Sets kya hota hai?

Sets Python mein ek aisa data structure hai jo unique elements ko store karta hai, yaani duplicate values allow nahi karta. Set mein elements unordered hote hain aur ye indexing ya slicing support nahi karta.

- Set ka basic syntax:

```
my_set = {1, 2, 3, 4}
```

- Set ko banane ke liye {} ka use hota hai, aur aap elements ko comma se separate karte hain.

Set ke kuch important features:

- Unordered: Elements kisi specific order mein nahi store hote.
- Unique elements: Duplicate elements ko ignore kar diya jata hai.
- Mutable: Set ke elements ko add ya remove kiya ja sakta hai.

set operators

Sets ke upar kaam karne ke liye Python kuch special operators provide karta hai jo mathematical set operations jaise union, intersection, difference, symmetric difference ko perform karne ke liye use hote hain.

Ye do sets ke sabhi unique elements ko ek saath merge karta hai.

- 1 (|) (union Operator)
- Inbuilt method: .union()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result_operator = set1 | set2  # {1, 2, 3, 4, 5}

# use inbuilt method:
result_method = set1.union(set2)  # {1, 2, 3, 4, 5}
```

Ye do sets ke common elements ko return karta hai.

- 2 (&) (Intersection Operator)
- Inbuilt method: .intersection()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result_operator = set1 & set2  # {3}

# use inbuilt method:
result_method = set1.intersection(set2)  # {3}
```

Ye first set se second set ke elements hata deta hai.

- 3 (-) (Difference Operator):
- Inbuilt method: .difference()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result_operator = set1 - set2  # {1, 2}

# use inbuilt method:
result_method = set1.difference(set2)  # {1, 2}
```

Ye dono sets ke uncommon elements ko return karta hai (jo ek set me hain par doosre me nahi).

- 4 (^) (Symmetric Difference Operator)
- Inbuilt method: .symmetric_difference()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result_operator = set1 ^ set2  # {1, 2, 4, 5}

# use inbuilt method:
result_method = set1.symmetric_difference(set2)  # {1, 2, 4, 5}
```

Agar ek set doosre set ka subset hai, toh ye True return karega.

- 5 (<=) (Subset Operator)
- Inbuilt method: .issubset()
- Example

```
set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
result_operator = set1 <= set2  # True
```

```
# use inbuilt method:
result_method = set1.issubset(set2)  # True
```

Agar ek set doosre set ka superset hai, toh ye True return karega.

- 6 (>=) (Superset Operator)
- Inbuilt method: .issuperset()
- Example

```
set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
result_operator = set2 >= set1  # True

# use inbuilt method:
result_method = set2.issuperset(set1)  # True
```

Agar do sets ke sabhi elements same hain (order doesn't matter), toh == operator True return karega.

- 7 (==) (Equality Operator)
- Inbuilt method: .__eq__()
- Example

```
set1 = {1, 2, 3}
set2 = {1, 2, 3}
result_operator = set1 == set2  # True

# use inbuilt method:
result_method = set1.__eq__(set2)  # True
```

Agar do sets ke elements alag hain, toh != True return karega.

- (!=) (Inequality Operator)
- Inbuilt method: __ne__()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
result_operator = set1 != set2  # True

# use inbuilt method:
result_method = set1.__ne__(set2)  # True
```

Summary:

- Operators jaise |, &, -, ^, <=,>=, ==, != directly likhe jaate hain.
- Unke corresponding inbuilt methods bhi hote hain jaise union(), intersection(), difference(), symmetric_difference(), issubset(), issuperset(), __eq__(), etc.
- Tum dono ka use kar sakte ho, par generally operators ko prefer kiya jata hai kyunki yeh zyada concise aur readable hote hain.

sets ke important methods:

- 1 add()
- Purpose: Is method se tum ek single element ko set mein add kar sakte ho.
- Syntax: .add()
- Example

```
my_set = {1, 2, 3}
my_set.add(4)
print(my_set)  # Output: {1, 2, 3, 4}
```

- 2 remove()
- Purpose: Is method se tum set se ek specific element ko remove kar sakte ho. Agar element set mein nahi hai, toh yeh KeyError raise karega.
- Syntax: .remove()
- Example

```
my_set = {1, 2, 3, 4}
my_set.remove(3)
print(my_set)  # Output: {1, 2, 4}
```

- 3 discard()
- Purpose: Is method se tum set se specific element ko remove kar sakte ho. Agar element set mein nahi hai, toh KeyError nahi aayega, bas koi effect nahi hoga.
- Syntax: .discard()
- Example

```
my_set = {1, 2, 3, 4}
my_set.discard(5)  # No error, even though 5 is not in the set
print(my_set)  # Output: {1, 2, 3, 4}
```

- 4 clear()
- Purpose: Is method se tum poore set ko clear kar sakte ho, matlab set ko empty bana sakte ho.
- Syntax:
- Example

```
my_set = {1, 2, 3}
my_set.clear()
print(my_set)  # Output: set()
```

- 5 copy()
- Purpose: Is method se tum set ka shallow copy bana sakte ho. Matlab original set ko modify karte waqt copied set unaffected rahega.
- Syntax: .copy()
- Example

```
my_set = {1, 2, 3}
new_set = my_set.copy()
print(new_set)  # Output: {1, 2, 3}
```

- 6 update()
- Purpose: Is method se tum ek set ko dono sets ke elements se update kar sakte ho. Yeh tumhe ek set ke saath ek aur set, list, tuple, ya koi iterable object add karne ki suvidha deta hai.
- Syntax: .update()
- Example

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1.update(set2)
print(set1)  # Output: {1, 2, 3, 4, 5}
```

- 7 issubset()
- Purpose: Is method se tum check kar sakte ho ki kya ek set dusre set ka subset hai ya nahi (yaani kya pehle set ke sabhi elements doosre set mein hain).
- Syntax: .issubset()
- Example

```
set1 = {1, 2, 3}
set2 = {1, 2, 3, 4, 5}
result = set1.issubset(set2)
print(result)  # Output: True
```

- 8 issuperset()
- Purpose: Is method se tum check kar sakte ho ki kya ek set dusre set ka superset hai ya nahi (yaani kya pehle set mein doosre set ke sabhi elements hain).
- Syntax: .issuperset()
- Example

```
set1 = {1, 2, 3, 4, 5}
set2 = {1, 2, 3}
result = set1.issuperset(set2)
print(result)  # Output: True
```

- 9 isdisjoint()
- Purpose: Is method se tum check kar sakte ho ki kya do sets ke beech koi common elements nahi hain (yaani unka intersection empty hai).
- syntax: .isdisjoint()
- Example

```
set1 = {1, 2, 3}
set2 = {4, 5, 6}
result = set1.isdisjoint(set2)
print(result)  # Output: True
```

Summary:

- add(), remove(), discard(), pop(), clear(): Sets ko modify karte hain.
- copy(): Set ka shallow copy banata hai.
- update(): Set ko update karne ke liye use hota hai.
- issubset(), issuperset(), isdisjoint(): Set ke relationship ko check karte hain doosre set ke saath.
- Yeh sab methods sets ke liye kaafi useful hote hain jab tumhe sets ko manipulate ya check karna ho.