set

In [2]:
```
s={}
s
```

Out[2]:
```
{}
```

In [4]:
```
type(s)
```

Out[4]:
```
dict
```

In [6]:
```
s1=set()
type(s1)
```

Out[6]:
```
set
```

In [8]:
```
s1
```

Out[8]:
```
set()
```

In [10]:
```
s2={20,100,3,45}
s2
```

Out[10]:
```
{3, 20, 45, 100}
```

In [12]:
```
s3={'z', 'l', 'c', 'e','f'}
s3
```

Out[12]:
```
{'c', 'e', 'f', 'l', 'z'}
```

In [14]:
```
s4 = {1, 2.3, 'nit', 1+2j, [1,2,3], (4,5,6), True}
s4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 s4 = {1, 2.3, 'nit', 1+2j, [1,2,3], (4,5,6), True}
      2 s4

TypeError: unhashable type: 'list'
```

In [16]:
```
s5 = {2, 3.4, 'nit', 1+2j, False}
```

In [18]:
```
s5
```

Out[18]:
```
{(1+2j), 2, 3.4, False, 'nit'}
```

In [20]:
```
print(s1)
print(s2)
print(s3)
print(s5)
```

```
set()
{45, 3, 100, 20}
{'c', 'e', 'f', 'z', 'l'}
{False, 2, 3.4, (1+2j), 'nit'}
```

In [22]:
```
s2
```
Out[22]:
```
{3, 20, 45, 100}
```

In [24]:
```
s2.add(30)
```

In [26]:
```
s2
```
Out[26]:
```
{3, 20, 30, 45, 100}
```

In [28]:
```
s2.add(200)
```

In [30]:
```
s2
```
Out[30]:
```
{3, 20, 30, 45, 100, 200}
```

In [32]:
```
s2
```
Out[32]:
```
{3, 20, 30, 45, 100, 200}
```

In [34]:
```
s2
```
Out[34]:
```
{3, 20, 30, 45, 100, 200}
```

In [36]:
```
s2[:]
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 s2[:]

TypeError: 'set' object is not subscriptable
```

In [38]:
```
s2
```
Out[38]:
```
{3, 20, 30, 45, 100, 200}
```

In [40]:
```
s2[1:5]
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[40], line 1
----> 1 s2[1:5]

TypeError: 'set' object is not subscriptable
```

In [42]:
```
s5
```
Out[42]:
```
{(1+2j), 2, 3.4, False, 'nit'}
```

In [44]:
```
s4 = s5.copy()
s4
```

```
Out[44]:  {(1+2j), 2, 3.4, False, 'nit'}

In [46]:  s4

Out[46]:  {(1+2j), 2, 3.4, False, 'nit'}

In [48]:  s4.add(2)

In [50]:  s4

Out[50]:  {(1+2j), 2, 3.4, False, 'nit'}

In [52]:  s5

Out[52]:  {(1+2j), 2, 3.4, False, 'nit'}

In [54]:  s4.add(5)
          s4

Out[54]:  {(1+2j), 2, 3.4, 5, False, 'nit'}

In [56]:  s4.add(5)
          s4

Out[56]:  {(1+2j), 2, 3.4, 5, False, 'nit'}

In [58]:  s5.clear()

In [60]:  s5

Out[60]:  set()

In [62]:  del s5

In [64]:  s5

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[64], line 1
----> 1 s5

NameError: name 's5' is not defined

In [66]:  s4

Out[66]:  {(1+2j), 2, 3.4, 5, False, 'nit'}

In [68]:  s4.remove((1+2j))

In [70]:  s4

Out[70]:  {2, 3.4, 5, False, 'nit'}

In [72]:  s3
```

```
Out[72]: {'c', 'e', 'f', 'l', 'z'}

In [74]: s3.discard('m')

In [76]: s3

Out[76]: {'c', 'e', 'f', 'l', 'z'}

In [78]: s3.remove('m')

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[78], line 1
----> 1 s3.remove('m')

KeyError: 'm'

In [80]: s3

Out[80]: {'c', 'e', 'f', 'l', 'z'}

In [82]: s3.discard('f')
         s3

Out[82]: {'c', 'e', 'l', 'z'}

In [84]: s3

Out[84]: {'c', 'e', 'l', 'z'}

In [86]: s3.pop()

Out[86]: 'c'

In [88]: s3

Out[88]: {'e', 'l', 'z'}

In [90]: s2

Out[90]: {3, 20, 30, 45, 100, 200}

In [92]: s2.pop(3)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[92], line 1
----> 1 s2.pop(3)

TypeError: set.pop() takes no arguments (1 given)

In [94]: s2.pop()

Out[94]: 3

In [96]: for i in s2:
             print(i)
```

```
100
200
45
20
30
```

