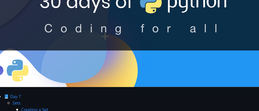
30 Days Of Python: Day 7 - Sets

Author: Asabeneh Yetayew
Second Edition: July, 2021

<< Day 6 | Day 8 >>



- Day 7
  - Sets
    - Creating a Set
    - Getting Set's Length
    - Accessing Items in a Set
    - Checking an Item
    - Adding Items to a Set
    - Removing Items from a Set
    - Clearing Items in a Set
    - Deleting a Set
    - Converting List to Set
    - Joining Sets
    - Finding Intersection Items
    - Checking Subset and Super Set
    - Checking the Difference Between Two Sets
    - Finding Symmetric Difference Between Two Sets
    - Joining Sets
  - 💻 Exercises: Day 7
    - Exercises: Level 1
    - Exercises: Level 2
    - Exercises: Level 3

📘 Day 7

Sets

Set is a collection of items. Let me take you back to your elementary or high school Mathematics lesson. The Mathematics definition of a set can be applied also in Python. Set is a collection of unordered and non-indexed distinct elements. In Python set is used to store unique items, and it is possible to find the *union, intersection, difference, symmetric difference, subset, super set and disjoint set* among sets.

Creating a Set

We use the *set()* built-in function.

- Creating an empty set

```
# syntax
st = set()
```

- Creating a set with initial items

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
```

Example:

```
# syntax
fruits = {'banana', 'orange', 'mango', 'lemon'}
```

Getting Set's Length

We use *len()* method to find the length of a set.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
len(st)
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
len(fruits)
```

Accessing Items in a Set

We use loops to access items. We will see this in loop section

Checking an Item

To check if an item exist in a list we use in membership operator.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
print('item1 set in contain item1: ', 'item1' in st) # Does set st contain item1? True
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
print('mango in fruits:', 'mango' in fruits) # True
```

Adding Items to a Set

Once a set is created we cannot change any items and we can also add additional items.

- Add one item using *add()*

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.add('item5')
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.add('lime')
```

- Add multiple items using *update()* The *update()* allows to add multiple items to a set. The *update()* takes a list argument.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.update(['item5','item6','item7'])
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
fruits.update(vegetables)
```

Removing Items from a Set

We can remove an item from a set using *remove()* method. If the item is not found *remove()* method will raise errors, so it is good to check if the item exist in the given set. However, *discard()* method doesn't raise any errors.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.remove('item4')
```

The *pop()* methods remove a random item from a list and it returns the removed item.

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.pop() # removes a random item from the set
```

If we are interested in the removed item.

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
removed_item = fruits.pop()
```

Clearing Items in a Set

If we want to clear or empty the set we use *clear* method.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.clear()
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.clear()
print(fruits) # set()
```

Deleting a Set

If we want to delete the set itself we use *del* operator.

```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
del st
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
del fruits
```

Converting List to Set

We can convert list to set and set to list. Converting list to set removes duplicates and only unique items will be reserved.

```
# syntax
lst = ['item1', 'item2', 'item3', 'item4', 'item2']
st = set(lst) # {'item1', 'item4', 'item2', 'item3'} - the order is random, because sets in general are unordered
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon','orange','banana'}
fruits = set(fruits) # {'mango', 'lemon', 'banana', 'orange'}
```

Joining Sets

We can join two sets using the *union()* or *update()* method.

- Union This method returns a new set

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st3 = st1.union(st2)
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
fruits_and_veg = fruits.union(vegetables)
print(fruits_and_veg) # {'lemon', 'carrot', 'tomato', 'banana', 'mango', 'orange', 'cabbage', 'potato', 'onion'}
```

- Update This method inserts a set into a given set

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st1.update(st2) # st1 contains and added to st1
```

Example:

```
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
fruits.update(vegetables)
print(fruits) # {'lemon', 'carrot', 'tomato', 'banana', 'mango', 'orange', 'cabbage', 'potato', 'onion'}
```

Finding Intersection Items

Intersection returns a set of items which are in both the sets. See the example

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item3', 'item4', 'item5', 'item6'}
st3.intersection(st2) # {'item3', 'item4'}
```

