Python - Tuple

Tuple is a collection of items of any Python data type, same as the list type. Unlike the list, tuple is immutable.

The tuple object contains one or more items, of the same or different types, separated by comma and enclosed in parentheses ().

Syntax:
tuple = (value1, value2, value3,...valueN)

The following declares a tuple type variable.

```
>>> names=("Jeff", "Bill", "Steve", "Mohan")
```

The tuple type can also contain elements of different types.

```
>>> orderItem=(1, "Jeff", "Computer", 75.50, True)
```

It is however not necessary to enclose the tuple elements in parentheses. The tuple object can include elements separated by comma without parentheses.

```
>>> names="Jeff", "Bill", "Steve", "Mohan"
>>> type(names)
<class 'tuple'>
```

The above tuple object orderItem includes five elements. Each individual element in the sequence is accessed by the index in the square brackets []. An index starts with zero, as shown below.

```
>>> orderItem=(1, "Jeff", "Computer", 75.50, True)
>>> orderItem[0]
1
>>> orderItem[1]
```

```
'Jeff'
>>> orderItem[2]
'Computer'
>>> orderItem[3]
75.50
>>> orderItem[4]
True
```

You can use backward indexing also.

```
>>> orderItem=(1, "Jeff", "Computer", 75.50, True)
>>> orderItem[-5]
1
>>> orderItem[-4]
'Jeff'
>>> orderItem[-3]
'Computer'
>>> orderItem[-2]
75.50
>>> orderItem[-1]
True
```

If the element at the specified index does not exist, error "index out of range" will be returned.

```
>>> orderItem=(1, "Jeff", "Computer", 75.50, True)
>>> orderItem[5]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
```

Tuple is immutable. So, once a tuple is created, any operation that seeks to change its contents is not allowed.

For instance, item orderItem cannot be modified and an attempt to do so will result in an error.

```
>>> orderItem=(1, "Jeff", "Computer", 75.50, True)
>>> orderItem[2]="Laptop"
TypeError: 'tuple' object does not support item assignment
```

Tuple is typically used to represent a data structure which may contain objects of different types.

```
>>> date=(2018, June, 15)
>>> student=('Kapil', 'A001', 23, 78.75, 'M', True)
```

Use the del keyword to delete the tuple object.

```
>>> del student
>>> student
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'student' is not defined
```

Tuple Operators

Like string, tuple objects are also a sequence. Hence, the operators used with strings are also available for tuple.

Operator	Description	Example

+ Concatenation	Returns a tuple containing all the elements of the first and the second tuple object.	>>> t1=(1,2,3)
		>>> t2=(4,5,6)
		>>> t1+t2
		(1, 2, 3, 4, 5, 6)
		>>> t2+(7,)
		(4, 5, 6, 7)

* Repetition	Concatenates multiple copies of the same tuple.	>>> t1=(1,2,3)
		>>> t1*4
		(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)

[] slice	Returns the item at the given index. A negative index counts the position from the right side.	>>> t1=(1,2,3,4,5,6)
		>>> t1[3]
		4
		>>> t1[-2]
		5

[:] - Range slice	Fetches the items in the range specified by two index operands separated by the : symbol.
	If the first operand is omitted, the range starts at zero index. If the second operand is omitted, the range goes up to the end of tuple.
		>>> t1=(1,2,3,4,5,6)
		>>> t1[1:3]
		(2, 3)
		>>> t1[3:]
		(4, 5, 6)
		>>> t1[:3]
		(1, 2, 3)

in	Returns true if an item exists in the given tuple.	>>> t1=(1,2,3,4,5,6)
		>>> 5 in t1
		True
		>>> 10 in t1
		False

not in	Returns true if an item does not exist in the given tuple.	>>> t1=(1,2,3,4,5,6)
		>>> 4 not in t1
		False
		>>> 10 not in t1
		True

Built-in Tuple Methods

len()

Returns the number of elements in the tuple.

```
>>> t1=(12,45,43,8,35)
>>> len(t1)
5
```

max()

If the tuple contains numbers, the heighest number will be returned. If the tuple contains strings, the one that comes last in alphabetical order will be returned.

```
>>> t1=(12, 45, 43, 8, 35)
>>> max(t1)
45
>>> t2=('python', 'java', 'C++')
>>> max(t2)
'python'
```

min()

If the tuple contains numbers, the lowest number will be returned. If the tuple contains strings, the one that comes first in alphabetical order will be returned.

```
>>> t1=(12,45,43,8,35)
>>> min(t1)
8
>>> t2=('python', 'java', 'C++')
```

```
>>> min(t2)
'C++'
```

Note that if the tuple object contains strings as well as numeric items, max() and min() functions throw error.

```
>>> t1=(1,'Test', 12.22)
>>> max(t1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'str' and 'int'
```