TUPLE

```
In [45]: t=()
         t

Out[45]: ()

In [46]: type(t)

Out[46]: tuple

In [47]: t=(10,20,30)
         t

Out[47]: (10, 20, 30)

In [48]: t.count(10)

Out[48]: 1

In [49]: t.count(30)

Out[49]: 1

In [50]: t1=(10,20,2.2,'arun',True,(1+2j),20)
         t1

Out[50]: (10, 20, 2.2, 'arun', True, (1+2j), 20)

In [51]: t1.count(20)

Out[51]: 2

In [52]: t1.index(10)

Out[52]: 0

In [53]: t1.index(20)

Out[53]: 1

In [54]: print(t)
         print(t1)

        (10, 20, 30)
        (10, 20, 2.2, 'arun', True, (1+2j), 20)

In [55]: print(len(t))
         print(len(t1))

        3
        7

In [56]: t
```

```
Out[56]:  (10, 20, 30)
```

```
In [57]:  t[0]
```

```
Out[57]:  10
```

```
In [58]:  t[0]=100  #in tuple we can't change value so tuple is immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[58], line 1
----> 1 t[0]=100  #in tuple we can't change value so tuple is immutable

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

```
In [59]:  bank_account=(345,'aks45k',100000)
          bank_account
```

```
Out[59]:  (345, 'aks45k', 100000)
```

```
In [ ]:   bank_account[2]=200000  # it'
          s an error bcz in tuple we can't assign value or we cant  change
```

```
In [60]:  t
```

```
Out[60]:  (10, 20, 30)
```

```
In [ ]:   t*3 # it repeat the tuple 3 time
```

```
In [61]:  t
```

```
Out[61]:  (10, 20, 30)
```

```
In [62]:  for i in t:
              print(i)
```

```
10
20
30
```

```
In [63]:  for i in enumerate(t):
              print(i)
```

```
(0, 10)
(1, 20)
(2, 30)
```

    list

LIST .MUTABLE (WE CAN CHANGE) .DUPLICATE IS ALLOWED .Append(),copy(),insert(),extend(),pop(),remove() .list is growable .multiple data type in a list .indexing & slicing is allowed

tuple

TUPLE .IMMUTABLE(UNCHANGEABLE) .DUPLICATION IS ALLOWED .REMOVE IS NOT ALLOWED .ONLY 2 FUNCTION WILL WORK(.index(),.count())

set

```
In [64]: s={}
         s
```

```
Out[64]: {}
```

```
In [65]: type(s)
```

```
Out[65]: dict
```

```
In [67]: s1=set()
         s1
```

```
Out[67]: set()
```

```
In [68]: type(s1)
```

```
Out[68]: set
```

```
In [69]: s2={90,10,50,40,25,10,50}  #remove dublicate and also auto short the int value
         s2
```

```
Out[69]: {10, 25, 40, 50, 90}
```

```
In [70]: type(s2)
```

```
Out[70]: set
```

```
In [ ]: s2
```

.copy()

```
In [71]: s3=s2.copy() # copy is allowed
         s3
```

```
Out[71]: {10, 25, 40, 50, 90}
```

```
In [72]: s3
```

```
Out[72]: {10, 25, 40, 50, 90}
```

```
    .add()  #we can add int,string,bool,complex
```

```
In [73]: s3.add(3.4)
         s3
```

```
Out[73]: {3.4, 10, 25, 40, 50, 90}
```

```
In [74]: s3.add(35)
```

```
s3
```

```
Out[74]:  {3.4, 10, 25, 35, 40, 50, 90}
```

```
In [ ]:  s3.add('arun')
         s3
```

```
In [75]: s3.add(1+2j)
         s3
```

```
Out[75]:  {(1+2j), 10, 25, 3.4, 35, 40, 50, 90}
```

```
In [76]: s3.add(True)
         s3
```

```
Out[76]:  {(1+2j), 10, 25, 3.4, 35, 40, 50, 90, True}
```

```
In [77]: s3.add('kumar')
         s3
```

```
Out[77]:  {(1+2j), 10, 25, 3.4, 35, 40, 50, 90, True, 'kumar'}
```

```
In [78]: s3.add(1,2) # shows error bcz in set we can add exactly one argument
         s3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[78], line 1
----> 1 s3.add(1,2) # shows error bcz in set we can add exactly one argument
      2 s3

TypeError: set.add() takes exactly one argument (2 given)
```

```
In [79]: s3
```

```
Out[79]:  {(1+2j), 10, 25, 3.4, 35, 40, 50, 90, True, 'kumar'}
```

```
In [80]: print(s)
         print(s1)
         print(s2)
         print(s3)
```

```
{}
set()
{50, 90, 40, 25, 10}
{True, 3.4, (1+2j), 10, 25, 90, 'kumar', 35, 40, 50}
```

