
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p align="center">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p align="center">Código: GUIA-PRLE-001</p>	<p align="right">Página: 1</p>

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Seguridad Informatica				
TÍTULO DE LA PRÁCTICA:	<i>CRIPTOGRAFÍA CLÁSICA: CIFRADO MONOALFABÉTICO Y POLIALFABÉTICO</i>				
NÚMERO DE PRÁCTICA:	<i>02</i>	AÑO LECTIVO:	<i>2023</i>	NRO. SEMESTRE:	<i>A</i>
FECHA DE PRESENTACIÓN	<i>16/06/2023</i>	HORA DE PRESENTACIÓN			
INTEGRANTE (s): Yoset Cozco Mauri				NOTA:	
DOCENTE(s): <i>Juan carlos Zuñiga</i>					

SOLUCIÓN Y RESULTADOS

I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

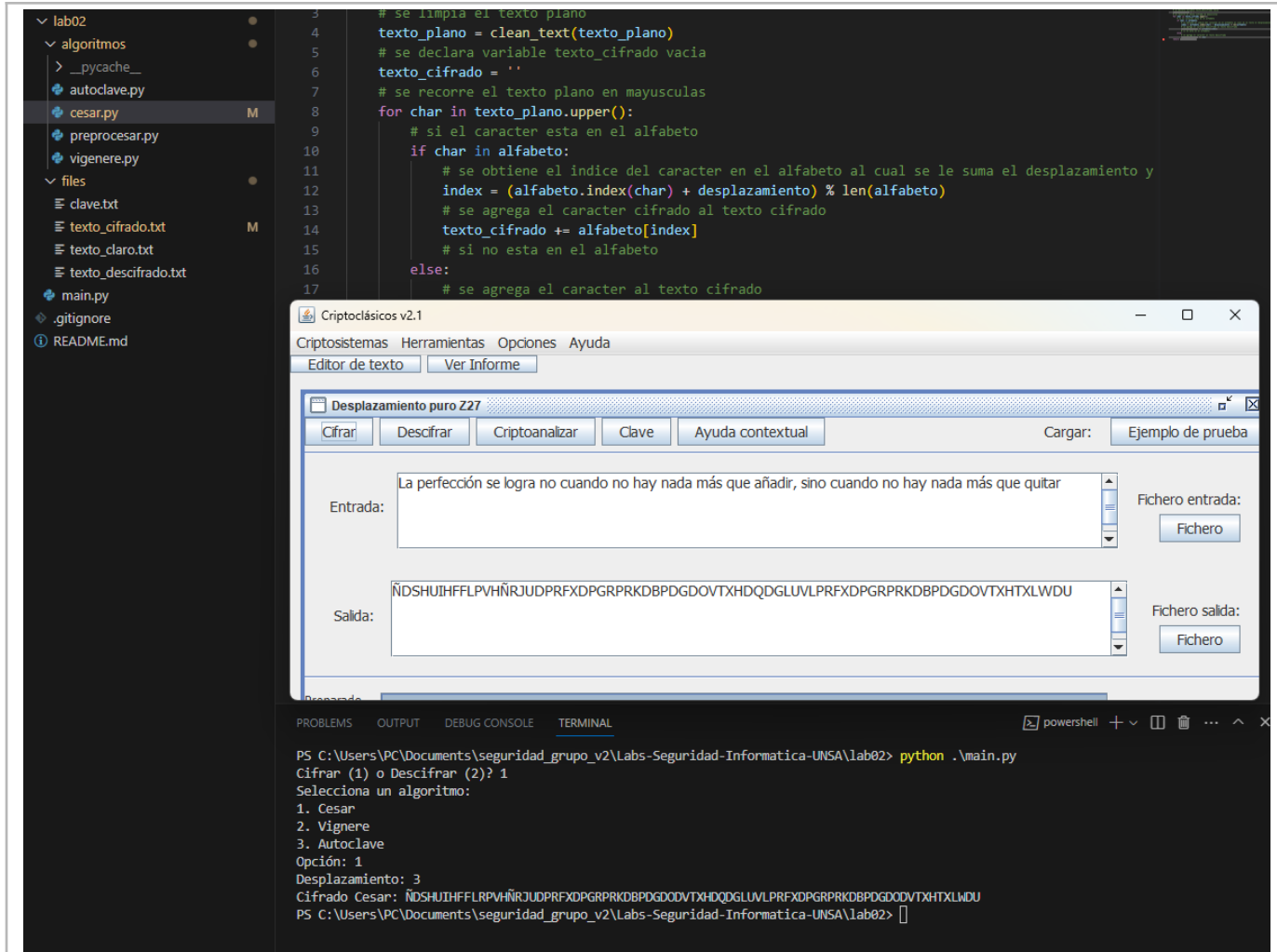
Cifrado de César

1. Implementar un algoritmo para construir un cifrador desplazamiento, basado en el cifrado de César, donde el desplazamiento sea elegible, trabajando sobre un alfabeto módulo 27. Se debe ingresar el texto claro (en archivo o por interface) y genere la cifra resultante

