# UNIT - 5

**Set Data Type**

## Date: 06-05-2020 Day 1

**Topics Covered:**

### Introduction

1. **Creation of Set Objects**

### Important functions / methods of set

* 1. add()
  2. update()
  3. copy()
  4. pop()
  5. remove()
  6. discard()
  7. clear()

### Mathematical operations on the Set

1. **Membership operators**
2. in
3. not in

### Set Comprehension

1. **Introduction**

localhost:8888/notebooks/Desktop/COMPLETE PYTHON/PythonCourse/Set Data Type.ipynb#3.-Important-functions-/-methods--of-set 1/14

If we want to represent a group of unique values as a single entity then we should go for set.

#### Key features of Set Data Type:

* 1. Duplicates are not allowed.
  2. Insertion order is not preserved.But we can sort the elements.
  3. Indexing and slicing not allowed for the set.
  4. Heterogeneous elements are allowed.
  5. Set objects are mutable i.e once we creates set object we can perform any changes in that object based on our requirement.
  6. We can represent set elements within curly braces and with comma seperation.
  7. We can apply mathematical operations like union,intersection,difference etc on set objects.

### Creation of Set Objects

#### Creation of set object with single value

In [5]:

s **=** {10}

print(type(s)) print(s)

<class 'set'>

{10}

#### Creation of set object with multiple values

In [6]:

s **=** {30,40,10,5,20} *# in the output order not preserved*

print(type(s)) print(s)

<class 'set'>

{5, 40, 10, 20, 30}

In [10]:

s **=** {30,40,10,5,20} *# in the output order not preserved*

print(type(s)) print(s[0])

<class 'set'>

#### ---------------------------------------------------------------------------

**TypeError** Traceback (most recent call last)

**<ipython-input-10-655c69b5c557>** in <module>

#### s = {30,40,10,5,20} # in the output order not preserved

* 1. print**(**type**(**s**))**

**----> 3** print**(**s**[0])**

**TypeError**: 'set' object is not subscriptable

In [11]:

s **=** {30,40,10,5,20} *# in the output order not preserved*

print(type(s)) print(s[0:6])

<class 'set'>

#### ---------------------------------------------------------------------------

**TypeError** Traceback (most recent call last)

**<ipython-input-11-05c9c76958c2>** in <module>

#### s = {30,40,10,5,20} # in the output order not preserved

1. print**(**type**(**s**))**

**----> 3** print**(**s**[0:6])**

**TypeError**: 'set' object is not subscriptable

#### Creation of set objects using set() function

We can create set objects by using **set()** function.

#### Syntax:

**s=set(any sequence) Eg 1:**

In [7]:

l **=** [10,20,30,40,10,20,10]

s**=**set(l)

print(s)

*# {40, 10, 20, 30} because duplicates are not allowed in set*

{40, 10, 20, 30}

#### Eg 2:

In [9]:

s**=**set(range(5))

print(s) *#{0, 1, 2, 3, 4}*

{0, 1, 2, 3, 4}

#### Eg 3:

In [12]:

s **=** set('karthi') print(s)

{'a', 'h', 'i', 't', 'k', 'r'}

#### Eg 4:

In [15]:

s**=** set('aaabbbb') print(s)

{'a', 'b'}

#### Note:

While creating empty set we have to take special care. Compulsory we should use set() function.

s={} ==>It is treated as dictionary but not empty set.

#### Eg :

In [13]:

s **=** {}

print(type(s))

<class 'dict'>

#### Eg :

In [14]:

s **=** set() *# set function without any arguments*

print(s)

print(type(s))

set()

<class 'set'>

### Important functions / methods of set:

#### add(x):

Adds item x to the set

#### Eg :

In [31]:

s**=**{10,20,30}

s.add(40); print(s)

*# ';' is optional for python statements*

*#{40, 10, 20, 30}*

{40, 10, 20, 30}

In [32]:

s**=**{10,20,30}

s.add('karthi'); print(s)

*# ';' is optional for python statements*

{10, 'karthi', 20, 30}

#### update(x,y,z):

This method is used to add multiple items to the set.

Arguments are not individual elements and these are Iterable objects like List,range etc.

All elements present in the given Iterable objects will be added to the set.

