2 . List Operations Theory:

❖Common list operations: concatenation, repetition, membership.

1. Concatenation (+) :

What it does : Joins two lists end-to-end, returning a new list.

Syntax :

```
list_a + list_b
```

2. Repetition (*) :

- **What it does :** Creates a new list by repeating the contents of the original list a given number of times.

- **Syntax** :

```
my_list * n
```

3. Membership Testing (in, not in) :

- **What it does** : Checks whether an element exists (or doesn't) in a list.

- **Syntax** :

    x in my_list

    x not in my_list

❖Understanding list methods like append(), insert(), remove(), pop().

append(x) :

- **What it does:** Adds the element x to the end of the list—like stacking something on top.
- **Time complexity:** Fast—O(1).
- **Returns:** None (in-place modification).

insert(i, x) :

- **What it does**: Inserts x before index i.
  a.insert(0, x) → front; a.insert(len(a), x) = append(x).
- **Time complexity**: O(n) – elements must shift to make space.

remove(x) :

- **What it does**: Removes the **first** element whose value equals x.
- **Raises**: ValueError if x is not found.
- **Time complexity**: O(n) – searches list for value.

pop(i=None) :

- **What it does**:
  - Without argument: removes and returns **last element**.
  - With index i: removes and returns element at that index.
- **Raises**: IndexError if list is empty or i is out of range.
- **Time complexity**: O(1) when popping last; O(n–i) otherwise.