Set methods:

add()

The add() method adds an element to the set.

If the element already exists, the add() method does not add the element

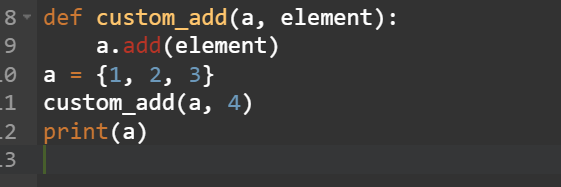
Example:

thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset) output:{‘banana’, ‘apple’, ‘orange’, ‘cherry’}

Logic:

 output: {1, 2, 3, 4}

Clear()

The clear() method removes all elements in a set.

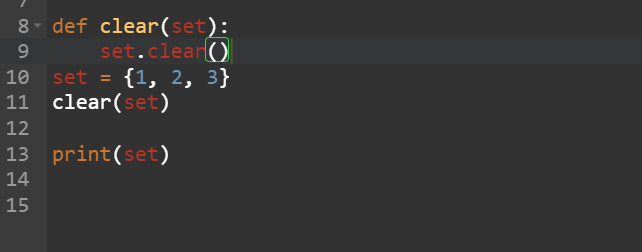
Example:

thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset) output: set()

Logic:

 output: set()

Copy()

The copy() method copies the set.

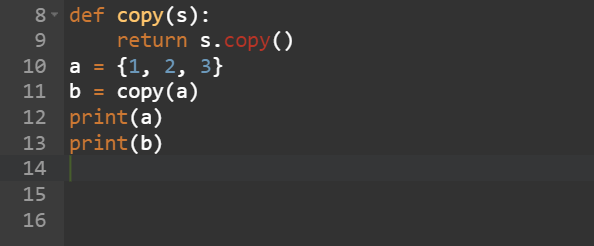
Example:

fruits = {"apple", "banana", "cherry"}

x = fruits.copy()

print(x) output: = {"apple", "banana", "cherry"}

Logic:

 output: {1, 2, 3}

difference()

The difference() method returns a set that contains the difference between two sets.

Meaning: The returned set contains items that exist only in the first set, and not in both sets.

Example:

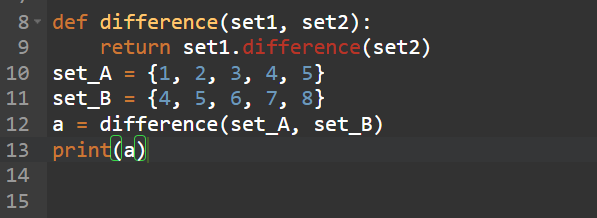
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.difference(y)

print(z) output: {‘cherry’, ’banana ’}

Logic:

 output: {1, 2, 3}

difference\_update()

The difference\_update() method removes the items that exist in both sets.

The difference\_update() method is different from the difference() method, because the difference() method returns a new set, without the unwanted items, and the difference\_update() method removes the unwanted items from the original set.

Example:

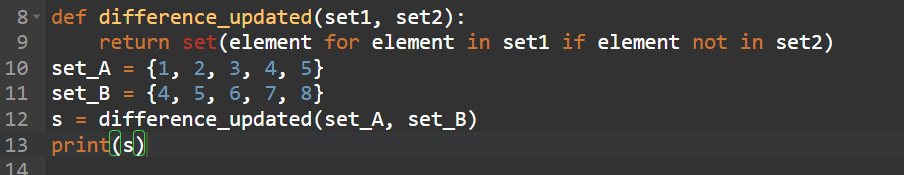
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.difference\_update(y)

print(x) output: {‘cherry’, ‘banana’}

Logic:

 output: {1,2,3,}

discard()

The discard() method removes the specified item from the set.

