SET

* A set is a collection which is unordered and unindexed. In Python, sets are written with curly brackets.
* Set’s values are **Immutable** objects that can’t change their values.
* Create a Set:

thisset = {"apple", "banana", "cherry"}  
 print(thisset)

* A Set is enclosed by *curly braces* ({ }), the items are separated by *commas*.
* Set’s values can’t access by referring to an index. But you can loop through the set items using a **for loop**, or ask if a specified value is present in a set, by using the **in**keyword.
* thisset = {"apple", "banana", "cherry"}  
    
  for x in thisset:  
    print(x)
* Add Items

To add one item to a set use the add() method.

To add more than one item to a set use the update() method.

Example

* Add an item to a set, using the add() method:
* thisset = {"apple", "banana", "cherry"}  
    
  thisset.add("orange")  
    
  print(thisset)

*Result:*

{'cherry', 'orange', 'banana', 'apple'}

* Add multiple items to a set, using the update() method:
* thisset = {"apple", "banana", "cherry"}  
    
  thisset.update(["orange", "mango", "grapes"])  
    
  print(thisset)

*Result:*

{'cherry', 'orange', 'mango', 'grapes', 'apple', 'banana'}

* Change Items

Once a set is created, you cannot change its items, but you can add new items.

* Remove Set Item

To remove an item in a set, use the remove(), or the discard() method.

Example

Remove "banana" by using the remove() method:

thisset = {"apple", "banana", "cherry"}  
  
 thisset.remove("banana")  
  
 print(thisset)

*Result:*

{'apple', 'cherry'}

Remove "banana" by using the discard() method:

thisset = {"apple", "banana", "cherry"}  
  
 thisset.discard("banana")  
  
 print(thisset)

*Result:*

{'cherry', 'apple'}

* You can also use the 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.

Example

Remove the last item by using the pop() method:

thisset = {"apple", "banana", "cherry"}  
  
 x = thisset.pop()  
  
 print(x)  
  
 print(thisset)

* The clear() method empties the set:

The clear() method empties the set:

thisset = {"apple", "banana", "cherry"}  
  
thisset.clear()  
  
print(thisset)

* The del keyword will delete the set completely:

thisset = {"apple", "banana", "cherry"}  
  
 del thisset  
  
 print(thisset)

SET Operations

In Python, below operands can be used for different operations.

*| for union.  
& for intersection.  
– for difference  
^ for symmetric difference*

|  |
| --- |
| # Program to perform different set operations  # as we do in  mathematics    # sets are define  A = {0, 2, 4, 6, 8};  B = {1, 2, 3, 4, 5};    # union  print("Union :", A | B)  Result:  Union : {0, 1, 2, 3, 4, 5, 6, 8}    # intersection  print("Intersection :", A & B)  Result:  Intersection : {2, 4}  # difference  print("Difference :", A - B)  Result:  Difference : {0, 8, 6}  # symmetric difference  print("Symmetric difference :", A ^ B)  Result:  Symmetric difference : {0, 1, 3, 5, 6, 8} |

Using Set Functions

Union

Return a set that contains all items from both sets, duplicates are excluded:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.union(y)   
  
print(z)

# Intersection

Return a set that contains the items that exist in both set x, and set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.intersection(y)   
  
print(z)

Difference

Return a set that contains the items that only exist in set x, and not in set y:

x = {"apple", "banana", "cherry"}  
y = {"google", "microsoft", "apple"}  
  
z = x.difference(y)   
  
print(z)

[Run example »](https://www.w3schools.com/python/showpython.asp?filename=demo_ref_set_difference)

Symmetric Difference

Return a set that contains all items from both sets, except items that are present in both sets:

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

//////////////////////////////////

**Example:**

sd = {1,2,3,4,4,6,5}

print(sd) # No duplicate values are printed

sd1 = {'a','b','c'}

print(sd1) # Each time prints in different order. That is, sets are unordered and can’t be call # with an index value.

**Result:**

{1, 2, 3, 4, 5, 6}

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

//////////////////////////////

**Example:**

a = [1,2,3]

b = set(a)

print(b)

**Result:**

{1, 2, 3}

/////////////////////////////

**Example:**

a = {1,2,3}

a.add('hello') # Adding different datatype

print(a)

**Result:**

{1, 2, 3, 'hello'}

/////////////////////////////

**Example:**

a = {1,2,3}

a.add(3) # Will not add duplicate elements

print(a)

**Result:**

{1, 2, 3}

/////////////////////////////

Two sets are said to be disjoint sets if they have no common elements.

The *isdisjoint()* method takes a single argument (a set).

You can also pass an iterable (list, tuple, dictionary and string) to disjoint().

The *isdisjoint()* method will automatically convert iterables to set and checks whether the sets are disjoint or not.

The *isdisjoint()* method returns

* True 🡪  if two sets are disjoint sets
* False 🡪 if two sets are not disjoint sets

///////////////////////////////////

**Example:**

A = {1, 2, 3, 4}

B = {5, 6, 7}

C = {4, 5, 6}

print('Are A and B disjoint?', A.isdisjoint(B))

print('Are A and C disjoint?', A.isdisjoint(C))

A = {1, 2, 3, 4}

B = {5, 6, 7}

C = {4, 5, 6}

print(A.isdisjoint(B))

print(A.isdisjoint(C))

**Result:**

True

False

**Example:**

A = {'a', 'b', 'c', 'd'} # set

B = ['b', 'e', 'f'] # list

C = '5de4' # string

D ={1 : 'a', 2 : 'b'} # dictionary

E ={'a' : 1, 'b' : 2}

print(A.isdisjoint(B))

print(A.isdisjoint(C))

print(A.isdisjoint(D))

print(A.isdisjoint(E))

**Result:**

False

False

True

False

**Example:**

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

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

z = x.issubset(y)

print(z)

**Result:**

True

**Example:**

x = {"a", "b", "c"}  
y = {"f", "e", "d", "c", "b"}  
  
z = x.issubset(y)  
  
print(z)

**Result:**

False

**Example:**

n = set(["a", "b","c"])

n.add("d")

print("Normal Set")

print(n)

f = frozenset(["e", "f", "g"])

print("Frozen Set")

print(f)

# Uncommenting below line would cause error as

# we are trying to add element to a frozen set

# f.add("h")

**Result:**

Normal Set

{'a', 'c', 'b', 'd'}

Frozen Set

frozenset({'f', 'g', 'e'})

**Example:**

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

p = {"d","e","f"}

print(n <= p)

print(n < p)

print(n != p)

print("\n")

n = {"d","e"}

p = {"d","e","f"}

print(n <= p)

print(n < p)

print(n != p)

print("\n")

n = {"d","e","f"}

p = {"d","e","f"}

print(n <= p)

print(n < p)

print(n != p)

**Result:**

False

False

True

True

True

True

True

False

False

**Example:**

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

p = {"d","e","f","g"}

print(n & p)

n = {1,2,3}

p = {3,4,5,6}

print(n & p)

n = {"a","b","c","d"}

p = {"d","e","f","g"}

print(n & p)

**Result:**

set()

{3}

{'d'}

**Assignments**

1

Take user input a set and a list. Compare their lengths and find biggest.

2

*I am a contractor and I love to design and construct buildings*. How many unique words have been used in the sentence? Use the *split* methods and *set* to get the unique words.

3

Write a Python program to remove the intersection of a 2nd set from the 1st set.