Python Sets

Dr. Seema Gupta Bhol

Python Sets

- A set is an unordered collection of items. Every element is unique and must be immutable.
- Python sets are mutable; you can add and remove items from it.
- Sets can be used to perform mathematical set operations like:
  - Union
  - Intersection
  - symmetric difference
  - etc.

Creating a Set

- A set is created by placing all the elements inside curly braces { }, separated by comma or by using the built-in function set().
- The elements can be of different types (integer, float, tuple, string etc.).
- But a set cannot have a mutable element, like list, set or dictionary, as its element.

```
#creating a set
numberSet = {1,2,3,4,3,2}
print(numberSet)

#creating an empty set
emptySet = {}   #This creates a dictionary
print(type(emptySet))

emptySet = set()   #This creates a empty set
print(type(emptySet))
```

```
{1, 2, 3, 4}
<class 'dict'>
<class 'set'>
```

Empty Set

- Dictionaries came first, when sets were introduced, the design decision was made to keep { } as an empty dictionary for backward compatibility, to avoid breaking a massive amount of existing code that relied on this behavior.
- To create an empty set, Python requires the use of the set() constructor.
- my_dict = { } results in an empty dictionary.
- my_set = set() results in an empty set.
- thisset = set(("apple", "banana", "cherry")) # note the double round-brackets

  print(thisset)

- A set can contain elements of different type

```
# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```

```
{1.0, 'Hello', (1, 2, 3)}
```

- A set can contain elements of different type

```
set_with_lists = {[1,2,3]}
```

```
------------------------------
TypeError
```

- A set cannot have a mutable element (like a list, set, or dictionary) as its element because set elements must be hashable, and mutable types are not .
- This is required for the set's internal mechanisms to function correctly.

Length of a Set

- Get the To determine how many items a set has, use the len() function.

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

Access Items

- We can not access items in a set by referring to an index or a key.
- But we can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:

  print(x)
```

Example 1: Check if "cherry" is present in the set

```
thisset = {"apple", "banana", "cherry"}

print("cherry" in thisset)
```

Example2: Check if “apple " is NOT present in the set

```
thisset = {"apple", "banana", "cherry"}

print("banana" not in thisset)
```

Adding elements to a set

- Sets are mutable. But since they are unordered, indexing have no meaning.
- We cannot access or change an element of set using *indexing* or *slicing*.
- We can add single element using the `add()` method and multiple elements using the `update()` method.
  - `add()` for single elements
  - `update()` for iterable
- The `update()` method can take ***tuples***, ***lists***, ***strings*** or ***other sets*** as its argument.
- In all cases, duplicates are avoided.

Adding Elements to set

Example: add method

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

Example :Add elements from another set to current set:

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

Add Any Iterable

- The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).

Example :Add elements of a list to a set

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
```

Remove Set Items

To remove an item in a set, use the remove(), or the discard() method.

Example : Remove "banana" by using the remove() method:

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

Note: If the item to remove does not exist, remove() will raise an error.

Example: Remove "cherry" by using the discard() method:

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("cherry")
print(thisset)
```

Note: If the item to remove does not exist, discard() will NOT raise an error.

Clear and del method

The clear() method empties the set:

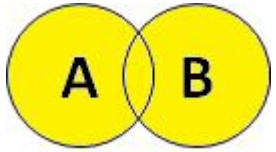
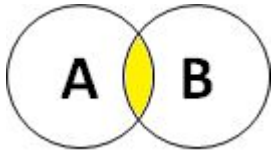
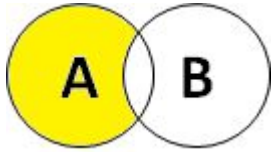
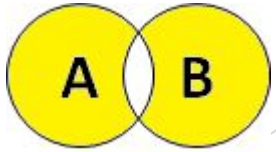
```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)
```

The del keyword will delete the set completely:

```
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset)
```

Python Set Operations

- Sets can be used to carry out mathematical set operations like *union*, *intersection*, *difference* and *symmetric difference*.
- We can do this with operators or methods.

union	|	
intersection	&	
difference	-	
Symmetric difference	^	

Set Operations

- If A and B are two python sets

```
1  A = {1,2,3,4}
2  B = {2,4,6,8, 6, 6}
3
4  print('A= ',A)
5  print('B= ',B)
6
7  s1 = A | B    # elements in a or b or both
8  print('Union: A | B =',s1)
9
10 s2 = A & B    # elements in both a and b
11 print('Intersection: A & B =',s2)
12
13 s3 = A - B    # elements in a but not in b
14 print('Difference: A - B =',s3)
15
16 s4 = A ^ B    # elements in a or b but not both
17 print('Symmetric Diff:  A ^ B =',s4)
```

```
A=  {1, 2, 3, 4}
B=  {8, 2, 4, 6}
A - B = {1, 3}
A | B = {1, 2, 3, 4, 6, 8}
A & B = {2, 4}
A ^ B = {1, 3, 6, 8}
```

Set operations

```
set1 = {"a", "b", "c",1,4}
set2 = {1, 2, 3, "a","d"}

set3 = set1.union(set2)
print(set3)

set4 = set1.intersection(set2)
print(set4)

set5 = set1.difference(set2)
print(set5)

set6 = set1.symmetric_difference(set2)
print(set6)
```

The symmetric_difference() method will keep only the elements that are NOT present in both sets.

The update() changes the original set, and does not return a new set.

```
set1 = {"a", "b", "c",1,4}
set2 = {1, 2, 3, "a","d"}
```

```
set1.update(set2) or
set1.intersection_update(set2) or
set1.difference_update(set2) or
set1.symmetric_difference_update(set2)
```

```
print(set1)
```

More Set Methods

- isdisjoint(): Returns whether two sets have a intersection or not

Return True if no items in set x is present in set y:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "facebook"}
z = x.isdisjoint(y)
print(z)
```

- issubset() or <= Returns True if all items of this set is present in another set

Return True if all items in set x are present in set y:

```
x = {"a", "b", "c"}
y = {"f", "e", "d", "c", "b", "a"}
z = x.issubset(y)
print(z)
```

Exercise

Q1 Create two sets:

A = {1, 2, 3, 4}

B = {3, 4, 5, 6}

Perform: Union, Intersection, Difference (A - B), Symmetric Difference

Q2 Write a program to find common elements between two user-defined sets.

Q3. Write a program to find all unique elements from two sets.

Q 4Write a program to check if one set is a subset of another.

Q 5. Given three sets, find elements common to all three.

Q 6. Create a set of numbers from 1 to 10 and remove all even numbers.

Q7. Write a program to check if two sets are equal.

Q8. Students participating in Cricket, Football, and Basketball. Find those in:

- All sports
- At least one sport
- Only one sport