Sets

1. Unordered & unindexed collection of items
2. Set elements are unique. Duplicates 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 = {1,1,2,2,3,3,4,4,5,6,7,8}  # Duplicate elements are not allowed.
my_set
```

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

In [4]:
```
myset1 = {1.54,2.3,1.2,56.1,45,4.56}   # set of float numbers
myset1
```

Out[4]:
```
{1.2, 1.54, 2.3, 4.56, 45, 56.1}
```

In [5]:
```
my_set2 = {'Aryan','Himanshu','John'}   # Set of strings
my_set2
```

Out[5]:
```
{'Aryan', 'Himanshu', 'John'}
```

In [6]:
```
myset3 = {10,20,"Hola",(11,22,33)}  # Mixed datatypes
myset3
```

Out[6]:
```
{(11, 22, 33), 10, 20, 'Hola'}
```

In [7]:
```
myset3 = {10,20,"Hola",[11,22,33]}  # set doesn't allowed mutable items like list
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 1
----> 1 myset3 = {10,20,"Hola",[11,22,33]}  # set doesn't allowed mutable items like
list
      2 myset3

TypeError: unhashable type: 'list'
```

```
In [ ]:  my_set4 = set()  # create an empty list
         print(type(my_set4))
```

```
In [8]:  my_set1 = set(('one','two','three','four'))
         my_set1

Out[8]:  {'four', 'one', 'three', 'two'}
```

Loop through a set

```
In [9]:  myset = {'one','two','three','four','five','six'}

         for i in myset:
             print(i)

         five
         two
         four
         six
         three
         one
```

```
In [10]: for i in enumerate(myset):
             print(i)

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

Set Membership

```
In [11]: myset

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

```
In [12]: 'one'in myset  # check if 'one' exist in the set

Out[12]: True
```

```
In [13]: 'eight' in myset  # check if 'ten' exist in the set

Out[13]: False
```

```
In [14]: 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 [15]:
```
if 'ten' in myset:     # check if 'eleven ' exist in the list
    print('ten is present in the set')

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

```
ten is not present in the set
```

Add & Remove Items

In [16]:
```
myset
```

Out[16]:
```
{'five', 'four', 'one', 'six', 'three', 'two'}
```

In [19]:
```
myset.add('NINE')     # ADD item to a set using add()
myset
```

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

In [20]:
```
myset.update(['TEN', 'ELEVEN' , 'TWELVE'])   # Add multiple item to a set using updat
myset
```

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

In [21]:
```
myset.remove('NINE')   # remove item in a set using remove() method
myset
```

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

In [22]:
```
myset.discard('TEN')  # remove item from a set using discard() method
myset
```

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

In [23]:
```
myset.clear()  #Delete all items in a set
myset
```

Out[23]:
```
set()
```

In [25]:
```
del myset  # delete the set object
myset
```

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

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

Copy Set

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

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

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

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

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

Out[28]:
```
(1851918068064, 1851918068064)
```

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

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

In [30]:
```
id(my_set)     # The address of my_set will be different from myset
```

Out[30]:
```
1851918066944
```

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

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

In [33]:
```
myset1       # myset1 will be also impacted as it is pointing to the same set
```

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

In [34]:
```
my_set      # Copy of the set won't be impacted due to changes made on th original se
```

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

Set Operation

Union

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

```
In [5]: A | B           # Union of A and B (All elements from both sets. )

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

```
In [6]: B | C   # Union of B and C

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

```
In [7]: A|B     # Union of A and B

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

```
In [8]: A.union(B)

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

```
In [9]: A.union(B, C)  # Union of A, B and C

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

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

Intersection

```
In [11]: A = {1,2,3,4,5}
         B = {4,5,6,7,8}
```

```
In [12]: A & B   # Intersection  of A & B (common items in both sets)

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

```
In [15]: A.intersection(B)   #Intersection of A & B

Out[15]: {4, 5}
```

In [17]:

```python
"""
Updates the set calling the intersection_update() method with the intersection of A,

for below example Set A will be updated with the intersection of A & B.
"""

A.intersection_update(B)
A
```

Out[17]:

```
{4, 5}
```

Difference

In [18]:

```python
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [19]:

```python
A - B  # set of elements that are only in A but not in B
```

Out[19]:

```
{1, 2, 3}
```

In [20]:

```python
A.difference(B)  # Difference of sets
```

Out[20]:

```
{1, 2, 3}
```

In [21]:

```python
B -A   # sets of elements that are only in B but not in A
```

Out[21]:

```
{6, 7, 8}
```

In [23]:

```python
B.difference(A)
```

Out[23]:

```
{6, 7, 8}
```

In [24]:

```python
"""
Updates the set calling the difference_update() method with the difference of set A,

for below example Set B  will be updated with the intersection of B & A.
"""

B.difference_update(A)
B
```

Out[24]:

```
{6, 7, 8}
```

Symmetric Difference

In [12]:

```python
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [13]:

```python
A ^ B    #Symmetric difference (Set of elements in A and B but not in both)
```

Out[13]:

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

In [14]:

```
A.symmetric_difference(B)   #Symmetric difference of sets
```

Out[14]:

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

In [15]:

```


A.symmetric_difference_update(B)
A
```

Out[15]:

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

Subset , Superset & Disjoint

In [16]:

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

In [17]:

```
B.issubset(A)    # Set B is said to be the subset of A if all elements of B are prese
```

Out[17]:

```
True
```

In [18]:

```
A.issuperset(B)   # Set A is said to be the superset of set B if all the elements are
```

Out[18]:

```
True
```

In [19]:

```
B.isdisjoint(A)   # Two sets are said to be disjoint sets if they have no common elem
```

Out[19]:

```
False
```

In [21]:

```
C.isdisjoint(A)
```

Out[21]:

```
True
```

In [22]:

```
C.isdisjoint(B)
```

Out[22]:

```
True
```

Other Builtin Functions

In [23]:

```
A
```

Out[23]:

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

In [24]:

```
sum(A)
```

Out[24]:

```
45
```

In [25]:

```
max(A)
```

Out[25]:

```
9
```

```
In [26]: min(A)
```

```
Out[26]: 1
```

```
In [27]: len(A)
```

```
Out[27]: 9
```

```
In [28]: list(enumerate(A))
```

```
Out[28]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

```
In [29]: D = sorted(A,  reverse=True)
         D
```

```
Out[29]: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Dictionary

```
1. Dictionary is a mutable data type in Python
2. A python dictionary is a collection of key and value pairs separated by a colon
(:) & enclosed in curly braces{}.
3. Keys must be unique in a dictionary , duplicate values are allowed
```

```
In [30]: mydict = dict()  # empty dictionary
         mydict
```

```
Out[30]: {}
```

```
In [31]: mydict = {}   # empty dictionary
         mydict
```

```
Out[31]: {}
```

```
In [32]: mydict = {1:'one', 2:'two', 3:'three',4:'four'}   # dictionary with integer keys
         mydict
```

```
Out[32]: {1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

```
In [35]: mydict = dict({1:'one',2:'two',3:'three',4:'four',5:'five'})  # Create dictionary usi
         mydict
```

```
Out[35]: {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

```
In [36]: mydict = {'A':'one', 'B':'two','C':'Three'}  # dictionary with character keys
         mydict
```

```
Out[36]: {'A': 'one', 'B': 'two', 'C': 'Three'}
```

In [38]:
```
mydict = {1:'one', 'A':'two' , 'Three':3 }  # dictionary with mixed data types
mydict
```

Out[38]:
```
{1: 'one', 'A': 'two', 'Three': 3}
```

In [40]:
```
mydict.keys()   # return dictionary keys using keys() method
```

Out[40]:
```
dict_keys([1, 'A', 'Three'])
```

In [41]:
```
mydict.values() #return dictionary keys using values() method
```

Out[41]:
```
dict_values(['one', 'two', 3])
```

In [42]:
```
mydict.items()  # Access each key-value pair witnin a dictionary
```

Out[42]:
```
dict_items([(1, 'one'), ('A', 'two'), ('Three', 3)])
```

In [1]:
```
mydict = {1:'one', 2:'two', 'A':['asif','john','Maria']}  # dictionary with different
mydict
```

Out[1]:
```
{1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}
```

In [4]:
```
mydict = {1:'one', 2:'two','A':{'Name': 'asif','Age ':25},'B':('Bat','Cat','hat')}
mydict
```

Out[4]:
```
{1: 'one',
 2: 'two',
 'A': {'Name': 'asif', 'Age ': 25},
 'B': ('Bat', 'Cat', 'hat')}
```

In [5]:
```
keys ={'a','b','c','d'}
value = 10
mydict3 = dict.fromkeys(keys, value)    # Create a dictionary from a sequence of keys
mydict3
```

Out[5]:
```
{'d': 10, 'c': 10, 'a': 10, 'b': 10}
```

In [6]:
```
keys ={'a','b','c','d'}
value = [10,20,30,40]
mydict3 = dict.fromkeys(keys, value)
mydict3
```

Out[6]:
```
{'d': [10, 20, 30, 40],
 'c': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'b': [10, 20, 30, 40]}
```

In [7]:

```
value.append(40)
mydict3
```

Out[7]:

```
{'d': [10, 20, 30, 40, 40],
 'c': [10, 20, 30, 40, 40],
 'a': [10, 20, 30, 40, 40],
 'b': [10, 20, 30, 40, 40]}
```

Accessing Items

In [8]:

```
mydict = {1:'one', 2:'two', 3:'three', 4:'four'}
mydict
```

Out[8]:

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [9]:

```
mydict[1]   # Access item using key
```

Out[9]:

```
'one'
```

In [16]:

```
mydict1 = {'Name':'Asif','ID':23105, 'DOB':1995, 'Job':'Analyst'}
mydict1
```

Out[16]:

```
{'Name': 'Asif', 'ID': 23105, 'DOB': 1995, 'Job': 'Analyst'}
```

In [17]:

```
mydict1['Name']  #Access the item using key
```

Out[17]:

```
'Asif'
```

In [21]:

```
mydict1.get('Job') #Access the item using get() method
```

Out[21]:

```
'Analyst'
```

Add, Remove & Change Items

In [23]:

```
mydict2 = {'Name':'Ram', 'ID':56321, 'DOB':2006, 'Address':'Nagpur'}
mydict2
```

Out[23]:

```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Nagpur'}
```

In [25]:

```
dict1 = {'DOB':1995}
mydict2.update(dict1)
mydict2
```

Out[25]:

```
{'Name': 'Ram', 'ID': 56321, 'DOB': 1995, 'Address': 'Nagpur'}
```

In [26]:

```
mydict2['Job'] = 'Analyst'   # Adding items in the dictionary
mydict2
```

Out[26]:

```
{'Name': 'Ram',
 'ID': 56321,
 'DOB': 1995,
 'Address': 'Nagpur',
 'Job': 'Analyst'}
```

In [27]:
```
mydict2.popitem()  # A random item is removed
```
Out[27]:
```
('Job', 'Analyst')
```

In [28]:
```
mydict2
```
Out[28]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 1995, 'Address': 'Nagpur'}
```

In [30]:
```
del[mydict2['ID']]  # Removing item using del method
mydict2
```
Out[30]:
```
{'Name': 'Ram', 'DOB': 1995, 'Address': 'Nagpur'}
```

In [31]:
```
mydict2.clear()   # Delete all items of the dictionary using clear method
mydict2
```
Out[31]:
```
{}
```

In [32]:
```
del mydict2   # Delete the dictionary object
mydict2
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 2
      1 del mydict2   # Delete the dictionary object
----> 2 mydict2

NameError: name 'mydict2' is not defined
```

Copy Dictionary

In [33]:
```
mydict = {'Name':'Ram', 'ID':56321, 'DOB':2006, 'Address':'Nagpur'}
mydict
```
Out[33]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Nagpur'}
```

In [34]:
```
mydict1 = mydict # Create a new reference "mydict1"
```

In [35]:
```
id(mydict1) , id(mydict)
```
Out[35]:
```
(1454296229632, 1454296229632)
```

In [36]:
```
mydict2 = mydict.copy()   # Create a copy of the dictionary
```

In [37]:
```
id(mydict2)           # The address of mydict2 will be different from mydict
```
Out[37]:
```
1454292359936
```

In [38]:
```
mydict['Address'] = 'Pune'
```

In [39]:
```
mydict
```
Out[39]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Pune'}
```

In [40]:
```
mydict1       # mydict1 will be also impacted as it is pointing to the same dictionar
```
Out[40]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Pune'}
```

In [42]:
```
mydict2   # Copy of list won't be impacted due to the changes made in the original
```
Out[42]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Nagpur'}
```

Loop through a Dictionary

In [43]:
```
mydict1 = {'Name':'Ram', 'ID':56321, 'DOB':2006, 'Address':'Nagpur'}
mydict1
```
Out[43]:
```
{'Name': 'Ram', 'ID': 56321, 'DOB': 2006, 'Address': 'Nagpur'}
```

In [44]:
```
for i in mydict:
    print(i, ':', mydict1[i])   # Key & value pair
```
```
Name : Ram
ID : 56321
DOB : 2006
Address : Nagpur
```

Dictionary Membership

In [45]:
```
mydict1 = {'Name':'Asif', 'ID':56321, 'DOB':2006, 'Address':'Nagpur'}
mydict1
```
Out[45]:
```
{'Name': 'Asif', 'ID': 56321, 'DOB': 2006, 'Address': 'Nagpur'}
```

In [47]:
```
'Name' in mydict1  # Test if a key is in a dictionary or not
```
Out[47]:
```
True
```

In [49]:
```
'Asif' in mydict1   # Membership test can be only done for keys
```
Out[49]:
```
False
```

In [50]:
```
'ID' in mydict1
```
Out[50]:
```
True
```

In [51]:
```
'Address' in mydict1
```
Out[51]:
```
True
```

ALL / ANY

```
The all() method returns:
    True - If all keys of the dictionary are true
    False -If any keys of the dictionary is false

The any() function returns True if any key of the dictionary is True. If not ,
any() returns false.
```

In [62]:

```
mydict1 = {'Name':'Asif', 'ID':1526, 'DOB':2006, 'Address':'Nagpur'}
mydict1
```

Out[62]:

```
{'Name': 'Asif', 'ID': 1526, 'DOB': 2006, 'Address': 'Nagpur'}
```

In [63]:

```
all(mydict1)  # Will return false as one value is false (value 0)
```

Out[63]:

```
True
```

In [64]:

```
any(mydict1)
```

Out[64]:

```
True
```