Complete Python List Methods Reference Guide

This document provides a comprehensive overview of all Python list methods with detailed explanations and three practical examples for each method.

Table of Contents

Introduction to Python Lists
Mutating Methods
Non-Mutating Methods
Additional List-Related Methods
Advanced List Operations
Quick Reference Summary

Introduction to Python Lists

What are Lists?

Lists are one of the most fundamental and versatile data structures in Python. They are ordered, mutable collections that can store multiple items of any data type. Lists are defined using square brackets `[]` and items are separated by commas.

Key Characteristics of Lists

- **Ordered**: Items maintain their position and can be accessed by index
- **Mutable**: Can be modified after creation (items can be added, removed, or changed)
- **Dynamic**: Can grow or shrink in size during runtime
- **Heterogeneous**: Can contain different data types in the same list
- **Allow Duplicates**: Same values can appear multiple times

Creating Lists

Example 1: Empty list

```
empty_list = []
# or
empty_list = list()
```

Example 2: List with initial values

```
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "orange"]
mixed = [1, "hello", 3.14, True, [4, 5, 6]]
```

Example 3: Creating lists from other iterables

```
# from string
letters = list("hello")  # ['h', 'e', 'l', 'l', 'o']

# from range
numbers = list(range(5))  # [0, 1, 2, 3, 4]

# from tuple
tuple_data = (1, 2, 3)
list_data = list(tuple_data)  # [1, 2, 3]
```

Mutating Methods

These methods modify the original list in place.

1. append(item)

Adds a single item to the end of the list.

Example 1: Adding a number

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

Example 2: Adding a string

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

Example 3: Adding a list as a single element

```
main_list = [1, 2, 3]
main_list.append([4, 5])
print(main_list)  # [1, 2, 3, [4, 5]]
```

2. extend(iterable)

Adds all items from an iterable to the end of the list.

Example 1: Extending with another list

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list1.extend(list2)
print(list1)  # [1, 2, 3, 4, 5, 6]
```

Example 2: Extending with a string

```
letters = ["a", "b"]
letters.extend("cd")
print(letters)  # ['a', 'b', 'c', 'd']
```

Example 3: Extending with a tuple

```
numbers = [1, 2]
numbers.extend((3, 4, 5))
print(numbers)  # [1, 2, 3, 4, 5]
```

3. insert(index, item)

Inserts an item at a specific position in the list.

Example 1: Inserting at the beginning

```
colors = ["red", "blue"]
colors.insert(0, "green")
print(colors)  # ['green', 'red', 'blue']
```

Example 2: Inserting in the middle

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

Example 3: Inserting at the end (same as append)

```
animals = ["cat", "dog"]
animals.insert(len(animals), "bird")
print(animals)  # ['cat', 'dog', 'bird']
```

4. remove(item)

Removes the first occurrence of an item from the list.

Example 1: Removing a number

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

Example 2: Removing a string

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

Example 3: Handling ValueError when item doesn't exist

```
colors = ["red", "blue", "green"]
try:
    colors.remove("yellow")
except ValueError:
    print("Item not found")
print(colors)  # ['red', 'blue', 'green']
```

5. pop(index=-1)

Removes and returns an item at a specific index (default is last item).

Example 1: Popping the last item

```
numbers = [1, 2, 3, 4]
last_item = numbers.pop()
print(f"Popped: {last_item}")  # Popped: 4
print(numbers)  # [1, 2, 3]
```

Example 2: Popping from a specific index

```
fruits = ["apple", "banana", "orange"]
second_fruit = fruits.pop(1)
print(f"Popped: {second_fruit}")  # Popped: banana
print(fruits)  # ['apple', 'orange']
```

Example 3: Popping from the beginning

```
queue = ["first", "second", "third"]
first = queue.pop(0)
print(f"Processed: {first_item}")  # Processed: first
print(queue)  # ['second', 'third']
```

6. clear()

Removes all items from the list.

Example 1: Clearing a number list

