



SNS COLLEGE OF ENGINEERING

Kurumbapalayam (PO), Coimbatore - 641 107

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

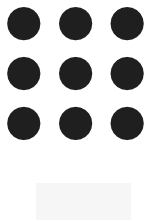
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: 23ITB202-PYTHON PROGRAMMING

II YEAR/ III SEM

Unit : LISTS, TUPLES, DICTIONARIES

Topic : SET





Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}


Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}

Sets
{1, 2, 3, 4}



Set = { "st

Sets:

- A set is an unordered collection of objects, unlike sequence objects such as lists and tuples
 - Sets cannot have duplicate members - a given object appears in a set 0 or 1 times
 - All members of a set have to be hashable, just like dictionary keys
 - Integers, floating point numbers, tuples, and strings are hashable
 - dictionaries, lists, and other sets (except **frozensets**) are not.
-
- Eg: `set(['b', 'e', 'o', 's', 'u', 't'])`
`set([32, 26, 12, 54])`

Constructing Sets:

- One way to construct sets is by passing any sequential object to the "set" constructor

```
>>> set([0, 1, 2, 3])
set([0, 1, 2, 3])
>>> set("obtuse")
set(['b', 'e', 'o', 's', 'u', 't'])
```

- Add elements to sets

```
>>> s = set([12, 26, 54])
>>> s.add(32)
>>> s
set([32, 26, 12, 54])
```

- Update set:

```
>>> s.update([26, 12, 9, 14])
>>> s
set([32, 9, 12, 14, 54, 26])
```

- Copy set:

- The set function also provides a copy constructor. However, remember that the copy constructor will copy the set, but not the individual elements

```
>>> s2 = s.copy()
>>> s2
set([32, 9, 12, 14, 54, 26])
```

Membership Testing:

- We can check if an object is in the set using the same "in" operator as with sequential data types

```
>>> 32 in s
True
>>> 6 in s
False
>>> 6 not in s
True
```

- We can also test the membership of entire sets. Given two sets `s1` and `s2`, we check if `s1` is a subset or a superset of `s2`

```
>>> s.issubset(set([32, 8, 9, 12, 14, -4, 54, 26, 19]))
True
>>> s.issuperset(set([9, 12]))
True
```

- Note that the `<=` and `>=` operators also express the `issubset` and `issuperset` functions respectively.

```
>>> set([4, 5, 7]) <= set([4, 5, 7, 9])
True
>>> set([9, 12, 15]) >= set([9, 12])
True
```

Removing Items:

- There are three functions which remove individual items from a set, called `pop`, `remove`, and `discard`
- The first, `pop`, simply removes an item from the set

```
>>> s = set([1, 2, 3, 4, 5, 6])
>>> s.pop()
1
>>> s
set([2, 3, 4, 5, 6])
```


- Note that the `<=` and `>=` operators also express the `issubset` and `issuperset` functions respectively.

```
>>> set([4, 5, 7]) <= set([4, 5, 7, 9])
True
>>> set([9, 12, 15]) >= set([9, 12])
True
```

Removing Items:

- There are three functions which remove individual items from a set, called `pop`, `remove`, and `discard`
- The first, `pop`, simply removes an item from the set

```
>>> s = set([1, 2, 3, 4, 5, 6])
>>> s.pop()
1
>>> s
set([2, 3, 4, 5, 6])
```

- We also have the "remove" function to remove a specified element.

```
>>> s.remove(3)
>>> s
set([2, 4, 5, 6])
```

- However, removing a item which isn't in the set causes an error.

```
>>> s.remove(9)
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in ?
```

```
KeyError: 9
```

- We also have another operation for removing elements from a set, **clear**, which simply removes all elements from the set

```
>>> s.clear()
```

```
>>> s
```

```
set([])
```



Iteration Over Sets:

- We can also have a loop move over each of the items in a set.
- However, since sets are unordered, it is undefined which order the iteration will follow.

```
>>> s = set("blerg")
```

```
>>> for n in s:
```

```
...     print n,
```

```
...
```

```
r b e l g
```

Set Operations:

- Python allows us to perform all the standard mathematical set operations, using members of set

- Union

The **union** is the merger of two sets. Any element in **s1** or **s2** will appear in their union

```
>>> s1 = set([4, 6, 9])
```

```
>>> s2 = set([1, 6, 8])
```

```
>>> s1.union(s2)
```

```
set([1, 4, 6, 8, 9])
```

```
>>> s1 | s2
```

```
set([1, 4, 6, 8, 9])
```

- Intersection

- Any element which is in both `s1` and `s2` will appear in their intersection

```
>>> s1 = set([4, 6, 9])
>>> s2 = set([1, 6, 8])
>>> s1.intersection(s2)
set([6])
>>> s1 & s2
set([6])
>>> s1.intersection_update(s2)
>>> s1
set([6])
```

- Symmetric Difference

- The **symmetric difference** of two sets is the set of elements which are in one of either set, but not in both

```
>>> s1 = set([4, 6, 9])
>>> s2 = set([1, 6, 8])
>>> s1.symmetric_difference(s2)
set([8, 1, 4, 9])
>>> s1 ^ s2
set([8, 1, 4, 9])
>>> s1.symmetric_difference_update(s2)
>>> s1
set([8, 1, 4, 9])
```

- Set Difference

- Python can also find the **set difference** of **s1** and **s2** , which is the elements that are in **s1** but not in **s2**.

```
>>> s1 = set([4, 6, 9])
>>> s2 = set([1, 6, 8])
>>> s1.difference(s2)
set([9, 4])
>>> s1 - s2
set([9, 4])
>>> s1.difference_update(s2)
>>> s1
set([9, 4])
```

Multiple sets:

- Starting with Python 2.6, "**union**", "**intersection**", and "**difference**" can work with multiple input by using the **set** constructor. For example,

```
>>> s1 = set([3, 6, 7, 9])
>>> s2 = set([6, 7, 9, 10])
>>> s3 = set([7, 9, 10, 11])
>>> set.intersection(s1, s2, s3)
set([9, 7])
```

Frozenset:

- A **frozenset** is basically the same as a set, except that it is immutable - once it is created, its members cannot be changed
- Since they are immutable, they are also hashable, which means that **frozensets** can be used as members in other sets and as dictionary keys
- frozensets have the same functions as normal sets, except none of the functions that change the contents (**update**, **remove**, **pop**, etc.) are available

```
>>> fs = frozenset([2, 3, 4])
>>> s1 = set([fs, 4, 5, 6])
>>> s1
set([4, frozenset([2, 3, 4]), 6, 5])
>>> fs.intersection(s1)
frozenset([4])
>>> fs.add(6)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute
'add'
```

