

Python Set

Python Sets

- A set is an unordered collection of items. Every element is unique and must be immutable.
- Python sets are mutable; you can add and remove items from it.
- Sets can be used to perform mathematical set operations like:
 - Union :
 - Intersection
 - symmetric difference
 - etc.

Creating a Set

- A set is created by placing all the elements inside curly braces {}, separated by comma or by using the built-in function set().
- The elements can be of different types (integer, float, tuple, string etc.).
- But a set cannot have a mutable element, like list, set or dictionary, as its element.

```
#creating a set
numberSet = {1,2,3,4,3,2}
print(numberSet) |

#creating an empty set
emptySet = {} #This creates a dictionary
print(type(emptySet))

emptySet = set() #This creates a empty set
print(type(emptySet))
```

{1, 2, 3, 4}

<class 'dict'>

<class 'set'>

- A set can contain elements of different type

```
# set of mixed datatypes  
my_set = {1.0, "Hello", (1, 2, 3)}  
print(my_set)
```

```
{1.0, 'Hello', (1, 2, 3)}
```

- A set can contain elements of different type

```
set_with_lists = {[1,2,3]}
```

TypeError

- We can convert a list to set using set function

```
set_with_lists = set([1,2,3])  
print(type(set_with_lists))  
print(set_with_lists)
```

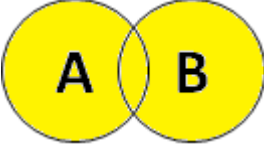
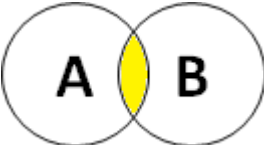
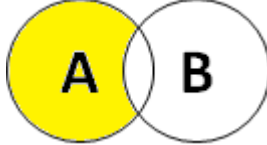
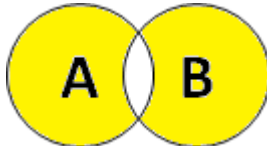
```
<class 'set'>  
{1, 2, 3}
```

Adding elements to a set

- Sets are mutable. But since they are unordered, indexing have no meaning.
- We cannot access or change an element of set using indexing or slicing.
- We can add single element using the `add()` method and multiple elements using the `update()` method.
 - `add()` for single elements
 - `update()` for iterable
- The `update()` method can take tuples, lists, strings or other sets as its argument.
- In all cases, duplicates are avoided.

Python Set Operations

- Sets can be used to carry out mathematical set operations like *union*, *intersection*, *difference* and *symmetric difference*.
- We can do this with operators or methods.

union		
intersection	&	
difference	-	
Symmetric difference	^	

Set Operations

- If A and B are two python sets

```
1 A = {1,2,3,4}
2 B = {2,4,6,8, 6, 6}
3
4 print('A= ',A)
5 print('B= ',B)
6
7 s1 = A | B    # elements in a or b or both
8 print('Union: A | B =',s1)
9
10 s2 = A & B    # elements in both a and b
11 print('Intersection: A & B =',s2)
12
13 s3 = A - B    # elements in a but not in b
14 print('Difference: A - B =',s3)
15
16 s4 = A ^ B    # elements in a or b but not both
17 print('Symmetric Diff: A ^ B =',s4)
```

A=	{1, 2, 3, 4}
B=	{8, 2, 4, 6}
A - B =	{1, 3}
A B =	{1, 2, 3, 4, 6, 8}
A & B =	{2, 4}
A ^ B =	{1, 3, 6, 8}

Operations on a set

operation	
<code>len(s)</code>	Return the number of elements in set <u>s</u> (cardinality of <u>s</u>).
<code>x in s</code>	Test <u>x</u> for membership in <u>s</u> .
<code>x not in s</code>	Test <u>x</u> for non-membership in <u>s</u> .
<code>set <= other</code>	Test whether every element in the <u>set</u> is in <u>other</u> .
<code>set < other</code>	Test whether the <u>set</u> is a <i>proper subset</i> of <u>other</u>
<code>set >= other</code>	Test whether every element in <u>other</u> is in the <u>set</u> .
<code>set > other</code>	Return a new set with elements from the <u>set</u> and all <u>others</u> .

Methods

Method	Description
add(elem)	Add element elem to the set.
remove(elem)	Remove element elem from the set. <u>Error</u> if elem is not contained in the set.
discard(elem)	Remove element elem from the set <u>if it is present</u> .
clear()	Remove all elements from the set.
pop()	Remove and return <i>an arbitrary element</i> from the set. <u>Error</u> if the set is empty.

Method	Description
isdisjoint(other)	Return True if the set has no elements in common with other. Sets are disjoint if and only if their intersection is the empty set.
issubset(other) set <= other	Test whether every element in the set is in other.
issuperset(other) set >= other	Test whether every element in other is in the set

Adding elements to a set

```
my_set = set()    #empty set
my_set.update([9 , 12])
my_set.update((3,5))
my_set.update("SIKANDER")

print(my_set)
```

```
my_set.update(("INDIA" , "BHARAT"))
print(my_set)
```

```
my_set.update(4,5)
```

TypeError

{3, 5, 9, 'A', 12, 'N', 'E', 'S', 'D', 'R', 'I', 'K'}

Set Comprehensions

```
>>>
>>> L = [1,3,2,6,4]
>>> ll = [x*10 for x in L]
>>> ll
[10, 30, 20, 60, 40]
>>> ss = {x*10 for x in L}
>>> ss
{40, 10, 20, 60, 30}
>>> L.extend([1,2])
>>> L
[1, 3, 2, 6, 4, 1, 2]
>>> ss = {x*10 for x in L}
>>> ss
{40, 10, 20, 60, 30}
>>>
```

```
>>> a = {1,2,3}
>>> b = {x*2 for x in a|{4,5}}
>>> b
{2, 4, 6, 8, 10}
>>>
>>> type(b)
<class 'set'>
>>>
```