```
In [81]: type(s)
```

```
Out[81]:  dict
```

```
In [ ]:  type(s1)
```

```
In [82]: s3
```

```
Out[82]:  {(1+2j), 10, 25, 3.4, 35, 40, 50, 90, True, 'kumar'}
```

```
In [83]: s3.remove((1+2J))
         s3

Out[83]: {10, 25, 3.4, 35, 40, 50, 90, True, 'kumar'}

In [ ]: s3.remove(200)
        s3

In [84]: s3

Out[84]: {10, 25, 3.4, 35, 40, 50, 90, True, 'kumar'}

In [85]: s3.discard(10)
         s3

Out[85]: {25, 3.4, 35, 40, 50, 90, True, 'kumar'}

In [ ]: s3.discard(200)
        s3

In [86]: s3.pop()

Out[86]: True

In [87]: s3

Out[87]: {25, 3.4, 35, 40, 50, 90, 'kumar'}

In [88]: s3.pop()
         s3

Out[88]: {25, 35, 40, 50, 90, 'kumar'}

In [89]: s3.pop()

Out[89]: 25

In [90]: s3

Out[90]: {35, 40, 50, 90, 'kumar'}
```

TypeError: set.pop() takes no arguments

```
In [91]: s3.pop(0)
         s3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[91], line 1
----> 1 s3.pop(0)
      2 s3

TypeError: set.pop() takes no arguments (1 given)
```

```
In [ ]:     # in set indexing and slicing are not allow
```

```
In [93]: s3[:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[93], line 1
----> 1 s3[:]

TypeError: 'set' object is not subscriptable
```

```
In [94]: s3
```

```
Out[94]: {35, 40, 50, 90, 'kumar'}
```

```
In [95]: 40 in s3
```

```
Out[95]: True
```

```
In [96]: 'kumar' in s3
```

```
Out[96]: True
```

set operation

union

```
In [97]: a={1,2,3,4,5}
         b={4,5,6,7,8,}
         c={8,9,10}
         a,b,c
```

```
Out[97]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

```
In [98]: a.union(b)
```

```
Out[98]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [100… a.union(b,c)
```

```
Out[100… {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
In [101… print(a)
         print(b)
         print(c)
```

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

```
In [102… a|b
```

```
Out[102… {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [103… b|c
```

Out[103… {4, 5, 6, 7, 8, 9, 10}

In [104…
```
a|b|c
```
Out[104…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [105…
```
a|c
```
Out[105…
```
{1, 2, 3, 4, 5, 8, 9, 10}
```

In [106…
```
a|c|b
```
Out[106…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

intersection

In [108…
```
a={1,2,3,4,5}
b={4,5,6,7,8,}
c={8,9,10}
a,b,c
```
Out[108…
```
({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

In [109…
```
a.intersection(b)
```
Out[109…
```
{4, 5}
```

In [110…
```
b.intersection(c)
```
Out[110…
```
{8}
```

In [111…
```
a&b
```
Out[111…
```
{4, 5}
```

In [112…
```
b&c
```
Out[112…
```
{8}
```

Difference

In [113…
```
a={1,2,3,4,5}
b={4,5,6,7,8,}
c={8,9,10}
a,b,c
```
Out[113…
```
({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

In [114…
```
a.difference(b)
```
Out[114…
```
{1, 2, 3}
```

```
In [115…  b.difference(a)

Out[115…  {6, 7, 8}

In [116…  b-c

Out[116…  {4, 5, 6, 7}

In [117…  c-b

Out[117…  {9, 10}

In [118…  a-b-c

Out[118…  {1, 2, 3}
```

more practices on Tuple slicing

```
In [2]:  mytuple=('one','two','three','four','five','six','seven','eight')
         mytuple

Out[2]:  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [3]:  type(mytuple)

Out[3]:  tuple

In [4]:  mytuple[0:2]

Out[4]:  ('one', 'two')

In [5]:  mytuple[0:3]

Out[5]:  ('one', 'two', 'three')

In [6]:  mytuple[2:5]

Out[6]:  ('three', 'four', 'five')

In [8]:  mytuple

Out[8]:  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [9]:  mytuple[:3]

Out[9]:  ('one', 'two', 'three')

In [10]:  mytuple[3:]

Out[10]:  ('four', 'five', 'six', 'seven', 'eight')

In [11]:  mytuple[-3:]

Out[11]:  ('six', 'seven', 'eight')
```

```
In [12]: mytuple

Out[12]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [13]: mytuple[:-3]

Out[13]: ('one', 'two', 'three', 'four', 'five')

In [14]: mytuple[-1]

Out[14]: 'eight'

In [15]: mytuple[:]

Out[15]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [16]: mytuple[::-1]

Out[16]: ('eight', 'seven', 'six', 'five', 'four', 'three', 'two', 'one')

In [17]: mytuple[::2]

Out[17]: ('one', 'three', 'five', 'seven')

In [18]: mytuple

Out[18]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [19]: mytuple[2::]

Out[19]: ('three', 'four', 'five', 'six', 'seven', 'eight')

In [20]: mytuple[2::2]

Out[20]: ('three', 'five', 'seven')

In [21]: mytuple[::-2]

Out[21]: ('eight', 'six', 'four', 'two')

In [22]: mytuple[2::-2]

Out[22]: ('three', 'one')

In [ ]:     # Loop through a tuple

In [23]: mytuple

Out[23]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [24]: for i in mytuple:
             print(i)
```

