# 8. Set

April 21, 2023

## 1 Set

- A set is an unordered collection of items.
- Every set element is unique (no duplicates) and must be immutable (cannot be changed).
- However, a set itself is mutable. We can add or remove items from it.

## 2 Creating Python Sets

- A set is created by placing all the items (elements) inside curly braces `{}`, separated by comma, or by using the built-in `set()` function.
- It can have any number of items and they may be of different types (integer, float, tuple, string etc.).
- But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

[1]:
```python
# set of integers
my_set = {1, 2, 3}
print(my_set)
```

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

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

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

## 3 Set cannot have duplicates

[3]:
```python
my_set = {1, 2, 3, 4, 3, 2}
print(my_set)
```

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

[4]:
```python
# we can make set from a list
my_set = set([1, 2, 3, 2])
print(my_set)
```

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

[5]:
```
# set cannot have mutable items
# here [3, 4] is a mutable list this will cause an error.

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_9289/1522997546.py in <module>
      2 # here [3, 4] is a mutable list this will cause an error.
      3
----> 4 my_set = {1, 2, [3, 4]}

TypeError: unhashable type: 'list'
```

## 4 Creating an empty set

- Empty curly braces {} will make an empty dictionary in Python.
- To make a set without any elements, we use the `set()` function without any argument.

[6]:
```
# initialize a with {}
a = {}
```

[7]:
```
# check data type of a
type(a)
```

[7]:
```
dict
```

[8]:
```
# initialize a with set()
b = set()
```

[9]:
```
# check data type of a
type(b)
```

[9]:
```
set
```

## 5 Modifying a set in Python

- Sets are mutable. However, since they are unordered, indexing has no meaning.
- We cannot access or change an element of a set using indexing or slicing. Set data type does not support it.
- We can add a single element using the `add()` method, and multiple elements using the `update()` method.
- The `update()` method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

[10]:
```python
my_set = {1, 3}
print(my_set)
```

```
{1, 3}
```

[11]:
```python
my_set[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_9289/2063814584.py in <module>
----> 1 my_set[0]

TypeError: 'set' object is not subscriptable
```

[12]:
```python
# add an element
my_set.add(2)
```

[13]:
```python
my_set
```

[13]:
```
{1, 2, 3}
```

[14]:
```python
# add multiple elements
my_set.update([2, 3, 4])
```

[15]:
```python
my_set
```

[15]:
```
{1, 2, 3, 4}
```

[16]:
```python
# add list and set
my_set.update([4, 5], {1, 6, 8})
```

[17]:
```python
my_set
```

[17]:
```
{1, 2, 3, 4, 5, 6, 8}
```

## 6 Removing elements from a set

- A particular item can be removed from a set using the methods `discard()` and `remove()`.
- The only difference between the two is that, the `discard()` function leaves a set unchanged if the element is not present in the set. On the other hand, the `remove()` function will raise an error if element is not present in the set.

[18]:
```python
# initialize my_set
my_set = {1, 3, 4, 5, 6}
print(my_set)
```

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

[19]:
```
# discard an element
my_set.discard(4)
```

[20]:
```
my_set
```

[20]:
```
{1, 3, 5, 6}
```

[21]:
```
# remove an element
my_set.remove(6)
print(my_set)
```

```
{1, 3, 5}
```

[22]:
```
# discard an element not present in my_set
my_set.discard(2)
print(my_set)
```

```
{1, 3, 5}
```

[23]:
```
# remove an element not present in my_set
# it will give an error
my_set.remove(2)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipykernel_9289/1514813433.py in <module>
      1 # remove an element not present in my_set
      2 # it will give an error
----> 3 my_set.remove(2)

KeyError: 2
```

- Similarly, we can remove and return an item using the `pop()` method.
- Since set is an unordered data type, there is no way of determining which item will be popped. It is completely arbitrary.

