Python UNIT 4-part 3

Tuple

Tuple In Python

- A tuple is an immutable data type in python, almost similar to a list in python in terms of indexing and having duplicate members.
- It is a collection data type that stores python objects separated by commas.
- Following is an example of how we can create or declare a tuple in python.

```
1  #creating a tuple
2  a = ('python', 'edureka')
3  #another approach
4  b = 'python' , 'edureka'
5  print(a)
6  print(b)
```

```
Output: ('python' , 'edureka')
        ('python' , 'edureka')
```

1. Tuples are ordered collections of objects of arbitrary types. Collections of objects are ordered from left to right.
2. Tuples are classified as immutable sequences. Direct change operations cannot be applied to tuples. However, if the element of the tuple is a mutable element (for example, a list), then this element can be changed in the tuple.
3. Tuples are heterogeneous. The term “heterogeneous” means that tuples can contain other composite objects, such as lists, rows, or other tuples.
4. Tuples support an arbitrary number of attachments.
5. Tuples can be represented as arrays of references to objects.

Differences between tuples and lists

The following differences can be distinguished between tuples and lists:

- the tuple is formed in parentheses (), the list is formed in square brackets [];
- tuples refer to immutable sequences; lists refer to mutable sequences. It is impossible to apply operations to tuples that directly modify them. Such operations can be applied to lists;
- tuples have a fixed length (number of elements), lists have a variable length. If you need to increase the size of the tuple, you need to create a copy.

Accessing Items In A Tuple

Accessing items in a tuple works similar to a list, we can access elements in a list using indexes.

We can specify the index value and it will return the item stored at that particular index value.

Indexing

It is a data structure technique to effectively retrieve information from a data structure. In python, several data types support indexing like lists, string, etc.

For example, let’s just say we have a tuple with 5 natural numbers as members. So the indexing will start with the value 0 where 1 will be stored and it will go until the end of the tuple i.e 5 and the index value at 5 will be 4.

example:

```
1  a = ('edureka', 'python' , 'data structure' , 'collections')
2  print(a[1])
3  print(a[3])
```

```
Output: python
        collections
```

Negative Indexing

use negative indexing as well to access elements in a tuple or any other data type that supports indexing.

```
1  a = (1,2,3,4,5,6,7,8,9,10)
2  print(a[-4])
3  print(a[-1])
```

```
Output: 7
        10
```

Slicing

It is a technique in which we use the slicing operator ':' to get a range of elements from a tuple or any other data type that supports indexing for accessing elements.

```
1  a = (1,2,3,4,5,6,7,8,9,10)
2  print(a[1:8])
3  print(a[1:])
4  print(a[:5])
```

```
Output: (2,3,4,5,6,7,8)
        (2,3,4,5,6,7,8,9,10)
        (1,2,3,4,5)
```

Basic tuple operations

1. **len() :**

   The len() method returns the number of elements in the tuple.

```
tuple1, tuple2 = (123, 'xyz', 'zara'), (456, 'abc')
print ("First tuple length : ", len(tuple1))
print ("Second tuple length : ", len(tuple2))
```

output:

```
First tuple length :  3
Second tuple length :  2
```

2. Concatenating and Multiplying Tuples

- Concatenation is done with the + operator,
- and multiplication is done with the * operator.

```
coral = ('blue coral', 'staghorn coral', 'pillar coral', 'elkhorn coral')
kelp = ('wakame', 'alaria', 'deep-sea tangle', 'macrocystis')

coral_kelp = (coral + kelp)

print(coral_kelp)
```

Output

('blue coral', 'staghorn coral', 'pillar coral', 'elkhorn coral', 'wakame', 'alaria', 'deep-sea tangle', 'macrocystis')

The repetition operator makes multiple copies of a tuple and joins them all together.

Tuples can be created using the repetition operator, *****.

```
numbers = (0, 1, 2) * 3

print numbers
```

Output

This will give the output −

```
(0, 1, 2, 0, 1, 2, 0, 1, 2)
```

3. Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Other Tuple operations

Changing A Tuple

Even though tuples in python are immutable in nature, a nested object in a tuple can be changed. Or in general, a tuple in python can be reassigned with a different value.

```
1  a = (1,2,3,[4,5])
2  a[3][0] = 14
3  print(a)
4  #reassigning the value
5  a = ('edureka', 'python')
6  print(a)
```

```
Output: (1,2,3,[14,5])
        ('edureka', 'python')
```

Deleting A Tuple

Being an immutable data type, a tuple in python does not allow any changes and you cannot even remove an element from a tuple after the declaration. But there is a keyword ‘del’ which will delete the tuple altogether.

```
a = (1,2,3,4,5)

del a

print(a)
```

You will get a Name error if you run the above program because there is no tuple named as present since we have deleted it.

Tuple Methods

Following are the tuple methods that we can use while working with a tuple in python.

count: Returns the count of the items.
index: It returns the index of the item specified.

```
a = (1,2,1,3,1,3,1,2,1,4,1,5,1,5)
print(a.count(1))
print(a.index(5))
```

Output: 7
11

Iterating Through A Tuple

Using a for loop we can iterate through a tuple in python. The following example shows how we can iterate through a tuple using a for loop.

```
a = ("edureka", "for data science", "for Artificial Intelligence")
for i in a:
    print("python", i)
```

Output: python edureka
python for data science
python for artificial intelligence

Tuple Constructor

It is possible to create a tuple using a tuple() constructor as well. We can even use the tuple constructor to change a list to a tuple.

```
a = [1,2,3,4,5]
b = tuple(a)
print(b)
c = tuple(('edureka', 'python'))
print(c)
```

Output: (1,2,3,4,5)
('edureka', 'python')

Membership Test In A Tuple

Using the membership operator 'in' in python we can check whether an element is present in a tuple or not. The following example shows how we can check if an element is present in a tuple or not.

```
1  a = (1,2,3,4,5,6,7,8,9,10)
2
3  print(6 in a)
4
5  print(15 in a )
```

Output: True
False