Hello & Welcome

List Methods

Table of contents

01

Adding Elements to a List

Adding

01 — `append()` — Adds **an element** to the end of the list.

02 — `extend()` — Adds **multiple elements** from an iterable

03 — `insert()` — Inserts an element at a **specific index**.

Adding – append()

- **Adds** an element to the **end** of the list.
- **Syntax**: list.append(element)
- **Returns**: None (modifies the list in-place)

Python	Output

`my_list = [1, 2, 3]`
`my_list.append(4)`
`print(my_list)`
	`[1, 2, 3, 4]`

Adding – extend()

- Adds **multiple elements** from an iterable (list, tuple, set, etc.) to the end of the list.
- **Syntax**: list.extend(iterable)
- **Returns**: None (modifies the list in-place)

Python	Output

`my_list = [1, 2, 3]`
`my_list.extend([4, 5, 6])`
`print(my_list)`
	`[1, 2, 3, 4, 5, 6]`

Difference Between append() and extend()

append()	extend()
Adds the entire object as a single element	Unpacks the iterable and adds its elements individually

`a = [1, 2, 3]`
`a.append([4, 5])`
`print(a)`
	`b = [1, 2, 3]`
	`b.extend([4, 5])`
	`print(b)`

`Output: [1, 2, 3, [4, 5]]`	`Output: [1, 2, 3, 4, 5]`

Adding - insert()

- Inserts an element at a **specific index**.
- **Syntax**: list.insert(index, element)
- **Returns**: None (modifies the list in-place)

Python	Output

`my_list = [1, 2, 4]`
`my_list.insert(2, 3)`
`print(my_list)`
	`[1, 2, 3, 4]`

02

Removing Elements from a List

Removing

01 — `remove()` — Removes the **first occurrence** of the specified value.

02 — `pop()` — **Removes** and **returns** the element at the given **index**

03 — `clear()` — **Removes** all elements from the list, making it **empty**

Removing - remove()

- Removes the **first occurrence** of the specified value.
- **Syntax**: list.remove(element)
- **Raises**: ValueError if the element is not found.
- **Returns**: None (modifies the list in-place)

Python	Output

`my_list = [1, 2, 3, 2, 4]`
`my_list.remove(2)`
`print(my_list)`
	`[1, 3, 2, 4]`

Removing – pop()

- **Removes** and **returns** the element at the given index.
- If index is not provided, removes and returns the **last element**.
- **Syntax**: list.pop(index)
- **Raises**: IndexError if the index is out of range.

Python	Output

`my_list = [10, 20, 30, 40]`
`print(my_list.pop(2))`
`print(my_list)`
	`30`
	`[10, 20, 40]`

Removing – pop()

- **Removes** and **returns** the element at the given index.
- If index is not provided, removes and returns the **last element**.
- **Syntax**: list.pop(index)
- **Raises**: IndexError if the index is out of range.

Python	Output

`my_list = [1, 2, 3]`
`print(my_list.pop())`
`print(my_list)`
	`3`
	`[1, 2]`

Removing – clear()

- **Removes all elements** from the list, making it empty.
- **Syntax**: list.clear()
- **Returns**: None

Python	Output

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

03

Searching & Counting Elements

Searching & Counting

01 — `index()` — Returns the **index** of the **first occurrence** of a value.

02 — `count()` — Returns the **number of times** a value appears in the list.

Searching & Counting – index()

- **Returns the index** of the **first occurrence** of a value.
- **Syntax**: list.index(element, start, end)
- **Raises**: ValueError if the element is not found.

Python	Output

`my_list = [10, 20, 30, 40, 20]`
`print(my_list.index(20))`
`print(my_list.index(20, 2))`
	`1`
	`4`

Searching & Counting – count()

- Returns the **number of times** a value appears in the list.
- **Syntax**: list. Count(element)

Python	Output
`my_list = [1, 2, 2, 3, 2, 4]` `print(my_list.count(2))`	`3`

04

Sorting & Reversing

Sorting & Reversing

01 — `sort()` — **Sorts** the list in **ascending order** (by default)

02 — `reverse()` — **Reverses the order** of elements in the list.

