Colección de actividades Aprender Conectados

Nivel Secundario

Programación

Mensaje encriptado

Actividad N° 8

APRENDER CONECTADOS

Ministerio de Educación, Cultura, Ciencia y Tecnología
Presidencia de la Nación

Autoridades

Presidente de la Nación
Mauricio Macri

Jefe de Gabinete de Ministros
Marcos Peña

Ministro de Educación, Cultura, Ciencia y Tecnología
Alejandro Finocchiaro

Secretario de Gobierno de Cultura
Pablo Avelluto

Secretario de Gobierno de Ciencia, Tecnología e Innovación Productiva
Lino Barañao

Titular de la Unidad de Coordinación General del Ministerio de Educación, Cultura, Ciencia y Tecnología
Manuel Vidal

Secretaria de Innovación y Calidad Educativa
Mercedes Miguel

Subsecretario de Coordinación Administrativa
Javier Mezzamico

Directora Nacional de Innovación Educativa
María Florencia Ripani

ISBN en trámite

Este contenido fue producido por el Ministerio de Educación, Cultura, Ciencia y Tecnología de la Nación en el marco del Plan Aprender Conectados

Introducción

El Plan Aprender Conectados es la primera iniciativa en la historia de la política educativa nacional que se propone implementar un programa integral de alfabetización digital, con una clara definición sobre los contenidos indispensables para toda la Argentina.

En el marco de esta política pública, el Consejo Federal de Educación aprobó, en 2018, los Núcleos de Aprendizajes Prioritarios (NAP) de Educación Digital, Programación y Robótica (EDPR) para toda la educación obligatoria, es decir, desde la sala de 4 años hasta el fin de la secundaria. Abarcan un campo de saberes interconectados y articulados, orientados a promover el desarrollo de competencias y capacidades necesarias para que los estudiantes puedan integrarse plenamente en la cultura digital, tanto en la socialización, en la continuidad de los estudios y el ejercicio de la ciudadanía, como en el mundo del trabajo.

La incorporación de Aprender Conectados en la Educación Secundaria permite poner a disposición estudiantes y docentes, tecnología y contenidos digitales que generan nuevas oportunidades para reconocer y construir la realidad: abre una ventana al mundo, facilita la comunicación y la iniciación a la producción digital.

La sociedad está cambiando a un ritmo más acelerado que nuestro sistema educativo y la brecha entre las propuestas pedagógicas que presentan las escuelas y la vida de los estudiantes se amplía cada vez más. Garantizar el derecho a aprender en el siglo XXI implica que todos los estudiantes puedan desarrollar las capacidades necesarias para actuar, desenvolverse y participar como ciudadanos en esta sociedad cada vez más compleja, con plena autonomía y libertad.

En este marco, Aprender Conectados presenta actividades, proyectos y una amplia variedad de recursos educativos para orientar la alfabetización digital en la educación obligatoria en todo el país. La actividad que se presenta a continuación y el resto de los recursos del Plan son un punto de partida sobre el cual cada docente podrá construir propuestas y desafíos que inviten a los estudiantes a disfrutar y construir la aventura del aprender.

María Florencia Ripani
Directora Nacional de Innovación Educativa

Objetivos generales

Núcleos de Aprendizajes Prioritarios

Educación Digital, Programación y Robótica – Educación secundaria	Ofrecer situaciones de aprendizaje que promuevan en los alumnas y alumnos: • La comprensión general del funcionamiento de los componentes de *hardware* y *software*, y la forma en que se comunican entre ellos y con otros sistemas, entendiendo los principios básicos de la digitalización de la información y su aplicación en la vida cotidiana. • El desarrollo de proyectos creativos que involucren la selección y la utilización de múltiples aplicaciones, en una variedad de dispositivos, para alcanzar desafíos propuestos, que incluyan la recopilación y el análisis de información. • La creación, la reutilización, la reelaboración y la edición de contenidos digitales en diferentes formatos, entendiendo las características y los modos de representación de lo digital. • La resolución de problemas a partir de su descomposición en partes pequeñas, aplicando diferentes estrategias, utilizando entornos de programación tanto textuales como icónicos, con distintos propósitos, incluyendo el control, la automatización y la simulación de sistemas físicos.

