if z != 25: z += 1  
                else:  
                    z = 0
```

```
        y = 0
        if z != 25: z += 1
        else: z = 0
    return final
```

Главная функция служит переключателем режимов шифрования.

```
def encryptDecrypt(mode, message):
    if mode == 'E':
        message = stageOne(message)
        message = stageTwo(mode, message)
    else:
        message = stageTwo(mode, message)
        message = stageOne(message)
    return message
```

Вывод получившегося сообщения.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```


027. Шифр двойной цифри.

(Двойная цифирь является шифром перестановки по определённому ключу. Принцип шифрования – разделение текста на две части в виде таблицы образуя тем самым пары символов. Далее пары символов индексируются и переставляются местами. В итоге индексы будут являться ключами.)

// file.py

Start_

```
from re import findall
from random import choice
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = list(input("Write the message: ").upper())
def regular(text):
    template = r"[0-9]+"
    return findall(template, text)
def encryptDecrypt(mode, message, final = "", key = []):
    if mode == 'E':
        if len(message) % 2 != 0: message.append(' ')
        listHalf = [
            message[x] for x in range(len(message)//2, len(message)),
            message[y] for y in range(len(message)//2)]
        keys = {x:[listHalf[0][x],listHalf[1][x]] for x in range(len(message)//2)}
        listKey = [x for x in range(len(keys))]
        newList = []
        for _ in range(len(keys)):
            choiceKey = choice(listKey); key.append(str(choiceKey))
            newList.append(keys[choiceKey]); listKey.remove(choiceKey)
        for listIndex in range(len(newList)):
            for symbol in newList[listIndex]:
                final += symbol
        return final, ' '.join(key)
    else:
        listHalf = [
            message[x] for x in range(len(message)) if x%2 != 0],
            message[y] for y in range(len(message)) if y%2 == 0]]
        key = regular(input("Write the key: "))
        key = [int(x) for x in key]
        keys = {y:[listHalf[0][x],listHalf[1][x]] for x,y in enumerate(key)}
        finalList = [
```

```

[keys[x][0] for x in range(len(keys)) if x in keys],
[keys[y][1] for y in range(len(keys)) if y in keys]]
    for i in range(2):
        for index in range(len(message)//2):
            final += finalList[i][index]
    return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: ('OEDOWHRLLL', '1.4.0.2.3')

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: OEDOWHRLLL

Write the key: 1.4.0.2.3

Final message: HELLOWORLD

]_End

Импортирование функций из модулей re и random.

from re import findall

from random import choice

Переключатель режимов шифрования

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E', 'D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщение для шифрования/расшифрования.

startMessage = list(input("Write the message: ").upper())

Функция для создания списка с числами.

def regular(text):

template = r"[0-9]+"

return findall(template, text)

Создание главной функции.

```
def encryptDecrypt(mode, message, final = "", key = []):
```

Если переключатель равен шифрованию и если длина сообщения не кратна двум, тогда добавить в конец сообщения символ Z.

```
if mode == 'E':
```

```
    if len(message) % 2 != 0: message.append('Z')
```

Создать два списка. В первом все символы от середины до конца сообщения. Во втором все символы от начала до середины сообщения.

```
listHalf = [
```

```
    [message[x] for x in range(len(message)//2, len(message))],
```

```
    [message[y] for y in range(len(message)//2)]]
```

Создание словаря ключей со списком в виде значений. Создание списка ключей.

```
keys = {x:[listHalf[0][x],listHalf[1][x]] for x in range(len(message)//2)}
```

```
listKey = [x for x in range(len(keys))]
```

Создание списка для добавления последовательности ключей.

```
NewList = []
```

Перебор всех ключей и выбор случайных из списка listKey.

```
for _ in range(len(keys)):
```

```
    choiceKey = choice(listKey); key.append(str(choiceKey))
```

```
    newList.append(keys[choiceKey]); listKey.remove(choiceKey)
```

Перебор всех ключей и всех символов в списке newList, и занесение получившихся символов в переменную final.

```
for listIndex in range(len(newList)):
```

```
    for symbol in newList[listIndex]:
```

```
        final += symbol
```

Вернуть сообщение и ключи.

```
return final, ' '.join(key)
```

Если переключатель равен расшифрованию, тогда создать два списка. В первом списке все нечётные символы сообщения, во втором все чётные символы сообщения.

```
else:
```

```
    listHalf = [
```

```
        [message[x] for x in range(len(message)) if x%2 != 0],
```

```
        [message[y] for y in range(len(message)) if y%2 == 0]]
```

Указание ключей и перевод их в целые значения.

```
key = regular(input("Write the key: "))
```

```
key = [int(x) for x in key]
```

Создание словаря с ключами и списком значений ключа.

```
keys = {y:[listHalf[0][x],listHalf[1][x]] for x,y in enumerate(key)}
```

Создание двух словарей. В первом находятся все первые элементы значений словаря key. Во втором находятся все вторые элементы значений словаря key.

```
finalList = [  
[keys[x][0] for x in range(len(keys)) if x in keys],  
[keys[y][1] for y in range(len(keys)) if y in keys]]
```

Занесение в переменную final всех символов двух предыдущих списков.

```
for i in range(2):
```

```
    for index in range(len(message)//2):
```

```
        final += finalList[i][index]
```

```
return final
```

Вывод получившегося сообщения.

```
print("Final message:",encryptDecrypt(cryptMode, startMessage))
```

028. Шифр Плейфера.

(Шифр Плейфера является биграммным шифром. Особенность шифра заключена в матрице 5x5 в которой мы заменяем символы нашего сообщения. Принцип шифрования – если два символа находятся на одной строке матрицы, значит сдвинуть индексы этих двух символов на одну позицию вперёд в этой же строке. Если символы находятся на разных строках, значит прочертить прямоугольник и заменить символы на другие края прямоугольника.)

// file.py

Start_1

```
from re import findall
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = list(input("Write the message: ").upper())
matrixKey = [
    ['S','O','M','E','T'],
    ['H','I','N','G','A'],
    ['B','C','D','F','K'],
    ['L','P','Q','R','U'],
    ['V','W','X','Y','Z']
]; addSymbol = 'X'
def regular(text):
    template = r"[A-Z]{2}"
    return findall(template, text)
def encryptDecrypt(mode, message, final = ""):
    if mode == 'E':
        for symbol in message:
            if symbol not in [chr(x) for x in range(65,91)]:
                message.remove(symbol)
        for index in range(len(message)):
            if message[index] == 'J': message[index] = 'I'
        for index in range(1,len(message)):
            if message[index] == message[index - 1]:
                message.insert(index,addSymbol)
        if len(message) % 2 != 0:
            message.append(addSymbol)
    binaryList = regular("".join(message))
    for binary in range(len(binaryList)):
        binaryList[binary] = list(binaryList[binary])
        for indexString in range(len(matrixKey)):
```

```

        for indexSymbol in range(len(matrixKey[indexString])):
            if binaryList[binary][0] == matrixKey[indexString]
[indexSymbol]:
                y0, x0 = indexString, indexSymbol
                if binaryList[binary][1] == matrixKey[indexString]
[indexSymbol]:
                    y1, x1 = indexString, indexSymbol
                    for indexString in range(len(matrixKey)):
                        if matrixKey[y0][x0] in matrixKey[indexString] and matrixKey[y1]
[x1] in matrixKey[indexString]:
                            if mode == 'E':
                                x0 = x0 + 1 if x0 != 4 else 0
                                x1 = x1 + 1 if x1 != 4 else 0
                            else:
                                x0 = x0 - 1 if x0 != 0 else 4
                                x1 = x1 - 1 if x1 != 0 else 4
                    y0,y1 = y1,y0
                    binaryList[binary][0] = matrixKey[y0][x0]
                    binaryList[binary][1] = matrixKey[y1][x1]
            for binary in range(len(binaryList)):
                for symbol in binaryList[binary]:
                    final += symbol
        return final
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: SGVQSPowUPXD

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: SGVQSPowUPXD

Final message: HELXLOWORLDX

]_End

Импорт функции из модуля re.

```
from re import findall
```

Переключатель режимов шифрования.

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E', 'D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

Сообщение для шифрования/расшифрования.

```
startMessage = list(input("Write the message: ").upper())
```

Создание матрицы-ключа и символа для редактирования текста.

```
matrixKey = [
```

```
    ['S', 'O', 'M', 'E', 'T'],
```

```
    ['H', 'I', 'N', 'G', 'A'],
```

```
    ['B', 'C', 'D', 'F', 'K'],
```

```
    ['L', 'P', 'Q', 'R', 'U'],
```

```
    ['V', 'W', 'X', 'Y', 'Z']
```

```
]; addSymbol = 'X'
```

Функция возвращающая пары символов.

```
def regular(text):
```

```
    template = r"[A-Z]{2}"
```

```
    return findall(template, text)
```

Создание главной функции.

```
def encryptDecrypt(mode, message, final = ""):
```

Перебор всех символов сообщения и если символ не находится в диапазоне A-Z, то удалить этот символ.

```
if mode == 'E':
```

```
    for symbol in message:
```

```
        if symbol not in [chr(x) for x in range(65,91)]:
```

```
            message.remove(symbol)
```

Перебор сообщения и если символ равен символу 'J', то заменить его на 'I'.

```
for index in range(len(message)):
```

```
    if message[index] == 'J': message[index] = 'I'
```

Перебор символов сообщения и если стоит два одинаковых символа вместе, то между ними поставить символ 'X'.

```
for index in range(1,len(message)):
    if message[index] == message[index - 1]:
        message.insert(index,addSymbol)
```

Если длина сообщения не кратна двум, то добавить в конец сообщения символ 'X'.

```
if len(message) % 2 != 0:
    message.append(addSymbol)
```

Создание списка из пар символов.

```
binaryList = regular("".join(message))
```

Перебор всех пар символов.

```
for binary in range(len(binaryList)):
    binaryList[binary] = list(binaryList[binary])
```

Перебор всех строк и всех символов в матрице.

```
for indexString in range(len(matrixKey)):
    for indexSymbol in range(len(matrixKey[indexString])):
```

Если первый символ пары будет равняться символу из матрицы, тогда запомнить координаты первого символа пары.

```
if binaryList[binary][0] == matrixKey[indexString][indexSymbol]:
    y0, x0 = indexString, indexSymbol
```

Если второй символ пары будет равняться символу из матрицы, тогда запомнить координаты второго символа пары.

```
if binaryList[binary][1] == matrixKey[indexString][indexSymbol]:
    y1, x1 = indexString, indexSymbol
```

Перебор всех строк в матрице и если первый и второй символ находятся на одной и той же строке матрицы, тогда работает переключатель режимов шифрования.

