Tuples

- In Python programming, a tuple is similar to a list.
- The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas in a list, elements can be changed.

Advantages of Tuple over List ¶

- We generally use tuple for heterogeneous (different) datatypes and list for homogeneous (similar) datatypes.
- Since tuple are immutable, iterating through tuple is faster than with list. So there is a slight performance boost.
- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Tuple Creation

In [1]:
```
#empty tuple
t = ()

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

#tuple with mixed datatypes
t = (1, 'raju', 28, 'abc')
print(t)

#nested tuple
t = (1, (2, 3, 4), [1, 'raju', 28, 'abc'])
print(t)
```

```
(1, 2, 3)
(1, 'raju', 28, 'abc')
(1, (2, 3, 4), [1, 'raju', 28, 'abc'])
```

In [2]:
```
#only parenthesis is not enough
t = ('satish')
type(t)
```

Out[2]:
```
str
```

```
In [6]: #need a comma at the end
        t= ('satish',)
        type(t)

Out[6]: tuple
```

```
In [2]: #parenthesis is optional
        t = "satish",
        print(type(t))

        print(t)

        <class 'tuple'>
        ('satish',)
```

Accessing Elements in Tuple

```
In [14]: t = ('Python', 'Java', 'C', 'C++')

         print(t[1])

         Java
```

```
In [15]: #negative index
         print(t[-1]) #print last element in a tuple

         C++
```

```
In [16]: #nested tuple
         t = ('Language', ('Python', 'Django', 'Flask'))
         print(t[1])

         ('Python', 'Django', 'Flask')
```

```
In [17]: print(t[1][2])

         Flask
```

```
In [18]: #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[:])

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

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.**

</b>

In [19]:
```
#creating tuple
t = (1, 2, 3, 4, [5, 6, 7])

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-9f4cbf6ee0de> in <module>()
      2 t = (1, 2, 3, 4, [5, 6, 7])
      3
----> 4 t[2] = 'x' #will get TypeError

TypeError: 'tuple' object does not support item assignment
```

In [20]:
```
t[4][1] = 'Python'
print(t)
```

```
(1, 2, 3, 4, [5, 'Python', 7])
```

In [23]:
```
#concatinating tuples

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

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

In [24]:
```
#repeat the elements in a tuple for a given number of times using the * operat
or.
t = (('satish', ) * 4)
print(t)
```

```
('satish', 'satish', 'satish', 'satish')
```

Tuple Deletion

- **We cannot change the elements in a tuple.**
- **That also means we cannot delete or remove items from a tuple.**
- **We can delete entire tuple using del keyword**

```
In [28]: t = (1, 2, 3, 4, 5, 6)

         #delete entire tuple
         del t
```

Tuple Count

```
In [29]: t = (1, 2, 3, 1, 3, 3, 4, 1)

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

Out[29]: 3
```

Tuple Index

```
In [33]: 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 first occurence

         2
```

Tuple Memebership

```
In [34]: #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)

         True

In [35]: print(7 in t)

         False
```

Built in Functions

Tuple Length

In [36]:
```
t = (1, 2, 3, 4, 5, 6)
print(len(t))
```
```
6
```

Tuple Sort

In [37]:
```
t = (4, 5, 1, 2, 3)

new_t = sorted(t)
print(new_t) #Take elements in the tuple and return a new sorted list
             #(does not sort the tuple itself).
```
```
[1, 2, 3, 4, 5]
```

In [43]:
```
#get the largest element in a tuple
t = (2, 5, 1, 6, 9)

print(max(t))
```
```
9
```

In [44]:
```
#get the smallest element in a tuple
print(min(t))
```
```
1
```

In [40]:
```
#get sum of elments in the tuple
print(sum(t))
```
```
23
```