Sets

A **set** is a collection of elements with no repeats and without insertion order but sorted order.

Basic uses include membership testing and eliminating duplicate entries. Set objects also support mathematical operations like union, intersection, difference, and symmetric difference.

They can hold multiple data in them, but only one of value. They are used in situations where it is only important that some things are grouped together, and not what order they were included.

### Creating a Set

Curly braces **'{}'** or the **set()** function can be used to create sets. But the only way to create an empty set is: use the set() function.

**⚠️Avoid ! :**

* Note that, to create an empty set you have to use **set()** function. Do not use **{}** to create an empty set. Otherwise, you will create an empty dictionary.

Let's create a simple empty set :

empty\_set = set()

This is our first set in this lesson. Now let's print its type.

input :

empty\_set = set()

print(type(empty\_set))

output :

<class 'set'>

We will now see how sets have unordered and unique objects.

input :

colorset = {'purple', 'orange', 'red', 'darkblue', 'yellow', 'red'}

print(colorset)

print(colorset)

output :

{'darkblue', 'orange', 'purple', 'red', 'yellow'}

{'darkblue', 'purple', 'orange', 'yellow', 'red'}

As you can see in the output, the two 'red' values ​​we have defined in the set have fallen to one. And every time you print the set, the order of the objects in the set changes.

Let's look at another example :

input :

s = set('unselfishness')

print(s)

output :

{'f', 'l', 'i', 'u', 'e', 'n', 'h', 's'}

As you can see, the letters of the string type data are only written once in the set. Within this scope, using sets can help you avoid repetitions. Let's convert a list into a set and look at the repetitions of its elements:

input :

flower\_list = ['rose', 'violet', 'carnation', 'rose', 'orchid', 'rose',

    'orchid']

flowerset = set(flower\_list)

flowerlist = list(flowerset)

print(flowerset)

print(flowerlist)

output :

{'orchid', 'carnation', 'violet', 'rose'}

['orchid', 'carnation', 'violet', 'rose']

**✏️Homework:**

* {'carnation', 'orchid', 'rose', 'violet'} 👈👉 {'rose', 'orchid', 'rose', 'violet', 'carnation'} Do these two sets give the same output and why? (Note: Try to figure out the answer before run on the Playground)

**Q**: Which one of the following is not the correct syntax for creating a set in Python?  
**A**:  
**a. set([[1,2],[3,4],[4,5]])**  
b. set([1,2,2,3,4,5])  
c. {1,2,3,4}  
d. set((1,2,3,4))  
  
**Explanation:** The argument given for the set must be an iterable

**- Interview Q&A**

### Main Operations with Sets

There are several methods that allow us to add and remove items to/from sets. Moreover, we have the methods of intersection, unification, and differentiation of sets :

These methods are :

* **.add()** : Adds a new item to the set.
* **.remove()** : Allows us to delete an item.
* **.intersection()** : Returns the intersection of two sets.
* **.union()** : Returns the unification of two sets.
* **.difference()** : Gets the difference of two sets.

Now, let's do some examples of these methods :

input :

a = set('abracadabra')

print(a)

output :

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

input :

a = set('abracadabra')

b = set('alacazam')

print(a - b) # same as '.difference()' method

print(a.difference(b)) # a difference from b

output :

{'b', 'd', 'r'}

{'b', 'd', 'r'}

input :

a = set('abracadabra')

b = set('alacazam')

print(a | b) # same as '.union()' method

print(a.union(b)) # unification of a with b

output :

{'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}

{'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}

input :

a = set('abracadabra')

b = set('alacazam')

print(a & b) # same as '.intersection()' method

print(a.intersection(b)) # intersection of a and b

output :

{'a', 'c'}

{'a', 'c'}

input :

a = set('abracadabra')

a.remove('c') # we delete 'c' from the set

print(a)

output :

{'a', 'b', 'd', 'r'}

input :

a = set('abracadabra')

a.add('c') # we add 'c' again into the set

print(a)

output :

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

Additionally, you can:

* Get the number of set’s elements using **len()** function,
* Check if an element belongs to a specific set(in / not in operators), you get the boolean value.

Thus, we have completed this topic which is the most important one in Python.