

Set

Declaration:

> $s = \{ "one", "two", 1, 2 \}$
> $s = \text{set}()$

- set not ordered and not indexed
- set not slicing
- list and dict don't be in a set.
only immutable element
- items must be unique.
- $s = \{ \}$ confused with dict

Set Methodes:

clear()

> $s.\text{clear}()$
→ removes all elements in a set
no parameters

Add()

> $s.\text{add}(\text{elmnt})$
→ add element to the set
if element exist, does not add the element.

elmnt: required, The element to add to the set

copy()

> $s.\text{copy}()$
→ copy the set shallow copy
no parameters

remove()

> s.remove(*item*)

- remove specified element from the set
- raise error when item does not exist.

discard() does not raise error

item: required, the item to search for, and remove.

discard()

> s.discard(*value*)

- remove specified item from the set
- don't raise error when value does not exist.

Value: required, the item to search for, and remove.

pop()

> s.pop()

- remove and return random item from set
- no parameters.

update()

> s.update(*set*)

- update current set by adding items from another set (or iterable)
- eliminates duplicates

Set: required, iterable insert into current set.

union()

> s.union(set1, set2)

> a | B

→ return a set contain all items ~~at~~ from original set
add others sets (can be iterable (list))

Set1: required, iterable to unify.

Set2: optional, iterable to unify.

Difference()

> s.difference(set)

→ return a set, ~~difference~~

→ return a set contains items that exist only in the first set
and not in both sets.

Set: required, the set to check for difference in

> b - a

Difference_update()

> s.difference_update(set)

→ remove the items exists in both side
of assigned to original set.

Set: required, the set to check for difference in

intersection():

> s.intersection(set1, set2, ...)

> s & set1

→ return a set, contains similarity between two or more

Set1: required, the set to search for equal items in
Set2: optional, The other set

intersection_update():

> s.intersection_update(set1, set2, ...)

→ removes the items that is not in both side.

Set1: required, the set to search for equal items in
Set2: optional, the other set to search for equal items in

Symmetric_Difference():

> s.symmetric_difference(set)

> a ^ b

→ return a set, contains all items from both set, not the items that are present in both sets.

→ return mix of items that are not present in both set

Set: required, the set to check for matches in

Symmetric_difference_update():

> s.symmetric_difference_update(set)

→ updates original set by removing items that are present in both set, and inserting the other items.

Set: required, The set to check for matches in

issuperset()

> s.issuperset(*set*)

→ return True if all items in the specified set exists in original set otherwise it return False.

True if $\text{set} \supseteq \text{dams } S$.

Set: required, the set to search for equal items in

issubset()

> s.issubset(*set*)

→ True if $S \subseteq \text{dams set}$

→ return True if all items in the set exists in the parameter set

Set: required, the set to search for equal items in

isdisjoint()

> s.isdisjoint(*set*)

→ False si l'intersection n'est pas null.

→ return True if none of the items are present in both sets.

Set: required, the set to search for equal ~~at~~ items in