```
numbers = [1, 2, 3, 4, 5]
numbers.clear()
print(numbers)  # []
```

Example 2: Clearing a string list

```
names = ["Alice", "Bob", "Charlie"]
names.clear()
print(f"Length after clear: {len(names)}")  # Length after clear: 0
```

Example 3: Clearing and repopulating

```
shopping_cart = ["item1", "item2", "item3"]
shopping_cart.clear()
shopping_cart.extend(["new_item1", "new_item2"])
print(shopping_cart)  # ['new_item1', 'new_item2']
```

7. reverse()

Reverses the order of items in the list in place.

Example 1: Reversing numbers

```
numbers = [1, 2, 3, 4, 5]
numbers.reverse()
print(numbers)  # [5, 4, 3, 2, 1]
```

Example 2: Reversing strings

```
words = ["hello", "world", "python"]
words.reverse()
print(words)  # ['python', 'world', 'hello']
```

Example 3: Reversing mixed data types

```
mixed = [1, "a", 3.14, True]
mixed.reverse()
print(mixed)  # [True, 3.14, 'a', 1]
```

8. sort(key=None, reverse=False)

Sorts the list in place.

Example 1: Sorting numbers in ascending order

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6]
numbers.sort()
print(numbers)  # [1, 1, 2, 3, 4, 5, 6, 9]
```

Example 2: Sorting strings in descending order

```
names = ["Charlie", "Alice", "Bob"]
names.sort(reverse=True)
print(names)  # ['Charlie', 'Bob', 'Alice']
```

Example 3: Sorting with a custom key function

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

Non-Mutating Methods

These methods return information about the list without modifying it.

9. count(item)

Returns the number of times an item appears in the list.

Example 1: Counting numbers

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

Example 2: Counting strings

```
fruits = ["apple", "banana", "apple", "orange", "apple"]
apple_count = fruits.count("apple")
print(apple_count)  # 3
```

Example 3: Counting non-existent items

```
colors = ["red", "blue", "green"]
yellow_count = colors.count("yellow")
print(yellow_count)  # 0
```

10. index(item, start=0, stop=len(list))

Returns the index of the first occurrence of an item.

Example 1: Finding index of a number

```
numbers = [10, 20, 30, 40, 50]
index_of_30 = numbers.index(30)
print(index_of_30)  # 2
```

Example 2: Finding index with start parameter

```
letters = ["a", "b", "c", "d", "b", "e"]
first_b = letters.index("b")  # Find first 'b'
print(first_b)  # 1

second_b = letters.index("b", 2)  # Find second 'b' starting from index 2
print(second_b)  # 4
```

Example 3: Handling ValueError when item doesn't exist

```
fruits = ["apple", "banana", "orange"]
try:
    pear_index = fruits.index("pear")
except ValueError:
    print("Item not found in list")
```

11. copy()

Returns a shallow copy of the list.

Example 1: Creating a copy of numbers

```
original = [1, 2, 3, 4]
copied = original.copy()
copied.append(5)
print(f"Original: {original}")  # Original: [1, 2, 3, 4]
print(f"Copied: {copied}")  # Copied: [1, 2, 3, 4, 5]
```

Example 2: Shallow copy with nested lists

```
original = [[1, 2], [3, 4]]
copied = original.copy()
copied[0].append(5)
print(f"Original: {original}")  # Original: [[1, 2, 5], [3, 4]]
print(f"Copied: {copied}")  # Copied: [[1, 2, 5], [3, 4]]
```

Example 3: Copy and modify independently

```
names = ["Alice", "Bob", "Charlie"]
backup = names.copy()
names.remove("Bob")
print(f"Names: {names}")  # Names: ['Alice', 'Charlie']
print(f"Backup: {backup}")  # Backup: ['Alice', 'Bob', 'Charlie']
```

Additional List-Related Methods

12. Slicing Operations

Slicing allows you to access a subset of the list.

