sets

November 12, 2025

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 un-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**

[9]:
```
# syntax
st = set()
```

- **Creating a set with initial items**

[12]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
```

Example:

[15]:
```
# syntax
fruits = {'banana', 'orange', 'mango', 'lemon'}
```

Getting Set's Length

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

[19]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
len(st)
```

[19]: 4

Example:

[22]:
```
fruits = {'banana', 'orange', 'mango', 'lemon'}
len(fruits)
```

[22]: 4

Accessing Items in a Set

We use loops to access items.

Checking an Item

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

[31]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
print("Does set st contain item3? ", 'item3' in st) # Does set st contain item3?
  ↪ True
```

```
Does set st contain item3?  True
```

Example:

[36]:
```
fruits = {'banana', 'orange', 'mango', 'lemon'}
print('mango' in fruits ) # True
```

```
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()*

[43]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.add('item5')
```

Example:

[46]:
```
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.*

[51]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.update(['item5','item6','item7'])
```

Example:

[54]:
```
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.

[61]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.remove('item2')
```

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

Example:

[67]:
```
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.pop()  # removes a random item from the set
```

[67]:
```
'mango'
```

If we are interested in the removed item.

[72]:
```
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.

[78]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
st.clear()
```

Example:

[81]:
```
fruits = {'banana', 'orange', 'mango', 'lemon'}
fruits.clear()
print(fruits) # set()
```

```
set()
```

Deleting a Set

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

[87]:
```
# syntax
st = {'item1', 'item2', 'item3', 'item4'}
del st
```

Example:

[92]:
```
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.

[98]:
```
# syntax
lst = ['item1', 'item2', 'item3', 'item4', 'item1']
st = set(lst)  # {'item2', 'item4', 'item1', 'item3'} - the order is random,␣
 ↪because sets in general are unordered
```

Example:

[101]:
```
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*

[108]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st3 = st1.union(st2)
```

Example:

[111]:
```
fruits = {'banana', 'orange', 'mango', 'lemon'}
vegetables = {'tomato', 'potato', 'cabbage','onion', 'carrot'}
print(fruits.union(vegetables)) # {'lemon', 'carrot', 'tomato', 'banana',␣
 ↪'mango', 'orange', 'cabbage', 'potato', 'onion'}
```

```
{'onion', 'tomato', 'cabbage', 'potato', 'carrot', 'banana', 'mango', 'orange',
'lemon'}
```

- *Update This method inserts a set into a given set*

[115]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item5', 'item6', 'item7', 'item8'}
st1.update(st2) # st2 contents are added to st1
```

Example:

[118]:
```
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'}
```

```
{'onion', 'tomato', 'cabbage', 'potato', 'carrot', 'banana', 'mango', 'orange',
'lemon'}
```

Finding Intersection Items

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

[124]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item3', 'item2'}
st1.intersection(st2) # {'item3', 'item2'}
```

[124]:
```
{'item2', 'item3'}
```

Example:

[129]:
```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
a = whole_numbers.intersection(even_numbers) # {0, 2, 4, 6, 8, 10}
print(a)

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

```
{0, 2, 4, 6, 8, 10}
{'o', 'n'}
```

Checking Subset and Super Set

A set can be a subset or super set of other sets: - *Subset: issubset()* - *Super set: issuperset*

[137]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.issubset(st1) # True
st1.issuperset(st2) # True
```

[137]:
```
True
```

Example:

[144]:
```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
a = whole_numbers.issubset(even_numbers) # False, because it is a super set
b = whole_numbers.issuperset(even_numbers) # True
print(a)
print(b)

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

```
False
True
False
```

Checking the Difference Between Two Sets

It returns the difference between two sets.

[148]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.difference(st1) # set()
st1.difference(st2) # {'item1', 'item4'} => st1\st2
```

[148]:
```
{'item1', 'item4'}
```

Example:

[153]:
```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
even_numbers = {0, 2, 4, 6, 8, 10}
a = whole_numbers.difference(even_numbers) # {1, 3, 5, 7, 9}

python = {'p', 'y', 't', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
b = python.difference(dragon)     # {'p', 'y', 't'}  - the result is unordered
 ↪(characteristic of sets)
c = dragon.difference(python)     # {'d', 'r', 'a', 'g'}
print(a)
print(b)
print(c)
```

```
{1, 3, 5, 7, 9}
{'p', 't', 'y'}
{'g', 'r', 'd', 'a'}
```

Finding Symmetric Difference Between Two Sets

It returns 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,

[190]:
```
# syntax
# mathematically: (A\B)   (B\A)
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
# it means (A\B) (B\A)
st2.symmetric_difference(st1) # {'item1', 'item4'}
```

[190]:
```
{'item1', 'item4'}
```

Example:

[164]:
```
whole_numbers = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
some_numbers = {1, 2, 3, 4, 5}
a =whole_numbers.symmetric_difference(some_numbers) # {0, 6, 7, 8, 9, 10}
```

```
python = {'p', 'y', 't', 'h', 'o','n'}
dragon = {'d', 'r', 'a', 'g', 'o','n'}
b =python.symmetric_difference(dragon)  # {'r', 't', 'p', 'y', 'g', 'a', 'd',␣
 ↪'h'}
print(a)
print(b)
```

```
{0, 6, 7, 8, 9, 10}
{'g', 'd', 'y', 'a', 'p', 'r', 'h', 't'}
```

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.

[169]:
```
# syntax
st1 = {'item1', 'item2', 'item3', 'item4'}
st2 = {'item2', 'item3'}
st2.isdisjoint(st1) # False
```

[169]:
```
False
```

Example:

[174]:
```
even_numbers = {0, 2, 4 ,6, 8}
odd_numbers = {1, 3, 5, 7, 9}
a =even_numbers.isdisjoint(odd_numbers) # True, because no common item

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

```
True
False
```

Exercise Assignment

[177]:
```
# sets
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle',␣
 ↪'Amazon'}
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
age = [22, 19, 24, 25, 26, 24, 25, 24]
```

- *Find the length of the set it_companies*
- *Add 'Twitter' to it_companies*
- *Insert multiple IT companies at once to the set it_companies*
- *Remove one of the companies from the set it_companies*
- *What is the difference between remove and discard*

- *Join A and B*
- *Find A intersection B*
- *Is A subset of B*
- *Are A and B disjoint sets*
- *Join A with B and B with A*
- *What is the symmetric difference between A and B*
- *Delete the sets completely*

- *Convert the ages to a set and compare the length of the list and the set, which one is bigger?*
- *Explain the difference between the following data types: string, list, tuple and set*
- *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.*

[ ]: