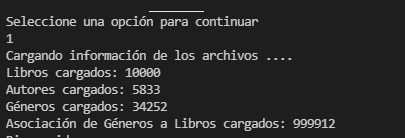
OBSERVACIONES DEL LA PRACTICA

Estudiante 1: Santiago Arciniegas Castro Cod:2020149994

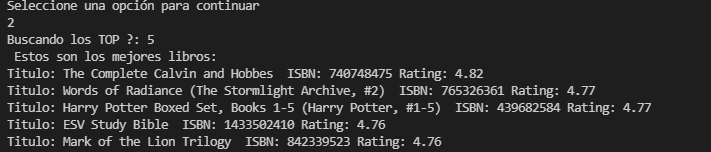
Estudiante 2:Maria Camila Luna Velasco Cod 201920993

1. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **view.py** con el usuario?

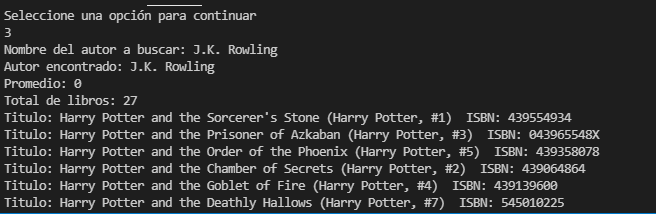
Primero despliega el menú y pide seleccionar una entrada. Al digitar la opción 1 , esta mostro en pantalla lo siguiente:



Al digitar la opción 2, esto pidió un numero sobre el cual deseáramos evaluar los libros(input), como se muestra a continuación:



Finalmente, al digitar la opción 3, se le pide al usuario que ingrese el nombre del autor(input) y el output fue el siguiente:

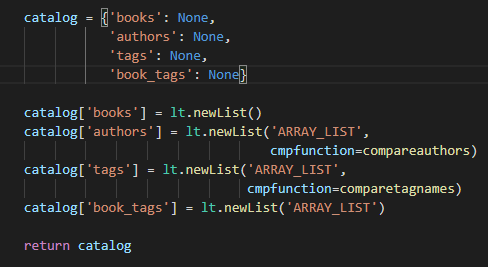


Y en la opción 4 le pide al usuario digitar sobre que etiqueta desea obtener el resultado, tal como se muestra a continuación:



1. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?

Lo primero que se hace en este caso es crear un catálogo vacío en forma de diccionario, en donde sus valores son arreglos vacíos por el momento:



Luego se utilizan las siguientes funciones para **almacenar información** en el catálogo:

- def addBook: Añade el libro a la lista de libros, luego se obtienen los autores del libro. Y después para cada autor se añade en la lista de libros del catálogo, y de igual manera se añade el libro en la lista de dicho autor.

- def addBookAuthor: se adiciona un autor a la lista de autores. Esta lista guarda referencias de los libros escritos por cada autor agregado.

- def addTag: Adiciona los tags a la lista de tags

-def addBookTag: Adiciona los book tags a la lista de books tags

Después se encuentran las funciones para **creación de datos** :

-def newAuthor: Esta se encarga de crear un diccionario donde esta el nombre y un arreglo vacío donde entraran los libros escritos por el autor.

-def newTag: Almacena los tags utilizados para los libros en una estructura

-def booksTag: Esta función crea una relación entre los tags y los libros marcados con dichos tags.

También están las funciones de consulta las cuales agrupan por autor, por tags o por rating. Adicional a esto, también hay funciones que sirven para comparar por autor, por rating y por tag. Estas funciones sirven para que las demás funciones carguen y sirvan de forma eficiente.

En conclusión, la forma en que se almacenan los datos es primero creando el catalogo vacio, luego creando arreglos por autor, tags, libros y bookstags. Una vez creados se añaden a los arreglos generales del catálogo.

1. ¿Cuáles son las funciones que comunican el **view.py** y el **model.py**?

-def initCatalog

-def loadBooks

-def loadTags

-def loadBooksTags

-def sortBooks

-def getBooksByAuthor

-def getBestBooks

-def countBooksByTag

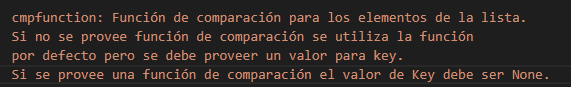
1. ¿Cómo se crea una lista?

lt.newList('SINGLE\_LINKED') #Lista encadenada

lt.newList('ARRAY\_LIST') #Arreglo

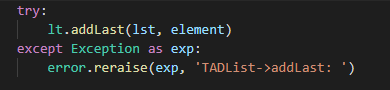
1. ¿Qué hace el parámetro **cmpfunction=None** en la función **newList()**?

En este caso, esto representa que no se hará ninguna comparación entre listas, como las que se hacen en las funciones de comparación.



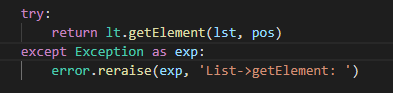
1. ¿Qué hace la función **addLast()**?

Añade un elemento al final de la estructura de datos. Y actualiza el apuntador a la última posición y se incrementa el tamaño de la lista en 1.



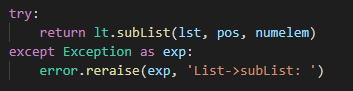
1. ¿Qué hace la función **getElement()**?

Toma por argumento una estructura de datos y un índice. Recorre la lista hasta encontrar el índice, el cual debe ser mayor a cero y menor al tamaño de la lista (esta no puede ser vacía)



1. ¿Qué hace la función **subList()**?

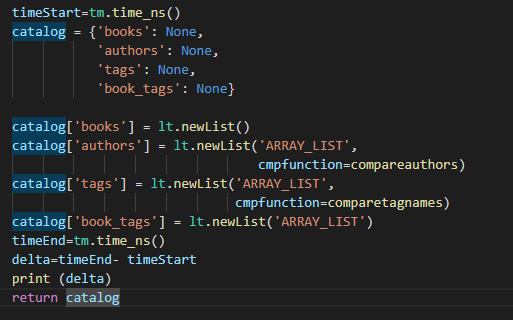
Esta retorna una sublista de la lista original. Esto apartir de tres argumentos: lista original, numero de elementos a copiar y posición a partir de la cual se inicia la “copia”.



1. ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**?

**“ARRAY\_LIST”**

En el caso del arreglo al medir el delta nos arrojo 0. La forma en que logramos justificar este comportamiento es ya que crear un arreglo vacío es constante O(1) (que es lo que realmente hace la función newCatalog). Sin embargo, creemos que a la hora de implementar la función de LoadData() esta tardara menos en términos de tiempo ya que en el código se hace uso de la operación getElement, la cual es O(1) en arreglos.



**“SINGLE\_LINKED”**

En este caso al medir la tiempos en la función requerida (NewCatalog) el prgrama de la misma manera arrojo “0” cm delta. Est se debe a que crear listas vacías es O(1). Sin embargo, al hacer un ánalisis del código, llegamos a la conclusión que al hacer este cambio la función de Loadata tardara más que con un arreglo, ya que al hacer uso de la peración getElement esta es O(n) en linstas encadenadas, lo que hará que los dats tarden más en cargar.

