TUPLES

Introduction: A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Each element of a sequence is assigned a number called index. The first element of the list index is zero, the second element index is one, and so forth.. Tuples are immutable, i.e., they cannot be altered once declared.

E.g.:a_tup = (1,2,"cat","dog",4)

Accessing Tuples: To access values from the tuples, index numbers are used . We can access tuples in indexing, slicing

Indexing tuples: Every element in a tuple is assigned with index number starting with zero .By indexing the value we can access the elements present in that particular index. Negative indexing can also be done in python. In negative indexing the last element index starts with -1 .

E.g.:

```
a_tup =(1,2,"cat","dog",4)
print (a_tup [1])   ## 2
print (a_tup [0])   ## 1
print (a_tup [-1])   ## 4
```

Slicing tuples: Slices work on tuples just as with strings, these can be used to change sub-parts of the tuple. Slicing of a can be done even in reverse order, here in reverse order index starts with -1 at the last element of thetuple ,-2 for second last element of the tuple , and so on.

E.g.::

```
b_tup = ('a', 'b', 'c', 'd')
print ( b_tup [1:-1])   ## ('b', 'c')
b_tup [0:2] = 'z'   ## replace ('a', 'b') with ('z')
print (b_tup)       ## ('z', 'c', 'd')
```

Operations: Tuple responds to concatenation (+) and repetition (*) operators.

Concatenation: Two tuples can be concatenated using +operator, but the concatenated value will be stored in a new tuple.

E.g.:

```
a_tup=(1,2,3,'a','b','c')
b_tup=(4,5,6,'d','e','e')
print(a_list+ b_list)   #(1,2,3,'a','b','c', 4,5,6,'d','e','e')
```

Repetition: Tuples repetetion can be done using * operator. Here elements in the Tuples can be repeated by * operators, and the result will be stored in an new tuple

E.g:

```
a=(1,2,3)
print a*4  ## (1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
```

Tuple Comprehension: Comprehensions is a tool, which creates a new dataset in a single and readable line. Suppose If we want to create tuple of integers from 0 to 10,in a single tuple we can use tuple comprehension but it return an generator object but not a tuple with elements as the tuple is immutable.

E.g.:
```
S = (x**2 for x in range(10))
      print (S)  # <generator object <genexpr> at 0x000001FBFDCF2830>
```

Functions and Methods: Python contains functions and methods to perform operations on tuples, as tuple is immutable. You cannot change elements of a tuple once it is assigned.

Len: Len finds the length of the tuple and returns the length of the tuple.

Sytax: len(tuple_name)

E.g:
```
a_tup=(1,2,3,4,"ad","sub',"hello",5,3,3)
len(a_tup) #10
```

max: finds the maximun element in the tuple and returns the maximum element in the tuple

Sytax: max(tuple_name)

E.g:
```
tup_max=(1,22,3,5,44,333,765,4215)
max(tup_max) #4215
```

min: finds the minimum element in the tuple and returns the minimum number in the tuple.

Sytax: min(tup_name)

E.g:
```
tup_min=(1,22,3,5,44,333,765,4215)
min(tup_max) #1
```

count: Counts the number of element present in the tuple and returns the count Vlue.

Sytax: tuple.count(element)

E.g:
```
vowels = ('a', 'e', 'i', 'o', 'i', 'o', 'e', 'i', 'u')
count = vowels.count('i')
```