```
s5={2,3,'nit',1+2j,True}
```

```
2 in s5
```

```
True
```

```
200 in s5
```

```
False
```

```
for i in s5:
  print(i)
```

```
True
2
3
(1+2j)
nit
```

```
for i in enumerate (s5):
  print(i)
```

```
(0, True)
(1, 2)
(2, 3)
(3, (1+2j))
(4, 'nit')
```

```
s5.update(400,500)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-290a8aa22e7b> in <cell line: 0>()
----> 1 s5.update(400,500)

TypeError: 'int' object is not iterable
```

```
s5
```

```
{(1+2j), 2, 3, True, 'nit'}
```

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

```
A.union(B)
```

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

```
print(A)
print(B)
print(C)
```

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

```
A | B | C
```

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

```
A | B
```

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

```
print(A)
print(B)
print(C)
```

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

```
A&B
```

```
{4, 5}
```

```
A.intersection(B)
```

```
{4, 5}
```

```
A.intersection(C)
```

```
set()
```

```
B.intersection(C)
```

```
{8}
```

```
A.intersection(B,C)
```

```
set()
```

```
print(A)
print(B)
print(C)
```

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

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

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

```
A-B
```

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

```
A-C
```

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

```
C-A
```

```
{8, 9, 10}
```

```
print(A)
print(B)
print(C)
```

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

```
A.symmetric_difference(B)
```

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

```
A.symmetric_difference(C)
```

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

```
B.symmetric_difference(C)
```

```
{4, 5, 6, 7, 9, 10}
```

Archana Kapu
Feb 1, 2025

Symmetric difference indicates with ^

```
A1={1,2,3,4,5,6,7,8,9}
B1={3,4,5,6,7,8}
C1={10,20,30,40}
```

```
B1.issubset(A1)
```

```
True
```

```
A1.issubset(B1)
```

```
False
```

```
A1.issuperset(B1)
```

```
True
```

```
C1.issubset(A1)
```

```
False
```

```
B1.issubset(C1)
```

```
False
```

```
C1.isdisjoint(A1)
```

```
True
```

```
C1.isdisjoint(B1)
```

```
True
```

```
A2={1,2,3,4,5,6,7,8,9}
B2={13,14,15,16,17,18}
C3={10,20,30,40}
```

```
B2.issubset(A2)
```

```
False
```

```
A2.issubset(B2)
```

```
False
```

```
A2.issuperset(B2)
```

```
False
```

```
B2.isdisjoint(A2)
```

```
True
```

```
for i in A:
  print(i)
```

```
1
2
3
4
5
```

```
for i in enumerate (A):
  print(i)
```

```
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
```

```
list(enumerate( A))
```

```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]
```

set completed

Set task assignmnet

1.Duplicate elements are not allowed.

2.

```
myset = {1,2,3,4,5} # Set of numbers
myset
```

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

```
len(myset) #Length of the set
```

```
5
```

```
my_set = {1,1,2,2,3,4,5,5}
my_set # Duplicate elements are not allowed.
```

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

```
myset1 = {1.79,2.08,3.99,4.56,5.45} # Set of float numbers
myset1
```

```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

```
myset2 = {'Archana' , 'Harish' , 'Prakash'} # Set of Strings
myset2
```

```
{'Archana', 'Harish', 'Prakash'}
```

```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

```
{(11, 22, 32), 10, 20, 'Hola'}
```

```
myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items like list
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-97-7a9ce5d5e700> in <cell line: 0>()
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items like
list
      2 myset3

TypeError: unhashable type: 'list'
```

```
myset4 = set() # Create an empty set
print(type(myset4))
```

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

```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

```
{'four', 'one', 'three', 'two'}
```

Loop through a Set

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

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

```
for i in enumerate(myset):
  print(i)
```

```
(0, 'two')
(1, 'four')
(2, 'one')
```

```
(3, 'seven')
(4, 'six')
(5, 'five')
(6, 'three')
(7, 'eight')
```

Set Membership

```
myset
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
'eight' in myset# Check if 'eight' exist in the set
```

