



Manual de usuario – Tarea Programada #1

Instituto Tecnológico de Costa Rica

Sede Campus Central Cartago

Escuela de Ingeniería de la programación

Taller de Programación

Periodo Lectivo

Segundo semestre del año 2021

Estudiantes

Jose Pablo Agüero Mora - 2021126372

Needler Bonilla Serrano - 2021043058

Fecha de entrega

09/10/2021

Estatus de la Entrega

Excelente

Introducción:

A la hora de identificar identidades individuales de las palabras en un texto es tedioso intentar catalogar cada una de ellas de forma manual y sin llevar un orden de los datos, por este motivo buscar una forma de procesar esta información más rápido agilizaría la tarea.

El presente programa representa una solución automatizada en el análisis morfológico de los textos según la clasificación de sus palabras. Para lograr esto cuenta con diferentes procesos que tokenizan la entrada y dan la opción de visualizar en múltiples formas estos datos (como archivos HTML, XML y binarios).

La implementación de estos procesos dinamiza la clasificación de palabras dentro de un texto y permite dar una selección de distintas formas en las que se pueden almacenar estos datos ordenados para ser usados en otros proyectos.

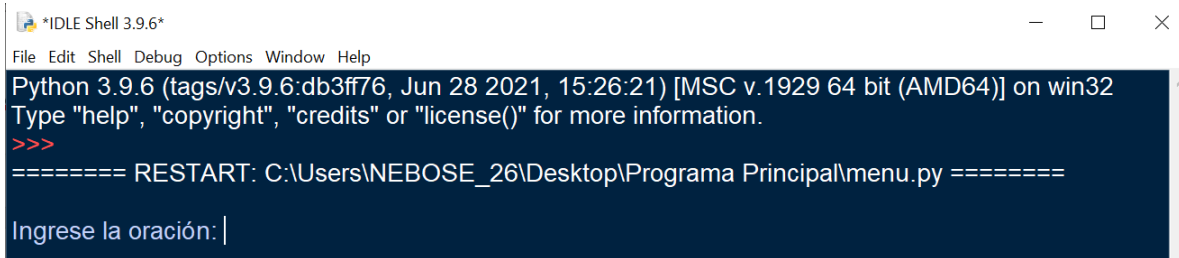
Qué funcionalidades implementadas tiene el software:

- **Entrada de datos y validación:** Al iniciar el programa cuenta con una validación que determina si ya existe un archivo de texto guardado, en este caso lee los datos de tal archivo. Si no existe un archivo previo, pide el texto al usuario y verifica que sea un contenido válido.
- **Creación de txt:** Si no existe un archivo txt previo, crea uno nuevo con los datos ingresados por el usuario al iniciar el programa.
- **Preparación de los datos:** Una vez ingresados los datos, estos pasan por un proceso para convertirse en una lista dividida por los espacios de la cadena y se procede a eliminar cualquier tipo de signos de puntuación o caracteres no alfanuméricos. Además, si por algún error del usuario este incluyó doble espacio, el proceso también eliminará los elementos que aparezcan como vacíos.
- **Menú de opciones:** Cuando ya se han registrado o leído datos y procesado cada uno de los filtros correspondientes, el programa muestra un menú principal que ofrece 5 opciones diferentes a elegir. Este menú aparecerá cada vez que el usuario termine de realizar alguna acción para permitirle seleccionar otras opciones.
- **Tokenizar:** Si se selecciona la opción de tokenizar, la lista original pasará por un proceso de clasificación de palabras y creación de nuevas sublistas. Esta clasificación depende de los atributos de 6 grandes grupos de palabras: artículos, preposiciones, pronombres, verbos, números y palabras sin clasificar. Se creará una lista general con una sublista por grupo de palabras.

