Sets :-

In python set is the collection of the unordered items.

Each element in the set must be unique and sets remove the duplicate elements from its

Sets is mutable in nature that means we can modify the 'sets' after its creation.

In python sets haven't index value like lists and tuples i.e :- we cannot directly access any element of the set by the index value.

Adding elements in sets :-

for adding value in previously created sets we need to use add(). function

for ex:-

```
fruits = {"apple", "mango", "grapes"}
print (fruits)
fruits. add ("banana")
print (fruits)
```

O/P :-

```
~~apple~~ {"grapes", "mango", "apple"}
{"banana", "grapes", "mango", "apple"}
```

update () → This function is also uses to add items but firstly it split each character of items ~~into~~ seprately and than it gives the each character seprately and avoid or remove the duplicate character.
for ex:- if we use update ("kiwi")
than it gives O/P like
{'k', 'i', 'w'}
~~if~~ it removes one 'i'.

for ex:-

```
fruits.update ('kiwi')
print (fruits)
```

O/P:
{'apple', 'banana', 'k', 'mango', 'i', 'w'}.

Remove () & discard ():-
In python we use discard () and remove () to remove items from sets.

→ The difference b/w discard () & remove() is that if we remove any item from sets which is not present than it give O/P same as ~~the~~ i/p
but discard () in this case will through an error.

their is a another funch pop ()

Pop ( ) → It is uses to remove the last added items of the Sets, we can't pass any value or argument in pop ( )

Ex:— discard ( )

```
fruits1 = {"apple", "mango", "grapes",
           "kiwi"}
print (fruits1)
fruits1. discard ('apple')
print (fruits1)
```

O/P:—

```
{"apple", "mango", "grapes",
 "kiwi"}
{"~~apple~~ mango", "grapes", "kiwi"}
```

Ex:— remove ( )

```
fruits1. remove ('mango')
Print (fruits1)
```

O/P:—

```
{"grapes", "kiwi"}
```

Ex:— pop ( )

```
fruits1. pop ( )
```

→ No need to give Value because it remove last value itself

```
print (fruits1)
```

O/P:

```
{"grapes", "mango"}
```

Other functions of sets are :-

i) union () → This function uses to combines the data present in both sets.

Ex:-
```
# union()
  my_set1 = {1,2,3,4}
  "      "2 = {3,4,5,6}

print (~~myset~~ my-set1.union (my-set2))

O/P:-
  {1,2,3,4,5,6}.
```

i intersection () → This funcn is uses to find data common in both sets.

Ex:
```
print ( my-set1.intersection( my-set2))

O/P:-
     { 3,4 }
```

difference () → It detects the data present in both and O/P data which is present in first set.

Ex:-
```
print (my-set1. difference (my_set2))
O/P
   {1,2}
"  ( "  2.    "     ( "  1))
O/P
 { 5,6 }
```

) Symmetric_difference() :— It is same as difference () but it ~~[illegible]~~ gives o/p data which is remaining difference of both sets.

Ex:—

Symmetric-difference( )
print(my-set1. difference(my-set2))

O/P:—
{1,2,5,6}

) issubset () → It is uses to check the given set is subset of other or not.

S_1 = {1,2,3}.

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

Ex:= # check is subset or not

a= {4,5}
b= {1,2,3,4,5}
a. issubset (b)
O/P
True
b. issubset (a)
O/P
False

check if a set is a subset, using comparision operator.

a = { 'a', 'b' }

b = { 'a', 'b', 'c' }

a<=b   or   b>=a
O/P   O/P
True   True

or

b<=a   or   a>=b
O/P   O/P
false.   false.

in and not in operator :- It uses to find that the element is present in set or not.

Ex:-   (in)

S = { 1, 2, 3, 4 }
5 in S
O/P
False

2 in S
O/P
True

(not in)
S = { 'x', 'y', 'z' }   'x' not in S
'w' not in S   O/P
O/P: True   false.