SET - DICT

February 3, 2025

1 SET

[3]:
```
s = {1,2,3,4,5,5,4,2,'Chandu',1+2j}
type(s)
```

[3]:
```
set
```

[4]:
```
len(s)
```

[4]:
```
7
```

[5]:
```
s   #Duplicate items are not allowed in set
```

[5]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu'}
```

[6]:
```
s1 = set() #to create an empty set
```

[7]:
```
type(s1)
```

[7]:
```
set
```

[10]:
```
for i in enumerate(s) : #loop through set
    print(i)
```

```
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 'Chandu')
(6, (1+2j))
```

[11]:
```
s
```

[11]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu'}
```

1.1 Set membership

[12]:
```
1 in s
```

[12]: True

[14]:
```
'Chandu' in s
```

[14]: True

[16]:
```
55 in s
```

[16]: False

1.2 Add or remove items in set

[19]:
```
s
```

[19]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu'}

[21]:
```
s.add('one') # add items in set
s
```

[21]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one'}

[26]:
```
s.update(['ten','nine','eleven']) # to addd multiple items in set
s
```

[26]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'eleven', 'nine', 'one', 'ten'}

[27]:
```
s.remove('nine') #remove an item in a set
s
```

[27]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'eleven', 'one', 'ten'}

[30]:
```
s.discard('eleven') # remove an item using set and this will not through error␣
 ↪if item is not present in sets
s
```

[30]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one', 'ten'}

[31]:
```
s1 =s.copy()
```

[32]:
```
s1
```

[32]: {(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one', 'ten'}

[33]:
```
s.clear() # this will delete all items in the set and set will be an empty set
s
```

[33]:
```
set()
```

[34]:
```
del s # this will delete the set
s
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[34], line 2
      1 del s # this will delete the set
----> 2 s

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

1.3 Copy set

[35]:
```
s = s1.copy()
```

[36]:
```
s
```

[36]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one', 'ten'}
```

[39]:
```
id(s),id(s1) # the address of the set will be diffrent
```

[39]:
```
(2240441121952, 2240441120608)
```

[ ]:
```

```

[37]:
```
s2 = s1 # create a new refrence set
```

[38]:
```
id(s2),id(s1) # the address of both set will be the same as it is a refrence set
```

[38]:
```
(2240441120608, 2240441120608)
```

[40]:
```
s2
```

[40]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one', 'ten'}
```

[43]:
```
s1.add('nine') # this will take affect on both set s1 and s2 because it is␣
 ↪pointing on same location set but not on s
s2
```

[43]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'nine', 'one', 'ten'}
```

[45]:
```
s # it wont imact due to changes made in original set
```

[45]:
```
{(1+2j), 1, 2, 3, 4, 5, 'Chandu', 'one', 'ten'}
```

1.4 Set Operations

1.4.1 Union

[50]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

[52]:
```
A.union(B) # union joins the all elements of A & B
```

[52]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[53]:
```
A | B # union symbolic gesture
```

[53]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[54]:
```
A.union(B,C)
```

[54]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[55]:
```
A|B|C
```

[55]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[56]:
```
A.update(B,C) # this will update the set A with items present in B & C
A
```

[56]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

1.4.2 Intersection

[58]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[60]:
```
A.intersection(B) # it returns the common items in A & B
```

[60]:
```
{4, 5}
```

[61]:
```
A & B # symbolic Representaion
```

[61]:
```
{4, 5}
```

[63]:
```
A.intersection_update(B) # This will update the items of A with common value of␣
↪A & B
A
```

[63]:
```
{4, 5}
```

1.4.3 Diffrenece

[69]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[72]:
```
A - B # return items that are only present in A not in B
```

[72]:
```
{1, 2, 3}
```

[74]:
```
A.difference(B) # returns items that are not in B but only in A
```

[74]:
```
{1, 2, 3}
```

[75]:
```
B - A # returns the item that are only present in B not in A
```

[75]:
```
{6, 7, 8}
```

[76]:
```
B.difference(A) # returns the item that are only present in B not in A
```

[76]:
```
{6, 7, 8}
```

[77]:
```
B.difference_update(A) # this will update B with diffrence of B & A
B
```

[77]:
```
{6, 7, 8}
```

[78]:
```
### Symmetric  Diffrence
```

[84]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[85]:
```
A.symmetric_difference(B)  # Set of elements A and B not in Both or Removes␣
↪common element
```

[85]:
```
{1, 2, 3, 6, 7, 8}
```

[86]:
```
A ^ B
```

[86]:
```
{1, 2, 3, 6, 7, 8}
```

[87]:
```
A.symmetric_difference_update(B) #this will update A with symmetric diffrence␣
↪of A & B
A
```

[87]:
```
{1, 2, 3, 6, 7, 8}
```

[88]:
```
### Subset , Superset & Disjoint
```

[89]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

[90]:
```
B.issubset(A) # B will be subset of A if all items in B are present in A
```

[90]:
```
True
```

[91]:
```
A.issuperset(B) # A will be the superset of B if all items B are present in A
```

[91]:
```
True
```

[92]:
```
C.isdisjoint(A) # two sets are said to be disjoint if they dont have common␣
↪elements
```

