```
# Sets are used to store multiple items in a single variable.

A set is a collection which is unordered, unchangeable*, and unindexed.
```

In [1]:
```
s={1,2,3,4}
type(s)
```

Out[1]:
```
set
```

In [2]:
```
# empty set
s=set()
print(type(s))
```

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

In [4]:
```
# uplicates Not Allowed
s={2,3,4,5,4,2,1,7}
print(s)
```

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

In [5]:
```
# Access Items
for x in s:
    print(x)
```

```
1
2
3
4
5
7
```

In [7]:
```
s[2]   # because set is unordered and unindexed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 s[2]

TypeError: 'set' object is not subscriptable
```

In [8]:
```
# Add Items
s.add("red")
print(s)
```

```
{1, 2, 3, 4, 5, 7, 'red'}
```

In [9]:
```

```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 3
      1 s1={1,3,5}
      2 s2={2,4,6}
----> 3 s1.add(s2)
      4 print(s1)

TypeError: unhashable type: 'set'
```

In [10]:
```
# To add items from another set into the current set, use the update() method.
s1={1,3,5}
s2={2,4,6}
s1.update(s2)
print(s1)
```

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

In [12]:
```
s1.update(["pink","blue"])
```

In [13]:
```
print(s1)
```

```
{1, 2, 3, 4, 5, 6, 'blue', 'pink'}
```

In [14]:
```
# to remove an item in a set, use the remove(), or the discard() method.
s1.remove("blue")
print(s1)
```

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

In [15]:
```
s1.remove("red")
print(s1)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[15], line 1
----> 1 s1.remove("red")
      2 print(s1)

KeyError: 'red'
```

In [16]:
```
s1.discard("pink")
print(s1)
```

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

In [17]:
```
s1.discard("yellow")
print(s1)
```

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

In [18]:
```
# if item to remove is not in set then remove() will raise error, discard() not raise error
```

In [19]:
```
# pop() method to remove an item, but this method will remove a random item,
s1.pop()
```

Out[19]:
```
1
```

In [20]:
```
print(s1)
```

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

In [22]:
```
print(s1)
print(s2)
```

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

In [23]:
```
s1.clear()
print(s1)
```

```
set()
```

In [24]:
```
del s2
print(s2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 2
      1 del s2
----> 2 print(s2)

NameError: name 's2' is not defined
```

In [25]:
```
# Join Two Sets
s1={"a","b","c"}
s2={1,2,3}
```

In [26]:
```
s=s1.union(s2)
print(s)
```

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

In [27]:
```
# update() method that inserts all the items from one set into another:
s1.update(s2)
print(s1)
```

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

In [28]:
```
s3={2,4,5,6}
s1.update(s3)
print(s1)
```

```
{'a', 1, 2, 3, 4, 5, 6, 'c', 'b'}
```

In [31]:
```
s1.update({2,3,8})
print(s1)
```

```
{'a', 1, 2, 3, 4, 5, 6, 8, 'c', 'b'}
```

In [33]:
```
# Keep ONLY the Duplicates
s1={"red","green", "yellow","pink"}
s2={"pink","blue","red","purple"}
s=s1.intersection(s2)
print(s)
```

```
{'pink', 'red'}
```

In [34]:
```
s1.intersection_update(s2)
print(s1)
```

```
{'pink', 'red'}
```

In [35]:
```
# The intersection() method will return a new set, that only contains the items that are present in both sets.
# The intersection_update() method will keep only the items that are present in both sets.
```

In [36]:
```
# Keep All, But NOT the Duplicates
s1={"red","green", "yellow","pink"}
s2={"pink","blue","red","purple"}
s=s1.symmetric_difference(s2)
print(s)
```

```
{'green', 'yellow', 'purple', 'blue'}
```

In [37]:
```
s1.symmetric_difference_update(s2)
print(s1)
```

```
{'green', 'yellow', 'purple', 'blue'}
```

In [38]:
```
# The symmetric_difference_update() method will keep only the elements that are NOT present in both sets.
# The symmetric_difference() method will return a new set, that contains only the elements that are NOT present in both sets
```

In [40]:
```
a,b=1,0
a=a^b
b=a^b
a=a^b
print(a)
```

```
0
```

In [41]:
```
2**2**3**1
```

Out[41]:
```
256
```

In [43]:
```
2**8
```

Out[43]:
```
256
```

In [ ]:
```

```