SET & DICT DATASTRUCTURES

Set

1.Sets are Unordered and Unindexed Collection of items. 2.Set elements are Unique.Duplicate elements are not allowed 3.Set itself is mutable.We can add or remove items from it. 4.Set elements are immutable or unhashable(cannot be changed)

Set Creation

```
In [149]: mys = {1,2,3,4,5} #set of numbers
          mys

Out[149]: {1, 2, 3, 4, 5}

In [151]: len(mys) #length of set

Out[151]: 5

In [153]: my_s = {1,1,2,2,3,4,5,5} #even if we give a similar elements in multiple times it
          my_s #because duplicate elemtnts are not allowed

Out[153]: {1, 2, 3, 4, 5}

In [155]: mys1 = {1.79,2.08,3.99,4.56,5.45} # set of float numbers
          mys1

Out[155]: {1.79, 2.08, 3.99, 4.56, 5.45}

In [157]: mys2 = {'asif','john','tyrion'} #set of strings
          mys2

Out[157]: {'asif', 'john', 'tyrion'}

In [161]: mys3 = {10,20,"Hola",(11,22,32)} #set of mixed datatypes
          mys3

Out[161]: {(11, 22, 32), 10, 20, 'Hola'}

In [163]: mys3 = {10,20,"Hola",[11,22,32]} #set doesn't allow mutable items like list
          mys3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[163], line 1
----> 1 mys3 = {10,20,"Hola",[11,22,32]} #set doesn't allow mutable items like list
      2 mys3

TypeError: unhashable type: 'list'
```

```
In [165]: mys4 = set() # Create an empty set
          print(type(mys4))
```

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

In [167]:
```
my_s1 = set(('one','two','three','four'))
my_s1
```

Out[167]:
```
{'four', 'one', 'three', 'two'}
```

Loop through a Set

In [194]:
```
mys = {'one','two','three','four','five','six','seven','eight'}
for i in mys:
    print(i)
```

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

In [196]:
```
for i in enumerate(mys):
    print(i)
```

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

Set Membership

In [198]:
```
mys
```

Out[198]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [200]:
```
'one' in mys
```

Out[200]:
```
True
```

In [202]:
```
'ten' in mys
```

Out[202]:
```
False
```

In [204]:
```
if 'three' in mys:
    print('Three is present in the set')
else:
    print('Three is not present in the set')
```

```
Three is present in the set
```

```
In [206]: if 'eleven' in mys:
              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 [208]: mys

Out[208]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [212]: mys.add('NINE') #we use add() method to add elements into the set
          mys

Out[212]: {'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [214]: mys.update(['TEN','ELEVEN','TWELVE']) # we use update() method to add multiple ite
          mys

Out[214]: {'ELEVEN',
           'NINE',
           'TEN',
           'TWELVE',
           'eight',
           'five',
           'four',
           'one',
           'seven',
           'six',
           'three',
           'two'}

In [226]: mys.remove('NINE') #we use remove method to remove item from the set
          mys

Out[226]: {'ELEVEN',
           'TEN',
           'TWELVE',
           'eight',
           'five',
           'four',
           'one',
           'seven',
           'six',
           'three',
           'two'}

In [228]: mys.discard('TEN') #we use discard method to remove item from the set
          mys
```

```
Out[228]: {'ELEVEN',
            'TWELVE',
            'eight',
            'five',
            'four',
            'one',
            'seven',
            'six',
            'three',
            'two'}
```

```
In [230]: mys.clear() # Deletes all the elements in the list
          mys
```

```
Out[230]: set()
```

```
In [232]: del mys # Delete the set object
          mys
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[232], line 2
      1 del mys # Delete the set object
----> 2 mys

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

Copy Set

```
In [237]: mys = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
          mys
```

```
Out[237]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [239]: mys1 = mys #create a new reference "mys1"
          mys1
```

```
Out[239]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [241]: id(mys),id(mys1) #The address of both mys and mys1 will be the same
```

```
Out[241]: (2525389413888, 2525389413888)
```

```
In [243]: my_s = mys.copy()  # Create a copy of the set
          my_s
```

```
Out[243]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [245]: id(my_s) # the address of my_s will be diffrent from mys because it is just a copy
```

```
Out[245]: 2525389412096
```

mys.add('nine') mys

```
In [249]: mys1 # mys1 wil also be updated because it is pointing to the same set
```

```
Out[249]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
In [251]: my_s # Copy of the set won't be impacted due to changes made on the orginal set
```

```
Out[251]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operations

Union - Gives all the elements from both sets and Duplicates are not allowed

```
In [259]: A = {1,2,3,4,5}
          B = {4,5,6,7,8}
          C = {8,9,10}
```

```
In [261]: A.union(B) #Gives all the elements from both A & B sets without duplicates
```

```
Out[261]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [264]: A | B # we can use '|' (pipe)  symbol for union
```

```
Out[264]: {1, 2, 3, 4, 5, 6, 7, 8}
```

```
In [268]: A.union(B,C) # gives all the elements in the three sets without duplicates
```

```
Out[268]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
In [270]: """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[270]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
In [ ]:
```

Intersection - Gives the Common Elements in the sets

```
In [278]: A = {1,2,3,4,5}
          B = {4,5,6,7,8}
```

```
In [280]: A.intersection(B) # 4,5 are the common elements in A and B sets
```

```
Out[280]: {4, 5}
```

```
In [282]: A & B # we use "&" symbol instead of 'intersection()'
```

```
Out[282]: {4, 5}
```

```
In [284]: A.intersection(B) Intersection of A and B
```

```
  Cell In[284], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

In [286]:
```
"""Updates the set calling the intersection_update() method with the intersection
For below example Set A will be updated with the intersection of A & B.
"""
A.intersection_update(B)
A
```

Out[286]:
```
{4, 5}
```

In [ ]:

Difference - Gives the elements that are only in the set A but not B

In [290]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [292]:
```
A.difference(B) # 4,5 are common and the remaining elements in A set be printed
```

Out[292]:
```
{1, 2, 3}
```

In [294]:
```
A - B  # we use "-" symbol instead of 'difference'
```

Out[294]:
```
{1, 2, 3}
```

In [298]:
```
B - A # 4,5 are common and prints the remaining elements in B
```

Out[298]:
```
{6, 7, 8}
```

In [300]:
```
"""Updates the set calling the difference_update() method with the difference of s
For below example Set B will be updated with the difference of B & A.
"""
B.difference_update(A)
B
```

Out[300]:
```
{6, 7, 8}
```

In [ ]:

Symmetric Difference - The symmetric difference of sets A and B includes everything that's in A or B, except what's in both.

In [306]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [308]:
```
A.symmetric_difference(B) #removes repeated or common elements and gives the eleme
```

Out[308]:
```
{1, 2, 3, 6, 7, 8}
```

In [312]:
```
A ^ B  # we use "^" symbol insted of 'symmetric_difference()'
```

Out[312]:
```
{1, 2, 3, 6, 7, 8}
```

In [314]:
```
"""Updates the set calling the symmetric_difference_update() method with the symme
For below example Set A will be updated with the symmetric difference of A & B.
"""
A.symmetric_difference_update(B)
A
```

Out[314]:
```
{1, 2, 3, 6, 7, 8}
```

In [ ]:

Subset , Superset & Disjoint

In [322]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [324]:
```
B.issubset(A) # Set B is subset of A if all the elements of B are in A
```

Out[324]:
```
True
```

In [328]:
```
A.issuperset(B) # set A is said to be superset of B if all the elements in B are
```

Out[328]:
```
True
```

In [330]:
```
C.isdisjoint(A) # Two sets are said to be disjoint if they have no Common elements
```

Out[330]:
```
True
```

In [332]:
```
B.isdisjoint(A) # because A and B has common elements
```

Out[332]:
```
False
```

In [ ]:

Other Built-In Functions

In [335]:
```
A
```

Out[335]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [337]:
```
sum(A) # to find summ i.e add all the elements
```

Out[337]:
```
45
```

In [339]:
```
max(A) # to find the highest element in the set
```

Out[339]:
```
9
```

In [341]:
```
min(A) # to find the least element in the set
```

Out[341]:
```
1
```

In [343]:
```
len(A) # to find the length of the set i.e number of elements in the set
```

```
Out[343]:  9

In [345]:  list(enumerate(A)) #gives all the elements with an order

Out[345]:  [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [347]:  D = sorted(A,reverse = True) # gives descending order
           D

Out[347]:  [9, 8, 7, 6, 5, 4, 3, 2, 1]

In [349]:  sorted(D) #gives ascending order

Out[349]:  [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [ ]:
```

Dictionary

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

Create Dictionary

```
In [354]:  myd = dict() #empty dictionary
           myd

Out[354]:  {}

In [356]:  myd = {}
           myd

Out[356]:  {}

In [360]:  myd = {1:'one',2:'two',3:'three'} #dictionary with integer keys
           myd

Out[360]:  {1: 'one', 2: 'two', 3: 'three'}

In [362]:  myd = dict({1:'one',2:'two',3:'three'}) # Create dictionary using dict()
           myd

Out[362]:  {1: 'one', 2: 'two', 3: 'three'}

In [364]:  myd = {'A':'one' ,'B' : 'two' ,'C' : 'three' } # dictionary with character keys
           myd

Out[364]:  {'A': 'one', 'B': 'two', 'C': 'three'}

In [366]:  myd = {1:'one', 'A':'two',3:'three'} #dictionary with mixed keys
           myd

Out[366]:  {1: 'one', 'A': 'two', 3: 'three'}
```

In [370]:
```
myd.keys() #Return Dictionary Keys using Keys() method
```
Out[370]:
```
dict_keys([1, 'A', 3])
```

In [372]:
```
myd.values() # Return Dictionary Keys using Values() method
```
Out[372]:
```
dict_values(['one', 'two', 'three'])
```

In [374]:
```
myd.items() # Access each key-value pair within a dictionary
```
Out[374]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

In [378]:
```
myd = {1:'one',2:'two', 'A':{'Name':'asif','Age':20}, 'B': ('Bat', 'cat', 'hat')}
myd
```
Out[378]:
```
{1: 'one',
 2: 'two',
 'A': {'Name': 'asif', 'Age': 20},
 'B': ('Bat', 'cat', 'hat')}
```

In [382]:
```
keys = {'a' , 'b' , 'c' , 'd'}# Create a dictionary from a sequence of keys
myd1 = dict.fromkeys(keys)
myd1
```
Out[382]:
```
{'d': None, 'a': None, 'c': None, 'b': None}
```

In [390]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
myd1 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of keys
myd1
```
Out[390]:
```
{'d': 10, 'a': 10, 'c': 10, 'b': 10}
```

In [392]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = [10,20,30]
myd1 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of keys
myd1
```
Out[392]:
```
{'d': [10, 20, 30], 'a': [10, 20, 30], 'c': [10, 20, 30], 'b': [10, 20, 30]}
```

In [394]:
```
value.append(40)
myd1
```
Out[394]:
```
{'d': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'c': [10, 20, 30, 40],
 'b': [10, 20, 30, 40]}
```

Acessing Items

In [399]:
```
myd = {1:'one',2:'two',3:'three',4:'four'}
myd
```
Out[399]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

```
In [401]: myd[1] #Access item using key

Out[401]: 'one'

In [405]: myd.get(1) #Acess item using get() method

Out[405]: 'one'

In [407]: myd1 = {'Name':'Asif' , 'ID': 74123 , 'DOB': 1991 , 'job' :'Analyst'}
          myd1