[92]:
```
True
```

[93]:
```
B.isdisjoint(A)
```

[93]:
```
False
```

[94]:
```
### Other Built in Function
```

[95]:
```
A
```

[95]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

[96]:
```
sum(A)
```

[96]:
```
45
```

[97]:
```
max(A)
```

[97]:
```
9
```

[98]:
```
min(A)
```

[98]:
```
1
```

[99]:
```
len(A)
```

[99]:
```
9
```

[100]:
```
list(enumerate(A))
```

[100]:
```
[(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]
```

[102]:
```
D = sorted(A,reverse =True)
D
```

[102]:
```
[9, 8, 7, 6, 5, 4, 3, 2, 1]
```

[103]:
```
sorted(D)
```

[103]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2 Dictionary

[3]:
```
d = {} # create empty dict
```

[4]:
```
type(d)
```

[4]:
```
dict
```

[6]:
```
d = dict() # Another way to create dictionary
```

[7]:
```
type(d)
```

[7]:
```
dict
```

[8]:
```
d = {1:'one' , 2:'two' , 3:'three'} #integer key dictionary
```

[9]:
```
d = dict({1:'one' , 2:'two' , 3:'three'}) # Create dictionary using dict()
```

[10]:
```
d  = {'A':'one' , 'B':'two' , 'C':'three'} # dict with charcter keys
```

[12]:
```
d =  {1:'one' , 'A':'two' , 3:'three'} # dict with mixed keys
```

[13]:
```
d.keys()  # return key names only
```

[13]:
```
dict_keys([1, 'A', 3])
```

[14]:
```
d.values() # return values only
```

[14]:
```
dict_values(['one', 'two', 'three'])
```

[15]:
```
d.items() # return all key and value pairs
```

[15]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

[16]:
```
d = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria']} # dictionary with mix␣
↪data types
```

[18]:
```
d = {1: 'one',2: 'two','A': ['asif', 'john', 'Maria'],'B': ('Bat', 'cat',␣
↪'hat')}
```

[19]:
```
d = {1: 'one',2: 'two','A': {'Name': 'asif', 'Age': 20},'B': ('Bat', 'cat',␣
↪'hat')}
```

```
#dictionary within dictionary
```

[23]:
```
keys = {'a','b','c','d'}
print(type(keys))
d1 = dict.fromkeys(keys) #create dictionary from a set of keys
d1
```

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

[23]:
```
{'b': None, 'a': None, 'c': None, 'd': None}
```

[39]:
```
keys = {'a','b','c','d'}
value = 11
d3 = dict.fromkeys(keys,value)
d3
```

[39]:
```
{'b': 11, 'a': 11, 'c': 11, 'd': 11}
```

[40]:
```
keys = {'a','b','c','d'}
value = [10,20,30] #
d3 = dict.fromkeys(keys,value) # create dictionary with SET of keys and LIST of␣
 ↪value
d3
```

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

[41]:
```
value.append(40)
# this will append 40 in  value and value is added to dictionary so it will␣
 ↪upate the dictionary also
print(value)
d3
```

```
[10, 20, 30, 40]
```

[41]:
```
{'b': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'c': [10, 20, 30, 40],
 'd': [10, 20, 30, 40]}
```

2.1 Accesing items

[42]:
```
d = {1:'one' , 2:'two' , 3:'three' , 4:'four'}
d
```

[42]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

[44]:
```
d[1] #Access item using Key
```

[44]:
```
'one'
```

[45]:
```
d.get(1) # access item usig gate() method
```

[45]:
```
'one'
```

[46]:
```
d = {'Name':'Asif' , 'ID': 74123 , 'DOB': 1991 , 'job' :'Analyst'}
d
```

[46]:
```
{'Name': 'Asif', 'ID': 74123, 'DOB': 1991, 'job': 'Analyst'}
```

[48]:
```
d['Name'] # acces dict using key
```

[48]:
```
'Asif'
```

[49]:
```
d.get('ID')
```

[49]:
```
74123
```

[50]:
```
### ADD , REMOVE & Chanege items
```

[52]:
```
d = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
d
```

[52]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

[53]:
```
d['DOB']
```

[53]:
```
1991
```

[54]:
```
d['DOB'] = 1992 # Changing dictionary item ir assining new value
d
```

[54]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1992, 'Address': 'Hilsinki'}
```

[56]:
```
dict1 = {'DOB':1995}
d.update(dict1)
d
```

[56]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Hilsinki'}
```

[58]:
```
d['JOB'] = 'Data Scientist' # adding new item to a dictionary
d
```

[58]:
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1995,
 'Address': 'Hilsinki',
 'JOB': 'Data Scientist'}
```

[60]:
```
d.pop('JOB') # removing key and item both
```

[60]:
```
'Data Scientist'
```

[61]:
```
d
```

[61]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Hilsinki'}
```

[62]:
```
d.popitem() # removes random item
d
```

[62]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995}
```

[64]:
```
del[d['ID']] # removing item using del method
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[64], line 1
----> 1 del[d['ID']] # removing item using del method
      2 d

KeyError: 'ID'
```

[65]:
```
d
```

[65]:
```
{'Name': 'Asif', 'DOB': 1995}
```

[66]:
```
d.clear() # this will clear dictionary items
```

[67]:
```
d
```

[67]:
```
{}
```

[68]:
```
del d # this will delete the dictionary
```

[69]:
```
d
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[69], line 1
----> 1 d

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

[71]:
```
d = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
```

2.2 Copy

[72]:
```
d2 = d.copy() # by this method address will be diffrent for both
```

[73]:
```
d3 = d2 # address will be thw same and changes made on d2 will reflect on d3
↪and vice versa
```

[74]:
```
d3['job'] = 'Analyst'
d3
```

[74]:
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1991,
 'Address': 'Hilsinki',
 'job': 'Analyst'}
```

[75]:
```
d2
```

[75]:
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1991,
 'Address': 'Hilsinki',
 'job': 'Analyst'}
```

[76]:
```
d
```

[76]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

[ ]: