PROCESAMIENTO DE ARCHIVO DE TEXTO

18/03/19

Objetivos de aprendizaje y competencias

1. Objetivos de aprendizaje:
    a. Leer y escribir un archivo
        i. open r, r+, w, a.
        ii. with open(file) as f:
        iii. for line in f:
        iv. write
    b. Formatear la salida.
    c. Leer strings usando input
    d. Usar listas simples y anidadas:
        i. Usar elementos de la lista mediante su posición
        ii. Buscar un elemento en la lista
        iii. Eliminar un elemento
        iv. Unir dos listas
        v. Usar for para recorrer una lista
        vi. Usar funciones y operaciones de listas
    e. Usar while para crear ciclo
    f. Usar break para salir de for y while
    g. Definir y usar funciones simples
    h. Usar funciones de strings: strip(), split(), len()
    i. Usar comprensión de listas.
    j. Usar diccionarios.
    k. Definir y usar funciones con argumentos keyword, valor default.

2. Competencias que se desarrollan:
    a. Analizar un problema para formular una solución
    b. Trabajar en equipo resolviendo un problema
    c. Presentar los resultados de un proyecto
    d. Idear un algoritmo para resolver un problema

Descripción

El objetivo de este proyecto es procesar un texto contenido en un archivo texto.txt, el que deberán crear utilizando las funciones de manejo de archivo presentes en el tutorial de pyhton (fopen, fclose, write, read) como se indica en el siguiente ejemplo:

```
# genere un archivo llamado "diccionario.txt" en formato de escritura.
diccionario = open('diccionario.txt','w')

# Una vez generado el archivo, se le indica a la variable "diccionario"
# creada que escriba en el archivo las siguientes lineas:
diccionario.write('gato felino minino chuchito\n mono simio gorila
macaco\n')

# Luego, se procede a cerrar el archivo en la variable.
diccionario.close()
```

El objetivo es reemplazar ciertas palabras por sus sinónimos usando un diccionario. Este diccionario está contenido en otro archivo llamado **diccionario.txt**, donde cada línea tiene el siguiente formato, ejemplo:

```
gato felino minino cuchito
mono simio gorila macaco
```

con el contenido de este archivo deben crear un diccionario de python, por ejemplo:

```
{'gato' : ['felino', 'minino','cuchito'],
 'mono' : ['simio','gorila','macaco']}
```

Para este efecto deberán escribir una función `leer_diccionario('nombre del archivo')` que construya y devuelva el diccionario como valor de retorno.

El texto a procesar se encuentra en otro archivo, indicado en la primera parte de este enunciado. Está compuesto por una secuencia de líneas y se pide hacer una función `def leer_texto('nombre del archivo')` que lea el archivo, línea por línea, y devuelva una lista donde cada elemento corresponde a una línea del texto.

Por ejemplo, para el siguiente texto:

```
Hola como estas
Bien y tu
```

La lista generada sería:

```
["Hola como estas", "Bien y tu"]
```

Luego es necesario programar las siguientes funciones:

1. `def separar_palabras(lista_de_lineas, separador=’ ‘)`

   Esta función tiene por objetivo separar las palabras contenidas en las líneas procesadas anteriormente. Es decir, cada línea procesada pasa a ser una nueva lista y, los elementos de esa línea son sus palabras.

   Por ejemplo, para la línea "Hola, como estas":

   ```
   ["Hola", "como", "estas"]
   ```

2. `def cambiar_palabras(lista_de_lineas, diccionario)`

   Cambiar las palabras de un texto usando el diccionario previamente leído, de la siguiente manera:

   - La primera vez que se ve una palabra, no se cambia.
   - Las siguiente vez que se encuentra una palabra se cambia por su primer sinónimo, luego por el segundo sinónimo, etc.
   - Cuando se acaba la lista de sinónimos se vuelve a iniciar desde el primer sinónimo.
   - Para hacer esta función es necesario mantener una lista de palabras que se han visto, para saber cuando una palabra se ve por primera vez.
   - También es necesario saber qué sinónimo se usará la próxima vez. Esto se puede hacer guardando el índice del siguiente sinónimo o haciendo una rotación en la lista para que el siguiente sinónimo sea siempre el primero de la lista.

   Por ejemplo, al leer el siguiente texto se reemplaza gato y mono usando el diccionario:

   Diccionario

   ```
   {'gato' : ['felino', 'minino','cuchito'],
    'mono' : ['simio','gorila','macaco']}
   ```

Texto leído:

```
El gato de la Su salió a pasear en el bosque
donde encontró otro gato que jugaba con un mono
el primer gato se enojó con el otro gato porque ese gato
le estaba pegando al pobre mono
en ese momento vio que venía otro mono a defender al
primer mono.
```

El resultado es:

```
El gato de la Su salió a pasear en el bosque
donde encontró otro felino que jugaba con un mono
el primer mínimo se enojó con el otro cuchito porque ese felino
le estaba pegando al pobre simio
en ese momento vio que venía otro gorila a defender al
primer macaco.
```

3. `def grabar_archivo('nombre de archivo', n=60)`
   Grabar el archivo línea por línea de forma tal que cada línea escrita no supere una cantidad `n` de caracteres (si no se especifica un valor para `n`, se usa `60`. Si sobran palabras se comienzan a grabar en la línea siguiente. Si faltan palabras, toman de la siguiente lista de palabras. Las palabras deben estar separadas por un solo espacio.

Antecedentes

Para realizar este proyecto deberán considerar la siguiente documentación del tutorial de Python en **python.org** (abrir en google chrome y traducir de inglés a español).

- 4.1 Listas, tuplas, diccionarios
  https://docs.python.org/3/tutorial/introduction.html#lists
  https://docs.python.org/3/tutorial/datastructures.html

- 4.2 Ciclo `for`
  https://docs.python.org/3/tutorial/controlflow.html#for-statements

- 4.3 La función `range()`
  https://docs.python.org/3/tutorial/controlflow.html#the-range-function

- 4.4 Leer y escribir archivos
  https://docs.python.org/3/tutorial/inputoutput.html#the-string-format-method

- 4.5 Definir de funciones

https://docs.python.org/3/tutorial/controlflow.html#defining-functions
https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions

Instrucciones

La entrega de los trabajos se debe hacer en un Cuaderno Jupyter, documentando las partes del código y los resultados como sea necesario.