

Tuples () : Creating Tuples

- In mathematics, a tuple is a finite ordered list (sequence) of elements. A tuple is defined as a data structure that comprises an ordered, finite sequence of immutable, heterogeneous elements that are of fixed sizes.
- A tuple is a finite ordered list of values of possibly different types which is used to bundle related values together without having to create a specific type to hold them.
- **Tuples are immutable**
- Once a tuple is created, you cannot change its values. A tuple is defined by putting a comma-separated list of values inside parentheses ().
- Each value inside a tuple is called an item.

Basic Tuple Operations

```
>>> tuple_items = (1, 9, 8, 8)
>>> 1 in tuple_items
True
>>> 25 in tuple_items
False
>>> tuple_1 = (9, 8, 7)
>>> tuple_2 = (9, 1, 1)
>>> tuple_1 > tuple_2
True
>>> tuple_1 != tuple_2
True
```



Built-In Functions Used on Tuples

Built-In Functions Used on Tuples

Built-In Functions	Description
<code>len()</code>	The <code>len()</code> function returns the numbers of items in a tuple.
<code>sum()</code>	The <code>sum()</code> function returns the sum of numbers in the tuple.
<code>sorted()</code>	The <code>sorted()</code> function returns a sorted copy of the tuple as a list while leaving the original tuple untouched.

```
For example:  
>>> years = (1987, 1985, 1981, 1986)  
>>> len(years)  
>>> sum(years)  
>>> sorted_years = sorted(years)  
>>> sorted_years
```

If an item within a tuple is mutable, then you can change it. Consider the presence of a list as an item in a tuple, then any changes to the list get reflected on the overall items in the tuple. For example,

```
1. >>> german_cars = ["porsche", "audi", "bmw"]
2. >>> european_cars = ("ferrari", "volvo", "renault", german_cars)
3. >>> european_cars
('ferrari', 'volvo', 'renault', ['porsche', 'audi', 'bmw'])
4. >>> european_cars[3].append("mercedes")
5. >>> german_cars
['porsche', 'audi', 'bmw', 'mercedes']
6. >>> european_cars
('ferrari', 'volvo', 'renault', ['porsche', 'audi', 'bmw', 'mercedes'])
```

Tuple Methods

>>> dir(tuple)

Various Tuple Methods

Tuple Methods	Syntax	Description
count()	tuple_name.count(item)	The <i>count()</i> method counts the number of times the item has occurred in the tuple and returns it.
index()	tuple_name.index(item)	The <i>index()</i> method searches for the given item from the start of the tuple and returns its index. If the value appears more than once, you will get the index of the first one. If the item is not present in the tuple, then <i>ValueError</i> is thrown by this method.

Note: Replace the word "tuple_name" mentioned in the syntax with your *actual tuple name* in your code.

```
1 channels = ("ngc", "discovery", "animal_planet", "history", "ngc")
2 print(channels.count("ngc"))
3 print(channels.index("history"))
```

2

3

Tuple Packing and Unpacking

The statement `t = 12345, 54321, 'hello!'` is an example of tuple packing.

```
t = 12345, 54321, 'hello!'
print(t)
(12345, 54321, 'hello!')
```

The reverse operation of tuple packing is also possible. This operation is called tuple unpacking and works for any sequence on the right-hand side.

```
x, y, z = t
print(x)
print(y)
print(z)
12345
54321
hello!
```

Using zip() Function

- The zip() function makes a sequence that aggregates elements from each of the iterables (can be zero or more).
- The syntax for zip() function is,
zip(*iterables)
- An iterable can be a list, string, or dictionary. It returns a sequence of tuples, where the i-th tuple contains the i-th element from each of the iterables.

```
1 x = [1, 2, 3]
2 y = [4, 5, 6]
3 zipped = zip(x, y)
4 print(list(zipped))
```

```
[(1, 4), (2, 5), (3, 6)]
```