Set

September 25, 2024

1 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.

[3]:
```
myset = {1,2,3,4,5} # Set of numbers
print(myset)
print(len(myset))
print(type(myset))
```

```
{1, 2, 3, 4, 5}
5
<class 'set'>
```

[4]:
```
myset1=set((1,2,3,4))
myset1
```

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

2 Loop through a Set

[4]:
```
for i in enumerate(myset):
 print(i)
```

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

3 Set Membership

[7]:
```
'one' in myset # Check if 'one' exist in the set
```

[7]:
```
True
```

[5]:
```
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

[9]:
```
dir(set)
```

[9]:
```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
```

```
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

[6]:
```
myset
```

[6]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[7]:
```
myset.add(1) # Add item to a set using add() method
myset
```

[7]:
```
{1, 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[16]:
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE']) # Add multiple item to a set
myset
```

[16]:
```
{1,
 'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
```

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

[9]:
```
myset.update({2,3})
myset
```

[9]:
```
{1, 2, 3, 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[12]:
```
A=myset.update({2,3})
print(myset)
print(A)
```

```
{1, 'seven', 'two', 2, 3, 4, 5, 'six', 'eight', 'three', 'one', 'five', 'four'}
None
```

[13]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
 print(i)
```

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

[14]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in range(myset):
 print(i)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-db4f9fdcd3cc> in <module>
      1 myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
----> 2 for i in range(myset):
      3  print(i)

TypeError: 'set' object cannot be interpreted as an integer
```

[10]:
```
myset.update({4:'key',5:'hi'})#sets only store individual elements, not␣
 ↪key-value pairs.
myset
```

[10]:
```
{1, 2, 3, 4, 5, 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[20]:
```
myset.remove('NINE') # remove item in a set using remove() method
myset
```

[20]:
```
{1,
 2,
 3,
 4,
 5,
 'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[11]:
```
myset.remove('NINE') # remove item in a set using remove() method
myset
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-11-20aeccfb97c4> in <module>
----> 1 myset.remove('NINE') # remove item in a set using remove() method
      2 myset

KeyError: 'NINE'
```

5 remove()

Purpose: Removes a specific element from the set. Behavior when the element is not found: Raises a KeyError if the element is not present in the set. Typical use case: Use remove() when you are certain the element is in the set and you want to be alerted (via an error) if it's not.

discard() Purpose: Removes a specific element from the set. Behavior when the element is not found: Does not raise an error if the element is not present; it simply does nothing. Typical use case: Use discard() when you want to remove an element without worrying about whether it is present in the set.

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

[23]:
```
{1,
 2,
 3,
 4,
 5,
 'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[24]:
```
myset.discard('TEN') # remove item from a set using discard() method
myset
```

[24]:
```
{1,
 2,
 3,
 4,
 5,
 'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[26]:
```
myset.clear() # Delete all items in a set
myset
```

[26]:
```
set()
```

[27]:
```
del myset # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-27-8096fc3734fe> in <module>
      1 del myset # Delete the set object
----> 2 myset

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

[28]:
```
myset1 = {'one', 'two', 'three', 'four', 'five'}
myset1
```

[28]: {'five', 'four', 'one', 'three', 'two'}

[29]:
```
myset2 = myset1 # Create a new reference "myset1"
myset2
```

[29]: {'five', 'four', 'one', 'three', 'two'}

[31]:
```
id(myset1) , id(myset2)
```

[31]: (1437823282976, 1437823282976)

[32]:
```
my_set3 = myset1.copy() # Create a copy of the list
my_set3
```

[32]: {'five', 'four', 'one', 'three', 'two'}

[33]:
```
id(my_set3)
```

[33]: 1437823280288

[36]:
```
myset1.add('nine')
myset1
```

[36]: {'five', 'four', 'nine', 'one', 'three', 'two'}

[37]:
```
myset2 # myset1 will be also impacted as it is pointing to the same Set
```

[37]: {'five', 'four', 'nine', 'one', 'three', 'two'}

[39]:
```
my_set3
```

[39]: {'five', 'four', 'one', 'three', 'two'}

6 remove and pop

[40]:
```
myset1
```

[40]:
```
{'five', 'four', 'nine', 'one', 'three', 'two'}
```

[41]:
```
myset1.pop()
```

[41]:
```
'three'
```

[42]:
```
myset1
```

[42]:
```
{'five', 'four', 'nine', 'one', 'two'}
```

[43]:
```
myset1.pop('one')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-b2f8e927f9fe> in <module>
----> 1 myset1.pop('one')

TypeError: pop() takes no arguments (1 given)
```

[44]:
```
myset1.remove('one')
```

[45]:
```
myset1
```

[45]:
```
{'five', 'four', 'nine', 'two'}
```

[46]:
```
myset1.remove()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-46-39adbad23c2e> in <module>
----> 1 myset1.remove()

TypeError: remove() takes exactly one argument (0 given)
```

7 Union, Intersection, difference, symmetric difference

[5]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
set1=A.union(B,C) # Union of A, B and C.
print(set1)
```

```
print(A)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{1, 2, 3, 4, 5}
```

[6]:
```
print(A.union(B)) #removes duplicates
A
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

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

[7]:
```
set2=B | C # union operator
set2
```

[7]:
```
{4, 5, 6, 7, 8, 9, 10}
```

[8]:
```
A.update(B,C) #updates A with B and C sets
print(A)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[10]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C=A & B # Intersection of A and B (Common items in both sets)
C
```

[10]:
```
{4, 5}
```

[11]:
```
D=A.intersection(B)
D
```

[11]:
```
{4, 5}
```

[12]:
```
D=A.intersection(C)
D
```

[12]:
```
{4, 5}
```

[18]:
```
A={5,6,7,8}
B={6,9,10,5}
A.intersection_update(B)
C=A.intersection_update(B)
print(A)
print(C)
```

```
{5, 6}
None
```

[21]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
D=A - B # set of elements that are only in A but not in B
D
```

[21]: {1, 2, 3}

[26]:
```
C={5,6,1,8,9}
D=A.difference(C)
F=B.difference(D)
B.difference_update(D)
E=B.difference_update(D)
print("D: ",D)
print("F:",F)
print("B:",B)
print("E:",E)
```

```
D:  {2, 3, 4}
F: {8, 5, 6, 7}
B: {5, 6, 7, 8}
E: None
```

[28]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C=A ^ B # Symmetric difference
#Returns a set with elements that are in either of the sets but not in both.
print(A)
print(C)
```

```
{1, 2, 3, 4, 5}
```

[28]: {1, 2, 3, 6, 7, 8}

[31]:
```
C={11,5,7,9}
D=A.symmetric_difference(C)
print(D)
print(A)
```

```
{1, 2, 3, 4, 7, 9, 11}
{1, 2, 3, 4, 5}
```

[2]:
```
A = {1,2,3,4,5}
B = {1,2,3,4,7,9,11}
A.symmetric_difference_update(B)
A
```

[2]: {5, 7, 9, 11}

8 Subset , Superset & Disjoint

[3]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
B.issubset(A)
#Returns True if all elements of this set are present in another specified set.
```

[3]: True

[4]:
```
A.issuperset(B)
# Returns True if this set contains all elements of another specified set.
```

[4]: True

[5]:
```
C.issuperset(B)
```

[5]: False

[6]:
```
C.isdisjoint(A)
# Returns True if two sets have no elements in common
```

[6]: True

[7]:
```
A.isdisjoint(B)
```

[7]: False

[ ]:
```

```