



# 1. LIST METHODS (Very Important in AI/ML + Python)

Lists are **mutable (changeable)**, so they have many methods.

Below are the **top 15–20** that are **practically useful**.

Method	Description	Example
<code>append()</code>	Adds an element at the end	<code>lst.append(5)</code>
<code>extend()</code>	Adds multiple elements from another list or iterable	<code>lst.extend([6, 7])</code>
<code>insert(index, value)</code>	Inserts an element at a specific position	<code>lst.insert(1, "apple")</code>
<code>remove(value)</code>	Removes first matching element	<code>lst.remove("apple")</code>
<code>pop([index])</code>	Removes and returns element (default last)	<code>lst.pop()</code> / <code>lst.pop(0)</code>
<code>clear()</code>	Removes all elements	<code>lst.clear()</code>
<code>index(value)</code>	Returns index of first match	<code>lst.index(10)</code>
<code>count(value)</code>	Counts occurrences of a value	<code>lst.count(2)</code>
<code>sort(reverse=False, key=None)</code>	Sorts list (in-place)	<code>lst.sort()</code>
<code>sorted(lst)</code> ( <i>function</i> )	Returns new sorted list (doesn't change original)	<code>sorted(lst)</code>
<code>reverse()</code>	Reverses list in place	<code>lst.reverse()</code>
<code>copy()</code>	Returns a shallow copy	<code>new = lst.copy()</code>
<code>sum(lst)</code> ( <i>built-in</i> )	Adds all numeric elements	<code>sum(lst)</code>
<code>len(lst)</code> ( <i>built-in</i> )	Returns number of elements	<code>len(lst)</code>
<code>max(lst) / min(lst)</code>	Returns largest/smallest element	<code>max(lst)</code>

<code>any(lst) / all(lst)</code>	Returns True if any/all elements are True	<code>any([0,1,0]) → True</code>
<code>list comprehension</code>	Creates list in one line	<code>[x**2 for x in lst]</code>

---

## AI/ML-Specific Common List Uses

- Handling **datasets** or **feature lists** before converting to **NumPy arrays**.
- **Appending** rows/columns of data.
- **Sorting** results, predictions, or probabilities.
- **Counting** label occurrences in classification tasks.

Example:

```
data = [1, 2, 3, 4, 5]
squared = [x**2 for x in data]
print(squared) # [1, 4, 9, 16, 25]
```

•

---

## ◆ 2. TUPLE METHODS (Simpler but Important)

Tuples are **immutable**, so they have **fewer methods**, but are **very useful** in AI/ML — e.g., returning multiple values (like `(x, y)` coordinates, shapes, etc.)

Method	Description	Example
<code>count(value)</code>	Counts how many times a value occurs	<code>(1,2,2,3).count(2) → 2</code>
<code>index(value)</code>	Returns first index of a value	<code>(10,20,30).index(20) → 1</code>

---

## AI/ML-Specific Tuple Uses

Used to represent **coordinates**, **shapes**, or **fixed data**:

```
shape = (28, 28) # image dimension
coord = (x, y)   # data point
```

- 

Used for **returning multiple outputs**:

```
def get_stats(data):
    return (min(data), max(data), sum(data)/len(data))
print(get_stats([2,4,6,8]))
```

- 

---

## BONUS: Built-in Functions Often Used with Lists/Tuples

Function	Description	Example
<code>zip()</code>	Combine multiple lists into tuples	<code>list(zip(a,b))</code>
<code>map()</code>	Apply function to all items	<code>list(map(str.upper, words))</code>
<code>filter()</code>	Filter elements using a condition	<code>list(filter(lambda x:x&gt;0, nums))</code>
<code>enumerate()</code>	Get index + value pairs	<code>for i, val in enumerate(lst)</code>
<code>list()</code> / <code>tuple()</code>	Convert between list and tuple	<code>tuple(lst) / list tpl</code>

---

## Example (AI/ML Style)

```
# Example: Working with dataset values
data = [12, 45, 23, 56, 34, 90, 12]
```

```
# Clean duplicates and sort
unique = list(set(data))
unique.sort()

# Normalize (using list comprehension)
normalized = [x / max(unique) for x in unique]

print("Unique:", unique)
print("Normalized:", normalized)
```

---

## ✓ Summary: Must-Memorize Ones

**List:** `append`, `extend`, `insert`, `remove`, `pop`, `sort`, `reverse`, `copy`, `count`, `index`

**Tuple:** `count`, `index`

**Also Useful:** `zip`, `map`, `filter`, `enumerate`, list comprehension

---

Would you like me to give you a **short Python practice sheet** (with exercises and answers) for List & Tuple methods (AI + normal coding focus)?