Objetivos de aprendizaje

Esta actividad permitirá introducir al lenguaje de programación Python y está orientada a desarrollar conocimientos iniciales vinculados con los siguientes objetivos de aprendizaje:

- Profundizar el conocimiento sobre estructuras de datos y su utilidad.
- Conocer funciones y transformaciones de la lista como estructura de datos.
- Reflexionar sobre la privacidad y la seguridad informática a través de conceptos básicos de encriptación.
- Aprender el uso de bucles para resolver problemas más complejos

Materiales y recursos

- Computadora.
- Python 2.x instalado.
- Lápiz negro.

Desafío

En esta actividad, creamos un sistema de encriptación que nos permite comunicarnos de forma secreta utilizando una clave previamente compartida.

< Inicio >

Disparador

Los sistemas de comunicación modernos, como la telefonía celular o la mensajería digital cuentan con poderosos métodos para mantener la privacidad de los mensajes transmitidos. ¿Sabías que esta práctica existe hace más de dos mil años?

En esta actividad vamos a crear un programa que recibe una clave numérica y, a partir de ella, encripta un mensaje ingresado a través del teclado.

A grandes rasgos, el problema se descompone en:

- Definir el dominio del alfabeto del sistema.
- Recibir la clave secreta.
- Recibir mensaje a ser cifrado
- Encriptar el mensaje.
- Imprimir el mensaje encriptado.

En este documento las líneas de código son presentadas con el siguiente formato, para su fácil identificación y copiado:

```
# Soy un comentario en el código
print 'Soy una línea de código'
Soy una línea de código
```

Las líneas de color azul son la respuesta al código introducido en las líneas anteriores y se presentan como un ejemplo del resultado a obtener. No deben ser copiadas ni ejecutadas en IDLE.

También se incluyen comentarios en gris, precedidos por el símbolo numeral. Estos comentarios son notas que dan claridad al código, pero que Python ignora y no son ejecutados.

El ejercicio se propone para ser escrito línea por línea, en IDLE. Es importante que todos los estudiantes estén frente a una computadora con IDLE, de Python, abierto al momento de comenzar la actividad y que, a medida que avanzan, ejecuten el programa para comprobar que su código esté bien y corregir errores, si los hubiera.

Al final de este documento se presentan todas las líneas del programa juntas, que pueden ser guardadas en un archivo desde IDLE para ser ejecutado como un programa.

< Desarrollo >

La ejecución de esta actividad se puede dividir en los siguientes pasos:

1. Definir el dominio del alfabeto del sistema.
2. Recibir la clave secreta.
3. Recibir el mensaje a ser cifrado.
4. Encriptar el mensaje.
5. Imprimir el mensaje encriptado.

El docente recuerda a los alumnos acerca de la importancia de utilizar los comentarios e invita colocar los códigos para que Python reconozca todos los acentos y signos en los sistemas operativos Linux y Windows. También se sugiere ingresar los datos del autor y el nombre del programa.

```
# -*- coding: cp1252 -*-
# -*- coding: 850 -*-
# -*- coding: utf-8 -*-
# Creador: Pedro Perez
# Cifrador César
```

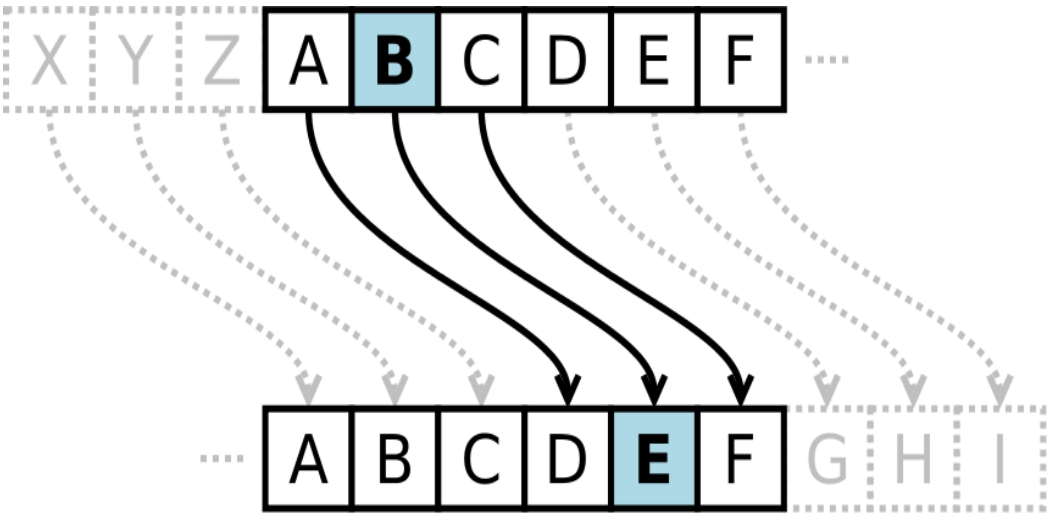
Recordá utilizar comentarios de forma frecuente, que expliquen tu código. ¡El programador que lo lea te lo agradecerá!