```
True
```

```
'eleven' in myset# Check if 'eleven' exist in the set
```

```
False
```

```
if 'eight' in myset:
  print('eight is present')
else:
  print('eight is not present')
```

```
eight is present
```

```
if 'eleven' in myset:
  print('eleven is present')
else:
  print('eleven is not present')
```

```
eleven is not present
```

Add & Remove Items

```
myset
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
myset.add('nine') # Add an item to the set
myset
```

```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

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

```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

```
myset.remove('nine') # remove item in a set using remove() method
myset
```

```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
```

```
 'three',
 'two'}
```

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

```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

```
myset.clear() # Delete all items in a set
myset
```

```
set()
```

```
myset
```

```
set()
```

```
del myset # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-116-8096fc3734fe> in <cell line: 0>()
      1 del myset # Delete the set object
----> 2 myset

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

Copy Set

```
myset={'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
myset1=myset#create a new reference 'myset1'
```

```
myset1
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
id(myset) , id(myset1) # The address of both myset & myset1 will be the same as
```

```
(134196586082176, 134196586082176)
```

```
my_set=myset.copy()## Create a copy of the list
my_set
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

```
id(my_set) # The address of my_set will be different from myset because my_set is reference of
```

```
134196586085760
```

```
myset.add('nine')
```

```
myset
```

```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
myset1# myset1 will be also impacted as it is pointing to the same Set
```

```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

```
my_set# Copy of the set won't be impacted due to changes made on the original Set
```

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

Union

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

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

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

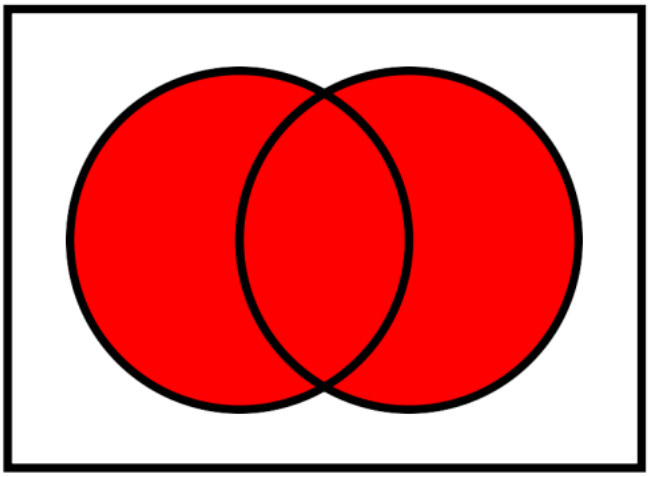
```
A.union(B) # Union of A and B
```

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

```
A.union(B,C) # Union of A, B and C
```

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



```
A.update(B,C)
A
```

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

Intersection

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

```
A & B# Intersection of A and B (Common items in both sets)
```

```
{4, 5}
```

```
A.intersection(B)
```

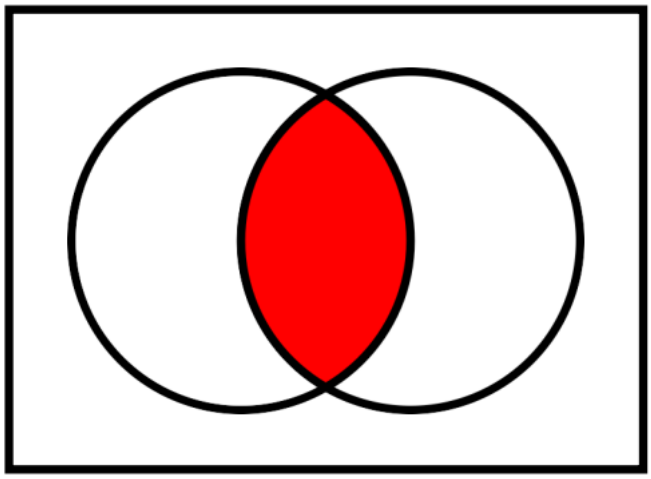
```
{4, 5}
```

Archana Kapu
6:01 PM Today
We should mention A.intersection(B)

