Python - Sets

A set is a unique collection of objects in Python. It can be denoted with a curly bracket {}. Duplicate will be removed.

Creation

Set is created with {} brackets.

In [62]:

```
empty_set = {}
empty_set
```

Out[62]:

```
{}
```

Set can have elements in it.

In [2]:

```
my_set = {'a' , 'b', 'c'}
my_set
```

Out[2]:

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

If Set elements are duplicate then they are removed automatically

In [4]:

```
my_set = {'a' , 'b', 'c', 'a' , 'b', 'c'}   # set elements are duplicate and removed automa
my_set
```

Out[4]:

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

Set elements can be of mixed type.

In [6]:

```
my_mixed_set = {1, '1', 2, '2', 3, '3', '3', 1.1, True}
my_mixed_set
```

Out[6]:

```
{1, '1', 1.1, 2, '2', 3, '3'}
```

Type of set variable can be checked with type()

In [8]:

```
type(my_set)
```

Out[8]:

```
set
```

Type of individual set elements can be testd with type() on those elements.

Size of set can be determined using len().

In [12]:

```
len(my_set)
```

Out[12]:

```
3
```

Creation of set from list

Set can be created from list of elements.

In [17]:

```
my_list = [1, '1', 2, '2', 3, 3, 3, '3', '4', 4]  #create a list from which set needs to be
my_list
```

Out[17]:

```
[1, '1', 2, '2', 3, 3, 3, '3', '4', 4]
```

In [19]:

```
my_new_set = set(my_list)   # set can be created from list, duplicates are removed
my_new_set
```

Out[19]:

```
{1, '1', 2, '2', 3, '3', '4', 4}
```

Conversion of set to list

In [28]:

```
my_set = {'mumbai', 'pune', 'solapur'}  #create a set of cities
print(my_set)
print(type(my_set))
```

```
{'solapur', 'mumbai', 'pune'}
<class 'set'>
```

In [29]:

```
my_new_list = list(my_set) # use list() to convert a set into list
print(my_new_list)
print(type(my_new_list))
```

```
['solapur', 'mumbai', 'pune']
<class 'list'>
```

Operations on set

add() can be used to add an element into set.

In [32]:

```
country_set = {"India", "US", "US", "India"}
country_set   #set has only two elements "India" and "US"
```

Out[32]:

```
{'India', 'US'}
```

In [38]:

```
country_set.add('UK')
country_set  #set has three elements "India" ,"US" & "UK"
```

Out[38]:

```
{'India', 'UK', 'US'}
```

remove() can be used to remove an element from the set.

In [37]:

```
country_set.remove("UK")  # remove
country_set  #set has only two elements "India" and "US"
```

Out[37]:

```
{'India', 'US'}
```

In [63]:

```
country_set.remove("China")  # as China not present in the set, error thrown
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-63-38193c40d51e> in <module>
----> 1 country_set.remove("China")

KeyError: 'China'
```

discard() can be used to remove an element from set. If element is not present, then error is not thrown.

In [64]:

```
country_set.discard("China")
```

pop() operation can be used to remove the first element from the set.

In [70]:

```
test_set = {1, 2, 3, 4, 5}
```

In [71]:

```
test_set.pop()   # removes first element from set i.e. 1
test_set
```

Out[71]:

```
{2, 3, 4, 5}
```

In [73]:

```
test_set.pop()   # removes first element from set i.e. 2, set is altered by this operation
test_set
```

Out[73]:

```
{4, 5}
```

clear() can be used to remove all the elements of set.

In [78]:

```
test_set = {4, 5, 6}
test_set
```

Out[78]:

```
{4, 5, 6}
```

In [80]:

```
test_set.clear()
test_set
```

Out[80]:

```
set()
```

'in' and 'not in' can be used to determine if the element is present in set or not

In [39]:

```
"India" in country_set
```

Out[39]:

```
True
```

In [40]:

```
"Australia" in country_set
```

Out[40]:

```
False
```

In [41]:

```
"India" not in country_set
```

Out[41]:

```
False
```

In [42]:

```
"Australia" not in country_set
```

Out[42]:

```
True
```

Working with sets

'&' operator can be used to determine common elements of two or more sets

In [48]:

```
maharashtra_cities = {"Pune", "Mumbai", "Nagpur"}
goa_cities = {"Panjim", "Vasco", "Madgaon"}
western_region_cities = {"Panjim", "Vasco", "Pune", "Mumbai", "Nagpur"}
```

In [45]:

```
maharashtra_cities & western_region_cities  # common cities between two sets
```

Out[45]:

```
{'Mumbai', 'Nagpur', 'Pune'}
```

In [50]:

```
goa_cities & western_region_cities  # common cities between two sets
```

Out[50]:

```
{'Panjim', 'Vasco'}
```

In [49]:

```
maharashtra_cities & goa_cities  # common cities between two sets, nothing common
```

Out[49]:

```
set()
```

intersection() method can also be used to determine the common elements between two sets.

In [55]:

```
western_region_cities.intersection(maharashtra_cities)
```

Out[55]:

```
{'Mumbai', 'Nagpur', 'Pune'}
```

In [56]:

```
western_region_cities.intersection(goa_cities)
```

Out[56]:

```
{'Panjim', 'Vasco'}
```

'-' opearator can be used to determine the elements which are only present in set1.

In [74]:

```
western_region_cities - maharashtra_cities
```

Out[74]:

```
{'Panjim', 'Vasco'}
```

In [75]:

```
western_region_cities - goa_cities
```

Out[75]:

```
{'Mumbai', 'Nagpur', 'Pune'}
```

In [76]:

```
maharashtra_cities  - goa_cities
```

Out[76]:

```
{'Mumbai', 'Nagpur', 'Pune'}
```

difference method() can be used to determine the elements which are part of set 1 only, not present in set 2.

In [53]:

```
western_region_cities.difference(maharashtra_cities)
```

Out[53]:

```
{'Panjim', 'Vasco'}
```

In [ ]:

```
western_region_cities.difference(maharashtra_cities)
```

'|' operator can be used to get all cities from both sets

In [47]:

```
maharashtra_cities | western_region_cities  # all cities between two sets
```

Out[47]:

```
{'Mumbai', 'Nagpur', 'Panjim', 'Pune', 'Vasco'}
```

In [57]:

```
maharashtra_cities | goa_cities  # all cities between two sets
```

Out[57]:

```
{'Madgaon', 'Mumbai', 'Nagpur', 'Panjim', 'Pune', 'Vasco'}
```

union() method can be used to get all the elements from both sets.

In [58]:

```
maharashtra_cities.union(goa_cities)
```

Out[58]:

```
{'Madgaon', 'Mumbai', 'Nagpur', 'Panjim', 'Pune', 'Vasco'}
```

Exercise

Q1. Consider following set definitions.
language set - python, java, c, c++
data science language set - pytho, r

```
     Answer the following questions with help of appropriate code.
     (a) What are all programming languages are available?
     (b) How many data science languages are available?
     (c) List languages which are not data science languages?
     (d) List languages which are both programming language and data science language
    s.
     (e) List languges which are only data science languages.
```

In [87]:

```
#Try it here
```

In [ ]:

```
```