```
for indexString in range(len(matrixKey)):
    if matrixKey[y0][x0] in matrixKey[indexString] and matrixKey[y1][x1] in
matrixKey[indexString]:
```


Если переключатель равен шифрованию, тогда перемещать позицию элементов пары по горизонтали на одну позицию вперёд. Если же позиция на которой стоит символ является последней, то позиция перемещается на нулевой элемент строки.

if mode == 'E':

x0 = x0 + 1 if x0 != 4 else 0

x1 = x1 + 1 if x1 != 4 else 0

Если переключатель равен расшифрованию, тогда перемещать позицию элементов пары по горизонтали на одну позицию назад. Если же позиция на которой стоит символ является нулевой, то позиция перемещается на последний элемент строки.

else:

x0 = x0 - 1 if x0 != 0 else 4

x1 = x1 - 1 if x1 != 0 else 4

Две переменные меняют значения между собой.

y0,y1 = y1,y0

Замена позиций пары в матрице.

binaryList[binary][0] = matrixKey[y0][x0]

binaryList[binary][1] = matrixKey[y1][x1]

Перебор всех пар и посимвольный ввод в переменную final.

for binary in range(len(binaryList)):

for symbol in binaryList[binary]:

final += symbol

Вернуть сообщения.

return final

Вывод получившегося сообщения.

print("Final message:",encryptDecrypt(cryptMode, startMessage))

029. Шифр ADFGVX.

(Шифр ADFGVX состоит из шифров замены и шифров перестановки. Шифр замены – аналогия с шифром Полибия. Шифр перестановки работает по ключу.)

// file.py

Start_

```
from re import findall
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
keyStageTwo = list(input("Write the key: ").upper())
for symbol in keyStageTwo:
    if keyStageTwo.count(symbol) > 1:
        keyStageTwo.remove(symbol)
keyStageOne = {
    'A':'AA','N':'FD','O':'VF',
    'B':'AD','O':'FF','1':'VG',
    'C':'AF','P':'FG','2':'VV',
    'D':'AG','Q':'FV','3':'VX',
    'E':'AV','R':'FX','4':'XA',
    'F':'AX','S':'GA','5':'XD',
    'G':'DA','T':'GD','6':'XF',
    'H':'DD','U':'GF','7':'XG',
    'I':'DF','V':'GG','8':'XV',
    'J':'DG','W':'GV','9':'XX',
    'K':'DV','X':'GX',
    'L':'DX','Y':'VA',
    'M':'FA','Z':'VD',
}
def regular(text):
    template = r"[A-Z]{2}"
    return findall(template, text)
def stageOne(mode, message, final = ""):
    if mode == 'E':
        for symbol in message:
            if symbol in keyStageOne:
                final += keyStageOne[symbol]
    else:
        for symbols in regular(message):
            for key in keyStageOne:
```

```

        if symbols == keyStageOne[key]:
            final += key
    return final
def stageTwo(mode, message, final = "", listCutWords = []):
    if mode == 'E':
        while len(message) % len(keyStageTwo) != 0:
            message += 'XX'
        lengthList = len(message) // len(keyStageTwo)
        for _ in range(lengthList):
            listCutWords.append([])
        index = 0; counter = 1
        for symbol in message:
            if counter % len(keyStageTwo) != 0:
                listCutWords[index].append(symbol)
                counter += 1
            else:
                listCutWords[index].append(symbol)
                index += 1; counter = 1
        keys = {x:[] for x in keyStageTwo}
        index = 0
        for key in keyStageTwo:
            for x in range(len(listCutWords)):
                keys[key].append(listCutWords[x][index])
            index += 1
        keySort = list(keyStageTwo); keySort.sort()
        keys = {key:keys[key] for key in keySort if key in keys}
        for listSymbol in keys:
            for symbol in keys[listSymbol]:
                final += symbol
    else:
        keySort = list(keyStageTwo); keySort.sort()
        lengthList = len(message) // len(keyStageTwo)
        for _ in range(len(keyStageTwo)):
            listCutWords.append([])
        index = 0; counter = 1
        for symbol in message:
            if counter % lengthList != 0:
                listCutWords[index].append(symbol)
                counter += 1
            else:
                listCutWords[index].append(symbol)
                index += 1; counter = 1

```

```

        keys = {keySort[symbol]:listCutWords[symbol] for symbol in
range(len(keySort))}
        keys = {key:keys[key] for key in keyStageTwo if key in keys}
        index = 0
        for _ in range(lengthList):
            for symbolOne in keys:
                final += keys[symbolOne][index]
            index += 1
        return final
def encryptDecrypt(mode, message):
    if mode == 'E':
        message = stageOne(mode, message)
        message = stageTwo(mode, message)
    else:
        message = stageTwo(mode, message)
        message = stageOne(mode, message)
    return message
print("Final message:",encryptDecrypt(cryptMode, startMessage))
]_End

```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Write the key: delta

Final message: DFFGDYGXDDVDAXFXVFFA

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: DFFGDYGXDDVDAXFXVFFA

Write the key: delta

Final message: HELLOWORLD

]_End

Импорт функции из модуля re.

from re import findall

Переключатель режимов шифрования.

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

```

if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
# Сообщение для шифрования/расшифрования.
startMessage = input("Write the message: ").upper()

# Создание ключа.
keyStageTwo = list(input("Write the key: ").upper())

# Посимвольный перебор ключа и если символ в ключе встречается больше одного
# раза, тогда удалить этот символ.
for symbol in keyStageTwo:
    if keyStageTwo.count(symbol) > 1:
        keyStageTwo.remove(symbol)

# Создание словаря для шифрования (шифр Полибия).
keyStageOne = {
    'A':'AA','N':'FD','0':'VF',
    'B':'AD','O':'FF','1':'VG',
    'C':'AF','P':'FG','2':'VV',
    'D':'AG','Q':'FV','3':'VX',
    'E':'AV','R':'FX','4':'XA',
    'F':'AX','S':'GA','5':'XD',
    'G':'DA','T':'GD','6':'XF',
    'H':'DD','U':'GF','7':'XG',
    'I':'DF','V':'GG','8':'XV',
    'J':'DG','W':'GV','9':'XX',
    'K':'DV','X':'GX',
    'L':'DX','Y':'VA',
    'M':'FA','Z':'VD',
}

# Функция возвращающая пары символов.
def regular(text):
    template = r"[A-Z]{2}"
    return findall(template, text)

# Функция stageOne является аналогом шифра Полибия.
def stageOne(mode, message, final = ""):
    if mode == 'E':
        for symbol in message:
            if symbol in keyStageOne:
                final += keyStageOne[symbol]

```

```

else:
    for symbols in regular(message):
        for key in keyStageOne:
            if symbols == keyStageOne[key]:
                final += key
    return final

```

Основная функция шифрования.

```
def stageTwo(mode, message, final = "", listCutWords = []):
```

Если переключатель равен шифрованию, тогда сделать цикл, при котором будут добавляться символы 'XX' пока длина сообщения не будет делиться на длину ключа без остатка.

```

if mode == 'E':
    while len(message) % len(keyStageTwo) != 0:
        message += 'XX'

```

Длина полного деления сообщения на ключ.

```
lengthList = len(message) // len(keyStageTwo)
```

Добавление в список вложенных списков.

```

for _ in range(lengthList):
    listCutWords.append([])

```

Добавление во вложенные списки символов по длине ключа.

```

index = 0; counter = 1
for symbol in message:
    if counter % len(keyStageTwo) != 0:
        listCutWords[index].append(symbol)
        counter += 1
    else:
        listCutWords[index].append(symbol)
        index += 1; counter = 1

```

Создание словаря.

```
keys = {x:[] for x in keyStageTwo}
```

Добавление в словарь на место значений ключа символов.

```

index = 0
for key in keyStageTwo:
    for x in range(len(listCutWords)):
        keys[key].append(listCutWords[x][index])

```

index += 1

Сортировка ключа по алфавиту.

keySort = list(keyStageTwo); keySort.sort()

keys = {key:keys[key] for key in keySort if key in keys}

Перебор всех всех символов значений ключа и добавление их к переменной final.

for listSymbol in keys:

for symbol in keys[listSymbol]:

final += symbol

Если же переключатель равен расшифрованию, тогда отсортировать ключи и вычислить длину вложенных списков.

else:

keySort = list(keyStageTwo); keySort.sort()

lengthList = len(message) // len(keyStageTwo)

Создание вложенных списков.

for _ in range(len(keyStageTwo)):

listCutWords.append([])

Добавление во вложенные списки символов по длине ключа.

index = 0; counter = 1

for symbol in message:

if counter % lengthList != 0:

listCutWords[index].append(symbol)

counter += 1

else:

listCutWords[index].append(symbol)

index += 1; counter = 1

Создание ключей и их отсортировка.

keys = {keySort[symbol]:listCutWords[symbol] for symbol in range(len(keySort))}

keys = {key:keys[key] for key in keyStageTwo if key in keys}

Перебор всех ключей и добавление в переменную final всех символов.

index = 0

for _ in range(lengthList):

for symbolOne in keys:

final += keys[symbolOne][index]

index += 1

Вернуть сообщение.

return final

Главная функция, которая является переключателем режимов шифрования.

def encryptDecrypt(mode, message):

if mode == 'E':

 message = stageOne(mode, message)

 message = stageTwo(mode, message)

else:

 message = stageTwo(mode, message)

 message = stageOne(mode, message)

return message

Вывод получившегося сообщения.

print("Final message:", encryptDecrypt(cryptMode, startMessage))

030. AES шифрование.

(AES шифрование позволяет зашифровывать файлы любого формата. В данном примере я не расписывал скрипт с нуля, а установил готовый модуль pyAesCrypt. Библиотека pyAesCrypt использует AES256-CBC.

Осторожно: Файл перезаписывается!)

// file.py

Start_

```
from pyAesCrypt import encryptFile, decryptFile
from os import remove
from os.path import splitext
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
fileName = input("Write the file: ")
paswFile = input("Write the password: ")
bufferSize = 64*1024
def encryptDecrypt(mode, file, password, final = ""):
    if mode == 'E':
        try:
            encryptFile(str(file), str(file)+".crp", password, bufferSize)
            remove(file)
        except FileNotFoundError: return "[x] File not found!"
        else: return "[+] File '{name}' overwritten!".format(name = str(file))
    else:
        try:
            decryptFile(str(file), str(splitext(file)[0]), password, bufferSize)
            remove(file)
        except FileNotFoundError: return "[x] File not found!"
        except ValueError: return "[x] Password is False!"
        else: return "[+] File '{name}' overwritten!".format(name = str(file))
print(encryptDecrypt(cryptMode, fileName, paswFile))
_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the file: book.txt

Write the password: helloworld

[+] File 'book.txt' overwritten!

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the file: book.txt.crp

Write the password: helloworld

[+] File 'book.txt.crp' overwritten!

_End

Импортирование функций из моделей aes и os.

from pyAesCrypt import encryptFile, decryptFile

from os import remove

from os.path import splitext

Переключатель режимов шифрования.

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E','D']:

print("Error: mode is not Found!"); raise SystemExit

Имя файла и пароль для шифрования/расшифрования.

fileName = input("Write the file: ")

paswFile = input("Write the password: ")

Размер шифрования/расшифрования данных.

bufferSize = 64*1024

Главная функция.

def encryptDecrypt(mode, file, password, final = ""):

Если переключатель равен шифрованию, тогда попытаться зашифровать файл и удалить открытую копию. Если же файл не был найден – вывести ошибку. Но если же всё удачно прошло, тогда написать сообщение о перезаписи файла.

if mode == 'E':

try:

encryptFile(str(file), str(file)+".crp", password, bufferSize)
remove(file)

except FileNotFoundError: return "[x] File not found!"

else: return "[+] File '{name}' overwritten!".format(name = str(file))

Если же переключатель равен расшифрованию, тогда попытаться расшифровать файл и удалить зашифрованную копию. Если же файл не был найден или пароль не верный – вывести ошибку. Но если же всё удачно прошло, тогда написать сообщение о перезаписи файла.

else:

try:

decryptFile(str(file), str(splitext(file)[0]), password, bufferSize)
remove(file)

except FileNotFoundError: return "[x] File not found!"

except ValueError: return "[x] Password is False!"

else: return "[+] File '{name}' overwritten!".format(name = str(file))

Вывод результата.

print(encryptDecrypt(cryptMode, fileName, paswFile))

031. Великий Шифр.

(Великий Шифр собрал в себя пять различных методов шифрования: омофоническое шифрование, кодирование, замена слогов, ложные символы, специальные символы (символы редактирования текста). Шифр является одним из самых криптостойких.)

