**María Alejandra Estrada García**

**202021060**

**Santiago Martínez Novoa**

**202112020**

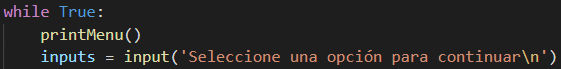
**Respuestas Paso 3:**

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

**R//**

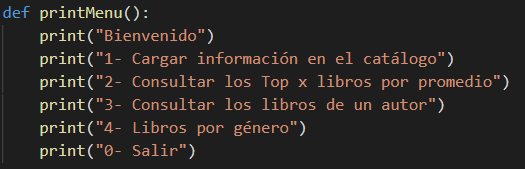
Input:

* La funcón “Input” funciona para que el usuario ingrese la información deseada. Un ejemplo es en las condicionales del Menú Principal, donde pide al usuario que seleccione una de las opciones que aparecen en la función printMenu().



Output:

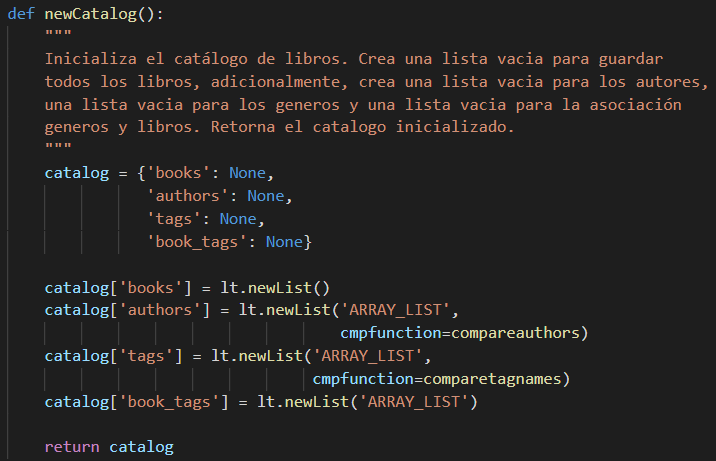
* La función “print” imprime –presenta- el texto o variables en pantalla al usuario. Un ejemplo es en la función printMenu() donde muestra en pantalla al usuario las diferentes opciones que puede escoger.



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

**R//** Los datos estan guardados en una lista, por medio del parámetro

“ARRAY\_LIST” en la función newCatalog().



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

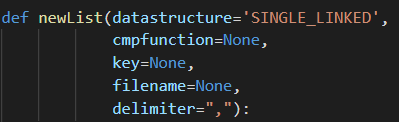
**R//** En primer lugar, no hay una función directa que se comunique desde el view.py al model.py, todas pasan primero por el controller.py. Esto se debe a que se utilizó el patrón Modelo-Vista-Controlador.   
Teniendo esto en cuenta, las funciones que están el view.py y que se conectan con el model.py son:

* initCatalog(): esta función llama al controlador a la misma función donde le pide al model que inicie el catálogo, donde se encuentran organizados los datos de GoodReads.
* getBestBooks(): Esta función por medio del controller llama al model para que retorne los mejores libros. El top #, con el número siendo el parámetro ingresado por el usuario.
* getBooksByAuthor(): Por medio de esta función le pide al controller que llame al model para que retorne un autor y sus libros a partir del nombre del autor que pide el usuario.
* countBooksByTag(): esta función le pide al controller que llame al model para que retorne los libros que fueron etiquetados con el tag.

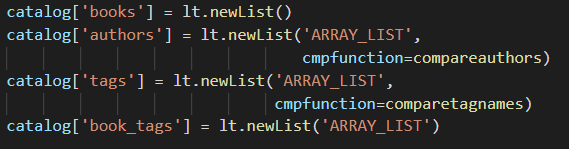
**Respuesta Paso 4:**

* **¿Cómo se crea una lista?:**

La lista se crea a través de la función newList(): Esta función recibe como parámetros la DataStructure que se quiere utilizar, el parámetro con el que va a comparar los elementos de la lista, el identificador utilizado para comparar dos elementos de la lista, el archivo del que se quieren sacar los datos y el delimitador para separar los valores del archivo utilizado. Sin embargo, el programa ya viene con una configuración predeterminada para cada parámetro de esta función:



La función por defecto crea una lista vacía, para modificar cualquiera de los parámetros solo hay que especificarlos a la hora de llamar la función:



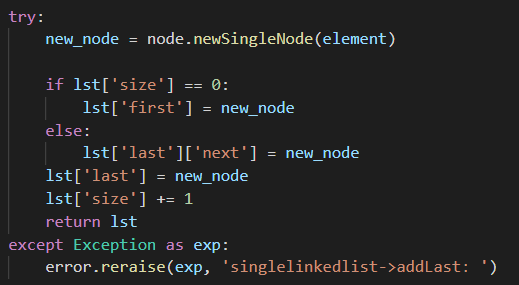
* **¿Qué hace el parámetro cmpfunction=None en la función newList()?**

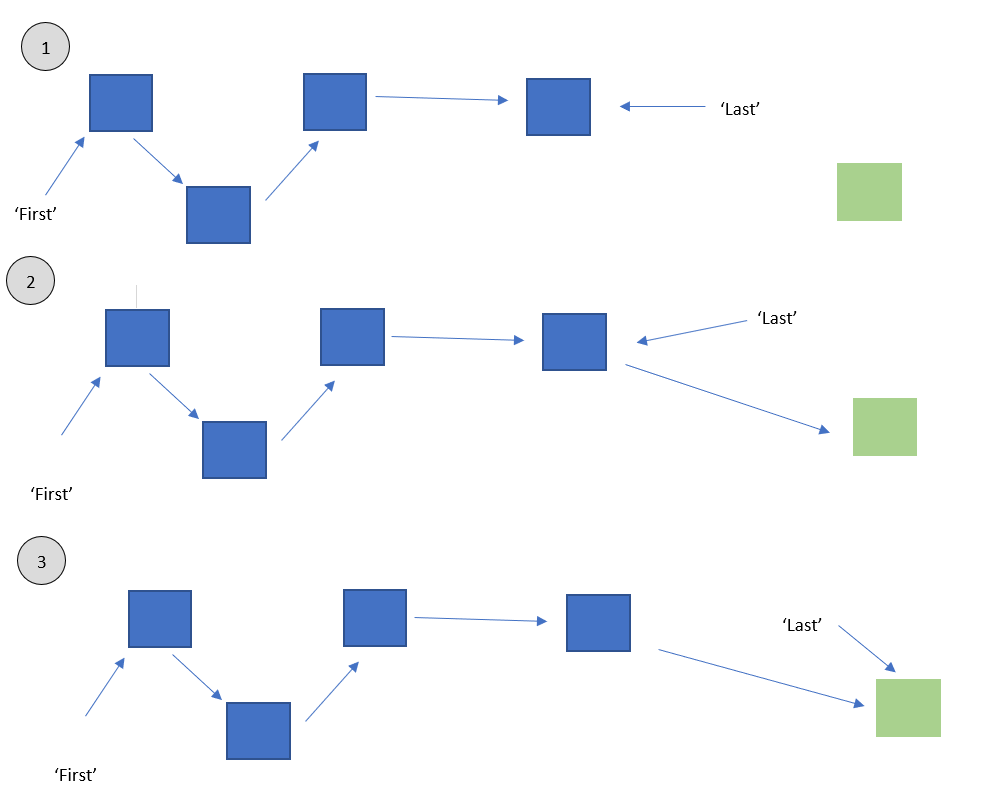
Si no se provee una función de comparación para cmpfunction este hará las comparaciones con el parámetro Key. Esto quiere decir que existe una relación entre ambos parámetros, si el cmpfunction=None entonces Key debe tener algún valor.Si por el contrario se ha brindado una función de comparación, Key = None.

* **¿Qué hace la función addLast()?**

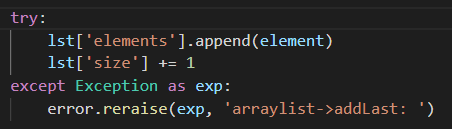
La función recibe una lista y el elemento añadir. Luego de eso la función depende de la DataStructure que se esté implementando.

