**ASSIGNMENT2:LIST OPERATIONS**

**Common list operations: concatenation, repetition, membership.**

* Python lists support several operations like concatenation, repetition, and membership testing. These operations make list manipulation simple and efficient.

**1. List Concatenation (+):**

Combines two or more lists into one.

**Example:**

python

Copy code

list1 = [1, 2, 3]

list2 = [4, 5, 6]

result = list1 + list2

print(result) # Output: [1, 2, 3, 4, 5, 6]

* **Note:** Concatenation does **not** modify the original lists; it creates a new list.

**2. List Repetition (\*):**

Repeats the elements of a list a specified number of times.

**Example:**

my\_list = [1, 2, 3]

repeated = my\_list \* 3

print(repeated) # Output: [1, 2, 3, 1, 2, 3, 1, 2, 3]

* Useful for initializing lists with default values.

**3. Membership Testing (in / not in):**

Checks if an element exists in the list.

**Example:**

colors = ["red", "green", "blue"]

print("green" in colors) # Output: True

print("yellow" in colors) # Output: False

print("yellow" not in colors) # Output: True

* Returns True if the element is found, otherwise False.
* not in returns the opposite.

**Practical Example:**

names1 = ["Alice", "Bob"]

names2 = ["Charlie", "David"]

# Concatenation

all\_names = names1 + names2

print("Concatenated List:", all\_names)

# Repetition

repeated\_names = names1 \* 2

print("Repeated List:", repeated\_names)

# Membership

print("Is 'Alice' in the list?", "Alice" in all\_names)

print("Is 'Eve' in the list?", "Eve" not in all\_names)

**Output:**

Concatenated List: ['Alice', 'Bob', 'Charlie', 'David']

Repeated List: ['Alice', 'Bob', 'Alice', 'Bob']

Is 'Alice' in the list? True

Is 'Eve' in the list? True

**Key Points:**

* **Concatenation (+)** joins lists without modifying the originals.
* **Repetition (\*)** repeats elements, great for initialization.
* **Membership (in)** is quick for element existence checks.

.

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

In Python, lists have several useful methods that allow you to manipulate the data they contain. Here's an overview of the methods you mentioned:

1. **append()**:
   * Adds an item to the end of the list.
   * Syntax: list.append(item)
   * Example:

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list) # Output: [1, 2, 3, 4]

1. **insert()**:
   * Inserts an item at a specific index in the list.
   * Syntax: list.insert(index, item)
   * Example:

my\_list = [1, 3, 4]

my\_list.insert(1, 2) # Insert 2 at index 1

print(my\_list) # Output: [1, 2, 3, 4]

1. **remove()**:
   * Removes the first occurrence of a specified item from the list.
   * Syntax: list.remove(item)
   * If the item is not found, it raises a ValueError.
   * Example:

my\_list = [1, 2, 3, 2]

my\_list.remove(2) # Removes the first occurrence of 2

print(my\_list) # Output: [1, 3, 2]

1. **pop()**:
   * Removes and returns the item at a specified index (default is the last item).
   * Syntax: list.pop(index)
   * Example:

my\_list = [1, 2, 3]

popped\_item = my\_list.pop() # Removes and returns the last item

print(popped\_item) # Output: 3

print(my\_list) # Output: [1, 2]