Day7_Set_Built-in_DS_3

May 24, 2025

1 Day 7: Sets in Python

Today I learned about **sets** in Python — a built-in data structure used to store **unordered** collections of **unique items**. Sets are different from lists and tuples in several important ways:

Key Properties of Sets:

1. Unordered & unindexed collection of items
2. Set elements are unique — duplicate elements are not allowed
3. Set elements are immutable (cannot be changed)
4. Set itself is mutable — we can add or remove items from it

You can create a set using curly braces like this: `my_set = {1, 2, 3}` Or using the `set()` constructor: `another_set = set([1, 2, 2, 3])` → will automatically remove duplicates

Useful Set Methods:

- `add()` – adds a single item
- `update()` – adds multiple items
- `remove()` – removes an item (error if not found)
- `discard()` – removes an item (no error if not found)
- `pop()` – removes a random item
- `clear()` – removes all items

Set Operations:

- `union()` – combines elements from both sets
- `intersection()` – gets common elements
- `difference()` – elements in one set but not the other
- `symmetric_difference()` – elements not common to both sets

Comparison Methods:

- `issubset()` – checks if one set is a subset of another
- `issuperset()` – checks if a set contains another set
- `isdisjoint()` – checks if sets have no elements in common

Note: Sets are mutable, but all elements stored in a set must be **immutable** (like numbers, strings, or tuples).

2 Set Creation

[3]:
```
myset = {1,2,3,4,5} # Set of numbers
myset
```

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

[4]:
```
len(myset) #Length of the set
```

[4]:
```
5
```

[5]:
```
my_set = {1,1,2,2,3,4,5,5}
my_set
# Duplicate elements are not allowed.
```

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

[6]:
```
myset1 = {1.79,2.08,3.99,4.56,5.45} # Set of float numbers
myset1
```

[6]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

[7]:
```
myset2 = {'Asif' , 'John' , 'Tyrion'} # Set of Strings
myset2
```

[7]:
```
{'Asif', 'John', 'Tyrion'}
```

[8]:
```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

[8]:
```
{(11, 22, 32), 10, 20, 'Hola'}
```

[9]:
```
myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items like␣
 ↪li
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable item␣
 ↪like li
      2 myset3

TypeError: unhashable type: 'list'
```

[10]:
```
myset4 = set() # Create an empty set
print(type(myset4))
```

```
<class 'set'>
```

[11]:
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

[11]:
```
{'four', 'one', 'three', 'two'}
```

[12]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

```
six
five
eight
four
two
three
seven
one
```

[13]:
```
for i in enumerate(myset):
    print(i)
```

```
(0, 'six')
(1, 'five')
(2, 'eight')
(3, 'four')
(4, 'two')
(5, 'three')
(6, 'seven')
(7, 'one')
```

3 Set Membership

[14]:
```
myset
```

[14]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[15]:
```
'one' in myset # Check if 'one' exist in the set
```

[15]:
```
True
```

[16]:
```
'ten' in myset # Check if 'ten' exist in the set
```

[16]:
```
False
```

[17]:
```
 if 'three' in myset: # Check if 'three' exist in the set
    print('Three is present in the set')
```

```
else:
    print('Three is not present in the set')
```

```
Three is present in the set
```

[18]:
```
if 'eleven' in myset: # Check if 'eleven' exist in the list
    print('eleven is present in the set')
else:
    print('eleven is not present in the set')
```

```
eleven is not present in the set
```

4 Add & Remove Items

[19]:
```
myset
```

[19]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[20]:
```
myset.add('NINE') # Add item to a set using add() method
myset
```

[20]:
```
{'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[21]:
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE']) # Add multiple item to a set using
myset
```

[21]:
```
{'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[22]:
```
myset.remove('NINE') # remove item in a set using remove() method
myset
```

[22]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
```

```
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[23]:
```
myset.discard('TEN') # remove item from a set using discard() method
myset
```

[23]:
```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[24]:
```
myset.clear() # Delete all items in a set
myset
```

[24]:
```
set()
```

[25]:
```
del myset # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 2
      1 del myset # Delete the set object
----> 2 myset

