Introducción a Python. Ejercicio U09\_E02

MasterD

# Conteo de palabras

Vamos a tratar de resolver un problema sencillo que suele utilizarse como caso típico para explicar algunas tecnologías Big Data. Aquí de momento vamos a ver cómo solucionarlo con las herramientas propias de Python.

Se trata de calcular la frecuencia con la que aparece cada palabra en una serie de textos, normalmente contenidos en fichero. Por simplificar, tomaremos un único texto que ya tendremos cargado en una variable.

texto = """Y el Cuervo nunca emprendió el vuelo.  
Aún sigue posado, aún sigue posado  
en el pálido busto de Palas.  
en el dintel de la puerta de mi cuarto.  
Y sus ojos tienen la apariencia  
de los de un demonio que está soñando.  
Y la luz de la lámpara que sobre él se derrama  
tiende en el suelo su sombra. Y mi alma,  
del fondo de esa sombra que flota sobre el suelo,  
no podrá liberarse. ¡Nunca más!"""

## Primera versión

Podemos resumir la secuencia de pasos de la siguiente forma

* Limpiar signos de puntuación
* Trocear el texto en palabras
* *"Normalizar"* las palabras
* Procesar y contar cada palabra

Vayamos uno por uno

### Limpiar signos de puntuación

Antes de dividir el texto en palabras, tenemos que deshacernos de caracteres que pueden interferir después para los conteos, como signos de puntuación (puntos, comas, interrogaciones, etc.) y otros símbolos no alfanuméricos.

Tu primera tarea es crear una función que reciba una cadena de texto como argumento y devuelva la cadena ya limpia como resultado.

**Pistas**

Recuerda qué método para cadenas de texto permitía reemplazar un carácter con otro. Puedes utilizar como carácter de reemplazo la cadena vacía.

Como tienes que reemplazar varios caracteres distintos, tal vez quieras usar algún bucle.

def limpiar\_puntuacion(txt):  
 """  
 Limpia de signos de puntuación la cadena txt  
 y devuelve el contenido ya limpio  
 """  
 for c in ".,;:¡!¿?":  
 txt = txt.replace(c, "")  
 return(txt)

Esta no es la única solución válida, se te han podido ocurrir alternativas similares. En esta solución nos aprovechamos de que una cadena de texto es una secuencia, y por tanto la podemos iterar, así que usamos una cadena para indicar todos los caracteres a reemplazar.

### Trocear en palabras

Este paso no es complejo, simplemente tenemos descomponer el texto en la secuencia de palabras que lo forman. Vamos a encapsular este paso también en una función. ¿Puedes completarla?

def trocear\_palabras(txt):  
 """  
 Devolver la secuencia de palabras   
 por separado del texto  
 """  
 return(txt.split())

### Normalización

Otra cosa que hay que hacer antes de empezar a contar es *normalizar* las palabras. Es decir, eliminar las diferencias que puedan hacer que se tomen como palabras distintas variaciones que representan lo mismo.

Más claro, en este caso: ignorar las diferencias entre mayúsculas y minúsculas. Para Python, como en la mayoría de los lenguajes de programación, la cadena "HOLA" es diferente de la cadena "hola". Sin embargo, para nuestras necesidades, ambas representan la misma palabra.

Otros casos más complejos de normalización podrían ser las abreviaturas o siglas. Por ejemplo, "Sr." equivaldría a "señor". Nosotros de momento nos vamos a conformar con normalizar el uso de mayúsculas y minúsculas.

Así que ahora tienes que crear una función que reciba una cadena de texto y pase todos los caracteres al mismo formato (p.ej. todos en minúscula).

def normalizar(cadena):  
 """  
 Devolver la cadena normalizada  
 """  
 return(cadena.lower())

### Conteo de palabras

Una vez que tenemos las palabras del texto por separado, ya podemos empezar a contar cuántas veces ocurre cada una.

Te toca completar la siguiente función que recibe la lista de palabras y devuelve algún tipo de estructura con el número de veces que aparece cada palabra en el texto. De momento es una versión muy *naïve*, después veremos alguna otra forma de hacerlo (a lo mejor a tí ya se te ha ocurrido).

**Pistas** ¿Qué tipo de estructura de datos te permite añadir elementos asociándoles un valor? Utiliza la estructura de datos para almacenar el conteo de cada palabra.