- **Ordenamiento alfabético:** De forma automática, cuando se selecciona la opción de tokenizar, existe un filtro extra que se encarga de reestructurar las sublistas en orden alfabético para que cuando se muestren lo hagan de forma organizada.
- **Generar archivo HTML:** Una vez tokenizado y procesado el texto con todo lo anterior, se puede seleccionar una de las 3 opciones de guardado/salida de datos. En la opción actual genera un archivo HTML con una tabla dinámica (mediante etiquetas) para el ordenamiento de la información, además de incluir dos secciones más: el texto original y un reporte de cantidades de elementos respectivas en cada grupo. El nombre de este archivo es dinámico ya que su estructura es: Analisis-dd-mm-aaaa-hh-mm-ss.html.
- **Generar archivo XML:** Otra de las opciones es generar un archivo XML el cual clasifica cada sublista en diferentes niveles de nodos que se guardan de forma ordenada.
- **Generar archivo binario:** Con esta opción se le permite al usuario crear un archivo que guarde los datos tokenizados y ordenados en forma binaria. Además, esta función permite mostrar los datos guardados en el Shell de tal archivo. Esta información se presenta de forma segmentada, y si ya existe un archivo procederá a leer los datos y mostrarlos en el Shell igualmente.
- **Salir:** Esta última funcionalidad permite parar el despliegue del menú y termina por completo el programa.

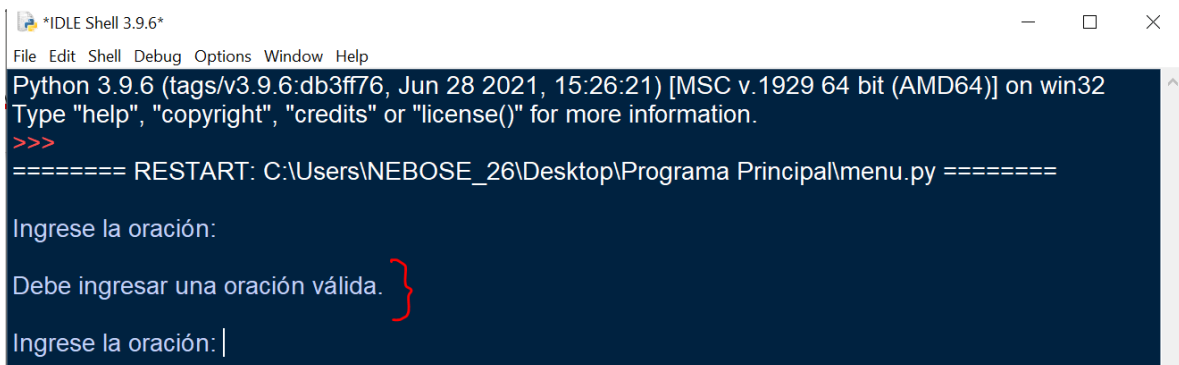
Explicación paso a paso de cómo probar cada uno de los algoritmos implementados:

Nada más ingresar al código, nos solicitará que ingresemos una oración/texto.



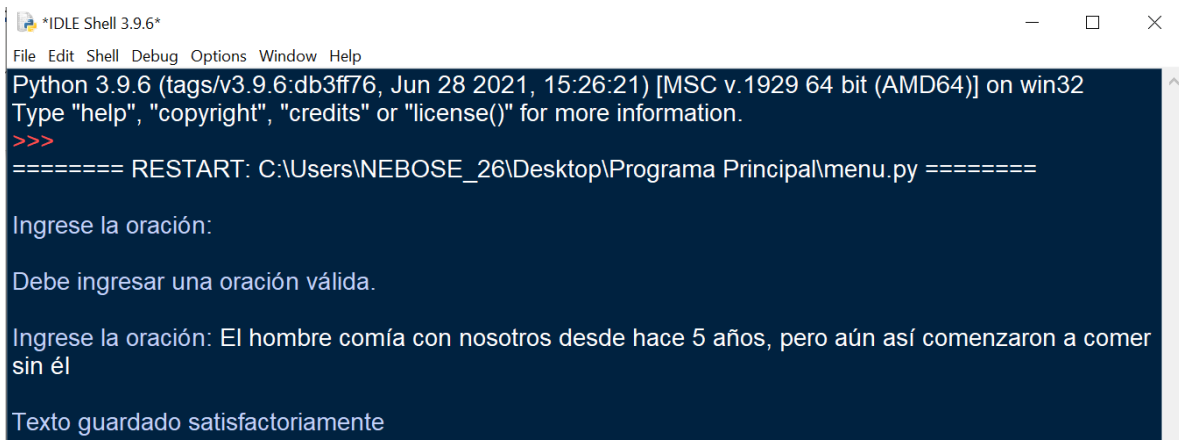
```
*IDLE Shell 3.9.6*
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\NEBOSE_26\Desktop\Programa Principal\menu.py =====
Ingrese la oración: |
```

