

Tuple Membership Test:

We can test if an item exists in a tuple or not, using the keyword `in`.

Example:

```
PyTuple=(1,2,3,[4,5])
print(type(PyTuple))#<class 'tuple'>
print(1 in PyTuple)
print(10 not in PyTuple)
print(5 in PyTuple)
```

Iterating Through a Tuple

Using a `for` loop we can iterate through each item in a tuple.

Example:

```
for name in ('KSRaju','Dinesh',"NareshIT"):
    print("Hai",name)
```

Slice a Tuple:

We can access a range of items in a tuple by using the slicing operator - colon ":"

Example:

```
PyTuple = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
print(PyTuple[3:5])
print(PyTuple[:6])
print(PyTuple[5:])
print(PyTuple[:])
print(PyTuple[-8:-4])
```

Slice of a tuple using step parameter

```
PyTuple = tuple("DATA SCIENCE")
print(PyTuple)
print(PyTuple[2:9:2])
print(PyTuple[::-4])
print(PyTuple[9:2:-3])
```

How operators `+` and `*` are used with a Python tuple?

```
#+ Operator Concatenation
PyTuple1=(3, 6, 9, 12, 15)
PyTuple2=("P", "Y", "T", "H", "O", "N")
PyTuple3=(True, False)
PyTuple4=PyTuple1+PyTuple2+PyTuple3
print(PyTuple4)
#Repetation * Operator
PyTuple5=5,
print(PyTuple5*6)
```

Python Tuple Methods:

Tuple object supports the following list of Methods:

Method Description

<code>count(x)</code>	Return the number of items that is equal to <code>x</code>
<code>index(x)</code>	Return index of first item that is equal to <code>x</code>

In Python how to know the number of times an item has repeated

```
PyTuple = 2, 4, 5, 6, 2, 3, 4, 4, 7
print(PyTuple)
```

```
print(PyTuple.count(4))
print(PyTuple.count(7))
print(PyTuple.count(5))
```

Find the index of an item of the tuple:

```
PyTuple=tuple("index tuple")
print(PyTuple)
print(PyTuple.index("p"))
print(PyTuple.index("e", 5))
print(PyTuple.index("e", 3, 6))
```

Add item in tuple!

Method-I

```
tuplez = (4, 6, 2, 8, 3, 1)
print(tuplez)
tuplez = tuplez + (9,)
print(tuplez)
```

Method-II

```
tuplez = (4, 6, 2, 8, 3, 1)
print(tuplez)
listx = list(tuplez)
listx.append(30)
tuplez = tuple(listx)
print(tuplez)
```

Example:

```
PyTuple=(1,2,3,4)
PyTuple.append((5,6,7))
print(len(PyTuple))
```

Deleting a Tuple:

We cannot change the elements in a tuple. That also means we cannot delete or remove items from a tuple. But deleting a tuple entirely is possible using the keyword del.

Example:

```
PyTuple=('P','Y','T','H','O','N','S','C','R')
del PyTuple
print(PyTuple) #NameError: name 'PyTuple' is not defined
```

Example:

Remove an item from a tuple (Through List)

```
PyTuple="P", "Y", "T", "H", "O", "N", "S"
PyList=list(PyTuple)
PyList.remove("P")
PyTuple=tuple(PyList)
print(PyList) #['Y', 'T', 'H', 'O', 'N', 'S']
```

Built-in Functions with Tuple

```
all()    any()    enumerate()      len()    max()    min()    sorted()
sum()    tuple()
```

Python enumerate()

The enumerate() method adds counter to an iterable and returns it (the enumerate object).

Syntax

```
enumerate(iterable, start=0)
```

Example:

```
bd = ('Big Data', 'Hadoop', 'Spark')
enumerateData = enumerate(bd)
print(type(enumerateData))
print(tuple(enumerateData))
enumerateData = enumerate(bd, 10)
print(tuple(enumerateData))
```

Example:

```
bd = ('Big Data', 'Hadoop', 'Spark')
for item in enumerate(bd):
    print(item)
```

Example:

```
for count, item in enumerate(bd, 100):
    print(count, item)
```

all() Function:

It returns True when all elements in the given iterable are true. If not, it returns False.

