015 Python Tuples

Tuples

A tuple is a container which holds a series of comma-separated values (items or elements) between parentheses such as an (x, y) co-ordinate. Tuples are like lists, except they are immutable (i.e. you cannot change its content once created) and can hold mix data types. Tuples play a sort of "struct" role in Python -- a convenient way to pass around a little logical, fixed size bundle of values.

Tuple: Commands

Tuple is an immutable and hashable list.

```
<tuple> = ()
<tuple> = (<el>, )
<tuple> = (<el_1>, <el_2> [, ...])
```

Named Tuple

Tuple's subclass with named elements.

```
>>> from collections import namedtuple
>>> Point = namedtuple('Point', 'x y')
>>> p = Point(1, y=2)
Point(x=1, y=2)
>>> p[0]
1
>>> p.x
1
>>> getattr(p, 'y')
2
>>> p._fields  # Or: Point._fields
('x', 'y')
```

Create a tuple?

To create a tuple, just list the values within parenthesis separated by commas. The "empty" tuple is just an empty pair of parenthesis

```
>>> #create an empty tuple

>>> tuplex = ()

>>> print (tuplex)
```

```
()
>>> #create a tuple with different data types
>>> tuplex = ('tuple', False, 3.2, 1)
>>> print (tuplex)
('tuple', False, 3.2, 1)
>>> #create a tuple with numbers, notation without parenthesis
>>> tuplex = 4, 7, 3, 8, 1
>>> print (tuplex)
(4, 7, 3, 8, 1)
>>> #create a tuple of one item, notation without parenthesis
>>> tuplex = 4,
>>> print (tuplex)
(4,)
>>> #create an empty tuple with tuple() function built-in Python
>>> tuplex = tuple()
>>> print (tuplex)
()
>>> #create a tuple from a iterable object
>>> tuplex = tuple([True, False])
>>> print (tuplex)
(True, False)
>>>
```

How to get an item of the tuple in Python?

```
>>> #create a tuple
>>> tuplex = ("g", "o", "o", "g", "l", "e", 5)

>>> print(tuplex)

('g', 'o', 'o', 'g', 'l', 'e', 5)

>>> #get item (4th element)of the tuple by index

>>> item = tuplex[3]

>>> print(item)

e

>>> #get item (4th element from last)by index negative

>>> item1 = tuplex[-4]

>>> print(item1)

u

>>>
```

How to know if an element exists within a tuple in Python?

```
>>> #create a tuple

>>> tuplex = ("g", "o", "o", "g", "l", "e", 5)

>>> print(tuplex)

('g', 'o', 'o', 'g', 'l', 'e', 5)

>>> #use in statement

>>> print("r" in tuplex)

True

>>> print(5 in tuplex)

False

>>>
```

List to tuple

```
>>> #create list

>>> listx = [5, 10, 7, 4, 15, 3]

>>> print(listx)

[5, 10, 7, 4, 15, 3]

>>> #use the tuple() function built-in Python, passing as parameter
the list

>>> tuplex = tuple(listx)

>>> print(tuplex)

(5, 10, 7, 4, 15, 3)

>>>
```

Unpack a tuple in several variables

```
>>> #create a tuple

>>> tuplex = 4, 8, 3

>>> print(tuplex)

(4, 8, 3)

>>> n1, n2, n3 = tuplex

>>> #unpack a tuple in variables

>>> print(n1 + n2 + n3)

15

>>> #the number of variables must be equal to the number of items of
the tuple

>>> n1, n2, n3, n4 = tuplex

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>
```

```
ValueError: need more than 3 values to unpack
>>>
```

Add item in Python tuple!

```
>>> #create a tuple
>>> tuplex = (4, 6, 2, 8, 3, 1)
>>> print(tuplex)
(4, 6, 2, 8, 3, 1)
>>> #tuples are immutable, so you can not add new elements
>>> #using merge of tuples with the + operator you can add an element
and it will create a new tuple
>>> tuplex = tuplex + (9,)
>>> print(tuplex)
(4, 6, 2, 8, 3, 1, 9)
>>> #adding items in a specific index
>>> tuplex = tuplex[:5] + (15, 20, 25) + tuplex[:5]
>>> print(tuplex)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3)
>>> #converting the tuple to list
>>> listx = list(tuplex)
>>> #use different ways to add items in list
>>> listx.append(30)
>>> tuplex = tuple(listx)
>>> print(tuplex)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3, 30)
>>>
```

Clone a tuple

```
>>> from   import deep
>>> #create a tuple
>>> tuplex = ("HELLO", 5, [], True)
>>> print(tuplex)
('HELLO', 5, [], True)
>>> #make a   of a tuple using deep () function
>>> tuplex_clone = deep (tuplex)
>>> tuplex_clone[2].append(50)
>>> print(tuplex_clone)
('HELLO', 5, [50], True)
>>> print(tuplex)
('HELLO', 5, [], True)
>>>
```

