Python Tuple

A tuple is a sequence of immutable objects, therefore tuple cannot be changed.

The objects are enclosed within parenthesis and separated by comma.

Tuple is similar to list. Only the difference is that list is enclosed between angular brackets, tuple between parenthesis and List have mutable objects whereas Tuple have immutable objects.

**eg:**

1. >>> data=(10,20,'ram',56.8)
2. >>> data2="a",10,20.9
3. >>> data
4. (10, 20, 'ram', 56.8)
5. >>> data2
6. ('a', 10, 20.9)
7. >>>

NOTE: If Parenthesis is not given with a sequence, it is by default treated as Tuple.

There can be an empty Tuple also which contains no object.

**eg:**

1. tuple1=()

For a single valued tuple, there must be a comma at the end of the value.

**eg:**

1. Tuple1=(10,)

Tuples can also be nested.

**eg:**

1. tupl1='a','mahesh',10.56
2. tupl2=tupl1,(10,20,30)
3. **print** tupl1
4. **print** tupl2

**Output:**

1. >>>
2. ('a', 'mahesh', 10.56)
3. (('a', 'mahesh', 10.56), (10, 20, 30))
4. >>>

Accessing Tuple

Tuple can be accessed in the same way as List.

Some examples are given below:

**eg:**

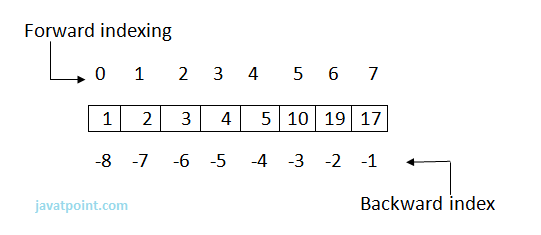
1. data1=(1,2,3,4)
2. data2=('x','y','z')
3. **print** data1[0]
4. **print** data1[0:2]
5. **print** data2[-3:-1]
6. **print** data1[0:]
7. **print** data2[:2]

**Output:**

1. >>>
2. 1
3. (1, 2)
4. ('x', 'y')
5. (1, 2, 3, 4)
6. ('x', 'y')
7. >>>

Elements in a Tuple

Data=(1,2,3,4,5,10,19,17)



1. Data[0]=1=Data[-8] , Data[1]=2=Data[-7] , Data[2]=3=Data[-6] ,
2. Data[3]=4=Data[-5] , Data[4]=5=Data[-4] , Data[5]=10=Data[-3],
3. Data[6]=19=Data[-2],Data[7]=17=Data[-1]

Tuple Operations

Various Operations can be performed on Tuple. Operations performed on Tuple are given as:

**a) Adding Tuple:**

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

**eg:**

1. data1=(1,2,3,4)
2. data2=('x','y','z')
3. data3=data1+data2
4. **print** data1
5. **print** data2
6. **print** data3

**Output:**

>>>

(1, 2, 3, 4)

('x', 'y', 'z')

(1, 2, 3, 4, 'x', 'y', 'z')

>>>

Note: The new sequence formed is a new Tuple.

**b) Replicating Tuple:**

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

**Eg:**

1. tuple1=(10,20,30);
2. tuple2=(40,50,60);
3. **print** tuple1\*2
4. **print** tuple2\*3

**Output:**

>>>

(10, 20, 30, 10, 20, 30)

(40, 50, 60, 40, 50, 60, 40, 50, 60)

>>>

**c) Tuple slicing:**

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

**Eg:**

data1=(1,2,4,5,7)

**print** data1[0:2]

**print** data1[4]

**print** data1[:-1]

**print** data1[-5:]

**print** data1

**Output:**

>>>

(1, 2)

7

(1, 2, 4, 5)

(1, 2, 4, 5, 7)

(1, 2, 4, 5, 7)

>>>

Note: If the index provided in the Tuple slice is outside the list, then it raises an IndexError exception.

Other Operations:

**a) Updating elements in a List:**

Elements of the Tuple cannot be updated. This is due to the fact that Tuples are immutable. Whereas the Tuple can be used to form a new Tuple.

**Eg:**

1. data=(10,20,30)
2. data[0]=100
3. **print** data

**Output:**

>>>

Traceback (most recent call last):

File "C:/Python27/t.py", line 2, in

data[0]=100

TypeError: 'tuple' object does not support item assignment

>>>

**Creating a new Tuple from existing:**

**Eg:**

1. data1=(10,20,30)
2. data2=(40,50,60)
3. data3=data1+data2
4. **print** data3

**Output:**

>>>

(10, 20, 30, 40, 50, 60)

>>>

**b) Deleting elements from Tuple:**

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

**Eg:**

1. data=(10,20,'rahul',40.6,'z')
2. **print** data
3. **del** data      #will delete the tuple data
4. **print** data  #will show an error since tuple data is already deleted

**Output:**

>>>

(10, 20, 'rahul', 40.6, 'z')

Traceback (most recent call last):

File "C:/Python27/t.py", line 4, in

print data

NameError: name 'data' is not defined

>>>

Functions of Tuple:

There are following in-built Type Functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| min(tuple) | Returns the minimum value from a tuple. |
| max(tuple) | Returns the maximum value from the tuple. |
| len(tuple) | Gives the length of a tuple |
| cmp(tuple1,tuple2) | Compares the two Tuples. |
| tuple(sequence) | Converts the sequence into tuple. |

**1) min(tuple):**

**Eg:**

1. data=(10,20,'rahul',40.6,'z')
2. **print** min(data)

**Output:**

>>>

10

>>>

**2) max(tuple):**

**Eg:**

1. data=(10,20,'rahul',40.6,'z')
2. **print** max(data)

**Output:**

>>>

z

>>>

**3) len(tuple):**

**Eg:**

1. data=(10,20,'rahul',40.6,'z')
2. **print** len(data)

**Output:**

>>>

5

>>>

**4) cmp(tuple1,tuple2):**

**Explanation:**If elements are of the same type, perform the comparison and return the result. If elements are different types, check whether they are numbers.

* If numbers, perform comparison.
* If either element is a number, then the other element is returned.
* Otherwise, types are sorted alphabetically .

If we reached the end of one of the lists, the longer list is "larger." If both list are same it returns 0.

**Eg:**

1. data1=(10,20,'rahul',40.6,'z')
2. data2=(20,30,'sachin',50.2)
3. **print** cmp(data1,data2)
4. **print** cmp(data2,data1)
5. data3=(20,30,'sachin',50.2)
6. **print** cmp(data2,data3)

**Output:**

>>>

-1

1

0

>>>

**5) tuple(sequence):**

**Eg:**

1. dat=[10,20,30,40]
2. data=tuple(dat)
3. **print** data

**Output:**

>>>

(10, 20, 30, 40)

>>>

Why Use Tuple?

1. Processing of Tuples are faster than Lists.
2. It makes the data safe as Tuples are immutable and hence cannot be changed.
3. Tuples are used for String formatting.