Sets , Frozen Set, Set Comprehensioon

Dr Venkat Raju

Features of Python Sets

- Sets are a **mutable** collection of **unique** values
- Values are **unordered**
- Does **not support indexing**
- Highly useful to efficiently **remove duplicate values from a list or tuple**
- Perform common math operations like **unions and intersections**

Set Creation and Initialization

- To declare a set, type a sequence of items separated by commas, inside curly braces { }

  and assign it to a variable

- Also by using **set()** built in function

- contain values of different types

- A set is mutable, but may not contain items like a list, set, or dictionary.

Set Creation and Initialization

```
s1={1,2.0,'three'}
print(s1)     # {1, 2.0, 'three'}


s2=set()
print(type(s2))  # <class 'set'>


# sets from lits
s3= set(['Python', ‘sets', 'are', 'mutable'])
print(s3)   #{'Python', 'are', 'mutable', ‘sets'}
```

```
1  s1={1,2.0,'three'}
2  s1
```

```
{1, 2.0, 'three'}
```

```
1  s2=set()
2  print(type(s2))
```

```
<class 'set'>
```

```
1  # sets from lits
2  s = set(['Python', 'sets', 'are', 'mutable'])
3  s
```

```
{'Python', 'are', 'mutable', 'sets'}
```

Imp points on sets

- since sets do not support indexing, they cannot be sliced

  s[:]

- Because a set isn’t indexed, can’t delete an element using its index.

```
# cannot contain duplicate elements.
s3={3,2,1,2}
print(s3)  # {1, 2, 3}

# Accessing a Set in Python
s1={1,2.0,'three'}
print(s1) # {1, 2.0, 'three'}
```

```
1  # cannot contain duplicate elements.
2  s3={3,2,1,2}
3  s3
```

```
{1, 2, 3}
```

```
1  # Accessing a Set in Python
2  s1={1,2.0,'three'}
3  s1
```

```
{1, 2.0, 'three'}
```

Adding elements

- Adding elements can ne done in two ways. 1. add() 2. update()
- To add or remove values from a set, Initialize it first
- To add single element using the add() method and multiple elements using the update() method.
- update() : can take tuples, lists, strings or other sets as its argument

  (duplicates are avoided)

Adding elements

s4 = {3,2,1,4,4,6,5}
print(s4) # {1, 2, 3, 4, 5, 6}

s4.add(3.5)
print(s4) # {1, 2, 3, 3.5, 4, 5, 6}

s4.add(4)
print(s4) # {1, 2, 3, 3.5, 4, 5, 6}

s4.update([7,8],{1,2,9})
print(s4) # {1, 2, 3, 3.5, 4, 5, 6, 7, 8, 9}

```
1  s4 = {3,2,1,4,4,6,5}
2  print(s4)
```

```
{1, 2, 3, 4, 5, 6}
```

```
1  s4.add(3.5)
2  s4
```

```
{1, 2, 3, 3.5, 4, 5, 6}
```

```
1  s4.add(4)
2  s4
```

```
{1, 2, 3, 3.5, 4, 5, 6}
```

```
1  s4.update([7,8],{1,2,9})
2  s4
```

```
{1, 2, 3, 3.5, 4, 5, 6, 7, 8, 9}
```

Removing elements

- To remove an element from set 1. remove() 2 discard() 3 pop() 4 clear()
- Difference 1. remove() 2 discard()
    - while using discard() if the item does not exist in the set, it remains unchanged
    - remove() will raise an error in such condition.
- pop() : Remove and return an arbitrary value from a set
- clear(): remove all values from a set

Removing elements

```
s4 = {3,2,1,4,4,6,5}
s4.discard(3)
print(s4)

s4.remove(6)
print(s4)

s4.pop()
print(s4)

s4.clear()
print(s4)
```

```
1  s4 = {3,2,1,4,4,6,5}
2  s4.discard(3)
3  print(s4)
```

```
{1, 2, 4, 5, 6}
```

```
1  #s4.remove(10)
2  #s4
```

```
1  s4.remove(6)
2  print(s4)
```

```
{1, 2, 4, 5}
```

```
1
```

```
1  s4.pop()
2  print(s4)
```

```
{2, 4, 5, 6}
```

```
1  s4.clear()
2  print(s4)
```

```
set()
```

Iterating over set using for loop

ds = {'Python', 'R', 'SQL', 'Tableau', 'SAS','ML','DL'}

for skillset in ds:

    print(skillset)

```
1  ds = {'Python', 'R', 'SQL', 'Tableau', 'SAS','ML','DL'}
2  for skillset in ds:
3     print(skillset)
```

```
DL
R
SQL
SAS
Tableau
ML
Python
```

Removing duplicates

- Use a **set** to remove duplicates from a list.

```
print(list(set([1,2,3,4,5,6,7,8,9,1,2,3,4])))
```

Removing duplicates from list

```
1  print(list(set([1,2,3,4,5,6,7,8,9,1,2,3,4])))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Iterating over set using for loop

ds = {'Python', 'R', 'SQL', 'Tableau', 'SAS','ML','DL'}

for skillset in ds:

    print(skillset)

```
1  ds = {'Python', 'R', 'SQL', 'Tableau', 'SAS','ML','DL'}
2  for skillset in ds:
3     print(skillset)
```

```
DL
R
SQL
SAS
Tableau
ML
Python
```

Removing duplicates

- Use a **set** to remove duplicates from a list.

```
print(list(set([1,2,3,4,5,6,7,8,9,1,2,3,4])))
```

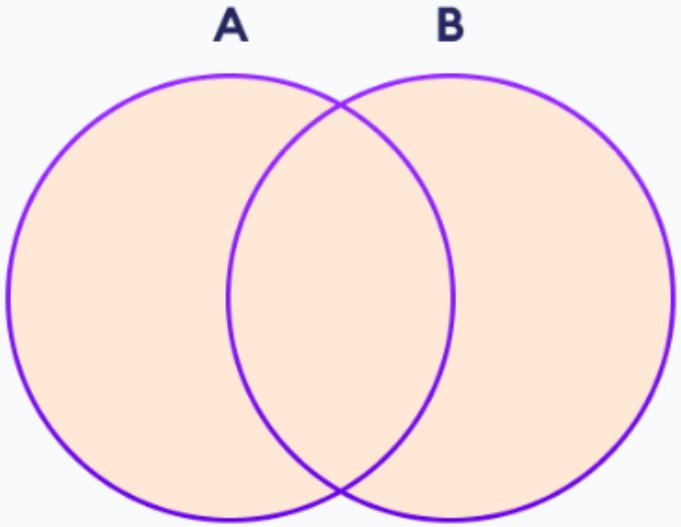
Removing duplicates from list

```
1  print(list(set([1,2,3,4,5,6,7,8,9,1,2,3,4])))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Set Union

The union of two sets **A** and **B** include all the elements of set **A** and **B**.

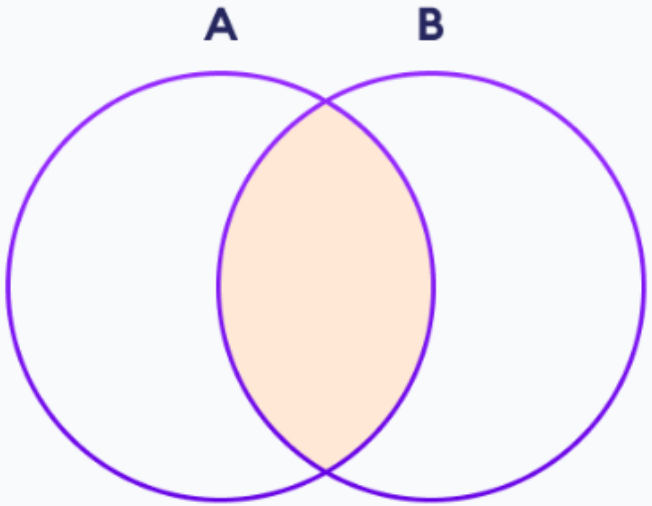


Set Union in Python

Note: `A|B` and `union()` is equivalent to `A U B` set operation.

Set Intersection

The intersection of two sets **A** and **B** include the common elements between set **A** and **B**.

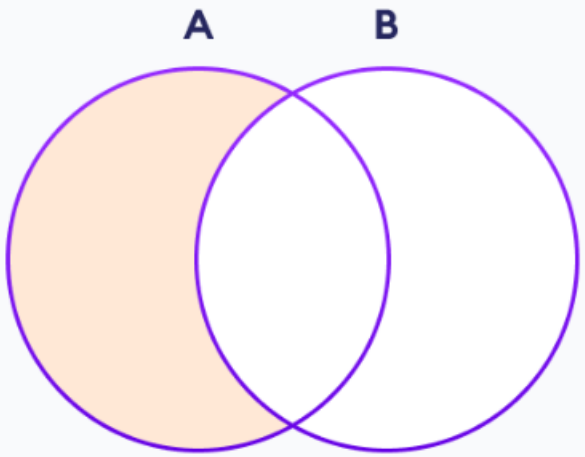


Set Intersection in Python

Note: `A&B` and `intersection()` is equivalent to `A ∩ B` set operation.

Difference between Two Sets

The difference between two sets **A** and **B** include elements of set **A** that are not present on set **B.**

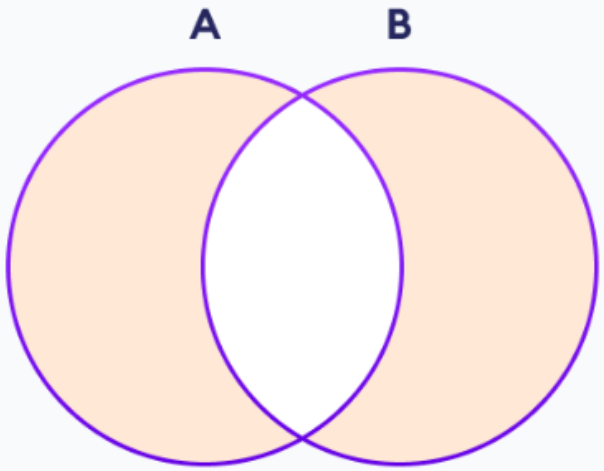


Set Difference in Python

Note: `A - B` and `A.difference(B)` is equivalent to `A - B` set operation.

Set Symmetric Difference

The symmetric difference between two sets **A** and **B** includes all elements of **A** and **B** without the common elements.



Set Symmetric Difference in Python

In Python, we use the ^ operator or the symmetric_difference() method to perform symmetric difference between two sets.

Frozen Set Objects

Features of Frozen Sets

- Frozen set is just an immutable version of a Python set object.
- While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.
- Due to this, frozen sets can be used as keys in Dictionary or as elements of another set.
- But like sets, it is not ordered (the elements can be set at any index).

Frozen set

- The frozenset() function takes a single parameter:

- iterable (Optional) - the iterable which contains elements to initialize the frozenset with.
  - Iterable can be set, dictionary, tuple, etc.

- The frozenset() function returns an immutable frozenset initialized with elements from the given iterable.

- If no parameters are passed, it returns an empty frozenset.

Set Comprehension

Set comprehension

- A set comprehension is like a list comprehension

  returns a **set**

```
s3 = {s for s in range(11) if s % 2}

print(s3)
```