Codigo:

```
from .preprocesar import clean_text, ALPHABET
def cifrar_cesar(texto_plano, desplazamiento, alfabeto=ALPHABET):
    # se limpia el texto plano
    texto_plano = clean_text(texto_plano)
    # se declara variable texto_cifrado vacia
    texto_cifrado = ''
    # se recorre el texto plano en mayusculas
    for char in texto_plano.upper():
        # si el caracter esta en el alfabeto
        if char in alfabeto:
            # se obtiene el indice del caracter en el alfabeto al cual se le suma el desplazamiento y
            # se obtiene el modulo
            index = (alfabeto.index(char) + desplazamiento) % len(alfabeto)
            # se agrega el caracter cifrado al texto cifrado
            texto_cifrado += alfabeto[index]
            # si no esta en el alfabeto
        else:
            # se agrega el caracter al texto cifrado
            texto_cifrado += char
    return texto_cifrado
def descifrar_cesar(texto_cifrado, desplazamiento, alfabeto=ALPHABET):
    # se limpia el texto cifrado
    texto_cifrado = clean_text(texto_cifrado)
    # se declara variable texto_descifrado vacia
    texto_descifrado = ''
    # se recorre el texto cifrado en mayusculas
    for char in texto_cifrado.upper():
        # si el caracter esta en el alfabeto
        if char in alfabeto:
            # se obtiene el indice del caracter en el alfabeto al cual se le resta el desplazamiento y
            # se obtiene el modulo
            index = (alfabeto.index(char) - desplazamiento) % len(alfabeto)
            # se agrega el caracter descifrado al texto descifrado
            texto_descifrado += alfabeto[index]
            # si no esta en el alfabeto:
        else:
            # se agrega el caracter al texto descifrado
            texto_descifrado += char
    return texto_descifrado
```

Pruebas:



The screenshot displays a development environment with a file explorer on the left showing a project named 'lab02' containing files like 'cesar.py', 'preprocesar.py', and 'vigenere.py'. The main editor shows a Python script implementing a Caesar cipher. The script takes a plain text, cleans it, and then iterates through each character, shifting it by a specified displacement (3) if it is an uppercase letter. The output is a ciphertext. Below the code editor, a window titled 'Criptoclásicos v2.1' is open, showing a graphical interface for encryption. The 'Entrada' (Input) field contains the text 'La perfección se logra no cuando no hay nada más que añadir, sino cuando no hay nada más que quitar'. The 'Salida' (Output) field shows the resulting ciphertext: 'NDSHUIHFFLPVHNRJUDPRFXDPGRPRKDBPDGDOVTXHDQDGLUPLPRFXDPGRPRKDBPDGDOVTXHTXLWDU'. The interface also includes buttons for 'Cifrar' (Encrypt), 'Descifrar' (Decrypt), 'Criptoanalizar' (Cryptanalyze), and 'Clave' (Key).

2. Verificar cifrando “La perfección se logra no cuando no hay nada más que añadir, sino cuando no hay nada más que quitar” Usando un desplazamiento puro de 3
3. Verificar el resultado obtenido a partir del cifrador Criptoclásicos v2.1 (https://www.criptored.es/software/sw_m001c.htm) haciendo las capturas de pantalla respectivas para los datos indicados e incorporando el proceso de cifrado
4. Identifique en el cypher cuál es el carácter que aparece mayor cantidad de veces y demuestra que este representa a la letra “e”, use la herramienta seleccionando Criptoanalizar.

Criptoclásicos v2.1

Criptosistemas Herramientas Opciones Ayuda

Editor de texto Ver Informe

Criptool Analisis Desplazamiento puro Z27

Frecuencias observadas en el criptograma

Carácter	Frecuencia	Num Apa...
A	17.105%	13
N	11.842%	9
O	9.211%	7
D	6.579%	5
U	6.579%	5
E	6.579%	5
S	5.263%	4
I	5.263%	4
C	5.263%	4
R	5.263%	4
Q	3.947%	3
M	2.632%	2
Y	2.632%	2
H	2.632%	2
L	2.632%	2
T	1.316%	1
P	1.316%	1
G	1.316%	1
F	1.316%	1
Ñ	1.316%	1

Frecuencias del alfabeto Español

Carácter	Frecuencia
E	13.65%
A	11.797%
O	9.195%
S	7.983%
R	6.696%
N	6.69%
I	6.86%
L	5.27%
D	5.19%
C	4.919%
T	4.802%
P	3.445%
U	3.996%
M	2.925%
B	1.0%
F	0.953%
V	0.693%
Q	0.875%
G	1.043%
H	0.585%
J	0.272%
Ñ	0.074%
K	0.022%
W	0.019%
X	0.183%
Y	0.523%
Z	0.291%

Par seleccionado (Texto - alfabeto)

Relación 1: -- Limpiar dato

Desplazamiento:

Texto Cifrado

LAPERFECCINSELOGRANOCUANDONOHAYNADAMSQUEAÑADIR

5. Proponga un algoritmo de descifrado y demuestre su funcionamiento correcto comparando con el resultado mostrado en la herramienta.

```
PS C:\Users\PC\Documents\seguridad_grupo_v2\Labs-Seguridad-Informatica-UNSA\lab02> python .\main.py
Cifrar (1) o Descifrar (2)? 2
Selecciona un algoritmo:
1. Cesar
2. Vignere
3. Autoclave
Opción: 1
Desplazamiento: 3
Descifrado Cesar: LAPERFECCINSELOGRANOCUANDONOHAYNADAMSQUEAÑADIR SINOCUANDONOHAYNADAMSQUEQUITAR
PS C:\Users\PC\Documents\seguridad_grupo_v2\Labs-Seguridad-Informatica-UNSA\lab02> |
```



UNIVERSIDAD NACIONAL DE SAN AGUSTIN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 5

Cifrado de Vignere

6. Implementar un algoritmo para construir un cifrador de Vignere, donde se pueda seleccionar el módulo, alfabeto módulo 27 o módulo 191 (ASCII), ingresar el texto claro (en archivo o por interface) y genere la cifra resultante.

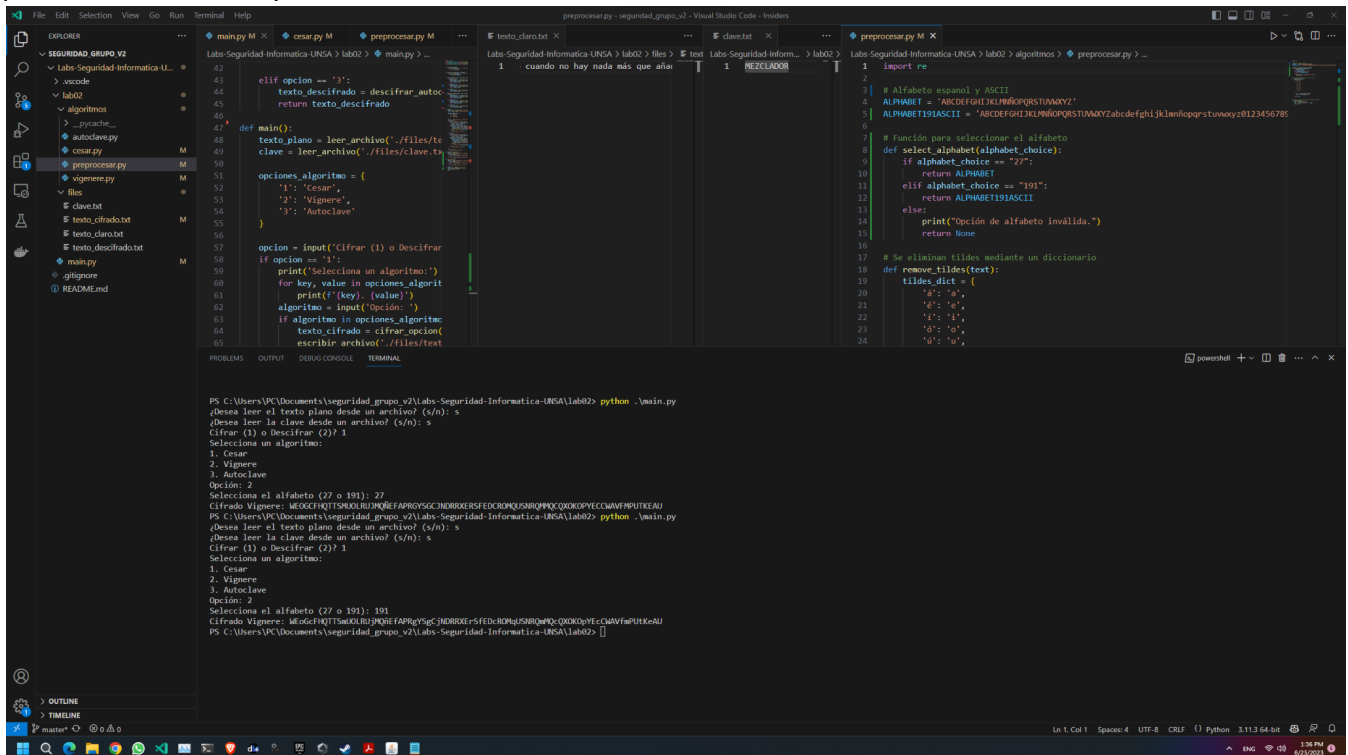
```
# Importar la librería re para utilizar expresiones regulares
import re
# Importar todas las funciones de preprocesar.py
from .preprocesar import *
# cifrar con vigenere
def cifrar_vigenere(text, key, alphabet_choice):
    # Seleccionar el alfabeto
    alphabet = select_alphabet(alphabet_choice)
    # Si el alfabeto es inválido, retornar None
    if alphabet is None:
        return None
    text = text.upper() # Convertir el texto a mayúsculas
    key = key.upper() # Convertir la clave a mayúsculas
    text = clean_text(text) # Limpiar el texto de caracteres no deseados
    key = clean_text(key) # Limpiar la clave de caracteres no deseados

    cyphered_text = '' # Inicializar el texto cifrado
    i = 0
    # Recorrer el texto de acuerdo a la longitud del texto cifrado
    while i < len(text):
        # si el caracter esta en el alfabeto
        if text[i] in alphabet:
            # Obtener el índice del carácter actual en el texto
            index_text = alphabet.index(text[i])
            # Obtener el índice del carácter actual en la clave (repetir la clave si es más corta que
            el texto)
            index_key = alphabet.index(key[i % len(key)])
            # Aplicar el cifrado de Vigenère y agregar el carácter cifrado al texto cifrado
            cyphered_text += alphabet[(index_text + index_key) % len(alphabet)]
        else:
            # Si el carácter no está en el alfabeto, se agrega al texto cifrado sin cifrar
            cyphered_text += text[i]
            # se incrementa el contador
            i += 1
    # retornar el texto cifrado
    return cyphered_text

# decifrar con vigenere
def descifrar_vigenere(cyphered_text, key, alphabet_choice):
    # Seleccionar el alfabeto
    alphabet = select_alphabet(alphabet_choice)
    # Si el alfabeto es inválido, retornar None
    if alphabet is None:
        return None
    # Convertir la clave a mayúsculas
    key = key.upper()
    # Convertir el texto cifrado a mayúsculas
    cyphered_text = cyphered_text.upper()
    # Inicializar el texto descifrado
    text = ''
    # Inicar el contador en 0
    i = 0
    # Recorrer el texto cifrado de acuerdo a la longitud del texto cifrado
    while i < len(cyphered_text):
        # si el caracter esta en el alfabeto
        if cyphered_text[i] in alphabet:
            # Obtener el índice del carácter cifrado actual en el texto cifrado
            index_cyphered_text = alphabet.index(cyphered_text[i])
            # Obtener el índice del carácter actual en la clave (repetir la clave si es más corta que
            el texto cifrado)
            index_key = alphabet.index(key[i % len(key)])
            # Aplicar el descifrado de Vigenère y agregar el carácter descifrado al texto descifrado
            text += alphabet[(index_cyphered_text - index_key) % len(alphabet)]
        else:
            # Si el carácter no está en el alfabeto, se agrega al texto descifrado sin descifrar
            text += cyphered_text[i]
            # se incrementa el contador
            i += 1
    # retornar el texto descifrado
    return text
```


