Python Data Type: SET

Github: @Aaryacode19

Set

Sets are used to store multiple items in a single variable. A set is a collection which is unordered, unchangeable*, and unindexed. Sets are written with curly brackets.

Create a Set:

```python
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Set Items

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered

Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable

Set items are unchangeable, meaning that we cannot change the items after the set has been created.

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

Duplicates Not Allowed

Sets cannot have two items with the same value

Duplicate values will be ignored:

```python
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
```

Output:

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

Get the Length of a Set

To determine how many items a set has, use the len() function.

```python
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

Output:

```
3
```

Set Items - Data Types
String, int and boolean data types:

```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
set4 = {"abc", 34, True, 40, "male"}
```

Access Items

We cannot access items in a set by referring to an index or a key.
But we can loop through the set items using a **for loop**, or ask if a specified value is present in a set, by using the in keyword.

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
    print(x)
```

Output:

```
cherry
apple
banana
```

Search Items:

Check if "banana" is present in the set:

```
thisset = {"apple", "banana", "cherry"}
print("banana" in thisset)
```

Output:

```
True
```

Change Items
Once a set is created, you cannot change its items, but you can add new items.
To add one item to a set use the **add()** method.

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

Output:

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

Add Sets

To add items from another set into the current set, use the **update()** method.

```
frutis1 = {"apple", "banana", "cherry"}
fruits2 = {"pineapple", "mango", "papaya"}
fruits.update(tropical)

print(fruits)
```

Output:

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

Add Any Iterable

The object in the **update()** method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).

Add elements of a list to at set:

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
mytuple = (1,2,3,4)
thisset.update(mylist, mytuple)

print (thisset)
```

Output:

```
{1, 2, 3, 4, 'orange', 'cherry', 'kiwi', 'banana', 'apple'}
```

Remove Item

To remove an item in a set, use the remove(), or the discard() method.

Remove "banana" by using the **remove()** method:

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

Output:

```
{'apple', 'cherry'}
```

Remove "banana" by using the discard() method:

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

Output:

```
{'apple', 'cherry'}
```

You can also use the pop() method to remove an item, but this method will remove the *last* item. Remember that sets are unordered, so you will not know what item that gets removed.

The return value of the **pop()** method is the removed item.

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

x = thisset.pop()
print(x)  #removed item
print(thisset)  #the set after removal
```

Output:

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

The **clear()** method empties the set:

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

Output:

```
set()
```

The **del** keyword will delete the set completely

```
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)  #this will raise an error because the set no longer exists
```

Loop Items
You can loop through the set items by using a for loop:

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
   print(x)
```

Output:

```
apple
banana
cherry
```

Join Two Sets

There are several ways to join two or more sets in Python.

You can use the **union()** method that returns a new set containing all items from both sets, or the **update()** method that inserts all the items from one set into another:

The **union()** method returns a new set with all items from both sets:

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

Output:

```
{3, 'b', 'c', 1, 'a', 2}
```

The **update()** method inserts the items in set2 into set1:

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

Output:

```
{'c', 2, 3, 'b', 1, 'a'}
```

Keep ONLY the Duplicates

The **intersection_update()** method will keep only the items that are present in both sets.
Keep the items that exist in both set x, and set y:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.intersection_update(y)
print(x)
```

Output:

```
{'apple'}
```

The **intersection()** method will return a *new* set, that only contains the items that are present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)
print(z)
```

Output:

```
{'apple'}
```

Keep All, But NOT the Duplicates

The **symmetric_difference_update()** method will keep only the elements that are NOT present in both sets.

```
x = {"apple", "banana", "cherry", "orange"}
y = {"google", "microsoft", "apple", "orange"}

x.symmetric_difference_update(y)
print(x)
```

Output:

The **symmetric_difference()** method will return a new set, that contains only the elements that are NOT present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)
print(z)
```

Output:

```
{'google', 'banana', 'microsoft', 'cherry'}
```

Result: The practical has been successfully studied.