Sets

Mathematically a set is a collection of items *(not in any particular order)*. A Python set is similar to this mathematical definition with below additional conditions.

- The elements in a set cannot be repeated i.e. an element can occur only once.
- The elements in the set are immutable*(cannot be modified)* but the set as a whole is mutable.
- There is no index attached to any element in a python set. So they do not support any indexing or slicing operation.

Set Operations

The sets in python are typically used for mathematical operations like union, intersection, difference, and complement, etc. We can create a set, access its elements, and carry out these mathematical operations as shown below.

Creating a set

A set is created by using the `set()` function or placing all the elements within a pair of curly braces, separated by commas.

```
Days=set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
Months={"Jan","Feb","Mar"}
Dates={21,22,17}
print(Days)
print(Months)
print(Dates)
```

When the above code is executed, it produces the following result.

```
set(['Wed', 'Sun', 'Fri', 'Tue', 'Mon', 'Thu', 'Sat'])
set(['Jan', 'Mar', 'Feb'])
set([17, 21, 22])
```

Note: The order of the elements has changed in the result.

Accessing Values in a Set

We cannot access individual values in a set as there is no specific order of elements in a set. We can only access all the elements together as shown. We can also get a list of individual elements by looping through the set.

```
Days=set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
for d in Days:
      print(d)
```

When the above code is executed, it produces the following result.

```
Wed
Sun
Fri
Tue
Mon
Thu
Sat
```

Adding Items to a Set

We can add elements to a set by using the `add()` method. Again as discussed there is no specific index attached to the newly added element.

```
>>> Days=set(["Mon","Tue","Wed","Thu","Fri","Sat"])
>>> Days.add("Sun")
>>> print(Days)
set(['Wed', 'Sun', 'Fri', 'Tue', 'Mon', 'Thu', 'Sat'])
```

Removing Item from a Set

We can remove elements from a set by using the `discard()` method. Again as discussed there is no specific index attached to the newly added element.

```
>>> Days=set(["Mon","Tue","Wed","Thu","Fri","Sat"])
>>> Days.discard("Sun")
>>> print(Days)
set(['Wed', 'Fri', 'Tue', 'Mon', 'Thu', 'Sat'])
```

Union of Sets

The union operation on two sets produces a new set containing all the distinct elements from both the sets. In the below example the element `"Wed"` is present in both the sets. The union operator is **'|'**.

```
DaysA = set(["Mon","Tue","Wed"])
DaysB = set(["Wed","Thu","Fri","Sat","Sun"])
AllDays = DaysA|DaysB #Union of Sets
print(AllDays)
```

When the above code is executed, it produces the following result. Please note the result has only one `"wed"`.

```
set(['Wed', 'Fri', 'Tue', 'Mon', 'Thu', 'Sat'])
```

Intersection of Sets

The intersection operation on two sets produces a new set containing only the common elements from both the sets. In the below example the element `"Wed"` is present in both the sets. The intersection operator is **"&"**.

```
DaysA = set(["Mon","Tue","Wed"])
DaysB = set(["Wed","Thu","Fri","Sat","Sun"])
AllDays = DaysA & DaysB #Intersection of Sets
print(AllDays)
```

When the above code is executed, it produces the following result. Please note the result has only one `"wed"`.

```
set(['Wed'])
```

Difference of Sets

The difference operation on two sets produces a new set containing only the elements from the first set and none from the second set. In the below example the element `"Wed"` is present in both the sets so it will not be found in the result set. The operator used is **"-"**.

```
DaysA = set(["Mon","Tue","Wed"])
DaysB = set(["Wed","Thu","Fri","Sat","Sun"])
AllDays = DaysA - DaysB #Difference of Sets
```

```
print(AllDays)
```

When the above code is executed, it produces the following result. Please note the result has only one `"wed"`.

```
set(['Mon', 'Tue'])
```

Compare Sets

We can check if a given set is a subset or superset of another set. The result is True or False depending on the elements present in the sets.

```
DaysA = set(["Mon","Tue","Wed"])
DaysB = set(["Mon","Tue","Wed","Thu","Fri","Sat","Sun"])
SubsetRes = DaysA <= DaysB #Check Subset
SupersetRes = DaysB >= DaysA #Check Superset
print(SubsetRes)
print(SupersetRes)
```

When the above code is executed, it produces the following result.

```
True
True
```