Itera las palabras y ves actualizando su contador. Ten en cuenta si es la primera vez que aparece la palabra para inicializar su contador particular.

def contar(palabras):  
 """  
 Devuelve una estructura con el número de ocurrencias  
 de cada palabra  
 """  
 conteos = {}  
 for palabra in palabras:  
 if palabra in conteos:  
 conteos[palabra] = conteos[palabra] + 1  
 else:  
 conteos[palabra] = 1  
 return(conteos)

### Probando

Ahora ya puedes combinar las funciones y probar que todo funciona. Si ejecutas el siguiente código deberías obtener el resultado esperado.

texto\_limpio = limpiar\_puntuacion(texto)  
palabras = [ normalizar(p) for p in trocear\_palabras(texto\_limpio) ]  
conteo = contar(palabras)  
print(conteo)

{'y': 4, 'el': 6, 'cuervo': 1, 'nunca': 2, 'emprendió': 1, 'vuelo': 1, 'aún': 2, 'sigue': 2, 'posado': 2, 'en': 3, 'pálido': 1, 'busto': 1, 'de': 7, 'palas': 1, 'dintel': 1, 'la': 4, 'puerta': 1, 'mi': 2, 'cuarto': 1, 'sus': 1, 'ojos': 1, 'tienen': 1, 'apariencia': 1, 'los': 1, 'un': 1, 'demonio': 1, 'que': 3, 'está': 1, 'soñando': 1, 'luz': 1, 'lámpara': 1, 'sobre': 2, 'él': 1, 'se': 1, 'derrama': 1, 'tiende': 1, 'suelo': 2, 'su': 1, 'sombra': 2, 'alma': 1, 'del': 1, 'fondo': 1, 'esa': 1, 'flota': 1, 'no': 1, 'podrá': 1, 'liberarse': 1, 'más': 1}

## Una versión más *pythonica*

La función contar() hace su trabajo, pero como ya habrás pensado, hay formas más elegantes de construir una colección de datos en Python. Sin ir más lejos, las definiciones por comprensión.

Intenta crear una expresión que asocie para cada palabra única un valor correspondiente al número de veces que aparece.

**Pistas**

Para obtener las palabras únicas puedes utilizar alguna de las colecciones de datos que vimos en la unidad.

Recuerda también que existen métodos predefinidos para obtener el número de veces que cierto elemento aparece en una secuencia.

conteo2 = { p : palabras.count(p) for p in set(palabras) }  
print(conteo2)

{'los': 1, 'nunca': 2, 'sus': 1, 'posado': 2, 'sigue': 2, 'fondo': 1, 'aún': 2, 'se': 1, 'soñando': 1, 'lámpara': 1, 'luz': 1, 'busto': 1, 'un': 1, 'sobre': 2, 'puerta': 1, 'vuelo': 1, 'dintel': 1, 'que': 3, 'más': 1, 'en': 3, 'emprendió': 1, 'y': 4, 'flota': 1, 'él': 1, 'está': 1, 'esa': 1, 'pálido': 1, 'sombra': 2, 'de': 7, 'demonio': 1, 'ojos': 1, 'tienen': 1, 'del': 1, 'derrama': 1, 'tiende': 1, 'su': 1, 'liberarse': 1, 'apariencia': 1, 'cuervo': 1, 'podrá': 1, 'la': 4, 'mi': 2, 'suelo': 2, 'el': 6, 'alma': 1, 'palas': 1, 'no': 1, 'cuarto': 1}

## Más *pythonico* todavía

Para terminar, Python nos permite solucionar este problema de forma mucho más breve y eficiente aprovechando la biblioteca estándar.

En este caso, vamos a utilizar una librería que no te habíamos presentado todavía. Es la librería collections. Este módulo incluye varias implementaciones optimizadas y muy eficientes de algunos tipos de datos compuestos muy específicos. Puedes ver más información en <https://docs.python.org/3/library/collections.html>

Una de las clases que incluye este módulo es Counter. Es una variante de un diccionario que sirve precisamente para contar automáticamente las ocurrencias de cada elemento de una lista o cualquier otra secuencia o iterable.

from collections import Counter  
conteo3 = Counter(palabras)  
print(conteo3)

Counter({'de': 7, 'el': 6, 'y': 4, 'la': 4, 'en': 3, 'que': 3, 'nunca': 2, 'aún': 2, 'sigue': 2, 'posado': 2, 'mi': 2, 'sobre': 2, 'suelo': 2, 'sombra': 2, 'cuervo': 1, 'emprendió': 1, 'vuelo': 1, 'pálido': 1, 'busto': 1, 'palas': 1, 'dintel': 1, 'puerta': 1, 'cuarto': 1, 'sus': 1, 'ojos': 1, 'tienen': 1, 'apariencia': 1, 'los': 1, 'un': 1, 'demonio': 1, 'está': 1, 'soñando': 1, 'luz': 1, 'lámpara': 1, 'él': 1, 'se': 1, 'derrama': 1, 'tiende': 1, 'su': 1, 'alma': 1, 'del': 1, 'fondo': 1, 'esa': 1, 'flota': 1, 'no': 1, 'podrá': 1, 'liberarse': 1, 'más': 1})

¡Así de simple!