Si al ingresar la oración nos equivocamos y pulsamos el botón enter, nos avisará que debemos de ingresar un valor válido para continuar con el programa y nos solicitará volver a ingresar una oración.



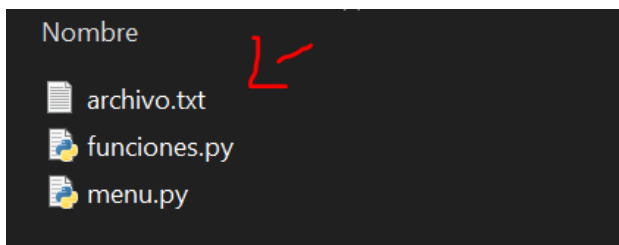
```
*IDLE Shell 3.9.6*
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\NEBOSE_26\Desktop\Programa Principal\menu.py =====
Ingrese la oración:
Debe ingresar una oración válida. }
Ingrese la oración: |
```

Si ingresamos correctamente la oración, nos notificará que el texto se ha guardado satisfactoriamente. Esto lo podemos confirmar si nos dirigimos a la carpeta donde reside el programa.



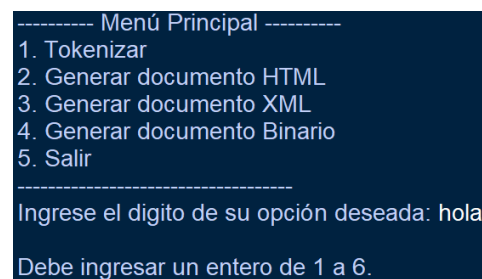
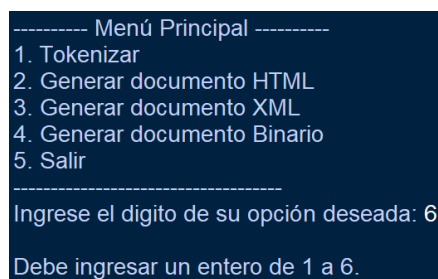
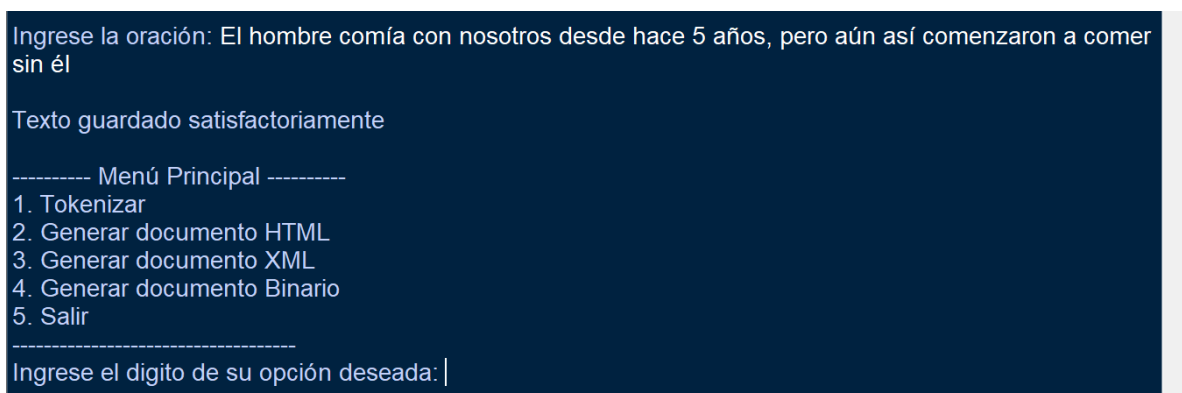
```
*IDLE Shell 3.9.6*
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\NEBOSE_26\Desktop\Programa Principal\menu.py =====
Ingrese la oración:
Debe ingresar una oración válida.
Ingrese la oración: El hombre comía con nosotros desde hace 5 años, pero aún así comenzaron a comer sin él
Texto guardado satisfactoriamente
```

Aquí podemos observar que se ha creado un documento .txt llamado archivo. En este archivo se guarda la oración que ingresamos antes.



