List Methods:

Python provides many built-in **methods** to work with lists.

1. append(item) – adds item at the end.

```
fruits = ["apple", "banana"]
fruits.append("mango")
print(fruits)   # ['apple', 'banana', 'mango']
```

2. insert(index, item) – adds item at specific index.

```
fruits.insert(1, "orange")
print(fruits)   # ['apple', 'orange', 'banana', 'mango']
```

3. remove(item) – removes first occurrence of value.

```
fruits.remove("banana")
print(fruits)   # ['apple', 'orange', 'mango']
```

4. pop(index) – removes by index (default last element).

```
fruits.pop()       # removes last element
fruits.pop(0)      # removes first element
```

5. sort() – sorts list in ascending order.

```
numbers = [3, 1, 4, 2]
numbers.sort()
print(numbers)   # [1, 2, 3, 4]
```

6. reverse() – reverses the order of list.

```
numbers = [1, 2, 3, 4]
numbers.reverse()
print(numbers)   # [4, 3, 2, 1]
```

7. copy() – makes a copy of the list.

```
fruits = ["apple", "banana"]
new_fruits = fruits.copy()
print(new_fruits)   # ['apple', 'banana']
```

8. count(item) – counts how many times a value appears.

```
numbers = [1, 2, 2, 3, 2, 4]
print(numbers.count(2))   # 3
```

9. index(item) – returns position of first occurrence.

```
fruits = ["apple", "banana", "mango"]
print(fruits.index("mango"))   # 2
```

Looping through Lists:

Method 1: Directly through items

```
fruits = ["apple", "banana", "mango"]
for fruit in fruits:
   print(fruit)
```

Method 2: Using index with range()

```
fruits = ["apple", "banana", "mango"]
   for i in range(len(fruits)):
   print(fruits[i])
```

Method 3: Using while loop

```
fruits = ["apple", "banana", "mango"]
   i = 0
   while i < len(fruits):
   print(fruits[i])
   i += 1
```

3. Importance of List Methods & Looping:

- Makes list handling **faster and easier**.
- Looping is required when we want to **process each element one by one** (like printing all student names, adding marks, searching, etc.).
- These concepts are foundation for **loops, data processing, and projects.**