Tuples

-> A tuple is similar to list

-> The diffence between the two is that we can't change the elements of tuple once it is assigned whereas in the list, elements can be changed

Tuple creation

In [ ]:

```
#empty tuple
t = ()

#tuple having integers
t = (1, 2, 3)
print(t)

#tuple with mixed datatypes
t = (1, 'veera', 35.0, 'abc')
print(t)

#nested tuple
t = (1, (2, 3, 4), [1, 'veera', 35, 'MG'])
print(t)
```

In [ ]:

```
#only parenthesis is not enough
t = ('veera',)
type(t)
```

In [ ]:

```
#need a comma at the end
t = ('veera',)
type(t)
```

In [ ]:

```
#parenthesis is optional
t = "veera",
print(type(t))

print(t)
```

Accessing Elements in Tuple

In [ ]:

```
t = ('Mentor', 'Cavium', 'Huawei', 'Sirveen')

print(t[1])
```

In [ ]:

```
#negative index
print(t[-1]) #print last element in a tuple
```

In [ ]:

```
#nested tuple
```

```
t = ('company', ('Mentor', 'Cavium', 'Huawei', 'Sirveen'))

print(t[1])
```

In [ ]:

```
print(t[1][2])
```

In [ ]:

```
#Slicing
t = (1, 2, 3, 4, 5, 6)

print(t[1:4])

#print elements from starting to 2nd last elements
print(t[:-2])

#print elements from starting to end
print(t[:])
```

Changing a Tuple

unlike lists, tuples are immutable

This means that elements of a tuple cannot be changed once it has been assigned. But, if the element is itself a mutable datatype like list, its nested items can be changed.

In [13]:

```
#creating tuple
t = (1, 2, 3, 4, [5, 6, 7])

#t[2] = 'x' #will get TypeError
```

In [14]:

```
t[4].append("MG")
print(t)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-14-a615cca1b497> in <module>()
----> 1 t.append("MG")
      2 print(t)

AttributeError: 'tuple' object has no attribute 'append'
```

In [15]:

```
#concatinating tuples

t = (1, 2, 3) + (4, 5, 6)
print(t)
```

```
(1, 2, 3, 4, 5, 6)
```

In [16]:

```
#repeat the elements in a tuple for a given number of times using the * operator.
t = (('salary', ) * 4)
print(t)
```

```
('salary', 'salary', 'salary', 'salary')
```

Tuple Deletion

In [ ]:

```
#we cannot change the elements in a tuple.
# That also means we cannot delete or remove items from a tuple.

#delete entire tuple using del keyword
t = (1, 2, 3, 4, 5, 6)

#delete entire tuple
del t
```

Tuple Count

In [17]:

```
t = (1, 2, 3, 1, 3, 3, 4, 1)

#get the frequency of particular element appears in a tuple
t.count(1)
```

Out[17]:

```
3
```

Tuple Index

In [ ]:

```
t = (1, 2, 3, 1, 3, 3, 4, 1)

print(t.index(3)) #return index of the first element is equal to 3

#print index of the 1
```

Tuple Memebership

In [ ]:

```
#test if an item exists in a tuple or not, using the keyword in.
t = (1, 2, 3, 4, 5, 6)

print(1 in t)
```

In [ ]:

```
print(7 in t)
```

Built in Functions

Tuple Length

In [ ]:

```
t = (1, 2, 3, 4, 5, 6)
print(len(t))
```

Tuple Sort

In [22]:

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

new_t = sorted(t)
print (type(new_t))
print(new_t) #Take elements in the tuple and return a new sorted list
             #(does not sort the tuple itself).
```

```
<class 'list'>
[1, 2, 3, 4, 5]
```

In [23]:

```
#get the largest element in a tuple
t = (2, 5, 1, 6, 9)

print(max(t))
```

```
9
```

In [24]:

```
#get the smallest element in a tuple
print(min(t))
```

```
1
```

In [25]:

```
#get sum of elments in the tuple
print(sum(t))
```

```
23
```