**Set**

* Set is a built-in data type
* Set is a collection which is unordered, unchangeable, do not allow duplicates, and unindexed
* Even though set is unchangeable, can remove and add items.
* Set is written within curly brackets
* Set can contain any data types ex: string, int, float, bool etc
* From python perspective set is defined as object with the data type ‘set’

Unordered means that the items in a set do not have a defined order. Set items can appear in a different order every time you use them, and cannot be referred to by index or key

**Create Set**

myset={"car", "van", "bike"}  
print(myset) *#{'car', 'van', 'bike'}*

**Duplicates are ignored**

setOne={"car", "van", "bike","car","bike"}  
print(setOne, "will only displayed and duplicates are ignored")*#{'van', 'car', 'bike'} will only displayed and duplicates are ignored*

**Length of a set**

(if there is duplicates, they will ignored and provides the result)

setOne={"car", "van", "bike","car","bike"}  
print(len(setOne)) *# 3*

**set()** can be used to create set but remember to put values in double round brackets

x=set(("car", "van", "bike"))  
print(x) *# {'bike', 'car', 'van'}*

**Access items in set**

Referring index or a key , cannot access the items in a set. Therefore, loop is used.

print("Loop through the set, and print the values")  
setOne={"car", "van", "bike","lorry","boat"}  
for i in setOne:  
 print(i)

# lorry

# car

# van

# bike

# boat

print("Check if \"boat\" is present in the set:")  
setOne={"car", "van", "bike","lorry","boat"}  
if "boat" in setOne:  
 print("boat is present in setOne set") *#boat is present in setOne set*

**add()**—add item to a set

setOne={"car", "van", "bike","lorry","boat"}  
setOne.add("bus")  
print(setOne) *#{'bus', 'car', 'boat', 'bike', 'van', 'lorry'}*

**update()—**add item from another set to a current set or any iterable (tuple, dictionary, list) to current set, and it will exclude duplicates

setTwo={"train","ship"}  
setOne.update(setTwo)  
print(setOne)*#{'bike', 'bus', 'boat', 'lorry', 'ship', 'car', 'train', 'van'}*

**remove() and discard()**—used to remove item from the set. **Note:** If the item to remove does not exist, remove() will raise an error.

setOne={'bike', 'bus', 'boat', 'lorry', 'ship', 'car', 'train', 'van'}  
setOne.remove("bike")  
print(setOne)*#{'train', 'van', 'bus', 'lorry', 'car', 'boat', 'ship'}*

Key error shows

setOne={'car', 'van', 'ship', 'boat', 'lorry', 'train', 'bus'}  
setOne.remove("bike")  
print(setOne)*#KeyError: 'bike'*

 If the item to remove does not exist, discard() will **NOT** raise an error.

setOne={'car', 'van', 'ship', 'boat', 'lorry', 'train', **'bus'**}  
setOne.discard("**bus**")  
print(setOne)*#{'van', 'train', 'ship', 'car', 'boat', 'lorry'}*setOne={'train', 'van', 'car', 'ship', 'boat', 'lorry'}  
setOne.discard("**bus**")  
print(setOne)*#{'van', 'train', 'ship', 'car', 'boat', 'lorry'}*

**pop() method**— to remove an item, but this method will remove the *last* item. Remember that sets are unordered, so you will not know what item that gets removed.The return value of the pop() method is the removed item.

setOne={'train', 'van', 'car', 'ship', 'boat', 'lorry'}  
setOne.pop()  
print(setOne)*#{'ship', 'van', 'lorry', 'boat', 'car'}*

**Clear()**—used to clear the items in the set

setOne={'ship', 'van', 'lorry', 'boat', 'car'}  
setOne.clear()  
print(setOne)*#set()*

**del** keyword used to entirely delete the set

setOne={}  
del setOne  
print(setOne)*#NameError: name 'setOne' is not defined*

**Union()**- is used to join the two sets to create a new set and it will exclude duplicates

print("join item from two sets to a new set")  
setOne={"car", "van", "bike","lorry","boat"}  
setTwo={"train","ship"}  
setThree=setOne.union(setTwo)  
print(setThree)*#{'car', 'van', 'bike', 'ship', 'boat', 'train', 'lorry'}*

