17. PYTHON - SET DATA STRUCTURE

Table of Contents

17. PYTHON - SET DATA STRUCTURE

1. Set data structure

- ✓ We can **create** set by using,
  - Curly braces **{}** symbols
  - set() predefined function.

- ✓ A set can **store group** of objects or elements.
  - A set can store **same** (Homogeneous) type of elements.
  - A set can store **different** (Heterogeneous)type of elements.

- ✓ A set size will **increase** dynamically.

- ✓ In set insertion order is not preserved means **not fixed**.
  - If we insert elements into 10, 20, 30, 40 then there is no guarantee for output as 10, 20, 30, 40
  - Example
    - Input => {10, 20, 30, 40}
    - Output => {20, 40, 10, 30}

- ✓ Duplicate elements are **not allowed**.
- ✓ Set having **mutable** nature.
  - Mutable means once we create a set object then we can change or modify the content of set object.

- ✓ Set data structure **cannot** store the elements in index order.

Note:

- ✓ set is a predefined class in python.
- ✓ Once if we create set object means internally object is creating for set class.

Note:

- ✓ Inside set every object can be separated by comma separator.

2. When should we go for set?

- ✓ If we want to represent a group of **unique** values as a single entity, then we should go for set.
- ✓ set cannot store duplicate elements.
- ✓ Insertion order is not preserved.

Note

- ✓ We can create set by using curly braces {} and all elements separated by comma separator in set.

3. Creating set by using same type of elements

Program Name: creating same type of elements by using set
demo1.py

```
s = {10, 20, 30, 40}
print(s)
print(type(s))
```

Output

```
{40, 10, 20, 30}
<class 'set'>
```

4. Creating set by using different type of elements

Program Name: creating different type of elements by using set
demo2.py

```
s = {10, "Daniel", 30.9, "Prasad", 40}
print(s)
print(type(s))
```

Output

```
{40, 10, 'Daniel', 'Prasad', 30.9}
<class 'set'>
```

5. Creating set by range(p) type of elements

Program Name: creating set by using range(p)
demo3.py

```
s = set(range(5))
print(s)
```

output

```
{0, 1, 2, 3, 4}
```

Program Name: set not allowed duplicates
demo4.py

```
s ={10, 20, 30, 40, 10, 10}
print(s)
print(type(s))
```

Output

```
{40, 10, 20, 30}
<class 'set'>
```

Make a note

- ✓ Observe the above programs output,
  - Order is not preserved
  - Duplicates are not allowed.

6. Creating set by using set(p) predefined function

- ✓ We can create set by using set(p) function.
- ✓ set(p) predefined function will take only one parameter.
- ✓ This parameter should be sequence (range, list, set, tuple, etc...) object otherwise we will get error.

Program Name: creating set by using set(parameter1) function
demo5.py

```
r = range(0, 10)
l = set(r)
print(l)
```

output

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

7. Empty set

- ✓ We can create empty set.
- ✓ While creating empty set compulsory we should use set() function.
- ✓ If we didn't use set function, then it treats as dictionary instead of set.

Program Name: Empty curly braces is not a set, it considers as a dictionary
demo6.py

```
s = {}
print(s)
print(type(s))
```

Output

```
{}
<class 'dict'>
```

Program Name: Using set() function to create empty set
demo7.py

```
s = set()
print(s)
print(type(s))
```

Output

```
set()
<class 'set'>
```

8. len(p) function

- ✓ By using len(p) predefined function we can find the length of set.
- ✓ This function returns the number of elements present in the set.

Program Name: To find length of set
demo8.py

```
n = {10, 20, 30, 40, 50}
print(len(n))
```

Output

```
5
```

Program Name: To find length of set
demo9.py

```
n = {10, 20, 30, 40, 50, 10, 10, 10}
print(len(n))
```

Output

```
5
```

9. Methods in set data structure

- ✓ As discussed, set is a predefined class.
- ✓ So, set class can contain methods because methods can be created inside of class only.
- ✓ We can check these methods by using dir(parameter1) predefined function.

- ✓ So, internally set class contains two types of methods,

    - With underscore symbol methods.
        - We no need to focus

    - Without underscore symbol methods.
        - We need to focus much on these

Program Name	Printing set data structure methods by using dir(set) function demo10.py print(dir(set))
output	['__and__', ... '__subclasshook__', '__xor__', **Important methods** 'add', 'clear', 'remove',]

Important point

- ✓ As per object-oriented principle,
  - If we want to access instance method, then we should access by using object name.
- ✓ So, all set methods we can access by using set object.

Important methods in set:

- ✓ add(p) method
- ✓ remove(p) method
- ✓ clear() method

9.1. add(p) method:

- ✓ add(p) is a method, we should access this method by using set object.
- ✓ This method adds element to the set.

Program Name	add(p) function can add element to the set demo11.py
	s = {10, 20, 30} s.add(40) print(s)
Output	{40, 10, 20, 30}

9.2. remove(p) method

- ✓ remove(p) is a method in set class, we should access this method by using set object.
- ✓ This method removes specified element from the set.
- ✓ If the specified element not present in the set, then we will get KeyError

Program Name: remove(p) method in set
demo12.py

```
s = {40, 10, 30, 20}
s.remove(30)
print(s)
```

Output

```
{40, 10, 20}
```

9.3. clear() method

- ✓ clear() is a method in set class, we should access this method by using set object.
- ✓ This method removes all elements from the set.

Program Name: clear() method in set
demo13.py

```
s = {10,20,30}
print(s)
s.clear()
print(s)
```

Output

```
{10, 20, 30}
set()
```

10. Membership operators: (in and not in)

- ✓ By using these operators, we can find the specified element is exists in set or not

Program Name: in operator
demo14.py

```
s = {1, 2, 3, 'daniel'}

print(s)
print(1 in s)
print('z' in s)
```

Output

```
{1, 2, 3, 'daniel'}
True
False
```

11. set comprehensions

- ✓ set comprehensions represents creating new set from Iterable object like a list, set, tuple, dictionary and range.
- ✓ set comprehensions code is very concise way.

Program Name: set comprehension
demo15.py

```
s = {x*x  for x in range(5)}
print(s)
```

output

```
{0, 1, 4, 9, 16}
```

12. Remove duplicates in list elements

- ✓ We can remove duplicates elements which are exists in list by passing list as parameter to set function

Program Name: removing duplicates from list
demo18.py

```
a = [10, 20, 30, 10, 20, 40]
s = set(a)
print(s)
```

Output

```
{40, 10, 20, 30}
```