[24]:
```
# initialize my_set
my_set = set("HelloWorld")
```

[25]:
```
my_set
```

[25]:
```
{'H', 'W', 'd', 'e', 'l', 'o', 'r'}
```

```
[26]: # pop an element
      # Output: random element
      my_set.pop()
```

```
[26]: 'd'
```

```
[27]: # pop another element
      my_set.pop()
```

```
[27]: 'e'
```

```
[28]: my_set
```

```
[28]: {'H', 'W', 'l', 'o', 'r'}
```

- We can also remove all the items from a set using the `clear()` method.

```
[29]: # clear my_set
      my_set.clear()
```

```
[30]: my_set
```

```
[30]: set()
```

## 7 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.

```
[31]: A = {1, 2, 3, 4, 5}
      B = {4, 5, 6, 7, 8}
```

### 7.1 Set Union

- Union of A and B is a set of all elements from both sets.
- Union is performed using `|` operator. Same can be accomplished using the `union()` method.

```
[32]: # use | operator
      print(A | B)
```

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

```
[33]: # use union function
      A.union(B)
```

```
[33]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
[34]: B.union(A)
```

```
[34]: {1, 2, 3, 4, 5, 6, 7, 8}
```

### 7.2 Set Intersection

- Intersection of A and B is a set of elements that are common in both the sets.
- Intersection is performed using `&` operator. Same can be accomplished using the `intersection()` method.

```
[35]: # use & operator
      print(A & B)
```

```
{4, 5}
```

```
[36]: # use intersection function on A
      A.intersection(B)
```

```
[36]: {4, 5}
```

```
[37]: # use intersection function on B
      B.intersection(A)
```

```
[37]: {4, 5}
```

### 7.3 Set Difference

- Difference of the set B from set A (A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of elements in B but not in A.
- Difference is performed using `-` operator. Same can be accomplished using the `difference()` method.

```
[38]: # use - operator on A
      print(A - B)
```

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

```
[39]: # use - operator on B
      B - A
```

```
[39]: {6, 7, 8}
```

```
[40]: # use difference function on A
      A.difference(B)
```

```
[40]: {1, 2, 3}
```

[41]:
```
# use difference function on B
B.difference(A)
```

[41]:
```
{6, 7, 8}
```

| Method | Description |
|---|---|
| `add()` | Adds an element to the set |
| `clear()` | Removes all elements from the set |
| `copy()` | Returns a copy of the set |
| `difference()` | Returns the difference of two or more sets as a new set |
| `difference_update()` | Removes all elements of another set from this set |
| `discard()` | Removes an element from the set if it is a member. (Do nothing if the element is not in set) |
| `intersection()` | Returns the intersection of two sets as a new set |
| `intersection_update()` | Updates the set with the intersection of itself and another |
| `isdisjoint()` | Returns True if two sets have a null intersection |
| `issubset()` | Returns True if another set contains this set |
| `issuperset()` | Returns True if this set contains another set |
| `pop()` | Removes and returns an arbitrary set element. Raises KeyError if the set is empty |
| `remove()` | Removes an element from the set. If the element is not a member, raises a KeyError |
| `symmetric_difference()` | Returns the symmetric difference of two sets as a new set |
| `symmetric_difference_update()` | Updates a set with the symmetric difference of itself and another |
| `union()` | Returns the union of sets in a new set |
| `update()` | Updates the set with the union of itself and others |

## 8 For Loop on Set

[42]:
```
A
```

[42]:
```
{1, 2, 3, 4, 5}
```

[43]:
```
for i in A:
    print(i**2)
    print('+'*5)
```

```
1
+++++
4
+++++
9
+++++
16
+++++
25
+++++
```

## 9 Frozenset

- Frozenset has the characteristics of a set, but its elements cannot be changed once assigned. While tuples are immutable lists, frozensets are immutable sets.
- Frozensets can be created using the `frozenset()` function.
- Being immutable, it does not have methods that add or remove elements.

[44]:
```
A = frozenset([1, 2, 3, 4])
B = frozenset([3, 4, 5, 6])
```

[45]:
```
A
```

[45]:
```
frozenset({1, 2, 3, 4})
```

[46]:
```
print(B)
```

```
frozenset({3, 4, 5, 6})
```

[47]:
```
type(A)
```

[47]:
```
frozenset
```

[48]:
```
A.difference(B)
```

[48]:
```
frozenset({1, 2})
```

[49]:
```
 A | B
```

[49]:
```
frozenset({1, 2, 3, 4, 5, 6})
```

[50]:
```
A.add(3)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_9289/2254226290.py in <module>
----> 1 A.add(3)
```

```
AttributeError: 'frozenset' object has no attribute 'add'
```