Lists and its methods

Python Lists and Methods:

# 1. **`append()` method:**
   - Adds an element to the end of the list.

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

# 2. **`extend()` method:**
   - Appends the elements of another iterable (list, tuple, string) to the end of the list.

   ```python
   numbers = [1, 2, 3]
   numbers.extend([4, 5, 6])
   print(numbers)  # Output: [1, 2, 3, 4, 5, 6]
   ```

# 3. **`insert()` method:**
   - Inserts an element at a specific index.

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

# 4. **`remove()` method:**
   - Removes the first occurrence of a specified element.

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

# 5. **`pop()` method:**
   - Removes and returns the element at the specified index. If no index is provided, removes and returns the last element.

   ```python
   numbers = [1, 2, 3]
   popped_element = numbers.pop(1)
   print(popped_element)  # Output: 2
   ```

   print(numbers)   # Output: [1, 3]

# 6. **`index()` method:**
   - Returns the index of the first occurrence of a specified element.

   ```python
   numbers = [1, 2, 3, 2]
   index_of_2 = numbers.index(2)
   print(index_of_2)  # Output: 1
   ```

# 7. **`count()` method:**
   - Returns the number of occurrences of a specified element in the list.

   ```python
   numbers = [1, 2, 3, 2]
   count_of_2 = numbers.count(2)
   print(count_of_2)  # Output: 2
   ```

# 8. **`sort()` method:**
   - Sorts the elements of the list in ascending order. It modifies the original list.

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

   - To sort in descending order, use the `reverse` parameter.

   ```python
   numbers.sort(reverse=True)
   print(numbers)  # Output: [9, 5, 4, 3, 2, 1, 1]
   ```

# 9. **`reverse()` method:**
   - Reverses the elements of the list in-place.

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

# 10. **`copy()` method:**
   - Returns a shallow copy of the list.

```python
numbers = [1, 2, 3]
copied_numbers = numbers.copy()
print(copied_numbers)  # Output: [1, 2, 3]
```

These are just a few of the many methods available for manipulating Python lists. Lists are a powerful and versatile data structure, and understanding these methods allows for effective manipulation and handling of data in a list format.