Python provides a variety of built-in functions for list manipulation. Below is a categorized list of commonly used functions and methods to manipulate lists.

1. Adding Elements

- append(x): Adds a single element to the end of the list.

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

- extend(iterable): Adds all elements of an iterable (e.g., list, tuple) to the end of the list.

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

- insert(index, x): Inserts an element at a specified position.

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

2. Removing Elements

- remove(x): Removes the first occurrence of the element x.

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

- pop(index=-1): Removes and returns the element at the given index (last element by default).

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

- clear(): Removes all elements from the list.

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

3. Reordering

- sort(key=None, reverse=False): Sorts the list in place (ascending by default). You can use the key argument to specify a custom sorting function.

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

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

# Sorting with a custom key
my_list = ["apple", "banana", "cherry"]
my_list.sort(key=len)
print(my_list)  # Output: ['apple', 'banana', 'cherry']
```

- reverse(): Reverses the order of the list in place.

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

4. Querying

- index(x, start=0, end=None): Returns the index of the first occurrence of x. Raises a ValueError if the element is not found.

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

- count(x): Returns the number of occurrences of x in the list.

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

5. Copying

- copy(): Returns a shallow copy of the list.

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

6. Joining and Splitting

- join() **(on strings)**: Converts a list of strings into a single string with a specified separator.

```
words = ["hello", "world"]
sentence = " ".join(words)
print(sentence)  # Output: "hello world"
```

- split(): Converts a string into a list (usually the inverse of join()).

```
sentence = "hello world"
words = sentence.split()
print(words)  # Output: ["hello", "world"]
```

7. Iterating

You can loop through a list using a for loop:

```
my_list = [1, 2, 3]
for item in my_list:
    print(item)
```

8. Advanced Manipulation

- **List Comprehensions**: A compact way to create or modify lists.

```
# Creating a list of squares
squares = [x**2 for x in range(5)]
print(squares)  # Output: [0, 1, 4, 9, 16]

# Filtering even numbers
evens = [x for x in range(10) if x % 2 == 0]
print(evens)  # Output: [0, 2, 4, 6, 8]
```

Summary of Common Methods

Method	Description
append(x)	Adds x to the end of the list.
extend(iterable)	Adds elements from an iterable to the list.
insert(i, x)	Inserts x at index i.
remove(x)	Removes the first occurrence of x.
pop(i)	Removes and returns the element at index i.

clear() Removes all elements.
index(x) Finds the index of x.
count(x) Counts occurrences of x.
sort() Sorts the list.
reverse() Reverses the list.
copy() Creates a shallow copy of the list.

Let me know if you'd like a deeper dive into any of these!