OBSERVACIONES DEL LA PRACTICA

Carlos Arturo Holguín Cardenás Cod 202012385

Daniel Hernández Pineda Cod 202013995

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

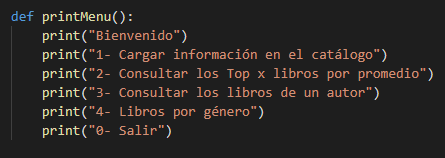


Ilustración 1: Menu de vista

En la ilustración 1, se observa el menú de la vista, esta corresponde a un mecanismo Output

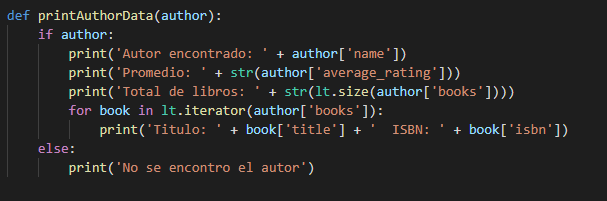


Ilustración 2: Función sobre los datos del autor

La ilustración 2, corresponde a un mecanismo Output, se puede observar que la función imprime datos sobre el autor de búsqueda, algunos son: Nombre del autor, promedio, total de libros, etc. En caso, de que no se encontró el autor.

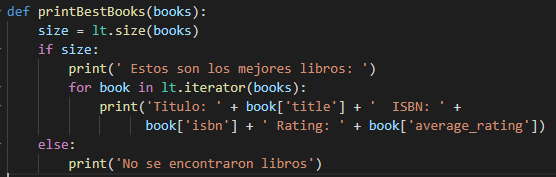


Ilustración 3: Función sobre los mejores libros

La ilustración 3, corresponde a un mecanismo Output, se puede observar que la función imprime datos sobre el mejor libro, estos son: titulo, ISBN y rating. En caso, de que no se encuentre libro se imprime no se encontraron libros.

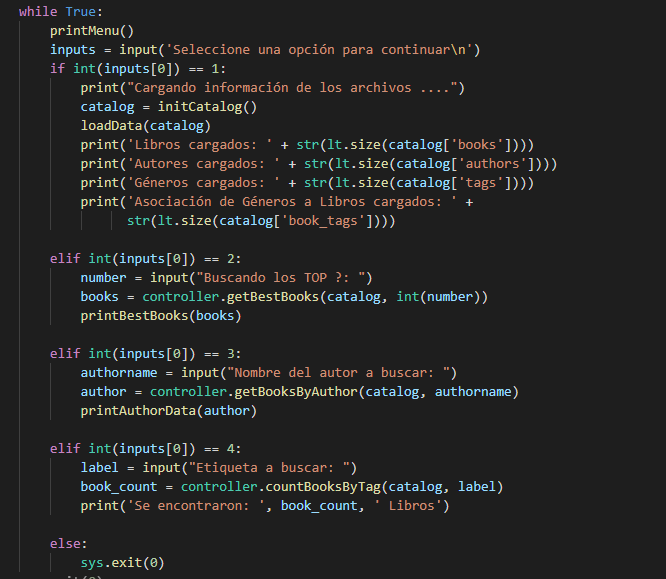


Ilustración 4: Menu principal

En la ilustración 4, se observa el menú principal, esta corresponde a un mecanismo Input/Output, ya que, en primer lugar, se le muestra al usuario al menu para posteriormente esperar a que el usuario digite el numero respectivo, para responder su pregunta de interes. Finalmente, se imprime los datos que el usuario buscaba, los cuales se encuentran en funciones anteriormente creados.