In [98]:
```
for i in enumerate(s2):
    print(i)
```

```
(0, 100)
(1, 200)
(2, 45)
(3, 20)
(4, 30)
```

In [100…
```
s2
```

Out[100…
```
{20, 30, 45, 100, 200}
```

In [102…
```
5 in s2
```

Out[102…
```
False
```

In [104…
```
45 in s2
```

Out[104…
```
True
```

In [106…
```
s2
```

Out[106…
```
{20, 30, 45, 100, 200}
```

In [108…
```
s3
```

Out[108…
```
{'e', 'l', 'z'}
```

In [110…
```
s2.update(s3)
```

In [112…
```
s2
```

Out[112…
```
{100, 20, 200, 30, 45, 'e', 'l', 'z'}
```

SET OPERATION

In [115…
```
s6 = {1,2,3,4,5}
s7 = {4,5,6,7,8}
s8 = {8,9,10}
```

In [117…
```
s6.union(s7)
```

Out[117…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [119…
```
s6.union(s7, s8)
```

Out[119…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [121…
```
s6 | s7
```
Out[121…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [123…
```
s6 | s7 | s8
```
Out[123…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [125…
```
print(s6)
print(s7)
print(s8)
```
```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [127…
```
s6.intersection(s7)
```
Out[127…
```
{4, 5}
```

In [129…
```
s6.intersection(s8)
```
Out[129…
```
set()
```

In [131…
```
s7.intersection(s8)
```
Out[131…
```
{8}
```

In [133…
```
s6 & s7
```
Out[133…
```
{4, 5}
```

In [135…
```
s6.difference(s7)
```
Out[135…
```
{1, 2, 3}
```

In [137…
```
s6 - s7
```
Out[137…
```
{1, 2, 3}
```

In [139…
```
s7 - s8
```
Out[139…
```
{4, 5, 6, 7}
```

In [141…
```
print(s6)
print(s7)
print(s8)
```
```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [143…
```
s8 - s7
```
Out[143…
```
{9, 10}
```

In [145…
```
print(s6)
print(s7)
print(s8)
```
```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}
```

In [147…
```
s6.symmetric_difference(s7)
```
Out[147…
```
{1, 2, 3, 6, 7, 8}
```

In [149…
```
s10 = {50, 4, 3, 10}
s10
```
Out[149…
```
{3, 4, 10, 50}
```

In [151…
```
print(s10)
```
```
{10, 3, 50, 4}
```

In [153…
```
print(s10)
```
```
{10, 3, 50, 4}
```

- superset
- subset
- disjoint

In [156…
```
s11 = {1,2,3,4,5,6,7,8,9}
s12 = {3,4,5,6,7,8}
s13 = {10,20,30,40}
```

In [158…
```
s12.issubset(s11)
```
Out[158…
```
True
```

In [160…
```
s11.issubset(s12)
```
Out[160…
```
False
```

In [162…
```
s11.issuperset(s12)
```
Out[162…
```
True
```

In [164…
```
s11 = {1,2,3,4,5,6,7,8,9}
s12 = {3,4,5,6,7,8}
s13 = {10,20,30,40}
```