```
A.intersection(B) Intersection of A and B
```

```
  File "<ipython-input-142-4e0e1fd8698c>", line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

""" 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



```
A.intersection_update(B)
A
```

```
{4, 5}
```

Difference

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

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

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

```
A.difference(B) # Difference of set
```

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

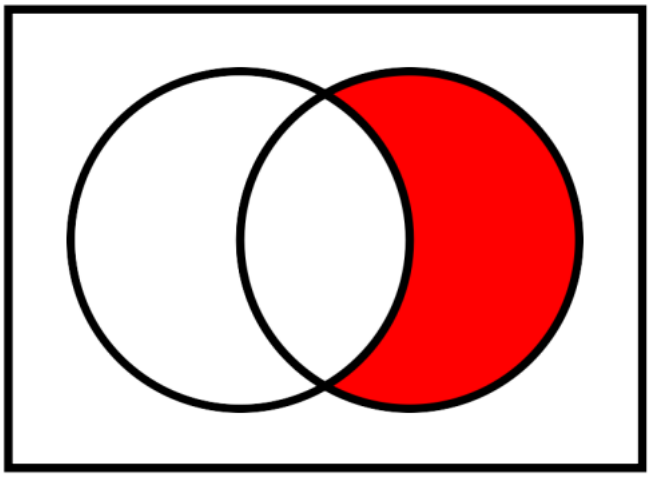
```
B- A # set of elements that are only in B but not in A
```

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

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

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

""" 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



```
B.difference_update(A)
B
```

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

Symmetric Difference

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

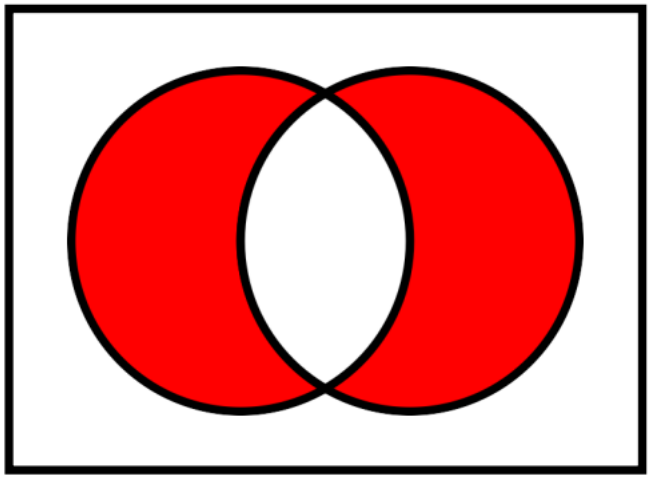
```
A.symmetric_difference(B)#Symmetric diff of 2 sets
```

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

```
A ^ B # Symmetric difference (Set of elements in A and B but not in both. "EXCLUSIE OF COMMON"
```

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

A ^ B # Symmetric difference (Set of elements in A and B but not in both. "EXCLU



```
A^B
```

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

Subset , Superset & Disjoint

```
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)## Set B is said to be the subset of set A if all elements of B are Presnt in A
```

```
True
```

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

```
True
```

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

```
True
```

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

```
False
```

Other Builtin functions

```
A
```

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

```
sum(A)
```

```
45
```

```
max(A)
```

```
9
```

```
min(A)
```

```
1
```

```
len(A)
```

```
9
```

```
list(enumerate(A))
```

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

```
D= sorted(A,reverse=True)
D
```

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

```
sorted(D)
```

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

Set task completed

Dict Task

create dictionary

```
my_dict=dict()#creating an empty dict
```

```
my_dict
```

```
{}
```

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

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

```
mydict = {'A':'one' , 'B':'two' , 'C':'three'} # dictionary with character keys
mydict
```

```
{'A': 'one', 'B': 'two', 'C': 'three'}
```

