Set

1. Set is a built-in collection data type.
2. Sets are used to store multiple items in a single variable.
3. Sets are written with curly brackets.

eg : s = {1,2,3}

In [1]:
```
#Create a Tuple

s = {1,2,3,4}
print("Example Set:",s)
```

```
Example Set: {1, 2, 3, 4}
```

Characteristics of a Set

In [ ]:
```
Set items are :
1. Unordered                        : Items in a Set do not have a defined order.Set items can appear in a different ord
                                      and cannot be referred to by index or key.
2. Unchangeable (Immutable)         : Once a set is created, you cannot change its items, but you can remove items and a
3. Do not allow duplicate values    : Do not allow same values more than one time.
```

In [4]:
```
#Check whether set allow duplicate values

s = {1,2,3,2,4,1}
s1 = {'a','b','c','d'}
print(s1) # print items in different order than the input order
print(s) #duplicate values has been removed.
```

```
{'b', 'a', 'd', 'c'}
{1, 2, 3, 4}
```

The values True and 1 (also, False and 0) are considered the same value in sets, and are treated as duplicates:

In [9]:
```
#The values True and 1 are considered the same value in sets, and are treated as duplicates
s = {1,2,True,3,0,False}
```

```
print(s) #True ,False has been removed
```

```
{0, 1, 2, 3}
```

Set Length

In [ ]:
```
len() :Method to find the length of a set.Length means the no. of items in a set.
```

In [10]:
```
#Length of a set
s = {1,2,3,4,5}
print("Length of set is: ",len(s))
```

```
Length of set is:  5
```

Set Items - Data Types

Set items can take any data type. Set can have different datatypes as their elements.

type() : Method to find the type of a variale

In [16]:
```
#set with integer,string,boolean data items.
s1 = {1,2,3,4} # integer data items
s2 = {"welcome","to","set","Tutorial"} #string data items
s3 = {True,False} #boolean data items
s4 = {3,"Welcome",True} #different data items
print(s1,"\n",s2,"\n",s3,"\n",s4)
```

```
{1, 2, 3, 4}
 {'Tutorial', 'set', 'to', 'welcome'}
 {False, True}
 {'Welcome', True, 3}
```

In [17]:
```
#identify the type of a variale
s = {1,2,3,4}
print(type(s))
```

```
<class 'set'>
```

In [ ]:
```
The set() Constructor
```

In [ ]:
```
set() : Constructor of class Set, used to make a set.
```

In [18]:
```
#set constructor
l = [1,2,3,4,5]
s =set(l)
print(s) # list turned to a set
print(type(s))
```

```
{1, 2, 3, 4, 5}
<class 'set'>
```

Set Methods

In [ ]:
```
Python has a set of built-in methods that you can use on sets.

Method             Description
add()                    Adds an element to the set
clear()                  Removes all the elements from the set
copy()                   Returns a copy of the set
difference()             Returns a set containing the difference between two or more sets
difference_update()      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()             Returns whether two sets have a intersection or not
issubset()               Returns whether another set contains this set or not
issuperset()             Returns whether this set contains another set or not
pop()                    Removes an element from the set
remove()                 Removes the specified element
union()                  Return a set containing the union of sets
update()                 Update the set with the union of this set and others
```

Access Set Items

In [ ]:
```
We cannot access items in a set by referring to an index or a key.
But you can loop through the set items using a for loop.
in  :Keyword to check whether an item present or not
```

In [19]:
```
#loop through each item using for loop
s = {1,2,3,4,5}
for i in s:
    print(i)
```

```
1
2
3
4
5
```

In [20]:
```
#in keyword to check an item present or not
s = {1,2,3,4,5}
if 4 in s:
    print("yes,Item present")
else:
    print("No,Item not present")
```

```
yes,Item present
```

Change Items

We can not change an existing item in a set but we can add new items to the set.

