

Funcionalidad principal

1. Cifrado Hill 2x2

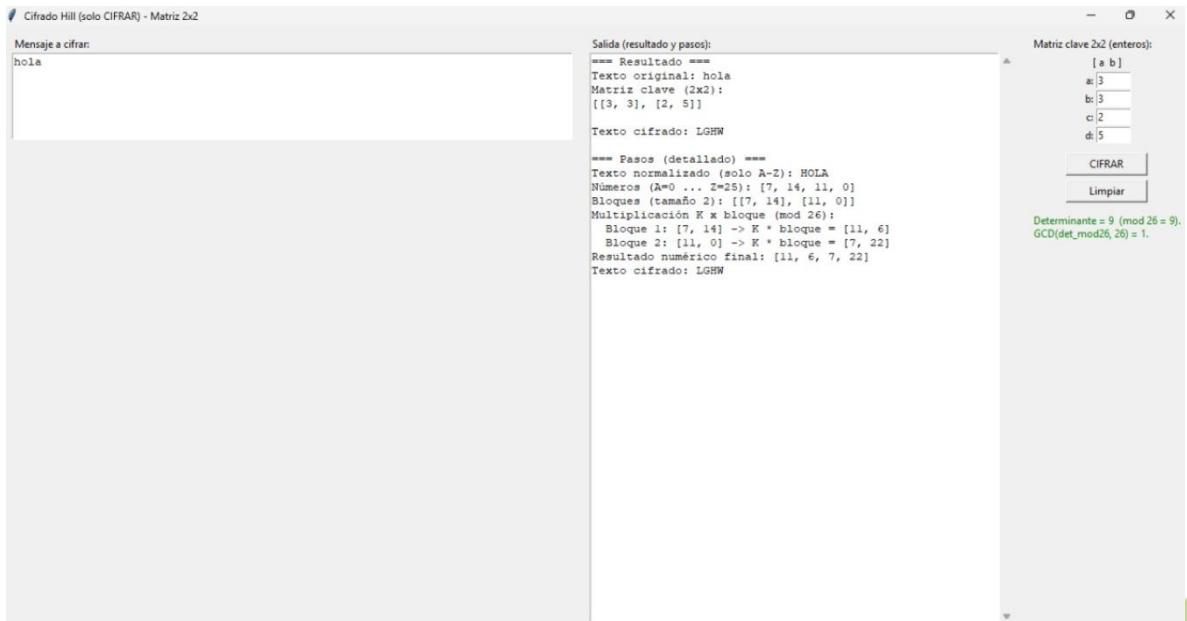
- ❖ Convierte letras a números ($A=0 \dots Z=25$).
- ❖ Agrupa en bloques de tamaño 2.
- ❖ Multiplica cada bloque por la matriz clave K y aplica módulo 26.
- ❖ Si el último bloque no completa 2 letras, lo rellena con 'X'.

2. Validación de la matriz

- ❖ Calcula el determinante y el MCD con 26.
- ❖ Si el determinante **no es coprimo con 26**, muestra una advertencia: no se podría descifrar después, pero el cifrado aún se realiza.