Archana Kapu
12:50 PM Today

dictionary with list,tuple,int values

```
mydict = {1:'one' , 'A':'two' , 3:'three'} # dictionary with mixed keys
mydict
```

```
{1: 'one', 'A': 'two', 3: 'three'}
```

```
mydict.keys() # Return Dictionary Keys using keys() method
```

```
dict_keys([1, 'A', 3])
```

Archana Kapu
12:51 PM Today

dictionary with in the disctionary as values possible

```
mydict.values() # Return Dictionary Values using values() method
```

```
dict_values(['one', 'two', 'three'])
```

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

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

```
mydict = {1:'one' , 2:'two' , 'A':['Archana' , 'Harish' , 'Sony']} # dictionary with different
mydict
```

```
{1: 'one', 2: 'two', 'A': ['Archana', 'Harish', 'Sony']}
```

```
mydict = {1:'one' , 2:'two' , 'A':['Archana' , 'Harish' , 'Sony'], 'B':('Bat' , 'cat','hat')}
mydict
```

```
{1: 'one',
 2: 'two',
 'A': ['Archana', 'Harish', 'Sony'],
 'B': ('Bat', 'cat', 'hat')}
```

```
mydict = {1:'one' , 2:'two' , 'A':{'Name':'Archana' , 'Age' :25}, 'B':('Bat' , 'cat','hat')}
mydict
```

```
{1: 'one',
 2: 'two',
 'A': {'Name': 'Archana', 'Age': 25},
 'B': ('Bat', 'cat', 'hat')}
```

Archana Kapu
1:05 PM Today

we can appened values to a dictionary

```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys) # Create a dictionary from a sequence of keys
mydict3
```

```
{'c': None, 'a': None, 'b': None, 'd': None}
```

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

```
{'c': 10, 'a': 10, 'b': 10, 'd': 10}
```

```
keys = {'a' , 'b' , 'c' , 'd'}
value = [10,20,30]
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of keys and values
mydict3
```

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

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

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

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

Accessing Items

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

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

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

```
'one'
```

```
mydict.get(1) # Access item using get() method
```

```
'one'
```

```
mydict1 = {'Name':'Archana' , 'ID': 5387 , 'DOB': 1995 , 'job' :'Data Analyst'}
mydict1
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'job': 'Data Analyst'}
```

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

```
'Archana'
```

Archana Kapu
2:58 PM Today
here we upadted dob as dic1 value in mydic1

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

```
'Data Analyst'
```

Add, Remove & Change Items

```
mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict1
```

```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

Archana Kapu
3:08 PM Today
by using pop menthod we can remove specified item

```
mydict1['Name']='Archana'
mydict1['DOB']=1995
```

```
mydict1
```

```
{'Name': 'Archana', 'ID': 12345, 'DOB': 1995, 'Address': 'Hilsinki'}
```

Archana Kapu
3:07 PM Today
by using popitem ,item will be removed randomly

```
dict1 = {'DOB':1996}
mydict1.update(dict1)
mydict1
```

```
{'Name': 'Archana', 'ID': 12345, 'DOB': 1996, 'Address': 'Hilsinki'}
```

Archana Kapu
3:06 PM Today
by using del menthod we can delte specified Key and its value

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

```
{'Name': 'Archana',
 'ID': 12345,
 'DOB': 1996,
 'Address': 'Hilsinki',
 'Job': 'Analyst'}
```

```
mydict1.pop('Job') # Removing items in the dictionary using Pop method
mydict1
```

```
{'Name': 'Archana', 'ID': 12345, 'DOB': 1996, 'Address': 'Hilsinki'}
```

Archana Kapu
3:11 PM Today
we can delete the dictionary by using del function

```
mydict1.popitem() # A random item is removed
```

```
('Address', 'Hilsinki')
```

```
mydict1
```

```
{'Name': 'Archana', 'ID': 12345, 'DOB': 1996}
```

```
del[mydict1['ID']] # Removing item using del method
mydict1
```

```
{'Name': 'Archana', 'DOB': 1996}
```

```
mydict1.clear() # Delete all items of the dictionary using clear method
mydict1
```

