Assignment 7 - Python

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.

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 like li
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 lik
e li
      2 myset3

TypeError: unhashable type: 'list'
```

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

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

In [9]:
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

Out[9]:
```
{'four', 'one', 'three', 'two'}
```

Loop through a Set

In [10]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset: print(i)
```

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

In [11]:
```
for i in enumerate(myset):
    print(i)
```

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

Set Membership

In [12]:
```
myset
```

Out[12]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [13]:
```
'one' in myset # Check if 'one' exist in the set
```

Out[13]: True

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

Out[14]: False

In [15]:
```
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 [16]:
```
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

Add & Remove Items

In [17]:
```
myset
```

Out[17]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [18]:
```
myset.add('NINE') # Add item to a set using add() method
myset
```

Out[18]: {'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [19]:
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE']) # Add multiple item to a set using
myset
```

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

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

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

In [21]:
```
myset.discard('TEN') # remove item from a set using discard() method
myset
```

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

In [22]:
```
myset.clear() # Delete all items in a set
myset
```

Out[22]:
```
set()
```

In [23]:
```
del myset # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[23], line 2
      1 del myset # Delete the set object
----> 2 myset

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

Copy Set

In [24]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

Out[24]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [25]:
```
myset1 = myset # Create a new reference "myset1"
myset1
```

Out[25]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [26]:
```
id(myset) , id(myset1) # The address of both myset & myset1 will be the same as
```

Out[26]:
```
(2114667759200, 2114667759200)
```

In [27]:
```
my_set = myset.copy() # Create a copy of the list
my_set
```

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

In [28]:
```
id(my_set) # The address of my_set will be different from myset because my_set i
```

Out[28]:
```
2114667760320
```

In [29]:
```
myset.add('nine')
myset
```

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

In [30]:
```
myset1 # myset1 will be also impacted as it is pointing to the same Set
```

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

In [31]:
```
my_set # Copy of the set won't be impacted due to changes made on the original S
```

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

Set Operation

Union

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

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

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

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

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

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

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

In [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
```

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

Intersection

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

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

Out[40]: {4, 5}

In [41]:
```
A.intersection(B) Intersection of A and B
```

```
  Cell In[41], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

In [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
```

Out[43]: {4, 5}

Difference

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

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

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

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

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

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

Out[47]: {6, 7, 8}

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

Out[48]: {6, 7, 8}

In [50]:
```
"""
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[50]: {6, 7, 8}

Symmetric Difference

In [51]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [52]:
```
A ^ B # Symmetric difference (Set of elements in A and B but not in both. "EXCLU
```

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

In [53]:
```
A.symmetric_difference(B) # Symmetric difference of sets
```

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

In [54]:
```
"""
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
```

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

Subset , Superset & Disjoint

In [55]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [56]:
```
B.issubset(A) # Set B is said to be the subset of set A if all elements of B are
```

Out[56]: True

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

Out[57]: True

In [58]:
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```

Out[58]: True

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

Out[59]: False

Other Builtin 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 [64]:
```
min(A)
```

Out[64]: 1

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

Out[65]: 9

In [66]:
```
list(enumerate(A))
```

Out[66]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [67]:
```
D= sorted(A,reverse=True)
D
```

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

In [68]:
```
sorted(D)
```

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