```
Set: {at least one element}
    it is collection of unique elements
    it is unordered type DS
    indexing is not possible on set
    it is mutable type -- insert/update/delete
```

```
p = [23,83,8,3,8,'Python','Java',5600,3000,'Hadoop',100]
```

```
p
```

```
[23, 83, 8, 3, 8, 'Python', 'Java', 5600, 3000, 'Hadoop', 100]
```

```
p1 = (23,83,8,3,8,'Python','Java',5600,3000,'Hadoop',100)
```

```
p1
```

```
(23, 83, 8, 3, 8, 'Python', 'Java', 5600, 3000, 'Hadoop', 100)
```

```
t1 = {1,}
```

```
type(t1)
```

```
set
```

```
py_set = {23, 83, 8, 3, 8, 'Python', 'Java', 5600, 3000, 'Hadoop', 100}
```

```
py_set
```

```
{100, 23, 3, 3000, 5600, 8, 83, 'Hadoop', 'Java', 'Python'}
```

```
list -- []
tuple --()
dict -- {}
str - '',"  ",''' ''' """ """
```

```
p = []
```

```
type(p)
```

```
list
```

```
t = ()
```

```
type(t)
```

```
tuple
```

```
#how to create empty set
```

```
symbolic method
p = []
t = ()
g = {}  - dict
pt = ' '



by using method name
```

```
p1 = list()
```

```
type(p1)
```

```
list
```

```
p1
```

```
[]
```

```
p2 = tuple()
```

```
p2
```

```
()
```

```
p3 = dict()
```

```
p3
```

```
{}
```

```
t2 = set()
```

```
t2
```

```
set()
```

```
t2
```

set()

```
uniary
operation on single set
```

```
py_set ={345,76,34,75,3,3,3,33,"Python",'Java'}
```

```
py_set
```

```
{3, 33, 34, 345, 75, 76, 'Java', 'Python'}
```

```
insertion:
    add()
    upate()


deletion



updatation
```

```
#add(element):
it is use to insert single element in set
```

```
py_set.add(750)
```

```
py_set
```

```
{3, 33, 34, 345, 75, 750, 76, 'Java', 'Python'}
```

```
set --{ elements}
```

```
py_set[5]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [30], in <cell line: 1>()
----> 1 py_set[5]

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

```
py_set
```

```
{3, 33, 34, 345, 75, 750, 76, 'Java', 'Python'}
```

```
py_set.add('Hadoop')
```

```
py_set
```

```
{3, 33, 34, 345, 75, 750, 76, 'Hadoop', 'Java', 'Python'}
```

```
py_set.update({100,300,5006,2040340,560606})
```

```
py_set
```

```
{100,
 2040340,
 3,
 300,
 33,
 34,
 345,
 5006,
 560606,
 75,
 750,
 76,
 'Hadoop',
 'Java',
 'Python'}
```

```
insertion:
    add() -- insert single element
    update() -- insert multiple element
```

```
#deletion operation:
```

```
a = {1,2,3}
```

```
a.update({100,200,300})
```

```
a
```

```
{1, 2, 3, 100, 200, 300}
```

```
a.update({'Python'})
```

```
a
```

```
{1, 100, 2, 200, 3, 300, 400, 'Python'}
```

```
#Deletion:

remove  ---
discard
pop
clear
```

```
#remove

if element then delete if not then return KeyError
Hard delete --> Error
```

```
py_set.remove(100)
```

```
py_set.remove(2040340)
```

```
py_set
```

```
{3, 300, 33, 34, 345, 5006, 560606, 75, 750, 76, 'Hadoop', 'Java', 'Python'}
```

```
py_set.remove('test')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [48], in <cell line: 1>()
----> 1 py_set.remove('test')

KeyError: 'test'
```

```
#dicard

if element then delete if not then skip (it will not return any error)
soft delete
```

```
py_set
```

```
{3, 300, 33, 34, 345, 5006, 560606, 75, 750, 76, 'Hadoop', 'Java', 'Python'}
```