NameError: name 'myset' is not defined
```

5 Copy Set

[26]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

[26]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[27]:
```
myset1 = myset # Create a new reference "myset1"
myset1
```

[27]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[28]:
```
id(myset) , id(myset1) # The address of both myset & myset1 will be the same as
```

[28]:
```
(2021026422496, 2021026422496)
```

[29]:
```
my_set = myset.copy() # Create a copy of the list
my_set
```

[29]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[30]:
```
id(my_set) # The address of my_set will be different from myset because my_set␣
↪i
```

[30]:
```
2021026424288
```

[31]:
```
myset.add('nine')
myset
```

[31]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

[32]:
```
 myset1 # myset1 will be also impacted as it is pointing to the same Set
```

[32]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

[33]:
```
my_set # Copy of the set won't be impacted due to changes made on the original␣
↪S
```

[33]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

6 Set Operation

6.1 Union

[34]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

[35]:
```
A | B # Union of A and B (All elements from both sets. NO DUPLICATES)
```

[35]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[ ]:
```

```

[36]:
```
A.union(B) # Union of A and B
```

[36]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[37]:
```
A.union(B, C) # Union of A, B and C.
```

[37]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[38]:
```
"""
Updates the set calling the update() method with union of A , B & C.
For below example Set A will be updated with union of A,B & C.
"""
A.update(B,C)
A
```

[38]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

6.2 Intersection

[39]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[40]:
```
A & B # Intersection of A and B (Common items in both sets)
```

[40]:
```
{4, 5}
```

[42]:
```
A.intersection(B) #Intersection of A and B
```

[42]:
```
{4, 5}
```

[43]:
```
"""
Updates the set calling the intersection_update() method with the intersection␣
↪o
For below example Set A will be updated with the intersection of A & B.
"""
A.intersection_update(B)
A
```

[43]:
```
{4, 5}
```

6.3 Difference

[44]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[45]:
```
A - B # set of elements that are only in A but not in B
```

[45]:
```
{1, 2, 3}
```

[46]:
```
A.difference(B) # Difference of sets
```

[46]:
```
{1, 2, 3}
```

[47]:
```
B- A # set of elements that are only in B but not in A
```

[47]: {6, 7, 8}

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

[48]: {6, 7, 8}

[49]:
```
"""
Updates the set calling the difference_update() method with the difference of␣
 ↪se
For below example Set B will be updated with the difference of B & A.
"""
B.difference_update(A)
B
```

[49]: {6, 7, 8}

6.4 Symmetric Difference

[50]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[51]:
```
A ^ B # Symmetric difference (Set of elements in A and B but not in both.␣
 ↪"EXCLUDE SAME"
```

[51]: {1, 2, 3, 6, 7, 8}

[52]:
```
A.symmetric_difference(B) # Symmetric difference of sets
```

[52]: {1, 2, 3, 6, 7, 8}

[53]:
```
"""
Updates the set calling the symmetric_difference_update() method with the␣
 ↪symmet
For below example Set A will be updated with the symmetric difference of A & B.
"""
A.symmetric_difference_update(B)
A
```

[53]: {1, 2, 3, 6, 7, 8}

7 Subset , Superset & Disjoint

[54]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

[55]:
```
B.issubset(A) # Set B is said to be the subset of set A if all elements of B␣
↪are in A
```

[55]: True

[56]:
```
A.issuperset(B)
```

[56]: True

[57]:
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common␣
↪elemets
```

[57]: True

[58]:
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common␣
↪elements
```

[58]: False

7.1 Other builtin function

[59]:
```
A
```

[59]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[60]:
```
sum(A)
```

[60]: 45

[61]:
```
max(A)
```

[61]: 9

[62]:
```
min(A)
```

[62]: 1

[63]:
```
len(A)
```

[63]: 9

[64]:
```
list(enumerate(A))
```

[64]:
```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

[65]:
```
D= sorted(A,reverse=True)
D
```

[65]:
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[ ]:
```
sorted(D)
```