PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Tuples in python

Prof. Sindhu R Pai
PCPS Theory Anchor - 2024
Department of Computer Science and Engineering

PES UNIVERSITY

CELEBRATING 50 YEARS

Tuple has the following attributes:

• The element of the tuple can be of any type.

(There is no restriction on the type of the element.). A tuple is same as list, except that

the **set of elements is enclosed in parentheses instead of square brackets** and these

elements are immutable.

Ex:

```
t = ('python',1,3.4) # heterogeneous
t1 = (11,22,33,44) #homogeneous
```

Tuple is a data structure containing zero or more elements.

Creation of tuple

- An empty tuple can be created using a constructor,
        tuple () or ().
- Tuple with two elements.
        t3 = (1, 2)
- Tuple with a single element.
        t3=(1,)

- Each element of a tuple can be referred to by a index or a subscript.

- Index is an integer.

Ex:

```
>>> t1=(11,22,33,44,55)
>>> t1[1]
    22
>>> t1[-1]
    55
```

- Tuples are immutable. Once created, we cannot change the number of elements – no append, no insert, no remove, no delete.

```
>>> z = (5, 4, 3)
>>> z[2] = 0
```

Traceback:'tuple' object does not support item

Assignment

Benefits of usage of type - Tuple:

- Tuples are faster than lists.

- If the user wants to protect the data from accidental changes, tuple can be used.

- Tuples can be used as keys in dictionaries, while lists can't.

- Elements in the tuple can repeat .

    >>> t1=(11,22,33,44,22,11,55)

- Tuple is a sequence .

- Tuple is also iterable - is eager and not lazy.

    t2=(5,10,15,20)
    for i in t2:
            print(i, end=' ')

- Tuples can be nested. We can have tuple of tuples.

    t=(1,2,3,4,(6,7,8))
    for i in t:
            print(i)

- Assignment of one tuple to another causes both to refer to the same tuple.

```
>>> t1 = (11, 22, 33, 44, 22, 11, 55)
>>> t2=t1
>>> print(id(t1))
        2397027797408
>>> print(id(t2))
        2397027797408
```

- tuples can be sliced. This creates a new tuple.

```
t1 = (11, 22, 33, 44, 22, 11, 55)
>>> t1[2:5]
    (33, 44, 22)
>>> t1[::-1]
    (55, 11, 22, 44, 33, 22, 11)
```

Built in Functions

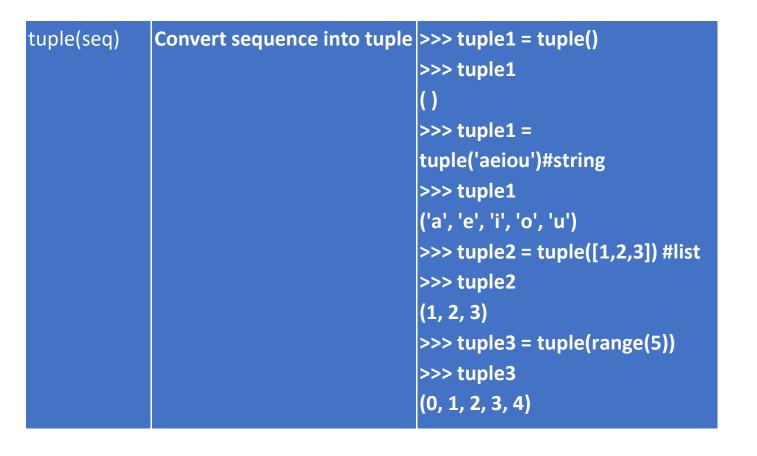| Method	Description	Example

len(tup)	Gives the total length of the tuple.	>>> tuple1 = (10,20,30,40,50)
		>>> len(tuple1)
		   5

sorted()	Takes elements in the tuple and returns a new sorted list. It should be noted that, sorted () does not make any changes to the original tuple	>>> tuple1 = ('rama', 'shama', 'bhama', 'balarama') >>>sorted(tuple1) ['balarama', 'bhama', 'rama', 'shama']

min()	Returns the element from tuple with max value	>>>tuple1=(22,33,11,55,44,120) >>>min(tuple1) 11
max()	Returns the element from tuple with min value	>>>tuple1=(22,33,11,55,44,120) >>>max(tuple1) 120
sum()	Returns the sum of elements of the tuple.	>>>tuple1=(22,33,11,55,44,120) >>>sum(tuple1) 285

tuple(seq)	Convert sequence into tuple	>>> tuple1 = tuple()
		>>> tuple1
		( )
		>>> tuple1 = tuple('aeiou')#string
		>>> tuple1
		('a', 'e', 'i', 'o', 'u')
		>>> tuple2 = tuple([1,2,3]) #list
		>>> tuple2
		(1, 2, 3)
		>>> tuple3 = tuple(range(5))
		>>> tuple3
		(0, 1, 2, 3, 4)

Tuple

Method	Description	Example

count()	Returns the count of the items.	>>> tuple1 = (10,20,30,10,40,10,50)
		>>> tuple1.count(10)
		   3
		>>> tuple1.count(90)
		   0

index()	Returns the index of the item specified	>>> tuple1 = (10,20,30,40,50)
		>>> tuple1.index(30)
		   2
		>>> tuple1.index(90)
		ValueError: tuple.index(x): x not in tuple.

- **Concatenation**:

    Allows us to join two tuples by using operator '+'.

Ex:

>>> tuple1 = (11,33,55,77,99)
>>> tuple2 = (22,44,66,88,100)
>>> tuple1 + tuple2
#concatenates two tuples
(11, 33, 55, 77, 99, 22, 44, 66, 88 100)

- **Repetition**:

The repetition operator **enables the tuple elements to be repeated** multiple

times. The repetition operator **requires the first operand to be a tuple and the**

second operand to be an integer only.

Ex:

```
>>> t1=(1,2,3,4)
>>> t1*2
(1, 2, 3, 4, 1, 2, 3, 4)
>>> t1*t1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'tuple
```

Membership Operator:

in and not in

in returns True if a particular item exists in the list. otherwise False

not in operator returns True if the element is not present, otherwise False

Ex: >>> tup1 = (10,2.2,(22,33,43),('python')) # heterogeneous tuple

             >>> 'python' in tup1

                  True
             >>> 'ruby' not in tup1
                  True

THANK YOU

Department of Computer Science and Engineering

Dr. Shylaja S S, Director, CCBD &CDSAML, PESU
Prof. Sindhu R Pai – sindhurpai@pes.edu
Prof. Chitra G M
Prof. Mohan Kumar A V