Sets

1. Unordered & Unindexed collection of items.
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.

Set Creation

In [33]:
```
a = set()  #Empty set
print(type(a))
```

```
<class 'set'>
```

In [9]:
```
myset={1,2,3,4,5}
myset
```

Out[9]:
```
{1, 2, 3, 4, 5}
```

In [13]:
```
len(myset)
```

Out[13]:
```
5
```

Duplicate elements are not allowed

In [15]:
```
myset1={1,1,1,2,3,4,5}
myset1
```

Out[15]:
```
{1, 2, 3, 4, 5}
```

In [18]:
```
myset2={1.79,2.08,3.99,4.56,5.45} # Set of float numbers
myset2
```

Out[18]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [20]:
```
myset3 = {'Tanishq' , 'abc' , 'xyz'} # Set of Strings
myset3
```

Out[20]:
```
{'Tanishq', 'abc', 'xyz'}
```

In [22]:
```
myset4 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset4
```

Out[22]:
```
{(11, 22, 32), 10, 20, 'Hola'}
```

set doesn't allow mutable items like lists

Sets iteslf are mutable but you cannot add other mutable items inside it

In [26]:
```
myset5 = {10,20, "Hola", [11, 22, 32]}
myset5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 myset5 = {10,20, "Hola", [11, 22, 32]}
      2 myset5

TypeError: unhashable type: 'list'
```

Loop through a Set

In [43]:
```
newset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
newset
```

Out[43]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [45]:
```
for i in newset:
    print (i)
```

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

In [47]:
```
for i in enumerate(newset):
    print (i)
```

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

Set Membership

Check if element exist in the set

In [51]:
```
newset
```

Out[51]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [55]:
```
'one' in newset  # Check if 'one' exist in the set
```

Out[55]: True

In [63]:
```
'ten' in newset # Check if 'ten' exist in the set
```

Out[63]: False

In [67]:
```
if 'three' in newset: # Check if 'three' exist in the set
    print('Three is present in the set')
else:
    print('Three is not present in the set')
```

Three is present in the set

Add & Remove Items

In [71]:
```
newset
```

Out[71]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

Add a item to a set using add() method

In [73]:
```
newset.add('Tanishq')
newset
```

Out[73]: {'Tanishq', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

Add multiple items to a set using update([list])

In [76]:
```
newset.update(['abc','xyz','plm'])
newset
```

Out[76]:
```
{'Tanishq',
 'abc',
 'eight',
 'five',
 'four',
 'one',
 'plm',
 'seven',
 'six',
 'three',
 'two',
 'xyz'}
```

remove item from a set using discard() method

In [85]:
```
newset.discard('Tanishq')
newset
```

Out[85]:
```
{'abc',
 'eight',
 'five',
 'four',
 'one',
 'plm',
 'seven',
 'six',
 'three',
 'two',
 'xyz'}
```

Delete all items in a set

In [88]:
```
newset.clear()
newset
```

Out[88]:
```
set()
```

Delete the set object

In [91]:
```
del newset
newset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[91], line 2
      1 del newset
----> 2 newset

NameError: name 'newset' is not defined
```

Copy Set

In [95]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

Out[95]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [99]:
```
MYset = myset # Create a new reference "MYset1"
MYset
```

Out[99]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [101…
```
id(MYset) , id(myset1) # The address of both myset & myset1 will be the same
```

Out[101…
```
(2315836379136, 2315806619264)
```

In [103…
```
MYset=myset.copy()
MYset
```

Out[103…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [105…
```
id(MYset) , id(myset1) # The address of both myset & myset1 will be different
```

Out[105…
```
(2315836382720, 2315806619264)
```

Both the sets are different

myset.add(1245632645) myset

In [109…
```
MYset
```

Out[109…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

.pop()

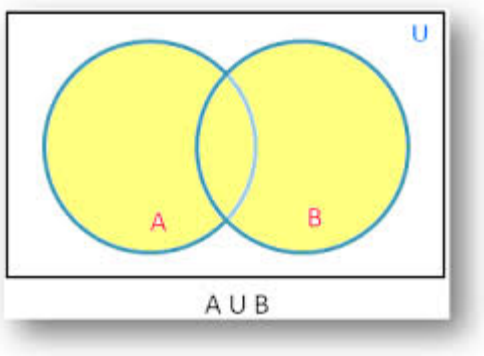
pops random elements out not like list/tuples' last element

In [121…
```
MYset.pop()
MYset
```

Out[121…
```
{'eight', 'five', 'four', 'one', 'six', 'three', 'two'}
```

Set Operation

1. Union



In [185…
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

In [155…
```
A | B # Union of A and B (All elements from both sets. NO DUPLICATES)
```

