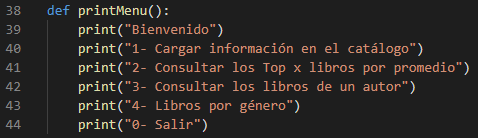
OBSERVACIONES DEL LA PRACTICA

Nicolas Merchan Cuestas - 202112109

Andrés Felipe Castro Leguizamon - 201920805

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

El programa inicia con un proceso iterativo que muestra las opciones del menú por medio de la referenciación de la función **PrintMenu()**.

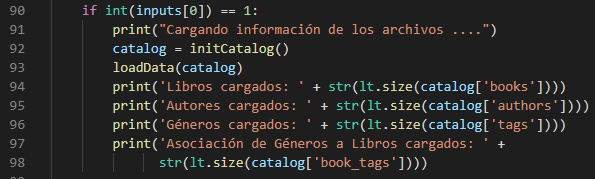
 

Posteriormente, realiza una verificación de la opción elegida por el usuario.

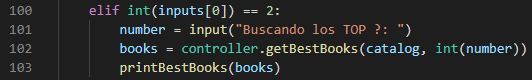
   

De ser la opción 1 elegida se crea una variable con la salida de la función **initCatalog()** y se ingresa esta por parámetro en la función **loadData()**, la cual tiene como función cargar los datos haciendo referencia a la función **initCatalog()** del controlador. Así mismo, se procede a importar el archivo **list** de la carpeta **DiSClib.ADT**. Finalmente, se utiliza la función **size** proveniente de **list** para calcular el número de libros, autores, géneros y asociaciones de géneros a libros cargados.

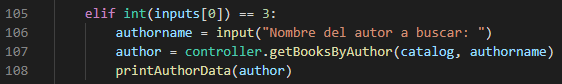




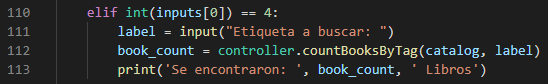
De ser la opción 2 elegida, se le solicita al usuario el número de los mejores libros que quiere que el programa muestre y ser guarda en la variable **number**. Posteriormente, se procede a llamar la función **getBestBooks()** del controlador utilizando las variables **catalog** creada en la opción 1 y **number** como parámetros. Después, se imprime el resultado de **getBestBooks()** en consola.



De ser la opción 3 elegida, se le solicita al usuario el nombre del autor del cual desea saber los libros que ha publicado y ser guarda en la variable **authorname**. Posteriormente, se procede a llamar la función **getBooksByAuthor()** del controlador utilizando la variables **catalog** creada en la opción 1 y **authorname** como parámetros. Después, se imprime el resultado de **getBooksByAuthor()** en consola.



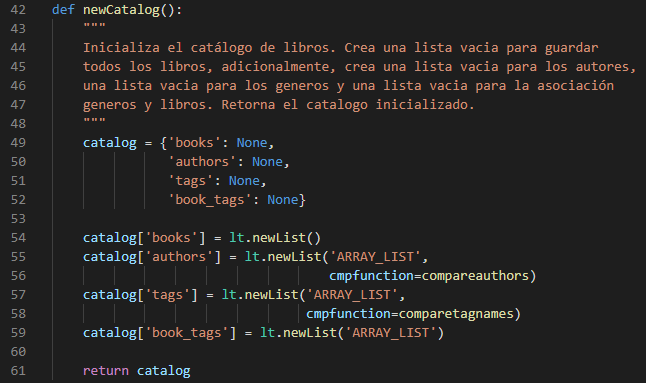
De ser la opción 4 elegida, se le solicita al usuario la etiqueta sobre la cual quiere saber cuántos libros existen y se guarda el valor en la variable **label**. Posteriormente, se procede a llamar la función **getBooksByAuthor()** del controlador utilizando las variables **catalog** creada en la opción 1 y **label** como parámetros. Después, se imprime el resultado de **getBooksByAuthor()** en consola.



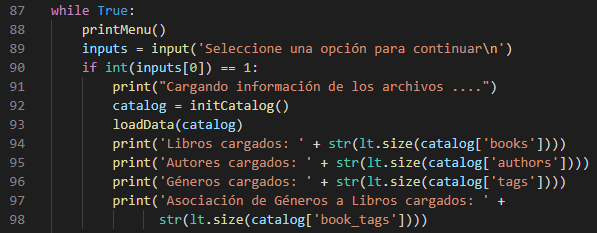
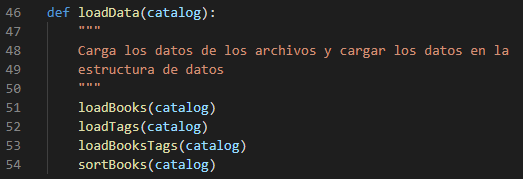
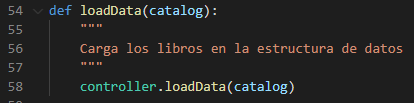
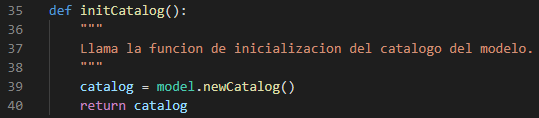
Finalmente, si el usuario elige la opción 9, el programa se detiene a causa de la función **exit(0)** de la librería **sys**.