#### Eg :

In [33]:

s**=**{10,20,30}

s.update('karthi'); print(s)

*# ';' is optional for python statements*

{'a', 10, 'h', 'i', 20, 't', 'k', 'r', 30}

In [23]:

s**=**{10,20,30}

l**=**[40,50,60,10]

s.update(l,range(5)) print(s)

{0, 1, 2, 3, 4, 40, 10, 50, 20, 60, 30}

In [24]:

s**=**{10,20,30}

l**=**[40,50,60,10]

s.update(l,range(5),100) print(s)

#### ---------------------------------------------------------------------------

**TypeError** Traceback (most recent call last)

**<ipython-input-24-d6e54bc11daa>** in <module> 1 s**={10,20,30}**

#### 2 l=[40,50,60,10]

**----> 3** s**.**update**(**l**,**range**(5),100)**

4 print**(**s**)**

**TypeError**: 'int' object is not iterable

In [25]:

s**=**{10,20,30}

l**=**[40,50,60,10]

s.update(l,range(5),'100') print(s)

{0, 1, 2, 3, 4, 40, 10, '0', '1', 50, 20, 60, 30}

In [26]:

s**=**{10,20,30}

l**=**[40,50,60,10]

s.update(l,range(5),'karthi') print(s)

{0, 1, 2, 3, 4, 'a', 40, 10, 'h', 'i', 50, 20, 't', 'k', 'r', 60, 30}

In [18]:

s **=**set()

s.update(range(1,10,2),range(0,10,2)) print(s)

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

#### Q. What is the difference between add() and update() functions in set?

We can use add() to add individual item to the Set,where as we can use update() function to add multiple items to Set.

add() function can take only one argument where as update() function can take any number of arguments but all arguments should be iterable objects.

#### Q. Which of the following are valid for set s?

1. s.add(10) **Valid**
2. s.add(10,20,30) **TypeError: add() takes exactly one argument (3 given)**
3. s.update(10) **TypeError: 'int' object is not iterable**
4. s.update(range(1,10,2),range(0,10,2)) **Valid**

#### copy():

Returns copy of the set. It is cloned object (Backup copy).

#### Eg :

In [20]:

s**=**{10,20,30}

s1**=**s.copy() print(s1) print(s)

{10, 20, 30}

{10, 20, 30}

#### pop():

It removes and returns some random element from the set.

#### Eg :

In [38]:

s**=**{40,10,30,20}

print(s)

print(s.pop())

print(s.pop())

print(s.pop()) print(s)

print(s.pop())

print(s) *# empty set*

print(s.pop())

{40, 10, 20, 30}

40

10

20

{30}

30

set()

#### ---------------------------------------------------------------------------

**KeyError** Traceback (most recent call last)

**<ipython-input-38-22f4166ffe90>** in <module>

1. print**(**s**.**pop**())**
2. print**(**s**) # empty set**

**----> 9** print**(**s**.**pop**())**

**KeyError**: 'pop from an empty set'

#### Consider the following case :

s**=**{40,10,30,20}

print(s)

print(s.pop())

print(s.pop()) print(s)

{40, 10, 20, 30}

40

10

{20, 30}

In [40]:

s**=**{40,10,30,20}

print(s)

print(s.pop())

print(s.pop()) print(s)

{40, 10, 20, 30}

40

10

{20, 30}

In [41]:

s**=**{40,10,30,20}

print(s)

print(s.pop())

print(s.pop()) print(s)

{40, 10, 20, 30}

40

10

{20, 30}

#### Note :

How many times you may execute the code, the elements which are popped from the set in same order. The reason is ---

All the elements of set are inserted based on some hashcode.If that order is fixed then it is always going to return one by one. But in which order these elements are inserted we don't know.

#### remove(x):

It removes specified element from the set.

If the specified element not present in the Set then we will get **KeyError**.

#### Eg :

s**=**{40,10,30,20}

s.remove(30)

print(s) *# {40, 10, 20}*

s.remove(50) *# KeyError: 50*

{40, 10, 20}

#### ---------------------------------------------------------------------------

**KeyError** Traceback (most recent call last)

**<ipython-input-30-fd29f2336f3b>** in <module>

1. s**.**remove**(30)**

3 print**(**s**) # {40, 10, 20}**

**----> 4** s**.**remove**(50) #KeyError: 50**

**KeyError**: 50

## Date: 07-05-2020 Day 2

#### discard(x):

It removes the specified element from the set. If the specified element not present in the set then we won't get any error.

In [1]:

s**=**{10,20,30}

s.discard(10) print(s)

s.discard(50) print(s)

*#{20, 30}*

*#{20, 30}*

{20, 30}

{20, 30}

#### Answer the following :

**Q. What is the difference between remove() and discard() functions in Set?**

#### Q. Explain differences between pop(),remove() and discard() functionsin Set?

**7.clear():**

To remove all elements from the Set.

In [2]:

s**=**{10,20,30}

print(s) s.clear() print(s)

{10, 20, 30}

set()

**4. Mathematical operations on the Set**

#### 1.union():

x.union(y) ==> We can use this function to return all elements present in both x and y sets wecan perform union operation in two ways:

1. **x.union(y)** ==> by calling through union() method.
2. **x|y** ==> by using '|' operator.