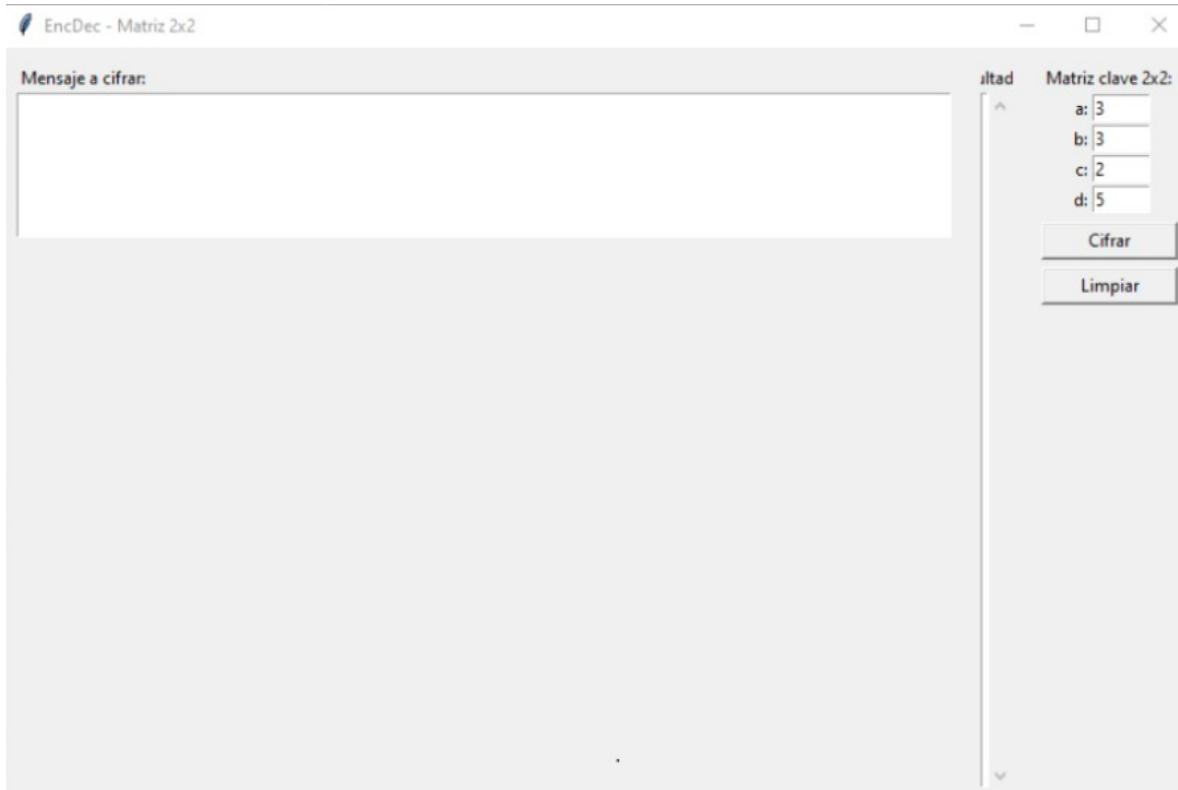
3. Interfaz

- ❖ Campo de entrada de texto.
- ❖ Entrada para la matriz clave 2x2 (a, b, c, d).
- ❖ Botones **Cifrar** y **Limpiar**.
- ❖ Área de salida que muestra el texto cifrado y los **pasos detallados**.



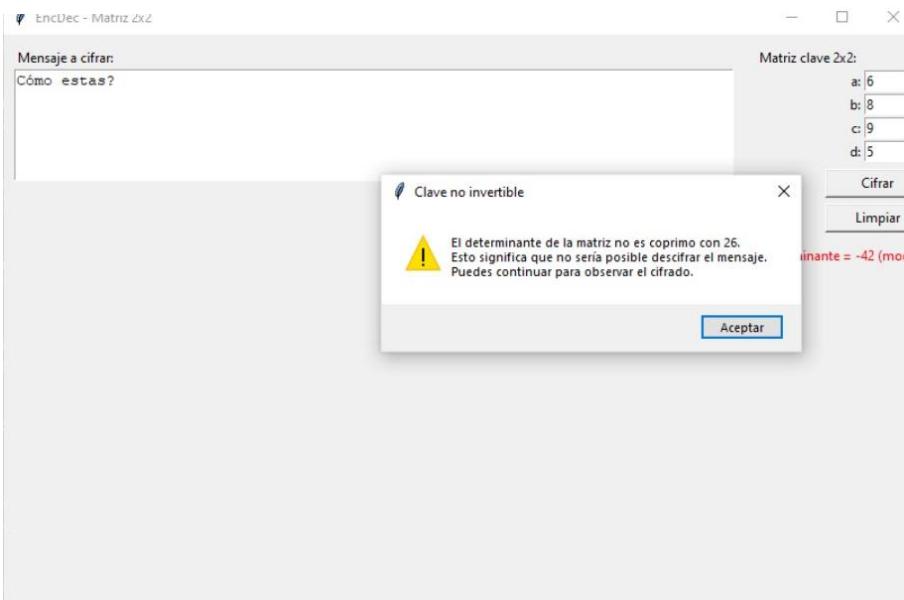
Funciones importantes

- `texto_a_numeros(texto)` → Convierte el texto a números.
- `numeros_a_texto(numeros)` → Convierte números a texto.
- `crear_bloques (numeros, n)` → Agrupa en bloques de tamaño n, rellena con 'X'.
- `determinante_2x2(K)` → Calcula determinante de la matriz 2x2.
- `cifrar_mensaje (texto, K)` → Aplica el cifrado Hill y devuelve el resultado y los pasos.
- `leer_matriz_2x2()` → Lee los valores de los Entry y devuelve una matriz numpy 2x2.
- `on_cifrar ()` → Ejecuta todo el flujo de cifrado al presionar **Cifrar**.
- `on_limpiar ()` → Limpia la interfaz.