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

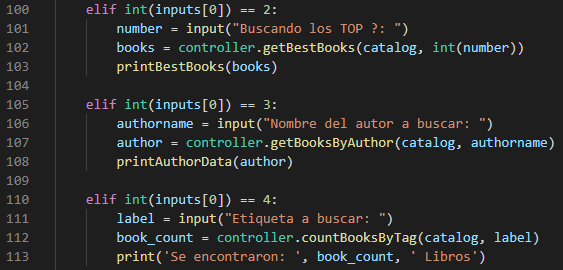
Los datos de **GoodReads** son almacenados en el diccionario **catalog** en la forma de TAD listas. Los datos de **books.csv** se almacenan en una lista encadenada (**SINGLE\_LINKED**) (tipo de almacenamiento por defecto de la función **it.newlist()**) por defecto y los datos de **authors.csv**, **tags.csv**  y **book\_tags.csv**  se almacenan en listas ordenadas (**ARRAY\_LIST**). Dichas listas se encuentran inicialmente vacías, pero posteriormente se les agregaran los elementos respectivos de cada archivo.



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

La función **initCatalog()** y **loadData()** comunica **view.py** con **model.py** cuando el programa procede a cargar los datos. La función **loadData()** en **controller.py** hacer referencia a funciones dentro de **controller.py** que, a su vez, hacen referencia a las funciones **addBook()**, **addTag()**, **addBookTag()** y **sortBooks()** presentes en **model.py**. 

Las funciones **getBestBooks()**, **getBooksAuthor()** y **countBooksByTag()** referenciadas en **view.py** y presentes en **controller.py** hacen referencia, a su vez, a funciones respectivas con nombres iguales en **model.py**.

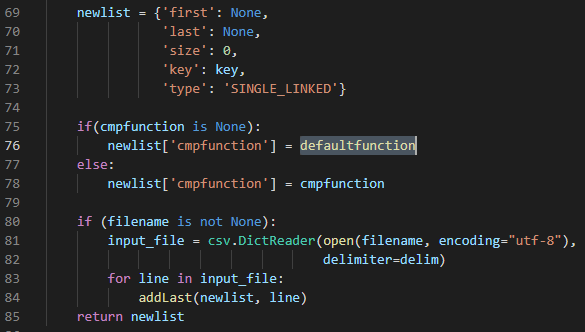


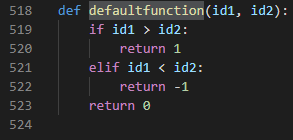
1. ¿Cómo se crea una lista?

Las listas se crean haciendo uso de la función **newlist()**. Dicha función crea una lista ordenada por defecto, pero puede crearse una lista encadenada simple en caso de que el usuario lo especifique por parámetro.

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

El parámetro **cmpfunction=None** compara dos elementos (**id1** y **id2**) de tal modo que, si **id1** es mayor que **id2** el parámetro se retorna 1, si **id1** es menor que **id2** el parámetro retorna -1, y si **id1=id2** entonces el parámetro retorna 0.





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

Agrega un elemento en la última posición de la lista. Se adiciona un elemento en la última posición de la lista y se actualiza el apuntador a la última posición. Se incrementa el tamaño de la lista en 1.

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

Retorna el elemento en la posición de la lista dada por parámetro. Se recorre la lista hasta tal elemento, el cual debe ser mayor que cero y menor o igual al tamaño de la lista. Se retorna el elemento en dicha posición sin eliminarlo. La lista no puede ser vacía.

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

Retorna una sublista de la lista dada por parámetro. Se retorna una lista que contiene los elementos a partir de la posición especificada por el usuario, con una longitud de elementos también especificada por el usuario. En ambos casos se realizan dichas especificaciones por parámetro. Se crea una copia de dichos elementos y se retorna una lista nueva.

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

El tiempo de ejecución del programa incremento de manera considerable. Al utilizar los archivos **.csv** reducidos (**-small**) se pudo observar que, en el caso original, el programa tardo en cargar los datos 0.8639171123504639 s. Sin embargo, cuando se realizo la misma acción después de haber cambiado la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”** el programa cargó los datos en 1.0415799617767334 s.