// file.py

Start_

```
from memory import Key, Limit
```

```
from random import randint, choice
```

```
from re import findall
```

```
keysCrypt = {
```

```
    'A':Key[0:8],      'B':Key[8:10],
```

```
    'C':Key[10:13],   'D':Key[13:17],
```

```
    'E':Key[17:29],   'F':Key[29:31],
```

```
    'G':Key[31:33],   'H':Key[33:39],
```

```
    'T':Key[39:45],   'J':Key[45],
```

```
    'K':Key[46],      'L':Key[47:51],
```

```
    'M':Key[51:53],   'N':Key[53:59],
```

```
    'O':Key[59:66],   'P':Key[66:68],
```

```
    'Q':Key[68],      'R':Key[69:75],
```

```
    'S':Key[75:81],   'T':Key[81:90],
```

```
    'U':Key[90:93],   'V':Key[93],
```

```
    'W':Key[94:96],   'X':Key[96],
```

```
    'Y':Key[97:99],   'Z':Key[99],
```

```
    ' ':Key[100:118]
```

```
}
```

```
listWord = ('TO','WHY','WITH','WAR','NOT','IN','OR','ELSE','THE','THAT','BY',  
'AND','HOW','BUT','IF','ONE','YOU','ME','USE','HIS','YOUR','ON','OF','WAS','BE',  
'THIS','WHAT','THEY','NO','YES','TRUE','FALSE','CALL','FEEL','CLOSE','VERY',  
'WHICH','CAR','ANY','HOLD','WORK','RUN','NEVER','START','EVEN','LIGHT','TH  
AN','AFTER','PUT','STOP','OLD','WATCH','FIRST','MAY','TALK','ANOTHER','BEHI  
ND','CUT','MEAN','SMILE','OUR','MUCH','IT','HE','SHE','ITS','HOUSE','KEEP','YEA  
H','PLACE','BEGIN','NOTHING','YEAR','MAN','WOMAN','BECAUSE','THREE','SEE  
M','ARE','WAIT','NEED','LAST','LATE','SURE','BIG','SMALL','FRONT','REALLY','N  
AME','ALL','NEW','GUY','ANYTHING','SHOULD','KILL','POINT','WALL','BLACK',  
'STEP','SECOND','LIFE','MAYBE','FALL','OWN','FAR','WHILE','FOR','HELP','END',  
'THOSE','SAME','REACH','GIRL','STREET','NEXT','FEW','FEET','SHOW','MUST','T  
ABLE','OK','IS','OKAY','BODY','PHONE','ADD','WATER','FIRE','INSIDE','BREAK','E  
VER','SHAKE','MEET','GREAT','MIND','ENOUGH','MINUTE','FOLLOW','ATTACK',  
'DEAD','ALMOST')
```

```

position = 118
keysCode = {listWord[x]:Key[x + position] for x in range(len(listWord))}
listSyllables = ('TH','WH','EE','AI','OO','IS','ING','ED','BE','ON','OR','ER',
'CH','SH','GH','EN','EA','OU','LL','US','SE','AL','ST','EV','WO','UI','IN','RE',
'!','?','!','!','!','@','#','$','%','*','^','-','+','=','/',':',';','&','~')
position = len(listWord) + 118
keysSyllables = {listSyllables[x]:Key[x + position] for x in range(len(listSyllables))}
listSpecial = ('<-','->','<+','+>')
position = len(listWord) + len(listSyllables) + 118
keysSpecial = {listSpecial[x]:Key[x + position] for x in range(len(listSpecial))}
position = len(listWord) + len(listSyllables) + len(listSpecial) + 118
traps = tuple([Key[x] for x in range(position, Limit)])
del listWord, listSpecial, position
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
def regular(text):
    template = r"[0-9]{3}"
    return findall(template, text)
def encryptDecrypt(mode, message, final = "", string = ""):
    if cryptMode == 'E':
        secondText = findall(r"[\s]+", message)
        del message
        for indexWord in range(len(secondText)):
            if secondText[indexWord] in keysSpecial:
                secondText[indexWord] =
keysSpecial[secondText[indexWord]]
            for indexWord in range(len(secondText)):
                if secondText[indexWord] in keysCode:
                    secondText[indexWord] = keysCode[secondText[indexWord]]
            for indexWord in range(len(secondText)):
                for syllable in keysSyllables:
                    if syllable in secondText[indexWord]:
                        secondText[indexWord] =
secondText[indexWord].replace(syllable,keysSyllables[syllable])
            for indexWord in range(len(secondText)):
                secondText[indexWord] = list(secondText[indexWord])
            for indexWord in range(len(secondText)):
                secondText[indexWord].append(' ')
            for indexWord in range(len(secondText)):
                for indexSymbol in range(len(secondText[indexWord]]):

```

```

        symbol = secondText[indexWord][indexSymbol]
        if symbol in keysCrypt:
            length = len(keysCrypt[symbol])
            secondText[indexWord][indexSymbol] =
keysCrypt[symbol][randint(0, length - 1)]
        for word in secondText:
            string += "".join(word)
        finalList = list(regular(string))
        for indexList in range(len(finalList)):
            randSwitch = randint(0,2); randPosition = randint(0,len(finalList))
            if not randSwitch: finalList.insert(randPosition,choice(traps))
        for word in finalList:
            final += "".join(word)
        return ".".join(regular(final))
    else:
        for symbolText in regular(message):
            for element in keysSpecial:
                if symbolText == keysSpecial[element]: final += element
            for word in keysCode:
                if symbolText == keysCode[word]: final += word
            for syllable in keysSyllables:
                if symbolText == keysSyllables[syllable]: final += syllable
            for symbol in keysCrypt:
                if symbolText in keysCrypt[symbol]: final += symbol
        listWord = findall(r"[^\s]+",final)
        for _ in range(len(listWord)):
            for element in keysSpecial:
                if element in listWord:
                    if element == '<-':
                        del listWord[listWord.index(element) - 1]
                        listWord.remove(element)
                    elif element == '->':
                        del listWord[listWord.index(element) + 1]
                        listWord.remove(element)
                    elif element == '<+':
                        listWord[listWord.index(element)] =
listWord[listWord.index(element) - 1]
                    elif element == '+>':
                        listWord[listWord.index(element)] =
listWord[listWord.index(element) + 1]
                    else: pass
        return " ".join(listWord)

```

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))  
]_End
```

// memory.py

Start_[

```
Limit = 350; Key = (  
'275','078','276','230','343','127','006','325','307','102',  
'334','185','004','002','008','283','277','260','256','305',  
'300','143','159','248','160','309','104','222','136','317',  
'264','053','218','137','177','315','301','244','040','072',  
'023','182','323','154','076','048','213','330','109','164',  
'257','179','043','166','207','240','220','205','228','073',  
'017','093','186','027','157','080','009','195','289','278',  
'116','238','028','271','058','001','132','236','044','029',  
'282','339','131','069','100','107','255','135','226','198',  
'335','035','187','119','293','133','270','347','097','273',  
'068','250','311','140','212','120','024','303','188','103',  
'209','114','034','267','061','184','261','225','349','167',  
'059','106','272','168','326','165','269','279','019','247',  
'087','156','239','088','092','026','071','304','241','180',  
'306','284','310','065','021','234','162','202','060','047',  
'039','318','321','152','117','324','067','348','031','308',  
'265','070','181','077','217','041','032','124','125','290',  
'235','037','231','003','342','158','020','322','216','091',  
'176','079','199','259','129','138','113','099','312','215',  
'115','145','122','193','254','314','153','344','123','340',  
'292','101','083','286','262','183','121','237','242','249',  
'246','139','018','042','295','178','245','022','148','163',  
'052','233','189','229','090','000','341','015','346','336',  
'082','345','281','036','005','089','038','064','141','046',  
'151','062','266','243','010','287','066','149','030','095',  
'147','171','173','332','055','045','253','298','227','299',  
'051','170','331','280','224','011','111','105','112','130',  
'192','268','169','128','033','098','075','194','211','085',  
'223','025','296','196','155','320','054','126','174','108',  
'118','144','172','302','146','294','190','012','210','086',  
'191','014','208','142','084','337','203','219','201','327',  
'197','057','221','016','338','206','333','134','150','063',  
'251','291','313','285','204','274','319','013','056','328',  
'214','263','200','096','288','007','252','161','050','110',  
'329','049','258','081','094','074','297','316','232','175',
```

```
)  
]_End
```

// terminal

Start_[

```
$ python file.py
```

```
[E]ncrypt[D]ecrypt: e
```

```
Write the message: -> hello world <+
```

```
Final message: 014.250.177.260.169.093.184.133.333.011.109.277.120.208.291.225
```

```
$ python file.py
```

```
[E]ncrypt[D]ecrypt: d
```

```
Write the message: 014.250.177.260.169.093.184.133.333.011.109.277.120.208.291.225
```

```
Final message: WORLD WORLD
```

]_End

```
# Импортирование функций из модуля random и re. Импортирование переменных  
из модуля memory.
```

```
from memory import Key, Limit
```

```
from random import randint, choice
```

```
from re import findall
```

```
# Омофоническое шифрование.
```

```
keysCrypt = {
```

```
    'A':Key[0:8],    'B':Key[8:10],  
    'C':Key[10:13], 'D':Key[13:17],  
    'E':Key[17:29], 'F':Key[29:31],  
    'G':Key[31:33], 'H':Key[33:39],  
    'I':Key[39:45], 'J':Key[45:],  
    'K':Key[46:],    'L':Key[47:51],  
    'M':Key[51:53], 'N':Key[53:59],  
    'O':Key[59:66], 'P':Key[66:68],  
    'Q':Key[68:],    'R':Key[69:75],  
    'S':Key[75:81], 'T':Key[81:90],  
    'U':Key[90:93], 'V':Key[93:],  
    'W':Key[94:96], 'X':Key[96:],  
    'Y':Key[97:99], 'Z':Key[99:],  
    ' ':Key[100:118]
```

```
}
```


Кодирование.

listWord =

('TO','WHY','WITH','WAR','NOT','IN','OR','ELSE','THE','THAT','BY','AND','
HOW','BUT','IF','ONE','YOU','ME','USE','HIS','YOUR','ON','OF','WAS','BE','
THIS','WHAT','THEY','NO','YES','TRUE','FALSE','CALL','FEEL','CLOSE','V
ERY','WHICH','CAR','ANY','HOLD','WORK','RUN','NEVER','START','EVEN'
, 'LIGHT','THAN','AFTER','PUT','STOP','OLD','WATCH','FIRST','MAY','TAL
K','ANOTHER','BEHIND','CUT','MEAN','SMILE','OUR','MUCH','IT','HE','SH
E','ITS','HOUSE','KEEP','YEAH','PLACE','BEGIN','NOTHING','YEAR','MAN'
, 'WOMAN','BECAUSE','THREE','SEEM','ARE','WAIT','NEED','LAST','LATE',
'SURE','BIG','SMALL','FRONT','REALLY','NAME','ALL','NEW','GUY','ANY
THING','SHOULD','KILL','POINT','WALL','BLACK','STEP','SECOND','LIFE'
, 'MAYBE','FALL','OWN','FAR','WHILE','FOR','HELP','END','THOSE','SAME'
, 'REACH','GIRL','STREET','NEXT','FEW','FEET','SHOW','MUST','TABLE',
'OK','IS','OKAY','BODY','PHONE','ADD','WATER','FIRE','INSIDE','BREAK',
EVER','SHAKE','MEET','GREAT','MIND','ENOUGH','MINUTE','FOLLOW',
ATTACK','DEAD','ALMOST')

Стартовая позиция (конец шифра с омофонами).

position = 118

Создание словаря с + позицией.

keysCode = {listWord[x]:Key[x + position] for x in range(len(listWord))}

Шифр замены/шифр с заменой слогов.

**listSyllables = ('TH','WH','EE','AI','OO','IS','ING','ED','BE','ON','OR','ER',
'CH','SH','GH','EN','EA','OU','LL','US','SE','AL','ST','EV','WO','UI','IN','RE',
'!','?','.',',','@','#','\$','%','*','^','-',',','+','=','/',':',';','&','~')**

Стартовая позиция (конец кодирования).

position = len(listWord) + 118

Создание словаря с + позицией.

keysSyllables = {listSyllables[x]:Key[x + position] for x in range(len(listSyllables))}

Создание специальных символов (для редактирования текста).

ListSpecial = ('<-','->','<+','+>')

Стартовая позиция (конец шифра замены).

position = len(listWord) + len(listSyllables) + 118

Создание словаря с + позицией.

```
keysSpecial = {listSpecial[x]:Key[x + position] for x in range(len(listSpecial))}
```

Стартовая позиция (конец специальных символов).

```
position = len(listWord) + len(listSyllables) + len(listSpecial) + 118
```

Создание символов-ловушек на все оставшиеся не занятые элементы памяти.

```
traps = tuple([Key[x] for x in range(position, Limit)])
```

Удалить ненужные элементы.

```
del listWord, listSpecial, position
```

Переключатель режимов шифрования.

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

Сообщение для шифрования/расшифрования.

```
startMessage = input("Write the message: ").upper()
```

Функция возвращающая триады чисел.

```
def regular(text):
```

```
    template = r"[0-9]{3}"
```

```
    return findall(template, text)
```

Главная функция.

```
def encryptDecrypt(mode, message, final = "", string = ""):
```

Если переключатель равен шифрованию, тогда разделить сообщение по словам.

```
if cryptMode == 'E':
```

```
    secondText = findall(r"[^\s]+", message)
```

Удалить ненужные элементы.

