OBSERVACIONES DEL LA PRACTICA

Estudiante Juan Sebastián Ortega Cod 202021703

Estudiante Yesid Camilo Almanza Cod 201921773

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

-En primera instancia existe un **Menu Principal** formado mediante un ciclo de tipo *while* que se ejectura al vez que se abre el programa, este cuenta tanto con inputs como con outputs y es el principal medio de comunicación entre el ususario y el programa.

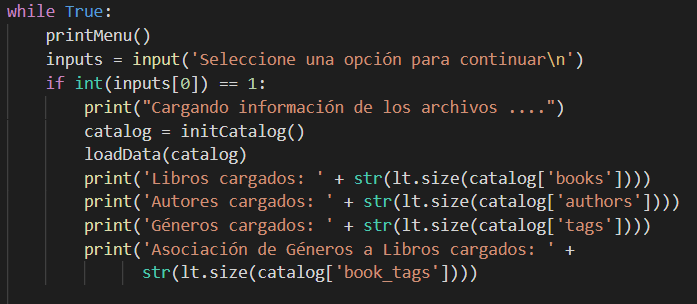


Figura 1: Menu Principal de view.py

|  |  |  |  |
| --- | --- | --- | --- |
| **INPUT** | **Descripción INPUT** | **OUTPUT** | **Descripción OUTPUT** |
| **N/A:** Se inicia la aplicación y se ejecuta el ciclo del Menu Principal |  | **Función:**  printMenu() | Menu que le muestra al usuario las opciones disponibles en la aplicación y relaciona cada una de estas con un número determinado. Es mostrado mediante el uso de varios print(). |
| -Opción del menu a ingresar  **(input = 1)** | Se produce una comunicación entre el view y el controller con el objetivo de cargar el catalogo de libros. | **Menu Principal:**  Múltiples prints que informan acerca del catalago de libros cargados. | Se retorna al usuario el total de: libros cargados, autores cargados, géneros cargados y la asociación de géneros a libros cargados. |
| * Opción del menú a ingresar **(input = 2)** * Top de libros a buscar **(number)** | -Con el catálogo ya cargado, el ususario le indica al programa que desea encontrar un top de los libros cargados (input = 2) printBestBooks(books) .  -El usuario le indica al programa el número de libros que desea que aparezca en el top (number).  Se ejecuta la función printBestBooks. | **Función:**  printBestBooks(books) | Se le retorna al ususario la información de los mejores libros que entran al top de longitud *number.*  Se le informa al usuario sobre el nombre, isbn y rating de cada libro dentro del top. |
| * Opción del menú a ingresar **(input = 3)** * Nombre del autor a buscar **(authorname)** | **-**El usuario le indica al programa que desea consultar los libros de un autor determinado  (input =3).  -El usuario le indica al programa el nombre del autor determinado (authorname)  Se ejecuta la función printAuthorData(author) | **Función:** printAuthorData(author) | Se le retorna al usuario el nombre del autor encontrado, el promedio de su rating, el total de sus libros y la información de estos.  Si no se encuentra nombre del autor, también se le informa mediante un mensaje. |
| -Opción del menú a ingresar **(input = 4)**  -Etiqueta que se desee buscar **(label)** | **-** El usuario le indica al programa que desea saber el número de libros bajo una etiqueta determinada (input = 4)  -El usuario le indica al programa el nombre de la etiqueta (label) | **menuPrincipal:** Multiples prints le informan al ususario sobre el total de libros relacionados a la etiqueta ingresada. | Se le retorna al usuario el número total de libros que contienen la etiqueta ya ingresada. Si no se encuentra la etiqueta, se retorna un mensaje de error. |

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

Los datos de la base de datos *GoodReads* se almacenan mediante el uso de un diccionario cuya estructura y jerarquía está establecida por la función newCatalog que se observa a continuación:

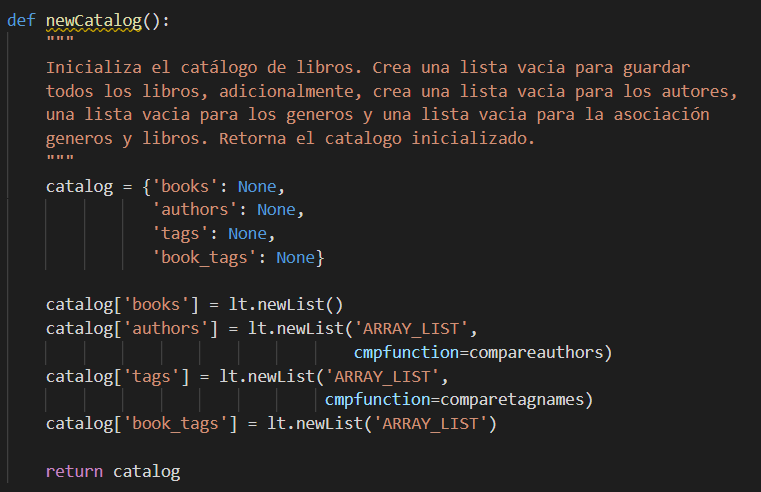


Figura 2: Estructura del diccionario de almacenamiento

De acuerdo con la función *newCatalog* los datos son almacenados en un diccionario que cuenta con 4 llaves: books, authors, tags y book\_tags. Cada una de estas contiene una lista del tipo “ARRAY\_LIST” donde se almacenará la información relacionada a cada libro dentro de la base de datos.

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

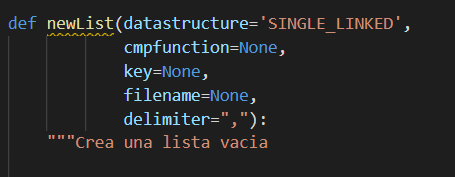
Todas las funciones que hacen parte del archivo **controller.py** sirven como vehículo para comunicar tanto INPUTS como OUTPUTS entre el **view.py** y el **model.py**. Estas funciones se subdividen en los siguientes grupos:

|  |  |  |
| --- | --- | --- |
| **Objetivo** | **Funciones Incluidas** | **Descripción** |
| Inizializar el catalogo de libros | -initCatalog() | Esta función es invocada al elegir la opción 1 dentro del Menu Principal del view.py. Esta le indica al model.py que debe prepararse para cargar los datos. |
| Cargar los datos | -loadData()  -loadBooks()  -loadTags()  -loadBooksTags() | Este set de funciónes también es invocado al elegir la opción 1 dentro del menú principal del view.py. Este le indica al model.py que debe cargar los datos en sus diferentes secciones. |
| Funciones de Consulta y Ordenamiento | -sortBooks(catalog)  -getBooksbyAuthor(catalog, authorname)  -getBestBooks(catalog, number)  -countBooksbyTag(catalog, tag) | Las funciones aquí presentadas son invocadas cuando el usuario escoge opciones específicas en el view.py y su objetivo es comunicarle al model que debe realizar dichas opciones específicas que ha solicitado el ususario. |

1. ¿Cómo se crea una lista?

Dentro del archivo ***list.py*** de la librería DISClib es posible encontrar la función newList(), esta indica los parametros y requisitos necesarios para dar con la creación de una lista. Cada uno de estos parametros modificará la forma, tipo e incluso comportamiento de la lista creada como se verá a continuación:

Indica el tipo de estructura de datos que será utilizado por el TAD lista. (Array o Single Linked List)



En dado caso de que se mantenga la función de comparación por defecto, este es un identificador para comparar dos elementos.

Función específica utilizada para comparar los elementos de la lista.

Figura 3. Función newList()

Valor que separará los elementos (Por defecto es “,”)

Nombre del archivo de tipo .CSV a partir de cuyos elementos se espera generar la lista.

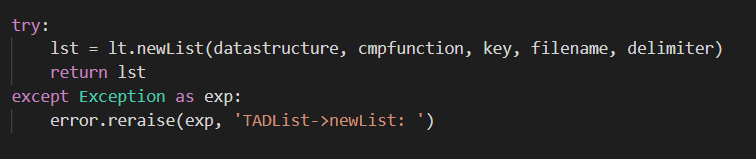


Figura 4. Implementación de la función newList() para crear el TAD lista.

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

Este parametro establece una función de comparación específica con la cual se podran comparar los elementos que pertenezcan al TAD Lista creado. En dado caso de que no se provea una función (cmpfunction = None), el programa automáticamente recurrirá a la función de comparación por defecto. Para más información es posible referirse a la documentación incluida en el archivo *list.py* referente a este paramentro:

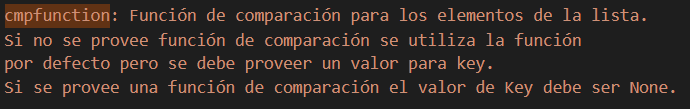


Figura 5: Documentación referente al parametro cmpfunction

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

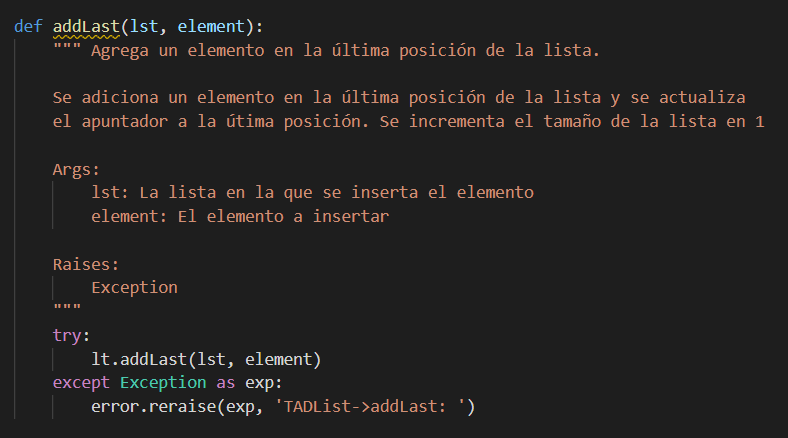


Figura 7: Función addLast()

La función addLast() añade un elemento a la última posición de la lista y actualiza el apuntador a la última posición mediante el incremento del tamaño de la lista en 1. Esta función hace parte del TAD Lista ya que permite realizar esta acción en una lista de tipo Array o en una Single Linked List. No obstante, es importante tener en cuenta que la efectividad de esta función variará dependiendo del tipo de estructura de datos que se utilice.

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

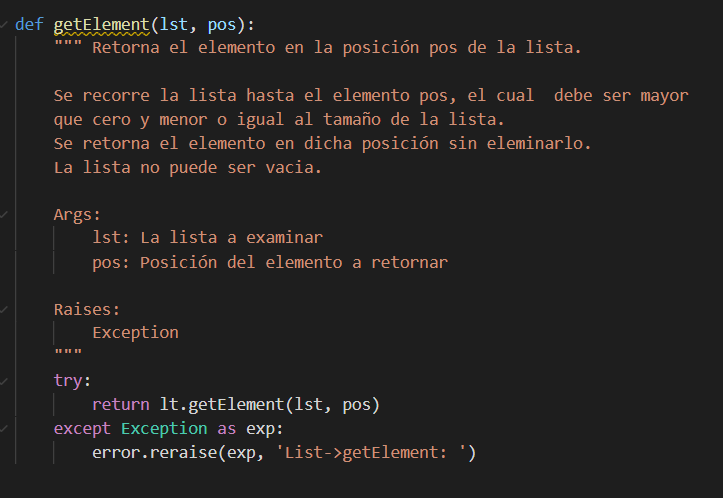


Figura 8: Función getElement()

La función getElement(lst, pos) recibe como parámetros la lista sobre la que se está trabajando y una posición en específico (a partir del 1). Con base en esta información esta función retornará e objeto que se encuentra en dicha posición (pos) dentro de la lista (lst). Nuevamente, el tiempo de impementación variará dependiendo de la estructura de datos utilizada.

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

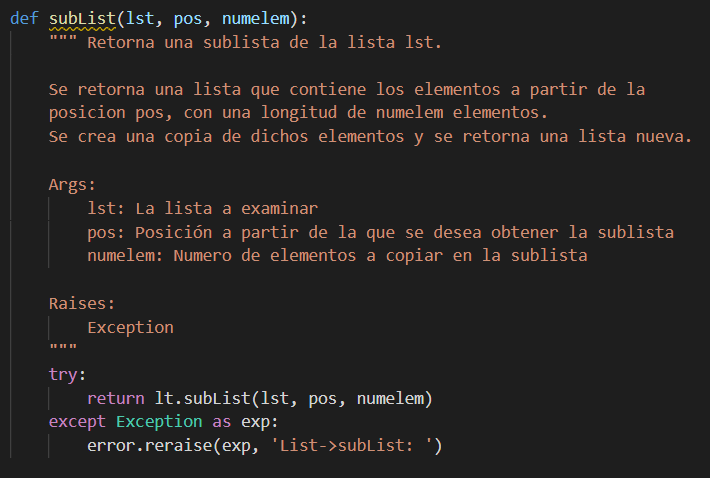


Figura 9: Función subList()

Esta función recibe tres parámetros: lst, post y numelem. El objetivo de esta función es el de retornar una sub-lista a partir de la lista principal (lst). Para determinar el rango de elemento que tomará la sub-lista se utiliza una posición (pos) y una cantidad de elementos (numelem), entonces se generará una sublista a partir de la posición pos y de tamaño numelem.

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