Set: A Set is a collection of unique elements — meaning it cannot contain duplicate values. It is unordered, so the items do not have a specific index like in a list or array.

Set Creation:

1) Set Literal:

```
Ex: void main()
    {
      var a={"apple","grapes","kiwi"};
      print(a); // {apple, grapes, kiwi}
      print(a.runtimeType); // LinkedSet<String>
    }
```

2) Set Constructor: Creates an empty hash set. By default, it's a Set<dynamic>, but you should specify type.

```
Ex: void main()

    {

     var b=Set();

     b.add(1);

     b.addAll([2,2,3]);

     print(b); // {1, 2, 3}

     print(b.runtimeType); // LinkedSet<dynamic>

    }
```

3) Typed Set Literal (`<type>{}`):

```
Ex: void main()

    {

        Set <String> lang={"c","java"};

        print(lang); // {c,java}

        print(lang.runtimeType);  // LinkedSet<String>
}
```

4) Set.from(Iterable): Takes an iterable (like list, set, etc.) .Removes duplicates automatically.

```
Ex: void main()
   {
      List<int> c=[1,1,2,2,3,4,5,6,6];
     Set<int> s=Set.from(c);
    print(s); // {1,2,3,4,5,6}
    print(s.runtimeType);  // LinkedSet<int>
  }
```

5) Set.of(Iterable):

```
Ex: void main()
  {
      List<String> fruits = ['apple', 'banana', 'banana'];
      Set<String> fruitSet = Set.of(fruits);
      print(fruitSet); // {apple, banana}
  }
```

6) Set.unmodifiable(): Creates an immutable set — cannot be modified. Trying to add/remove throws a runtime error.

```
Ex: void main()
     {
        List<int> l=[1,2,3,4,4,5];
       Set<int> s1=Set.unmodifiable(l);
       print(s1); // {1,2,3,4,5}
      // s1.add(4); // gives error
```

```
        }
```

Set Methods:

1) add(value): Adds an element to the set. Returns `true` if the element was added (not already present).

```
  Ex: void main()
   {
     var s = <int>{1, 2, 3};
     s.add(4);
     Print(s); // s becomes {1, 2, 3, 4}
   }
```

2) addAll(iterable): Adds all elements from an iterable to the set.

```
   Ex: void main()
         {
         var s = <int>{1, 2};
         s.addAll([3, 4]);
         print(s); // {1, 2, 3, 4}
        }
```

3) clear(): Removes all elements from the set.

```
  Ex: void main()
        {
          var s = {1, 2, 3};
          s.clear();
          print(s); // {}
        }
```

4) contains(value): Returns `true` if the set contains the element.

```
Ex: void main()
   {
     var s = {1, 2, 3};
     print(s.contains(2)); // true
   }
```

5) difference(otherSet): Returns a new set with elements in this set but not in `otherSet`.

```
   Ex: void main()

       {

          var s1 = {1, 2, 3};
          var s2 = {2, 3, 4};
          print(s1.difference(s2)); // {1}
      }
```

6) intersection(otherSet): Returns a new set with elements common to both sets.

```
   Ex: void main()

        {

            var s1 = {1, 2, 3};
            var s2 = {2, 3, 4};
            print(s1.intersection(s2)); // {2, 3}
        }
```

7) **i**sEmpty / isNotEmpty: Check if the set is empty or not.

```
   Ex: void main()

        {

            var s = <int>{};
            print(s.isEmpty);    // true
            print(s.isNotEmpty); // false
        }
```

8) remove(value): Removes an element from the set. Returns true if the element was present.

```
Ex: void main()

      {

         var s = {1, 2, 3};
         s.remove(2);
         print(s); // {1, 3}
      }
```

9) removeAll(iterable): Removes all elements that are present in the given iterable.

```
Ex: void main()

     {

        var s = {1, 2, 3, 4};
        s.removeAll([2, 4]);
       print(s); // {1, 3}
     }
```

10) retainAll(iterable): Keeps only elements that are in the given iterable (intersection in place).

```
  Ex: void main()

        {

  var s = {1, 2, 3, 4};
  s.retainAll([2, 3]);
  print(s); // {2, 3}
}
```

11) toList() / toSet(): Converts the set to a list or another set (useful for copying or transformations).

```
Ex: void main()

      {

         var s = {1, 2, 3};
         var list = s.toList();
         print(list); // [1, 2, 3]

     }
```

12) lookup(element): Returns the element in the set equal to the given element, or `null` if none.

```
Ex: void main()

      {

         var s = {'apple', 'banana'};
         print(s.lookup('banana')); // banana
         print(s.lookup('cherry')); // null
      }
```