Se sugiere al docente comenzar la clase con una discusión sobre la privacidad en internet y la importancia de cuidar nuestros datos personales y nuestras comunicaciones. A continuación, se puede debatir sobre la ciencia de la criptología y su uso en la historia como desencadenante de la actividad.

Uno de los sistemas de encriptación más antiguos y famosos es el cifrado César, que recibe su nombre de Julio César, quien se supone lo utilizaba como método para enviar, de forma segura, sus mensajes importantes, evitando curiosos por el camino.

El método consiste en desplazar las letras del mensaje un número de posiciones previamente acordado.

Imaginá el abecedario como una cinta circular, que al llegar a la letra Z se vuelve a unir con la letra A. Si acordamos el número 3 como casilleros a desplazar para encriptar el mensaje, al escribir la letra B, se quiere decir E.



De Cepheus - Trabajo propio, Dominio público, https://commons.wikimedia.org/w/index.php?curid=1235339

Utilizando este mecanismo, cada estudiante debe escribir un mensaje y compartirlo junto a su clave con algún compañero.
Una vez comprendido el mecanismo, procedemos a crear un programa que lo encripte por nosotros.

1) Definir el dominio del alfabeto del sistema.

Para poder enviar un mensaje encriptado utilizando el sistema de cifrado César, lo primero que hay que definir son los caracteres reconocidos por el sistema.
Dado que debemos guardar estos caracteres de forma ordenada, la mejor estructura de datos para resolver este problema es la lista. Creá una lista con las letras del abecedario.

```
letras=['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']
```

¿Está faltando algo en esta lista?

El problema con el ejemplo anterior es que define un dominio de caracteres muy limitado. Solo permite escribir mensajes en minúscula y sin utilizar espacios.
Para resolver esto, cambiá el nombre de la lista, y creá otra complementaria.

```
#letras=['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']
minusculas=['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']
mayusculas=['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']
```

¿Cómo podrías extender incluso más esta lista?

Ahora nos encontramos con el alfabeto dividido en dos partes. Por suerte es extremadamente fácil operar con listas en Python. Para unir dos listas, una detrás de la otra, se puede,simplemente, sumarlas.

```
alfabeto = minusculas + mayusculas
print alfabeto
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'\xf1', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A',
'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', '\xd1',
'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
```

La ñ y la Ñ se ven codificadas porque los sistemas informáticos utilizan tablas de códigos para guardar ciertos caracteres.

Pero aún está faltando algo muy importante: los signos de puntuación. Sin ellos, los mensajes pueden ser confusos o difíciles de leer.
Agregá estos caracteres, utilizando la función append() de la lista. Esta permite sumar un objeto al final.

```
alfabeto.append(' ')
alfabeto.append(',')
alfabeto.append('.')
print alfabeto
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'\xf1', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A',
'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', '\xd1',
'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', ' ', ',',
'.']
```

Solo se suma el espacio, el punto y la coma al programa ejemplo. *¿Qué otros caracteres agregarías para sumar complejidad y claridad a tus mensajes?*

2) Recibir la clave secreta.