7. Verificar cifrando “La perfección se logra no cuando no hay nada más que añadir, sino cuando no hay nada más que quitar” Usando la clave MEZCLADOR

8. Verificar el resultado obtenido a partir del cifrador Criptoclásicos v2.1 (https://www.criptored.es/software/sw_m001c.htm) haciendo las capturas de pantalla respectivas para los módulos 27 y 191



Resultado modulo 27:

```
PS C:\Users\PC\Documents\seguridad_grupo_v2\Labs-Seguridad-Informatica-UNSA\lab02> python .\main.py
¿Desea leer el texto plano desde un archivo? (s/n): s
¿Desea leer la clave desde un archivo? (s/n): s
Cifrar (1) o Descifrar (2)? 1
Selecciona un algoritmo:
1. Cesar
2. Vignere
3. Autoclave
Opción: 2
Selecciona el alfabeto (27 o 191): 27
Cifrado Vignere: WEOGCFHQTTSMUOLRUJMQÑEFAPRGYSGCJNDRRXERSFEDCROMQUSNRQMMQCQXOKOPYECCWAVFMPUTKEAU
```

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 8

Vigenère Z27

Cifrar

Descifrar

Criptoanalizar

Clave

Ayuda contextual

Cargar: Ejemplo de prueba

Entrada:

La perfección se logra no cuando no hay nada más que añadir, sino cuando no hay nada más que quitar

Fichero entrada:

Fichero

Salida:

WEOGCFHQTTSMUOLRUJMQÑEFAPRGYSGCJNDRRXERSFEDCROMQUSNRQMMQCQXOKOPYECCWAVFMPUTKEAU

Fichero salida:

Fichero

Preparado

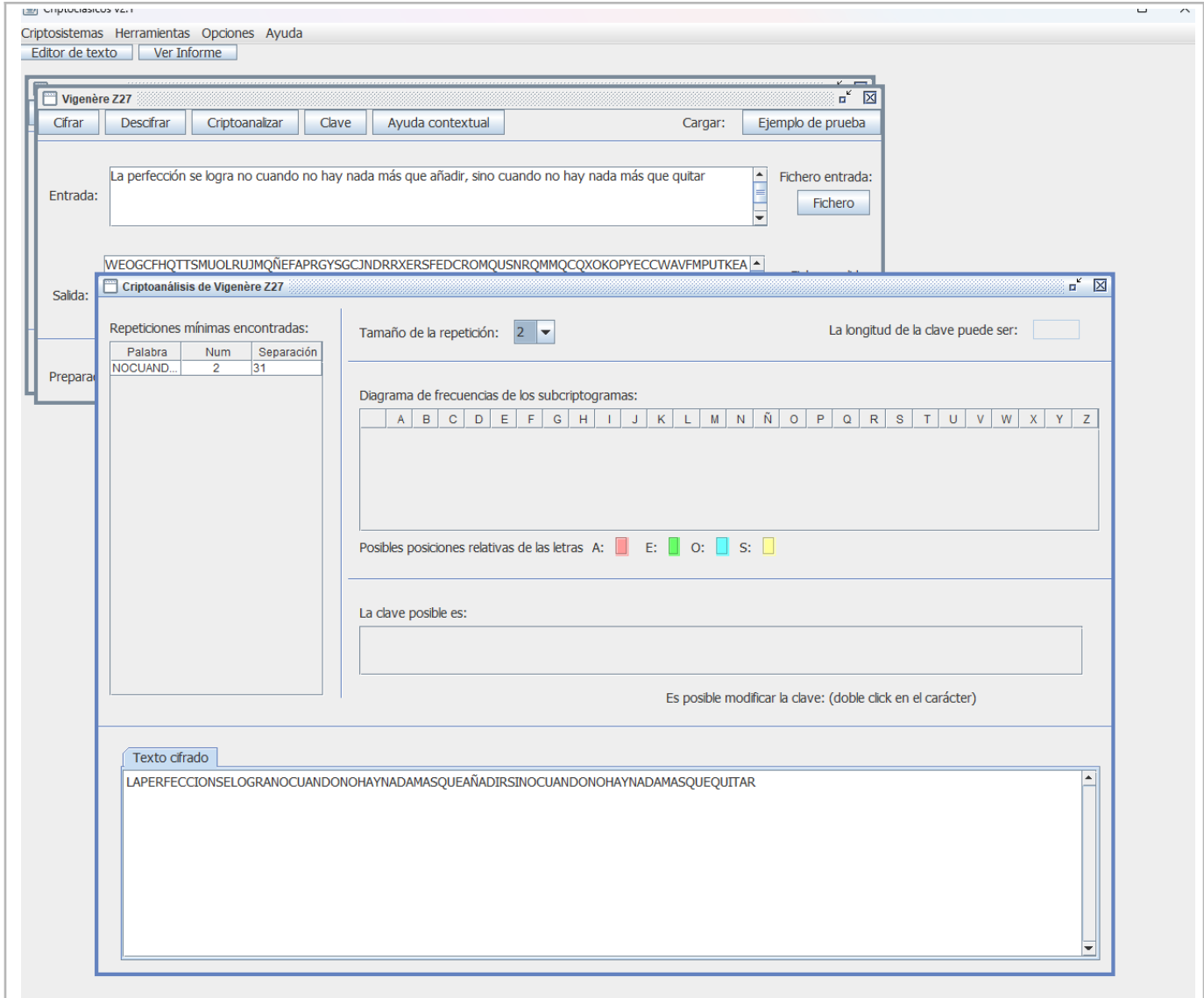
100%

Resultado modulo 191:

```
PS C:\Users\PC\Documents\seguridad_grupo_v2\Labs-Seguridad-Informatica-UNSA\lab02> python .\main.py
¿Desea leer el texto plano desde un archivo? (s/n): s
¿Desea leer la clave desde un archivo? (s/n): s
Cifrar (1) o Descifrar (2)? 1
Selecciona un algoritmo:
1. Cesar
2. Vignere
3. Autoclave
Opción: 2
Selecciona el alfabeto (27 o 191): 191
Cifrado Vignere: WEoGcFHQTTsmUOLRUjMQÑEfAPRGySGcJNDRRXErSFEDcROMqUSNRQmMQcQXOKOpYEccWAVfmPUtKeAU
```

No sale la opcion 191 en el criptoclasicos v2.1.jar

9. Muestra las frecuencias de cada letra del mensaje original usando como claves MALEFICIO, QUESO y MIA, compare y comente sobre la variación de las frecuencias en base a la longitud de la clave



La herramienta no muestra las cantidad de veces que se repiten texto

10. Desarrolle un algoritmo que encuentre el texto claro si recibió la cifra

Cipher_text :

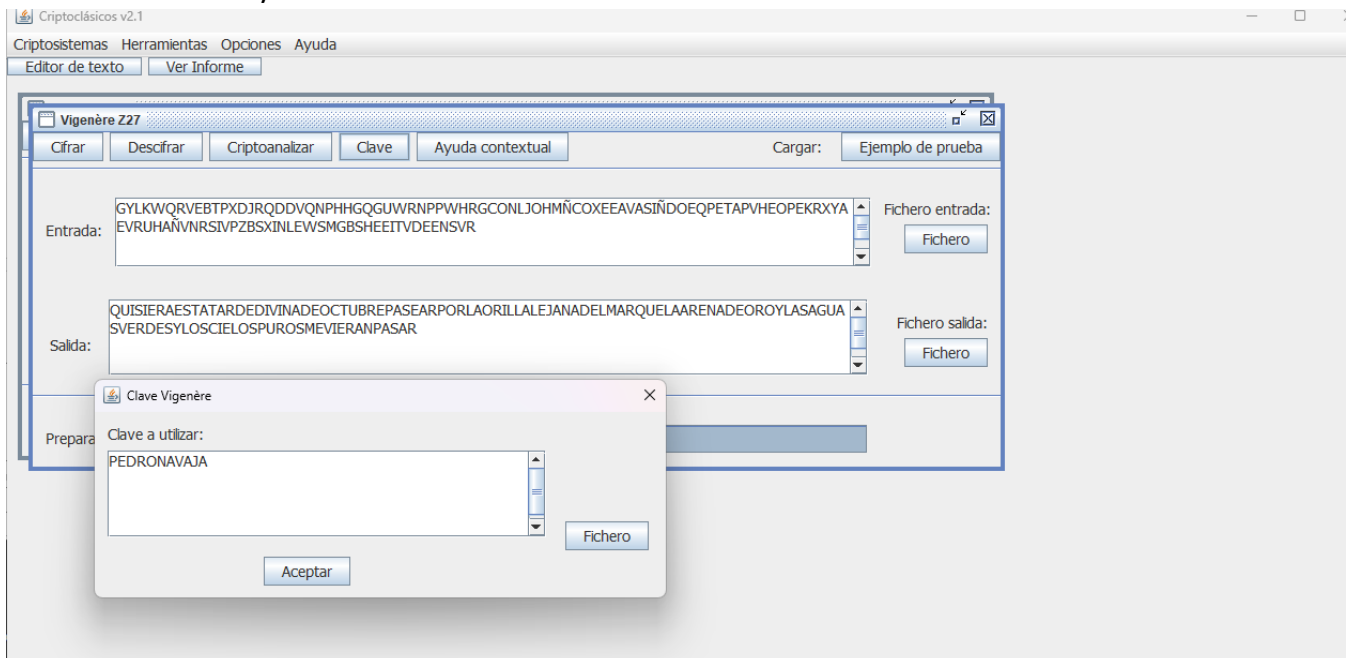
GYLKWQRVEBTPXDJRQDDVQNPHHGQGUGWRNPPWHRGCONLJOHMÑCOXEEAVASIÑDOEQPETAPVHEOPE
 KRXYAEVRUHAÑVNRSIVPZBSXINLEWSMGBSHEEITVDEENSVR

y se sabe que ha cifrado con la clave PEDRONAVAJA

Prueba de código:

```
Cifrar (1) o Descifrar (2)? 2
Selecciona un algoritmo:
1. Cesar
2. Vignere
3. Autoclave
Opción: 2
¿Desea leer la clave desde un archivo? (s/n): n
Ingrese la clave: PEDRONAVAJA
¿Desea leer el texto cifrado desde un archivo? (s/n): n
Ingrese el texto cifrado: GYLKWQVEBTPXDJRQDDVQNPVHGQGUWRNPPWHRGCONLJOHMHCOXEEAVASINDOEQPETAPVHEOPEKRXYAEVRUHAHVNRISVPZBSXINLEWSMGBSHEETVDEENSVR
Selecciona el alfabeto (27 o 191): 27
Descifrado Vignere: QUISIERAESTATARDEDIVINADEOCTUBREPASEARPORLAORILLALEJANADELMARQUELAARENADEOROYLASAGUASVERDESYSOSCIELOS PUROSMEVIERANPASAR
PS C:\Users\PC\Documents\seguridad_grupo_v2\Labs-Seguridad-Informatica-UNSA\lab02>
```

11. Usando el software anterior, verifique el resultado, eligiendo el cifrado Vignere con módulo 27, muestre el informe y comente



Se cumple de manera satisfactoria el descifrado con vigenere, a través de nuestra interfaz se eligieron las opciones de descifrado con vigenere luego directamente también se eligen las opciones para ingresar datos desde consola , al final también se elige el alfabeto 27 a usarse, dando como resultado el mismo resultado en descifrado.

12. Usando matemáticas discretas, descifre manualmente YGVMSSKKOX si la clave fue FORTALEZA en un alfabeto de 27 caracteres

Texto Cifrado	Y	Q	V	M	S	S	K	K	O	X
---------------	---	---	---	---	---	---	---	---	---	---

Clave Extendida	F	O	R	T	A	L	E	Z	A	F
Valor numérico (texto cifrado)	24	6	21	12	18	18	10	10	14	23
Valor numerico (clave)	5	14	17	19	18	7	6	11	14	18
Valor numerico(texto original)	19	18	4	19	18	7	6	11	14	18

El último valor se reemplaza con letras,

Convirtiendo de numeros a letras > texto original: THISISATEST

Cifrado con autoclave

13. Descifra el texto, usando la clave UNODELOSMASGRANDESCRIPTOGRAFOS:

Cipher_text:

XHGDQESDMPKÑDEEDKNGJZPFJSUIFZOLFCINFJCESVZTGBFXCIUDAYNUUDIZYWWZBEYNNVQWIVUNKZEP
HDODQUZZLBDNDRWTHQSERÑIVMLERCMGIFLSORZXTSDIGLOXQSDJHWVCIWQXQJCKMBPOKMPSKMU
VIMNJDNBLCSSZHXHNYYUIXDBSOXHSLXWVGJGXHWLTDWKÑSAQIMZLNBVMLXHUOQQXIQQGWGUFTW
KZKMOKUDNINSIFJDUOZIJBSVVOWFAIEÑGYOWPSOAP

```
PS C:\Users\PC\Documents\seguridad_grupo_v2\labs-Seguridad-Informatica-UNSA\lab02> python .\main.py
Cifrar (1) o Descifrar (2)? 2
Selecciona un algoritmo:
1. Cesar
2. Vignere
3. Autoclave
Opción: 3
¿Desea leer la clave desde un archivo? (s/n): n
Ingrese la clave: UNODELOSMASGRANDESCRIPTOGRAFOS
¿Desea leer el texto cifrado desde un archivo? (s/n): n
Ingrese el texto cifrado: XHGDQESDMPKÑDEEDKNGJZPFJSUIFZOLFCINFJCESVZTGBFXCIUDAYNUUDIZYWWZBEYNNVQWIVUNKZEP
HDODQUZZLBDNDRWTHQSERÑIVMLERCMGIFLSORZXTSDIGLOXQSDJHWVCIWQXQJCKMBPOKMPSKMU
VIMNJDNBLCSSZHXHNYYUIXDBSOXHSLXWVGJGXHWLTDWKÑSAQIMZLNBVMLXHUOQQXIQQGWGUFTW
KZKMOKUDNINSIFJDUOZIJBSVVOWFAIEÑGYOWPSOAP
Descifrado Autoclave: DURANTE LAPRIMERAGUERRAMUNDIALMILLAMPREDERICKFREDMAN SIRVIOCOMOTENIENTE EN LA UNIDAD DE CRIPTOLOGIA DE LA FUERZA EXPEDICIONARIA ESTADOUNIDENSE JISTINGUIENDOSE POR SUS TRABAJOS Y PROEZAS EN LA LATA DE CODIGO SE NEPLOGO INVOLOABLES POR LO CU
AL RECTOR DEL RECONOCIMIENTO DEL GOBIERNO ESTADOUNIDENSE.
```

```
# importar funciones de preprocesamiento
from .preprocesar import clean_text, ALPHABET

# cifrado por autoclave
def cifrar_autoclave(text, key, alphabet=ALPHABET):
    # se convierte el texto y la clave a mayúsculas
    text = text.upper()
    key = key.upper()
    # se limpia el texto y la clave
    text = clean_text(text)
    key = clean_text(key)
    # se declara variable texto_cifrado vacia
    cyphered_text = ''
    # se inicializa el indice en 0
    i = 0
    # se recorre el texto desde el indice 0 hasta el tamaño del texto
    while i < len(text):
        # si el caracter esta en el alfabeto
        if text[i] in alphabet:
            # se obtiene el indice del caracter en el alfabeto al cual se le suma el desplazamiento y
            # se obtiene el modulo
            text_index = alphabet.index(text[i])
            # se obtiene el indice del caracter y se busca [i % len(key)] para que no se salga de rango
            key_index = alphabet.index(key[i % len(key)])

            if i > len(key)-1:
                key_index = alphabet.index(text[(i - len(key)) % len(text)])

            cyphered_text += alphabet[(text_index + key_index) % len(alphabet)]
        else:
            # Si el carácter no está en el alfabeto, se agrega directamente al texto cifrado
            cyphered_text += text[i]

        # Incrementa el índice para avanzar al siguiente carácter del texto
        i += 1

    return cyphered_text

# descifrado con autoclave
def descifrar_autoclave(cyphered_text, key, alphabet=ALPHABET):
    # se convierte el texto cifrado y la clave a mayúsculas
    cyphered_text = cyphered_text.upper()
    key = key.upper()

    # se limpia el texto cifrado y la clave
    cyphered_text = clean_text(cyphered_text)
    key = clean_text(key)

    # se declara variable texto vacia
    text = ''
    # se inicializa el indice en 0
    i = 0
    # se recorre el texto cifrado desde el indice 0 hasta el tamaño del texto cifrado
    while i < len(cyphered_text):
        # si el caracter cifrado esta en el alfabeto
        if cyphered_text[i] in alphabet:
            # se obtiene el indice del caracter cifrado en el alfabeto
            text_index = alphabet.index(cyphered_text[i])
            # se obtiene el indice del caracter de la clave y se busca [i % len(key)] para que no se
            # salga de rango
            key_index = alphabet.index(key[i % len(key)])

            if i > len(key)-1:
                # Utiliza caracteres del texto descifrado para generar el índice de la clave
                key_index = alphabet.index(text[(i - len(key)) % len(text)])

            # Calcula el índice resultante restando los índices del texto cifrado y de la clave, y
            # luego tomando
            # el módulo con respecto a la longitud del alfabeto para asegurar que no se salga de rango
            text += alphabet[(text_index - key_index) % len(alphabet)]
        else:
            # Si el carácter cifrado no está en el alfabeto, se agrega directamente al texto descifrado
            text += cyphered_text[i]

        # Incrementa el índice para avanzar al siguiente carácter del texto cifrado
        i += 1

    return text
```

