**OBSERVACIONES**

Laura Daniela Arias - 202020621

David Leonardo Almanza - 202011293

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

El principal mecanismo de interacción es el menú (printMenu()), el cual le muestra al usuario todas las opciones que tiene el programa. Seguidamente, se le pide al usuario que ingrese el número de la opción que desea usar. Al ingresar el número, el programa sabe qué operaciones debería llevar a cabo y las ejecuta. Adicionalmente, si la opción elegida requiere de más inputs, view.py le pide al usuario los datos necesarios y le da el espacio para digitarlos en consola. Finalmente, el programa retorna e imprime en pantalla los resultados de la opción seleccionada, tal que el usuario pueda visualizarlo

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

Los datos de GoodReads son almacenados por la función newCatalog en una variable llamada catalog que corresponde a un diccionario con 4 keys: ‘books’, ‘authors’, ‘tags’ y ‘book\_tags’. Estas 4 llaves corresponden cada una a una lista en la cual otras funciones del programa cargan los libros, autores, tags, y book-tags respectivamente.

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

Todas las funciones que comunican el view con el model están en controller. El view llama del controller funciones –tales como initCatalog()– que a su vez llaman funciones del model como, por ejemplo, newCatalog().

Las funciones que van del view al controller son:

Controller.initCatalog()

Controller.loadData()

Controller.getBestBooks()

Controller.getBooksByAuthor()

Controller.countBooksByTag

Las funciones que van del controller al model son:

model.newCatalog()

model.addBook(catalog, book)

model.addTag(catalog, tag)

model.addBookTag(catalog, booktag)

model.sortBooks(catalog)

model.getBooksByAuthor(catalog, authorname)

model.getBestBooks(catalog, number)

model.countBooksByTag(catalog, tag)

1. **¿Cómo se crea una lista?**

Se crea una nueva lista a partir de la función newList() en la cual se proporcionan los parametros datastructure (tipo de estructura de datos a utilizar), cmpfuncion (La función de comparación para los elementos), key (Identificador necesario si no se especificó nada en cmpfunction), filename (nombre de un archivo del cual generar la lista) y delimeter (el valor de separación en caso de usar filename) para su correcta construcción. Adicionalmente, la función lleva a cabo un try en caso de haber un fallo en la construcción de la lista. De no haber error alguno, retorna la lista armada con los parámetros proporcionados.

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

Establece la función por defecto para la comparación de los elementos de la lista. Esto también permite que la función sea más estándar para la posterior creación de diferentes estilos de lista según se requiera.

1. **¿Qué hace la funció addLast()?**

Llama a otra función del programa para agregar un elemento al final de la lista. Incluye un try en caso de error.

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

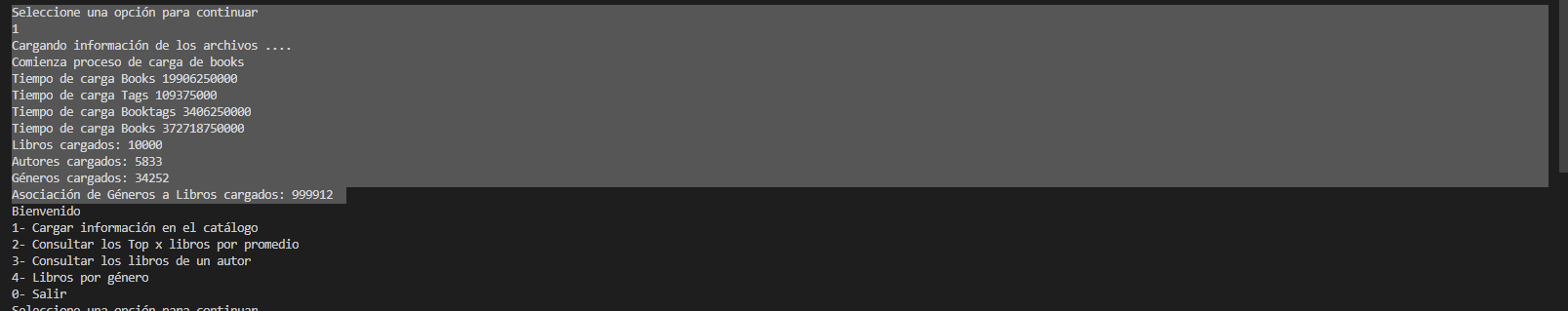
Llama a otra función del programa para rescatar el elemento en una posición dada de una lista. Incluye un try en caso de error.

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

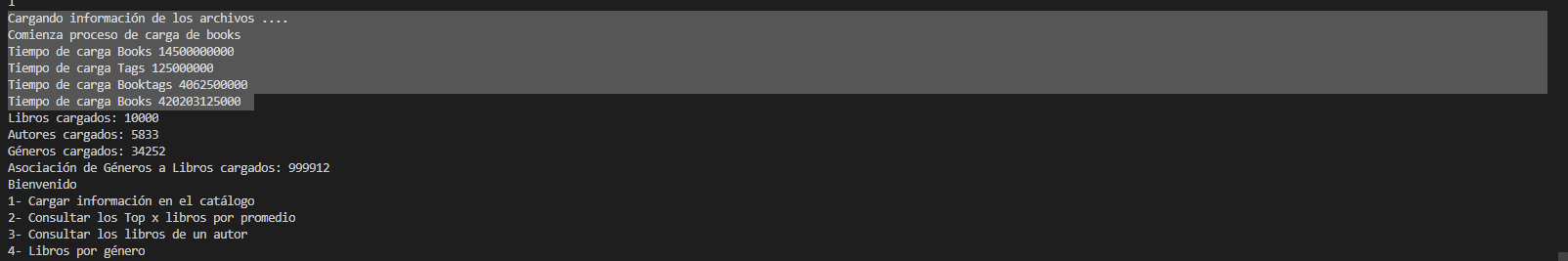
Llama a otra función del programa para generar una sublista, la cual contiene solo un fragmento de la lista dada, teniendo en cuenta una posición de inicio y una cantidad de elementos que incluir en la sublista. Incluye un try en caso de error.

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

A continuacion se enlistaran los tiempos de carga de cada lista. El segundo tiempo de carga de Books que sale corresponde en verdad a la aplicación de la funcion sort.

Tiempos de carga con parámetro ARRAY\_LIST:

Tiempos de carga con parámetro SINGLE\_LINKED:



El tiempo de carga de books no se va a comparar porque a este no se le cambió el tipo de implementación.

El tiempo de carga de Tags es del mismo orden de magnitud (9) para las dos implementaciones, sin embargo la implementación array llevó a cabo el proceso más rápido que la implementación SingleLinked.

El tiempo de carga de Book-Tags es del mismo orden de magnitud (10) para las dos implementaciones, sin embargo la implementación array llevó a cabo el proceso más rápido que la implementación SingleLinked.

El tiempo de carga de la funcion SORT es del mismo orden de magnitud (12) para las dos implementaciones, sin embargo la implementación array llevó a cabo el proceso más rápido que la implementación SingleLinked.

En conclusión, en general el programa tardó más tiempo en ejecutarse con listas SINGLE\_LINKED que con listas ARRAY.