- Python Built in Functions
- Tuple Examples
- Dictionary Practice Problem
- Set Functions
- Set Operation
- Set Examples

Python Built in Functions

- **sorted()**

Example

```
numbers = [4, 2, 12, 8]

sorted_numbers = sorted(numbers)
print(sorted_numbers)

# Output: [2, 4, 8, 12]
```

Set Functions

Python has a set of built-in methods that you can use on sets.

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two or more sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with another set, or any other iterable

copy() in Set

Copy()

set.copy()

Example:

```
fruits = {"apple", "banana", "cherry"}

x = fruits.copy()

print(x)
```

pop() function

pop( )

pop() → Removes Random element

```
fruits = {"apple", "banana", "cherry"}

fruits.pop()

print(fruits)
```

clear() function

clear()

clear() → Removes all element in SET

```
fruits = {"apple", "banana", "cherry"}
fruits.clear()
print(fruits)
```

update() function

update()

The `update()` method updates the current set, by adding items from another set (or any other iterable).

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.update(y)

print(x)
```

Intersection() in Set

Set intersection() Method

The `intersection()` method returns a set that contains the similarity between two or more sets.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.intersection(y)

print(z)
```

Set Intersection_update()

Set intersection_update() Method

- **The `intersection_update()` method is different from the `intersection()` method,**
- **because the `intersection()` method *returns a new set*, without the unwanted items, and the `intersection_update()` method *removes* the unwanted items from the original set.**

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.intersection_update(y)

print(x)
```

difference()

Set difference() Method

The `difference()` method returns a set that contains the difference between two sets.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.difference(y)

print(z)
```

difference_update()

Set difference_update() Method

the `difference_update()` method *removes* the unwanted items from the original set.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.difference_update(y)

print(x)
```

isdisjoint() mehtod

Set isdisjoint() Method

The `isdisjoint()` method returns True if none of the items are present in both sets, otherwise it returns False.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "facebook"}

z = x.isdisjoint(y)

print(z)
```

issubset() method

issubset() Method

The `issubset()` method returns True if all items in the set exists in the specified set, otherwise it returns False

```
x = {"a", "b", "c"}

y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)

print(z)
```

issuperset() method

issuperset() Method

The `issuperset()` method returns True if all items in the specified set exists in the original set, otherwise it returns False.

```
x = {"f", "e", "d", "c", "b", "a"}

y = {"a", "b", "c"}

z = x.issuperset(y)

print(z)
```

symmetric_difference()

symmetric_difference() Method

The `symmetric_difference()` method returns a set that contains all items from both set, but not the items that are present in both sets.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)

print(z)
```

symmetric_difference_update()

symmetric_difference_update() Method

The `symmetric_difference_update()` method updates the original set by removing items that are present in both sets, and inserting the other items.

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)

print(x)
```