

Tuples

A tuple is a sequence of immutable objects, therefore tuple cannot be changed. It can be used to collect different types of object.

The objects are enclosed within parenthesis and separated by comma.

Tuple is similar to list. Only the difference is that list is enclosed between square bracket, tuple between parenthesis and List has mutable objects whereas Tuple has immutable objects.

Example

```
data = (43, "Ram", "Python", 59, 99)
print(type(data))
print(data)

data2 = 65, 43, "Akshay", "Language"
print(type(data2))
print(data2)

# -----OUTPUT-----
<class 'tuple'>
(43, 'Ram', 'Python', 59, 99)
<class 'tuple'>
(65, 43, 'Akshay', 'Language')
```

Python Empty Tuple Example

There can be an empty Tuple also which contains no object. Let's see an example of empty tuple:-

```
tuple1 = ()
print(tuple1)
# Python Single Object Tuple Example
# For a single valued tuple, there must be a comma at the end of the value.

tuple1 = (10,)
print(tuple1)

# -----OUTPUT-----
()
(10,)
```

Python Tuple of Tuples Example

Tuples can also be nested, it means we can pass tuple as an element to create a new tuple. See, the following example in which we have created a tuple that contains tuples as the object:-

```
tupl1 = 'a', 'mahesh', 10.56
tupl2 = tupl1, (10, 20, 30)

print(tupl1)
print(tupl2)

# -----OUTPUT-----
('a', 'mahesh', 10.56)
(('a', 'mahesh', 10.56), (10, 20, 30))
```

Accessing Tuple

Accessing of tuple is pretty easy; we can access tuple in the same way as List. See, the following example:-

```
data1 = (1, 2, 3, 4)
data2 = ('x', 'y', 'z')

print(data1[0])
print(data1[0:2])
print(data2[-3:-1])
print(data1[0:])
print(data2[:2])

# -----OUTPUT-----
1
(1, 2)
('x', 'y')
(1, 2, 3, 4)
('x', 'y')
```

As we know Tuples are immutable that means we can't change/update value at position

```
data = (54, "Python", 21, 43, 54, "87", "Language")
data[3]=100
print(data)

# -----OUTPUT-----
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    data[3]=100
TypeError: 'tuple' object does not support item assignment
```

Python Tuple Operations

Python allows us to perform various operations on the tuple. Following are the common tuple operations:-

Adding Tuples Example

Tuple can be added by using the concatenation operator(+) to join two tuples.

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")
data2 = (65, 11, "Computer", "Cycle", 76, 88)
data3 = data1+data2
print(data1)
print(data2)
print(data3)

# -----OUTPUT-----
(54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88)
(54, 'Python', 21, 43, 54, '87', 'Language', 65, 11, 'Computer', 'Cycle', 76,
88)
```

Replicating Tuple Example

Replicating means repeating. It can be performed by using '*' operator by a specific number of time.

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")
data2 = (65, 11, "Computer", "Cycle", 76, 88)
data3 = data1*2
data4 = data2*3
print(data1)
print(data2)
print(data3)
print(data4)

# -----OUTPUT-----
(54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88)
(54, 'Python', 21, 43, 54, '87', 'Language', 54, 'Python', 21, 43, 54, '87', 'Language')
(65, 11, 'Computer', 'Cycle', 76, 88, 65, 11, 'Computer', 'Cycle', 76, 88, 65, 11, 'Computer', 'Cycle', 76, 88)
```

Python Tuple Slicing Example

A subpart of a tuple can be retrieved on the basis of index. This subpart is known as tuple slice:-

```
data1 = (54, "Python", 21, 43, 54, "87", "Language")

print(data1[0:2])
print(data1[4])
print(data1[:-1])
print(data1[-5:])
print(data1)

# -----OUTPUT-----
(54, 'Python')
54
(54, 'Python', 21, 43, 54, '87')
(21, 43, 54, '87', 'Language')
(54, 'Python', 21, 43, 54, '87', 'Language')
```

Python Tuple Deleting Example

Deleting individual element from a tuple is not supported. However the whole of the tuple can be deleted using the **del** statement.

```
data = (54, "Python", 21, 43, 54, "87", "Language")
print(data)
del data

print(data)

# -----OUTPUT-----
(54, 'Python', 21, 43, 54, '87', 'Language')
Traceback (most recent call last):
  File "test.py", line 5, in <module>
    print(data)
NameError: name 'data' is not defined
```

COMPUSOFT

Functions of Tuples

min(tuple)

It returns the minimum value from a tuple.

max(tuple)

It returns the maximum value from the tuple.

len(tuple)

It gives the length of a tuple.

tuple(sequence)

It converts the sequence into tuple.

Python Tuple min(tuple) Method Example

This method is used to get min value from the sequence of tuple.

```
data = (10, 20, 40.6, -33, 540)
print(min(data))

# -----OUTPUT-----
-33
```

Python Tuple max(tuple) Method Example

This method is used to get max value from the sequence of tuple.

```
data = (10, 20, 40.6, -33, 540)
print(max(data))

# -----OUTPUT-----
540
```

Python Tuple len(tuple) Method Example

This method is used to get length of the tuple.

```
data = (10, "Ravi", 40.6, "Python", 540)
print(len(data))

# -----OUTPUT-----
5
```

Python tuple(sequence) Method Example

It is used to convert sequence into tuple.

```
data1 = [54, 76, 98, "Python", 12]
print(type(data1), data1)

data2 = tuple(data1)
print(type(data2), data2)

# -----OUTPUT-----
<class 'list'> [54, 76, 98, 'Python', 12]
<class 'tuple'> (54, 76, 98, 'Python', 12)
```

Why should we use Tuple? (Advantages of Tuple)

Processing of Tuples are faster than Lists.

It makes the data safe as Tuples are immutable and hence cannot be changed.