sets

In [6]:
```
#create a set
empty_set=set() #empty set with only set() ortherwise it will be an empty di
print(empty_set)
set_name = {"bharu", "lovely", "bbp","rasna", 12345, 100.00, "bharu"}
print(set_name)
```

```
set()
{100.0, 'bharu', 'rasna', 'bbp', 12345, 'lovely'}
```

In [3]:
```
set1= {("bbp","angry"), "1234"}
print(set1)
```

```
{'1234', ('bbp', 'angry')}
```

In [4]:
```
set2=set((['1',34,"bbp"]))
print(set2)
```

```
{34, 'bbp', '1'}
```

In [8]:
```
# set elements are immutable,so a list cannot be a member of a set
x=set()
x={"abc",[1,2,3,4]}
print(x)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-1b4e510c6274> in <module>
      1 # set elements are immutable, so a listor dictionary cannot be a member
of a set
      2 x=set()
----> 3 x={"abc",[1,2,3,4]}
      4 y={'a':"apple", "b":"Ball"}
      5 print({y})

TypeError: unhashable type: 'list'
```

In [9]:
```
y={'a':"apple", "b":"Ball"} # dictionary cannot be a member of a set
print({y})
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-3c54dc1ad202> in <module>
      1 y={'a':"apple", "b":"Ball"}
----> 2 print({y})

TypeError: unhashable type: 'dict'
```

In [11]:
```
# in operator to check whether an ele is present in set or not
print("1" in set2)
```

```
True
```

Operations

In [15]:
```
# Union operator is '|' and method is union() no duplicates all the elements
a={"abcd",12,"efgh"}
b={"efgh",13,56}
print(a.union(b))
print(a|b)
print(a.union(('bbp','lovely',123)))
print(a|('bbp','lovely',123))
```

```
{'efgh', 56, 'abcd', 12, 13}
{'efgh', 56, 'abcd', 12, 13}
{12, 'efgh', 'bbp', 'abcd', 123, 'lovely'}

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-07afca3d345a> in <module>
      5 print(a|b)
      6 print(a.union(('bbp','lovely',123)))
----> 7 print(a|('bbp','lovely',123))

TypeError: unsupported operand type(s) for |: 'set' and 'tuple'
```

In [17]:
```
#Intersection '&' , intersection() common elements
a={"abcd",12,"efgh"}
b={"efgh",13,56}
print(a&b)
print(a.intersection(b))
```

```
{'efgh'}
{'efgh'}
```

In [20]:
```
#difference '-' , method is difference() only first with no common elements
p={'a',1,'b',3,'cd'}
q={'cd','a',1,13}
print(p-q)
print(q-p)
```

```
{3, 'b'}
{13}
```

In [24]:
```
#symmetric difference all elements with no common elements
p={'a',1,'b',3,'cd'}
q={'cd','a',1,13}
r={'ef','b',78,13}
print(p^q)
print(p^q^r)
print(p.symmetric_difference(q))
print(p.symmetric_difference(q,r)) #method takes only one argument
```

```
{3, 13, 'b'}
{3, 78, 'ef'}
{3, 13, 'b'}

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-9cf72f85c63d> in <module>
      6 print(p^q^r)
      7 print(p.symmetric_difference(q))
----> 8 print(p.symmetric_difference(q,r))

TypeError: symmetric_difference() takes exactly one argument (2 given)
```

In [27]:
```
#to find any common elements or not disjoint returns boolean
p={'a',1,'b',3,'cd'}
q={'cd','a',1,13}
r={'x',9,'y'}
print(p.isdisjoint(q))
print(p.isdisjoint(r))
```

```
False
True
```

In [33]:
```
#issubset '<=' every element of first shoulb be in second
s={'a','b'}
print(r <= q)
print(q <= p)
print(s <= p)
print(p.issubset(p))
```

```
False
False
True
True
```

In [37]:
```
# a proper subset (sets should not be identical )
i={'a',1,'b',2,'c',3}
j={'a',1,'b',2,'c',3,'d',4}
k={'a',1,'b',2,'c',3}
print(i < j) # i is a properset of j
print(i < k) # i and k are identical
```

```
True
False
```

In [40]:
```
# a superset every element of second should be present in first
print(j >= k)
print(k >= j)
print(k <= j)
```

```
True
False
True
```

In [43]:
```
#a proper superset , a set itself is not a proper superset of itself
print(i>j)
print(j>i)
```

```
False
True
```

Modifications cannot modify individual elements but can modify the set

In [44]:
```
# adding elements using the add()
u={'a','b','c','d',1,2,3}
u.add(4)
print(u)
```

```
{1, 2, 3, 4, 'c', 'a', 'b', 'd'}
```

In [46]:
```
#using update to add elements
v={'p'}
v.update(['q','r'])
print(v)
```

```
{'r', 'q', 'p'}
```

In [48]:
```
#remove throws an exception if ele not found
u.remove('d')
print(u)
```

```
{1, 2, 3, 4, 'c', 'a', 'b'}
```

In [53]:
```
#pop , removes the ele and returns the popped element and throws an error
u.pop()
#w={}
#w.pop()
print(u)
```

```
{'a', 'b'}
```

In [58]:
```
#discard
w={'a'}
w.pop()
print(w)
w.discard(1)
```

```
set()
```

In [59]:
```
#clear is used to clear
u.clear()
print(u)

```

```
set()
```

modifications using operations

In [63]:
```
#update intersection updates the first set with intersection of 1 and 2
one={'a','b','c','d'}
two={'c','d'}
two.intersection_update(one)
print(two)
```

```
{'c', 'd'}
```

In [68]:
```
#difference update, updates the first set with difference of 1-2
one={'a','b','c','d'}
print(id(one))
two={'c','d'}
one.difference_update(two)
print(one)
print(id(one))
```

```
97989576
{'b', 'a'}
97989576
```

In [69]:
```
#symmetric_update, updates the
three={'a','b','p','q','r'}
four={'i','j','a','b','p'}
five=set()
#five.symmetric_difference_update(three)
five = three ^ four
print(five)
```

```
{'q', 'i', 'r', 'j'}
```

Frozen sets

In [75]:
```
#a frozen set is an immutable set
f1=frozenset(['angry','bird','bhargavi',13])
print(f1)
print(id(f1))
```

```
frozenset({'angry', 'bhargavi', 13, 'bird'})
97991592
```

In [74]:
```
f2=set((1,2,3))
f1^=f2    # the id of f1 changes here since, a new object is created for f1
print(f1)
print(id(f1))
```

```
frozenset({'angry', 'bird', 13, 'bhargavi'})
97991816
```

In [77]:
```
#if we want to pass sets as dictionary key we can use frozen sets
d1=frozenset('dsgf')
print(d1)
d2=frozenset({'abcd'})
print(d2)
```

```
frozenset({'s', 'g', 'f', 'd'})
frozenset({'abcd'})
```

In [ ]:
```

```