Para poder codificar nuestro mensaje, el sistema debe conocer la clave secreta. Utilizamos la función *raw_input*() para recibir los datos a través del teclado y guardarlos en una variable de nombre “clave”.

```
clave = raw_input("-+-+- Ingresa tu clave numérica -+-+- \n")
```

La función *raw_input*() entrega una cadena de caracteres como resultado. Dado que realizaremos operaciones matemáticas con la clave numérica, convertimos la variable clave a un número entero utilizando la función int().

```
clave = int(clave)
```

3) Recibir el mensaje a ser cifrado.

A continuación, recibimos el mensaje a ser encriptado a través del teclado y lo guardamos en una variable de nombre “mensaje”. Aprovechamos la oportunidad para crear una variable de nombre “mensaje_encriptado”, que utilizaremos en el siguiente paso.

```
print
mensaje = raw_input("-+-+- Ingresa tu mensaje sin encriptar -+-+- \n")
mensaje_encriptado = ""
```

¿Notaste el print?. Imprimí una línea en blanco antes del siguiente mensaje, y aportá claridad a la interacción entre el sistema y el usuario. El mismo efecto se consigue utilizando recursos como los símbolos en el texto para resaltar los mensajes del sistema. Creá tus propios recursos para mejorar la experiencia del usuario.

4) Encriptar el mensaje.

Esta es la parte central de nuestro código. Es el algoritmo que realiza la transformación del mensaje. Para resolver este algoritmo vamos a utilizar el poder del bucle “*for*” junto a la flexibilidad de las listas y las cadenas de caracteres en Python. Lo veremos línea a línea para explicarlo en profundidad.

El bucle “*for*” acepta una cadena de caracteres como parámetro, y recorre la cadena carácter a carácter.

```
for caracter in mensaje:
```

Esto nos permitirá trabajar de forma individual con cada carácter del mensaje.

Utilizaremos la función “index()” de la lista “alfabeto” para obtener la posición del carácter del mensaje a encriptar dentro de nuestro alfabeto. “index()” compara lo que recibe por parámetro con cada elemento de nuestro alfabeto, y si el elemento es igual al parámetro, nos devuelve su posición dentro del alfabeto.
Guardamos la posición de la letra en iteración dentro una variable de nombre “posición”.

```
for caracter in mensaje:
  posicion = alfabeto.index(caracter)
```

¿Notaste la indentación? Recordáque esta línea y las siguientes están dentro del bucle “for”, por lo que se repetirán por cada carácter del mensaje.

Calculamos la nueva posición del carácter sumando la clave a la posición y guardamos el resultado en una variable de nombre “nueva posición”.

```
for caracter in mensaje:
  posicion = alfabeto.index(caracter)
  nueva_posicion = (posicion + clave)
```

Pero debemos resolver un problema. La lista de nuestro alfabeto es circular, por lo que al pasar la Z debemos volver a la A. Se puede resolver este problema aplicando la operación módulo. En computación, la operación módulo obtiene el remanente de una división entre dos números. Por ejemplo:

```
print 3 % 1 # Resultado cero, ya que 3 se divide por 1 sin resto.
0
print 3 % 2 # Resultado 1, dado que 3 dividido 2 nos resta 1.
1
```

Para resolver el problema de la lista circular, podemos calcular el módulo de la cantidad de caracteres de nuestro alfabeto a la nueva posición.

Dado que nuestro alfabeto puede evolucionar en futuras versiones del programa, es importante que calculemos la cantidad de caracteres de forma dinámica. Para esto, utilizamos la función len() que nos devuelve la cantidad de elementos del parámetro recibido.

```
for caracter in mensaje:
  posicion = alfabeto.index(caracter)
  nueva_posicion = (posicion + clave)
  nueva posicion = nueva_posicion % len(alfabeto)
```

Utilizamos la nueva posición como índice en nuestra lista “alfabeto” y guardamos el carácter codificado en una variable de nombre “caracter_codificado”.

```
for caracter in mensaje:
  posicion = alfabeto.index(caracter)
  nueva_posicion = (posicion + clave)
  nueva posicion = nueva_posicion % len(alfabeto)
  caracter_encriptado = alfabeto[nueva_posicion]
```

En Python, las cadenas de caracteres cuentan con su propio *set* de funciones para facilitar el trabajo con las mismas. Podemos sumar nuevos caracteres al final de la cadena como una simple suma. Dado que está dentro del bucle, la línea de código a continuación suma uno por uno y en orden cada carácter encriptado a una nueva variable de nombre “mensaje_encriptado”.

```
for caracter in mensaje:
  posicion = alfabeto.index(caracter)
  nueva_posicion = (posicion + clave)
  nueva posicion = nueva_posicion % len(alfabeto)
  caracter_encriptado = alfabeto[nueva_posicion]
  mensaje_encriptado = mensaje_encriptado + caracter_encriptado
```

Podés realizar la misma suma, escribiendo menos código de la siguiente manera: mensaje_encriptado += caracter_encriptado

Ya casi completamos el algoritmo, solo falta resolver un último problema. Si el usuario ingresa caracteres que no forman parte de nuestro alfabeto definido, el programa encripta de forma errónea el mensaje. Por suerte es muy fácil resolver esto con la utilización de una estructura de selección.

Anidamos el cuerpo del bucle dentro una estructura “if”, y solo ejecutamos su contenido si el caracter del mensaje a encriptar existe dentro del alfabeto, ignorándolo en caso contrario.

```
for caracter in mensaje:
  if caracter in alfabeto:
    posicion = alfabeto.index(caracter)
    nueva_posicion = (posicion + clave)
    nueva_posicion = nueva_posicion % len(alfabeto)
    caracter_encriptado = alfabeto[nueva_posicion]
    mensaje_encriptado = mensaje_encriptado + caracter_encriptado
```

¿Notaste la doble indentación del código después del if? Python es estricto en su sintaxis y fallará la ejecución del programa si no está correctamente indentado.

¡Felicitaciones! Completaste el algoritmo que resuelve el problema.

5) Imprimir el mensaje encriptado.

Ahora solo nos resta entregar al usuario el mensaje encriptado. Lo haremos utilizando la funcion “*print*()”.

```
print
print "-+-+- Tu mensaje encriptado a continuación -+-+-"
print mensaje_encriptado
```