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

```
>>> #create a tuple
>>> tuplex = 2, 4, 5, 6, 2, 3, 4, 4, 7
>>> print(tuplex)
(2, 4, 5, 6, 2, 3, 4, 4, 7)
>>> #return the number of times it appears in the tuple.
>>> count = tuplex.count(4)
>>> print(count)
```

```
3

>>> count = tuplex.count(7)

>>> print(count)

1

>>> count = tuplex.count(5)

>>> print (count)

1

>>>
```

Remove an item from a tuple

```
>>> #create a tuple

>>> tuplex = "g", "o", "o", "g", "l", "e",3

>>> print(tuplex)

('g', 'o', 'o', 'g', 'l', 'e', 5)

>>> #tuples are immutable, so you can not remove elements

>>> #using merge of tuples with the + operator you can remove an item
and it will create a new tuple

>>> tuplex = tuplex[:2] + tuplex[3:]

>>> print(tuplex)

('g',  'o', 'g', 'l', 'e',3)

>>> #converting the tuple to list

>>> listx = list(tuplex)

>>> #use different ways to remove an item of the list

>>> listx.remove("l")

>>> #converting the tuple to list

>>> tuplex = tuple(listx)
>>> print(tuplex)

('w', 3, 'r', 'e', 's')

>>>
```

Slice a tuple

```
>>> #create a tuple

>>> tuplex = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)

>>> #used tuple[start:stop] the start index is inclusive and the stop
index

>>> _slice = tuplex[3:5]

#is exclusive.

>>> print(_slice)

(5, 4)

>>> #if the start index isn't defined, is taken from the beg inning of
the tuple.

>>> _slice = tuplex[:6]

>>> print(_slice)

(2, 4, 3, 5, 4, 6)

>>> #if the end index isn't defined, is taken until the end of the
tuple

>>> _slice = tuplex[5:]

>>> print(_slice)

(6, 7, 8, 6, 1)

>>> #if neither is defined, returns the full tuple

>>> _slice = tuplex[:]

>>> print(_slice)

(2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
```

```
>>> #The indexes can be defined with negative values

>>> _slice = tuplex[-8:-4]

>>> print(_slice)

(3, 5, 4, 6)

>>>
```

Find the index of an item of the tuple

```
>>> #create a tuple

>>> tuplex = tuple("index tuple")

>>> print(tuplex)

('i', 'n', 'd', 'e', 'x', ' ', 't', 'u', 'p', 'l', 'e')

>>> #get index of the first item whose value is passed as parameter

>>> index = tuplex.index("p")

>>> print(index)

8

>>> #define the index from which you want to search

>>> index = tuplex.index("p", 5)

>>> print(index)

8

>>> #define the segment of the tuple to be searched

>>> index = tuplex.index("e", 3, 6)

>>> print(index)

3

>>> #if item not exists in the tuple return ValueError Exception

>>> index = tuplex.index("y")
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: tuple.index(x): x not in tuple
>>>
```

The size of a tuple

```
>>> tuplex = tuple("google")       #create a tuple
>>> print(tuplex)
('g', 'o', 'o', 'g', 'l', 'e')
>>> #use the len() function to known the length of tuple.
>>> print(len(tuplex))
10
>>>
```

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

```
>>> #create a tuple
>>> tuplex = 5, #create a tuple
>>> #The * operator allow repeat the items in the tuple
>>> print(tuplex * 6)
(5, 5, 5, 5, 5, 5)
>>> #create a tuple with repeated items.
>>> tuplex = (5, 10, 15) * 4
>>> print(tuplex)
(5, 10, 15, 5, 10, 15, 5, 10, 15, 5, 10, 15)
>>> #create three tuples
>>> tuplex1 = (3, 6, 9, 12, 15)

>>> tuplex2 = ("g", "o", "o", "g", "l", "e")

>>> tuplex3 = (True, False)

>>> #The + operator allow create a tuple joining two or more tuples

>>> tuplex = tuplex1 + tuplex2 + tuplex3

>>> print(tuplex)

(3, 6, 9, 12, 15, 'g', 'o', 'o', 'g', 'l', 'e', True, False)

>>>
```

Slice of a tuple using step parameter

```
>>> #create a tuple

>>> tuplex = tuple("HELLO WORLD")

>>> print(tuplex)

('H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D')

>>> #step specify an increment between the elements to cut of the
tuple.

>>> _slice = tuplex[2:9:2]  #tuple[start:stop:step]

>>> print(_slice)

('L', 'O', 'W', 'R')

>>> #returns a tuple with a jump every 3 items.

>>> _slice = tuplex[::4]

>>> print(_slice)

('H', 'O', 'R')

>>> #when step is negative the jump is made back

>>> _slice = tuplex[9:2:-4]

>>> print(_slice)
```

```
('L', ' ')

>>> #when step is negative the jump is made back

>>> _slice = tuplex[9:2:-3]

>>> print(_slice)

('L', 'W', 'L')

>>>
```

Modify items of a tuple

```
>>> #create a tuple

>>> tuplex = ("g", 5, "l", [], False)

>>> print(tuplex)

('g', 5, 'l', [], False)

>>> #tuples are immutable, so you can not modify items which are also
immutable, as str, boolean, numbers etc.

>>> tuplex[3].append(200)

>>> print(tuplex)

('w', 3, 'r', [200], False)

>>>
```