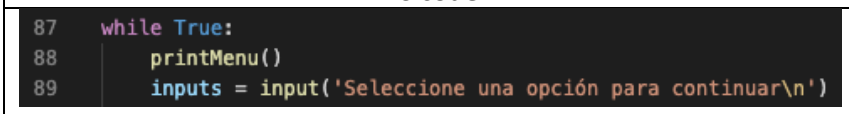
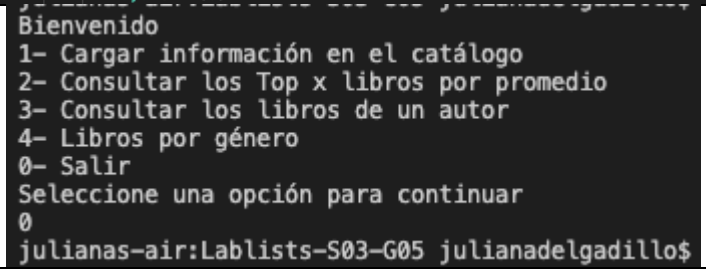
OBSERVACIONES DEL LA PRACTICA

María Alejandra Moreno Bustillo Cod. 202021603
Juliana Delgadillo Cheyne Cod. 202020986

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

El menú principal del programa ofrece al usuario 5 posibles alternativas para ejecutar determinado funcionamiento con los datos cargados. El usuario por su parte, debe escibir el número correspondiente a la opción que desea con el fin de que esta sea ejecutada. Dependiendo del número que el usuario escriba en la consola, se ejecutará la función correspondiente.

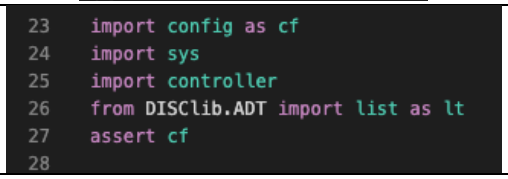
VS code	Descripción
	Se imprime el menú de opciones, se solicita al usuario igresar el número de la opción que desea ejecutar.
	A partir de condicionales, el programa identifica que opción es la que el usuario desea ejecutar (según el número introducido) y de acuerdo a ello llama las funciones necesarias para generar una respuesta acorde a lo solicitado.
	Resultado en consola para la visualización del menú y la solicitud (input) de seleccionar una opción.

2) ¿Cómo se almacenan los datos de **GoodReads** en el **`model.py`**?

En el código se puede evidenciar que los datos de GoodReads se cargan de la siguiente manera: Primero, una función llamada "newCatalog()" se encarga de crear un diccionario llamado catalog que tienes las siguientes llaves: books, authors, tags y book_tags. Luego, se crean cuatro listas vacías dentro de su correspondiente llave, de las cuales 3 utilizan la estructura de array_list y una de ellas usa la estructura predeterminada que es Single_linked. Estas listas estarán destinadas para almacenar la información de los libros que hay, los tags, los autores y los book_tags. Luego, se crean unas funciones específicas para recorrer el archivo e ir añadiendo la información correspondiente a cada una de las listas.

3) ¿Cuáles son las funciones que comunican el **`view.py`** y el **`model.py`**?

Para comunicar el archivo view.py con el model.py es necesario la incorporación del controller pues estos dos archivos no cuentan con una comunicación directa. Para llevar acabo esta conexión el controller importa a model y lo utiliza para llamar determinadas funciones creadas en dicho archivo, que permitan obtener los datos organizados con los que se desea interactuar. Asimismo, desde view se importa el archivo controller y se utiliza para llamar funciones creadas que satisfagan lo solicitado por el usuario. De esta maner cuando el usuario pide ejecutar una opción, view llama a controller para ejecutar dicha función y a su vez controller llama a model para obtener los datos necesarios para llevar a cabo aquello que se solicita.

VScode	Descripción
	Se evidencia la importación que se lleva a cabo de model dentro del archivo controller.py
	Se evidencia la importación que se lleva a cabo de controller dentro del archivo view.py

4) ¿Cómo se crea una lista?

Las listas se crean en el archivo list.py, el cual importa la librería "DISClib.DataStructures". Gracias a esta librería, en la función "newList" es posible invocarla y así retornas una nueva lista creada con la estructura de datos y otras especificaciones puestas como parámetros para esta función. Más a profundidad, en la librería que se invoca se pueden encontrar los archivos que definen como se crean en realidad estas listas como "singlelinkedlist.py" entre otros que muestran como se crea un diccionario con las diferentes llaves que se requieren y luego a estas se les van agregando la información correspondiente de acuerdo a los parámetros que reciba.

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

El parámetro cmpfunction es usado para poder comparar los elementos dentro de la lista que se está creando en la función newList(), al igualar el valor a none se está diciendo que se busca proveer a la lista una función de comparación.

6) ¿Qué hace la funció **`addLast()`**?

   La función addlast() se encarga de añadir un elemento en la última posición de la lista, luego actualiza el apuntador de la última posición para que apunte al nuevo elemento que es el último e incrementa el tamaño de la lista en 1. Esto lo hace invocando la misma librería "DISClib.DataStructures".

7) ¿Qué hace la función **`getElement()`**?

   La función getElement() se encarga de retornar un elemento en una determinada posición. Lo que esta función hace es recorrer la lista hasta llegar a la posición especificada en el parámetro que le entra a la función y retornar el elemento que se encuentra en dicha posición sin eliminarlo de la lista.

8) ¿Qué hace la función **`subList()`**?

La función subList() se encarga de actualizar un elemento dentro de una lista en una posición específica. El retorno que esta produce es una sublista de la lista original de acuerdo a una posición y un número de elementos el cual indicará cuantos elementos, a partir de la posición dada, se encontrarán en la sublista.

9) ¿Observó algún cambio en el comportamiento del programa al cambiar la implementación del parámetro **"ARRAY_LIST"** a **"SINGLE_LINKED"**?

Al cambiar el tipo de estructura de datos en el archivo model.py, escribiendo SINGLE_LINKED como paramentro, se notó un aumeto de tiempo significativo en la respuesta al usuario al ejecutar la opción 1, correspondiente a cargar información en el catálogo, esto se debe a que en este caso utilizar una arreglo de lista simplifica los procesos internos que se llavan a cabo para cumplir con la función que se desea ejecutar. Esto implica que el orden de crecimiento para la lista encadenada es mayor que el del arreglo y por ende es menos eficiente.