```
{}
```

```
mydict1
```

```
{}
```

```
del mydict1 # Delete the dictionary object
mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-59-da2fba4eca0f> in <cell line: 0>()
      1 del mydict1 # Delete the dictionary object
----> 2 mydict1

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

Copy Dictionary

```
mydict = {'Name':'Archana' , 'ID':5387 , 'DOB': 1995 , 'Address' : 'Anantapur'}
mydict
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'Address': 'Anantapur'}
```

Archana Kapu
3:22 PM Today

here address value is impacted when we change adderess value of mydict, as it is pointing to mydict1

```
mydict1=mydict#creat a new reference mydict1
```

```
id(mydict),id(mydict1)#the address  of both dictionaries  will be same
```

```
(134196585234944, 134196585234944)
```

Archana Kapu
3:24 PM Today

Copy of list won't be impacted due to the changes made in the original

```
mydict2=mydict.copy()#create a copy of the dictioanriy
```

```
id(mydict2)#the address og mydict2 will be different as we are coppying into new dictiaonry
```

```
134197464808384
```

```
mydict
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'Address': 'Anantapur'}
```

```
mydict['Address']='Mumbai'
```

```
mydict
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'Address': 'Mumbai'}
```

```
mydict1## mydict1 will be also impacted as it is pointing to the same dictionary
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'Address': 'Mumbai'}
```

Archana Kapu
3:31 PM Today

this syntax prints values of a dictionary

```
mydict2 # Copy of list won't be impacted due to the changes made in the original
```

```
{'Name': 'Archana', 'ID': 5387, 'DOB': 1995, 'Address': 'Anantapur'}
```

Loop through a Dictionary

```
mydict1 = {'Name':'Archana' , 'ID': 5387 , 'DOB': 1995 , 'Address' : 'ATP' ,'Job': 'Analyst'}
mydict1
```

```
{'Name': 'Archana',
 'ID': 5387,
 'DOB': 1995,
 'Address': 'ATP',
 'Job': 'Analyst'}
```

```
for i in mydict1:
  print(i ,':',mydict1[i])#key & value pair
```

```
Name : Archana
ID : 5387
DOB : 1995
Address : ATP
Job : Analyst
```

```
for i in mydict1:
      print(mydict1[i]) # Dictionary items
```

```
Archana
5387
1995
ATP
Analyst
```

Dictionary Membership

```
mydict1 = {'Name':'Archana' , 'ID': 5387 , 'DOB': 1995 , 'Address' : 'ATP' ,'Job': 'Analyst'}
mydict1
```

```
{'Name': 'Archana',
 'ID': 5387,
 'DOB': 1995,
 'Address': 'ATP',
 'Job': 'Analyst'}
```

```
'Name' in mydict1#Test if the key is in dictioanry or not
```

```
True
```

```
'Archana' in mydict1#membership test can be done on keys only
```

```
False
```

```
'ID' in mydict1
```

```
True
```

```
'name' in mydict1
```

```
False
```

All / Any

The all() method returns:

True - If all all keys of the dictionary are true

False - If any key of the dictionary is false

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

```
mydict1 = {'Name':'Archana' , 'ID': 5387 , 'DOB': 1995 , 'Address' : 'ATP' ,'Job': 'Analyst'}
mydict1
```

```
{'Name': 'Archana',
 'ID': 5387,
 'DOB': 1995,
 'Address': 'ATP',
 'Job': 'Analyst'}
```

```
all(mydict1)
```

```
True
```

```
any(mydict1)
```

```
True
```

class Explanation of Dict

```
d={}#empty dictionary
```

```
type(d)
```

```
dict
```

```
d={1:'one',2:'two',3:'three'}#dictionary with integer values
d
```

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

```
d.keys()
```

```
dict_keys([1, 2, 3])
```

```
d.values()
```

```
dict_values(['one', 'two', 'three'])
```

```
d[1]
```

```
'one'
```

```
d['one']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-195257aa2718> in <cell line: 0>()
----> 1 d['one']

KeyError: 'one'
```