In [166…
```
s13.isdisjoint(s12)
```
Out[166…
```
True
```

In [168…
```
s13.isdisjoint(s11)
```
Out[168…
```
True
```

In [170…
```
s12 = {1,2,3,4,5}
s13 = {10,20,30}
s14 = {15,25,35}
```

In [172…
```
s13.issubset(s12)
```

Out[172…
```
False
```

In [174…
```
s12.issuperset(s13)
```

Out[174…
```
False
```

In [176…
```
s14.isdisjoint(s12)
```

Out[176…
```
True
```

In [178…
```
s14.isdisjoint(s13)
```

Out[178…
```
True
```

In [180…
```
s15 = {1,2,3,4,5,6}
s16 = {4,5,6}
s17 = {10,20}
```

In [182…
```
s16.issubset(s15)
```

Out[182…
```
True
```

In [184…
```
s17.isdisjoint(s15)
```

Out[184…
```
True
```

In [186…
```
s17.isdisjoint(s16)
```

Out[186…
```
True
```

In [188…
```
s15
```

Out[188…
```
{1, 2, 3, 4, 5, 6}
```

In [190…
```
for i in s15:
    print(i)
```

```
1
2
3
4
5
6
```

In [192…
```
for i in enumerate(s15):
    print(i)
```

```
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
```

In [194…
```
s15
```
Out[194…
```
{1, 2, 3, 4, 5, 6}
```

In [196…
```
sum(s15)
```
Out[196…
```
21
```

In [198…
```
min(s15)
```
Out[198…
```
1
```

dictionary

In [201…
```
d = {}
d
```
Out[201…
```
{}
```

In [203…
```
type(d)
```
Out[203…
```
dict
```

In [205…
```
d1 = {1 : 'one', 2 : 'two', 3: 'three'}
d1
```
Out[205…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [207…
```
d1.keys()
```
Out[207…
```
dict_keys([1, 2, 3])
```

In [209…
```
d1.values()
```
Out[209…
```
dict_values(['one', 'two', 'three'])
```

In [211…
```
d2 = d1.copy()
d2
```
Out[211…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [213…
```
d1.items()
```
Out[213…
```
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
```

In [215…
```
d1[1]
```

```
Out[215… 'one'
```

```
In [217… d1[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[217], line 1
----> 1 d1[0]

KeyError: 0
```

```
In [219… keys = {'ram' , 'b' , 'c' , 'd'}
         value = [10,20,30]
         mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of
         mydict3
```

```
Out[219… {'d': [10, 20, 30], 'b': [10, 20, 30], 'c': [10, 20, 30], 'ram': [10, 20, 30]}
```

```
In [221… value.append(50)
         mydict3
```

```
Out[221… {'d': [10, 20, 30, 50],
          'b': [10, 20, 30, 50],
          'c': [10, 20, 30, 50],
          'ram': [10, 20, 30, 50]}
```

```
In [223… range(10)
```

```
Out[223… range(0, 10)
```

```
In [225… list(range(0,10))
```

```
Out[225… [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Set and Dict PDF

Set Creation

```
In [231… myset = {1,2,3,4,5} # Set of numbers
         myset
```

```
Out[231… {1, 2, 3, 4, 5}
```

```
In [233… len(myset) #Length of the set
```

```
Out[233… 5
```

```
In [235… my_set = {1,1,2,2,3,4,5,5}
         my_set # Duplicate elements are not allowed.
```

```
Out[235… {1, 2, 3, 4, 5}
```

```
In [237… myset1 = {1.79,2.08,3.99,4.56,5.45} # Set of float numbers
         myset1
```

Out[237…
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [239…
```
myset2 = {'Asif' , 'John' , 'Tyrion'} # Set of Strings
myset2
```

Out[239…
```
{'Asif', 'John', 'Tyrion'}
```

In [245…
```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

Out[245…
```
{(11, 22, 32), 10, 20, 'Hola'}
```

In [247…
```
myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items like li
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[247], line 1
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items
like li
      2 myset3

TypeError: unhashable type: 'list'
```

In [249…
```
myset4 = set() # Create an empty set
print(type(myset4))
```

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

In [253…
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

Out[253…
```
{'four', 'one', 'three', 'two'}
```

Loop through a Set

In [257…
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

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

In [259…
```
for i in enumerate(myset):
    print(i)
```

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

Set Membership

In [262...
```
myset
```
Out[262...
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [264...
```
'one' in myset # Check if 'one' exist in the set
```
Out[264...
```
True
```

In [266...
```
'ten' in myset # Check if 'ten' exist in the set
```
Out[266...
```
False
```

In [268...
```
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 [270...
```
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 [273...
```
myset
```
Out[273...
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [275...
```
myset.add('NINE') # Add item to a set using add() method
myset
```
Out[275...
```
{'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [277...
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE']) # Add multiple item to a set using
myset
```

```
Out[277…  {'ELEVEN',
           'NINE',
           'TEN',
           'TWELVE',
           'eight',
           'five',
           'four',
           'one',
           'seven',
           'six',
           'three',
           'two'}
```

```
In [279…  myset.remove('NINE') # remove item in a set using remove() method
          myset
```

```
Out[279…  {'ELEVEN',
           'TEN',
           'TWELVE',
           'eight',
           'five',
           'four',
           'one',
           'seven',
           'six',
           'three',
           'two'}
```

```
In [281…  myset.discard('TEN') # remove item from a set using discard() method
          myset
```

```
Out[281…  {'ELEVEN',
           'TWELVE',
           'eight',
           'five',
           'four',
           'one',
           'seven',
           'six',
           'three',
           'two'}
```

```
In [283…  myset.clear() # Delete all items in a set
          myset
```

```
Out[283…  set()
```

```
In [285…  del myset # Delete the set object
          myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[285], line 2
      1 del myset # Delete the set object
----> 2 myset

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

Copy Set

```
In [290…  myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
          myset

Out[290…  {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [292…  myset1 = myset # Create a new reference "myset1"
          myset1

Out[292…  {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [294…  id(myset) , id(myset1) # The address of both myset & myset1 will be the same as

Out[294…  (2846446922528, 2846446922528)

In [296…  my_set = myset.copy() # Create a copy of the list
          my_set

Out[296…  {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [298…  id(my_set) # The address of my_set will be different from myset because my_set i

Out[298…  2846446912672

In [302…  myset.add('nine')
          myset

Out[302…  {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [304…  myset1 # myset1 will be also impacted as it is pointing to the same Set

Out[304…  {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [306…  my_set # Copy of the set won't be impacted due to changes made on the original S

Out[306…  {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

Union

```
In [310…  A = {1,2,3,4,5}
          B = {4,5,6,7,8}
          C = {8,9,10}

In [312…  A | B # Union of A and B (All elements from both sets. NO DUPLICATES)

Out[312…  {1, 2, 3, 4, 5, 6, 7, 8}

In [314…  A.union(B) # Union of A and B

Out[314…  {1, 2, 3, 4, 5, 6, 7, 8}

In [316…  A.union(B, C) # Union of A, B and C.
```

Out[316…

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

In [320…

```
A.update(B,C) #Updates the set calling the update() method with union of A , B &
A
```

Out[320…

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

Intersection

In [323…

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
A & B # Intersection of A and B (Common items in both sets)
```

Out[323…

```
{4, 5}
```

In [327…

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

Out[327…

```
{4, 5}
```

In [333…

```
A.intersection_update(B) # Updates the set calling the intersection_update() met
A
```

Out[333…

```
{4, 5}
```

Difference

In [336…

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

In [338…

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

Out[338…

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

In [340…

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

Out[340…

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

In [342…

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

Out[342…

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

In [344…

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

Out[344…

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

In [348…