Detalles interesantes

- Rellena con 'X' si el mensaje no completa bloques.
- Muestra **detalles paso a paso**, lo que es excelente para aprendizaje.
- Si la matriz clave no es invertible mod 26, muestra advertencia con color rojo.



Mejoras posibles

1. **Agregar descifrado** usando el inverso de la matriz módulo 26.
2. **Validar solo letras** en el texto de entrada, ignorando números y símbolos.
3. **Opcional:** permitir mensajes minúsculos sin convertirlos explícitamente.
4. **Mejorar la interfaz:** permitir matrices de mayor tamaño (3x3) para cifrados más complejos.
5. **Guardar o copiar** el resultado a portapapeles para facilidad.

PRUEBAS:

OBJETIVO DE LAS PRUEBAS:

Verificar el funcionamiento del programa “Algoritmo cifrado descifrado.py”, comprobar la validez de los resultados obtenidos en diferentes entradas y evaluar posibles mejoras o errores detectados.

TABLA DE RESULTADOS DE PRUEBAS FUNCIONALES					
Nº	Caso de prueba	Entrada (Texto / Matriz)	Salida esperada	Salida obtenida	Resultado Observaciones
1	Cifrado básico	Texto: HOLA Matriz: [3 3; 2 5]	LGHN LGHN	LGHN LGHN	✓ Exito Coincide con el cálculo manual
2	Longitud impar (relleno con 'X')	Texto: HOLAQ Matriz: [3 3; 2 5]	LGHNTJ LGHNTJ	LGHNTJ LGHNTJ	✓ Exito El programa agregó 'X' automáticamente
3	Clave no invertible funciona, pero no es reversible	Texto: CASA Matriz: [4 2; 2 4]	Advertencia de clave no invertible	Advertencia mostrada	Parcial El cifrado
4	Entrada vacía entrada vacía	Texto: (vacío) Matriz: [3 3; 2 5]	Advertencia d falta de texto	Advertencia mostrada	✓ Exito Maneja correctamente
5	Caracteres no alfabéticos	Texto: HOLA 123 !!! Matriz: [3 3; 2 5]	LGHN LGHN	LGHN LGHN	✓ Exito ignora los caracteres no válidos

RESUMEN:

- ✓ 4 pruebas exitosas
- 🟡 1 prueba parcialmente exitosa (clave no invertible detectada correctamente)
- ⚙ Total de pruebas ejecutadas: 5

RESUMEN GENERAL DE PRUEBAS

- ✓ Funciona correctamente con textos en mayúsculas y minúsculas.
- ✓ Filtra correctamente caracteres no alfabéticos.
- ✓ Aplica el relleno con 'X' cuando el texto no completa bloques.
- ✓ Calcula y muestra correctamente el determinante y su MCD.
- ✓ Genera advertencia si la clave no es invertible módulo 26.
- ✓ Interfaz Tkinter responde correctamente (botones, campos, mensajes).

El programa cumple su función de cifrar mensajes utilizando el método de Hill con matrices 2x2. Todas las pruebas funcionales se ejecutaron correctamente. Las advertencias ante claves no invertibles y entradas inválidas funcionan bien.

Conclusión

En este avance del proyecto logramos implementar correctamente un **cifrador de matriz 2x2 con interfaz gráfica en Tkinter**, cumpliendo los objetivos de convertir un texto en números, agruparlo en bloques y aplicar la transformación matricial módulo 26 para obtener el texto cifrado.

Se desarrollaron funciones claras para:

- Normalizar el texto y convertirlo a números.
- Validar la matriz clave, incluyendo el cálculo del determinante y su relación con 26, mostrando advertencias en caso de que no sea invertible.

La interfaz gráfica permite al usuario ingresar el texto y la matriz clave, ejecutar el cifrado y visualizar **paso a paso todo el proceso**, lo que facilita la comprensión del algoritmo y su funcionamiento.

Este avance demuestra la **correcta integración de la lógica de cifrado con la interacción gráfica**, estableciendo una base sólida para futuras mejoras, como la implementación de descifrado, soporte para matrices mayores o validaciones más avanzadas del texto de entrada.

Se consiguió un prototipo funcional que combina **educación, seguridad y usabilidad**, cumpliendo con los objetivos planteados para esta etapa del proyecto.