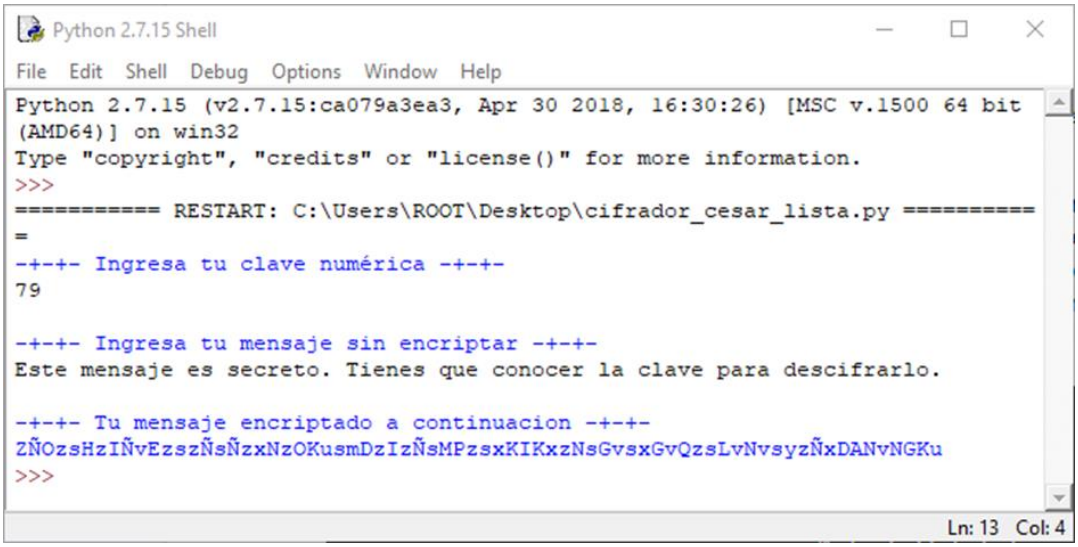
Código completo

A continuación presentamos el código completo para ser copiado en un nuevo archivo de IDLE. Podés guardarlo con el nombre cifrador_cesar.py

```
# -*- coding: cp1252 -*-
# -*- coding: 850 -*-
# -*- coding: utf-8 -*-
# Creador: Pedro Perez
# Cifrador César
```

```
#letras=['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']
minusculas =
['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']
mayusculas =
['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']
alfabeto = minusculas + mayusculas
#print alfabeto
alfabeto.append(' ')
alfabeto.append(',')
alfabeto.append('.')
#print alfabeto

clave = raw_input("-+-+- Ingresa tu clave numérica -+-+- \n")
clave = int(clave)

print
mensaje = raw_input("-+-+- Ingresa tu mensaje sin encriptar -+-
+- \n")
mensaje_encriptado = ""

for caracter in mensaje:
  if caracter in alfabeto:
    posicion = alfabeto.index(caracter)
    nueva_posicion = (posicion + clave)
    nueva_posicion = nueva_posicion % len(alfabeto)
    caracter_encriptado = alfabeto[nueva_posicion]
    mensaje_encriptado = mensaje_encriptado +
caracter_encriptado

print
print "-+-+- Tu mensaje encriptado a continuación -+-+-"
print mensaje_encriptado
```

Después de guardar el archivo, podés ejecutarlo desde IDLE, presionando la tecla “F5”.



Ventana ejemplo de la ejecución del programa en IDLE

< Cierre >

El docente pide a los estudiantes que se agrupen de a dos, para probar sus proyectos. Si ven que algo del programa no funciona, trabajan juntos para encontrar el error y solucionarlo.

Luego, se sugiere hacer una puesta en común para intercambiar conclusiones sobre lo que aprendieron, lo que más les gustó hacer y sobre los desafíos que se les presentaron.

Evaluación:

El docente puede evaluar el proyecto tanto a través de la observación, durante el desarrollo de las actividades, como en relación al programa final, para lo cual podrá acercarse a cada alumno, o bien, si se quisiese ver en detalle el código, se podrán copiar los archivos con los programas desarrollados.

Se tendrán en cuenta los objetivos específicos de la actividad, como otros aspectos vinculados a la creatividad, la cooperación entre pares y el aprendizaje a partir de la exploración y el error. Para evaluar esto, se le puede pedir a cada alumno que explique qué desafíos encontró y cómo los solucionó.

A continuación, se presentan preguntas orientadoras:

- ¿Logró resolver el desafío propuesto?
- ¿Pudo crear la estructura de datos sin dificultades?
- ¿Trabajó en grupo cuando se encontró con problemas?
- ¿Escribió el programas propuesto de forma lógica y ordenada, solucionando los errores de sintaxis, de haberse presentado?
- ¿Modificó de alguna forma su código para mejorar o modificar el del ejemplo de la actividad?
- ¿Comprendió la lógica del bucle y la estructura de selección?

El proceso de evaluación de la evolución del aprendizaje podrá continuar con la actividad propuesta a continuación.

Para seguir aprendiendo

Para continuar sugerimos al docente pedir a los estudiantes que creen un nuevo programa que desencripta mensajes encriptados, utilizando las mismas herramientas aprendidas en el transcurso de esta actividad.

Anexo

Se incluye como referencia un código que recibe un mensaje encriptados y lo desencripta, utilizando las mismas herramientas aplicadas en la actividad. Podés guardar el código en un archivo de nombre descifrador_cesar.py

```
# -*- coding: cp1252 -*-
# -*- coding: 850 -*-
# -*- coding: utf-8 -*-
# Creador: Pedro Perez
# Descifrador César

minusculas =
['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']
mayusculas =
['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']
alfabeto = minusculas + mayusculas

alfabeto.append(' ')
alfabeto.append(',')
alfabeto.append('.')


clave = raw_input("-+-+- Ingresa tu clave numérica -+-+- \n")
clave = int(clave)

print
mensaje_encriptado = raw_input("-+-+- Ingresa tu mensaje encriptado -+-
+- \n")
mensaje = ""

for caracter_encriptado in mensaje_encriptado:
  if caracter_encriptado in alfabeto:
    posicion = alfabeto.index(caracter_encriptado)
    nueva_posicion = (posicion - clave)# Nótese que se resta la posición
    nueva_posicion = nueva_posicion % len(alfabeto)
    caracter = alfabeto[nueva_posicion]
    mensaje = mensaje + caracter

print
print "-+-+- Tu mensaje desencriptado a continuación -+-+-"
print mensaje
```