```
B.difference_update(A) # Updates the set calling the difference_update() method
B
```

Out[348…

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

Symmetric Difference

In [351…
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [353…
```
A ^ B # Symmetric difference (Set of elements in A and B but not in both. "EXCLU
```
Out[353…
```
{1, 2, 3, 6, 7, 8}
```

In [355…
```
A.symmetric_difference(B) # Symmetric difference of sets
```
Out[355…
```
{1, 2, 3, 6, 7, 8}
```

In [357…
```
A.symmetric_difference_update(B) # Updates the set calling the symmetric_differe
A
```
Out[357…
```
{1, 2, 3, 6, 7, 8}
```

Subset , Superset & Disjoint

In [360…
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [362…
```
B.issubset(A) # Set B is said to be the subset of set A if all elements of B are
```
Out[362…
```
True
```

In [364…
```
A.issuperset(B) # Set A is said to be the superset of set B if all elements of B
```
Out[364…
```
True
```

In [366…
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[366…
```
True
```

In [368…
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```
Out[368…
```
False
```

Other Builtin functions

In [371…
```
A
```
Out[371…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [373…
```
sum(A)
```
Out[373…
```
45
```

In [375…
```
max(A)
```
Out[375…
```
9
```

In [377…
```
min(A)
```

```
Out[377…  1

In [379…  len(A)

Out[379…  9

In [381…  list(enumerate(A))

Out[381…  [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

In [385…  D= sorted(A,reverse=True)
          D

Out[385…  [9, 8, 7, 6, 5, 4, 3, 2, 1]

In [387…  sorted(D)

Out[387…  [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Dictionary

Create Dictionary

```
In [391…  mydict = dict() # empty dictionary
          mydict

Out[391…  {}

In [393…  mydict = {} # empty dictionary
          mydict

Out[393…  {}

In [395…  mydict = {1:'one' , 2:'two' , 3:'three'} # dictionary with integer keys
          mydict

Out[395…  {1: 'one', 2: 'two', 3: 'three'}

In [397…  mydict = dict({1:'one' , 2:'two' , 3:'three'}) # Create dictionary using dict()
          mydict

Out[397…  {1: 'one', 2: 'two', 3: 'three'}

In [399…  mydict = {'A':'one' , 'B':'two' , 'C':'three'} # dictionary with character keys
          mydict

Out[399…  {'A': 'one', 'B': 'two', 'C': 'three'}

In [401…  mydict = {1:'one' , 'A':'two' , 3:'three'} # dictionary with mixed keys
          mydict

Out[401…  {1: 'one', 'A': 'two', 3: 'three'}
```

In [403…
```
mydict.keys() # Return Dictionary Keys using keys() method
```
Out[403…
```
dict_keys([1, 'A', 3])
```
In [405…
```
mydict.values() # Return Dictionary Values using values() method
```
Out[405…
```
dict_values(['one', 'two', 'three'])
```
In [407…
```
mydict.items() # Access each key-value pair within a dictionary
```
Out[407…
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```
In [409…
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria']} # dictionary with
mydict
```
Out[409…
```
{1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}
```
In [413…
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria'], 'B':('Bat' , 'cat
mydict
```
Out[413…
```
{1: 'one',
 2: 'two',
 'A': ['asif', 'john', 'Maria'],
 'B': ('Bat', 'cat', 'hat')}
```
In [429…
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[429], line 1
----> 1 mydict[0:3,1]

KeyError: (slice(0, 3, None), 1)
```
In [431…
```
mydict = {1:'one' , 2:'two' , 'A':{'Name':'asif' , 'Age' :20}, 'B':('Bat' , 'cat
mydict
```
Out[431…
```
{1: 'one',
 2: 'two',
 'A': {'Name': 'asif', 'Age': 20},
 'B': ('Bat', 'cat', 'hat')}
```
In [435…
```
mydict['A']
```
Out[435…
```
{'Name': 'asif', 'Age': 20}
```
In [437…
```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys) # Create a dictionary from a sequence of keys
mydict3
```
Out[437…
```
{'d': None, 'b': None, 'a': None, 'c': None}
```
In [441…
```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of
mydict3
```

```
Out[441… {'d': 10, 'b': 10, 'a': 10, 'c': 10}
```

```
In [445… keys = {'a' , 'b' , 'c' , 'd'}
         value = [10,20,30]
         mydict3 = dict.fromkeys(keys , value) # Create a dictionary from a sequence of
         mydict3
```

```
Out[445… {'d': [10, 20, 30], 'b': [10, 20, 30], 'a': [10, 20, 30], 'c': [10, 20, 30]}
```