This method is different from the remove() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

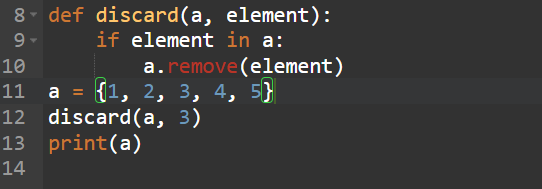
Example:

thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset) output: {"apple", "cherry"}

Logic:

 output: {1,2,4,5}

intersection()

The intersection() method returns a set that contains the similarity between two or more sets.

Meaning: The returned set contains only items that exist in both sets, or in all sets if the comparison is done with more than two sets.

Example:

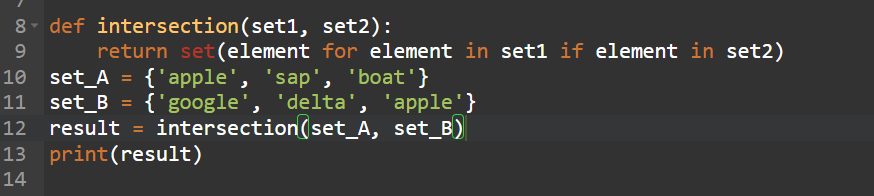
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z) output: {‘apple’}

Logic:

 output: {‘apple’}

intersecetion\_update()

The intersection\_update() method removes the items that is not present in both sets (or in all sets if the comparison is done between more than two sets).

The intersection\_update() method is different from the intersection() method, because the intersection() method returns a new set, without the unwanted items, and the intersection\_update() method removes the unwanted items from the original set.

Example:

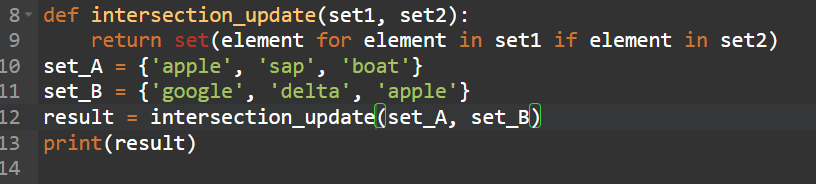
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.intersection\_update(y)

print(x) output: {‘apple’}

Logic:

 output: {‘apple’}

Isdisjoint()

The isdisjoint() method returns True if none of the items are present in both sets, otherwise it returns False.

Example:

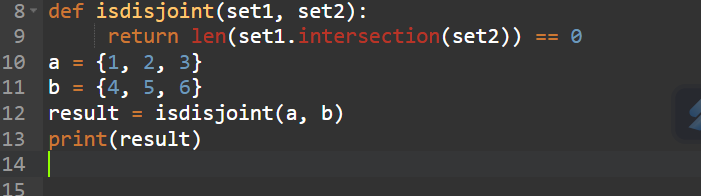
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "facebook"}

z = x.isdisjoint(y)

print(z) output: True

Logic:

 output: True

issubset()

The issubset() method returns True if all items in the set exists in the specified set, otherwise it returns False.

Example:

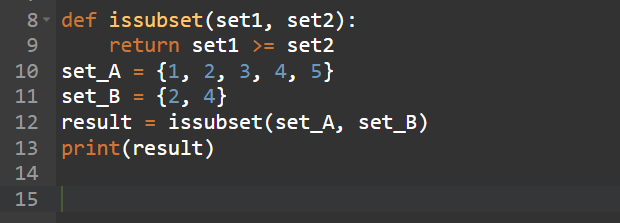
x = {"a", "b", "c"}

y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)

print(z) output: True

Logic:

 output: True

issuperset()

The issuperset() method returns True if all items in the specified set exists in the original set, otherwise it returns False.

Example:

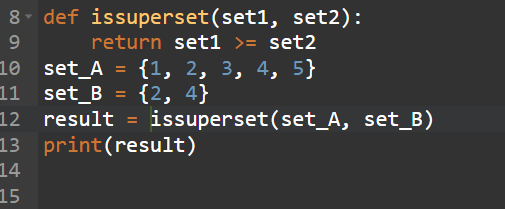
x = {"f", "e", "d", "c", "b", "a"}

y = {"a", "b", "c"}

z = x.issuperset(y)

print(z) output: True

Logic:

 output: True

Pop()

The pop() method removes a random item from the set.