**intersection()** - return a new set, that only contains the items that are present in both sets.

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
z = x.intersection(y) *# create new set by having only duplicates*print(z)*#{'apple'}*

**intersection\_update()**—keep only the items, that are present in both sets, in the specified set.

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
x.intersection\_update(y) *# keep only the items ,that are present in both sets,in the specified set*print(z)*#{'apple'}*

**symmetric\_difference\_update() method** —will keep only the elements that are NOT present in both sets.

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
x.symmetric\_difference\_update(y)  
print(x)

**symmetric\_difference() method**— will return a new set, that contains only the elements that are NOT present in both sets.

set\_1={11,12,5,16,4}  
set\_2={5,12,25,86,100}  
set\_3=set\_1.symmetric\_difference(set\_2)  
print(set\_3) *#{16, 100, 86, 4, 25, 11}*

**copy()—**uses to copy the specified set to a new set

set\_1={11,12,5,16,4}  
set\_2=set\_1.copy()  
print(set\_2) *# {16, 4, 5, 11, 12}*

**difference()—**Return a new set that contains the items that only exist in specified set, and not in other set

set\_1={11,12,5,16,4}  
set\_2={5,12,25,86,100}  
set\_new=set\_1.difference(set\_2)  
print(set\_new, "are only exists in set\_1 set but not in set\_2 set")

*#{16, 11, 4} are only exists in set\_1 set but not in set\_2 set*

**difference\_update()—**remove the items that exists in both sets and returns the items only for original set (specified set)

set\_1={11,12,5,16,4}  
set\_2={5,12,25,86,100}  
set\_2.difference\_update(set\_1)  
print(set\_2)*#{100, 86, 25}*

# 12 and 5 are in both sets, set\_2 returns with items that are presented only that set.

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

print("Return True if no items in set x is present in set y")  
x={11,16,4}  
y={5,12,25,86,100}  
z=x.isdisjoint(y)  
print(z)*#True*

print("Return False if one ore more items are present in both sets:")  
set\_1={11,12,5,16,4}  
set\_2={5,12,25,86,100}  
z=set\_1.isdisjoint(set\_2)  
print(z) *#False*

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

print("Return True if all items in set x are present in set y")  
x = {"a", "b", "c"}  
y = {"f", "e", "d", "c", "b", "a"}  
z= x.issubset(y)  
print(z)*#True*

print("Return False if not all items in set x are present in set y")  
x = {"a", "b", "c"}  
y = {"f", "e", "d", "c"}  
z= x.issubset(y)  
print(z)*#False*

All Set Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| add() | Adds an element to the set |
| clear() | Removes all the elements from the set |
| [copy()](https://www.w3schools.com/python/ref_set_copy.asp) | Returns a copy of the set |
| [difference()](https://www.w3schools.com/python/ref_set_difference.asp) | Returns a set containing the difference between two or more sets |
| [difference\_update()](https://www.w3schools.com/python/ref_set_difference_update.asp) | Removes the items in this set that are also included in another, specified set |
| discard() | Remove the specified item |
| intersection() | Returns a set, that is the intersection of two other sets |
| intersection\_update() | Removes the items in this set that are not present in other, specified set(s) |
| [isdisjoint()](https://www.w3schools.com/python/ref_set_isdisjoint.asp) | Returns whether two sets have a intersection or not |
| [issubset()](https://www.w3schools.com/python/ref_set_issubset.asp) | Returns whether another set contains this set or not |
| [issuperset()](https://www.w3schools.com/python/ref_set_issuperset.asp) | Returns whether this set contains another set or not |
| pop() | Removes an element from the set |
| remove() | Removes the specified element |
| symmetric\_difference() | Returns a set with the symmetric differences of two sets |
| symmetric\_difference\_update() | inserts the symmetric differences from this set and another |
| union() | Return a set containing the union of sets |
| update() | Update the set with the union of this set and others |