```
del message
```

Внедрение в код специальных символов.

```
for indexWord in range(len(secondText)):
```

```
    if secondText[indexWord] in keysSpecial:
```

```
        secondText[indexWord] = keysSpecial[secondText[indexWord]]
```

Кодирование слов на числа.

```
for indexWord in range(len(secondText)):
    if secondText[indexWord] in keysCode:
        secondText[indexWord] = keysCode[secondText[indexWord]]
```

Замена слогов на числа.

```
for indexWord in range(len(secondText)):
    for syllable in keysSyllables:
        if syllable in secondText[indexWord]:
            secondText[indexWord] =
secondText[indexWord].replace(syllable,keysSyllables[syllable])
```

Разделение всех слов на символы.

```
for indexWord in range(len(secondText)):
    secondText[indexWord] = list(secondText[indexWord])
```

Добавление в конец каждого списка пробел.

```
for indexWord in range(len(secondText)):
    secondText[indexWord].append(' ')
```

Шифрование всех символов на числа при помощи омофоноф.

```
for indexWord in range(len(secondText)):
    for indexSymbol in range(len(secondText[indexWord])):
        symbol = secondText[indexWord][indexSymbol]
        if symbol in keysCrypt:
            length = len(keysCrypt[symbol])
            secondText[indexWord][indexSymbol] = keysCrypt[symbol]
[randint(0, length - 1)]
```

Соединение всех числе в одну строку.

```
for word in secondText:
    string += " ".join(word)
```

Разделение чисел по триадам.

```
finalList = list(regular(string))
```

Внедрение в сообщение ложных символов.

```
for indexList in range(len(finalList)):
    randSwitch = randint(0,2); randPosition = randint(0,len(finalList))
    if not randSwitch: finalList.insert(randPosition,choice(traps))
```

Занесение в переменную final зашифрованный текст.

```
for word in finalList:  
    final += "".join(word)
```

Вернуть зашифрованное сообщение.

```
return ".".join(regular(final))
```

Если переключатель равен расшифрованию, тогда сделать перебор сообщения через функцию, возвращающая триады чисел.

```
else:  
    for symbolText in regular(message):
```

Перебор всех специальных символов.

```
for element in keysSpecial:  
    if symbolText == keysSpecial[element]: final += element
```

Перебор всех кодов.

```
for word in keysCode:  
    if symbolText == keysCode[word]: final += word
```

Перебор всех символов и слогов шифра.

```
for syllable in keysSyllables:  
    if symbolText == keysSyllables[syllable]: final += syllable
```

Перебор всех омофонов.

```
for symbol in keysCrypt:  
    if symbolText in keysCrypt[symbol]: final += symbol
```

Разделить слова между собой в список.

```
listWord = findall(r"[\s]+", final)
```

Перебор списка слов.

```
for _ in range(len(listWord)):
```

Если специальный символ будет являться словом в списке, тогда сделать действия.

```
for element in keysSpecial:  
    if element in listWord:
```

Если найдено слово '<-', тогда удалить предыдущее слово и этот символ.

```
if element == '<-':  
    del listWord[listWord.index(element) - 1]  
    listWord.remove(element)
```

Если найдено слово '->', тогда удалить последующее слово и этот символ.

```
elif element == '->':  
    del listWord[listWord.index(element) + 1]  
    listWord.remove(element)
```

Если найдено слово '<+', тогда продублировать предыдущее слово заменив этот символ дублированным словом.

```
elif element == '<+':  
    listWord[listWord.index(element)] = listWord[listWord.index(element) - 1]
```

Если найдено слово '+>', тогда продублировать последующее слово заменив этот символ дублированным словом.

```
elif element == '+>':  
    listWord[listWord.index(element)] = listWord[listWord.index(element) + 1]  
else: pass
```

Вернуть расшифрованное сообщение.

```
return " ".join(listWord)
```

Вывод результата.

```
print("Final message:", encryptDecrypt(cryptMode, startMessage))
```

032. RSA шифрование.

(RSA является асимметричным методом шифрования. В данном примере я не расписывал скрипт с нуля, а установил готовый модуль rsa.)

// file.py

Start_

```
from rsa import newkeys
pub, priv = newkeys(1024)
print(str(pub) + "\n\n" + str(priv))
fileEncrypt = "encryptRSA.py"
fileDecrypt = "decryptRSA.py"
with open(fileEncrypt, "w") as encryptScript:
    encryptScript.write("""
from rsa import encrypt, PublicKey
message = input("Write the message: ").encode("utf8")
fileName = "encryptMessage.txt"
encryptMessage = encrypt(message, ""+str(pub)+"")
with open(fileName, "wb") as file:
    file.write(encryptMessage)
    print("Encrypt message:\n{message}\n[+] File: '{name}' successfully
saved!".format(message = str(encryptMessage), name = fileName))
""")
    print("[+] File: '{name}' successfully saved!".format(name = fileEncrypt))
with open(fileDecrypt, "w") as decryptScript:
    decryptScript.write("""
from rsa import decrypt, PrivateKey
fileName = input("Write the filename: ")
with open(fileName, "rb") as file:
    decryptMessage = decrypt(file.read(), ""+str(priv)+"")
    print("Decrypted message:", str(decryptMessage.decode("utf8")))
""")
    print("[+] File: '{name}' successfully saved!".format(name = fileDecrypt))
]_End
```

// terminal

Start_

\$ python file.py

PublicKey(11771105794757592845646068472184849673904233810164636440170937
94284982013380942638204590120306097937179449331728392551603116715566944
80039811535960917598398562389338987778280064829139819522521772044574277
41136679863841512439390464182934066451466047958923845070829576308634826
6970724862588375182544212130104851, 65537)

PrivateKey(1177110579475759284564606847218484967390423381016463644017093
79428498201338094263820459012030609793717944933172839255160311671556694
48003981153596091759839856238933898777828006482913981952252177204457427
74113667986384151243939046418293406645146604795892384507082957630863482
66970724862588375182544212130104851, 65537,
14422689401697281009282989735820123423631078245208360256736291206659452
77850522511677809278126549343966153185800233789977116319941798792620484
05235490911123269709217612610807517347146060130464443497456812718443107
52679428764593835461429891497872537100737917243749225085096967756594546
26541770110084822970073,
46616629005215852188334700013363750449465185302750313331009524641445141
13770394561856730062023434361862218206533265343459798363260211684442953
5204470188733828475791,
25250873016666530376601557713475906103023184986294875200452371794500410
82721690833580449382223608585578004085530873315454398540590680971398777
661)

[+] File: 'encryptRSA.py' successfully saved!

[+] File: 'decryptRSA.py' successfully saved!

\$ python encryptRSA.py

Write the message: helloworld

Encrypt message:

b'2gu\xcf\n\x18\x01\xa3\xba\xc3L\xc3lVzu,z7\xa0\xf7\x13\x92k2\x8fU<\xa8\xb6\x16\x
9fS\xd8-\xca\x0cakJ*\x81>\x0c\x9c\x86\xc2^\x18\xb9r\xce0\x82o\x0f\xdf;m\x15G\xcd\
xcf\xda\xa54\xab\xe1T\x83\n\\LG\xb4\xd1\x08^Y\xdf\\ \x94\x10\xca\x81\x93<\xad\xfb
H\xf1\xcdf0v\xbe\x9c(\xc6P\xac\xc9\x87\xfcOm\xbd\xa9\x83\x1f\xde=\x95<kY\x10\x
a1\xd35\xb1D\xcd\xfb.\x96\xc4'

[+] File: 'encryptMessage.txt' successfully saved!

\$ python decryptRSA.py

Write the filename: encryptMessage.txt

Decrypted message: helloworld

End

```
# Импорт функции из модуля rsa.
from rsa import newkeys

# Создание публичного и приватного ключей.
pub, priv = newkeys(1024)
print(str(pub) + "\n\n" + str(priv))

# Имена файлов для шифрования/расшифрования.
fileEncrypt = "encryptRSA.py"
fileDecrypt = "decryptRSA.py"

# Создание файла для шифрования сообщений.
with open(fileEncrypt, "w") as encryptScript:
    encryptScript.write(''

# Импорт функций из модуля rsa.
from rsa import encrypt, PublicKey

# Сообщение для шифрования.
message = input("Write the message: ").encode("utf8")

# Файл с зашифрованным сообщением.
fileName = "encryptMessage.txt"

# Шифрование сообщения.
encryptMessage = encrypt(message, ""+str(pub)+"")

# Создание файла с зашифрованным сообщением.
with open(fileName, "wb") as file:
    file.write(encryptMessage)
    print("Encrypt message:\n{message}\n[+] File: '{name}' successfully
saved!".format(message = str(encryptMessage), name = fileName))
    print("[+] File: '{name}' successfully saved!".format(name = fileEncrypt))

# Создание файла для расшифрования сообщений.
with open(fileDecrypt, "w") as decryptScript:
    decryptScript.write(''

# Импорт функций из модуля rsa.
from rsa import decrypt, PrivateKey
```


Имя файла с зашифрованным текстом.

fileName = **input**("Write the filename: ")

Открытие файла на чтение и его расшифровка.

with open(**fileName**, "**rb**") **as file**:

decryptMessage = **decrypt**(**file.read**(), **str(priv)**)

print("Decrypted message:", **str**(**decryptMessage.decode**("utf8")))

'')

print("[+] File: '{name}' successfully saved!".**format**(**name** = **fileDecrypt**))

033. Аффинный шифр.

(Аффинный шифр основан на шифре Цезаря. Особенность этого метода шифрования заключена в модульной арифметике.)

// file.py

Start_

```
print("Possible Key[a]: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.")
```

```
cryptMode = input("[E]ncrypt[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E','D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
startMessage = input("Write the message: ").upper()
```

```
mainKey = input("Write the keys: ").split()
```

```
for key in mainKey:
```

```
    try: int(key)
```

```
    except: print("Only int numbers!"); raise SystemExit
```

```
if len(mainKey) != 2:
```

```
    print("Error: quality keys must be 2"); raise SystemExit
```

```
def encryptDecrypt(mode, message, key, final = ""):
```

```
    for symbol in message:
```

```
        if mode == 'E':
```

```
            final += chr((int(key[0]) * ord(symbol) + int(key[1]) - 13)%26 +
```

```
ord('A'))
```

```
        else:
```

```
            final += chr(pow(int(key[0]),11) * ((ord(symbol) + 26 - int(key[1]) -
```

```
13))%26 + ord('A'))
```

```
    return final
```

```
print("Final message:",encryptDecrypt(cryptMode, startMessage, mainKey))
```

_End

// terminal

Start_

```
$ python file.py
```

```
Possible Key[a]: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.
```

```
[E]ncrypt[D]ecrypt: e
```

```
Write the message: helloworld
```

```
Write the keys: 11 8
```

```
Final message: HAZZGQGNZP
```

```
$ python file.py
Possible Key[a]: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.
[E]ncrypt|[D]ecrypt: d
Write the message: HAZZGQGNZP
Write the keys: 11 8
Final message: HELLOWORLD
```

]_End

```
# Возможные комбинации первого ключа.
```

```
print("Possible Key[a]: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25.")
```

```
# Переключатель режимов шифрования.
```

```
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
```

```
if cryptMode not in ['E', 'D']:
```

```
    print("Error: mode is not Found!"); raise SystemExit
```

```
# Сообщение для шифрования/расшифрования.
```

```
startMessage = input("Write the message: ").upper()
```

```
# Ввод ключей.
```

```
mainKey = input("Write the keys: ").split()
```

```
Если один из ключей не является целым числом, тогда вывести ошибку.
```

```
for key in mainKey:
```

```
    try: int(key)
```

```
    except: print("Only int numbers!"); raise SystemExit
```

```
# Если ключей больше или меньше двух – вывести ошибку.
```

```
if len(mainKey) != 2:
```

```
    print("Error: quality keys must be 2"); raise SystemExit
```

```
# Главная функция.
```

```
def encryptDecrypt(mode, message, key, final = ""):
```

```
# Посимвольный перебор символов сообщения.
```

```
for symbol in message:
```

```
# Если переключатель равен 'E', тогда зашифровать сообщение при помощи двух ключей.
```