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

```
In [26]: for i in enumerate(mytuple):
             print(i)
```

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

Tuple membership

```
In [27]: mytuple
```

```
Out[27]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

```
In [28]: 'one' in mytuple
```

```
Out[28]: True
```

```
In [29]: 'five' in mytuple
```

```
Out[29]: True
```

```
In [31]: 'ten' in mytuple
```

```
Out[31]: False
```

```
In [33]: if 'four' in mytuple:
             print('four is present in the tuple')
         else:
             print('four is not present in the tuple')
```

```
four is present in the tuple
```

```
In [34]: if 'nine' in mytuple:
             print('nine is present in the tuple')
         else:
             print('nine is not present in the tuple')
```

```
nine is not present in the tuple
```

index position tuple

```
In [35]: mytuple
```

```
Out[35]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

```
In [36]: mytuple[0]
```

```
Out[36]: 'one'
```

```
In [42]: mytuple
```

```
Out[42]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

```
In [44]: mytuple.count('one')
```

```
Out[44]: 1
```

```
In [45]: mytuple.index('one')
```

```
Out[45]: 0
```

```
In [46]: mytuple[1]
```

```
Out[46]: 'two'
```

```
In [47]: mytuple[1][0]
```

```
Out[47]: 't'
```

```
In [49]: mytuple[-1]
```

```
Out[49]: 'eight'
```

sorting in tuple

```
In [50]: mytuple
```

```
Out[50]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

```
In [52]: t=(45,65,67,34,21,87,88,98,23,43)
         t
```

```
Out[52]: (45, 65, 67, 34, 21, 87, 88, 98, 23, 43)
```

```
In [53]: sorted(t)
```

```
Out[53]: [21, 23, 34, 43, 45, 65, 67, 87, 88, 98]
```

```
In [54]: sorted(t,reverse=True)
```

```
Out[54]: [98, 88, 87, 67, 65, 45, 43, 34, 23, 21]
```

Sets

1)unordered & unindexed collection of items 2)Set element are unique.Duplicate elements are not allowed 3)set itself is mutable.we can add or remove items from it. 4) but Set element are immutable

Set creation

```
In [1]:  myset={1,2,3,4,5}  # set of number
         myset
```

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

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

```
Out[2]:  5
```

```
In [4]:  my_set={1,2,3,4,5,2,22,3,3,4,4} #duplicate element are not allowed
         my_set
```

```
Out[4]:  {1, 2, 3, 4, 5, 22}
```

```
In [6]:  myset1={1.1,2.2,3.
                 4}
         myset1              # set of float number
```

```
Out[6]:  {1.1, 2.2, 3, 4}
```

```
In [7]:  myset2={'arun','kumar','sahu',}
         myset2                            #set of string
```

```
Out[7]:  {'arun', 'kumar', 'sahu'}
```

```
In [8]:  myset3={1,'one',12.34,(10,20)}
         myset3                          # set of mixed data type
```

```
Out[8]:  {(10, 20), 1, 12.34, 'one'}
```

```
In [9]:  len(myset3)
```

```
Out[9]:  4
```

```
In [11]:  myset4={10,20,"arun",[10,20,30]}# set does't allowed mutable item like list
          myset4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 myset4={10,20,"arun",[10,20,30]}
      2 myset4

TypeError: cannot use 'list' as a set element (unhashable type: 'list')
```

Loop through a set

```
In [12]: myset={'one','two','three','four','five','six'}
         for i in myset:
             print(i)
```

```
one
six
five
four
two
three
```

```
In [13]: for i in enumerate(myset):
             print(i)
```

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

set membership

```
In [14]: myset

Out[14]: {'five', 'four', 'one', 'six', 'three', 'two'}
```

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

Out[15]: True
```

```
In [16]: if 'three' in myset:
             print('Three is present in the set')
         else:
             print('Three is not present in the set')
```

```
Three is present in the set
```

```
      #    Add and remove item
```

```
In [18]: myset

Out[18]: {'five', 'four', 'one', 'six', 'three', 'two'}
```

```
In [20]: myset.add('nine') #Add item to a set using add() method
         myset

Out[20]: {'five', 'four', 'nine', 'one', 'six', 'three', 'two'}
```

```
In [21]: myset

Out[21]: {'five', 'four', 'nine', 'one', 'six', 'three', 'two'}
```

```
In [26]: myset.update(['TEN' ,'ELEVEN' ,'TWELVE'])
         myset
```

```
Out[26]: {'ELEVEN',
          'TEN',
          'TWELVE',
          'e',
          'eleven',
          'five',
          'four',
          'l',
          'n',
          'nine',
          'one',
          'six',
          't',
          'ten',
          'three',
          'twelve',
          'two',
          'v',
          'w'}
```