```
In [447… value.append(40)
         mydict3
```

```
Out[447… {'d': [10, 20, 30, 40],
          'b': [10, 20, 30, 40],
          'a': [10, 20, 30, 40],
          'c': [10, 20, 30, 40]}
```

Accessing Items

```
In [450… mydict = {1:'one' , 2:'two' , 3:'three' , 4:'four'}
         mydict
```

```
Out[450… {1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

```
In [452… mydict[1] # Access item using key
```

```
Out[452… 'one'
```

```
In [454… mydict.get(1) # Access item using get() method
```

```
Out[454… 'one'
```

```
In [458… mydict1 = {'Name':'Asif' , 'ID': 74123 , 'DOB': 1991 , 'job' :'Analyst'}
         mydict1
```

```
Out[458… {'Name': 'Asif', 'ID': 74123, 'DOB': 1991, 'job': 'Analyst'}
```

```
In [460… mydict1['Name'] # Access item using key
```

```
Out[460… 'Asif'
```

```
In [462… mydict1.get('job') # Access item using get() method
```

```
Out[462… 'Analyst'
```

Add, Remove & Change Items

```
In [465… mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
         mydict1
```

```
Out[465… {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

```
In [469… mydict1['DOB'] = 1992 # Changing Dictionary Items
         mydict1['Address'] = 'Delhi'
```

```
mydict1
```

Out[469…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1992, 'Address': 'Delhi'}
```

In [471…
```
dict1 = {'DOB':1995}
mydict1.update(dict1)
mydict1
```

Out[471…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}
```

In [473…
```
mydict1['Job'] = 'Analyst' # Adding items in the dictionary
mydict1
```

Out[473…
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1995,
 'Address': 'Delhi',
 'Job': 'Analyst'}
```

In [475…
```
mydict1.pop('Job') # Removing items in the dictionary using Pop method
mydict1
```

Out[475…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}
```

In [477…
```
mydict1.popitem() # A random item is removed
```

Out[477…
```
('Address', 'Delhi')
```

In [479…
```
mydict1
```

Out[479…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995}
```

In [481…
```
del[mydict1['ID']] # Removing item using del method
mydict1
```

Out[481…
```
{'Name': 'Asif', 'DOB': 1995}
```

In [483…
```
mydict1.clear() # Delete all items of the dictionary using clear method
mydict1
```

Out[483…
```
{}
```

In [485…
```
del mydict1 # Delete the dictionary object
mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[485], line 2
      1 del mydict1 # Delete the dictionary object
----> 2 mydict1

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

Copy Dictionary

In [488…
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
```

```
mydict
```

Out[488…

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

In [490…

```
mydict1 = mydict # Create a new reference "mydict1"
```

In [492…

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

Out[492…

```
(2846445490560, 2846445490560)
```

In [494…

```
mydict2 = mydict.copy() # Create a copy of the dictionary
```

In [496…

```
id(mydict2) # The address of mydict2 will be different from mydict because mydic
```

Out[496…

```
2846457808448
```

In [498…

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

In [500…

```
mydict
```

Out[500…

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

In [502…

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

Out[502…

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

In [504…

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

Out[504…

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

Loop through a Dictionary

In [507…

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

Out[507…

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

In [511…

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

```
Name : Asif
ID : 12345
DOB : 1991
Address : Hilsinki
```

In [513…

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

```
Asif
12345
1991
Hilsinki
```

Dictionary Membership

```
In [516…  mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
          mydict1

Out[516…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}

In [518…  'Name' in mydict1 # Test if a key is in a dictionary or not.

Out[518…  True

In [520…  'Asif' in mydict1 # Membership test can be only done for keys.

Out[520…  False

In [522…  'ID' in mydict1

Out[522…  True

In [524…  'Address' in mydict1

Out[524…  False
```

All / Any

```
In [529…  mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
          mydict1

Out[529…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}

In [531…  all(mydict1) # Will Return false as one value is false (Value 0)

Out[531…  True

In [533…  any(mydict1)

Out[533…  True
```