```
py_set.discard('Hadoop')
```

```
py_set
```

```
{3, 300, 33, 34, 345, 5006, 560606, 75, 750, 76, 'Java', 'Python'}
```

```
py_set.discard(750)
```

```
py_set
```

```
{3, 300, 33, 34, 345, 5006, 560606, 75, 76, 'Java', 'Python'}
```

```
py_set.discard('test')
```

```
product name --

invalid -- delete -- nahi --> process -- discard -- soft delete

invalid --- dalete nahi --> stop process -- remove -- Hard delete
```

```
error nahi --> soft delete
error hai -- hard delete
```

```
#pop --> randomly delete
```

```

```

```
py_set.pop()
```

```
33
```

```
py_set
```

```
{3, 300, 34, 345, 5006, 560606, 75, 76, 'Java', 'Python'}
```

```
py_set.pop()
```

```
34
```

```
py_set.pop()
```

```
3
```

```
py_set.pop() #
```

```
560606
```

```
py_set
```

```
set()
```

```
p = {3, 300, 34, 345, 5006, 560606, 75, 76, 'Java', 'Python'}
```

```
p
```

```
{3, 300, 34, 345, 5006, 560606, 75, 76, 'Java', 'Python'}
```

```
p.pop(34)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [71], in <cell line: 1>()
----> 1 p.pop(34)

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

```
p.clear()
```

```
p
```

```
set()
```

```
p
```

```
set()
```

```
#union
```

```
loan = {'keshav','madhav','suresh','naresh','pranita','sudeep'}
```

```
credit = {'madhav','naresh','justin','swapnil'}
```

```
len(loan.union(credit))
```

```
8
```

```
#we want customer who have taken loan as well as credit
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
credit
```

```
{'justin', 'madhav', 'naresh', 'swapnil'}
```

```
loan.intersection(credit)
```

```
{'madhav', 'naresh'}
```

```
#loan not in credit
```

```
loan.difference(credit)
```

```
{'keshav', 'pranita', 'sudeep', 'suresh'}
```

```
credit.difference(loan)
```

```
{'justin', 'swapnil'}
```

```
credit.intersection(loan)
```

```
{'madhav', 'naresh'}
```

```
t1 = {1,34,6,3}
```

```
t2 = {34,67,2,6}
```

```
t1.intersection(t2)
```

```
{6, 34}
```

```
t1.difference(t1.intersection(t2))
```

```
{1, 3}
```

```
union
intersection
difference

symmetric diff

common element drop
```

```
loan.symmetric_difference(credit)
```

```
{'justin', 'keshav', 'pranita', 'sudeep', 'suresh', 'swapnil'}
```

```
education_loan = {'sonali','keshav'}
```

```
loan.intersection(credit,education_loan)
```

```
{'keshav'}
```

```
credit = {'justin', 'madhav', 'naresh', 'swapnil','sonali','keshav'}
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
loan.intersection(credit,education_loan)
```

```
{'keshav'}
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
test = credit.difference(education_loan)
```

```
education_loan
```

```
{'keshav', 'sonali'}
```

```
loan.difference(credit,education_loan)
```

```
{'pranita', 'sudeep', 'suresh'}
```

```
loan.difference(credit,education_loan)
```

```
{'pranita', 'sudeep', 'suresh'}
```

```
test
```

```
{'justin', 'madhav', 'naresh', 'swapnil'}
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
test = credit.difference(education_loan)
```

```
test
```

```
{'justin', 'madhav', 'naresh', 'swapnil'}
```

```
loan.difference(test)
```

```
{'keshav', 'pranita', 'sudeep', 'suresh'}
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
loan.difference(credit,education_loan)
```

```
{'pranita', 'sudeep', 'suresh'}
```

```
check and let you know
```

```