```
In [27]: myset.remove('one')
         myset
```

```
Out[27]: {'ELEVEN',
          'TEN',
          'TWELVE',
          'e',
          'eleven',
          'five',
          'four',
          'l',
          'n',
          'nine',
          'six',
          't',
          'ten',
          'three',
          'twelve',
          'two',
          'v',
          'w'}
```

```
In [28]: myset.discard('ten')
         myset
```

```
Out[28]: {'ELEVEN',
          'TEN',
          'TWELVE',
          'e',
          'eleven',
          'five',
          'four',
          'l',
          'n',
          'nine',
          'six',
          't',
          'three',
          'twelve',
          'two',
          'v',
          'w'}
```

copy set

```
In [29]: s={1,2,3,4,5,6}
         s
```

```
Out[29]: {1, 2, 3, 4, 5, 6}
```

```
In [30]: s1=s
         s1
```

```
Out[30]: {1, 2, 3, 4, 5, 6}
```

```
In [31]: id(s),id(s1)
```

```
Out[31]: (1666936090720, 1666936090720)
```

```
In [32]: s3=s.copy()
         s3
```

```
Out[32]: {1, 2, 3, 4, 5, 6}
```

```
In [33]: id(s3)   # id will be different
```

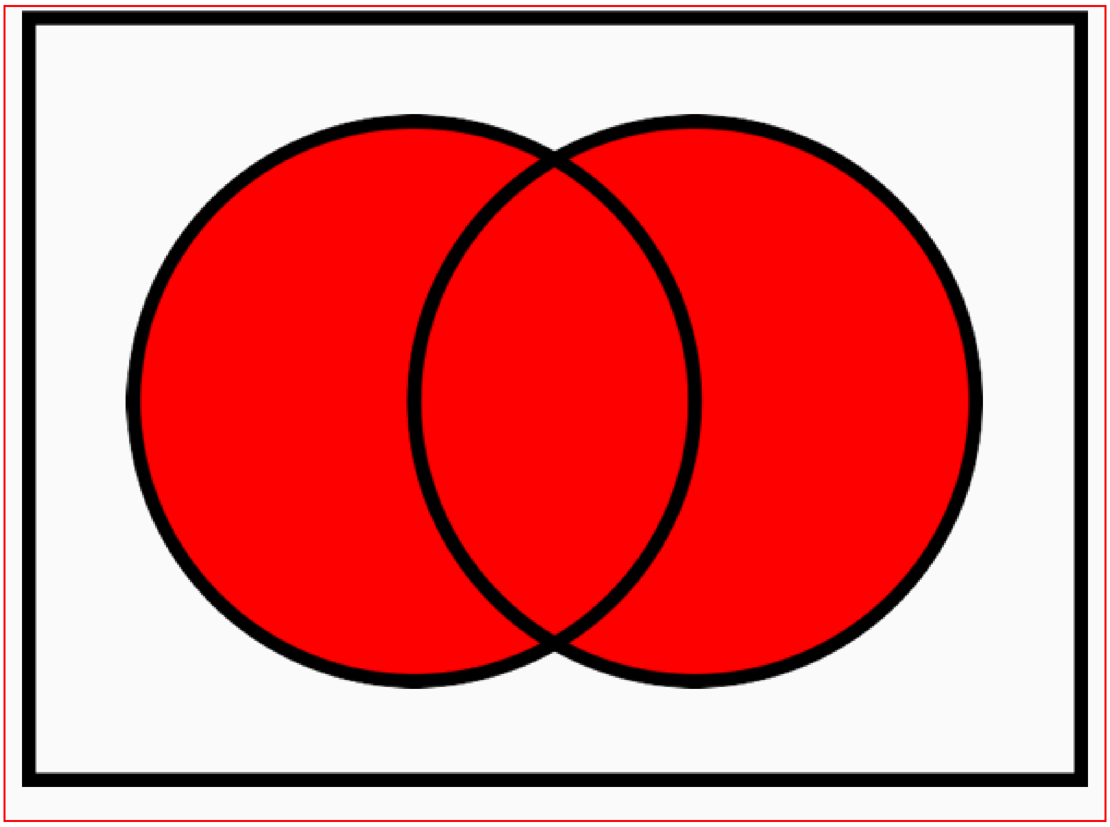
```
Out[33]: 1666914016544
```

