List Methods in Python

1. **append()**

The `append()` method adds an element to the end of the list.

Syntax:

`list.append(element)`

- `element`: The item to be added to the list.

Example:

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

2. **extend()**

The `extend()` method extends the list by appending all the elements from the iterable.

Syntax:

`list.extend(iterable)`

- `iterable`: A collection of items (such as a list) to be added to the list.

Example:

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

3. **insert()**

The `insert()` method inserts an element at a specified position.

Syntax:

`list.insert(index, element)`

- `index`: The position to insert the element.
- `element`: The element to insert.

Example:

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

4. **remove()**

The `remove()` method removes the first occurrence of the specified value.

Syntax:

`list.remove(value)`

- `value`: The value to be removed from the list.

Example:

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

5. **pop()**

The `pop()` method removes the element at the specified position and returns it. If no index is specified, it removes and returns the last item.

Syntax:

`list.pop(index)`

- `index`: The position to remove the element from. If not specified, removes the last item.

Example:

```
my_list = [1, 2, 3, 4]
popped_item1 = my_list.pop()
popped_item2 = my_list.pop(1)
print(popped_item1)  # Output: 4
print(popped_item2)  # Output: 2
print(my_list)      # Output: [1, 3]
```

6. **clear()**

The `clear()` method removes all elements from the list.

Syntax:

`list.clear()`

Example:

```
my_list = [1, 2, 3, 4]
my_list.clear()
print(my_list)  # Output: []
```

7. **index()**

The `index()` method returns the index of the first occurrence of the specified value. Raises a `ValueError` if the value is not found.

Syntax:

`list.index(value)`

- `value`: The value to search for.

Example:

```
my_list = [1, 2, 3, 4]
index = my_list.index(3)
print(index)  # Output: 2
```

8. **count()**

The `count()` method returns the number of times the specified value appears in the list.

Syntax:

`list.count(value)`

- `value`: The value to count.

Example:

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

9. **sort()**

The `sort()` method sorts the list in ascending order. It can take optional arguments for reverse sorting or a custom sort function.

Syntax:

`list.sort(reverse=False, key=None)`

- `reverse`: If True, sorts the list in descending order.
- `key`: A function to specify sorting criteria.

Example:

```
my_list = [3, 1, 4, 2]
my_list.sort()
print(my_list)  # Output: [1, 2, 3, 4]

# Sorting in reverse order
my_list.sort(reverse=True)
print(my_list)  # Output: [4, 3, 2, 1]
```

10. **reverse()**

The `reverse()` method reverses the elements of the list in place.

Syntax:

`list.reverse()`

Example:

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

11. **copy()**

The `copy()` method returns a shallow copy of the list.

Syntax:

`list.copy()`

Example:

```
my_list = [1, 2, 3, 4]
copied_list = my_list.copy()
print(copied_list)  # Output: [1, 2, 3, 4]
```