```
if mode == 'E':
```

```
    final += chr((int(key[0]) * ord(symbol) + int(key[1]) - 13)%26 + ord('A'))
```

Если переключатель равен 'D', тогда расшифровать сообщение с двумя ключами.

else:

final += chr(pow(int(key[0]),11) * ((ord(symbol) + 26 - int(key[1]) - 13))%26 + ord('A'))

Вернуть сообщение.

return final

Вывод сообщения.

print("Final message:",encryptDecrypt(cryptMode, startMessage, mainKey))

034. Шифр Хилла.

(Шифр Хилла основан на линейной алгебре и модульной арифметике. В данном примере описана матрица 2x2.)

// file.py

Start_

```
from re import findall
matrixLength = 2
MatrixKey = [[15,4],[11,3]]
det = MatrixKey[0][0]*MatrixKey[1][1] - MatrixKey[0][1]*MatrixKey[1][0]
if det != 1:
    print("Error: determinant != 1"); raise SystemExit
iMatrixKey = [
    [MatrixKey[1][1],-MatrixKey[0][1]],
    [-MatrixKey[1][0],MatrixKey[0][0]]
]
alpha = tuple("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()
if cryptMode not in ['E','D']:
    print("Error: mode is not Found!"); raise SystemExit
startMessage = input("Write the message: ").upper()
for symbol in startMessage:
    if symbol not in [chr(x) for x in range(65,91)]:
        startMessage = startMessage.replace(symbol,"")
while len(startMessage) % matrixLength != 0: startMessage += 'Z'
def regular(text):
    template = r"[A-Z]{" + str(matrixLength) + "}"
    return findall(template, text)
def encryptDecrypt(message, matrix, summ = 0, final = ""):
    for double in range(len(message)):
        for string in range(matrixLength):
            for column in range(matrixLength):
                summ += matrix[string][column] *
alpha.index(message[double][column])
                final += alpha[(summ)%26]; summ = 0
    return final
if cryptMode == 'E': finalMessage = encryptDecrypt(regular(startMessage), MatrixKey)
else: finalMessage = encryptDecrypt(regular(startMessage), iMatrixKey)
print("Final message:",finalMessage)
_End
```

// terminal

Start_

\$ python file.py

[E]ncrypt|[D]ecrypt: e

Write the message: helloworld

Final message: RLBYMMSXVA

\$ python file.py

[E]ncrypt|[D]ecrypt: d

Write the message: RLBYMMSXVA

Final message: HELLOWORLD

End

Импорт функции из модуля re.

from re import findall

Создание матрицы 2x2.

matrixLength = 2

MatrixKey = [[15,4],[11,3]]

Нахождение определителя матрицы и если он не равен единице, тогда вывести ошибку.

det = MatrixKey[0][0]*MatrixKey[1][1] - MatrixKey[0][1]*MatrixKey[1][0]

if det != 1:

print("Error: determinant != 1"); raise SystemExit

Создание обратной матрицы.

iMatrixKey = [

[MatrixKey[1][1],-MatrixKey[0][1],

[-MatrixKey[1][0],MatrixKey[0][0]]

]

Создание кортежа с текстом.

alpha = tuple("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

Переключатель режимов шифрования.

cryptMode = input("[E]ncrypt|[D]ecrypt: ").upper()

if cryptMode not in ['E','D']:

print("Error: mode is not Found!"); raise SystemExit

Сообщение для шифрования/расшифрования.

startMessage = input("Write the message: ").upper()

Перебор всех символов сообщения и если символ не найден в диапазоне A-Z, тогда удалить этот символ.

```
for symbol in startMessage:  
    if symbol not in [chr(x) for x in range(65,91)]:  
        startMessage = startMessage.replace(symbol, "")
```

Пока длина сообщения не будет делиться на длину матрицы без остатка – добавлять символ 'Z'.

```
while len(startMessage) % matrixLength != 0: startMessage += 'Z'
```

Функция возвращающая пары символов.

```
def regular(text):  
    template = r"[A-Z]{" + str(matrixLength) + "}"  
    return findall(template, text)
```

Главная функция.

```
def encryptDecrypt(message, matrix, summ = 0, final = ""):
```

Шифрование/расшифрование пар символов по матрице и алфавиту.

```
for double in range(len(message)):  
    for string in range(matrixLength):  
        for column in range(matrixLength):  
            summ += matrix[string][column] * alpha.index(message[double]  
[column])  
            final += alpha[(summ)%26]; summ = 0
```

Вернуть сообщение.

```
return final
```

Если переключатель равен шифрованию, тогда подставить обычную матрицу.

```
if cryptMode == 'E': finalMessage = encryptDecrypt(regular(startMessage),  
MatrixKey)
```

Если переключатель равен расшифрованию, тогда подставить обратную матрицу.

```
else: finalMessage = encryptDecrypt(regular(startMessage), iMatrixKey)
```

Вывод сообщения.

```
print("Final message:", finalMessage)
```

Криптоанализ

101. BruteForce шифра Цезаря.

(Всего существует 25 вариантов дешифровки шифра Цезаря.)

// file.py

Start_

```
cryptMessage = input("Write the message: ").upper()
print("All possible variants of decrypt:")
for key in range(26):
    if key < 10: print("[ 0%d ]"%(key), end = ' ')
    else: print("[ %d ]"%(key), end = ' ')
    for symbol in cryptMessage:
        print(chr((ord(symbol)-key-13)%26+ord('A')), end = " ")
    print()
```

End

// terminal

Start_

\$ python file.py

Write the message: PMTTWEWZTL

All possible variants of decrypt:

```
[ 00 ] PMTTWEWZTL
[ 01 ] OLSSVDVYSK
[ 02 ] NKRRUCUXRJ
[ 03 ] MJQQTBTWQI
[ 04 ] LIPPSASVPH
[ 05 ] KHOORZRUOG
[ 06 ] JGNNQYQTNF
[ 07 ] IFMMPXPSME
[ 08 ] HELLOWORLD
[ 09 ] GDKKNVNQKC
[ 10 ] FCJJMUMPJB
[ 11 ] EBIILTLOIA
[ 12 ] DAHHKSKNHZ
[ 13 ] CZGGJRJMGY
[ 14 ] BYFFIQILFX
[ 15 ] AXEEHPHKEW
[ 16 ] ZWDDGOGJDV
```



```
[ 17 ] YVCCFNFIGU
[ 18 ] XUBBEMENBT
[ 19 ] WTAADLDGAS
[ 20 ] VSZZCKCFZR
[ 21 ] URYYBJBEYQ
[ 22 ] TQXXAIADXP
[ 23 ] SPWWZHZCWO
[ 24 ] ROVVYGYBVN
[ 25 ] QNUUXFXAUM
]_End
```

```
# Сообщение для дешифровки.
```

```
cryptMessage = input("Write the message: ").upper()
```

```
# Вывод получившихся вариантов дешифровки.
```

```
print("All possible variants of decrypt:")
```

```
# Перебор всех ключей от 0 до 25 включительно.
```

```
for key in range(26):
```

```
# Если ключ меньше 10 – выводить индекс первый с нулём.
```

```
if key < 10: print("[ 0%d ]"%(key), end = ' ')
```

```
# Иначе выводить индекс ключа без нуля.
```

```
else: print("[ %d ]"%(key), end = ' ')
```

```
# Посимвольный перебор зашифрованного сообщения и подстановка ключа.
```

```
for symbol in cryptMessage:
```

```
    print(chr((ord(symbol)-key-13)%26+ord('A')), end = '')
```

```
print()
```

102. Частотный криптоанализ.

(Частотный криптоанализ помогает дешифровать сообщения, которые были зашифрованы при помощи простого шифра замены.)

// file.py

Start_1

```
print("""
\t\t A = 8.17% \t N = 6.75%
\t\t B = 1.49% \t O = 7.51%
\t\t C = 2.78% \t P = 1.93%
\t\t D = 4.25% \t Q = 0.10%
\t\t E = 12.7% \t R = 5.99%
\t\t F = 2.23% \t S = 6.33%
\t\t G = 2.02% \t T = 9.06%
\t\t H = 6.09% \t U = 2.76%
\t\t I = 6.97% \t V = 0.98%
\t\t J = 0.15% \t W = 2.36%
\t\t K = 0.77% \t X = 0.15%
\t\t L = 4.03% \t Y = 1.97%
\t\t M = 2.41% \t Z = 0.05%
""")
name = input("File-name: ")
text = ""
try:
    with open(name, "r") as file:
        original = file.read()
except FileNotFoundError:
    print("File is not found!")
else:
    for symbol in original:
        if symbol != " ":
            text += symbol
dictionary = set(text)
def check(words, char):
    k = 0
    for symbol in words:
        if symbol == char: k += 1
    return k
var = 0
print("[*] Result: ")
for symbol in dictionary:
```

```

stat = 100 * check(text,symbol) / len(text)
if var%2 == 0:
    print("\t\t{0} - {1}%\t".format(symbol, round(stat,2)), end = "")
else:
    print("{0} - {1}%" .format(symbol, round(stat,2)))
var += 1
print()
]_End

```

// book.txt

Start_

```

:*.}(!{[}*%?(^.:)$:}>([%-?:.}{?&!@%[[%>/;(>([$:[*]/%.:]>?!+%!?$(?[*%;>!(/%[*
%>%:^^/%?$/!==(+/(&*[[:+!-%+%[:?&]%[(%$/;%!-(?&:^.:]>^!+%@[)/(?#
%.:]>}>[*}($%!/[*%:#%?!?/[*%*]+@=%!/[*%;>:]$%/[!(!=$:[*@%!>+./!]#.@!>-(?^
%>(:>^!>[:*(/:?..]>@>:|$+!(?${[*}(=^]==.!!;,%!>.:]>/*!==.}%/[*%=;}(==*:=+$+%];!
^=:[*]*(=/[*%];;?..]>/:]?=$=%//$%%%;$:[*>($%:>@%(?&}>!#-%$(!+!}:>[*=%//@:![*
%:^![!==@)($(?&!?$:^&::$.=;>($%[*%?(^*%[*>({%!?$(@%#!/[!]!.[*%}:>/[!]!/[*(/
+.=:{%}!/+.$%#!.

```

]_End

// terminal

Start_

\$ python file.py

```

A = 8.17%  N = 6.75%
B = 1.49%  O = 7.51%
C = 2.78%  P = 1.93%
D = 4.25%  Q = 0.10%
E = 12.7%  R = 5.99%
F = 2.23%  S = 6.33%
G = 2.02%  T = 9.06%
H = 6.09%  U = 2.76%
I = 6.97%  V = 0.98%
J = 0.15%  W = 2.36%
K = 0.77%  X = 0.15%
L = 4.03%  Y = 1.97%
M = 2.41%  Z = 0.05%

```

File-name: book.txt

[*] Result:

.	- 3.28%]	- 3.93%
^	- 2.62%	!	- 8.08%
+	- 2.62%	>	- 5.9%
\$	- 4.8%	-	- 1.09%
#	- 1.31%	}	- 3.49%
(- 7.42%	@	- 2.18%
:	- 8.95%	[- 7.42%
/	- 6.11%	?	- 4.8%
{	- 0.44%	;	- 2.62%
&	- 1.53%	%	- 10.48%
=	- 4.8%	*	- 6.11%

l_End

Вывод частоты встречаемости символов английского алфавита.

```
print('
\t\t A = 8.17% \t N = 6.75%
\t\t B = 1.49% \t O = 7.51%
\t\t C = 2.78% \t P = 1.93%
\t\t D = 4.25% \t Q = 0.10%
\t\t E = 12.7% \t R = 5.99%
\t\t F = 2.23% \t S = 6.33%
\t\t G = 2.02% \t T = 9.06%
\t\t H = 6.09% \t U = 2.76%
\t\t I = 6.97% \t V = 0.98%
\t\t J = 0.15% \t W = 2.36%
\t\t K = 0.77% \t X = 0.15%
\t\t L = 4.03% \t Y = 1.97%
\t\t M = 2.41% \t Z = 0.05%
')
```

Имя файла для чтения и создание переменной text.

```
name = input("File-name: "); text = ""
```

Попытка открыть файл, если файл не найден – вывести ошибку. Иначе переписать все символы в переменную text за исключением пробелов.

