Set Creation

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

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

In [4]:
```
len(myset) #Length of the set
```

Out[4]:
```
5
```

In [6]:
```
my_set ={1,1,2,2,3,4,5,5} # Duplicate elements are not allowed
my_set
```

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

In [8]:
```
myset1 ={1.79,2.08,3.99,4.56,5.45} #Set of float numbers
myset1
```

Out[8]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [12]:
```
myset2 ={'ajay','ramu','vijay'} # Set of Strings
myset2
```

Out[12]:
```
{'ajay', 'ramu', 'vijay'}
```

In [14]:
```
myset3 ={10,20,"hola",(11,22,33)} #Mixed datatypes
myset3
```

Out[14]:
```
{(11, 22, 33), 10, 20, 'hola'}
```

In [16]:
```
myset3 ={10,20,"hola",[11,22,33]} # set doesn't allow mutable items like list
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 myset3 ={10,20,"hola",[11,22,33]} # set doesn't allow mutable items like
list
      2 myset3

TypeError: unhashable type: 'list'
```

In [18]:
```
myset4 =set() # create an empty set
print(type(myset4))
```

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

In [22]:
```
my_set1 = set(('one','two','three','four'))
my_set1
```

Out[22]:
```
{'four', 'one', 'three', 'two'}
```

Loop through a set

In [26]:
```
myset = {'one','two','three','four','five','six','seven','eight'}
for i in myset:
    print(i)
```

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

In [32]:
```
for i in enumerate(myset):
     print(i)
```

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

Set Membership

In [36]:
```
myset
```

Out[36]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [38]:
```
'one' in myset #check if 'one'exist in the set
```

Out[38]:
```
True
```

In [40]:
```
'ten' in myset #check if 'ten'exist in the set
```

Out[40]:
```
False
```

In [47]:
```
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 [49]:
```
if 'eleven' in myset: #check if 'eleven'exist in the set
    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 [54]:
```
myset
```

Out[54]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [56]:
```
myset.add('nine') # add item to a set using add() method
myset
```

Out[56]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [58]:
```
myset.update(['ten','eleven','twelve']) # add multiple item to a set using
myset
```

Out[58]:
```
{'eight',
 'eleven',
 'five',
 'four',
 'nine',
 'one',
 'seven',
 'six',
 'ten',
 'three',
 'twelve',
 'two'}
```

In [60]:
```
myset.remove('nine') # remove item in a set using remove() method
myset
```

Out[60]:
```
{'eight',
 'eleven',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'ten',
 'three',
 'twelve',
 'two'}
```

In [62]:
```
myset.discard('ten') # remove item from a set using discard() method
myset
```

Out[62]:
```
{'eight',
 'eleven',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'twelve',
 'two'}
```

In [64]:
```
myset.clear() # delete all items in a set
myset
```

Out[64]:
```
set()
```

In [66]:
```
del myset # Delete the set object
myself
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[66], line 2
      1 del myset # Delete the set object
----> 2 myself

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

Copy Set

In [71]:
```
myset = {'one','two','three','four','five','six','seven','eight'}
myset
```

Out[71]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [73]:
```
myset1 = myset #Create a new reference "myset1"
myset1
```

Out[73]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [75]:
```
id(myset) , id(myset1) # the address of both myset & myset1 will same as
```

Out[75]:
```
(2442405851456, 2442405851456)
```

In [77]:
```
myset.add('nine')
myset
```

Out[77]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [79]:
```
myset1 #myset1 will be also impacted as it is pointing to the same set
```

Out[79]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [87]:
```
my_set ={'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
my_set
```

Out[87]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [89]:
```
my_set # copy of the set won't be impacted due to changes made on the  original
```

Out[89]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

union

In [112…
```
A = {1,2,3,4,5}
A
```

Out[112…
```
{1, 2, 3, 4, 5}
```

In [114…
```
B = {4,5,6,7,8}
B
```

Out[114…
```
{4, 5, 6, 7, 8}
```

In [116…
```
C = {8,9,10}
C
```

Out[116…
```
{8, 9, 10}
```

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

In [118…
```
A|B # union of Aand B (all elements from both sets.NO DUPLICATES)
```

Out[118…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [120…
```
A.union(B) # union of A and B
```

Out[120…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [122…
```
A.union(B,C) # union of A,B and C
```

Out[122…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

intersection

In [128…
```
A
```

Out[128…
```
{1, 2, 3, 4, 5}
```

In [130…
```
B
```

Out[130…
```
{4, 5, 6, 7, 8}
```

In [132…
```
A & B # Intersection of A and B (common items in both sets)
```

Out[132…
```
{4, 5}
```

In [136…
```
A.intersection(B)   Intersection of A and B
```

```
  Cell In[136], line 1
    A.intersection(B)   Intersection of A and B
                        ^
SyntaxError: invalid syntax
```

Difference

In [139…
```
A
```

Out[139… {1, 2, 3, 4, 5}

In [141…
```
B
```

Out[141… {4, 5, 6, 7, 8}

In [145…
```
A-B # set of elements that are only in A but not in B
```

Out[145… {1, 2, 3}

In [143…
```
A.difference(B) # Difference of sets
```

Out[143… {1, 2, 3}

In [147…
```
B-A #set of elements that are only in Bbut not in A
```

Out[147… {6, 7, 8}

In [149…
```
B.difference(A)
```

Out[149… {6, 7, 8}

Symmetric Difference

In [154…
```
A
```

Out[154… {1, 2, 3, 4, 5}

In [156…
```
B
```

Out[156… {4, 5, 6, 7, 8}

In [158…
```
A^B # Symmetric difference (set of elements in A and B but not in both)
```

Out[158… {1, 2, 3, 6, 7, 8}

In [160…
```
A.symmetric_difference(B) #Symmetric difference of sets
```

Out[160… {1, 2, 3, 6, 7, 8}

Subset,Superset,Disjoint

In [179…
```
A ={1,2,3,4,5,6,7,8,9}
A
```

Out[179… {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [181…
```
B = {3,4,5,6,7,8}
B
```

Out[181… {3, 4, 5, 6, 7, 8}

In [183…
```
C = {10,20,30,40}
C
```

Out[183…
```
{10, 20, 30, 40}
```

In [185…
```
B.issubset(A) #set B is said to be the subset of set A if all elements of B are
```

Out[185…
```
True
```

In [189…
```
A.issuperset(B) #set A is said to be the subset of set B if all elements of B ar
```

Out[189…
```
True
```

In [191…
```
C.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```

Out[191…
```
True
```

In [195…
```
B.isdisjoint(A) # Two sets are said to be disjoint sets if they have no common e
```

Out[195…
```
False
```

Other Builtin Functions (Sum,Max,Min,List,Len & Sorted)

In [200…
```
A
```

Out[200…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [202…
```
sum(A)
```

Out[202…
```
45
```

In [204…
```
max(A)
```

Out[204…
```
9
```

In [206…
```
min(A)
```

Out[206…
```
1
```

In [208…
```
len(A)
```

Out[208…
```
9
```

In [210…
```
list(enumerate(a))
```

Out[210…
```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

In [212…
```
D = sorted(A,reverse=True)
D
```

Out[212…
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [216…
```
sorted(D)
```

Out[216…
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Dictionary

In [ ]:
```
# dictionary is a mutable data type in python
# A pyton dictionary is the collection of key and value pairs separated by a col
# Keys must be unique in a dictionary ,duplicate values are allowed
```

Dictionary Creation

In [5]:
```
d = dict() # empty dictionary
d
```

Out[5]:
```
{}
```

In [7]:
```
d1 = {1:'one',2:'two',3:'three'} # dict with integer keys
d1
```

Out[7]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [9]:
```
d1 = dict({1:'one',2:'two',3:'three'}) # dictionary creation using dict function
d1
```

Out[9]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [11]:
```
d2 = {'a':'apple','b':'bat','c':'cat','d':'dog'} # dictionary with charater key
d2
```

Out[11]:
```
{'a': 'apple', 'b': 'bat', 'c': 'cat', 'd': 'dog'}
```

In [13]:
```
d3 = {1:'one','A':'two',3:'three'} # dict with mixed data type
d3
```

Out[13]:
```
{1: 'one', 'A': 'two', 3: 'three'}
```

In [15]:
```
d3.keys() # this returns keys using key() method
```

Out[15]:
```
dict_keys([1, 'A', 3])
```

In [17]:
```
d3.values() # values() method uses for values returning
```

Out[17]:
```
dict_values(['one', 'two', 'three'])
```

In [19]:
```
d3.items() # access each key-value pair within a dictionary
```

Out[19]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

In [23]:
```
d4 = {1:'one',2:'two','A':['python','jaya','css']} # dictionary with list
d4
```

Out[23]:
```
{1: 'one', 2: 'two', 'A': ['python', 'jaya', 'css']}
```

In [25]:
```
d5 = {1:'one',2:'two','A':['python','jaya','css'],'B':('apple','bat','cat')}
d5
```

Out[25]:
```
{1: 'one',
 2: 'two',
 'A': ['python', 'jaya', 'css'],
 'B': ('apple', 'bat', 'cat')}
```

In [31]:
```
d6 = {1:'one',2:'two','A':{'python':'jaya','age':20},'B':('apple','bat','cat')}
d6
```

Out[31]:
```
{1: 'one',
 2: 'two',
 'A': {'python': 'jaya', 'age': 20},
 'B': ('apple', 'bat', 'cat')}
```

In [33]:
```
d1
```

Out[33]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [35]:
```
keys = {'a','b','c','d'} # create a dict from sequence of keys and value
value = 20
d8 = dict.fromkeys(keys,value)
d8
```

Out[35]:
```
{'c': 20, 'b': 20, 'a': 20, 'd': 20}
```

In [37]:
```
keys = {'a','b','c','d'} # create a dict from sequence of keys and value
value = [10,20,30]
d9 = dict.fromkeys(keys,value)
d9
```

Out[37]:
```
{'c': [10, 20, 30], 'b': [10, 20, 30], 'a': [10, 20, 30], 'd': [10, 20, 30]}
```

In [39]:
```
value.append(40)
d9
```

Out[39]:
```
{'c': [10, 20, 30, 40],
 'b': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'd': [10, 20, 30, 40]}
```

Accesding items

In [42]:
```
d1
```

Out[42]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [44]:
```
d1[2]
```

Out[44]:
```
'two'
```

In [46]:
```
d1.get(1)
```

Out[46]: 'one'

In [48]:
```
d10 = {'Name':'Ramu','ID':23030939, 'DOB':2003,'Job':'analyst'}
d10
```

Out[48]: {'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst'}

In [ ]:
```
{'Name':'Ramu','ID':23030939, 'DOB':2003, 'Job':'analyst'}
```

In [60]:
```
d10['Name']
```

Out[60]: 'Ramu'

In [66]:
```
d10.get('Job')
```

Out[66]: 'analyst'

Add,Remove & Change items

In [70]:
```
d11 = {'Name':'Ramu','ID':23030939, 'DOB':2003, 'Job':'analyst','Address':'hyder
d11
```

Out[70]:
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [72]:
```
d11 = {'DOB':2005}
d11.update(d11)
d11
```

Out[72]: {'DOB': 2005}

In [74]:
```
d11['Job'] = 'analyst'
d11
```

Out[74]: {'DOB': 2005, 'Job': 'analyst'}

In [124…
```
d11.pop('Job')
d11
```

Out[124… {'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Address': 'hyderabad'}

In [119…
```
d11.popitem()
```

Out[119… ('ID', 23030939)

In [121…
```
d11 = {'Name':'Ramu','ID':23030939, 'DOB':2003, 'Job':'analyst','Address':'hyder
d11
```

Out[121…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [ ]:
```
d10 = {'Name':'Ramu','ID':23030939, 'DOB':2003,'Job':'analyst'}
```

In [134…
```
d11
```

Out[134…
```
{'Name': 'Ramu', 'DOB': 2003, 'Address': 'hyderabad'}
```

In [144…
```
del[d11['ID']]
d11
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[144], line 1
----> 1 del[d11['ID']]
      2 d11

