Escuela de Programación - Python B1

Seguimiento Temas 1-3

Introducción

Este documento describe un ejercicio para evaluar los conocimientos de los temas 1-3 del nivel Python B1. El proyecto consiste en la implementación de una serie de funciones sobre un texto. Para su desarrollo se ponen en práctica los conocimientos adquiridos durante los temas 1 a 3 del curso.

Descripción de las actividades

El objetivo general del ejercicio es crear una serie de funciones que nos permitan realizar operaciones sobre un texto.

Para este ejercicio, no se debe usar la función **`split`** de Python. En vez de ello, deberás usar las siguientes funciones auxiliares que serán de gran ayuda al resolver el ejercicio. Asimismo, se pueden elegir crear nuevas funciones adicionales. A continuación, presentaremos una descripción de estos métodos:

- **`is_newline(character):`** Es una función que detecta el final de una oración. Deberás suponer que las frases están separadas por "\n" (nueva línea). Si el carácter es este símbolo, devolverá `True`.
- **`is_space(character):`** Es una función que detecta si un carácter es un espacio en blanco. Si el carácter es este símbolo, devolverá `True`.
- **`remove_punctuation_marks(cad):`** Una función que elimina los signos de puntuación de una palabra o un texto. Este método devuelve como resultado una cadena de caracteres sin signos de puntuación.

Las funciones descritas en el apartado anterior forman parte del módulo denominado `'text_manager.py'`, por lo tanto, es preciso importar estas en el módulo `'ejb1_x1_main.py'`, el cual es el módulo principal en el que desarrollaremos nuestra solución.

En este ejercicio utilizaremos la variable "TEXT" de tipo cadena de caracteres(definida en el módulo `text_manager.py`), la cual será empleada en cada una de las siguientes funciones como parámetro. Los métodos que se solicita desarrollar son:

`find_largest_word(text):` Un método que permite detectar la palabra más larga en un texto. Este método debe devolver como resultado una cadena de caracteres correspondiente a la palabra más larga. Al evaluar la palabra no debe contener signos de puntuación.

`is_palindrome_word(word):` Es una función **recursiva** que nos permitirá detectar si una palabra es palíndromo. Un palíndromo es una palabra que se lee igual en un sentido que en otro. Por ejemplo las siguientes palabras son palíndromos: Ata; Aviva; Azuza; Apa; Afromorfa. Para el ejercicio, el texto

se encuentra en lengua inglesa, por lo que no se requiere realizar ningún tipo de acción en relación con tildes o acentos. Al evaluar la palabra no debe contener signos de puntuación. El valor que devuelve es de tipo booleano. Si es un palíndromo devolverá `True`, y en el caso contrario `False`.

`count_palindrome_words(text):` Se trata de una función que nos permitirá enumerar las apariciones de palíndromos en el texto, por lo tanto, esta retorna un número entero. Para esto debemos hacer uso de la anterior función **`is_palindrome_word(word).`**

`find_size_largest_sentence(text, filter):` Se trata de una función que permite encontrar el tamaño de la oración más larga cuyo valor de filtro esté en esa sentencia. Si no existe una oración que coincida con el filtro deberá lanzar una excepción del tipo `ValueError`. El valor a retornar es un número entero que representa la longitud de la cadena en cuestión.

Por ejemplo: si se invoca a la función con los parámetros text = "Hola, Pepe.\n¿Cómo estás, amigo?", y el parámetro filter = "a", esta debe devolver 19, ya que en la segunda oración "¿Cómo estás, amigo?", se encuentra incluido el valor pasado como filtro y la oración tiene una longitud de la cadena de texto más larga.