```
try:
    with open(name, "r") as file:
        original = file.read()
except FileNotFoundError:
    print("File is not found!")
else:
```

```
for symbol in original:  
    if symbol != " "  
        text += symbol
```

```
# Создание множества в котором нет повторяющихся символов.  
dictionary = set(text)
```

```
# Функция для подсчёта количества символов.
```

```
def check(words, char):  
    k = 0  
    for symbol in words:  
        if symbol == char: k += 1  
    return k
```

```
# Создание переменной var.
```

```
print("[*] Result: "); var = 0
```

```
# Посимвольный перебор множества dictionary.
```

```
for symbol in dictionary:
```

```
# Вычисление статистика встречаемости символа в тексте.
```

```
stat = 100 * check(text,symbol) / len(text)
```

```
# Если переменная var делится на два без остатка, тогда выводить сообщение без  
символа новой строки.
```

```
if var%2 == 0:  
    print("\\t\\t{0} – {1}%\\t".format(symbol, round(stat,2)), end = "")
```

```
# Если же переменная var не делится на два без остатка, тогда выводить  
сообщение с символом новой строки.
```

```
else:  
    print("{0} – {1}%".format(symbol, round(stat,2)))  
var += 1  
print()
```

103. BruteForce криптографической хеш-функции по словарю.

(Зашифрованные сообщения при помощи криптографических хеш-функций в основном являются паролями.

Словарь, который я использую в этой и последующих программах:

<http://scrapmaker.com/download/data/wordlists/dictionaries/rockyou.txt>)

// file.py

Start_1

```
#!/usr/bin/python3
```

```
from sys import argv
```

```
from hashlib import sha256, sha512, md5
```

```
# chmod +x main.py
```

```
# ./main.py sha256 hash.txt rockyou.txt
```

```
line = "-----"
```

```
try:
```

```
    hashAlgr, fileHash, fileDict = argv[1], argv[2], argv[3]
```

```
except IndexError:
```

```
    print("Error: Arguments!")
```

```
    raise SystemExit
```

```
with open(fileHash) as file:
```

```
    hashFunc = file.read()
```

```
    hashFunc = hashFunc.replace("\n", "")
```

```
def generator(string):
```

```
    for word in string:
```

```
        passwd = word.replace("\n", "")
```

```
        if encrypt(passwd) == hashFunc:
```

```
            yield line + "\n[True]: " + passwd
```

```
            return
```

```
        else:
```

```
            yield "[False]: " + passwd
```

```
def encrypt(string):
```

```
    passwd = string.encode()
```

```
    if hashAlgr == "md5":
```

```
        signature = md5(passwd).hexdigest()
```

```
    elif hashAlgr == "sha256":
```

```
        signature = sha256(passwd).hexdigest()
```

```
    elif hashAlgr == "sha512":
```

```
        signature = sha512(passwd).hexdigest()
```

```
    else: print("Error: not found algorithm."); raise SystemExit
```

```
    return signature
```

```
print(line)
with open(fileDict, errors = "ignore") as dictionary:
    for password in generator(dictionary):
        print(password)
```

```
print(line)
```

```
l_End
```

```
// hash.txt
```

```
Start_[
```

```
481f6cc0511143ccdd7e2d1b1b94faf0a700a8b49cd13922a70b5ae28acaa8c5
```

```
l_End
```

```
// terminal
```

```
Start_[
```

```
$ chmod +x file.py
```

```
$ ./file.py sha256 hash.txt rockyou.txt
```

```
-----
[False]: 123456
[False]: 12345
[False]: 123456789
[False]: password
[False]: iloveyou
[False]: princess
[False]: 1234567
[False]: rockyou
[False]: 12345678
[False]: abc123
[False]: nicole
[False]: daniel
[False]: babygirl
[False]: monkey
[False]: lovely
[False]: jessica
```

```
-----
[True]: 654321
-----
```

```
l_End
```

```
# Возможность использовать chmod +x.  
#!/usr/bin/python3
```

```
# Импортирование функции из sys и функций из hashlib.
```

```
from sys import argv  
from hashlib import sha256, sha512, md5
```

```
# Примеры использования программы.
```

```
# chmod +x main.py
```

```
# ./main.py sha256 hash.txt rockyou.txt
```

```
# Линия-разделитель.
```

```
line = "-----"
```

```
# Аргументы через терминал, если же каких-то аргументов не хватает – вывести ошибку.
```

```
try:
```

```
    hashAlgr, fileHash, fileDict = argv[1], argv[2], argv[3]
```

```
except IndexError:
```

```
    print("Error: Arguments!")
```

```
    raise SystemExit
```

```
# Чтение файла с хешем и удаление символа новой строки.
```

```
with open(fileHash) as file:
```

```
    hashFunc = file.read()
```

```
    hashFunc = hashFunc.replace('\n', '')
```

```
# Создание функции-генератора.
```

```
def generator(string):
```

```
# Перебор всех слов в словаре.
```

```
for word in string:
```

```
# Удаление в слове символа новой строки.
```

```
passwd = word.replace('\n', '')
```

```
# Сравнение слова с криптографической хеш-функцией. Если криптографическая хеш-функция равна слову, тогда вывести True и закрыть программу.
```

```
if encrypt(passwd) == hashFunc:
```

```
    yield line + "\n[True]: " + passwd
```

```
    return
```



```
# Иначе вывести False и продолжать поиск слова.
else:
    yield "[False]: "+passwd
# Функция перевода слова в криптографическую хеш-функцию для последующего
сравнения.
def encrypt(string):

# Перевод слова в кодировку.
passwd = string.encode()

# Если алгоритм равен md5, то использовать функцию md5.
if hashAlgr == "md5":
    signature = md5(passwd).hexdigest()

# Если алгоритм равен sha256, то использовать функцию sha256.
elif hashAlgr == "sha256":
    signature = sha256(passwd).hexdigest()

# Если алгоритм равен sha512, то использовать функцию sha512.
elif hashAlgr == "sha512":
    signature = sha512(passwd).hexdigest()

# Если алгоритм не был найден – вывести ошибку.
else: print("Error: not found algorithm."); raise SystemExit

# Вернуть криптографическую хеш-функцию.
return signature

# Открыть словарь на чтение и с игнорированием ошибок.
print(line)
with open(fileDict, errors = "ignore") as dictionary:

# Перебор каждого слова в словаре через генератор.
for password in generator(dictionary):
    print(password)
print(line)
```

104. BruteForce запароленного zip-архива по словарю.

(Поддерживаются только архивы формата zip.)

// file.py

Start_

```
#!/usr/bin/python3
from zipfile import ZipFile
from sys import argv
from os import mkdir
# chmod +x main.py
# ./main.py archive.zip rockyou.txt
try:
    archiveFile, dictionaryFile = argv[1], argv[2]
except IndexError:
    print("Error: Arguments!")
    raise SystemExit
line = "-----"
def generator(string):
    for word in string:
        passwd = word.replace("\n", "")
        archive.setpassword(passwd.encode())
        try:
            archive.extractall(directory)
        except:
            yield "[False]: " + passwd
        else:
            yield line + "\n[True]: " + passwd; return
directory = "ExtractArchive"
try: mkdir(directory)
except FileExistsError: pass
print(line)
with open(dictionaryFile, errors = 'ignore') as dictionary:
    with ZipFile(archiveFile) as archive:
        print(line)
        for password in generator(dictionary):
            print(password)
print(line)
_End
```

// terminal

Start_

\$ chmod +x file.py

\$./file.py archive.zip rockyou.txt

[False]: 123456
[False]: 12345
[False]: 123456789
[False]: password
[False]: iloveyou
[False]: princess
[False]: 1234567
[False]: rockyou
[False]: 12345678
[False]: abc123
[False]: nicole
[False]: daniel
[False]: babygirl
[False]: monkey
[False]: lovely
[False]: jessica
[False]: 654321
[False]: michael
[False]: ashley

[True]: qwerty

End

Возможность использовать chmod +x.

#!/usr/bin/python3

Импортирование функций из модулей zipfile, sys, os.

from zipfile import ZipFile

from sys import argv

from os import mkdir

Примеры использования программы.

chmod +x main.py

./main.py archive.zip rockyou.txt

Ввод аргументов с терминала.

```
try:  
    archiveFile, dictionaryFile = argv[1], argv[2]
```

Если же не поступило какого-то аргумента, тогда вывести ошибку.

```
except IndexError:  
    print("Error: Arguments!")  
    raise SystemExit
```

Линия-разделитель.

```
line = "-----"
```

Создание функции-генератор.

```
def generator(string):
```

Перебор слов в словаре.

```
for word in string:
```

Удаление символа новой строки в слове.

```
passwd = word.replace('\n', '')
```

Установка пароля под архив, который пытаемся открыть.

```
archive.setPassword(passwd.encode())
```

Попытка разархивировать архив.

```
try:  
    archive.extractall(directory)
```

Если пароль не верный – вывести False и продолжить искать пароль.

```
except:  
    yield "[False]: " + passwd
```

Если же пароль верный – вывести True и закрыть программу.

```
else:  
    yield line + "\n[True]: " + passwd; return
```

Директория, в которой будут помещены разархивированные файлы. Если директория уже создана – ничего не делать.

```
directory = "ExtractArchive"
```

```
try: mkdir(directory)
```

```
except FileExistsError: pass
```

Открытие словаря на уровне чтения с игнорирование ошибок.

```
print(line)
```

```
with open(dictionaryFile, errors = 'ignore') as dictionary:
```

Открытие zip-архива на чтение.

```
with ZipFile(archiveFile) as archive:
```

```
    print(line)
```

Перебор каждого слова в словаре через генератор.

```
for password in generator(dictionary):
```

```
    print(password)
```

```
print(line)
```

Стеганография

201. Сообщение в файле на уровне байтов.

(Чтение занесённого текста в следующем скрипте. Сам файл не теряет своих прежних функций. Осторожно: Файл перезаписывается!)

// file.py

Start_[

#!/usr/bin/python3

from sys import argv

chmod +x main.py

./main.py file.jpg

try:

 nameFile = argv[1]

 textFile = argv[2:]

except IndexError:

 print("Error: Arguments!"); raise SystemExit

with open(nameFile, 'ab') as file:

 file.write("".join(textFile).encode("utf-8"))

 print("[+] File successfully overwritten!")

End_

// terminal

Start_[

\$ chmod +x file.py

\$./file.py picture.jpeg helloworld!

[+] File: picture.jpeg successfully overwritten!

End_

Возможность использовать chmod +x.

#!/usr/bin/python3

Импорт аргументов из модуля sys.

from sys import argv

Пример использования программы.

chmod +x main.py

./main.py file.jpg

Попытка указать аргументы в виде переменных.

try:

nameFile = argv[1]

textFile = argv[2:]

Если произошла ошибка с аргументами, тогда закрыть программу.

except IndexError:

print("Error: Arguments!"); raise SystemExit

Открыть файл на уровне байт и добавить в файл введённый текст.

with open(nameFile, 'ab') as file:

file.write("".join(textFile).encode("utf-8"))

print("[+] File successfully overwritten!")

202. Чтение байтов в файле.

(Побайтовое чтение файла на поиск скрытых сообщений.)

// file.py

Start_

```
#!/usr/bin/python3
```

```
from sys import argv
```

```
# chmod +x main.py
```

```
# ./main.py file.jpg
```

```
try:
```

```
    nameFile = argv[1]
```

```
except IndexError:
```

```
    print("Error: arguments"); raise SystemExit
```

```
try:
```

```
    with open(nameFile, "rb") as file:
```

```
        byte = file.read(1)
```

```
        counter = 0
```

```
        while byte:
```

```
            print(byte)
```

```
            byte = file.read(1)
```

```
            counter += 1
```

```
except FileNotFoundError:
```

```
    print("[x] File: '{name}' is not defined!".format(name = nameFile))
```

```
else:
```

```
    print("Number of bytes in the '{name}': {number}".format(name = nameFile,
```

