LABORATORIO 3: ARREGLOS Y LISTAS ENCADENADAS

1 Objetivo

Comprender la implementación del Tipo Abstracto de Datos Lista (ADT List) y el uso de esta estructura para solucionar problemas. Al finalizar este laboratorio el estudiante estará en capacidad de:

1) Desarrollar las operaciones principales del TAD Lista.
2) Utilizar la estructura de datos Lista implementada para almacenar datos.
3) Realizar la carga de archivos de tipo **`.csv.`**

2 Trabajo Propuesto

El trabajo del laboratorio se basa en el proyecto Laboratorio-3. Bifurque el repositorio en su organización siguiendo las convenciones de nombramiento **Laboratorio3-G<<YY>>** donde **YY** el número del grupo de trabajo (ej.: Laboratorio3-G01).

Agregue los nombres y usuarios de los integrantes del equipo en el archivo README del repositorio, esto solo debe hacerlo un estudiante del equipo. No olvide subir los cambios a GitHub y actualizar los repositorios de los demás estudiantes del grupo.

Descargue o copie los datos de prueba GoodReads usados anterior mente en los laboratorios 1 y 2 desde el aula unificada de BN y guárdelos en la carperta /Data/GoodReads/ del proyecto.

2.1 Implementación de estructuras

En este laboratorio implementará la primera estructura de datos, usando la documentación del curso encontrada en DISC – Data Structures como guía para el desarrollo de las estructuras que le pediremos en el curso.

Al abrir el enlace de la documentación lo reenviará a una página como la que se presenta en la siguiente captura de pantalla:

Ilustración 1Página de inicio de documentación

Lea detalladamente el apartado *Guía de documentación* donde encontrará una breve explicación de que encontrará dentro de la documentación del curso y un par de ejemplos de cómo convertir el contenido de la documentación al código de sus laboratorios y retos. En la sección izquierda de la documentación encontrará las estructuras de datos junto a sus implementaciones o componentes fundamentales para el desarrollo de la implementación. Para este laboratorio se implementará las ***Listas*** como ***Array List*** **y** ***Linked List***, ingrese a la documentación de array_list.py como se presenta en la siguiente captura de pantalla:

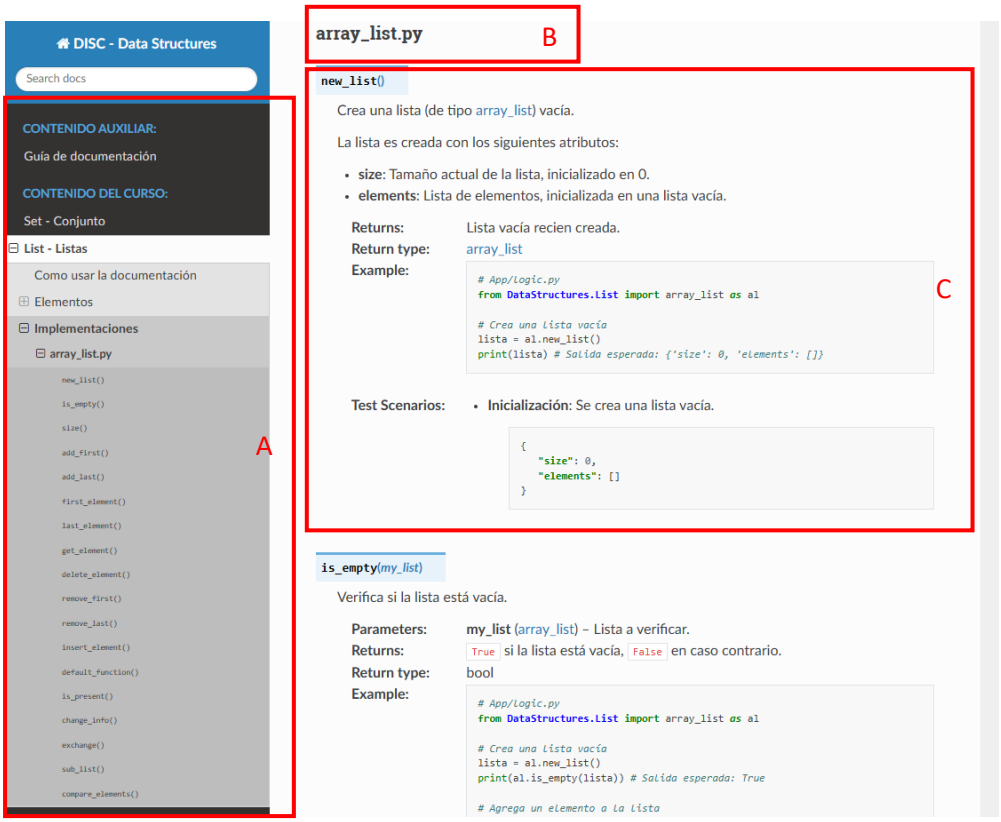

Ilustración 2 - Implementación de Lista con Array List

En la ilustración 2 se pueden identificar los 3 apartados de la documentación con los cuales se interactuará para identificar los desarrollos de las estructuras.

A. **Listado de estructuras e implementaciones:** en este apartado podrá navegar entre las diferentes estructuras vistas en el curso junto a sus implementaciones y métodos de cada una.
B. **Archivo de la implementación de estructura:** nombre con el que se debe crear el archivo dentro de su repositorio. Este nombre debe ser exactamente el mismo presentado en la documentación, en caso de que no sea exactamente el mismo puede presentarse el escenario que los tests no funcionen. Por ejemplo, para este laboratorio deberá crear un archivo llamado *array_list.py* y *single_linked_list.py* dentro de la carpeta List de la carpeta DataStructures.
C. **Funciones que implementar**: al ingresar al detalle del archivo de implementación encontrará las funciones que se espera que tenga la implementación completa del código. Dentro de la definición de las funciones encontrará la descripción de la función y que se espera que esta haga, los parámetros y el orden que espera la función y el retorno esperado de la función. Por ejemplo, en la ilustración 2 se aprecia la función *new_list()* la cual no espera ningún parámetro de entrada pero si indica que se espera de la lista creada junto a su tipo de dato de respuesta, que en este caso es una lista de tipo array_list.

2.2 Implementación de funcionalidad

Una vez esté clonado el repositorio en su máquina local encontrará una estructura de proyecto como la presentada en la Ilustración 3. La estructura es similar a la trabajada en los laboratorios 1 y 2 pero con la diferencia que esta vez la carpeta *DataStructures* solo contiene la carpeta *List* la cual solo contiene los Tests de la estructura de datos, el objetivo de este momento en adelante es que como equipo se encarguen de desarrollar las estructuras de datos vistas en el curso en los laboratorios y estas serán usadas en retos.

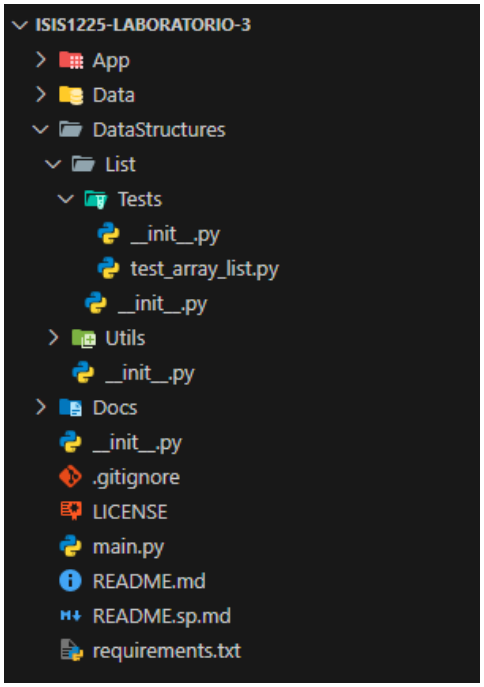

Ilustración 3 - Estructura del laboratorio 3

2.2.1 Implementando las primeras Listas

Teniendo en cuenta la documentación de DISC – Data Structures creará las primeras implementaciones de estructura de datos.

1. Cree los archivos *array_list.py y single_linked_list.py* dentro la carpeta *DataStructures/List/*. Debe asegurarse que este no sea creado dentro de la carpeta *Tests* que se encuentra dentro de *List*.
2. Lea la documentación de la función *new_list()* e identifique que se espera que se haga al momento de crear una nueva lista. Para este caso la lista solo tendrá que ser creada vacía y con un size inicializado en 0.
3. Copie y pegue el siguiente código en el archivo del *array_list.py* para implementar la función new_list()

```
def new_list():
    newlist = {
        'elements': [],
        'size': 0,
    }
    return newlist
```

Ilustración 4 - Implementación de new_list() en *array_list*

4. Copie y pegue el siguiente código en el archivo de *single_linked_list.py* para implementar la función new_list()

```
def new_list():
    newlist = {
        "first": None,
        "last": None,
        "size": 0,
    }

    return newlist
```

Ilustración 5 - Implementación de new_list() en *single_linked_list*

5. En *logic.py* importe la estructura recién creada de la siguiente manera:

```
# TODO Importar la librería para el manejo de listas
from DataStructures.List import array_list as lt
```

6. De esta manera se importarán las estructuras que crearemos en los laboratorios del curso. Para comprobar que el desarrollo de una función esté correctamente hecho, hemos creado una serie de pruebas pensadas para comprobar el desarrollo mínimo de la estructura esperada. Es necesario aclarar que estas pruebas no validan la complejidad temporal del código implementado sino solamente validan el correcto nombramiento y resultados parciales de las funciones de la estructura, por lo tanto, se podría hacer el desarrollo de una función con complejidad exponencial que soluciona el problema, pero no es óptima.
7. Para la ejecución de las pruebas es necesario instalar en Python la librería pytest (esto es necesario hacerlo solo una vez, una vez instalado no es necesario volverlo a instalar en los laboratorios futuros) Ejecute desde terminar de VS Code el siguiente comando:

pip install pytest

8. Abra el archivo *test_array_list.py* ubicado en la carpeta *Tests/* dentro de *List*. Este archivo contiene todas las pruebas unitarias hechas sobre la estructura array_list. Como se puede apreciar en la línea de código 2, los tests buscan el archivo con el mismo nombre definido en la documentación para la ejecución de las funciones; tanto el nombre del archivo como nombre de

las funciones y orden de los atributos de entrada de las funciones deben coincidir con los definidos en la documentación para que se puedan ejecutar de manera correcta

El decorador @handle_not_implemented presente justo antes de todas las funciones de tests, como el presente en la línea 21, maneja el error en el escenario donde la función no se ha implementado aún y omite la prueba, todo esto con el objetivo de omitir resultados con errores antes de implementar una función.

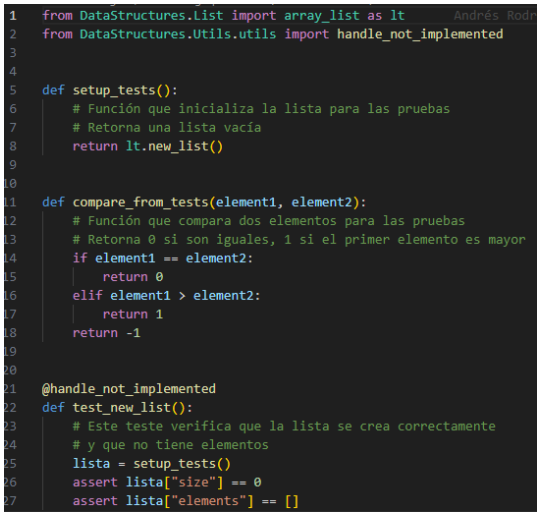

Ilustración 6 - Tests unitarios en test_array_list

9. Para facilitar la ejecución de los tests se ha creado un *script* llamado run_tests.py. Para correr el script ejecute el siguiente comando desde el directorio principal del laboratorio:

10. Al ejecutar el script aparecerá el menú de ejecución de los tests, ingrese la opción “2” luego presione enter. Deberá aparecer una ejecución como la presentada a continuación:

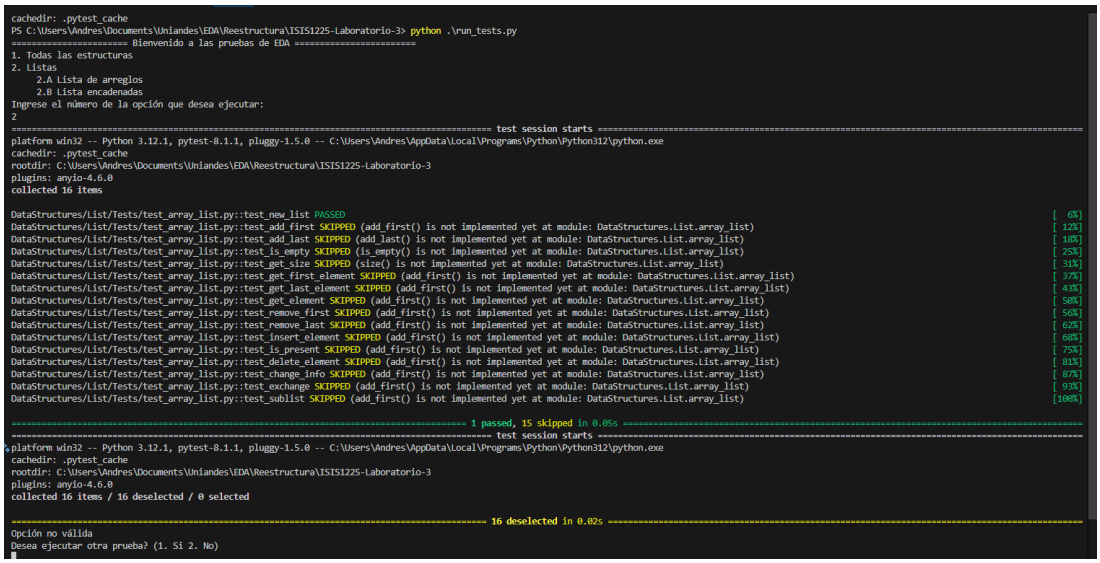

Ilustración 7 - Ejecución de tests sobre array_list

11. Como se muestra en la Ilustración 7, solo ejecuta las funciones implementadas hasta el momento. De esta manera evitará que al ejecutar las pruebas estos estén indiciando errores sobre funciones que aún no ha implementado.
12. Agregue las siguientes funciones en la implementación de array_list:

```
def get_element(my_list, index):

    return my_list["elements"][index]


def is_present(my_list, element, cmp_function):

    size = my_list["size"]
    if size > 0:
        keyexist = False
        for keypos in range(0, size):
            info = my_list["elements"][keypos]
            if cmp_function(element, info) == 0:
                keyexist = True
                break
        if keyexist:
            return keypos
    return -1
```

Ilustración 8 - funciones de apoyo en la implementación de array_list

13. Agregue las siguientes funciones en la implementación de *single_linked_list:*

```
def get_element(my_list, pos):
    searchpos = 0
    node = my_list["first"]
    while searchpos < pos:
        node = node["next"]
        searchpos += 1
    return node["info"]


def is_present(my_list, element, cmp_function):
    is_in_array = False
    temp = my_list["first"]
    count = 0
    while not is_in_array and temp is not None:
        if cmp_function(element, temp["info"]) == 0:
            is_in_array = True
        else:
            temp = temp["next"]
            count += 1

    if not is_in_array:
        count = -1
    return count
```