Sorting & Reversing – sort()

- Sorts the list in **ascending order** (by default)
- **Syntax**: list.sort(*reverse*=False, *key*=None)
- **Returns**: None (modifies the list in-place)

Python	Output

`my_list = [4, 2, 9, 1]`
`my_list.sort()`
`print(my_list)`
	`[1, 2, 4, 9]`

Sorting & Reversing – sort()

- Sorts the list in **ascending order** (by default)
- **Syntax**: list.sort(*reverse*=False, *key*=None)
- **Returns**: None (modifies the list in-place)

Python	Output
`my_list.sort(reverse=True)` `print(my_list)`	`[9, 4, 2, 1]`

Sorting & Reversing – sort()

- Sorts the list in **ascending order** (by default)
- **Syntax**: list.sort(*reverse*=False, *key*=None)
- **Returns**: None (modifies the list in-place)

Python	Output

`words = ["apple", "banana", "cherry"]`
`words.sort(key=len)`
`print(words)`
	`['apple', 'cherry', 'banana']`

Sorting & Reversing – reverse()

- **Reverses** the **order** of elements in the list.
- **Syntax**: list.reverse()
- **Returns**: None (modifies the list in-place)

Python	Output

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

05

Copying Lists

Copying Lists

01 — `copy()` — Creates a **shallow copy** of the list

Copying Lists – copy()

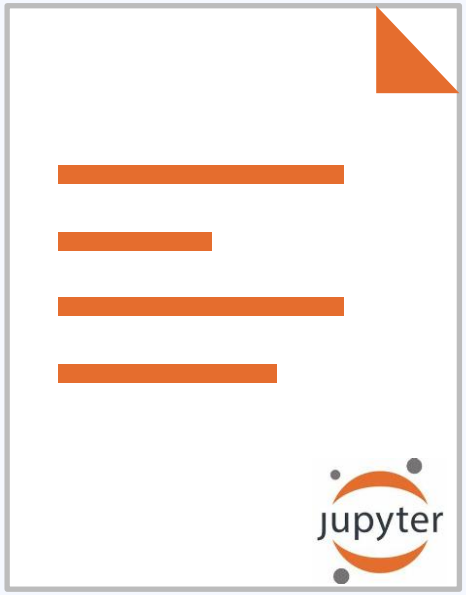
- Creates a **shallow copy** of the list.
- **Syntax**: list.copy()

Python	Output

`original = [1, 2, 3]`
`copy_list = original.copy()`
`print(copy_list)`
	`[1, 2, 3]`

Summary Table

Category	Methods
Adding Element	append() extend() insert()
Removing Element	remove() pop() clear()
Searching & Counting	index() count()
Sorting & Reversing	sort() reverse()
Copying	copy()

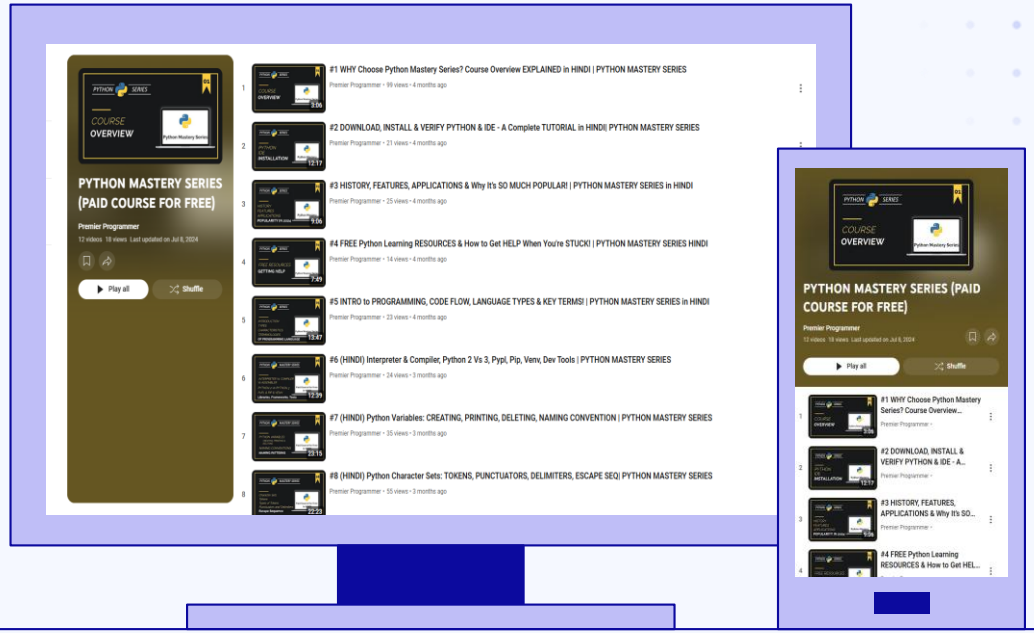


Practice Set 4

Download Link in **Description**

WATCH

Level up your coding with each episode in this focused Python series.



Next Video!

Practice Set 4 Solution