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 the items from it.
```

Set Creation

In [145…
```
myset={1,2,3,4} # set of integer numbers
myset
```
Out[145…
```
{1, 2, 3, 4}
```

In [147…
```
len(myset)
```
Out[147…
```
4
```

In [149…
```
my_set={1,1,2,2,3,3,3} # Duplicate elements are not allowed
my_set
```
Out[149…
```
{1, 2, 3}
```

In [151…
```
myset1={3.5,5.3,6.7} # set of float numbers
myset1
```
Out[151…
```
{3.5, 5.3, 6.7}
```

In [153…
```
myset2={'one','two','three','four','five','six','seven'} # set of strings
myset2
```
Out[153…
```
{'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [155…
```
myset3={19,3.8,'three',(8,9,3)} # mixed data types
myset3
```
Out[155…
```
{(8, 9, 3), 19, 3.8, 'three'}
```

In [157…
```
myset4={89,4.5,[6,8,4]}  # set doesn't allow mutable items like lists
myset4
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[157], line 1
----> 1 myset4={89,4.5,[6,8,4]}  # set doesn't allow mutable items like lists
      2 myset4

TypeError: unhashable type: 'list'
```

Loop through a Set

```
In [160…  myset3

Out[160…  {(8, 9, 3), 19, 3.8, 'three'}

In [162…  for i in myset3:
              print(i)

          3.8
          (8, 9, 3)
          19
          three

In [164…  for i in enumerate(myset3):
              print(i)

          (0, 3.8)
          (1, (8, 9, 3))
          (2, 19)
          (3, 'three')
```

Set Membership

```
In [167…  'one' in myset2 # check if 'one' exists in the set

Out[167…  True

In [169…  'three' in myset2 # check if 'three' exists in the set

Out[169…  True

In [171…  if 'one' in myset2:    # check if 'one' exists in the set
              print('one is present in the set')

          one is present in the set

In [173…  if'seven' in myset2:       # check if 'seven' exists in the set
              print('seven is in the set')
          else:
              print('seven is not in the set')

          seven is in the set
```

Add & Remove items

```
In [176…  myset3

Out[176…  {(8, 9, 3), 19, 3.8, 'three'}

In [178…  myset3.add(89) # Add element to a set using add()method
          myset3

Out[178…  {(8, 9, 3), 19, 3.8, 89, 'three'}

In [180…  myset3.update(['one','two']) # Add multiple items in the set using list
          myset3
```

```
Out[180…  {(8, 9, 3), 19, 3.8, 89, 'one', 'three', 'two'}
```

```
In [182…  myset3.remove('one') # remove item in a set using remove() method
          myset3
```

```
Out[182…  {(8, 9, 3), 19, 3.8, 89, 'three', 'two'}
```

```
In [184…  myset3.discard('two') # remove item in a set using discard() method
          myset3
```

```
Out[184…  {(8, 9, 3), 19, 3.8, 89, 'three'}
```

```
In [186…  myset.clear() # deletes all items in the set
          myset
```

```
Out[186…  set()
```

```
In [187…  del myset # delete  the set object
          myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[187], line 2
      1 del myset # delete  the set object
----> 2 myset

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

Copy Set

```
In [191…  myset={'one','two','three','four','five','six','seven','eight','nine'}
          myset
```

```
Out[191…  {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [193…  myset1=myset # create a new reference 'myset1'
          myset1
```

```
Out[193…  {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [195…  id(myset),id(myset1) # Both the addresses will be the same
```

```
Out[195…  (1612568891616, 1612568891616)
```

```
In [197…  my_set=myset.copy() # create a copy of list
          my_set
```

```
Out[197…  {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [199…  id(my_set)
```

```
Out[199…  1612568892736
```

```
In [201…  myset1.add('nine') # myset1 will be also impacted as it is pointing to the same
          myset1
```

Out[201…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [203…
```
my_set # Copy of the set won't be impacted due to changes made on the original S
```

Out[203…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operations

Union

In [207…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
C={8,9,10}
```

In [209…
```
A|B # # Union of A and B (All elements from both sets. NO DUPLICATES)
```

Out[209…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [211…
```
A.union(B) # Union of A and B
```

Out[211…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [213…
```
A.union(B,C) # union of A,B and C
```

Out[213…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [215…
```
"""
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[215…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Intersection

In [218…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
```

In [220…
```
A&B  # Intersection of A and B (Common items in both sets)
```

Out[220…
```
{4, 5}
```

In [222…
```
A.intersection(B)  Intersection of A and B
```

```
  Cell In[222], line 1
    A.intersection(B)  Intersection of A and B
                       ^
SyntaxError: invalid syntax
```

In [224…
```
"""
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[224…
```
{4, 5}
```

Difference

In [227…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
```

In [229…
```
A-B # set of elements that are only in A but not in B
```

Out[229…
```
{1, 2, 3}
```

In [231…
```
A.difference(B) # Difference of sets
```

Out[231…
```
{1, 2, 3}
```

In [233…
```
B-A # set of elements that are only in B but not in A
```

Out[233…
```
{6, 7, 8}
```

In [235…
```
B.difference(A) # difference of sets
```

Out[235…
```
{6, 7, 8}
```

In [237…
```
"""
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[237…
```
{6, 7, 8}
```

Symmetric Difference

In [240…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
```