```

```
loan.intersection(credit,education_loan)
```

```
set()
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
credit
```

```
{'justin', 'madhav', 'naresh', 'swapnil'}
```

```
education_loan
```

```
{'keshav', 'sonali'}
```

```
loan.difference(credit,education_loan)
```

```
{'pranita', 'sudeep', 'suresh'}
```

```
loan
```

```
{'keshav', 'madhav', 'naresh', 'pranita', 'sudeep', 'suresh'}
```

```
credit
```

```
{'justin', 'madhav', 'naresh', 'swapnil'}
```

```
education_loan
```

```
{'keshav', 'sonali'}
```

```
a = {1,2,3}

b= {1,7,9}

c = {5,8,4}
```

```
a.intersection(b,c)
```

set()

```
a ---> b
a ---> c
```

```

```

```
a.
```

{1, 2, 3}

```
a
```

{1, 2, 3}

```
b
```

{1, 7, 9}

```
test = a.intersection(b)
```

```
a
```

{1, 2, 3}

```
b
```

{1, 7, 9}

```
test
```

{1}

```
variable save -->


memory impact

code

memory optimization --> time complexity


lines --- memory
```

```

```

```
insertsection + update
```

```
a
```

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

```
b
```

```
{1, 7, 9}
```

```
a.intersection_update(b)
```

```
a
```

```
{1}
```

```
b
```

```
{1, 7, 9}
```

```
a = {23,456,834,86,354}
```

```
b = {354,78,23,57}
```

```
a.difference(b)
```

```
{86, 456, 834}
```

```
a.difference_update(b)
```

```
a
```

```
{86, 456, 834}
```

```
Note:

    Python function/method ---> is start ---> True/False
```

```

```

```
p = {45,23,75,34,12,34}
```

```
q = {45,75}
```

```
p.issubset(q)
```

```
False
```

```
q.issubset(p)
```

```
True
```

```
p.issuperset(q)
```

```
True
```

```
relation --> bet p & q
```

```
p
```

```
{12, 23, 34, 45, 75}
```

```
q
```

```
{45, 75}
```

```
z = {'a','p','Java',12}
```

```
p.isdisjoint(z)
```

```
False
```

```
new_copy = p.copy()
```

```
new_copy
```

```
{12, 23, 34, 45, 75}
```

```
p
```

```
{12, 23, 34, 45, 75}
```

```
Interview : -

    shallow copy
    deep copy
```

```

```

```
data :

    original changes
```

```
p
```

```
{12, 23, 34, 45, 75}
```

```
new_copy
```

```
{12, 23, 34, 45, 75}
```

```

```

```
t = p
```

```
t
```

```
{12, 23, 34, 45, 75}
```

```
p
```

```
{12, 23, 34, 45, 75}
```

```
t.add(950)
```

```
t
```

```
{12, 23, 34, 45, 75, 950}
```

```
p
```

```
{12, 23, 34, 45, 75, 950}
```

```
a = 100+10
```

```
b = 100
```

```
c = 100
```

```
id(a)
```

```
2675485988304
```

```
id(b)
```

```
2675485988304
```

```
id(c)
```

```
2675485988304
```

```
p
```

```
{12, 23, 34, 45, 75, 950}
```

```
t
```

```
{12, 23, 34, 45, 75, 950}
```

```
id(p)
```

```
2675603687680
```

```
id(t)
```

```
2675603687680
```

```
a
```

```
110
```

```
a = 100
```

```
id(a)
```

```
2675485988304
```

```
a = 500
```

```
id(a)
```

```
2675613758928
```

```
a = 100
```

```
id(a)
```

```
2675485988304
```

```
z = 100
```

```
id(z)
```

```
2675485988304
```

```
100 = a
```

```
  Input In [77]
    100 = a
    ^
SyntaxError: cannot assign to literal
```

```
p
```

```
{12, 23, 34, 45, 75, 950}
```

```
new_copy
```

```
{12, 23, 34, 45, 75}
```

```
backup --> shallow copy
new copy create but reference will there
```

```
original -- deep copy
new copy create but no reference will be there
```

```

```

```
p = {1,2,3}
```

```
tr = p.copy()
```

```
tr
```

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

```
tr.add(500)
```

```
tr
```

```
{1, 2, 3, 500}
```

```
p
```

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

```
```

```
```

```
a = 950
```

```
id(a)
```

```
2675614451088
```

```
PVM --- Python Virtual Machine
```