Tuple.

1. Excatly same as list expect it is immutable.
2. Tuple is read only version of List.
3. Data is fixed and never changes than we should go for tuple.
4. Insertion order is preserved.
5. Duplicates are allowed.
6. Heterogeneous objects are allowed.
7. We can preserve insertion order and differentiate duplicates by using index.
8. Index will play an important roll in tuple also.
9. Suports +ve and -ve index.
10. Represent tuple element with () paranthesis and with coma saperator.

2.1 Creation of tuple.

In [1]:
```
t=(10)
print(type(t))
```

```
<class 'int'>
```

2.2 Accessing elements of tuple.

1. By using Index.
2. By using slicing operator.

In [2]:
```
t=(10,20,30,0,50,60,80)
print(t[2:4])
print(t[2:100])
print(t[::2])
print(t[-1:-2])
print(t[-1:-4:-1])
```

```
(30, 0)
(30, 0, 50, 60, 80)
(10, 30, 50, 80)
()
(80, 60, 50)
```

2.3 Mathemetical Operators.

1. " + " ----> "Concatation"
2. " * " ----> "Repetation"

In [5]:
```
t1=(10,20,30)
t2=(40,50,60)
t=t1+t2
print(t)
tt=t1*3
print(tt)
```

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

2.4 Important Functions of tuple.

1. len()
2. count()
3. index()
4. sorted() ---> By default the answer in list.
5. min() and max()
6. revesed() ---> Compute reverse of a givan sequence object and return it in term of list.
7. enumerate()

In [9]:
```
t=(10,20,30,40,10)
print(len(t))
print(t.count(10))
print(t.count(50))
print(t.index(10))
```

```
print(t.index(40))
print(t.index(50))
```

```
5
2
0
0
3
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-4db4bde9f11f> in <module>
      5 print(t.index(10))
      6 print(t.index(40))
----> 7 print(t.index(50))

ValueError: tuple.index(x): x not in tuple
```

In [13]:
```
t=(15,22,10,50,30)
t1=sorted(t)
print(t1)
print("--------")
t2=sorted(t,reverse=True)
print(t2)
```

```
[10, 15, 22, 30, 50]
--------
[50, 30, 22, 15, 10]
```

In [14]:
```
t=("abc","ABC","def","DEF")
print(max(t))
print(min(t))
```

```
def
ABC
```

In [1]:
```
s="Python"
print(list(reversed(s)))
x=('p','y','t','h','o','n')
print(list(reversed(x))
y=range(5,9)
      print(list(reversed(y))
```

```
  File "<ipython-input-1-5cceecde8a54>", line 5
    y=range(5,9)
    ^
SyntaxError: invalid syntax
```

In [3]:
```
l1=("eat","sleep","walk")
print(list(enumerate(l1)))
print(list(enumerate(l1,10)))
```

```
[(0, 'eat'), (1, 'sleep'), (2, 'walk')]
[(10, 'eat'), (11, 'sleep'), (12, 'walk')]
```

2.5 Tuple packing.

In [15]:
```
a=10
b=20
c=30
d=40
t=a,b,c,d
print(t)
```

```
(10, 20, 30, 40)
```

2.6 Tuple unpacking.

In [16]:
```
t=(10,20,3,2)
a,b,c,d=t
```

```
print(a,b,c,d)
```

```
10 20 3 2
```

In [17]:
```
t=(12,34,45,56)
a,b,d,c,f,v=t
print(a,v,d,c,f)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-be9f7d7d79a5> in <module>
      1 t=(12,34,45,56)
----> 2 a,b,d,c,f,v=t
      3 print(a,v,d,c,f)

ValueError: not enough values to unpack (expected 6, got 4)
```

2.7 Loop through tuple.

In [24]:
```
t=("Apple","Banana","Cherry")
for i in t:
    print(i)
print("------------")
for i in range(len(t)):
    print(t[i])
print("------------")
i=0
while i!=len(t):
    print(t[i])
    i+=1
```

```
Apple
Banana
Cherry
------------
Apple
Banana
Cherry
------------
Apple
Banana
Cherry
```

In [ ]:
```
```