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

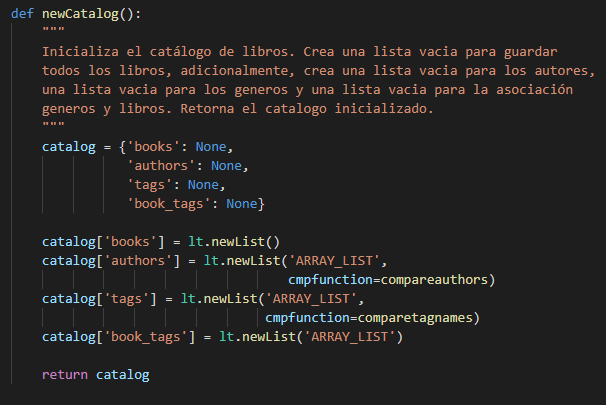


Ilustración 5: Almacenamiento de datos

En la ilustración 5, se observa que los datos se guardan en forma de “ARRAY LIST”, este corresponde al TAD. Además, se observa que se crean listas vacías para los libros, autores, etc.

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

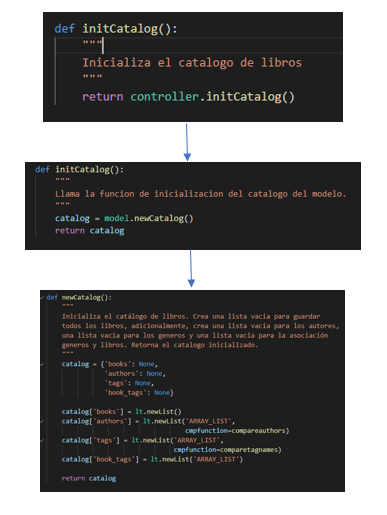


Ilustración 6: Comunicacion en initCatalog

En la ilustacion 6, se puede observar la comunicación de la funcion newCatolog entre la vista y el modelo, pero antes debe entrar en el controlador.

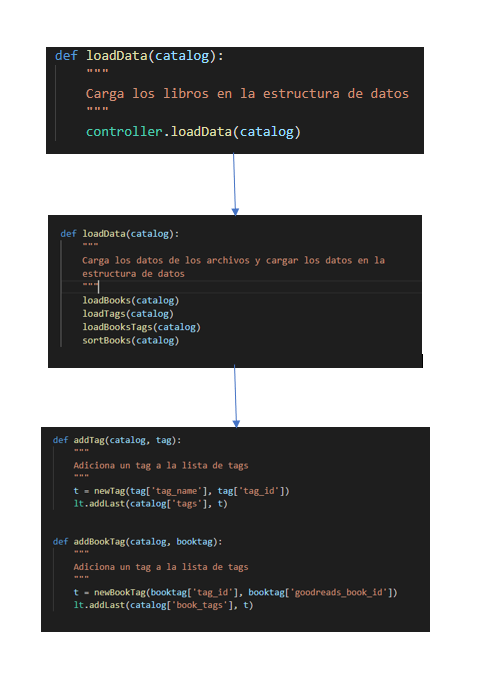


Ilustración 7: Conexion de funcion load data

La funcion loadData() en la vista se redirecciona al controlador, en donde se crean cuatro subfunciones con las direcciones de los archivos, estas funciones nos van a servir para cargar los datos. Posteriormente, en el modelo se encuentran cuatro funciones encargadas de guardar los datos.



Ilustración 8: Funciones de busqueda

En la ilustracion 8, se puede observar las funciones de búsqueda. En la vista se solicita un número para buscar la información en específico. Posteriormente se remite al controlador, para finalmente, realizar la lógica en el modelo. Tanto en el el modelo como en el controlador hay 3 funciones para el menú.

1. ¿Cómo se crea una lista?

Para crear una lista, la función newList() invoca la función newList() del archivo **liststructure.py**, el cual, a su vez, discrimina el tipo de lista (arreglo o lista encadenada) para invocar la función newList() que corresponda. En cambos casos, las listas se guardan como diccionarios de python, teniendo en común llaves como el tamaño (“size”), un identificador que se usa para comparar elementos de la lista (“key”), el tipo (“type”) y una función de comparación (“cmpfunction”).

Cuando se trata de un arreglo, la función newList() que se encuentra en el archivo **arraylist.py** crea una lista vacía que, además de las características descritas anteriormente, tiene una llave “elements” en la cual se guarda una lista de python que almacena la información deseada.

