OBSERVACIONES DEL LA PRÁCTICA

Federico Melo Barrero, 202021525

Juan Camilo Prieto Avella, 201814815

**PASO 3: Estudiar el ejemplo en VS Code**

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

El **view.py** interactúa con el usuario a través de un menú de opciones, que despliega cinco posibilidades. De esas cinco:

* La primera tiene como función cargar los datos.
* Las tres siguientes dan al usuario la posibilidad de realizar solicitudes, una solicitud distinta por cada opción:
  + Consultar los Top x libros por promedio.
  + Consultar los libros de un autor.
  + Libros por género.
* La última brinda al usuario la posibilidad de salir del menú, deteniendo la ejecución del programa.

La primera opción no requiere de input del usuario y como output imprime qué datos se han cargado. Las tres opciones que permiten al usuario realizar solicitudes sí requieren un input del usuario e imprimen la información solicitada de acuerdo al input ingresado.

No sé si haya que ahondar en esta respuesta… o sea no sé si pretenden que se desglose el funcionamiento de cada función: cómo se arma el menú, cómo se dan los primeros *n* libros, etc.

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

Tuyo.

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

Tuyo.

**PASO 4: Estudiar el uso de listas**

1. ¿Cómo se crea una lista?

Una lista se crea a través de la función **newList**. Esa función está definida en la librería del curso, concretamente en el directorio ADT donde están las estructuras abstractas de datos, en el archivo **lis.py**. La función sólo recibe parámetros opcionales, si se usa sin parámetros genera una lista encadenada. Los parámetros opcionales son: el tipo de estructura de datos, que por defecto es una lista encadenada; la función de comparación; el identificador para la función de comparación por defecto, que debe añadirse en caso de que no se incluya por parámetro una función de comparación; el nombre de un archivo CSV, que si es provisto es usado para generar la lista a partir del archivo; y el separador para generar la lista a partir del CSV, que por defecto es una coma.

Dentro de la función **newList** del archivo **lis.py**, se utiliza la función **newList** del archivo **liststructure.py**, en el directorio de DataStructures donde están las estructuras de datos. Esa función básicamente dirige a la estructura de datos que se desea (que entró como parámetro), ejecutando la función **newList** del archivo **arraylist.py**, que genera un arreglo, si el parámetro es **"ARRAY\_LIST"**, o la función **newList** del archivo **singlelinkedlist.py**, que genera una lista encadenada, de lo contrario.

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

El parámetro opcional **cmpfunction** corresponde a la función de comparación. Esa es una función que diseña el programador para cuando se necesite comparar dos elementos de la lista. Dependiendo del tipo de elementos en una lista, tiene sentido comparar diferentes atributos (i.e., tamaños, etiquetas, valores, códigos, entre otros), que Python no puede identificar por sí mismo. La función de comparación que se pasa por ese parámetro es la que se usará en la función **compareElements** que compara dos elementos de la lista entre sí.

Como el parámetro es opcional, si no se pasa una función de comparación el programa se remite a la función de comparación por defecto **defaultfunction**, que requiere del identificador. (Se habló del identificador en el punto anterior, es uno de los parámetros opcionales al crear una lista con la función **newList**. Siempre se debe incluir o bien la función de comparación o bien el indicador para que pueda funcionar la función de comparación preestablecida.)

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

La función **addLast()** ejecuta la operación del TAD lista de añadir un elemento a la última posición de la lista. Nótese que dependiendo de la estructura de datos que se use, la complejidad de ejecutar esa operación puede variar. La función **addLast()** en un arreglo hace lo mismo que en una lista encadenada pero de forma distinta.

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

La función **getElement()** ejecuta la operación del TAD lista de retornar aquel elemento de la lista que se encuentra en una posición específica dada como parámetro. Al igual que antes, su funcionamiento exacto varía de acuerdo a la estructura de datos.

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

La función **subList()** ejecuta la operación del TAD lista de retornar una sublista, es decir, un pedazo específico de la lista. Recibe como parámetros una posición y un número de elementos (además de la lista sobre la que se va a ejecutar la función, claro está) y retorna el segmento de la lista que inicia en la posición dada y cuya longitud es igual al número de elementos que entró por parámetro.

**PASO 5: Cambios en el uso de TAD Lista (ADT List)**

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

Tuyo. No he hecho el cambio en el código, o sea aún está como “ARRAY\_LIST”.