Presentación

En esta práctica nos familiarizaremos con el lenguaje de programación Python mediante la implementación de un criptosistema histórico, concretamente una variante de la cifra bífida. Este criptosistema fué propuesto por el criptógrafo francés Félix-Marie Delastelle (1840-1902).

Objetivos

Los objetivos de esta práctica son:

1. Familiarizarse con el entorno de trabajo en Python y el *framework unittest*.
2. Implementar un criptosistema histórico.

Descripción de la Pràctica a realizar

La cifra bífida fué propuesta por el criptógrafo francés Félix-Marie Delastelle (1840-1902) y se trata de un criptosistema de transposición que incluye un fraccionamentio del texto. En esta página web http://practicalcryptography.com/ciphers/bifid-cipher/ podéis encontar la especificación del criptosistema con algún ejemplo. En la práctica implementaremos una extensión del modelo que se explica en esta página web. En lugar de utilizar matrices de 5x5 para representar el alfabeto, usaremos matrices de 6x6. De esta manera, nuestras matrices incluirán las 26 letras del abecedario inglés más los diez dígitos numéricos, del 0 al 9.
Aunque en la descripción del algoritmos que se hace en el enlace proporcionado no usa la letra J, nosotros sí la usaremos.

Para implementar la cifra bífida, dividiremos el trabajo en tres funciones:

1. La generación de la clave,
2. el cifrado de un texto en claro,
3. y el descifrado de un texto cifrado.

Cada uno de los ejercicios corresponde a la programación de una de estas funciones. A continuación se muestran los detalles.

1. Implementación de la cifra bífida (10 punts)

1. Función que implementa la generación de la clave bífida. (**2 punts**) La función recibirá como argumento una palabra y un número y retornará la matriz que se genera con esta palabra (ver el Anexo 1 de este enunciado) y el número como periodo.

   - La variable `keyword` contendrá la palabra para construir la matriz.
   - La variable `period` contendrña un número indicando el periodo de los grupos.
   - La función retornará la matriz y el periodo.

   Ejemplo:

   - `keyword`: `HELLO5`
   - `period`: `4`
   - `salida`: `[[['H', 'E', 'L', 'O', '5', 'A'], ['B', 'C', 'D', 'F', 'G', 'I'], ['J', 'K', 'M', 'N, 'P', 'Q'], ['R', 'S', 'T', 'U', 'V', 'W'], ['X', 'Y', 'Z', '0', '1', '2'], ['3', '4', '6', '7', '8', '9']],4]`

2. Función que implementa el cifrado de la cifra bífida. (**4 punts**)

   La función tomará como variables de entrada dos parámetros: `message` y `key`; y retornará el mensaje cifrado.

   - La variable `message` contendrá una cadena de caracteres con el texto en claro a cifrar. La cadena de caracteres podrá contener letras, en mayúscula y números.
   - La variable `key` contendrá la clave de cifrado, tal y como la retorna la función de generación de claves.
   - La función retornará una cadena de caracteres con el texto cifrado correspondiente al texto en claro.

   Ejemplo:

   - `key`: `[[['H', 'E', 'L', 'O', '5', 'A'], ['B', 'C', 'D', 'F', 'G', 'I'], ['J', 'K', 'M', 'N, 'P', 'Q'], ['R', 'S', 'T', 'U', 'V', 'W'], ['X', 'Y', 'Z', '0', '1', '2'], ['3', '4', '6', '7', '8', '9']],4]`
   - `message`: `CRYPTOGRAPHY`
   - `salida`: `FZBGRFNXL58E`

3. Función que implementa el proceso de descifrado de la cifra bífida. (**4 punts**)

   La funció tomará como variables de entrada dos parámetros: `ciphertext` y `key`; y retornará el mensaje en claro.

   - La variable `ciphertext` contendrá una cadena de caracteres con el texto cifrado. La cadena de caracteres podrá contener letras en mayúscula y números.
   - La variable `key` contendrá la clave de cifrado, tal y como la retorna la función de generación de claves.
   - La función retornará una cadena de caracteres con el texto en claro correspondiente al texto cifrado.

   Ejemplo:

   - `key`: `[['H', 'E', 'L', 'O', '5', 'A'], ['B', 'C', 'D', 'F', 'G', 'I'], ['J', 'K', 'M', 'N', 'P', 'Q'], ['R', 'S', 'T', 'U', 'V', 'W'], ['X', 'Y', 'Z', 'O', '1', '2'], ['3', '4', '6', '7', '8', '9']],4]`
   - `ciphertext`: `FZBGRFNXL58E`
   - `salida`: `CRYPTOGRAPHY`

Criterios de valoración

La puntuación de cada ejercicio se encuentra detallada en el enunciado.

Por otro lado, es necesario tener en cuenta que el código que se envíe debe contener los comentarios necesarios para facilitar su seguimiento. En caso de no incluir comentarios, la corrección de la práctica se realizará únicamente de forma automática y no se proporcionará una corrección detallada. No incluir comentarios puede ser motivo de reducción de la nota.

Formato y fecha de entrega

La fecha máxima de envío es el **16/03/2020** (a las 24 horas).

Junto con el enunciado de la práctica encontrareis el esqueleto de la misma (fichero con extensión .py). Este archivo contiene las cabeceras de las funciones que hay que implementar para resolver la práctica. Este mismo archivo es el que se debe entregar una vez se codifiquen todas las funciones. No se tiene que enviar el fichero comprimido. No se aceptarán ficheros en otros foramtos que no sean .py.

Adicionalmente, también os proporcionaremos un fichero con tests unitarios para cada una de las funciones que hay que implementar. Podeis utilizar estos tests para comprobar que vuestra implementación gestiona correctamente los casos principales, así como para obtener más ejemplos concretos de lo que se espera que retornen las funciones (más allá de los que ya se proporcionan en este enunciado). Nótese, sin embargo, que los tests no son exhaustivos (no se prueban todas las

entradas posibles de las funciones). Recordad que no se puede modificar ninguna parte del archivo de tests de la práctica.

La entrega de la práctica constará de un único fichero Python (extensión .py) donde se haya incluido la implementación.

Annex 1

Existen difererntes criptosistemas históricos que utilizan como clave (o como parte de ella) una matriz donde se incluyen las letras del abecedario. Estas matrices se denominan cuadrados de Polibi, ya que fué este historiador griego (que vivió alrededor de los años 205 a 120 a.c.) quien propuso su uso.

Un sistema muy simple para poder compartir cuadrados de Polibi entre emisor y receptor sin tener que recordar toda la matriz consisten en generarla a partir de una palabra clave que es fácil de memorizar. Para generar el cuadrado de Polibi, se parte de una palabra clave y se va rellenando la matriz, primero poniendo las letras de la clave sin repetirlas, despúes completando la matriz con las letras del abacedario que faltan, en orden alfabético. En caso de que se incluyan números, se añaden al final.

Por ejemplo, si la palabra clave es “HELLO5” la matriz resultante sería:

$$\begin{bmatrix} H & E & L & O & 5 & A \\ B & C & D & F & G & I \\ J & K & M & N & P & Q \\ R & S & T & U & V & W \\ X & Y & Z & 0 & 1 & 2 \\ 3 & 4 & 6 & 7 & 8 & 9 \end{bmatrix}$$