OBSERVACIONES DEL LA PRACTICA

Sofia Velasquez Marin Cod 202113334

Valeria Caro Ramirez Cod 202111040

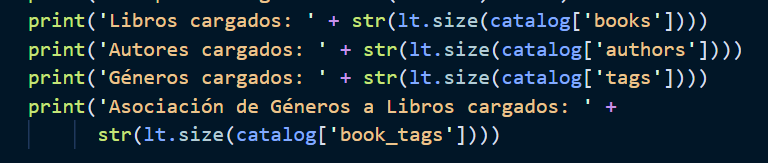
1. ¿Cuáles son los mecanismos de interacción (I/O: Input/Output) que tiene el **view.py** con el usuario?
2. ¿Cómo se almacenan los datos de **GoodReads** en el **model.py**?
3. ¿Cuáles son las funciones que comunican el el **view.py** y el **model.py**?
4. ¿Cómo se crea una lista?
5. ¿Qué hace el parámetro **cmpfunction=None** en la función **newList()**?
6. ¿Qué hace la función **addLast()**?
7. ¿Qué hace la función **getElement()**?
8. ¿Qué hace la función **subList()**?
9. ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**?

Respuestas:

1. Input:



Output:

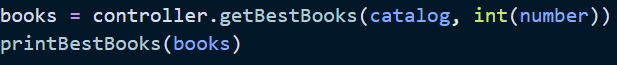


1. Input:





Output:



1. Input:



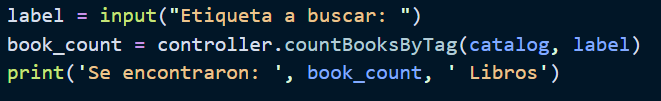


Output:

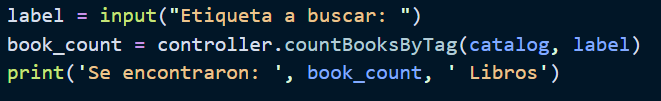


1. Input:





Output:



1. Input:



Output:



1. En un diccionario con las llaves “books”, “authors”, “tags”, “book\_tags” cuyos valores son listas.

* En la lista de “authors” se almacena un diccionario con las llaves ‘name’, su valor es un string con el nombre del autor; ‘books’ y como valor una nueva lista; y ‘average\_rating’.
* En la lista de “tags” se almacena un diccionario con las llaves ‘name’, su valor es un string con el nombre del tag y ‘tag\_id’.
* En la lista de “book\_tags” se almacena un diccionario con las llaves ‘tag\_id’ y ‘book\_id’. Este diccionario crea una relación entre un tag y los libros que han sido marcados con dicho tag.

1. -----------------
2. Para crear una lista se utiliza la función **newList()**, esta recibe 5 parámetros dataestructures, cmpfunction, key, filename y delimiter. A continuación, se hará un breve resumen de los parametros:

* Dataestructures: Tipo de estructura de datos a utilizar para implementar en la lista. Los tipos pueden ser: ARRAY\_LIST Y SINGLE\_LINKED.
* Cmpfunction: Función para comparar los elementos de la lista
* Key: Identificador para comparar los elementos de la lista
* Filename: Crea una lista a partir de los elementos encontrados en el CSV UTF8.
* Delimiter: Se utiliza este valor para separar los campos y el valor por defecto son las comas.

1. -----------------
2. La función **addLast()** se encarga de agregar un elemento en la última posición de la lista.
3. La función **getElement()** se encarga de retornar el elemento en la posición pos de la lista, esto lo hace recorriendo toda la lista hasta encontrar el elemento pos, este si o si debe ser mayor a cero y menor o igual al tamaño de la lista. Para que esta función pueda funcionar correctamente la lista que recibe no puede estar vacía porque en tal caso, no se podrá retornar el elemento en la posición pos de la lista. La imagen a continuación demuestra un ejemplo de su funcionamiento:

Texto

Descripción generada automáticamente

1. La función **subList ()** se encarga de retornar una sublista de la lista, esta recibe tres parámetros la lista previamente creada, la posición del elemento en la lista original desde la cual se desea realizar una copia de los elementos y el número de elementos a copiar en la nueva lista. La imagen a continuación demuestra un ejemplo de su funcionamiento:

Texto

Descripción generada automáticamente

1. Para observar el cambio del comportamiento se utilizó la librería de time para detectar en cuanto tiempo se ejecutaba el programa al cambiar la implementación del parámetro **“ARRAY\_LIST”** a **“SINGLE\_LINKED”**. De ello se obtuvo que al ejecutar el programa con el primer parámetro se tardó aproximadamente \_\_\_\_\_\_\_ (segundos o minutos), mientras que con el segundo se tardó \_\_\_\_\_\_\_\_ (segundos o minutos). A partir de ahí, se puede concluir que \_\_\_\_\_\_\_\_\_\_\_.