2. List Operations

1. Common list operations: concatenation, repetition, membership

* Concatenation :
  + joining two lists end‑to‑end to form a **new list**
  + both operands **must be lists**; mixing types (e.g. list + int) raises a typeerror .
  + example: [1,2] + [3,4] → [1,2,3,4].
  + this does **not** nest sublists; it simply merges the elements
  + for in‑place concatenation, use a.extend(b) or augmented assignment a += b
* repetition :
  + replicates the **entire list's items** a given number of times, producing a **new list**
  + example: [4,5] \* 3 → [4,5,4,5,4,5].
  + works only with lists and integers; [item] \* n creates a list with item repeated n times.
  + caution: for lists of mutable objects, repetition duplicates references—not deep copies—so modifying one nested element affects all repetitions
* membership :
* checks whether a **value appears** in the list.
* x in lst returns True if any element equals x; otherwise False.
* x not in lst is the inverse.
* example: 'b' in ['a','b','c'] → True; 'z' not in ['a','b','c'] → True
* typically implemented with a **linear search**, so complexity is O(n).

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

1. **append()**

* adds element x to the **end** of the list in-place.
* **amortized time complexity** is O(1), meaning most calls are constant time on average
* space-wise, occasional resizing happens but is efficient
  1. **insert()**
* inserts element x at position i, shifting later elements to the right.
* time complexity is **O(n)** due to shifting elements
* useful for ordered insertion, but can be slow on large lists.
  1. **remove()**
* removes the **first occurrence** of value x in the list.
* runs a linear search to find x, then shifts following elements.
* time complexity is **O(n)** in the worst case
  1. **pop()**
* removes and **returns** the element at index i.without i, it removes the **last item**.
* pop() (no index) is **O(1)** — constant time removal from end.
* pop(i) for arbitrary i incurs **O(n)** due to element shifting