Para SINGLE\_LINKED:

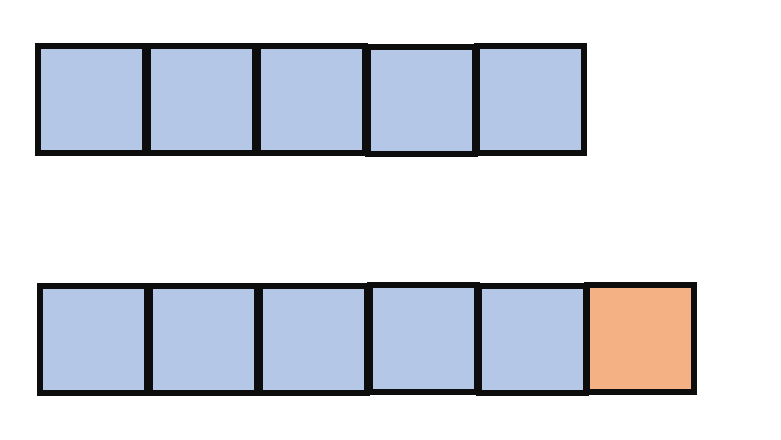


Representación gráfica:  


Para ARRAY\_LIST:

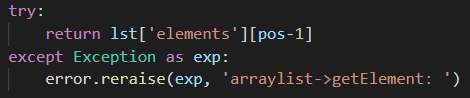


Representación gráfica:

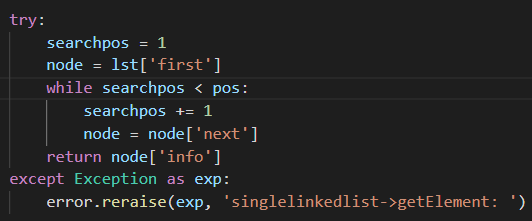


* **¿Qué hace la función getElement()?**

La función getElement() recibe como parámetro una lista y una posición específica y retorna el elemento que se encuentre en esa posición. Para el ARRAY\_LIST es solo cuestión de llamar el elemento con lista[posición-1]:



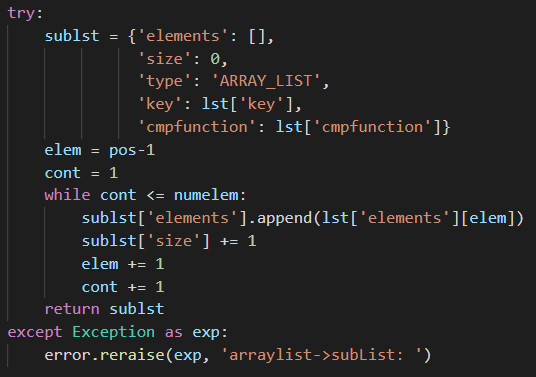
Mientras que en SINGLE\_LINKED se utiliza un while loop, y mediante un contador se itera a lo largo de cada elemento de la lista hasta encontrar la posición deseada:



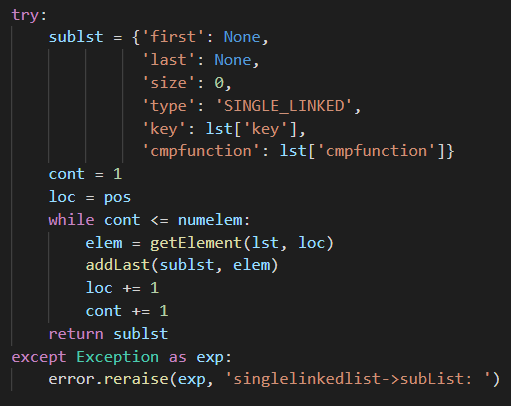
* ¿**Qué hace la función subList()?**

La función subList() retorna una lista de elementos a partir de una posición dada por el usuario, con una longitud de cierto número de elementos también dado por el usuario.

De nuevo aquí va a depender de la estructura de Datos implementada. Si se utiliza ARRAY\_LIST se debe utilizar un while loop que “copie” en una nueva lista los elementos que cumplan con los parámetros de longitud y posición establecidos.



Para SINGLE\_LINKED también será a través de un while loop que se realizará la nueva lista, la diferencia en este caso con la otra estructura de datos será la manera que se añaden los elementos a la sublista(se llaman a funciones mencionadas anteriormente) y las llaves utilizadas para la sublista:



**Respuestas Paso 5:**

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

**R//** La SINGLE\_LINKED toma más tiempo en construirse y en cargar los datos, esto se debe a que cada lista tiene más elementos de los que tiene que estar atento, como la información de lst[‘next’], lst[first], lst[‘last’] entre otros, por lo cual no solo ocupa más memoria sino también se demora más buscando información posición por posición. Igualmente, a la hora de construir la lista de elementos.