Set is an unordered collection of data items that are unique. In other words, Python Set is a collection of elements (Or objects) that contains no duplicate elements.

Characteristics of Set :

1. Unordered
2. Unchangeable
3. Heterogeneous
4. Unique

# Creating a Set

1. Using Curly{} brackets
2. Using set() constructor

In [4]:

```
Set = {1,4,6,2,8,3}
print(Set)
```

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

In [5]:

```
sample_set = {'Sanvee', 25, 75.25}
print(sample_set)
```

```
{25, 75.25, 'Sanvee'}
```

In [1]:

```
Set1 = set((1,2,3,4))
print(Set1)
```

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

In [2]:

```
type(Set1)
```

Out[2]:

```
set
```

In [3]:

```
len(Set1)
```

Out[3]:

```
4
```

# Add items

Once a set is created, you cannot change its items, but you can add new items.

In [5]:

```python
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

```
{'banana', 'apple', 'cherry', 'orange'}
```

In [9]:

```python
thisset = {"apple", "banana", "cherry"}
thisset.update(["mango"])
print(thisset)
```

```
{'banana', 'apple', 'cherry', 'mango'}
```

In [12]:

```python
a = {"apple", "banana", "cherry"}
b = {"pineapple", "mango", "papaya"}

a.update(b)

print(a)
```

```
{'pineapple', 'mango', 'banana', 'papaya', 'apple', 'cherry'}
```

## Remove Items

In [17]:

```python
#Remove method
color_set = {'red', 'orange', 'yellow', 'white', 'black', 'blue', 'green'}

# remove single item
color_set.remove('yellow')
print(color_set)
```

```
{'red', 'white', 'blue', 'black', 'orange', 'green'}
```

In [18]:

```python
#discard method
color_set.discard('white')
print(color_set)
```

```
{'red', 'blue', 'black', 'orange', 'green'}
```

In [19]:

```
#pop() method
## remove any random item from a set

deleted_item = color_set.pop()
print(deleted_item)
```

```
red
```

In [20]:

```
# remove all items
color_set.clear()
print(color_set)
```

```
set()
```

In [21]:

```
color_set = {'red', 'orange', 'yellow', 'white', 'black', 'blue', 'green'}
print(color_set)
```

```
{'red', 'white', 'blue', 'yellow', 'black', 'orange', 'green'}
```

In [22]:

```
del color_set
print(color_set)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-22-8d3ad6ca884f> in <module>()
      1 del color_set
----> 2 print(color_set)

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

**remove() vs discard() :**

The remove() method throws a keyerror if the item you want to delete is not present in a set

The discard() method will not throw any error if the item you want to delete is not present in a set

# Set Operations

In [23]:

```
#Union
color_set = {'violet', 'indigo', 'blue', 'green', 'yellow'}
remaining_colors = {'indigo', 'orange', 'red'}

# union of two set using OR operator
vibgyor_colors = color_set | remaining_colors
print(vibgyor_colors)
```

```
{'violet', 'green', 'red', 'orange', 'blue', 'indigo', 'yellow'}
```

In [24]:

```
vibgyor_colors = color_set.union(remaining_colors)
print(vibgyor_colors)
```

```
{'violet', 'green', 'red', 'orange', 'blue', 'indigo', 'yellow'}
```

In [25]:

```
#Intersection
color_set = {'violet', 'indigo', 'blue', 'green', 'yellow'}
remaining_colors = {'indigo', 'orange', 'red'}

# intersection of two set using & operator
new_set = color_set & remaining_colors
print(new_set)
```

```
{'indigo'}
```

In [26]:

```
color_set = {'violet', 'indigo', 'blue', 'green', 'yellow'}
remaining_colors = {'indigo', 'orange', 'red'}

# difference using '-' operator
print(color_set - remaining_colors)

# using difference() method
print(color_set.difference(remaining_colors))
```

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

In [27]:

```
#Symmetric difference
color_set = {'violet', 'indigo', 'blue', 'green', 'yellow'}
remaining_colors = {'indigo', 'orange', 'red'}

# symmetric difference between using ^ operator
unique_items = color_set ^ remaining_colors
print(unique_items)
# Output {'blue', 'orange', 'violet', 'green', 'yellow', 'red'}

# using symmetric_difference()
unique_items2 = color_set.symmetric_difference(remaining_colors)
print(unique_items2)
```

```
{'red', 'violet', 'blue', 'yellow', 'orange', 'green'}
{'red', 'violet', 'blue', 'yellow', 'orange', 'green'}
```

In [29]:

```
color_set1 = {'violet', 'indigo', 'blue', 'green', 'yellow', 'orange', 'red'}
color_set2 = {'indigo', 'orange', 'red'}

# subset
print(color_set2.issubset(color_set1))
print(color_set1.issubset(color_set2))
```

```
True
False
```

In [30]:

```
print(color_set2.issubset(color_set1))
# True
print(color_set1.issubset(color_set2))
```

```
True
False
```

In [31]:

```
color_set1 = {'violet', 'blue', 'yellow', 'red'}
color_set2 = {'orange', 'red'}
color_set3 = {'green', 'orange'}

# disjoint
print(color_set2.isdisjoint(color_set1))
# Output 'False' because contains 'red' as a common item

print(color_set3.isdisjoint(color_set1))
```

```
False
True
```

In [32]:

```
set1 = {1, 2, 3, 4}
set2 = {0, 2, 4, 6, 8}  # set with one false value '0'
set3 = {True, True}  # set with all true
set4 = {True, False}  # set with one false
set5 = {False, 0}  # set with both false values

# checking all true value set
print('all() With all true values:', all(set1))
print('any() with all true Values:', any(set1))
```

```
all() With all true values: True
any() with all true Values: True
```

In [33]:

```
set1 = {2, 4, 6, 10, 8, 15}
set2 = {'ABC', 'abc'}

# Max item from integer Set
print(max(set1))  # 15

# Max item from string Set
print(max(set2))  # abc

# Minimum item from integer Set
print(min(set1))  # 2

# Minimum item from string Set
print(min(set2))  # ABC
```

```
15
abc
2
ABC
```

In [ ]:

```

```