14. Tome como referencias la documentación del curso y las funciones compartidas anteriormente para implementar las funciones *add_first(), add_last(), size() y first_element()* para que la funcionalidad 1 de Carga de información en el catálogo de libros funcione correctamente. Implemente las funciones para ambas implementaciones de Listas, tanto *array_list* como *single_linked_list*. Por el momento solo debe usar una de las dos implementaciones para el cumplimiento de la funcionalidad 1.
15. Guarde y suba los cambios de clase a Github. Cree un ***release*** con el título “Entrega Inicial Laboratorio 3” y el tag “0.0.1” en su repositorio grupal.

2.2.2 Completando la estructura y aplicación

Continue trabajando en la implementación de las funciones necesarias para el desarrollo de la estructura *array_list* y *single_linked_list* tomando como guía la documentación del curso y la ejecución de las pruebas compartidas en este laboratorio

1. Complete las funciones que se encuentran incompletas en el archivo logic.py como se presenta en la Ilustración para que la aplicación de GoodReads quede completamente implementada con la lista Array_List

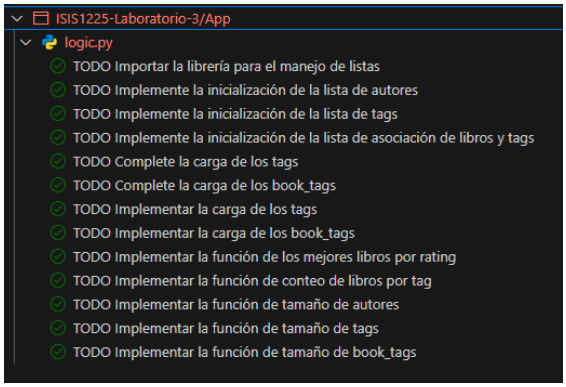

Ilustración 9 - ToDos del laboratorio 3

2. Asegúrese que al finalizar la implementación de las estructuras esté correctamente implementada ejecutando las pruebas unitarias y que estas estén completamente en color verde tal cómo se presenta en la Ilustración

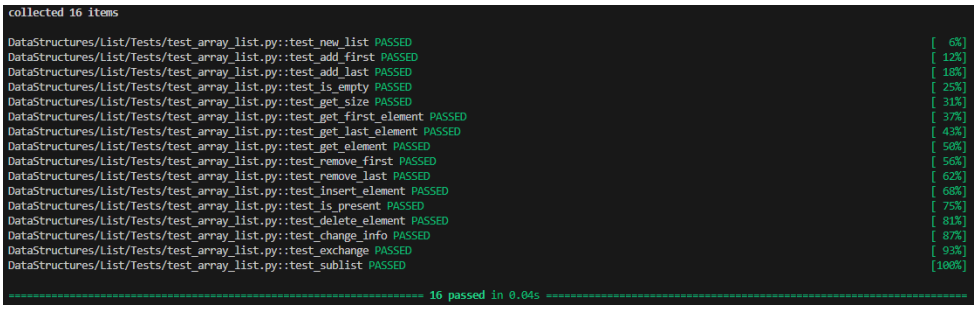

Ilustración 10 - Ejecución final de la estructura array_list

3 Entrega

3.1 Confirmar cambios finales

Confirme los cambios finales siguiendo los pasos aprendidos en prácticas anteriores y cree un *release* con el título “Entrega Final Laboratorio 3” antes de la fecha límite de entrega del laboratorio

3.2 Compartir los resultados con los evaluadores

Finalmente, para realizar la entrega de los resultados de la práctica de laboratorio se debe enviar el enlace (URL) del repositorio GitHub a los monitores y profesores de **su sección**, antes de la fecha definida en la actividad del curso.