In [242…
```
A^B  # Symmetric difference (Set of elements in A and B but not in both. "EXCLUD
```

Out[242…
```
{1, 2, 3, 6, 7, 8}
```

In [244…
```
A.symmetric_difference(B) #  Symmetric Difference of sets
```

Out[244…
```
{1, 2, 3, 6, 7, 8}
```

In [246…
```
"""
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[246…
```
{1, 2, 3, 6, 7, 8}
```

Subset, Superset & Disjoint

In [249…
```
A={1,2,3,4,5,6,7,8,9}
B={3,4,5,6,7,8}
C={10,20,30,40}
```

In [251…
```
B.issubset(A)  #  Set B is said to be the subset of set A if all elements of B a
```
Out[251…
```
True
```

In [253…
```
A.issuperset(B) # Set A is said to be the superset of set B if all elements of B
```
Out[253…
```
True
```

In [255…
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[255…
```
True
```

In [257…
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[257…
```
False
```

Other Buitin Functions

In [260…
```
A
```
Out[260…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [262…
```
sum(A)
```
Out[262…
```
45
```

In [264…
```
max(A)
```
Out[264…
```
9
```

In [266…
```
min(A)
```
Out[266…
```
1
```

In [268…
```
len(A)
```
Out[268…
```
9
```

In [270…
```
list(enumerate(A))
```
Out[270…
```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

In [272…
```
D=sorted(A,reverse=True)
D
```
Out[272…
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [274…
```
sorted(D)
```
Out[274…
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

DICTIONARY

1. Dictionary is a mutable datatype in python.
2. A python dictionary is a collection of key and value pairs seperated by a colon(:) and enclosed in curly braces{}.
3. Keys must be unique in dictionary, Duplicate values are allowed.

Create Dictionary

In [278…
```
mydict=dict() # empty dictionary
mydict
```
Out[278…
```
{}
```

In [280…
```
mydict={} # empty dictionary
mydict
```
Out[280…
```
{}
```

In [282…
```
mydict={1:'one',2:'two',3:'three'} # dictionary with integer keys
mydict
```
Out[282…
```
{1: 'one', 2: 'two', 3: 'three'}
```

mydict=({1:'one',2:'two',3:'three'}) # create dictionary using dict() method mydict

In [285…
```
mydict={'A':'one','B':'two','C':'three'} # dictionary with character keys
mydict
```
Out[285…
```
{'A': 'one', 'B': 'two', 'C': 'three'}
```

In [287…
```
mydict={1:'one','A':'two',3:'three'} # dictionary with multiple keys
mydict
```
Out[287…
```
{1: 'one', 'A': 'two', 3: 'three'}
```

In [289…
```
mydict.keys() # Return Dictionary Keys using keys() method
```

```
Out[289… dict_keys([1, 'A', 3])
```

```
In [291… mydict.values()  # Return Dictionary Values using values() method
```

```
Out[291… dict_values(['one', 'two', 'three'])
```

```
In [293… mydict.items() # Access each key-value pair within a dictionary
```

```
Out[293… dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

```
In [295… mydict={1:'one',2:'two','A':['Lahari','PG']} #
         mydict
```

```
Out[295… {1: 'one', 2: 'two', 'A': ['Lahari', 'PG']}
```

```
In [297… mydict={1:'one',2:'two','A':['Welcome','Thank you'],'B':('Bat','Cat','Hat')}
         mydict
```

```
Out[297… {1: 'one', 2: 'two', 'A': ['Welcome', 'Thank you'], 'B': ('Bat', 'Cat', 'Hat')}
```

```
In [299… mydict={1:'one',2:'two','A':{'Name':'Lahari','Age':24},'B':('Bat','Cat','Hat')}
         mydict
```

```
Out[299… {1: 'one',
          2: 'two',
          'A': {'Name': 'Lahari', 'Age': 24},
          'B': ('Bat', 'Cat', 'Hat')}
```

```
In [301… keys={'a', 'b', 'c', 'd'}
         mydict3=dict.fromkeys(keys) # create a dictionary from a sequence of keys
         mydict3
```

```
Out[301… {'d': None, 'b': None, 'a': None, 'c': None}
```

```
In [303… keys={'a', 'b', 'c', 'd'}
         value=10
         mydict3=dict.fromkeys(keys,value) # create a dictionary from a sequence of keys
         mydict3
```

```
Out[303… {'d': 10, 'b': 10, 'a': 10, 'c': 10}
```

```
In [305… keys={'a', 'b', 'c', 'd'}
         value={10,20,30}
         mydict3=dict.fromkeys(keys,value) # create dictionary from a sequence of keys an
         mydict3
```

```
Out[305… {'d': {10, 20, 30}, 'b': {10, 20, 30}, 'a': {10, 20, 30}, 'c': {10, 20, 30}}
```

```
In [307… value.append(40)
         mydict3
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[307], line 1
----> 1 value.append(40)
      2 mydict3

