Set and Froen_set
in
Python

What is set?

Set is similar to the set in math, set is a buildin data structure in python that we can store data on that.

Set properties

- Set can only store unique data.
- In set data will sort automatically.

In python here exist tow kind of set:

1. Set.
2. Frozen_set.

Different between **Set** and **Frozen_Set**

Set:

- Set is Mutable, meaning you can add or remove elements after the set has been created.
- We can create set by {} and set() set constructor.

Forzen_set:

- Forzenset is immutable, meaning we can not add or remove elements after the forzenset has been created.
- We can create FrozenSet just by frozenset() constructor.

Set

Set creation:

1: using {}:

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

What is the out put?

2:using set(). Set constructor:

```
sett=set([1,2,3,3,4,5,5,5,4])
```

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

Set build in methods

basic

```
sett={1,2,3,4,5,6,7,8,9}
print(len(sett))
print(2 in sett) # True
Print(10 in sett) #False
x=sett.__contains__(1)
print(x) #False
```

Getting copy of set

```
sett2=sett.copy()
print(sett2)
```

1. **Adding element:**

```
Sett={1,2,3,4,5,6}
Sett.add(7)
Sett.add([8,9,10])
```

Set build in methods

2. **removing element of set by remove method, discard, pop and clear method:**

   - remove: Removing an element using remove(item) method (raises KeyError if not found) and dose not return anything.

```
Sett={1,2,3,20}
try:
     Sett.remove(20)
except:
     print(“element not found”)
```

   - discard: Removing an element using discard() method (dose not raises KeyError if not found) and dose not return anything.

```
Sett={1,2,3,20}
Sett.discard(20)
```

Set build in methods…

- Pop: remove the first element of the set and return the removed element.

```
sett={5,4,3,2,1}
x=sett.pop()
print(x)
# guess what is the output?
```

- Clear: removing all elements of set and dose not return anything.

```
sett={1,2,3,4,5}
sett.clear()
print(sett)
```

Set build in methods…

3: Set operations:

- **Joining sets: we can join tow or more then tow sets using union() method and |"or" operator, both will return a new set**

```
sett1= {1,2,3,4}
sett2 = {1,2,3,5}
sett3 ={8,9,10}
sett4 = sett1.union(sett2,sett3)
print(sett3)

sett5 = sett1| sett2 | sett3
print(sett5)
```

Set build in methods…

- finding the similar elements between tow or more then tow sets:

```
a={1,2,3,4,5}
b={1,2,3,4,6}
c={1,2,3}
d={1,2,4,6,7,8,9}
d=b.intersection(a,c,d)
print(d)
print(a&b&c&d)
```

- finding those elements that form set1 dose not sexist in set2

```
e=a.difference(b,c,d)
print(e)   #{5}
print(d-a-b-c)  #{8,9,7}
```

Set build in methods…

- Symmetric Difference: finding all elements that are being unique in combination of to set. For example, set a={1,2,3,4,5,7} and set b={1,2,3,4,6}, form 1 up to 4 present in both sets it will ignore them and will return a new set that contains{5,6,7} that are unique.
- Note: Symmetric_Difference() method just get one parameter.

```
a={1,2,3,4,5,7}
b={1,2,3,4,6}
e=a.symmetric_difference(b)
print(e)
print(b^a)
```

Set build in methods…

4: Set Comparisons:

```
a = {1,2,3,4,5,6}
b ={1,2,3}
```

- issubset: set1 is subset of set2.

```
print(a.issubset(b)) #False
print(b.issubset(a))  #True
```

- issuperset: set1 is superset of set2.

```
print(a.issuperset(b))
print(b.issuperset(a))
```

- isdisjoint: return true if tow set being completely diffirent.

```
print(a.isdisjoint(b))
```

- difference_update: Removes elements from the set that are present in another set. it is in place and dose not return any things

```
a.difference_update(b)
print(a)
```

Set build in methods…

a={1,2,3,4,5,6}

b={3,4,5,7,8,9

- intersection_update: Updates the set, keeping only elements found in both sets (in place and dose not return any thing).

```
a.intersection_update(b)
print(a) #{3,4,5}
```

- symmetric_difference_update: Updates the set, keeping only elements in either set but not in both (in place).

```
a.symmetric_difference_update(b)
print(a) #{1,2,6,7,8,9}
```

Some basic set techniques

1. Set Comprehensions: You can create sets using a comprehension, similar to list comprehensions.

```
squared_set ={x**2 for x in range(10+1)}
print(squared_set)
```

2. Set with Conditional Logic we can also add conditions while creating sets.

```
even_set = {x for x in range(20) if x % 2 == 0}
```

3. Check if all elements in set are greater than 0.

```
set_a={1,2,3,4,5,6}
result = all(x > 0 for x in set_a)
```

4. Check the set contains at the least one exact element to the condition

```
result = any(x == 2 for x in set_a)
```

Frozen_set

- A frozenset in Python is an immutable version of a set, which means that once a frozenset is created, it cannot be modified. This makes it useful when you need a set that should not change and can be used as a key in a dictionary or an element of another set

.

- **Characteristics of frozenset:**
  - Immutability: You cannot add or remove elements after creation.
  - Unordered: The elements do not maintain any specific order.
  - Unique Elements: Like sets, frozensets cannot have duplicate elements.
  - Creating a frozenset:
  - You can create a frozenset using the frozenset() constructor:

Frozen_set methods

We can use all set methods on frozenset that except those method that make changes on set like: add(), remove(), pop()…..

When we can use Set?

1. **Use set if you are working with unique data or you need to store unique data.**
2. **Use set when you fell a problem can be solve by that.**

END