This operation returns all elements present in both sets x and y (without duplicate elements).

#### Eg :

In [3]:

|  |  |  |
| --- | --- | --- |
| x**=**{10,20,30,40}  y**=**{30,40,50,60}  print(x.union(y)) *#{10, 20, 30, 40, 50, 60}* | | |
| print(x**|**y) *#{10,* | *20,* | *30, 40, 50, 60}* |
| {40, 10, 50, 20, | 60, | 30} |
| {40, 10, 50, 20, | 60, | 30} |
| **2. intersection():** |  |  |

wecan perform intersection operation in two ways:

1. **x.intersection(y)** --> by calling through intersection() method.
2. **x&y** --> by using '&' operator.

This operation returns common elements present in both sets x and y.

#### Eg :

In [4]:

x**=**{10,20,30,40}

y**=**{30,40,50,60}

print(x.intersection(y)) *#{40, 30}*

print(x**&**y) *#{40, 30}*

{40, 30}

{40, 30}

#### difference():

wecan perform difference operation in two ways:

1. **x.difference(y)** --> by calling through difference() method.
2. **x-y** --> by using '-' operator.

This operation returns the elements present in x but not in y.

#### Eg :

In [5]:

x**=**{10,20,30,40}

y**=**{30,40,50,60}

print(x.difference(y)) *#{10, 20}*

print(x**-**y) *#{10, 20}*

print(y**-**x) *#{50, 60}*

{10, 20}

{10, 20}

{50, 60}

#### 4.symmetric\_difference():

wecan perform symmetric\_difference operation in two ways:

1. **x.symmetric\_difference(y)** --> by calling through symmetric\_difference method.
2. **x^y** --> by using '^' operator.

This operation returns elements present in either x or y but not in both.

#### Eg :

In [6]:

x**=**{10,20,30,40}

y**=**{30,40,50,60}

print(x.symmetric\_difference(y)) *#{10, 50, 20, 60}*

print(x**^**y) *#{10, 50, 20, 60}*

{10, 50, 20, 60}

{10, 50, 20, 60}

### 5. Membership operators:

Membership operators are used to check whether a particular object is available or not. For any sequence, we can apply membership operators.

Follwoing are the membership operators:

#### in

1. **not in**

#### Eg :

In [8]:

s**=**set("karthi") print(s)

print('a' **in** s)

print('z' **in** s)

{'r', 'a', 'i', 't', 'k', 'h'}

True False

### 6. Set Comprehension

Set comprehension is possible.

#### Syntax:

**s = {expression for x in sequence condition}**

In [9]:

s **=** {x**\***x

print(s)

**for** x **in** range(6)}

{0, 1, 4, 9, 16, 25}

In [10]:

s**=**{2**\*\***x **for** x **in** range(2,10,2)} print(s)

{16, 256, 64, 4}

**Note :**

### set objects won't support indexing and slicing.

#### Eg:

s**=**{10,20,30,40}

print(s[0])

print(s[1:3])

*#TypeError: 'set' object does not support indexing*

*#TypeError: 'set' object is not subscriptable*

**---------------------------------------------------------------------------**

**TypeError** Traceback (most recent call last)

**<ipython-input-11-6f6a7552f39c>** in <module> 1 s**={10,20,30,40}**

#### ----> 2 print(s[0]) #TypeError: 'set' object does not support in dexing

1. print**(**s**[1:3]) #TypeError: 'set' object is not subscriptabl**

#### e

**TypeError**: 'set' object is not subscriptable

### Example Programs

#### Q 1. Write a program to eliminate duplicates present in the list. Approach-1:

In [12]:

l**=**eval(input("Enter List of values: ")) s**=**set(l)

print(s)

Enter List of values: 10,20,30,10,20,40

{40, 10, 20, 30}

#### Approach-2:

In [13]:

l**=**eval(input("Enter List of values: ")) l1**=**[]

**for** x **in** l:

**if** x **not in** l1: l1.append(x)

print(l1)

Enter List of values: 10,20,30,10,20,40 [10, 20, 30, 40]

#### Q. Write a program to print different vowels present in the given word.

w**=**input("Enter word to search for vowels: ") s**=**set(w)

v**=**{'a','e','i','o','u'}

d**=**s.intersection(v)

print("The different vowel present in",w,"are",d) print('The number of different vowels : ',len(d))

Enter word to search for vowels: Learning python is very easy

The different vowel present in Learning python is very easy are {'o', 'a', 'i', 'e'}

The number of different vowels : 4

In [