Out[155… {1, 2, 3, 4, 5, 6, 7, 8}

In [175…
```
A.union(B) # Union of A and B
```

Out[175… {1, 2, 3, 4, 5, 6, 7, 8}

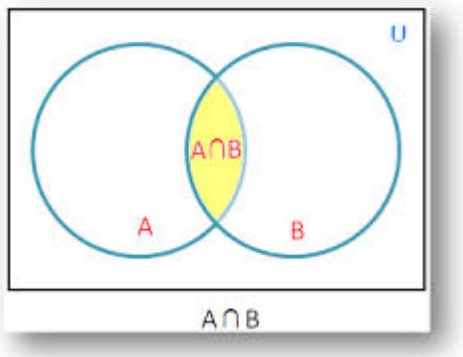
In [159…
```
A.union(B,C) # Union of A, B and C.
```

Out[159… {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [161…
```
A | B | C
```

Out[161… {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

2. Intersection



In [189…
```
A = {1,2,3,4,5}
B = {1,2,4,5,6,7,8}
C = {8,9,10,7,2}
```

In [191…
```
A & B  # Intersection of A and B (Common items in both sets)
```

Out[191… {1, 2, 4, 5}

In [193…
```
A.intersection(B)
```

Out[193… {1, 2, 4, 5}

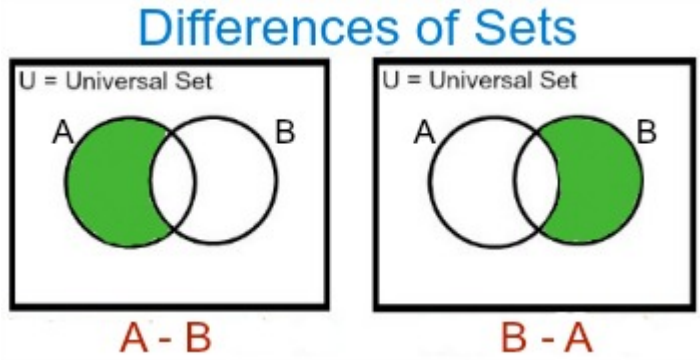
In [195…
```
A & B & C
```

Out[195… {2}

In [197…
```
A.intersection(B,C)
```

Out[197… {2}

3. Difference



In [204…
```
A = {1,2,3,4,5}
B = {4,5,6,7,}
```

In [212…
```
A - B   # set of elements that are only in A but not in B
#Refer first fig
```

Out[212…
```
{1, 2, 3}
```

In [218…
```
A.difference(B)
```

Out[218…
```
{1, 2, 3}
```

Simillarly

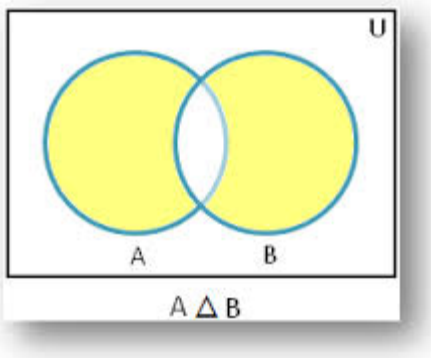
In [216…
```
B - A # set of elements that are only in B but not in A
```

Out[216…
```
{6, 7, 8}
```

In [224…
```
B.difference(A)
```

Out[224…
```
{6, 7, 8}
```

4. Symmetric Difference



In [234…
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {4,5,9,10,11,12,13}
```

In [238…
```
A ^ B  #All elements in both sets except common ones
```

Out[238…
```
{1, 2, 3, 6, 7, 8}
```

In [240…
```
A.symmetric_difference(B)
```

Out[240…
```
{1, 2, 3, 6, 7, 8}
```

In [242…
```
A^B^C
```

Out[242…
```
{1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13}
```

Superset, Subset, Disjoint

1. Superset --> Parent
2. Subset --> Child
3. Disjoint --> No relation

In [248…
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [252…
```
B.issubset(A)  #All elements of B are in A
```

Out[252…
```
True
```

In [256…
```
A.issuperset(B)  #All elements of B are in A
```

Out[256…
```
True
```

In [260…
```
C.isdisjoint(A)  #No elements of A & C are common
```

Out[260…
```
True
```

In [262…
```
B.isdisjoint(A)
```

Out[262…
```
False
```

Other built-in functions

In [266…
```
A
```

Out[266… {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [270…
```
sum(A)
```

Out[270… 45

In [272…
```
max(A)
```

Out[272… 9

In [274…
```
min(A)
```

Out[274… 1

In [276…
```
len(A)
```

Out[276… 9

In [278…
```
list(enumerate(A))
```

Out[278… [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [282…
```
x = sorted(A,reverse=True)
x
```

Out[282… [9, 8, 7, 6, 5, 4, 3, 2, 1]

In [284…
```
sorted(x)
```

Out[284… [1, 2, 3, 4, 5, 6, 7, 8, 9]