Out[407]: {'Name': 'Asif', 'ID': 74123, 'DOB': 1991, 'job': 'Analyst'}

In [409]: myd1['Name'] # Access item using key

Out[409]: 'Asif'

In [411]: myd1.get('job') # Access item using get() method

Out[411]: 'Analyst'
```

Add, Remove & Change Items

```
In [414]: myd1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
          myd1

Out[414]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}

In [418]: myd1['DOB'] = 1992 # Changing Dictionary Items
          myd1['Address'] = 'Delhi'
          myd1

Out[418]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1992, 'Address': 'Delhi'}

In [422]: d1 = {'DOB':1995}
          myd1.update(d1)
          myd1

Out[422]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}

In [424]: myd1['Job'] = 'Analyst' # Adding items in the dictionary
          myd1

Out[424]: {'Name': 'Asif',
           'ID': 12345,
           'DOB': 1995,
           'Address': 'Delhi',
           'Job': 'Analyst'}

In [426]: myd1.pop('Job') # Removing items in the dictionary using Pop method
          myd1

Out[426]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}
```

```
In [428]: myd1.popitem() # A random item is removed

Out[428]: ('Address', 'Delhi')

In [430]: myd1

Out[430]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1995}

In [432]: del[myd1['ID']] # Removing item using del method
          myd1

Out[432]: {'Name': 'Asif', 'DOB': 1995}

In [434]: myd1.clear() # Delete all items of the dictionary using clear method
          myd1

Out[434]: {}

In [436]: del myd1 # Delete the dictionary object
          myd1

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[436], line 2
      1 del myd1 # Delete the dictionary object
----> 2 myd1

NameError: name 'myd1' is not defined
```

Copy Dictionary

```
In [439]: myd = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
          myd

Out[439]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}

In [441]: myd1 = myd # Create a new reference "mydict1"

In [443]: id(myd) , id(myd) # The address of both mydict & mydict1 will be the same

Out[443]: (2525352263168, 2525352263168)

In [447]: myd2 = myd.copy() # Create a copy of the dictionary

In [449]: id(myd2) # The address of myd2 will be different from myd because myd2 is copy of

Out[449]: 2525358227200

In [453]: myd['Address'] = 'Mumbai'

In [455]: myd

Out[455]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```

In [457]:
```
myd1 #myd1 will be also impacted as it is pointing to the same dictionary
```
Out[457]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```
In [459]:
```
myd2 # Copy of list won't be impacted due to the changes made in the original
```
Out[459]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

Loop through a Dictionary

In [464]:
```
myd1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki' ,'Job'
myd1
```
Out[464]:
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1991,
 'Address': 'Hilsinki',
 'Job': 'Analyst'}
```
In [470]:
```
for i in myd1:
    print(i , ':' , myd1[i]) # Key & value pair
```
```
Name : Asif
ID : 12345
DOB : 1991
Address : Hilsinki
Job : Analyst
```
In [474]:
```
for i in myd1:
    print(myd1[i]) # Dictionary items
```
```
Asif
12345
1991
Hilsinki
Analyst
```

Dictionary Membership

In [477]:
```
myd1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
myd1
```
Out[477]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}
```
In [479]:
```
'Name' in myd1 # Test if a key is in a dictionary or not.
```
Out[479]:
```
True
```
In [481]:
```
'Asif' in myd1 # Membership test can be only done for keys.
```
Out[481]:
```
False
```
In [483]:
```
'ID' in myd1
```

```
Out[483]:  True

In [485]:  'Address' in myd1

Out[485]:  False
```

All / Any

The all() method returns: => True - If all keys of the are true => False - If all keys of the are false The any() function returns True if any key of the dictionary is True. If not, any() returns False

```
In [488]:  myd1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
           myd1

Out[488]:  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}

In [490]:  all(myd1) # Will return false as one value is false(Vale 0)

Out[490]:  True

In [ ]:
```