Python Sets

A Python Set is a Collection. Collections in Python include the following: List, Sets, Dictionary and Tuple. Let us now see the key points to understand Sets in Python:

- A Set is a collection in Python
- Set items i.e. elements are placed inside curly braces {}
- Unordered and Unchangeable
- Unchangeable, but you can add/ remove an item
- Duplicate values aren't allowed in a Set

Create a Set in Python

To create a set, place the elements inside curly brackets, separated by comma.

Let us see an example wherein we will create a Set:

```
# Create a Set in Python

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)
```

The output is as follows:

```
{'amit', 'john', 'kane', 'steve', 'warner'}
```

Create a set with different data types

In this example, we will learn how to create a set with different data types:

```
# Create a set with different datatypes

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)
```

```
myset2 = {5, 10, 15, 20, 25, 30}
print(myset2)

myset3 = {True, True, False, True, False, False}
print(myset3)

myset4 = {"amit", True, 65, 89, "john", "tim"}
print(myset4)
```

The output is as follows:

```
{'steve', 'john', 'amit', 'kane', 'warner'}
{20, 5, 25, 10, 30, 15}
{False, True}
{65, True, 'john', 89, 'amit', 'tim'}
```

Loop through the Set and Print all the values

In this example, loop through the set and display the entire set:

```
# Loop through the set and print all the values

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

for i in myset:
    print(i)
```

The output is as follows:

```
{'steve', 'john', 'kane', 'warner', 'amit'}
steve
john
kane
warner
amit
```

Get the length of a Set

In this example, we will get the length of a Set i.e. the total number of elements using the len() method:

```
# Get the length of a set

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

print(len(myset))
```

The output is as follows:

```
{'steve', 'warner', 'john', 'amit', 'kane'}
5
```

How to access items in Python Sets

In this example, we will learn how to access any item from a Set:

```
# Access items in Python Sets

myset = {"amit", "john", "kane", "warner", "steve"}

for i in myset:
    print(i)

print("amit" in myset)
print("jacob" in myset)
```

The output is as follows:

```
kane
warner
john
steve
amit
True
False
```

Add an item to a Python Set

In this example, we will learn how to add an item to a Set in Python:

```
# Add an item to a Python Set

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

myset.add("jacob")
myset.add("katie")

print("\nUpdated Set = ",myset)
```

The output is as follows:

```
{'john', 'kane', 'amit', 'steve', 'warner'}
```

```
Updated Set =  {'john', 'jacob', 'katie', 'kane', 'amit', 'stev
```

Add items from another set into the current set (Update)

In this example, we will learn how to add items from another set to the current set:

```
# Add items from another set into the current set (Update)

myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"katie", "jacob", "emma"}
print(myset2)

myset1.update(myset2)

print("\nUpdated Set1 = ",myset1)
```

The output is as follows:

```
{'kane', 'steve', 'warner', 'amit', 'john'}
{'jacob', 'katie', 'emma'}

Updated Set1 =  {'jacob', 'kane', 'katie', 'emma', 'steve', 'wa
```

Remove an item from the Set using remove()

In this example, we will remove an item from the Set using remove():

```
# Remove an item from the set using the remove() method

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

myset.remove("kane")

# removing an element not in the set
# myset.remove("jacob")

print("\nUpdated Set = ",myset)
```

The output is as follows:

```
{'john', 'amit', 'steve', 'warner', 'kane'}

Updated Set =  {'john', 'amit', 'steve', 'warner'}
```

Remove an item from the Set using discard()

In this example, we will remove an item from the Set using discard():

```
# Remove an item from the set using the discard() method

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

myset.discard("warner")

# no error thrown for removing an item not in the list
# myset.discard("jacob")

print("\nUpdated Set = ",myset)
```

The output is as follows:

```
{'kane', 'steve', 'amit', 'warner', 'john'}

Updated Set =  {'kane', 'steve', 'amit', 'john'}
```

