Day 07 – Sets in Python

Understand the set data structure in Python and learn key operations such as creation, membership, set algebra, and built-in set functions.

What is a Set?

A **set** is an unordered, unindexed collection of unique items in Python. It is mutable, meaning you can add or remove elements after creation, but the elements themselves must be immutable.

Key Properties:

- Unordered: no guaranteed order
- No duplicates: every item is unique
- Mutable: can be changed (add/remove items)
- Supports mathematical set operations

Use cases: removing duplicates, membership testing, and performing set operations like union, intersection, etc.

1. Set Creation

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

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

In [2]:
```
len(myset) #Length of the set
```

Out[2]:
```
5
```

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

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

In [4]:
```
myset1 = {1.79,2.08,3.99,4.56,5.45} # Set of float numbers
myset1
```

Out[4]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [5]:
```
myset2 = {'Asif' , 'John' , 'Tyrion'} # Set of Strings
myset2
```

Out[5]:
```
{'Asif', 'John', 'Tyrion'}
```

In [6]:
```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

Out[6]:
```
{(11, 22, 32), 10, 20, 'Hola'}
```

In [7]:
```
myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items
      2 myset3

TypeError: unhashable type: 'list'
```

In [8]:
```
myset4 = set() # Create an empty set
print(type(myset4))
```

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

In [10]:
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

Out[10]:
```
{'four', 'one', 'three', 'two'}
```

2. Loop through a Set

In [11]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

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

In [12]:
```
for i in enumerate(myset):
    print(i)
```

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

3. Set Membership

In [13]:
```
myset
```

Out[13]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [14]:
```
'one' in myset # Check if 'one' exist in the set
```

Out[14]:
```
True
```

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

Out[15]:
```
False
```

In [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
```

In [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

In [19]:
```
myset
```

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

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

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

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

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

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

Out[22]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

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

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

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

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

In [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. Copying Sets

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

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

In [28]:
```
myset1 = myset # Create a new reference "myset1"
myset1
```

Out[28]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [29]:
```
id(myset) , id(myset1) # The address of both myset & myset1 will be the same as
```

Out[29]: (2592530041312, 2592530041312)

In [30]:
```
my_set = myset.copy() # Create a copy of the list
my_set
```

Out[30]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [31]:
```
id(my_set) # The address of my_set will be different from myset because my_set i
```

Out[31]: 2592530045792

In [33]:
```
myset.add('nine')
myset
```

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

In [34]:
```
myset1 # myset1 will be also impacted as it is pointing to the same Set
```

Out[34]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [35]:
```
my_set # Copy of the set won't be impacted due to changes made on the original S
```

Out[35]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

6. Set Operations

Union

In [36]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

In [37]:
```
A | B # Union of A and B (All elements from both sets. NO DUPLICATES)
```

Out[37]: {1, 2, 3, 4, 5, 6, 7, 8}

In [38]:
```
A.union(B) # Union of A and B
```

Out[38]: {1, 2, 3, 4, 5, 6, 7, 8}

In [39]:
```
A.union(B, C) # Union of A, B and C.
```

Out[39]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [42]:
```
"""
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
```

Out[42]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Intersection

In [43]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [44]:
```
A & B # Intersection of A and B (Common items in both sets)
```

Out[44]: {4, 5}

In [45]:
```
A.intersection(B) Intersection of A and B
```

```
  Cell In[45], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

In [46]:
```
"""
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
```

Out[46]: {4, 5}

Difference

In [47]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [48]:
```
A - B # set of elements that are only in A but not in B
```

Out[48]: {1, 2, 3}

In [49]:
```
A.difference(B) # Difference of sets
```

Out[49]: {1, 2, 3}

In [50]:
```
B- A # set of elements that are only in B but not in A
```

Out[50]: {6, 7, 8}

In [51]:
```
B.difference(A)
```

Out[51]: {6, 7, 8}

In [52]:
```
"""
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
```

Out[52]: {6, 7, 8}

Symmetric Difference

In [53]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [54]:
```
A ^ B # Symmetric difference (Set of elements in A and B but not in both. "EXCLU
```

Out[54]: {1, 2, 3, 6, 7, 8}

In [55]:
```
A.symmetric_difference(B) # Symmetric difference of sets
```

Out[55]: {1, 2, 3, 6, 7, 8}

7. Subset, Superset & Disjoint

In [56]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [57]:
```
B.issubset(A) # Set B is said to be the subset of set A if all elements of B are
```

Out[57]: True

In [58]:
```
A.issuperset(B) # Set A is said to be the superset of set B if all elements of B
```

Out[58]: True

In [59]:
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```

Out[59]: True

In [60]:
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```

Out[60]: False

8. Other Built-in Set Functions

In [61]:
```
A
```

Out[61]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [62]:
```
sum(A)
```

Out[62]: 45

In [63]:
```
max(A)
```

Out[63]: 9

In [65]:
```
min(A)
```

Out[65]: 1

In [66]:
```
len(A)
```

Out[66]: 9

In [67]:
```
list(enumerate(A))
```

Out[67]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [68]:
```
D = sorted(A,reverse=True)
D
```

Out[68]: [9, 8, 7, 6, 5, 4, 3, 2, 1]

In [69]:
```
sorted(D)
```

Out[69]: [1, 2, 3, 4, 5, 6, 7, 8, 9]