It'll help you memorize and apply these easily.

**You said:**

i want to every method with short and simple example

**ChatGPT said:**

Perfect 👍 Deepanshu!

Here's a **complete list of all important `list` and `tuple` methods in Python** with **short, simple examples** — very easy to remember and useful in **AI/ML + normal coding**.

---

## ✂ LIST METHODS

✓ Lists are **mutable** (can be changed).

### 1. `append()`

Add one item at the end.

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

---

## 2. **extend()**

Add multiple items from another list or iterable.

```
nums = [1, 2]
nums.extend([3, 4])
print(nums)  # [1, 2, 3, 4]
```

---

## 3. **insert(index, value)**

Insert an item at a specific position.

```
colors = ["red", "blue"]
colors.insert(1, "green")
print(colors)  # ['red', 'green', 'blue']
```

---

## 4. **remove(value)**

Remove the first matching item.

```
nums = [1, 2, 3, 2]
nums.remove(2)
print(nums)  # [1, 3, 2]
```

---

## 5. **pop([index])**

Remove and return an item (default last).

```
nums = [10, 20, 30]
```

```
nums.pop()      # removes 30
nums.pop(0)     # removes 10
print(nums)     # [20]
```

---

## 6. `clear()`

Remove all items.

```
data = [1, 2, 3]
data.clear()
print(data)    # []
```

---

## 7. `index(value)`

Return index of first match.

```
nums = [5, 10, 15, 10]
print(nums.index(10))  # 1
```

---

## 8. `count(value)`

Count how many times a value appears.

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

---

## 9. `sort()`

Sort list in ascending order (in-place).

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

**Descending order:**

```
nums.sort(reverse=True)
print(nums)  # [3, 2, 1]
```

---

## 10. **reverse()**

Reverse the list.

```
letters = ['a', 'b', 'c']
letters.reverse()
print(letters)  # ['c', 'b', 'a']
```

---

## 11. **copy()**

Return a shallow copy of the list.

```
a = [1, 2, 3]
b = a.copy()
print(b)  # [1, 2, 3]
```

---

## 12. **len()**

Get the number of items.

```
nums = [10, 20, 30]
print(len(nums))  # 3
```

---

## 13. **max() / min()**

Get largest or smallest value.

```
nums = [5, 9, 2]
print(max(nums))  # 9
print(min(nums))  # 2
```

---

## 14. `sum()`

Get total sum of numeric list.

```
nums = [1, 2, 3]
print(sum(nums)) # 6
```

---

## 15. `any()` / `all()`

Check True/False conditions.

```
data = [0, 1, 0]
print(any(data)) # True (at least one True)
print(all(data)) # False (not all True)
```

---

## 16. List Comprehension (shortcut to create lists)

```
nums = [1, 2, 3, 4]
squared = [x**2 for x in nums]
print(squared) # [1, 4, 9, 16]
```

---

# TUPLE METHODS

✅ Tuples are **immutable** (cannot be changed).

## 1. `count(value)`

Count how many times a value appears.

```
t = (1, 2, 2, 3)
print(t.count(2)) # 2
```

---



## 2. `index(value)`

Return index of first occurrence.

```
t = (10, 20, 30)
print(t.index(20)) # 1
```

---

# Built-in Functions Useful for Lists & Tuples

## 1. `list()` / `tuple()`

Convert between list and tuple.

```
t = (1, 2, 3)
print(list(t)) # [1, 2, 3]
l = [4, 5, 6]
print(tuple(l)) # (4, 5, 6)
```

---

## 2. `zip()`

Combine multiple lists element-wise.

```
a = [1, 2, 3]
b = ['x', 'y', 'z']
print(list(zip(a, b))) # [(1, 'x'), (2, 'y'), (3, 'z')]
```

---

## 3. `map()`

Apply a function to all items.

```
nums = [1, 2, 3]
print(list(map(lambda x: x*2, nums))) # [2, 4, 6]
```

---

#### 4. `filter()`

Filter items using a condition.

```
nums = [1, 2, 3, 4, 5]
print(list(filter(lambda x: x > 2, nums))) # [3, 4, 5]
```

---

#### 5. `enumerate()`

Get index and value while looping.

```
fruits = ["apple", "mango", "banana"]
for i, fruit in enumerate(fruits):
    print(i, fruit)
# 0 apple
# 1 mango
# 2 banana
```

---

## Summary Table

Type	Important Methods
<b>List</b>	append, extend, insert, remove, pop, clear, index, count, sort, reverse, copy
<b>Tuple</b>	count, index
<b>Built-in</b>	len, max, min, sum, any, all, zip, map, filter, enumerate