Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

SET DATA TYPE - 1

UNIT – V:

Sets: Creation of set objects, Accessing the elements of set. Important functions of set –add(), update(), copy(), pop(),remove(),discard(),clear(). Basic Operations on set - Mathematical Operators for set objects, Membership operators on list, Set Comprehensions. Illustrative examples on all the above topics.

Topics Covered:

1. Introduction
2. Creation of Set Objects
3. Important functions / methods of set
    1. add()
    2. update()
    3. copy()
    4. pop()
    5. remove()
    6. discard()
    7. clear()
4. Mathematical operations on the Set
5. Membership operators
    i) in
    ii) not in
6. Set Comprehension

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. Introduction
2. Creation of Set Objects
3. Important functions / methods of set datatype
    i. add()
    ii. update()

1.INTRODUCTION

SET DATATYPE:

- 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 separation.

7. We can apply mathematical operations like union,intersection,difference etc., on set objects.

2. Creation of Set Objects

i) Creation of set object with single value

Eg:

```
s = {10}
print(type(s))
print(s)
```

Output:

```
<class 'set'>
{10}
```

ii) Creation of set object with multiple values

Eg:

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

print(type(s))

print(s)
```

Output:

```
<class 'set'>

{5, 40, 10, 20, 30}
```

Eg:

```
s = {30,40,10,5,20}          # in the output order not preserved
print(type(s))
print(s[0])
```

Output:

```
<class 'set'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-655c69b5c557> in <module>
      1 s = {30,40,10,5,20}    # in the output order not preserved
      2 print(type(s))
----> 3 print(s[0])

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

Eg:

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

print(type(s))

print(s[0:6])

Output:

```
<class 'set'>

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-05c9c76958c2> in <module>
      1 s = {30,40,10,5,20}   # in the output order not preserved
      2 print(type(s))
----> 3 print(s[0:6])

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

iii) Creation of set objects using set() function

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

Syntax:

s=set(any sequence)

Eg1:

```
l = [10,20,30,40,10,20,10]
s=set(l)
print(s)          # {40, 10, 20, 30} because duplicates are not allowed in set
```

Output:

{40, 10, 20, 30}

Eg 2:

```
s=set(range(5))
print(s)              #{0, 1, 2, 3, 4}
```

Output:

{0, 1, 2, 3, 4}

Eg 3:

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

Output:

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

Eg 4:

```
s= set('aaabbbb')

print(s)
```

Output:

```
{'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:

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

Output:

```
<class 'dict'>
```

Eg:

```
s = set() # set function without any arguments
print(s)
print(type(s))
```

Output:

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

3. Important functions / methods of set datatype

1. add(x):

- Adds item **x** to the set.

Eg:

```
s={10,20,30}
s.add(40);          # ';' is optional for python statements
print(s)            #{40, 10, 20, 30}
```

Output:

{40, 10, 20, 30}

Eg:

```
s={10,20,30}
s.add('karthi');       # ';' is optional for python statements
print(s)
```

Output:

```
{10, 'karthi', 20, 30}
```

2. 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 String, List, range etc.
- All elements present in the given Iterable objects will be added to the set.

Eg:

```
s={10,20,30}
s.update('karthi');        # ';' is optional for python statements
print(s)
```

Output:

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

Eg:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5))
print(s)
```

Output:

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

Eg:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),100)
print(s)
```

Output:

```
---------------------------------------------------------------------------
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
```

Eg:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),'100')
print(s)
```

Output:

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

Eg:

```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5),'karthi')
print(s)
```

Output:

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

Eg:

```
s =set()

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

print(s)
```

Output:

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

Frequently Asked Questions:

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 the 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

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You