```
number = counter))
```

End_

// terminal

Start_

```
$ chmod +x file.py
```

```
$ ./file.py picture.jpeg
```

```
...
```

```
...
```

```
...
```

```
b'K'
```

```
b'\xfa'
```

```
b'c'
```

```
b'\x18'
```

```
b'\xae'
```


b'R'
b'\xbe'
b'\
b'p'
b'\x07'
b'\x1e'
b'!'
b'q'
b'\xae'
b'\x95'
b'5'
b'\x02'
b'\xbf'
b'\xff'
b'\xd9'
b'h'
b'e'
b'l'
b'l'
b'o'
b'w'
b'o'
b'r'
b'l'
b'd'
b'!'

Number of bytes in the 'picture.jpeg': 1620896

l_End

Возможность использовать chmod +x.

#!/usr/bin/python3

Импорт аргументов из модуля sys.

from sys import argv

Пример использования программы.

chmod +x main.py

./main.py file.jpg

Попытка внести аргументы в переменную.

try:

nameFile = argv[1]

Если вышла ошибка с аргументами, тогда закрыть программу.

except IndexError:

print("Error: arguments"); raise SystemExit

Попробовать открыть файл и выводить побайтово всё содержимое файла.

try:

with open(nameFile, "rb") as file:

byte = file.read(1)

counter = 0

while byte:

print(byte)

byte = file.read(1)

counter += 1

Если файл не был найден, тогда написать об этом.

except FileNotFoundError:

print("[x] File: '{name}' is not defined!".format(name = nameFile))

Иначе, если всё прошло успешно – написать количество байт в файле.

else:

print("Number of bytes in the '{name}': {number}".format(name = nameFile, number = counter))

203. Занесение архива в файл.

(В итоге файл можно будет использовать как архив с функцией разархивации. Сам файл не теряет своих прежних функций. Осторожно: Файл перезаписывается!)

// file.py

Start_[

```
#!/usr/bin/python3
```

```
from sys import argv
```

```
# chmod +x main.py
```

```
# ./main.py file.jpg
```

```
try:
```

```
    nameFile, archiveFile = argv[1], argv[2]
```

```
except IndexError:
```

```
    print("Error: Arguments!")
```

```
    raise SystemExit
```

```
try:
```

```
    with open(nameFile,"rb") as file:
```

```
        readFile = file.read()
```

```
    with open(archiveFile,"rb") as archive:
```

```
        readArchive = archive.read()
```

```
except FileNotFoundError:
```

```
    print("Error: File is not found!")
```

```
    raise SystemExit
```

```
with open(nameFile,"wb") as file:
```

```
    file.write(readFile)
```

```
    file.write(readArchive)
```

```
    print("[+] File: successfully overwritten!")
```

]_End

// terminal

Start_[

```
$ chmod +x file.py
```

```
$ ./file.py picture.jpeg archive.zip
```

```
[+] File: successfully overwritten!
```

```
$ unzip picture.jpeg
```

]_End

Возможность использовать chmod +x.

```
#!/usr/bin/python3
```

```
# Импорт функции из модуля sys.  
from sys import argv  
  
# Пример использования программы.  
# chmod +x main.py  
# ./main.py file.jpg  
  
# Указание аргументов в терминале.  
try:  
    nameFile, archiveFile = argv[1], argv[2]  
  
# Если не хватает какого-то аргумента, тогда вывести ошибку.  
except IndexError:  
    print("Error: Arguments!")  
    raise SystemExit  
  
# Попытка открыть файл и архив.  
try:  
    with open(nameFile, "rb") as file:  
        readFile = file.read()  
    with open(archiveFile, "rb") as archive:  
        readArchive = archive.read()  
  
# Если же какой-то файл не был найден – вывести ошибку.  
except FileNotFoundError:  
    print("Error: File is not found!")  
    raise SystemExit  
  
# Перезаписать файл и внести в него сначала себя, потом архив.  
with open(nameFile, "wb") as file:  
    file.write(readFile)  
    file.write(readArchive)  
    print("[+] File: successfully overwritten!")
```

Дополнительные скрипты

301. Шифровальщик файлов.

(Шифровальщик основан на AES шифровании, но зашифровывает он не один файл, а целую директорию, которая будет являться корневой директорией.

Осторожно: Файлы перезаписываются!)

// file.py

Start_1

```
direct = input("Write the root directory: ")
password = input("Write the password: ")
with open("encrypt.py", "w") as encryptFile:
    encryptFile.write("""
import os
from sys import argv
from pyAesCrypt import encryptFile
def encrypt(file):
    print("-----")
    password = ""+str(password)+"
    bufferSize = 128*1024
    encryptFile(file, file+".crp", password, bufferSize)
    print("[encrypted] '{name}.crp".format(name = file))
    os.remove(file)
def walk(dir):
    for name in os.listdir(dir):
        path = os.path.join(dir, name)
        if os.path.isfile(path): encrypt(path)
        else: walk(path)
walk(""+str(direct)+"")
print("-----")
os.remove(argv[0])
""")
    print("[+] File 'encrypt.py' successfully saved!")
with open("decrypt.py", "w") as decryptFile:
    decryptFile.write("""
import os
from sys import argv
from pyAesCrypt import decryptFile
def decrypt(file):
```

```

print("-----")
password = ""+str(password)+"
bufferSize = 128*1024
decryptFile(file, os.path.splitext(file)[0], password, bufferSize)
print("[decrypted] '{name}'".format(name = os.path.splitext(file)[0]))
os.remove(file)
def walk(dir):
    for name in os.listdir(dir):
        path = os.path.join(dir, name)
        if os.path.isfile(path):
            try: decrypt(path)
            except: pass
        else: walk(path)
walk(""+str(direct)+"")
print("-----")
os.remove(argv[0])
")
    print("[+] File 'decrypt.py' successfully saved!")
]_End

```

// terminal

Start_1

\$ python file.py

Write the root directory: /home/user/Templates/Python/TEST/

Write the password: helloworld

[+] File 'encrypt.py' successfully saved!

[+] File 'decrypt.py' successfully saved!

\$ python encrypt.py

[encrypted] '/home/user/Templates/Python/TEST/DIR/some.rb.crp'

[encrypted] '/home/user/Templates/Python/TEST/DIR/picture.jpeg.crp'

[encrypted] '/home/user/Templates/Python/TEST/crypter.py.crp'

[encrypted] '/home/user/Templates/Python/TEST/SOMETHING/main.py.crp'

[encrypted] '/home/user/Templates/Python/TEST/SOMETHING/file.txt.crp'

[encrypted] '/home/user/Templates/Python/TEST/decrypt.py.crp'

```
-----  
[encrypted] '/home/user/Templates/Python/TEST/encrypt.py.crp'  
-----
```

```
$ python decrypt.py
```

```
-----  
[decrypted] '/home/user/Templates/Python/TEST/crypter.py'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/DIR/some.rb'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/DIR/picture.jpeg'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/encrypt.py'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/SOMETHING/file.txt'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/SOMETHING/main.py'  
-----
```

```
[decrypted] '/home/user/Templates/Python/TEST/decrypt.py'  
-----
```

]End

```
# Корневая директория с которой будет начинаться шифрование файлов.
```

```
direct = input("Write the root directory: ")
```

```
# Установка пароля для всех файлов.
```

```
password = input("Write the password: ")
```

```
# Создание python-скрипта, который будет зашифровывать файлы.
```

```
with open("encrypt.py","w") as encryptFile:
```

```
    encryptFile.write(''
```

```
# Импортирование модуля os и функций argv и encryptFile.
```

```
import os
```

```
from sys import argv
```

```
from pyAesCrypt import encryptFile
```

```
# Функция encrypt, которая отвечает за шифрование отдельного файла.
```

```
def encrypt(file):
```

```
# Перенос пароля из билдера в скрипт шифрования.
print("-----")
password = '''+str(password)'''

# Установка размера буфера для шифрования.
bufferSize = 128*1024

# Шифрование файла.
encryptFile(file, file+".crp", password, bufferSize)

# Вывести результат и удалить копию незашифрованного файла.
print("[encrypted] '{name}.crp'".format(name = file))
os.remove(file)

# Функция walk, которая отвечает за хождение по директориям и выборку файлов.
def walk(dir):

# Перебор всех файлов и директорий в выбранной директории.
for name in os.listdir(dir):

# Полный путь к файлу или директории.
path = os.path.join(dir, name)

# Если объект является файлом, тогда использовать функцию encrypt.
if os.path.isfile(path): encrypt(path)

# Если же объект является директорией, тогда перейти в эту директорию.
else: walk(path)

# Указание корневой директории из билдера.
walk(''+str(direct)+'')
print("-----")

# Удаление скриптового файла шифрования после выполнения всех функций.
os.remove(argv[0])
'''

# Вывод результата создания скрипта.
print("[+] File 'encrypt.py' successfully saved!")
```


Создание python-скрипта, который будет расшифровывать файлы.

```
with open("decrypt.py","w") as decryptFile:  
    decryptFile.write("""
```

Импортирование модуля os и функций argv и decryptFile.

```
import os  
from sys import argv  
from pyAesCrypt import decryptFile
```

Функция decrypt, которая отвечает за расшифрование отдельного файла.

```
def decrypt(file):
```

Перенос пароля из билдера в скрипт шифрования.

```
print("-----")  
password = ""+str(password)+"
```

Установка размера буфера для шифрования.

```
bufferSize = 128*1024
```

Расшифрование файла.

```
decryptFile(file, os.path.splitext(file)[0], password, bufferSize)
```

Вывести результат и удалить копию зашифрованного файла.

```
print("[decrypted] '{name}'.format(name = os.path.splitext(file)[0]))  
os.remove(file)
```

Функция walk, которая отвечает за хождение по директориям и выборку файлов.

```
def walk(dir):
```

Перебор всех файлов и директорий в выбранной директории.

```
for name in os.listdir(dir):
```

Полный путь к файлу или директории.

```
path = os.path.join(dir, name)
```

Если объект является файлом, тогда попробовать расшифровать файл. Если же файл не зашифрован, тогда пропустить.

```
if os.path.isfile(path):  
    try: decrypt(path)  
    except: pass
```

Если же объект является директорией, тогда перейти в эту директорию.

else: **walk**(path)

Указание корневой директории из билдера.

walk(""+**str**(direct)+"")

print("-----")

Удаление скриптового файла расшифрования после выполнения всех функций.

os.remove(argv[0])

''')

Вывод результата создания скрипта.

print("[+] File 'decrypt.py' successfully saved!")

302. Скрипт для создания .onion сайта в сети Tor.

(Скрипт создан для операционной системы Linux. Скрипт можно использовать не только для создания .onion сайта, но и для его последующего включения в сеть, т.к. скрипт проверяет наличие предыдущих директорий и файлов не давая их удалять.)

// file.py

Start_1

```
#!/usr/bin/python3
# $ chmod +x main.py
# $ sudo ./main.py
from os import system, listdir, mkdir, chdir, getcwd
from time import sleep
'''
dist = ["apt-get install", "pacman -S"]
prog = ["tor"]
for distribution in dist:
    for program in prog:
        system("{dist} {prog}".format(dist = distribution, prog = program))
'''

www = [False, "/var/www/"]
onion = [False, "/var/www/onion/"]
main_files = "/var/lib/tor/onion/"
html_file = [False, "/var/www/onion/index.html"]
host_file = "/var/lib/tor/onion/hostname"
key_file = "/var/lib/tor/onion/private_key"
readme = [False, "README"]
torrc = [False, "/etc/tor/torrc"]
string1 = "HiddenServiceDir /var/lib/tor/onion"
string2 = "HiddenServicePort 80 127.0.0.1:80"
if "www" in listdir("/var/"):
    www[0] = True
if www[0] == False:
    mkdir(www[1])
    print("Directory '{dir}' created".format(dir = www[1]))
else:
    if "onion" in listdir(www[1]):
        onion[0] = True
if onion[0] == False:
    mkdir(onion[1])
    print("Directory '{dir}' created".format(dir = onion[1]))
if "index.html" in listdir(onion[1]):
```

```

        html_file[0] = True
if html_file[0] == False:
    with open(html_file[1], "w") as html:
        html.write("""
<!DOCTYPE html>
<html>
    <head>
        <title>Script_by_#%&!</title>
        <meta charset="utf-8">
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html> """)
        print("File '{file}' created".format(file = html_file[1]))
with open(torrc[1], "r") as tor:
    for string in tor:
        if string == string1 or string == string2:
            torrc[0] = True
            break
if torrc[0] == False:
    with open(torrc[1], "a") as tor:
        tor.write(string1+"\n"+string2)
        print("Strings appended in the '{file}' file.".format(file = torrc[1]))
        print("{0}\n{1}".format(string1, string2))
system("systemctl start tor.service")
system("systemctl restart tor.service")
sleep(1)
with open(host_file, "r") as host:
    hostname = host.read()
    print("File '{file}' read".format(file = host_file))
with open(key_file, "r") as key:
    private_key = key.read()
    print("File '{file}' read".format(file = key_file))
if readme[1] in listdir(getcwd()):
    readme[0] = True
if www[0] == False or onion[0] == False or html_file[0] == False or readme[0] ==
False:
    with open(readme[1], "w") as info:
        info.write("""
Tor configuration: ""+torrc[1]+"":
- ""+string1+""

```