Después de haber guardado la oración en el archivo.txt, la información pasó por una preparación de datos que elimina los signos de puntuación, espacios y caracteres no alfanuméricos. (Este proceso no es visible para el usuario)

Después de este proceso, se nos muestra un menú con 5 opciones a escoger para el cual se puede ingresar el dígito de la opción deseada. Si se ingresa algún texto o número diferente a las opciones indicadas notificará un mensaje de error y volverá a mostrar el menú.



Pero primero deberemos de “Tokenizar” en la opción 1 antes de generar cualquier archivo. De lo contrario mostrará un mensaje de error diciendo que no se ha tokenizado previamente. (Manera incorrecta y manera correcta)

```

----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 2

No se ha tokenizado el texto previamente.

```

```

----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 1

El texto se ha tokenizado satisfactoriamente

```

Ya despues de haber tokenizado, nos notifica que ha cumplido el proceso satisfactoriamente. Una vez hecho esto ya podremos acceder a las demás opciones del menú (Generar distintos tipos de archivos).

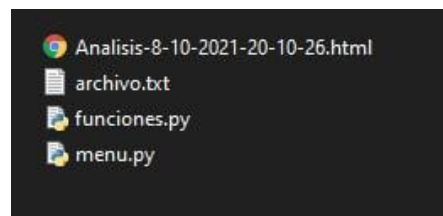
Ahora probaremos la segunda opción (Generar documento HTML) que nos notificará su creación satisfactoria y creará un documento en la carpeta actual del programa con un nombre dinámico.

```

----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 2

El archivo HTML se ha creado satisfactoriamente.

```



Este sería el documento HTML que en su interior contiene los elementos de la oración ordenadas en sublistas según su tipo.

Contenido del archivo original: el hombre comia con nosotros desde hace 5 años, pero aun asi comenzaron a comer sin el

Análisis del documento					
Artículos	Preposiciones	Pronombres	Verbos	Números	Sin Clasificar
el	a	nosotros	comer	5	asi
	con				aun
	desde				años
	sin				comenzaron
					comia
					hace
					hombre
					pero

Reporte: El texto original tiene 16 tokens de los cuales hay: 1 artículos, 4 preposiciones, 1 pronombres, 1 verbos, 1 números y 8 sin clasificar.

Si anteriormente existía un documento HTML en la carpeta actual, esta nos notificará que ya existía y nos retornará al menú otra vez.

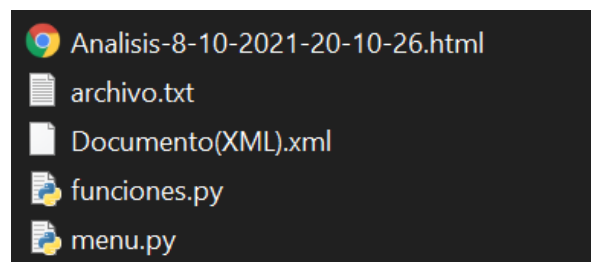
```
----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 2

Ya existe un archivo HTML creado.
```

Después de esto, nos dirigiremos a probar la tercera opción (Generar documento XML) seleccionando la opción 3 del menú. Esto nos generará un archivo XML en la carpeta actual y nos mostrará un mensaje de que tal archivo se ha creado satisfactoriamente.

```
----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 3

El archivo XML se ha creado satisfactoriamente.
```