AttributeError: 'set' object has no attribute 'append'
```

Accessing Items

In [310…
```
mydict={1:'one',2:'two',3:'three',4:'four'}
mydict
```
Out[310…
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [312…
```
mydict[1] # Access item using key
```
Out[312…
```
'one'
```

In [314…
```
mydict.get(1) # access item using get() method
```
Out[314…
```
'one'
```

In [316…
```
mydict1={'Name':'Lahari','ID':2431,'DOB':2001,'Job':'Fresher'}
mydict1
```
Out[316…
```
{'Name': 'Lahari', 'ID': 2431, 'DOB': 2001, 'Job': 'Fresher'}
```

In [318…
```
mydict1['Name'] # Access item using key
```
Out[318…
```
'Lahari'
```

In [320…
```
mydict1.get('ID') # Access item using get() method
```
Out[320…
```
2431
```

Add , Remove & Change Items

In [323…
```
mydict1={'Name':'Lahari','DOB':2001,'ID':2431,'Address':'Andhra Pradesh'}
mydict1
```
Out[323…
```
{'Name': 'Lahari', 'DOB': 2001, 'ID': 2431, 'Address': 'Andhra Pradesh'}
```

In [325…
```
mydict1['DOB']=2002 # changing dictionary items
mydict1['Address']= 'India'
mydict1
```
Out[325…
```
{'Name': 'Lahari', 'DOB': 2002, 'ID': 2431, 'Address': 'India'}
```

In [327…
```
dict1={'DOB':2004}
mydict1.update(dict1)
mydict1
```

```
Out[327… {'Name': 'Lahari', 'DOB': 2004, 'ID': 2431, 'Address': 'India'}
```

```
In [329… mydict1['Job']='Analyst' # Adding items in the dictionary
         mydict1
```

```
Out[329… {'Name': 'Lahari',
          'DOB': 2004,
          'ID': 2431,
          'Address': 'India',
          'Job': 'Analyst'}
```

```
In [331… mydict1.pop('Job') # removing items in the dictionary using pop() method
         mydict1
```

```
Out[331… {'Name': 'Lahari', 'DOB': 2004, 'ID': 2431, 'Address': 'India'}
```

```
In [333… mydict1.popitem()
```

```
Out[333… ('Address', 'India')
```

```
In [335… del[mydict1['ID']] # removing item using del method
         mydict1
```

```
Out[335… {'Name': 'Lahari', 'DOB': 2004}
```

```
In [337… mydict1.clear() # delete all items of the dictionary using clear() method
         mydict1
```

```
Out[337… {}
```

```
In [339… del mydict1 # delete the dictionary object
         mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[339], line 2
      1 del mydict1 # delete the dictionary object
----> 2 mydict1

NameError: name 'mydict1' is not defined
```

Copy Dictionary

```
In [342… mydict={'Name':'Lahari','ID':2567,'DOB':2006,'Address':'India'}
         mydict
```

```
Out[342… {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'India'}
```

```
In [343… mydict1=mydict # create a new reference "mydict1"
         mydict1
```

```
Out[343… {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'India'}
```

```
In [346… mydict2=mydict.copy() # create a copy of the dictionary
In [348…  id(mydict2) # The address of both mydict & mydict1 will be the same

Out[348…  1612569143936

In [350…  mydict['Address']='Bangalore'
          mydict

Out[350…  {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'Bangalore'}

In [352…  mydict1 # mydict1 will be also impacted as it is pointing to the same dictionary

Out[352…  {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'Bangalore'}

In [354…  mydict2 # Copy of list won't be impacted due to the changes made in the original

Out[354…  {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'India'}
```

Loop through a Dictionary

```
In [357…  mydict1

Out[357…  {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'Bangalore'}

In [359…  for i in mydict1:
              print(i ,':',mydict[i]) # key & value pair

Name : Lahari
ID : 2567
DOB : 2006
Address : Bangalore

In [361…  for i in mydict1:
              print(mydict1[i]) # Dictionary items

Lahari
2567
2006
Bangalore

In [363…  for i in enumerate(mydict1):
              print(i)

(0, 'Name')
(1, 'ID')
(2, 'DOB')
(3, 'Address')
```

Dictionary Membership

```
In [366…  mydict1

Out[366…  {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'Bangalore'}

In [368…  'Name' in mydict1 # Test if keys is in a dictionary or not
```

```
Out[368…   True

In [370…   '789' in mydict1 # Membership test can be only done for keys

Out[370…   False

In [372…   'ID' in mydict1

Out[372…   True

In [374…   'Address' in mydict1

Out[374…   True
```

All/Any

The all()method returns:

1. True- If all keys of the dictionary are True
2. False- If any key of the dictionary is False

The any()function returns True if any key of the dictionary is True.Ifnot,any()returns False

```
In [377…   mydict1

Out[377…   {'Name': 'Lahari', 'ID': 2567, 'DOB': 2006, 'Address': 'Bangalore'}

In [379…   all(mydict1) # will return false as one value is false (value 0)

Out[379…   True

In [383…   any(mydict1)

Out[383…   True
```