Example:

```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.intersection(even_numbers) # {0, 2, 4, 6, 8, 10}

python = {'p', 'y', 't', 'h', 'o', 'n'}
dragon = {'D', 'r', 'a', 'g', 'o', 'n'}
python.intersection(dragon)  # {'o', 'n'}
```

Checking Subset and Super Set

A set can be a subset or super set of other sets:

- Subset: *issubset*
- Super set: *issuperset*

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item1', 'item2'}
st1.issuperset(st2) # True
st2.issubset(st1) # True
```

Example:

```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.issubset(even_numbers) # False, because it is a super set
whole_numbers.issuperset(even_numbers) # True

python = {'p', 'y', 't', 'h', 'o', 'n'}
dragon = {'D', 'r', 'a', 'g', 'o', 'n'}
python.issubset(dragon)  # False
```

Checking the Difference Between Two Sets

It returns the difference between two sets.

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item3', 'item4'}
st1.difference(st2) # {'item1', 'item2'}
st2.difference(st1) # {'item3', 'item4'} == st1-st2
```

Example:

```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.difference(even_numbers) # {1, 3, 5, 7, 9}

python = {'p', 'y', 't', 'h', 'o', 'n'}
dragon = {'D', 'r', 'a', 'g', 'o', 'n'}
dragon.difference(python)  # {'D', 'r', 'a', 'g'}  - the result is unordered (characteristic of sets)
dragon.difference(dragon)  # set() # {'D', 'r', 'o', 'n', 'a', 'g'}
```

Finding Symmetric Difference Between Two Sets

It returns the the symmetric difference between two sets. It means that it returns a set that contains all items from both sets, except items that are present in both sets, mathematically: (A∪B) - (B∩A)

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item3', 'item4'}
# It means (A∪B)-(B∩A)
st1.symmetric_difference(st2) # {'item1', 'item2'}
```

Example:

```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
whole_numbers.symmetric_difference(even_numbers) # {1, 3, 5, 7, 9}

python = {'p', 'y', 't', 'h', 'o', 'n'}
dragon = {'D', 'r', 'a', 'g', 'o', 'n'}
python.symmetric_difference(dragon)  # {'p', 'y', 't', 'h', 'D', 'r', 'a', 'g'}
```

Joining Sets

If two sets do not have a common item or items we call them disjoint sets. We can check if two sets are joint or disjoint using *isdisjoint()* method.

```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6'}
st1.isdisjoint(st2) # False
```

Example:

```
even_numbers = {0, 2, 4 ,6, 8}
odd_numbers = {1, 3, 5, 7, 9}
even_numbers.isdisjoint(odd_numbers) # True, because no common item

python = {'p', 'y', 't', 'h', 'o', 'n'}
dragon = {'D', 'r', 'a', 'g', 'o', 'n'}
python.isdisjoint(dragon)  # False, there are common items {'o', 'n'}
```

🌝 You are a rising star. You have just completed day 7 challenges and you are 7 steps ahead in to your way to greatness. Now do some exercises for your brain and muscles.

💻 Exercises: Day 7

```
# sets
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
A = {19, 22, 24, 27, 29, 31}
B = {19, 22, 24, 26, 27, 29, 31, 37, 17}
age = [22, 19, 24, 25, 26, 24, 25, 24, 29]
```

Exercises: Level 1

1. Find the length of the set it_companies
2. Add 'Twitter' to it_companies
3. Insert multiple IT companies at once to the set it_companies
4. Remove one of the companies from the set it_companies
5. What is the difference between remove and discard

Exercises: Level 2

1. Join A and B
2. Find A intersection B
3. Is A subset of B
4. Are A and B disjoint sets
5. Join A with B and B with A
6. What is the symmetric difference between A and B
7. Delete the sets completely

Exercises: Level 3

1. Convert the ages to a set and compare the length of the list and the set, which one is bigger?
2. Explain the difference between the following data types: string, list, tuple and set
3. *I am a teacher and I love to inspire and teach people.* How many unique words have been used in the sentence? Use the split methods and set to get the unique words.

🎉 CONGRATULATIONS ! 🎉