Set

- unorderd data types
- it is mutable
- it has more functionality
- duplicates items are not allowed
- it is written inside a { }
- it returns unique values
- it cannot be access by indexing and slicing

Functions in set() :-

1. .add()

- in-place function
- it use to add only one element in a set
- however to add multiple elements in a set we either an use seperate .add() or
- If the element already exists, the add() method does not add the element.
- syntax =

```
- set_var.add(element)
```

In [ ]:
```
1
```

Note:- bellow methods accepts argument as a iterators su string, range etc...

however if dictionary passed then it only accept values fro

2. .update()

- in - place function
- it updates the current set, by adding a item from another set (or any other itera
- If an item is present in both sets, only one appearance of this item will be pres
- can update more than two sets at the same time
- syntax =

```
- set.update(set1,set2,set3...etc)
```

3. .union()

- returns a new set
- it is not a in- place method
- The union() method returns a set that contains all items from the original set, a set(s).
- You can specify as many sets you want, separated by commas.
- It does not have to be a set, it can be any iterable object.
- If an item is present in more than one set, the result will contain only one appe
- syntax:-

```
- set.union(set1, set2...)
```

4 .intersection()

- return new set that contains common elements or symetric elements from two
- Meaning: The returned set contains only items that exist in both sets, or in all with more than two sets.
- syntax

```
- set1.intersection(set2, set3 ... etc)
```

5. .intersection_update()

- this method removes the items that are not present in the both (or more than t comparison between more than two sets)
- it not important to be pass a set only as arguments in .intersection_update() m
- in-place fucntion
- syntax

```
    - set1.intersection_update(iterator1,iterator2, etc....)
```

6. .difference()

- The difference() method returns a set that contains the difference between two
- in-place function
- Meaning: The returned set contains items that exist only in the first set, and no
- returns the set without unwanted items
- not necesserely to have set only as aurguments
- syntax:-

```
- set.difference(iterator1,iterator2, etc....)
```

7. .difference_update()

- it removes the items that are present both the sets
- this method removes the items from the original set and updates the current s
- syntax :-

```
- set.difference_update(iterator1,iterator2, etc....)
```

8. .symetric_difference()

- The symmetric_difference() method returns a set that contains all items from l are present in both sets.
- meaning: This method will ignore all symetrical or same elemets from both the uncommon elements.
- Meaning: The returned set contains a mix of items that are not present in both
- syntax

```
- set.symetric_difference(iterator1,iterator2, etc....)
```

9. .symetric_difference_update()

- The symmetric_difference_update() method updates the original set by remov both sets, and inserting the other items.

Note : - this method only accept one iterator as a argument other wise it will iterator is passed

- syntax

```
- set.symetric_difference_update(iterator1, iterator2,etc....)
```

Delete Methods

1. .remove()

- this method will remove the element from the set

Note :- if we try to delete the item which is not present in a list then it will ret

2. .discard()

- This method will remove the element from the set

Note :- if we try to discard the item which is not present in a set then it will re

meaning it won't return an error, instead it will return a set with no change

3. .pop()

- The pop() method removes a random item from the set.
- This method returns the removed item.

4. .clear()

- this method will delete all items from a set and return an empty set

other methods

In [ ]:
```
1
```

1. .issubset()

- The issubset() method returns True if all items in the set exists in the specifiec
- in this method we can pass different iterators like list, tuple, range, dictionary,
- < use for proper subset and <= use for subset

Note : - Instead of using this issubset() you can also the ('<' or '<=') operator set is a subset of another set. this only works on sets

- Test whether every element in the set1 is in set2.

Note :- while .issubset() can also work on any type of iterators

In [ ]:
```
1
```

2. .issuperset()

- The issuperset() method returns True if all items in the specified set exists in t retuns False.

> - use for proper superset and >= use for superset

***Note : - Instead of using this issuperset() you can also use the ('>' or '>=') op
the set is a subset of another set. this only works on sets***

Note :- while .issuperset() can also work on any type of iterators

In [ ]:
```
1
```

3. .isdisjoint()

- The .isdisjoint() method returns True if none of the items are present in both s

set Comprehension

- new_set = {new_item for item in iterator}
- new_set = {new_item for item in iterator if condition}
- new_set = {new_item if condition else "statement" for item in iterator}

***Note :- after else statement if there is print statement then only use statemer
cut the print function from comprehension***

In [ ]:
```
1
```