```
In [34]: s.add(9)
         s
```

```
Out[34]: {1, 2, 3, 4, 5, 6, 9}
```

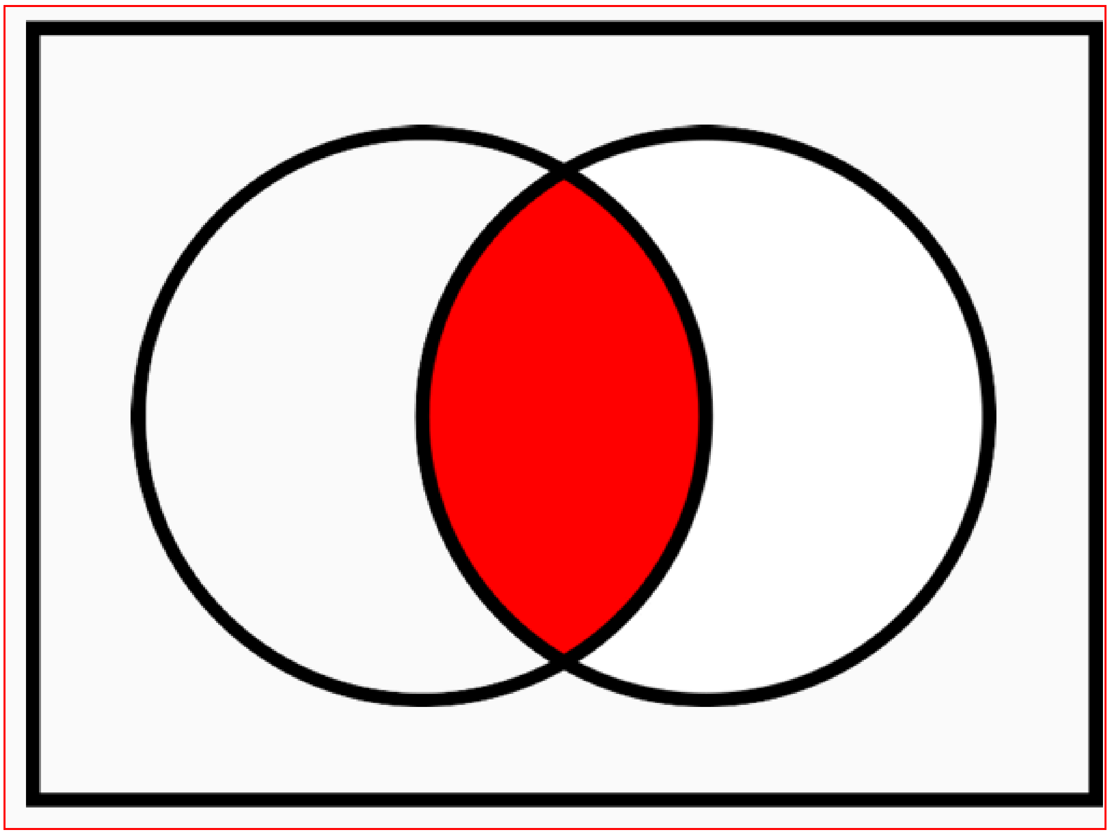
set operation

union .... symbol===|



```
In [35]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C

Out[35]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})

In [36]: A|B

Out[36]: {1, 2, 3, 4, 5, 6, 7, 8}

In [37]: B|C

Out[37]: {4, 5, 6, 7, 8, 9, 10}

In [38]: A|B

Out[38]: {1, 2, 3, 4, 5, 6, 7, 8}

In [39]: A|B|C

Out[39]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [41]: A.union(B)

Out[41]: {1, 2, 3, 4, 5, 6, 7, 8}
```

INTERSECTION ==== SYMBOL===&



```
In [42]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C

Out[42]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})

In [43]: A&B

Out[43]: {4, 5}

In [44]: B&C

Out[44]: {8}

In [45]: A&B&C

Out[45]: set()

In [46]: A.intersection(B)

Out[46]: {4, 5}
```

DIFFERENCE======= -

```
In [72]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C
```

```
Out[72]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

```
In [73]: A-B
```

```
Out[73]: {1, 2, 3}
```

```
In [74]: B-C
```

```
Out[74]: {4, 5, 6, 7}
```

```
In [75]: A-B-C
```

```
Out[75]: {1, 2, 3}
```

```
In [76]: B-A
```

```
Out[76]: {6, 7, 8}
```

.update()

```
In [77]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C
```

```
Out[77]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

```
In [78]: A.update(B)
         A
```

```
Out[78]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [79]: B.update(C)
         B
```

```
Out[79]: {4, 5, 6, 7, 8, 9, 10}
```

intersection_update()

```
In [80]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C
```

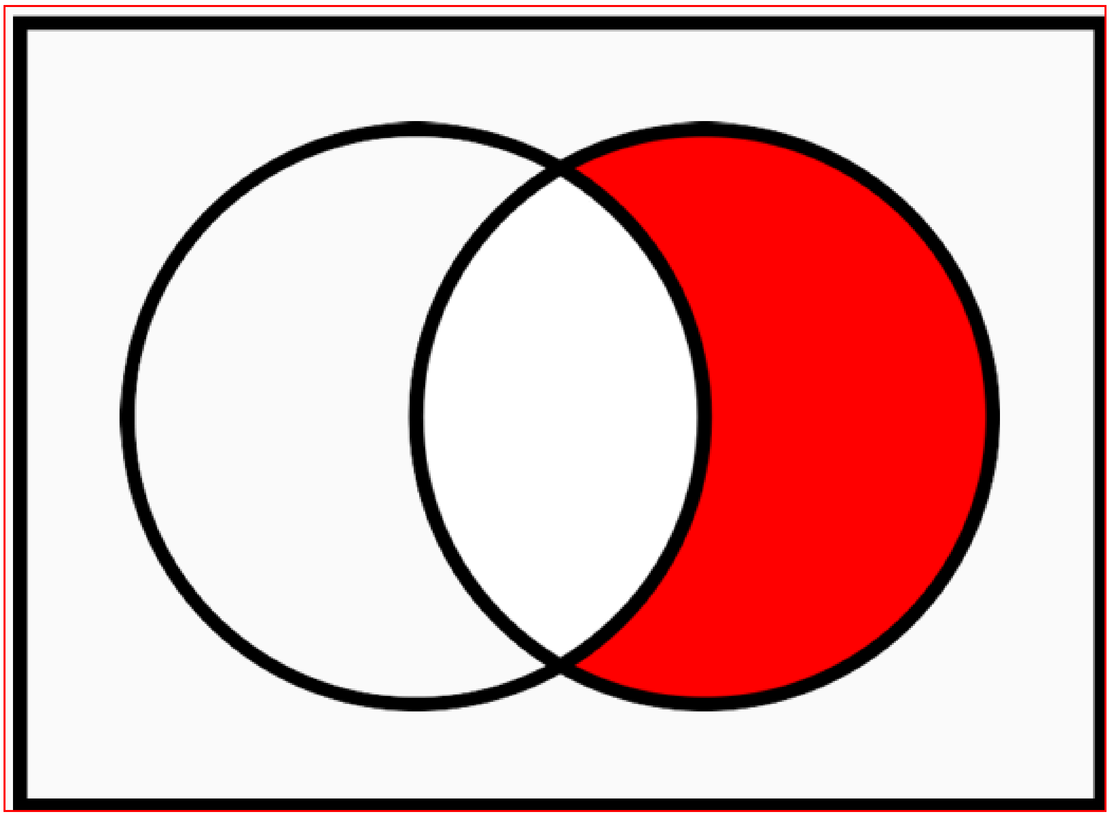
```
Out[80]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})
```

```
In [81]: A.intersection_update(B)
         A
```

```
Out[81]: {4, 5}
```

difference_update()

""" Updates the set calling the difference_update() method with the difference of se For below example Set B will be updated with the difference of B & A. """
B.difference_update(A) B



```
In [61]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C

Out[61]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})

In [64]: A={1,2,3,4,5}
         B={4,5,6,7,8}
         C={8,9,10}
         A,B,C

Out[64]: ({1, 2, 3, 4, 5}, {4, 5, 6, 7, 8}, {8, 9, 10})

In [82]: B.difference_update(A)
         B

Out[82]: {6, 7, 8}
```

Symmetric Difference====^

```
In [97]: A={1,2,3,4,5}
         B={4,5,6,7,8}
```

```
In [98]:  A.symmetric_difference(B)

Out[98]:  {1, 2, 3, 6, 7, 8}

In [102…  A^B # ELEMENT OF A AND B BUT NOT IN BOTH.

Out[102…  {1, 2, 3, 6, 7, 8}

In [103…  A.symmetric_difference(B)

Out[103…  {1, 2, 3, 6, 7, 8}

In [105…  A.symmetric_difference_update(B)
          A

Out[105…  {1, 2, 3, 6, 7, 8}

In [106…  A

Out[106…  {1, 2, 3, 6, 7, 8}
```

SUBSET,SUPERSET,& DISJOINT

```
In [1]:   a={1,2,3,4,5,6,7,8,9}
          b={3,4,5,6,7,8}
          c={10,20,30,40,50}

In [3]:   b.issubset(a) # set b is said to be the subset of a if all the element of b are

Out[3]:   True

In [4]:   a.issuperset(b)

Out[4]:   True

In [5]:   c.isdisjoint(a)

Out[5]:   True

In [6]:   b.isdisjoint(a)

Out[6]:   False

In [7]:   a

Out[7]:   {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [8]:   sum(a)

Out[8]:   45

In [9]:   max(a)
```

```
Out[9]:  9
```

```
In [10]: min(a)
```

```
Out[10]: 1
```

```
In [11]: len(a)
```

```
Out[11]: 9
```

```
In [12]: list(enumerate(a))
```

```
Out[12]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

```
In [14]: d=sorted(a,reverse=True)
         d
```