Add Set Items

In [ ]:
```
add()      : Method in set used to add new items to the set. This method takes exactly one argument.
update()   : To add items from another set into the current set. The object in the update() method
             does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).
```

In [23]:
```
# Add new item to the set
s = {1,2,3,4,5}
s.add(6)
print(s)
```

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

In [26]:
```
# update() method
s1 = {1,2,3,4,5}
s2 = {6,7,8,9,10}
l = ['a','b','c','d']
s1.update(s2) # add items from another set
s2.update(l) #add items from a list
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{6, 7, 8, 9, 10, 'c', 'd', 'b', 'a'}
```

Remove Set Items

In [ ]:
```
remove()   : Method to remove an item from set.If the item to remove does not exist, remove() will raise an error.

discard()  : Method to remove an item from set.If the item to remove does not exist, discard() will not raise an error.

pop()      : Method to remove an item, but this method will remove a random item,
             so you cannot be sure what item that gets removed.

clear()    : Method to empties the set.

del        : Keyword to delete the set completely.
```

In [3]:
```
#use of remove()
s ={1,2,3,4,5}
s.remove(3)
#s.remove(6) #throw error since item dosen't exist.
print(s)
```

```
{1, 2, 4, 5}
```

In [6]:
```
#use of discard()
s ={1,2,3,4,5}
s.discard(3)
s.discard(6) #will not throw error even if the item dosen't exist.
print(s)
```

```
{1, 2, 4, 5}
```

In [7]:
```
#use of pop()
s ={1,2,3,4,5}
s.pop() #remove a random item from the set
print(s)
```

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

In [10]:
```
#use of clear()
s ={1,2,3,4,5}
s.clear() #Empties the set
print(s)
```

```
set()
```

In [11]:
```
#use of del keyword
s ={1,2,3,4,5}
del s #completely deleted the set
print(s) #throw error since the set was deleted
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 4
      2 s ={1,2,3,4,5}
      3 del s
----> 4 print(s)

NameError: name 's' is not defined
```

Loop Sets

We can use for loop to iterate through set elements.

In [12]:
```
#Use of for loop

s ={"Welcome","to","pyhton","set","tutorial"}
for i in s:
    print(i) #items comes in different order since set is unorderd
```

```
set
tutorial
to
pyhton
Welcome
```

Join Sets

In [ ]:
```
There are several ways to join two or more sets in Python:

1. union()               : Returns a new set containing all items from both sets.

2. update()              : Method to inserts all the items from one set into another.

3. intersection_update()  : Method to keep only the items that are present in both sets.

4. intersection()        : Method will return a new set, that only contains the items that are present in both sets.
```

```
5. symmetric_difference() : Method will return a new set, that contains only the elements that are NOT present in both

6. symmetric_difference_update() : Method will keep only the elements that are not present in both sets.

Both union() and update() will exclude any duplicate items.
```

In [13]:
```
#use of join()

s1 = {1,2,3,4}
s2 = {5,6,7,8}
s = s1.union(s2)
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [16]:
```
#use of update()

s1 = {1,2,3,4}
s2 = {5,6,7,8}
s1.update(s2)
print(s1)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [17]:
```
#use of intersection_update()

s1 = {1,2,5,4}
s2 = {5,2,7,8}
s1.intersection_update(s2)
print(s1)
```

```
{2, 5}
```

In [18]:
```
#use of intersection()

s1 = {1,2,5,4}
s2 = {5,2,7,8}
s3 = s1.intersection(s2)
print(s3)
```

```
{2, 5}
```

In [20]:
```
#use of symmetric_difference()

s1 = {1,2,5,4}
s2 = {5,2,7,8}
```

```
s3 = s1.symmetric_difference(s2)
print(s3)
```

```
{1, 4, 7, 8}
```

In [22]:
```
#use of symmetric_difference_update()

s1 = {1,2,5,4}
s2 = {5,2,7,8}
s1.symmetric_difference_update(s2)
print(s1)
```

```
{1, 4, 7, 8}
```

End