Dentro del archivo XML se puede observar una ramificación de nodos con cada uno de los elementos de la oración en distintas sub listas según su clasificación.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Documento>
  <Parte Seccion="Articulos">
    <Contenido>['el']</Contenido>
  </Parte>
  <Parte Seccion="Preposiciones">
    <Contenido>['a', 'con', 'desde', 'sin']</Contenido>
  </Parte>
  <Parte Seccion="Pronombres">
    <Contenido>['nosotros']</Contenido>
  </Parte>
  <Parte Seccion="Verbos">
    <Contenido>['comer']</Contenido>
  </Parte>
  <Parte Seccion="Numeros">
    <Contenido>['5']</Contenido>
  </Parte>
  <Parte Seccion="SinClasificar">
    <Contenido>['asi', 'aun', 'anos', 'comenzaron', 'comia', 'hace', 'hombre', 'pero']</Contenido>
  </Parte>
</Documento>
```


Se recomienda abrir el archivo XML con su buscador predeterminado.

Después de retornar al menú principal probaremos la cuarta opción (Generar Archivo Binario). Esto nos creará un archivo binario llamado BD dentro de la carpeta actual y nos mostrará en el Shell de Python el contenido de cada sublista organizado por su clasificación, para después retornarnos al menú nuevamente.

```
Ingrese el digito de su opción deseada: 4
El archivo binario se ha creado satisfactoriamente.

Artículos:
- el

Preposiciones:
- a
- con
- desde
- sin

Pronombres:
- nosotros

Verbos:
- comer

Números:
- 5

Sin Clasificar:
- así
- aun
- años
- comenzaron
- comía
- hace . . . . .
```

Como quinta y ultima opción del menú (Salir) nos permite parar de desplegar el menú y salir del programa.

```
----- Menú Principal -----
1. Tokenizar
2. Generar documento HTML
3. Generar documento XML
4. Generar documento Binario
5. Salir
-----
Ingrese el digito de su opción deseada: 5

Saliendo del programa...
```

Pendientes de implementar y su justificación:

Al momento de desarrollar el código, se plantea la búsqueda de elementos dentro de una lista de otros elementos mediante el uso de una agrupación de letras. Esta agrupación de letras nos permite buscar dentro de dicha lista y determinar cualquier cantidad de verbos que contenga en su interior y agregarlos a una nueva lista.

Esto sería el funcionamiento principal de la búsqueda de verbos mediante su terminación, Pero esta función debería de ofrecer más que eso, ya que no se pudo implementar una característica secundaria pero no menos importante, esta sería subdividir los segmentos de los verbos en sus distintos tipos como lo son los gerundios, participios e infinitivos.

Esto es debido al escaso tiempo que se disponía para modificar el código nuevamente e implementarlo.

Bibliografía

Anónimo.(17 de Julio del 2020). *¿Como se puede ordenar palabras en orden alfabético sin usar sort() EN PYTHON?*. Stack Overflow en español.

<https://es.stackoverflow.com/questions/374300/como-se-puede-ordenar-palabras-en-orden-alfab%C3%A9tico-sin-usar-sort-en-python>

Anónimo.(s.f). *Cómo obtener la fecha y/o hora actual*. Recursos Python.

<https://micro.recursospython.com/recursos/como-obtener-la-fecha-y-hora-actual.html>

César Cancino .(14 de Agosto del 2020). *Videotutorial 12 Taller Práctico Programación con Python. Creación y lectura de archivo XML*. [Video]. <https://youtu.be/HZvI4yxBT0>

iTecnoGalaxy.(01 de Abril del 2020). *Crear pagina HTML con Python*. [Video].

<https://youtu.be/4W7HeJvAaD0>

J2LOGO (20 de Abril del 2020). *Listar directorio en Python. Listar ficheros de un directorio*. [Video]. <https://j2logo.com/python/listar-directorio-en-python/>

JAORSOFTWARE. (23 de Octubre del 2020). *01915 Curso Python 120 Creando un Archivo XML*. [Video].

<https://www.youtube.com/watch?v=GKdeM8Dch8Q&list=LL&index=14>

P.L Diego (s.f). *Tablas en HTML*. Htmlquick.

<https://www.htmlquick.com/es/tutorials/tables.html>