12. Python 3 – Tuples

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists. Tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally, you can put these comma-separated values between parentheses also. For example-

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

The empty tuple is written as two parentheses containing nothing.

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value.

```
tup1 = (50,)
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain the value available at that index. For example-

```
#!/usr/bin/python3
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
```

When the above code is executed, it produces the following result-

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Tuples are immutable, which means you cannot update or change the values of tuple elements. You are able to take portions of the existing tuples to create new tuples as the following example demonstrates.

```
#!/usr/bin/python3

tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2
print (tup3)
```

When the above code is executed, it produces the following result-

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example-

```
#!/usr/bin/python3
tup = ('physics', 'chemistry', 1997, 2000);
print (tup)
del tup;
print "After deleting tup : "
print tup
```

This produces the following result.

Note: An exception is raised. This is because after **del tup,** tuple does not exist any more.

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
```

```
NameError: name 'tup' is not defined
```

Basic Tuples Operations

Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.

In fact, tuples respond to all of the general sequence operations we used on strings in the previous chapter.

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1,2,3) : print (x, end=' ')	1 2 3	Iteration

Indexing, Slicing, and Matrixes

Since tuples are sequences, indexing and slicing work the same way for tuples as they do for strings, assuming the following input-

```
T=('C++', 'Java', 'Python')
```

Python Expression	Results	Description
T[2]	'Python'	Offsets start at zero
T[-2]	'Java'	Negative: count from the right
T[1:]	('Java', 'Python')	Slicing fetches sections

No Enclosing Delimiters

No enclosing Delimiters is any set of multiple objects, comma-separated, written without identifying symbols, i.e., brackets for lists, parentheses for tuples, etc., default to tuples, as indicated in these short examples.

Built-in Tuple Functions

Python includes the following tuple functions-

SN	Function with Description
1	cmp(tuple1, tuple2) No longer available in Python 3.
2	len(tuple) Gives the total length of the tuple.
3	max(tuple) Returns item from the tuple with max value.
4	min(tuple) Returns item from the tuple with min value.
5	tuple(seq) Converts a list into tuple.

Tuple len() Method

Description

The **len()** method returns the number of elements in the tuple.

Syntax

Following is the syntax for len() method-

```
len(tuple)
```

Parameters

tuple - This is a tuple for which number of elements to be counted.

Return Value

This method returns the number of elements in the tuple.

Example

The following example shows the usage of len() method.

```
#!/usr/bin/python3
tuple1, tuple2 = (123, 'xyz', 'zara'), (456, 'abc')
print ("First tuple length : ", len(tuple1))
print ("Second tuple length : ", len(tuple2))
```

When we run above program, it produces following result-

```
First tuple length :  3
Second tuple length :  2
```

Tuple max() Method

Description

The **max()** method returns the elements from the tuple with maximum value.

Syntax

Following is the syntax for max() method-

```
max(tuple)
```

Parameters

tuple - This is a tuple from which max valued element to be returned.

Return Value

This method returns the elements from the tuple with maximum value.

Example

The following example shows the usage of max() method.

```
#!/usr/bin/python3
tuple1, tuple2 = ('maths', 'che', 'phy', 'bio'), (456, 700, 200)
print ("Max value element : ", max(tuple1))
print ("Max value element : ", max(tuple2))
```

When we run the above program, it produces the following result-

```
Max value element :  phy
Max value element :  700
```

Tuple min() Method

Description

The **min()** method returns the elements from the tuple with minimum value.

Syntax

Following is the syntax for min() method-

```
min(tuple)
```

Parameters

tuple - This is a tuple from which min valued element is to be returned.

Return Value

This method returns the elements from the tuple with minimum value.

Example

The following example shows the usage of min() method.

```
#!/usr/bin/python3
tuple1, tuple2 = ('maths', 'che', 'phy', 'bio'), (456, 700, 200)
print ("min value element : ", min(tuple1))
print ("min value element : ", min(tuple2))
```

When we run the above program, it produces the following result-

```
min value element :  bio
min value element :  200
```

Tuple tuple() Method

Description

The **tuple()** method converts a list of items into tuples.

Syntax

Following is the syntax for tuple() method-

```
tuple( seq )
```

Parameters

seq - This is a tuple to be converted into tuple.

Return Value

This method returns the tuple.

Example

The following example shows the usage of tuple() method.

```
#!/usr/bin/python3
list1= ['maths', 'che', 'phy', 'bio']
tuple1=tuple(list1)
print ("tuple elements : ", tuple1)
```

When we run the above program, it produces the following result-

```
tuple elements :  ('maths', 'che', 'phy', 'bio')
```