Tuple

1. Tuple is similar to List except that the objects in tuple are immutable which me
   elements of a tuple once assigned.
2. When we do not want to change the data over time, tuple is a preferred data t
3. Iterating over the elements of a tuple is faster compared to iterating over a list

In [1]:
```
t = (10, 20, 30)
t
```

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

In [2]:
```
len(t)
```

```
3
```

In [3]:
```
t1 = ('Asif', 25 ,[50, 100],[150, 90] , {'John' , 'David'} , (99,22,33))
t1
```

```
('Asif', 25, [50, 100], [150, 90], {'David', 'John'}, (99, 22, 33))
```

In [6]:
```
print(t1[0])
print(t1[2][0])
print(t1[-1])
```

```
Asif
50
(99, 22, 33)
```

Slicing

In [7]:
```
mytuple = ('one' , 'two' , 'three' , 'four' , 'five' , 'six' , 'seven' , 'e
mytuple[0:3]
```

```
('one', 'two', 'three')
```

In [8]:
```
mytuple[-3:]
```

```
('six', 'seven', 'eight')
```

In [9]:
```
for i in mytuple:
 print(i)
```

```
one
two
three
four
five
six
seven
eight
```

In [10]:
```
for i in enumerate(mytuple):
 print(i)

```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

Operations

count(1 arg), index(1 arg)

In [1]:
```
mytuple1 =('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
mytuple1.count('one') # Number of times item "one" occurred in the tuple.
```

```
3
```

In [13]:
```
mytuple1.index('two')  # first occurance will be display
```

```
1
```

Tuple Membership

In [15]:
```
'one' in mytuple1
```

```
True
```

In [16]:
```
mytuple2 = (43,67,99,12,6,90,67)
```

In [17]:
```
sorted(mytuple2)
```

```
[6, 12, 43, 67, 67, 90, 99]
```

In [18]:
```
sorted(mytuple2, reverse=True)
```

```
[99, 90, 67, 67, 43, 12, 6]
```

In [19]:
```
u = (9, 'p')
```

In [21]:
```
o = u * 2
o
```

```
(9, 'p', 9, 'p')
```

Range

3 argument

in python which is the most common data structure - range()
range() datatypes represent a sequence of values
always immutable
range() datatypes we have multiple forms lets see one by one
List insertation order is preserved but set insertation is not preserved

In [22]:
```
range(3)
```

```
range(0, 3)
```

In [26]:
```
e = range(10, 20)
e
list(e)
```

```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [27]:
```
f = range(0, 10, 3)
f
list(f)
```

```
[0, 3, 6, 9]
```

In [29]:
```
g = range(10, 100, 25)
list(g)
```

```
[10, 35, 60, 85]
```

Set

Difference between list & set --

duplicates are allowed, order - LIST
no dupliacates, & order also not important - SET
LIST are represent as [] && SET are represent as {}
SET object does not support indexing or slicing
insertation order are not preserved,order & duplicates are not allowed
index concept are not allowed , hetrogeneuous object are allowed
set is mutable
add & remove method is use for SET but append method is used only for LIST

In [1]:
```
s = {}
type(s)
```

```
dict
```

In [2]:
```
#Declare empty set
s1 = set()
type(s1)
```

```
set
```

1. Unordered & Unindexed collection of items, slicing is not allowed.
2. Set elements are unique. Duplicate elements are not allowed.
3. Set elements are immutable (cannot be changed).
4. Set itself is mutable. We can add or remove items from it.

In [3]:
```
my_set = {100,100,10,2,20,3,4,5,5}
my_set
```

```
{2, 3, 4, 5, 10, 20, 100}
```

In [4]:
```
s2 = {'x', 'k', 'e', 'y', 'z' }
s2
```

```
{'e', 'k', 'x', 'y', 'z'}
```

In [5]:
```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

```
{(11, 22, 32), 10, 20, 'Hola'}
```

Functions

add(1 arg), copy, clear, discard(1 arg), pop, remove(1 arg-value), update(str)

In [6]:
```
r = {}       #empty set is not changed
#r.add(90)
```

In [7]:
```
q = {22, 7, 8}
q.add(6)
q
```

```
{6, 7, 8, 22}
```

In [8]:
```
q.add(True)
q
```

```
{True, 6, 7, 8, 22}
```

In [9]:
```
q.add(1)
q
```

```
{True, 6, 7, 8, 22}
```

In [10]:
```
#q.remove(44)
q.discard(23)
```

In [11]:
```
q.pop()
q
```

```
{6, 7, 8, 22}
```

In [12]:
```
n = {'one','two','three'}
```

In [13]:
```
n.update('four')
n
```

```
{'f', 'o', 'one', 'r', 'three', 'two', 'u'}
```

In [14]:
```
n.update('10')
n
```

```
{'0', '1', 'f', 'o', 'one', 'r', 'three', 'two', 'u'}
```

In [15]:
```
del q
```

In [16]:
```
for i in my_set:
 print(i)
```

```
2
3
100
5
4
20
10
```

In [17]:
```
for i in enumerate(my_set):
 print(i)
```

```
(0, 2)
(1, 3)
(2, 100)
(3, 5)
(4, 4)
(5, 20)
(6, 10)
```

In [18]:
```
2 in my_set # Check if 'one' exist in the set
```

```
True
```

In [19]:
```
myset = {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [20]:
```
print('one' in myset)
print('ten' in myset)
```

```
True
False
```

Set Operation

Union, Intersection, intersection_update, Difference, difference_update, Symmetric
symmetric_difference_update, Subset , Superset & Disjoint

Union

In [21]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

In [22]:
```
print(A | B )   # Union
print(A.union(B, C))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [23]:
```
print(A | B & C)              #b&c is left to right
print(B.intersection(C))
```

```
{1, 2, 3, 4, 5, 8}
{8}
```

In [24]:
```
print(A - B)
print(A.difference(B))
```

```
{1, 2, 3}
{1, 2, 3}
```

In [25]:
```
print(A ^ B)
print(A.symmetric_difference(B))
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

In [26]:
```
B.difference_update(A)
B

```

```
{6, 7, 8}
```

Subset , Superset & Disjoint

In [27]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [28]:
```
print(B.issubset(A))
print(A.issuperset(B))
print(C.isdisjoint(A))
print(B.isdisjoint(A))
```

```
True
True
True
False
```

Other Builtin functions

sum, min, max, len, sorted

In [29]:
```
A = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [30]:
```
print(sum(A))
print(max(A))
print(min(A))
print(len(A))
print(list(enumerate(A)))
print(sorted(A,reverse=True))
```

```
45
9
1
9
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

frozen set

In [31]:
```
1 q = {False, True, 2, 3, 4, 10, 20}
```

In [32]:
```
1 fs = frozenset(q)
```