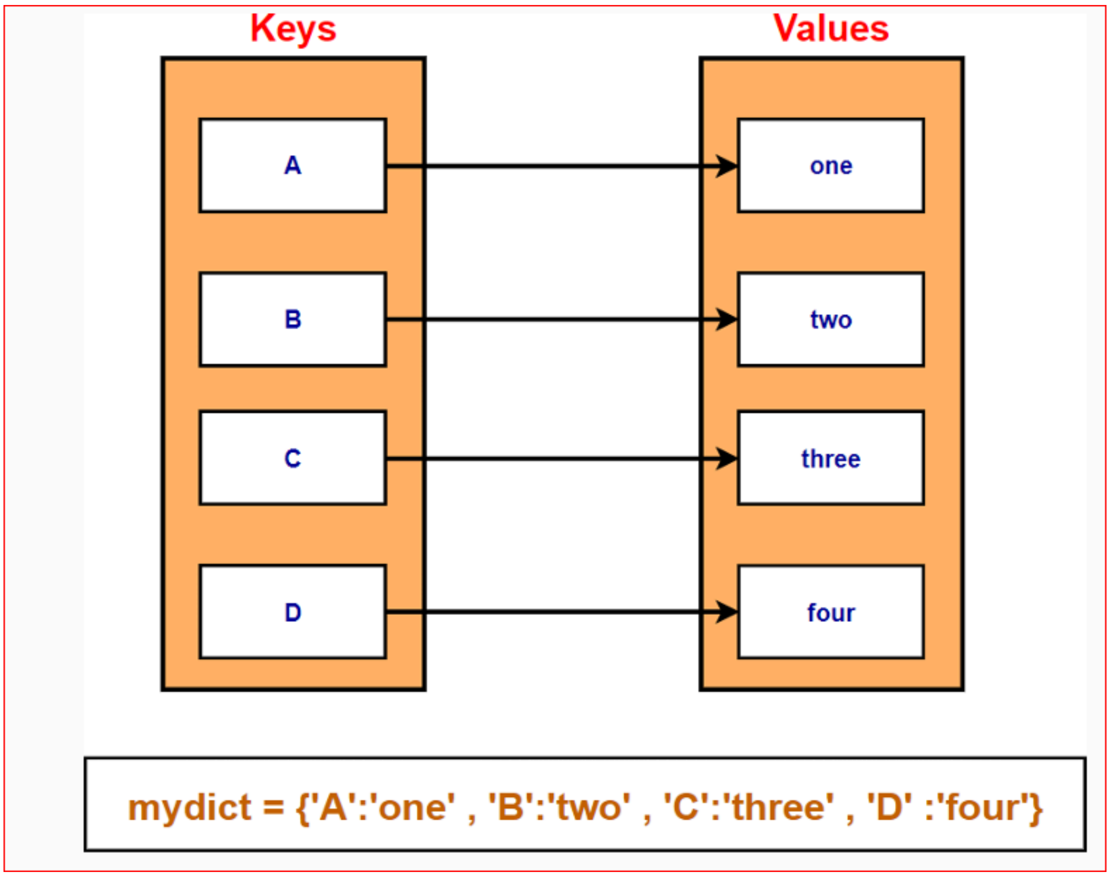
```
Out[14]: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

```
In [15]: sorted(d)
```

```
Out[15]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

DICTIONARY

.Dictionary is a mutable data type in python. .A python dictionary is a collection of key and value pairs seperated by a colon (:)& enclosed in a curley braces {}. .Keys must be unique in a dictionary,duplicate value are allowed


Keys
Values
A
one
B
two
C
three
D
four
mydict = {'A':'one' , 'B':'two' , 'C':'three' , 'D' :'four'}

CREATE DICTIONARY

```
In [16]: mydict=dict()
         mydict   # empty dictionary

Out[16]: {}

In [17]: mydict={1:'one',2:'two',3:'three'}
         mydict           # dictionary with integer keys

Out[17]: {1: 'one', 2: 'two', 3: 'three'}

In [23]: mydict= dict({1:'one',2:'two',3:'three'})  #create dictionary using dict()
         mydict

Out[23]: {1: 'one', 2: 'two', 3: 'three'}

In [26]: mydict={'a':'one','b':'two','c':'three'} #dict with character keys
         mydict

Out[26]: {'a': 'one', 'b': 'two', 'c': 'three'}

In [27]: mydict={1:'one','a':'two',3:'three'}  # Dict wiyh mixed keys
         mydict

Out[27]: {1: 'one', 'a': 'two', 3: 'three'}

In [28]: mydict.keys()# return Dict keys using keys() method

Out[28]: dict_keys([1, 'a', 3])

In [29]: mydict.values()  #return Dictionary values using values() method

Out[29]: dict_values(['one', 'two', 'three'])

In [31]: mydict.items() #Acess each key-value pair withnin a dictionary

Out[31]: dict_items([(1, 'one'), ('a', 'two'), (3, 'three')])

In [32]: mydict={1:'one',2:'two','A':['ARUN','kumar','sahu']}# dict with List
         mydict

Out[32]: {1: 'one', 2: 'two', 'A': ['ARUN', 'kumar', 'sahu']}

In [33]: dict={1:'one','a':'two','A':['shreeansh','sidhi'],'B':(1,2,3)}
         dict

Out[33]: {1: 'one', 'a': 'two', 'A': ['shreeansh', 'sidhi'], 'B': (1, 2, 3)}

In [34]: keys={'a','b','c','d'}
         mydict1=dict.fromkeys(keys)
         mydict1

Out[34]: {'c': None, 'a': None, 'd': None, 'b': None}
```

```
In [37]: keys={'a','b','c','d'}
         value=10
         mydict2=dict.fromkeys(keys,value)
         mydict2

Out[37]: {'c': 10, 'a': 10, 'd': 10, 'b': 10}

In [39]: keys={'a','b','c','d'}
         value=[10,20,30,40]
         mydict3=dict.fromkeys(keys,value)
         mydict3

Out[39]: {'c': [10, 20, 30, 40],
          'a': [10, 20, 30, 40],
          'd': [10, 20, 30, 40],
          'b': [10, 20, 30, 40]}

In [41]: value.append(50)
         mydict3

Out[41]: {'c': [10, 20, 30, 40, 50, 50],
          'a': [10, 20, 30, 40, 50, 50],
          'd': [10, 20, 30, 40, 50, 50],
          'b': [10, 20, 30, 40, 50, 50]}
```