Cuando se trata de una lista encadenada, la función newList() que se encuentra en el archivo **singlelinkedlist.py,** además de las características descritas anteriormente, tiene una llave “first” que contiene el primer nodo y una llave “last” que contiene el último nodo. Cada nodo es a su vez un diccionario de python, cuyas llaves son “info” y “next”: “info” contiene la información que se desea que guarde este elemento de la lista, mientras que “next” contiene el nodo en la posición que le sigue.

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

El parámetro **cmpfunction=None** es un parámetro opcional en el cual se brindan funciones de comparación, las cuales, como su nombre lo dice, se encargar decomparar elementos específicos.

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

La función addLast() agrega un elemento al final de una lista. Los únicos dos parámetros de esta función son, precisamente, la lista y el elemento a agregar al final de esta. Esta función invoca la función addLast() del archivo **liststructure.py**, el cual, a su vez, discrimina el tipo de lista (arreglo o lista encadenada) para invocar la función addLast() que corresponda.

Cuando se trata de un arreglo, la función addLast() que se encuentra en el archivo **arraylist.py** recurre al método .append() para agregar el respectivo elemento al final del arreglo. Además, aumenta en 1 el tamaño de la lista.

Cuando se trata de una lista encadenada, la función addLast() que se encuentra en el archivo **singlelinkedlist.py** revisa primero si la lista es vacía o no. En caso de serlo, entonces crea un primer (y último) nodo. En caso de no ser vacía, entonces crea un nuevo nodo *new\_node* cuyo *new\_node[“next”]* será *None* y, a su vez, modifica el actual último elemento *lst[“last”]* para que *lst[“last”][“next”]* sea *new\_node* y *lst[“last”]* pase a ser *new\_node*. En cambos casos, se aumenta en 1 el tamaño de la lista.

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

La función getElement() retorna el elemento que se solicitó según la posición dada por parámetro. Esta función invoca la función getElement() del archivo **liststructure.py**, el cual, a su vez, discrimina el tipo de lista (arreglo o lista encadenada) para invocar la función getElement() que corresponda.

Cuando se trata de un arreglo, la función getElement() que se encuentra en el archivo **arraylist.py** puede acceder directamente al elemento debido a que conoce su posición, sabiendo que la posición en el arreglo está dada por *pos – 1* (*pos* es la posición que se dio por parámetro).

Cuando se trata de una lista encadenada, la función getElement() que se encuentra en el archivo **singlelinkedlist.py** recorre todos los elementos de la lista hasta que encuentra la posición dada, y retorna el elemento que se encontró en el nodo respectivo.

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

La función subList() retorna un fragmento de lista (sublista) según la posición de inicio y el tamaño de la sublista dados por parámetro. Esta función invoca la función subList() del archivo **liststructure.py**, el cual, a su vez, discrimina el tipo de lista (arreglo o lista encadenada) para invocar la función subList() que corresponda.

Cuando se trata de un arreglo, la función subList() que se encuentra en el archivo **arraylist.py** crea una lista vacía, y le va agregando elementos consecutivos con el método .append() a partir de la posición *pos-1* (*pos* es la posición que se dio por parámetro) hasta que alcanza el tamaño que se pidió por parámetro.

Cuando se trata de una lista encadenada, la función subList() que se encuentra en el archivo **singlelinkedlist.py** crea una lista vacía. Para encontrar el elemento de arranque (primer nodo), se recurre a la función getElement(), la cual devuelve el elemento deseado según la posición dada por parámetro. A partir de ahí, se van agregando elementos con la función addLast() hasta que la lista alcanza el tamaño que se pidió por parámetro.

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

En este caso, el cambio mas notable se encuentra en el tiempo, el tiempo de carga con el tipo **ARRAY\_LIST** fue de 0.13 S, mientras que con el **SINGLE\_LINKED** fue de 0.16 S, esto con los archivos de books\_tags-small, books-small y tags. Por otro lado, con los archivos grandes el tiempo de carga para **ARRAY\_LIST** fue de 351.03 S, mientras que con el **SINGLE\_LINKED** fue de 368.29 s. Ademas, de el tiempo hay otros diferencias como el funcionamiento del los **TAD** y las ventaja y desventajas que nos ofrecen cada uno, aunque lo anterior no es notable con solo correr el codigo.