In real development, it is often necessary to store a set (more than one) of data for later use by the code.
In python, we can use "list" to do so.
Formally, the list puts all the elements inside a pair of brackets `[]` , separated by commas between adjacent elements, as follows:

```
[element1, element2, element3, ..., elementn]
```

Element1 ~ elementn represents an unlimited number of elements in a list, as long as they are data types supported by Python.

create list

Directly use to create the list `[]`.
After you create a list using `[]` , you typically assign it to a variable using `=` , in the following format:

```
listname = [element1 , element2 , element3 , ... , elementn]
```

where listname represents a variable name, element1 ~ elementn represents list elements.
For example, the following defined lists are legal:

```
num = [1, 2, 3, 4, 5, 6, 7]
program = ["C++", "Python", "Java"]
```

Access the list element

You can use an Index to access an element in a list (which gives you the value of an element) or you can use a slice to access a set of elements in a list (which gives you a new sublist) .
The format for using an index to access a list element is:

```
listname[i]
```

where listname represents the name of list, i represents an index value. The index of a list can be positive or negative.

Negative means from back to front.
The format for using slices to access list elements is:

```
listname[start : end : step]
```

where, listname represents the name of list, start represents the starting position, end represents the ending position, step represents step-size.
For exaple,

```
numbers = [7,5,48,61,16,8,1]

print(numbers[3])  #positive index
print(numbers[-4])  #negative index

print(numbers[9: 18])  #positive slice
print(numbers[9: 18: 3])  #specify the step-size
print(numbers[-6: -1])  #negative slice
```

Tuple

Tuples are another data type, similar to list.
Tuples are identified by `()` . Internal elements are separated by commas. However, tuples can not be assigned twice, which is equivalent to a **read-only** list.

```
numbers = (7,5,48,61,16,8,1)
```