Accessing items

```
In [42]: mydict={1:'one',2:'two',3:'three',4:'four'}
         mydict

Out[42]: {1: 'one', 2: 'two', 3: 'three', 4: 'four'}

In [43]: mydict[1]

Out[43]: 'one'

In [45]: mydict.get(1)

Out[45]: 'one'

In [46]: mydict.get(4)

Out[46]: 'four'

In [47]: mydict[2]

Out[47]: 'two'

 In [2]: mydict1={'Name':'arun','id':26,'dob':2000,'job':'datascience'}
         mydict1

 Out[2]: {'Name': 'arun', 'id': 26, 'dob': 2000, 'job': 'datascience'}

 In [3]: mydict1['Name']

 Out[3]: 'arun'
```

```
In [4]: mydict1.get('job')
```

```
Out[4]: 'datascience'
```

add remove change item

```
In [5]: d={'name':'arun','id':26,'dob':2000,'address':'delhi'}
        d
```

```
Out[5]: {'name': 'arun', 'id': 26, 'dob': 2000, 'address': 'delhi'}
```

```
In [7]: d['dob']=2002           #changing dictinory items
        d['address']='odisha'
        d
```

```
Out[7]: {'name': 'arun',
         'id': 26,
         'dob': 2002,
         'address': 'odisha',
         'adress': 'odisha'}
```

```
In [10]: d['job']='data science' #Adding item in the dictionary
         d
```

```
Out[10]: {'name': 'arun',
          'id': 26,
          'dob': 2002,
          'address': 'odisha',
          'adress': 'odisha',
          'job': 'data science'}
```

```
In [11]: d.pop('job')# removing item using pop methode
```

```
Out[11]: 'data science'
```

```
In [12]: d
```

```
Out[12]: {'name': 'arun',
          'id': 26,
          'dob': 2002,
          'address': 'odisha',
          'adress': 'odisha'}
```

```
In [14]: d.popitem()   # A random item is removed
```

```
Out[14]: ('adress', 'odisha')
```

```
In [15]: d
```

```
Out[15]: {'name': 'arun', 'id': 26, 'dob': 2002, 'address': 'odisha'}
```

```
In [16]: del[d['id']]  # removing item using del methode
         d
```

```
Out[16]: {'name': 'arun', 'dob': 2002, 'address': 'odisha'}
```

```
In [17]: d.clear()
         d          #delete all item of the dict using clear method

Out[17]: {}
```

copy dictionary

```
In [20]: d= {'Name':'piku' , 'ID': 45 , 'DOB': 1998 , 'Address' : 'dpl'}
         d

Out[20]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [22]: d1=d
         d1

Out[22]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [23]: id(d),id(d1)

Out[23]: (1452956321088, 1452956321088)

In [24]: d2=d.copy()
         d2

Out[24]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [25]: id(d),id(d2)

Out[25]: (1452956321088, 1452941499008)

In [26]: d

Out[26]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [27]: d2

Out[27]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [ ]: All/any

In [28]: d

Out[28]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [29]: all(d)

Out[29]: True

In [30]: d

Out[30]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl'}

In [31]: d1={'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl','c':0}
         d1
```

```
Out[31]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl', 'c': 0}
```

```
In [32]: all(d1)
```

```
Out[32]: True
```

```
In [34]: d2={'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl',False:2}
         d2
```

```
Out[34]: {'Name': 'piku', 'ID': 45, 'DOB': 1998, 'Address': 'dpl', False: 2}
```

```
In [35]: all(d2)
```

```
Out[35]: False
```

```
In [36]: any(d)
```

```
Out[36]: True
```

```
In [37]: any(d2)
```

```
Out[37]: True
```

```
In [ ]: The all() method returns:
        True - If all all keys of the dictionary are true
        False - If any key of the dictionary is false
        The any() function returns True if any key of the dictionary is True. If not, ar
```