- **created by**: Sudip Ghimire
- **URL**: https://www.sudipghimire.com.np
- **GitHub**: https://github.com/ghimiresdp

go to course contents

Chapter 3.4. Set

Table of Contents

Introduction

Sets are similar to lists except they can contain unique items. Some features of sets are as follows.

- Sets are similar to lists except they have unique data
- Sets are represented by comma separated values enclosed within braces `{}`
- Sets are not ordered, so we can not access sets using indices.
- Set items can not be changed individually, but can be added to it or removed from it.

Creating a set

As explained above, sets can be created using comma separated values enclosed within braces.

Example:

```
numbers = {18, 24, 56, 21, 44, 27, 99, 100, 64}
animals = {'cat', 'dog', 'tiger'}
random_lists = {1, 'John', 'Moon', True, 45.62}
```

If we create a set with duplicate values, they get discarded.

```
numbers = {18, 24, 16, 18, 44, 21, 24, 18, 19, 44, 18}
print(numbers)

# {44, 16, 18, 19, 21, 24}
```

Adding an item to the set

we can add an item to the set using `set.add()` method.

```
prime = {2, 3, 5, 7}
prime.add(11)
print(prime)
# {2, 3, 5, 7, 11}
```

Adding multiple items to the set

We can add multiple items to the set using `set.update()` method. The `update()` method takes an iterable which might be a list, tuple, set, or any iterable.

```
prime = {2, 3, 5, 7}
prime.update([11, 13, 17])
# {2, 3, 5, 7, 11, 13, 17}
```

Removing items from the set

We can use either of the following methods to remove items from the set:

- `set.discard()` method
- `set.remove()` method

```
even = {2, 4, 6, 8, 10, 12}
even.discard(12)
print(even)
# {2, 4, 6, 8, 10}

even.remove(10)
print(even)
# {2, 4, 6, 8}
```

Here removing an item with either of the method gives the same result, however if an item does not exist in the set then the `remove()` method throws a KeyError.

```

```

```
even = {2, 4, 6, 8, 10}
even.discard(12)  # does not throw any error

even.remove(12)  # KeyError: 12
```

finding out the length of the set

similar to other iterables, we can use `len()` function or `__len__()` method to find the length of the list.

```
odd = {1, 3, 5, 7, 9, 11}
print(len(odd))  # 6
print(odd.__len__())  # 6
```

Mathematical model of the set

Sets are useful when we want to find out the similarity, difference, etc using different set methods. We can perform different operations between sets which are as follows.

Union

Union gives all the items that are contained in either of the set. This gives all items without any duplicate number of items. In mathematics, it is represented by `A U B`.

```
odd = {1, 3, 5, 7, 9, 11}
prime = {2, 3, 5, 7, 11}

print(odd.union(prime))     # {1, 2, 3, 5, 7, 9, 11}
print(odd | prime)          # {1, 2, 3, 5, 7, 9, 11}
```

Intersection

Intersection gives common items that represents both sets. In mathematics, it is represented by `A n B`.

```
odd = {1, 3, 5, 7, 9, 11}
prime = {2, 3, 5, 7, 11}

print(odd.intersection(prime))  # {11, 3, 5, 7}
print(odd & prime)              # {11, 3, 5, 7}
```

Difference

Difference gives sets that are contained in only the specified set. If there are any common, they are discarded. In mathematics, it is represented by `A - B` or `B - A`. It is also called as `A-Only` and `B-only`

```
odd = {1, 3, 5, 7, 9, 11}
prime = {2, 3, 5, 7, 11}

print(odd.difference(prime))    # {1, 9}
print(odd - prime)              # {1, 9}

print(prime.difference(odd))    # {2}
print(prime - odd)              # {2}
```

Symmetric Difference

A symmetric difference between two sets can be defined by the items that share no common intersections. In mathematics, it is represented by `(A U B) - (A n B)` or `(A - B) U (B - A)`

```
odd = {1, 3, 5, 7, 9, 11}
prime = {2, 3, 5, 7, 11}

print(odd.symmetric_difference(prime))  # {1, 2, 9}
print(odd ^ prime)                      # {1, 2, 9}
```

Accessing items of a set

We can not access individual item of the set, however we can iterate through all items of a set using for loop.

```
prime = {2, 3, 5, 7, 11, 13, 17}

for item in prime:
    print(prime)
```

Some useful Set Methods

- `set.clear()`
- `set.copy()`
- `set.isdisjoint()`
- `set.issubset()`
- `set.issuperset()`

Practical Implementation: removing duplicate of a list

If we want to remove duplicate of a list, we can just typecast a list to the set and again back to list so that all duplicates are removed.

Note: while removing duplicates with this method, it is not guarenteed that the list becomes ordered as previous.

```
numbers = [1, 5, 3, 2, 6, 5, 3, 7, 1, 0, 10, 5, 6, 8]
numbers = list(set(numbers))
print(numbers)
# [0, 1, 2, 3, 5, 6, 7, 8, 10]
```