KeyError: 'ID'
```

In [151…
```
d11.clear()
d11
```

Out[151…
```
{}
```

In [153…
```
del d11
d11
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[153], line 2
      1 del d11
----> 2 d11

NameError: name 'd11' is not defined
```

Copy Dictionary

In [156…
```
d12 =  {'Name':'Ramu','ID':23030939, 'DOB':2003, 'Job':'analyst','Address':'hyde
d12
```

Out[156…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [158…
```
dict1 = d12
dict1
```

Out[158…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [160…
```
id(d12), id(dict1)
```

Out[160…
```
(2322590248000, 2322590248000)
```

In [164…
```
dict2 = dict1.copy()
dict2
```

Out[164…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [166…
```
dict2['Address'] = 'Andhra prasdesh'
```

In [168…
```
dict2
```

Out[168…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'Andhra prasdesh'}
```

Loop through dict

In [171…
```
for i in dict2:
    print(dict2[i])
```

```
Ramu
23030939
2003
analyst
Andhra prasdesh
```

In [179…
```
for i in enumerate (dict2):
    print (dict2)
```

```
{'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst', 'Address': 'Andhr
a prasdesh'}
{'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst', 'Address': 'Andhr
a prasdesh'}
{'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst', 'Address': 'Andhr
a prasdesh'}
{'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst', 'Address': 'Andhr
a prasdesh'}
{'Name': 'Ramu', 'ID': 23030939, 'DOB': 2003, 'Job': 'analyst', 'Address': 'Andhr
a prasdesh'}
```

Dict Membership

In [182…
```
dict1
```

Out[182…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [184…
```
'Name' in dict1 # keys only
```

Out[184…
```
True
```

In [186…
```
'Chinni' in dict1
```

Out[186…
```
False
```

In [188…
```
'Ramu' in dict1
```

Out[188…
```
False
```

In [190…
```
'analyst' in dict1
```

Out[190…
```
False
```

In [192…
```
'Job' in dict1
```

Out[192…
```
True
```

All/Any

In [ ]:
```
# the all method returns:
#True-if all the 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
```

In [195…
```
dict1
```

Out[195…
```
{'Name': 'Ramu',
 'ID': 23030939,
 'DOB': 2003,
 'Job': 'analyst',
 'Address': 'hyderabad'}
```

In [197…
```
all(dict1)
```

Out[197…
```
True
```

In [ ]:
```

```