Trabajo Práctico Grupal
Mensajes secretos - Parte 1

Objetivo General

Escribir una aplicación que permita cifrar mensajes secretos para ser enviados y que no puedan ser descifrados, a excepción de ser el destinatario del mensaje.

Este programa permitirá encriptar un mensaje utilizando algoritmos criptográficos simples.

Planificación para el desarrollo de la aplicación

Para que puedan alcanzar el objetivo general exitosamente, los vamos a guiar en la construcción de la aplicación. Es importante que sigan las indicaciones aquí descritas y las del docente a cargo.

Deberán ir cumpliendo objetivos que los ayudarán a organizar el trabajo y alcanzar el objetivo final. Por ello, antes de explicar cómo será la interfaz del usuario y abordar su construcción, comenzarán por desarrollar los algoritmos básicos que van a utilizar para encriptar mensajes.

A medida que cumplan los objetivos, deberán subir el resultado al campus en la actividad correspondiente.

Objetivo 1: Cifrado César

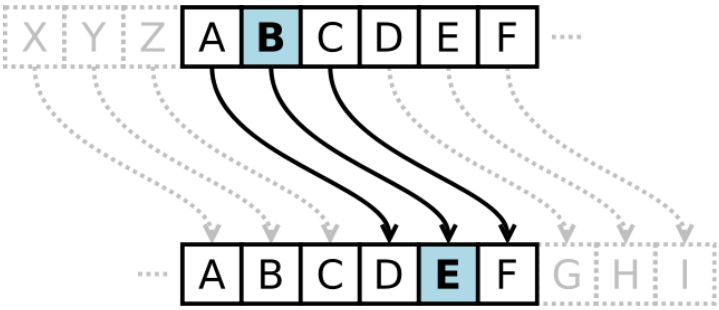
➔ *Escribir una función que reciba el mensaje a cifrar (cadena de caracteres) y la clave de cifrado, y devuelva el mensaje cifrado, mediante el cifrado césar. Probarla utilizando doctest, con al menos 10 casos diferentes.*

El **cifrado de César** sustituye cada letra del mensaje por otra. La letra sustituida se obtiene "desplazando" cada letra del abecedario una cierta cantidad de posiciones a la derecha. Esta cantidad de lugares que una letra se desplaza es la clave secreta que se debe pasar al receptor del mensaje para poder descifrarlo.

Ejemplo

Clave = 3

Tener una clave 3 significa que las letras del abecedario son desplazadas 3 lugares a la derecha. Por ejemplo la letra B se codifica como E, C como F, etc.



Para codificar un mensaje reemplazamos cada letra del mensaje por la letra desplazada.

Un mensaje completo quedaría entonces de la siguiente manera:

```
HOLA MUNDO
KROD PXQGR
```

Nótese que cada letra siempre se reemplaza de la misma manera. Por ejemplo la O siempre se reemplaza por la R

Si quieren leer sobre este cifrador y su historia recomendamos la wikipedia: Cifrado Cesar

Consideraciones

- Se deben considerar tanto las letras mayúsculas como minúsculas.
- Adapte el método para encriptar de forma similar los dígitos numéricos que pueda tener el mensaje.
- Los espacios y otros símbolos no se codifican, quedan igual.
- Sugerimos que lean sobre las funciones ord() y chr(). Cada letra tiene asignado un código numérico (llamado código ASCII) y letras consecutivas tienen números consecutivos.

Objetivo 2: Cifrado Atbash

➔ *Escribir una función que reciba el mensaje a cifrar (cadena de caracteres), y devuelva el mensaje cifrado, mediante el cifrado atbash. Probarla utilizando doctest, con al menos 10 casos diferentes.*

Este cifrador también reemplaza cada letra por otra pero en este caso en lugar de un desplazamiento se utiliza la versión invertida del abecedario. En nuestro caso, además cambiaremos mayúsculas por minúsculas, y viceversa.

Original	a	b	c	d	e	f	g	h	i	j	k	l	m	n	ñ	o	p	q	r	s	t	u	v	w	x	y	z
Clave	Z	Y	X	W	V	U	T	S	R	Q	P	O	Ñ	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Este cifrador no tiene una clave variable. Con que el receptor sepa que el emisor utiliza cifrado Atbash es suficiente para descifrarlo.