Ataque de Kasiski

14. Criptoanalizar el siguiente criptograma mod 27, encontrar la clave y el texto en claro.

Cipher_text:

MAXYHGAVAPUUGZHEGZQOWOBNIPQKRÑMEXIGONIICUCAWIGCTEAGMNOLRSZJNLWÑAWWIGLDDZ
SNIZDNBIXGZLAYMXÑCVEKIETMOEOPBEWPTNIXCXUIHMECXLNOCECYXEQPBWUFANIICÑJIKISCZUAILBG
SOANKBFWUAYWNSCHLCWYDZHDZAQVMPTVGFGPVAJWFVPUOYMXCWERVLQCZWECIFVITUZSNCZUAI
KBFMÑALIEGLBSZLQUXÑOHWOCGHNYWÑQKDANZUDIFOIMXNPHNUWQOKLMVBNNKRMKONDPDPN
MIKAWOXMEEIVEKGBGSFHVADWPGOYMHIOUEEIPGOLENZBSCHAGKQTZDRÑMÑNWTUZIÑCMÑAXKQU
WDLVANNIHLÑCQNWGEHIPGZDTZTÑNWÑEEWFUMGIÑXNTWXNVIXCZOAZSOQUVENDNFWUSZYHGLR
ACPGGUGIYWHOTRMZUGQQDDZIZFWHVSHCUGOGIFKBXAPBOBRDVDUCMVTGKIKDRSZLUQSDVPM
XVIVEYMFGEANIMQLHLGPQOHRYWCFEWFOWISNÑPUAYINNÑXNÑPGKWGOILQGAFOILQTAHEIIDWMÑE
ÑXNEPRCVDQTURSK

II. SOLUCIÓN DEL CUESTIONARIO

1. En el ataque a Vigenere por Kasiski ¿Qué buscamos preferentemente?

En el ataque a Vigenère por Kasiski, buscamos principalmente secuencias de letras que se repiten en el texto cifrado. Estas repeticiones pueden indicar que una misma secuencia de letras en el texto plano ha sido cifrada por la misma secuencia de la clave. El método de Kasiski utiliza estas repeticiones para deducir la longitud de la clave.

2. Encontradas las cadenas repetidas en el criptograma, con separación d1, d2, d3 y d4 ¿Cuál sería la longitud L de la clave?

Encontradas las cadenas repetidas en el criptograma, con separación d1, d2, d3 y d4, la longitud L de la clave sería el máximo común divisor (MCD) de esas distancias. Esto se debe a que es probable que las cadenas repetidas correspondan a la misma secuencia del texto plano cifrada por la misma secuencia de la clave.

3. Si las distancias entre repeticiones de cadenas en un criptograma son 35, 112, 70. ¿Cuál sería la longitud L de la clave?

Si las distancias entre repeticiones de cadenas en un criptograma son 35, 112, 70, entonces la longitud L de la clave sería el MCD de esas distancias. En este caso, el MCD de 35, 112 y 70 es 7, por lo tanto, la longitud de la clave sería 7.

4. ¿Qué diferencia tiene la regla AEOS de AEO en Kasiski?

La regla AEOS de Kasiski es una extensión de la regla AEO. Ambas reglas se refieren a las frecuencias relativas de las letras en los idiomas. En inglés, las letras más comunes son E, T, A, O, I, N, S, H, R, D y L. En la regla AEO, solo se consideran las letras A, E y O, mientras que en la regla AEOS se consideran también la S. Estas reglas se usan para realizar un análisis de frecuencia en el texto cifrado y ayudar a descifrar la clave.



III. CONCLUSIONES

En conclusión, he desarrollado una comprensión detallada de los algoritmos de cifrado César, Vigenère y Autoclave, y he implementado exitosamente sus funciones de cifrado y descifrado en Python. Estos métodos de cifrado son importantes en la historia de la criptografía y proporcionan una base sólida para comprender conceptos más avanzados. Sin embargo, es importante tener en cuenta que estos métodos son vulnerables a varios tipos de ataques criptográficos, como el método de Kasiski en el caso de Vigenère. En la actualidad, con la potencia de cálculo y los algoritmos sofisticados disponibles, estos métodos de cifrado pueden ser descifrados con relativa facilidad. Para proteger los datos en el mundo digital actual, se requieren métodos de cifrado más robustos y complejos, como el cifrado de clave pública RSA o el cifrado simétrico AES. .

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA

Link archivos : <https://github.com/franquic23/Labs-Seguridad-Informatica-UNSA/tree/main/lab02>