**Tuples**

A tuple is a sequence of immutable Python objects.

Tuples are immutable, and usually contain a heterogeneous sequence of elements that are accessed via unpacking or indexing.

Difference between the tuples and the lists:

1. Tuples cannot be changed unlike lists
2. Tuples use parentheses, whereas lists use square brackets
3. Tuples are usually faster than lists

# Creating a tuple

To create a tuple put comma-separated values between parentheses

tup1 = ('physics', 'chemistry', 1997, 2000)

tup2 = (1, 2, 3, 4, 5 )

tup3 = "a", "b", "c", "d"

tup4 = (); # empty tuple

tup5 = (50,)

Note:

* To write a tuple containing a single value you have to include a comma, even though there is only one value.
* Like string indices, tuple indices start at 0.

The statement t = 12345, 54321, 'hello!' is an example of tuple packing.

The reverse operation is also possible:

x, y, z = t

This is called, appropriately enough, sequence unpacking and works for any sequence on the right-hand side.

Sequence unpacking requires that there are as many variables on the left side of the equals sign as there are elements in the sequence.

Note that multiple assignment is really just a combination of tuple packing and sequence unpacking.

# Accessing Values in Tuples

Use the square brackets for slicing along with the index or indices to obtain the value available at that index.

#!/usr/bin/python3

tup1 = ('physics', 'chemistry', 1997, 2000)

tup2 = (1, 2, 3, 4, 5 )

tup3 = "a", "b", "c", "d"

tup4 = () # empty tuple

tup5 = (50,)

print ("tup1[0] : ", tup1[0])

print ("tup2[0] : ", tup2[0])

print ("tup3[0] : ", tup3[0])

#print ("tup4[0] : ", tup4[0]) # IndexError: tuple index out of range

print ("tup5[0] : ", tup5[0])

print ("tup1[-1] : ", tup1[-1])

print ("tup2[-1] : ", tup2[-1])

print ("tup3[-1] : ", tup3[-1])

#print ("tup4[-1] : ", tup4[-1]) # IndexError: tuple index out of range

print ("tup5[-1] : ", tup5[-1])

print ("tup2[1:3] : ", tup2[1:3])

print ("tup2[-4:-1] : ", tup2[-4:-1])

Output:

tup1[0] : physics

tup2[0] : 1

tup3[0] : a

tup5[0] : 50

tup1[-1] : 2000

tup2[-1] : 5

tup3[-1] : d

tup5[-1] : 50

tup2[1:3] : (2, 3)

tup2[-4:-1] : (2, 3, 4)

# 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.

#!/usr/bin/python3

tup1 = (1, 2, 3, 4, 5)

tup2 = ('word1', 'word2', 'word3', 'word4', 'word5')

#tup1[0] = 100; # TypeError: 'tuple' object does not support item assignment

tup3 = tup1[1:3] + tup2[-3:-1]

print (tup3)

Output:

(2, 3, 'word3', 'word4')

# 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.

#!/usr/bin/python3

tup = ('physics', 'chemistry', 1997, 2000);

print ("Tuple : ", tup)

# del tup[0]; # TypeError: 'tuple' object doesn't support item deletion

del tup;

# print ("After deleting tup : ", tup) # NameError: name 'tup' is not defined

Output:

Tuple : ('physics', 'chemistry', 1997, 2000)

# Basic Tuples Operations

Note: Result is a new tuple

|  |  |  |
| --- | --- | --- |
| 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 |