Ejemplo

Un mensaje completo quedaría entonces de la siguiente manera.

```
HOLA MUNDO
sloz ˜nfnwl
```

Si quieren leer sobre este cifrador y su historia recomendamos la wikipedia: Cifrado Atbash

Consideraciones

- Se deben considerar tanto las letras mayúsculas como minúsculas.
- Adapte el método para encriptar de modo similar, los dígitos numéricos que pueda tener el mensaje.
- Los espacios y otros símbolos no se codifican, quedan igual.

Objetivo 3: Interfaz Gráfica de usuario

➔ *Escribir el conjunto de funciones necesarias para implementar la interfaz gráfica que utilizará el usuario y que debe cumplir con los requerimientos descritos a continuación.*

Al iniciar el programa debe abrirse una ventana de bienvenida con el siguiente mensaje:

“Bienvenido a la aplicación de mensajes secretos del grupo [colocar el nombre del grupo]. Para continuar presione continuar, de lo contrario cierre la ventana”

- La ventana debe tener un ícono preferentemente diseñado por ustedes y el título debe ser “TP Grupal Parte 1 - Grupo: [Nombre_del_Grupo]”.
- Debajo del botón de continuar debe decir: “Construída por:” y deben figurar los integrantes del grupo uno debajo del otro.
- Pueden agregar cualquier otro elemento de diseño que consideren pertinente.

Luego de presionar continuar, se debe abrir otra ventana que solicite el ingreso de un mensaje y debajo ubique botones para:

- Cifrar mensaje César
- Cifrar mensaje Atbash
- Descifrar mensaje César
- Descifrar mensaje Atbash

También se debe ubicar adecuadamente, una caja de ingreso, para dar la posibilidad del ingreso de la clave, ya que en caso de utilizar el método César, deberá ser utilizada.

Según lo elegido por el usuario, se deberá hacer uso de las funciones desarrolladas en los objetivos 1 y 2, y finalmente mostrar al usuario lo solicitado en la misma ventana.

Agregar la posibilidad de continuar ingresando mensajes para cifrar y descifrar.

La interfaz gráfica descripta, es una propuesta básica a la que ustedes pueden agregarle mejoras funcionales.

Algunas ayudas para programar menos!

- Un mensaje cifrado con César con clave N se puede descifrar aplicando el mismo método pero con clave -N (es decir, puede utilizar la misma función para cifrar y para descifrar)
- Un mensaje cifrado con Atbash se puede descifrar aplicando nuevamente Atbash sobre el mensaje codificado.

Condiciones de Entrega

Las siguientes condiciones deben ser respetadas en su totalidad para que la entrega sea considerada válida:

1. Cada función que forma parte del código debe tener debajo de su firma, una descripción corta de cuál es su objetivo y quien es el autor o responsable de dicha función.

2. Se deben incluir pruebas unitarias en al menos 4 funciones.

3. Al cumplir cada objetivo, se debe subir el código correspondiente al campus, en la actividad habilitada a tal fin. El nombre a dar al archivo debe ser Obj#_*NombreGrupo*, reemplazando el "*#*" por el número del objetivo, y el "*NombreGupo*", por el nombre dado al grupo.

4. Finalizado el código correspondiente a la Parte 1 y antes del vencimiento, debe ser subido al campus, en la actividad habilitada a tal fin. El nombre a dar al archivo será TP1_*NombreGrupo*.py. Deberán reemplazar *NombreGrupo*, por el nombre dado a su grupo. Si la entrega está compuesta por más de un archivo .py, generar un .zip con todos los archivos .py, y nombrarlo de igual modo, pero con extensión zip.

5. Deben llevar adelante el desarrollo del código fuente mediante el uso de git mediante la plataforma de github, el mismo será revisado durante la corrección del trabajo práctico.

6. Se deberá incluir un archivo README.md en donde se incluya el nombre del grupo, sus integrantes y un link al repositorio de github (debe ser público).

7. Sólo por la entrega total de la Parte 1 (no por cada uno de los objetivos), deberán grabar 2 videos y subirlos a un canal de Youtube, ó a Google Drive.

    a. El **primer video**, **cada integrante del equipo**, deberá contar mostrando el código, qué parte estuvo bajo su responsabilidad y los puntos de solución dados, que considere más relevantes. El video total no debe superar los 10 minutos. Comenzar cada uno de los relatos, diciendo el nombre y apellido.
    Las exposiciones se deben entender y ver claramente; y deben intentar que sean homogéneas.

    b. Deberán grabar un **segundo video**, en el que se muestre al menos una jugada completa, y que contemple distintos casos que muestran que la aplicación responde según lo esperado. Deberán ir relatando los eventos de la jugada. En este caso el video puede estar realizado por 1 único integrante y no debe superar los 10 minutos.