Documentación sobre Listas Enlazadas

Autor: Luis Aguado Diego

Introducción

Este proyecto implementa diferentes tipos de listas enlazadas utilizando Python. Se incluyen:

1. Listas enlazadas simples.
2. Listas doblemente enlazadas.
3. Listas circulares enlazadas.
4. Listas circulares doblemente enlazadas.

Cada tipo de lista tiene funcionalidades básicas como insertar, eliminar, buscar, imprimir nodos y copiar la lista en un archivo.

Detalles de Implementación

Funciones Generales

- **`lista_vacia(lista)`**: Verifica si una lista está vacía.
- **`crear_nodo(valor)`**: Crea un nodo con los atributos `valor`, `siguiente` y `anterior`.
- **`insertar_inicio(lista, nodo)`**: Inserta un nodo al inicio de la lista.
- **`insertar_final(lista, nodo)`**: Inserta un nodo al final de la lista.

- **`insertar_nodo(lista, nodo_nuevo, valor_existente, posicion)`**: Inserta un nodo nuevo antes o después de otro nodo existente.
- **`contar_nodos(lista)`**: Devuelve la cantidad de nodos en la lista.
- **`eliminar_nodo(lista, valor)`**: Elimina un nodo por su valor.
- **`imprimir_valor_lista(lista)`**: Imprime solo los valores de los nodos.
- **`imprimir_lista_completa(lista)`**: Imprime los valores y las conexiones (`siguiente` y `anterior`) de cada nodo.
- **`imprimir_reves(lista)`**: Imprime los nodos desde el final al principio.
- **`buscar_nodo(lista, valor)`**: Busca un nodo en la lista por su valor.
- **`copiar_lista(lista, tipo)`**: Copia la lista en un archivo de texto.

Ejemplos de Uso

1. Lista Enlazada

Se crean nodos y se prueban las funcionalidades básicas:

- Insertar al inicio y al final.
- Insertar un nodo después de otro.
- Imprimir los valores de los nodos.

2. Lista Doblemente Enlazada

Además de las funcionalidades básicas, se prueba la navegación en ambas direcciones gracias a los enlaces `anterior` y `siguiente`.

3. Lista Circular Enlazada

El último nodo se conecta al primero para formar un ciclo.

4. Lista Circular Doble Enlazada

Aquí, además del enlace circular, también se conecta el primer nodo al último en el atributo `anterior`.

Dificultades Encontradas

1. **Reutilización de nodos:** Reutilizar los mismos nodos en diferentes listas causaba problemas debido a la manipulación de referencias. La solución fue crear nodos nuevos para cada lista.
2. **Comprensión de las conexiones circulares:** Asegurar que los nodos se conectaran correctamente en listas circulares (en ambos sentidos) fue complejo.
3. **Organización de la impresión:** Diseñar una forma clara de mostrar los nodos y sus conexiones tomó tiempo.
4. **Documentación:** Explicar los conceptos y asegurarse de que todo estuviera claro fue un desafío, pero permitió consolidar el aprendizaje.

Lecciones Aprendidas

- La gestión de referencias es crítica al trabajar con estructuras de datos dinámicas.
- La organización del código y una buena separación de funciones facilitan la comprensión y el mantenimiento.
- Es fundamental realizar pruebas exhaustivas, especialmente con listas

circulares.

Próximos Pasos

- Implementar validaciones más robustas.
- Extender las funcionalidades para incluir métodos de ordenamiento y búsqueda más eficientes.
- Crear una interfaz gráfica para visualizar las listas dinámicamente.