Empty the Set

In this example, learn how to empty a Set in Python using the clear() method:

```
# Empty the Set

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

myset.clear()

print(myset)
```

The output is as follows:

```
{'steve', 'kane', 'john', 'warner', 'amit'}
set()
```

Delete the complete Set

In this example, learn how to delete a Set in Python using the del() method:

```
# Delete the complete set

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

del myset

# throws an error since we deleted the set above
# print(myset)
```

The output is as follows:

```
{'kane', 'steve', 'warner', 'amit', 'john'}
```

Join two Sets in Python

In this example, we will learn how to join two sets in Python:

```
# Join Two Sets in Python

myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"amy","brad", "tom"}
print(myset2)

myset1.update(myset2)
print(myset1)
```

The output is as follows:

```
{'steve', 'warner', 'kane', 'john', 'amit'}
{'amy', 'brad', 'tom'}
{'brad', 'steve', 'warner', 'amy', 'kane', 'tom', 'john', 'amit
```

Return the Difference between two or more sets

In this example, we will learn how to return the Difference between two or more sets using the difference() method:

```
# Return the difference between two or more sets

myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"jacob", "john", "mark", "anthony", "steve"}
print(myset2)

res = myset1.difference(myset2)
print(res)
```

The output is as follows:

```
{'kane', 'amit', 'john', 'steve', 'warner'}
{'anthony', 'mark', 'jacob', 'john', 'steve'}
{'kane', 'amit', 'warner'}
```

Keep only the duplicate items in two sets

In this example, we will keep only the duplicate items in two sets using the intersection_update() method in Python:

```
# Keep only duplicate items in two sets

myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"jacob", "john", "mark", "anthony", "steve"}
print(myset2)

myset1.intersection_update(myset2)
print(myset1)
```

The output is as follows:

```
{'steve', 'amit', 'warner', 'john', 'kane'}
{'jacob', 'steve', 'mark', 'anthony', 'john'}
{'steve', 'john'}
```

Keep all the items in the two sets except the duplicates

In this example, keep all the items in the two sets except the duplicates using the symmetric_difference_update() method in Python:

```
# Keep all the items in the two sets except the duplicates
```

```
myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"jacob", "john", "mark", "anthony", "steve"}
print(myset2)

myset1.symmetric_difference_update(myset2)
print(myset1)
```

The output is as follows:

```
{'amit', 'kane', 'warner', 'steve', 'john'}
{'mark', 'jacob', 'anthony', 'steve', 'john'}
{'mark', 'jacob', 'anthony', 'amit', 'kane', 'warner'}
```

Make a copy of the Set

To make a copy of the Set, use the copy() method:

```
# Make a copy of the Set

myset = {"amit", "john", "kane", "warner", "steve"}
print(myset)

res = myset.copy()
print(res)
```

The output is as follows:

```
{'amit', 'kane', 'warner', 'steve', 'john'}
{'amit', 'kane', 'warner', 'steve', 'john'}
```

Union of Sets

To get the Union of Sets in Python, use the union() method

```
# Union of Sets

myset1 = {"amit", "john", "kane", "warner", "steve"}
print(myset1)

myset2 = {"jacob", "john", "mark", "anthony", "steve"}
print(myset2)

res = myset1.union(myset2)
print(res)
```

The output is as follows:

```
{'kane', 'amit', 'john', 'steve', 'warner'}
```

```
{'anthony', 'john', 'steve', 'mark', 'jacob'}
{'john', 'steve', 'warner', 'kane', 'amit', 'anthony', 'mark',
```

In this tutorial, we learned what are Sets in Python. Moreover, we worked around creating Python Sets and their operations like adding, deleting, updating, union, joining, etc. Clearly, it's quite easy to work on Python Sets. Apart from this, if you are aware of a topic we missed in the Set, then kindly mention it in the comment section.

Recommended Posts

- Python Tuples
- Python Dictionary
- Python Lists
- Strings in Python
- Functions in Python
- DateTime Tutorial in Python
- Python Multiple Choice Questions (MCQs)