Python Sets

In Python, a **Set** is an unordered collection of data types that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set.

Creating a Set

Sets can be created by using the built-in **set()** function with an iterable object or a sequence by placing the sequence inside curly braces, separated by a 'comma'.
Note: *A set cannot have mutable elements like a list or dictionary, as it is mutable.*

```
# Python program to demonstrate

# Creation of Set in Python

set1 = set()

print("Initial blank Set: ")

print(set1)
```

```
# Creating a Set with a List of Numbers

# (Having duplicate values)

set1 = set([1, 2, 4, 4, 3, 3, 3, 6, 5])

print("\nSet with the use of Numbers: ")

print(set1)

# Creating a Set with a mixed type of values

# (Having numbers and strings)

set1 = set([1, 2, 'Disha', 4, 'Computer', 6, 'Institute'])

print("\nSet with the use of Mixed Values")

print(set1)
```

Creating a set with another method

```
# Another Method to create sets in Python3

my_set = {1, 2, 3}

print(my_set)
```

Adding Elements to a Set

Using add() method

Elements can be added to the Set by using the built-in **add()** function. Only one element at a time can be added to the set by using add() method, loops are used to add multiple elements at a time with the use of add() method.

Note: Lists cannot be added to a set as elements because Lists are not hashable whereas Tuples can be added because tuples are immutable and hence Hashable.

```
# Python program to demonstrate Addition of elements in a Set

# Creating a Set

set1 = set()

print("Initial blank Set: ")

print(set1)

# Adding element and tuple to the Set

set1.add(8)

set1.add(9)

set1.add((6, 7))

print("\nSet after Addition of Three elements: ")

print(set1)

for i in range(1, 6):

        set1.add(i)

print("\nSet after Addition of elements from 1-5: ")

print(set1)
```

Using update() method

For the addition of two or more elements Update() method is used. The update() method accepts lists, strings, tuples as well as other sets as its arguments. In all of these cases, duplicate elements are avoided.

```
# Python program to demonstrate Addition of elements to the Set

# using Update function

set1 = set([4, 5, (6, 7)])

set1.update([10, 11])

print("\nSet after Addition of elements using Update: ")

print(set1)
```

Accessing a Set

Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you 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.

```
# Python program to demonstrate

# Accessing of elements in a set

set1 = set(["Disha", "Computer", "Institute"])

print("\nInitial set")

print(set1)

print("\nElements of set: ")

for i in set1:

        print(i, end=" ")

print("Disha" in set1)
```

Removing elements from the Set

Using remove() method or discard() method:

Elements can be removed from the Set by using the built-in remove() function but a KeyError arises if the element doesn't exist in the set. To remove elements from a set without KeyError, use discard(), if the element doesn't exist in the set, it remains unchanged.

```
# Python program to demonstrate Deletion of elements in a Set

set1 = set([1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12])

print("Initial Set: ")

print(set1)

set1.remove(5)

set1.remove(6)

print("\nSet after Removal of two elements: ")

print(set1)


# Removing elements from Set using Discard() method

set1.discard(8)

set1.discard(9)

print("\nSet after Discarding two elements: ")

print(set1)

# Removing elements from Set using iterator method

for i in range(1, 5):

set1.remove(i)

print("\nSet after Removing a range of elements: ")

print(set1)
```

Using pop() method:

Pop() function can also be used to remove and return an element from the set, but it removes only the last element of the set.

Note: *If the set is unordered then there's no such way to determine which element is popped by using the pop() function.*

```
# Python program to demonstrate Deletion of elements in a Set

set1 = set([1, 2, 3, 4, 5, 6,7, 8, 9, 10, 11, 12])

print("Initial Set: ")

print(set1)

set1.pop()

print("\nSet after popping an element: ")

print(set1)
```

Using clear() method:

To remove all the elements from the set, clear() function is used.

```
#Creating a set

set1 = set([1,2,3,4,5])

print("\n Initial set: ")

print(set1)

set1.clear()

print("\nSet after clearing all the elements: ")

print(set1)
```

Frozen sets in Python are immutable objects that only support methods and operators that produce a result without affecting the frozen set or sets to which they are applied. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.
If no parameters are passed, it returns an empty frozenset.

```
# Python program to demonstrate working of a FrozenSet

# Creating a Set

String = ('D', 'i', 's', 'h', 'a', 'C', 'o', 'm',’p’,’u’,’t’,’e’,’r’)

Fset1 = frozenset(String)

print("The FrozenSet is: ")

print(Fset1)

print("\nEmpty FrozenSet: ")

print(frozenset())
```