Example 1: Basic slicing

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
first_three = numbers[:3]  # [0, 1, 2]
middle = numbers[3:7]  # [3, 4, 5, 6]
last_three = numbers[-3:]  # [7, 8, 9]
```

Example 2: Slicing with step

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
evens = numbers[::2]  # [0, 2, 4, 6, 8]
odds = numbers[1::2]  # [1, 3, 5, 7, 9]
reversed = numbers[::-1]  # [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

Example 3: Slice assignment

```
numbers = [0, 1, 2, 3, 4, 5]
numbers[1:3] = [10, 20]
print(numbers)  # [0, 10, 20, 3, 4, 5]
```

13. Length and Membership

Example 1: Length

```
numbers = [1, 2, 3, 4, 5]
length = len(numbers)
print(f"Length: {length}")  # Length: 5
```

Example 2: Membership testing

```
fruits = ["apple", "banana", "orange"]
has_apple = "apple" in fruits
has_grape = "grape" in fruits
print(f"Has apple: {has_apple}")  # Has apple: True
print(f"Has grape: {has_grape}")  # Has grape: False
```

Example 3: Conditional operations

```
numbers = [1, 2, 3, 4, 5]
if len(numbers) > 3:
    print("List has more than 3 items")
if 1 in numbers and 5 in numbers:
    print("List contains both 1 and 5")
```

Advanced List Operations

14. Nested Lists

Example 1: Creating nested lists

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matrix[0][1])  # 2
print(matrix[1][2])  # 6
```

Example 2: Modifying nested lists

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix[0].append(4)
matrix[1][0] = 40
print(matrix)  # [[1, 2, 3, 4], [40, 5, 6], [7, 8, 9]]
```

Example 3: Iterating through nested lists

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
for row in matrix:
    for item in row:
        print(f"{item}", end=" ")
    print()
# Output:
# 1 2 3
# 4 5 6
# 7 8 9
```

15. Common Patterns

Example 1: Stack operations (LIFO)

```
stack = []
stack.append("first")
stack.append("second")
stack.append("third")
print(f"Stack: {stack}")  # Stack: ['first', 'second', 'third']
top = stack.pop()
print(f"Popped: {top}")  # Popped: third
print(f"Stack: {stack}")  # Stack: ['first', 'second']
```

Example 2: Queue operations (FIFO)

```
queue = []
queue.append("first")
queue.append("second")
queue.append("third")
print(f"Queue: {queue}")  # Queue: ['first', 'second', 'third']
first = queue.pop(0)
print(f"Processed: {first}")  # Processed: first
print(f"Queue: {queue}")  # Queue: ['second', 'third']
```

Example 3: Filtering and transformation

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
evens = []
for num in numbers:
    if num % 2 == 0:
        evens.append(num)
print(f"Evens: {evens}")  # Evens: [2, 4, 6, 8, 10]

squares = [num ** 2 for num in numbers]
print(f"Squares: {squares}")  # Squares: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Quick Reference Summary

Method	Purpose	Modifies List	Returns
`append(item)`	Add item to end	Yes	None
`extend(iterable)`	Add items from iterable	Yes	None
`insert(index, item)`	Insert at position	Yes	None
`remove(item)`	Remove first occurrence	Yes	None
`pop(index=-1)`	Remove and return item	Yes	Item
`clear()`	Remove all items	Yes	None
`reverse()`	Reverse order	Yes	None
`sort(key=None, reverse=False)`	Sort in place	Yes	None
`count(item)`	Count occurrences	No	Count
`index(item, start=0, stop=len(list))`	Find first occurrence	No	Index
`copy()`	Shallow copy	No	New list
`len(list)`	Length	No	Length
`item in list`	Membership test	No	Boolean
`list[index]`	Access item	No	Item
`list[start:stop:step]`	Slicing	No	New list