Práctica 0 de SAR

Boletín de ejercicios para practicar

Ejercicio 1: Un pangrama es una frase que contiene todas las letras del alfabeto Inglés, al menos una vez, por ejemplo: *"The quick brown fox jumps over the lazy dog"*. Escribe una función para comprobar si una cadena de texto recibida como argumento es un pangrama o no. La función debe devolver un booleano.

Debes ignorar el hecho de usar mayúsculas o minúsculas.

Aunque se podría hacer de otras formas, se aconseja utilizar la clase `set` de Python para crear el conjunto de letras de la cadena así como el conjunto de letras del alfabeto inglés y así determinar de manera eficiente si uno de los conjuntos está incluído en el otro.

Puedes utilizar la cadena `string.ascii_lowercase` disponible tras importar el módulo `string`:

```
>>> import string
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
```

Ejercicio 2: Un pangrama *"perfecto"* es un pangrama en el que cada letra del alfabeto inglés aparece una sola vez.

Haz una función que reciba una cadena y determine si se trata de un pangrama perfecto o no. Para ello debes suponer que los únicos caracteres que puede contener la cadena, además de las propias letras, son espacios en blanco.

Debes ignorar el hecho de usar mayúsculas o minúsculas (por ejemplo, pasaró a minúscula con el método `.lower()`).

Sugerencia: comprobar que cada una de las letras de `string.ascii_lowercase` aparece una sola vez en la cadena.

Ejercicio 3: Una cadena es *anagrama* de otra si contiene las mismas letras (y para cada letra el mismo número de veces) que la otra cadena.

Por ejemplo, la cadenas *"Life on mars"* es un anagrama de *"alien forms"*.

Hay que eliminar los espacios y símbolos de puntuación antes de ver que ambas cadenas tienen exactamente las mismas letras (sin importar el orden).

Observa que si conviertes las cadenas a conjunto (`set`) **no es correcto** porque no tiene en cuenta si una letra aparece más de una vez.

Ejercicio 4: Haz una función que simule de una manera "limitada" el comando 'grep' de Unix. La entrada de la función es:

- La ruta de un fichero de texto (que supondremos con encoding UTF-8)
- Una cadena de texto

La salida es una lista que contenga (en el mismo orden) todas las líneas del fichero que contengan alguna ocurrencia de la cadena de texto recibida como argumento.

Por ejemplo, si el fichero prueba.txt contiene:

```
hola, esta es la primera línea
la segunda línea del fichero prueba.txt
otra línea más
por fin termina este fichero de 4 líneas hola hola
```

La llamada:

```
funcion_grep('prueba.txt','la')
```

devolvería la lista:

```
["hola, esta es la primera línea",
 "la segunda línea del fichero prueba.txt",
 "por fin termina este fichero de 4 líneas hola hola"]
```

Observa que la última línea del fichero no contiene la palabra `"la"` pero sí la ocurrencia de la subcadena `"la"` (forma parte de la palabra `"hola"`).

Ejercicio 5: Escribe una función `char_freq()` que toma una cadena y crea una lista de frecuencias de los carácteres que figuran en el mismo. Representa la frecuencia de carácteres con un diccionario Python. Prueba con algo como `char_freq("abbabcbdbabdbdbababababcbcbab")`.

> **Nota:** Puedes utilizar la clase `Counter` del módulo `collections` (ver el siguiente enlace).

Ejercicio 6: Escribe una función que Determine si una cadena formada por corchetes es equilibrada, es decir, si consiste enteramente de pares de apertura / cierre de corchetes (en ese orden), y todos los corchetes abiertos se cierran correctamente. Ejemplos:

```
[]          OK
[][]        OK
[[][]]      OK
][          NOT OK
][][        NOT OK
[]][[]      NOT OK
```

Se sugiere recorrer la cadena con un contador que se incremente al encontrar `[` y se decremente al encontrar `]`, el resto es fácil... si se ve un `]` con el contador a 0 no es ok, y al final el contador ha de quedar a 0.

Solamente hay que procesar 2 símbolos, el resto se ignoran, con lo que estos ejemplos también debe de ser capaz de procesarlos:

```
[hola]        OK
]hola[        NOT OK
```