Trabajo Práctico N° 4 – Listas Pilas Colas Arboles

Ejercicio N° 1

Se tiene el programa `listasenlazadas_tp.py`. Modificar el programa, de manera tal que al ingresar un nuevo elemento a la lista enlazada, lo haga siempre al principio de la lista.

En el ejemplo:

```
10 → 70 → 30 → None
```

Debería mostrar:

```
30 → 70 → 10 → None
```

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def is_empty(self):
        return self.head is None

    def insert_at_beginning(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    def display(self):
        if self.is_empty():
            print("Linked list is empty")
        else:
            current = self.head
            while current:
                print(f"|{current.data}| -> ", end="")
                current = current.next
            print("None")
```

Ejercicio N° 2

Analizar el código del programa `pilas_tp.py`. Identificar la clase, los métodos. Agregar los métodos:

- Eliminar un elemento
- Visualizar la pila como pila

En el ejemplo:

```
| 5 |
| 10 |
| 15 |
| 8 |
| 2 |
```

Debería mostrar:

```
| 2 |
| 8 |
| 15 |
| 10 |
| 5 |
```

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Stack:
    def __init__(self):
        self.head = None

    def is_empty(self):
        return self.head is None

    def push(self, data):
        new_node = Node(data)
        if self.is_empty():
            self.head = new_node
        else:
            new_node.next = self.head
            self.head = new_node

    def pop(self):
        if self.is_empty():
            return None
        else:
            popped_node = self.head
            self.head = self.head.next
```

```
            popped_node.next = None
            return popped_node.data

    def peek(self):
        if self.is_empty():
            return None
        else:
            return self.head.data

    def display(self):
        current = self.head
        stack_values = []
        while current:
            stack_values.append(str(current.data))
            current = current.next
        print("Stack:")
        for value in stack_values:
            print("|", value, "|")
        print("-------")
```

Ejercicio N° 3

Analizar el código del programa `arboles_tp.py`. Identificar la clase, los métodos. Documentar el programa. Qué métodos se deberían agregar y qué mejoras se podrían proponer. Sustentarlas.

> Este queda para que cada uno lo haga