```

-      ""+string2+"\n
HTML file: ""+html_file[1)+"\n
Main files: ""+main_files+"
-   hostname ["+host_file+"]:
""+hostname+"
-   private_key ["+key_file+"]:
""+private_key+"
Use this command for run tor service:
[for one time, after rebooting use this command again]
-   systemctl start tor.service
[for always time]
-   systemctl enable tor.service\n
Use this command for activate port 80:
[use in the directory: ""+onion[1)+""]
-   python3 -m http.server 80
""
        if readme[0] == True:
            print("File '{file}' updated".format(file = readme[1]))
        else:
            print("File '{file}' created".format(file = readme[1]))
chdir(onion[1])
system("python3 -m http.server 80")
]_End

```

// terminal

Start_

```

$ chmod +x file.py
$ sudo ./file.py
Directory '/var/www/' created
Directory '/var/www/onion/' created
File '/var/www/onion/index.html' created
Strings appended in the '/etc/tor/torrc' file:
'HiddenServiceDir /var/lib/tor/onion'
'HiddenServicePort 80 127.0.0.1:80'
File '/var/lib/tor/onion/hostname' read
File '/var/lib/tor/onion/private_key' read
File 'README' created
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
]_End

```

// README

Start_1

Tor configuration: /etc/tor/torrc:

- HiddenServiceDir /var/lib/tor/onion
- HiddenServicePort 80 127.0.0.1:80

HTML file: /var/www/onion/index.html

Main files: /var/lib/tor/onion/

- hostname [/var/lib/tor/onion/hostname]:

jua25maeyrn4mthj.onion

#!RoNxhLKJ!m-MMEOreomw8ThCo3SW7WXz8564PbGDOOv1qv93bJXM

- private_key [/var/lib/tor/onion/private_key]:

-----BEGIN RSA PRIVATE KEY-----

MIICXAIBAAKBgQD1Pg05PdBRbIjtACSnck2sTLWbqlLNiVgt+GcwFH5lDpeinIEM
Bmn7dKIOEynMEz+7xCL0vltohgygD0Wvxp3AAAtvheu9yzllXLoG2Ipmog5LlUjLZ
MA+DKgAd0wd5BAXEFoEiVleI2ieSaXf//ER5nkulfVISFB+
+YqmH90OoRQIDAQAB

AoGAAqRSfCVgUyA6MWNpAazHjW2eHzksfy5VltkwM2Jli2QEJ5i/YAsMBtHi6NF
Nf+XFiy8u6o5Tdzz0d2YQJaBKIT5WPon4XLH05/cbGDTXHsQVrC2BudWpXIcevdN
aNK2eVyeBcabYk8vz90sosq4NBzd/0e75hXR3qBE35VsTkECQQD7GqDLRM8ib47
W

DWDLBVJmlc8B+I89/9f1/5mY8AGUjlVIRIk2X4wpumMtqHcsqAHf69/CcKOLl6Qo
C1zf0EqNAkEA+gYqYnuk4deBY1C2EVQLQyC2NGjXl5Kl2ho7yNKGntw11iSEz0b
W

xWnnFHfLw6WX/W1/e1wO6Ear8kcyPHoCmQJAP1qFjSOMOzz4RctUS4TJOHa9ptq
m

kVb2jutxPmP3khqjK7uW/u/2diS/lyp0/wBYkL17VByFNtgIo83SHen4lQJAHqo+
7JfJqcFqsymRCxMJxpPuhQMO3jJIUTXCe2EGzdkoaTlVaK7BjLjudJ40yaw3tgeG
CTVDRs3ULQT6blxwkQJBAMiY3h3FB34NCtMzcVQt1gFtax7U8Uu7goo6k/3AD
O

wCgUM7XUDPtQ8B/Lz0w20JMxkO/D0WgvQWL2hRZE9MA=

-----END RSA PRIVATE KEY-----

Use this command to run tor service:

[for one time, after rebooting use this command again]

- systemctl start tor.service

[for always time]

- systemctl enable tor.service

Use this command to activate port 80:

[use in the directory: /var/www/onion/]

- python3 -m http.server 80

]_End

Даёт возможность chmod +x.

#!/usr/bin/python3

Инструкция по запуску скрипта.

\$ chmod +x main.py

\$ sudo ./main.py

Импортирование функций из модуля os и time.

from os import system, listdir, mkdir, chdir, getcwd

from time import sleep

Раскомментировать, если нужно установить tor. (можно также дополнить список и дописать туда допустим nginx).

'''

dist = ["apt-get install", "pacman -S"]

prog = ["tor"]

for distribution in dist:

for program in prog:

system("{dist} {prog}".format(dist = distribution, prog = program))

'''

Директории, файлы и строки, которые подлежат проверке или созданию.

www = [False, "/var/www/"]

onion = [False, "/var/www/onion/"]

main_files = "/var/lib/tor/onion/"

html_file = [False, "/var/www/onion/index.html"]

host_file = "/var/lib/tor/onion/hostname"

key_file = "/var/lib/tor/onion/private_key"

readme = [False, "README"]

torrc = [False, "/etc/tor/torrc"]

string1 = "HiddenServiceDir /var/lib/tor/onion"

string2 = "HiddenServicePort 80 127.0.0.1:80"

Если директория 'www' уже находится в директории '/var/', тогда присвоить значение True.

if "www" in listdir("/var/"):

www[0] = True

Если www[0] всё-таки равно значению False (нет в директории '/var/'), тогда создать директорию 'www'.

```
if www[0] == False:
    mkdir(www[1])
    print("Directory '{dir}' created".format(dir = www[1]))
```

Если же www[0] равно значению True, тогда проверить наличие директории 'onion' в директории '/var/www/'. Если директория найдена – присвоить True.

```
else:
    if "onion" in listdir(www[1]):
        onion[0] = True
```

Если onion[0] всё-таки равно значению False (нет в директории '/var/www/'), тогда создать директорию 'onion'.

```
if onion[0] == False:
    mkdir(onion[1])
    print("Directory '{dir}' created".format(dir = onion[1]))
```

Если в директории '/var/www/onion/' находится файл 'index.html', тогда присвоить значение True.

```
if "index.html" in listdir(onion[1]):
    html_file[0] = True
```

Если всё-таки html_file[0] равен значению False, тогда создать html файл в директории '/var/www/onion/'.

```
if html_file[0] == False:
    with open(html_file[1], "w") as html:
        html.write('
<!DOCTYPE html>
<html>
    <head>
        <title>Script_by_#%&!/title>
        <meta charset="utf-8">
    </head>
    <body>
        <p>Hello World!</p>
    </body>
</html> ')
    print("File '{file}' created".format(file = html_file[1]))
```

Открыть конфигурационный файл tor'а на чтение.

```
with open(torrc[1], "r") as tor:
```


Перебор всех строк в файле 'torrc'.

for string **in** tor:

Если строка равна строке1 или строке2, тогда присвоить значение True и остановить цикл.

if string == string1 **or** string == string2:

 torrc[0] = True

break

Если же значение torrc[0] равно False, тогда открыть конфигурационный файл tor'a на дополнение.

if torrc[0] == False:

with open(torrc[1], "a") **as** tor:

Внести в конец файла две строки: string1, string2.

tor.write(string1+"\n"+string2)

print("Strings appended in the '{file}' file:".format(file = torrc[1]))

print("'{0}'\n'{1}'".format(string1,string2))

Запустить сервисы tor'a и подождать секунду.

system("systemctl start tor.service")

system("systemctl restart tor.service")

sleep(1)

Открыть 'hostname' файл на чтение и скопировать всё в переменную hostname.

with open(host_file, "r") **as** host:

 hostname = host.read()

print("File '{file}' read".format(file = host_file))

Открыть 'private_key' файл на чтение и скопировать всё в переменную private_key.

with open(key_file, "r") **as** key:

 private_key = key.read()

print("File '{file}' read".format(file = key_file))

Если файл 'README.txt' находится в директории запуска скрипта, тогда присвоить значение True.

if readme[1] **in** listdir(getcwd()):

 readme[0] = True

Если директория 'www' создалась только при запуске скрипта или же директория 'onion' создалась только при запуске скрипта, или же 'index.html' был создан только при запуске скрипта или же 'README' был создан только при запуске скрипта.

```
if www[0] == False or onion[0] == False or html_file[0] == False or readme[0] == False:
```

Создать/перезаписать файл 'README'.

```
with open(readme[1], "w") as info:
```

```
    info.write("""
```

```
Tor configuration: ''+torrc[1]+'':
```

```
- ''+string1+''
```

```
- ''+string2+'\n'
```

```
HTML file: ''+html_file[1]+''\n'
```

```
Main files: ''+main_files+''
```

```
- hostname [''+host_file+'']:
```

```
''+hostname+''
```

```
- private_key [''+key_file+'']:
```

```
''+private_key+''
```

```
Use this command for run tor service:
```

```
[for one time, after rebooting use this command again]
```

```
- systemctl start tor.service
```

```
[for always time]
```

```
- systemctl enable tor.service\n'
```

```
Use this command for activate port 80:
```

```
[use in the directory: ''+onion[1]+'']
```

```
- python3 -m http.server 80
```

```
''')
```

Если файл 'README' до этого был уже создан, тогда написать, что файл обновлён, иначе написать, что он создан.

```
if readme[0] == True:
```

```
    print("File '{file}' updated".format(file = readme[1]))
```

```
else:
```

```
    print("File '{file}' created".format(file = readme[1]))
```

Перейти в директорию '/var/www/onion/' и от туда запустить python сервер.

```
chdir(onion[1])
```

```
system("python3 -m http.server 80")
```

409. ÄᄁГҺᄇÄжӘçÇ³ăċăä³¼³ăċăä³HҺүГҺYᄀ³⁰.

410. [%%,:\$(!#-(==%>:,,\$(!#)!/!/%>(!=-((==%>}*::,%>![%\$(??:>[%%>?#!=(^:>?(!^>:
+![=%!/[[*%=![%[:[*%%!>=.[*%-(==%>(\$%?[(.>>+!(?/)?-?:}?[*%,:\$(!#+]>\$%>%\$
{(#[(+/(?@%?(#!{!==(%)!-!%@%>.>./!!?\$/!?!^>!?!#(/#:@%[]%%?%\$#%+@%>!?
\$:#[:@%>^:]>+%?!?%\$[*>%%}:+%?@%[]%%?[*%!&%/:^!?\$}%>%[!>&%[%\$[*%-
(==%>:>(&?![%\$[*%?!+%,,\$(!#(!/!/%>(%/:^!?!?[(?&=%[%>/%?[[[:[*%=#!=@!>!
%!;>%//[*%/%=%[%>/(?#=\$%\$^:]>#>.,[:&>!+/:>#(;*%>/:^[*%^:]>#>.,[:&>!+/%?
[:?=.?:%*!/@%%?%\$%^?([({%=./:={%%\$

411. Инверсия SVOOLDLIOW?

412. PRNYJGTASILOVIOWLAUCTJFJJZ [[9,?],[2,11]].

413. 5.6.13.8.4.3.9.11.0.1.7.12.2.10.

414.401/55/546/288/275/522/558/292/513/438/77/129/526/297/206/99/49/78/85/118/44
9/134/450/325/414/516/513/39/525/88/291/352/132/532/.

415. 0//**?YS 7IGA V3 P]::!VI:]]S.

Связаться с автором книги можно здесь: [byuqfàÛÛreüəɔdÛεYdæjxÔÏ](#)
GitHub профиль автора: [N:U/1BBG/SMPEI5H.HCTM/7UOTRT](#)
YouTube канал автора: [hts/wwwyuuecmCytultp:/w.otb.o//rpFnT](#)
Сайт автора (иногда работает, иногда нет): 302.