Syntax:

```
all(iterable)
```

all() Parameters

iterable - any iterable (list, tuple, dictionary, etc.) which contains the elements

The all() method returns:

True - If all elements in an iterable are true
False - If any element in an iterable is false

Truth table for all()

When	Return Value
All values are true	True
All values are false	False
One value is true (others are false)	False
One value is false (others are true)	False
Empty Iterable	True

NOTE: 0 and 1 are the binary values like False, True

Example: How all() works for tuple and lists?

```
s = (1, 3, 4, 5)
print(all(s))
```

```
s = (0, False)
print(all(s))
```

```
s = [0, False, 5]
print(all(s))
```

```
s = []
print(all(s))
```

Python any()

It Returns True if any element of an iterable is true. If not, this method returns False.

Syntax:

```
any(iterable)
```

The any method returns:

True if at least one element of an iterable is true

False if all elements are false or if an iterable is empty

When	Return Value
All values are true	True
All values are false	False
One value is true (others are false)	True
One value is false (others are true)	True
Empty Iterable	False

NOTE: 0 and 1 are the binary values like False, True

Example:

```
s = (1, 3, 4, 0)
print(any(s))
```

```
s = (0, False)
print(any(s))
```

```
s = [0, False, 5]
print(any(s))
```

```
s = []
print(any(s))
```

Python len()

The len() function returns the number of items (length) of an object.

Syntax

```
len(s)
```

Example:

```
testList = []
print(len(testList))
```

Example:

```
#The size of a tuple
tuplez = tuple("PYTHON")
print(tuplez)
print(len(tuplez))
```

Example:

```
testRange = range(1, 10)
print(len(testRange))
```

Python sorted()

The sorted() method sorts the elements of a given iterable in a specific order - Ascending or Descending.

Syntax
sorted(iterable[, reverse])

Parameters:

iterable - sequence (string, tuple, list) or collection (set, dictionary, frozen set) or any iterator
reverse (Optional) - If true, the sorted list is reversed (or sorted in Descending order)

Sort a given sequence: string, list and tuple

Example:

```
pyList = ['e', 'a', 'u', 'o', 'i']
print(sorted(pyList))
```

Example

```
pyString = 'Python'
print(sorted(pyString))
```

Example

```
pyTuple = ('e', 'a', 'u', 'o', 'i')
print(sorted(pyTuple))
```

Example

```
pyTuple = ('e', 'a', 'u', 'o', 'i')
print(sorted(pyTuple, reverse=True))
```

Python sum()

The sum() function adds the items of an iterable and returns the sum.

Syntax:

```
sum(iterable)
```

sum() Parameters

iterable - iterable (list, tuple, dict etc) whose item's sum is to be found. Normally, items of the iterable should be numbers.

Example:

```
numbers = [2.5, 3, 4, -5]
numbers = sum(numbers)
print("Sum of Numbers is: " ,numbers)
```

Python tuple() Function

The tuple() built-in is used to create a tuple in Python.

Syntax:

```
tuple(iterable)
```

Parameter:

iterable (optional) an iterable (list, range etc.) or an iterator object

```
# creating a tuple from a list
t2 = tuple([1, 4, 6])
print(t2)
```

```
# creating a tuple from a string
```

```
t1 = tuple('Python')
print(t1)
```

```
#tuple to List
MyTuple=(1,2,3,4)
print(type(MyTuple))
print(MyTuple)
MyList=list(MyTuple)
print(type(MyList))
print(MyList)
```

Python Tuple vs List:

Lists are mutable while Tuples are immutable.
Lists have variable length while tuple has fixed length.
Lists has more functionality than tuple.
List object size is comparatively larger than Tuple.
Execution of tuple is faster than Lists.

`cmp()`: Deprecated(Removed) from PYTHON-3.x version
1 It compares the elements of both tuples.
2 If both tuples are equal then returns 0
3 If the first tuple is less than second tuple then it returns -1
4 If the first tuple is greater than second tuple then it returns +1

Example:

```
PyTuple1=(10,20,30)
PyTuple2=(40,50,60)
PyTuple3=(10,20,30)
print(cmp(PyTuple1,PyTuple2))#-1
print(cmp(PyTuple1,PyTuple3))#0
print(cmp(PyTuple2,PyTuple3))#1
```

Note:

`cmp()` function is available only in Python2 but not in Python 3

Tuple Comprehension:

Tuple Comprehension is not supported by Python.

```
t = ( x**2 for x in range(1,6))
```

Here we are not getting tuple object and we are getting generator object.

Example:

```
PyTuple=( x**2 for x in range(1,6))
print(type(PyTuple))
for x in PyTuple:
    print(x)
```