List

```
[1, 2, 3]
```

Tuple

```
(1, 2, 3)
```

Set

```
{1, 2, 3}
```

Dictionary

```
{1:'a', 2:'b', 3:'c'}
```

Feature 1:

Can be changed after it is created (mutability).

Feature 1:

Can be changed after it is created (mutability).

Feature 1:

Can be changed after it is created (mutability).

Feature 2:

Can store multiple identical values (duplicates).

Feature 2:

Can store multiple identical values (duplicates).

Feature 2:

Can store multiple identical values (duplicates).

Feature 3:

Maintains a fixed order of the values.

Feature 3:

Maintains a fixed order of the values.

Feature 4:

The values can be accessed by an index.

Feature 4:

The values can be accessed by an index.

Feature 5:

The values can be accessed by a key.

Feature 6:

Supports the operator `+` to add such structures together.

Feature 6:

Supports the operator `+` to add such structures together.

Feature 7:

Supports the method `remove()` for deleting elements.

Feature 7:

Supports the method `remove()` for deleting elements.

Use case 1:

You want to compute and store the frequency of individual words in the lyrics of a song.

Use case 2:

You are programming a chess game, and you want to store the state of the chessboard.

Use case 3:

You want to store a database of 10,000 different words along with their synonyms.

Use case 4:

You are programming an e-shop, and you want to store the contents of a shopping cart.

Use case 5:

You want to store the names of all the people who were born on a certain date (e.g., Jan 1, 2000).

Use case 6:

You want to store the first 50 million prime numbers, and you will often ask whether a given number is within these primes or not.

Use case 7:

You want to store information about the user's screen resolution, which your program receives from the operating system.

Use case 8:

You are programming a 2D game, and you want to store the coordinates of static objects (such as towers) positioned on the game plan.