This method returns the removed item.

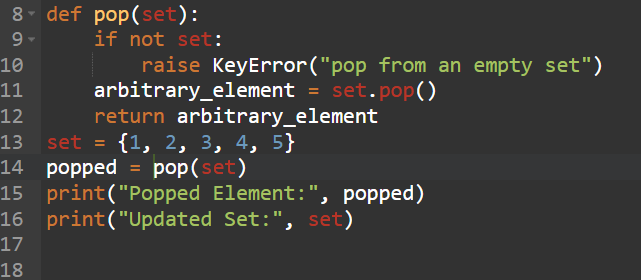
Example:

fruits = {"apple", "banana", "cherry"}

fruits.pop()

print(fruits) output: {‘apple’,’banana’}

Logic:

 output: popped element: 1 set: {2, 3, 4, 5}

remove()

The remove() method removes the specified element from the set.

This method is different from the discard() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

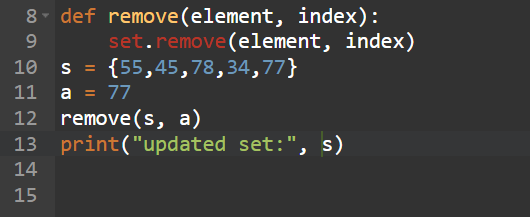
Example:

fruits = {"apple", "banana", "cherry"}

fruits.remove("banana")

print(fruits) output: {‘apple’,’cherry’}

Logic:

 output: updated set: {34, 55, 45, 78}

symmetric\_differenc()

he symmetric\_difference() method returns a set that contains all items from both set, but not the items that are present in both sets.

Meaning: The returned set contains a mix of items that are not present in both sets.

Example:

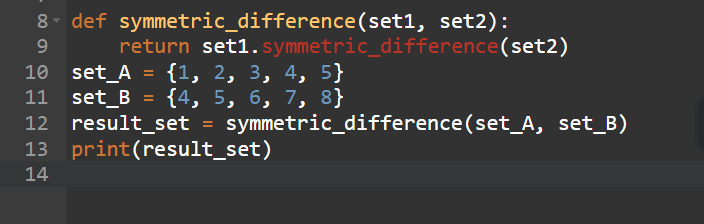
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.symmetric\_difference(y)

print(z) output: {‘cherry’, ‘banana’, ‘microsoft’, ‘google’,}

Logic:

 output: {1, 2, 3, 6, 7, 8}

union()

The union() method returns a set that contains all items from the original set, and all items from the specified set(s).

You can specify as many sets you want, separated by commas.

It does not have to be a set, it can be any iterable object.

If an item is present in more than one set, the result will contain only one appearance of this item.

Example:

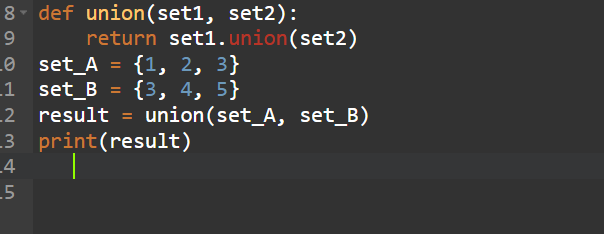
x = {"apple", "banana", “cherry”}

y = {"google", "microsoft", "apple"}

z = x.union(y)

print(z) output: {"google", "banana","cherry",”apple”,” microsoft”}

Logic:

 output: {1, 2, 3, 5, 6}

update()

The update() method updates the current set, by adding items from another set (or any other iterable).

If an item is present in both sets, only one appearance of this item will be present in the updated set.

Example:

x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.update(y)

print(x) output: { “microsoft", "banana",”apple”, “google”,"cherry"}

logic:

