Combinations of dict with set, list and tuple:

In [3]:
```
With respect to dict:
        1.      dict in dict - Possible
        2.      set in dict - Possible (as values; not as keys unless frozen)
        3.      list in dict - Possible (as values; not as keys)
        4.      tuple in dict - Possible (only hashable tuples can be used as keys)

_____

With respect to set:
        1.      set in set - Not possible (TypeError: unhashable type)
        2.      list in set - Not possible (TypeError: unhashable type)
        3.      tuple in set - Possible (if elements of the tuple are hashable)
        4.      dict in set - Not possible (TypeError: unhashable type)

_____

With respect to list:
        1.      list in list - Possible
        2.      tuple in list - Possible
        3.      set in list - Possible
        4.      dict in list - Possible
```

```
  Cell In[3], line 2
    1.  dict in dict - Possible
                     ^
SyntaxError: invalid character '-' (U+2013)
```

In [15]:
```
d1={"sno":1,"name":"Mahaboob","Y1_Marks":{"EOM":60,"HI":68,"DBMS":67},"Uname":"MRII
for m,k in d1.items():
    print(m,"--->", k,"--->",type(k),"--->",type(d1))
```

```
sno ---> 1 ---> <class 'int'> ---> <class 'dict'>
name ---> Mahaboob ---> <class 'str'> ---> <class 'dict'>
Y1_Marks ---> {'EOM': 60, 'HI': 68, 'DBMS': 67} ---> <class 'dict'> ---> <class 'dic
t'>
Uname ---> MRIIRS ---> <class 'str'> ---> <class 'dict'>
```

In [17]:
```
d1["sno"]
```

Out[17]:
```
1
```

In [19]:
```
d1["name"]
```

Out[19]:
```
'Mahaboob'
```

In [21]:
```
d1["Y1_Marks"]
```

Out[21]:
```
{'EOM': 60, 'HI': 68, 'DBMS': 67}
```

In [23]:
```
d1["Uname"]
```

Out[23]: 'MRIIRS'

In [25]:
```
d1["Y1_Marks"]
for p,k in d1["Y1_Marks"].items():
    print(p,"--->",k)
```

```
EOM ---> 60
HI ---> 68
DBMS ---> 67
```

In [27]:
```
d1["Y1_Marks"]
```

Out[27]: {'EOM': 60, 'HI': 68, 'DBMS': 67}

In [35]:
```
d1["Y1_Marks"]["DBMS"]=98 #Updates DBMS marks to 98
d1["Y1_Marks"]
```

Out[35]: {'EOM': 60, 'HI': 68, 'DBMS': 98}

In [37]:
```
d1={"sno":1,"name":"Mahaboob","Y1_Marks":{"EOM":60,"HI":68,"DBMS":67},"Uname":"MRII
d1["s1"]={"clab":77,"pce":70}
print(d1)
```

```
{'sno': 1, 'name': 'Mahaboob', 'Y1_Marks': {'EOM': 60, 'HI': 68, 'DBMS': 67}, 'Unam
e': 'MRIIRS', 's1': {'clab': 77, 'pce': 70}}
```

In [39]:
```
for m,k in d1.items():
    print(m,"--->", k,"--->",type(k),"--->",type(d1))
```

```
sno ---> 1 ---> <class 'int'> ---> <class 'dict'>
name ---> Mahaboob ---> <class 'str'> ---> <class 'dict'>
Y1_Marks ---> {'EOM': 60, 'HI': 68, 'DBMS': 67} ---> <class 'dict'> ---> <class 'dic
t'>
Uname ---> MRIIRS ---> <class 'str'> ---> <class 'dict'>
s1 ---> {'clab': 77, 'pce': 70} ---> <class 'dict'> ---> <class 'dict'>
```

In [43]:
```
d1["s1"].popitem() #removing one item dict from inner dictinary
```

Out[43]: ('pce', 70)

In [53]:
```
d1.pop("Y1_Marks") # removing complete one dictinary
print(d1,type(d1),id(d1))
```

```
{'sno': 1, 'name': 'Mahaboob', 'Uname': 'MRIIRS'} <class 'dict'> 2512007898880
```

In [9]:
```
s1={10,"Mahaboob",{"EOM":60,"HI":68,"DBMS":67},"MRIIRS"} # dict in set – NOT POSSIB
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 s1={10,"Mahaboob",{"EOM":60,"HI":68,"DBMS":67},"MRIIRS"}

TypeError: unhashable type: 'dict'
```

In [13]:
```
lst=[10,"Mahaboob",{"EOM":60,"HI":68,"DBMS":67},"MRIIRS"]
print(lst,type(lst),id(lst))
```

```
[10, 'Mahaboob', {'EOM': 60, 'HI': 68, 'DBMS': 67}, 'MRIIRS'] <class 'list'> 2240098
933888
```

In [19]:
```
for val in lst:
    print(val,"--->",type(val),"--->",type(lst))
```

```
10 ---> <class 'int'> ---> <class 'list'>
Mahaboob ---> <class 'str'> ---> <class 'list'>
{'EOM': 60, 'HI': 68, 'DBMS': 67} ---> <class 'dict'> ---> <class 'list'>
MRIIRS ---> <class 'str'> ---> <class 'list'>
```

In [21]:
```
lst[2] # printing dict
```

Out[21]:
```
{'EOM': 60, 'HI': 68, 'DBMS': 67}
```

In [23]:
```
for m,k in lst[2].items(): # In detail printing the dict
    print(m,k)
```

```
EOM 60
HI 68
DBMS 67
```

In [25]:
```
lst[2]["DBMS"]
```

Out[25]:
```
67
```

In [27]:
```
lst[2]["DBMS"]=25
print(lst,type(lst),id(lst))
```

```
[10, 'Mahaboob', {'EOM': 60, 'HI': 68, 'DBMS': 25}, 'MRIIRS'] <class 'list'> 2240098
933888
```

In [35]:
```
lst=[10,"Mahaboob",{"EOM":60,"HI":68,"DBMS":67},"MRIIRS"] # adding one more dict in
lst.append({"Hindi":85,"Telugu":20})
print(lst)
```

```
[10, 'Mahaboob', {'EOM': 60, 'HI': 68, 'DBMS': 67}, 'MRIIRS', {'Hindi': 85, 'Telug
u': 20}]
```

In [37]:
```
lst=[10,"Mahaboob",{"EOM":60,"HI":68,"DBMS":67},"MRIIRS"] # removing of inner dicti
lst.remove({"EOM":60,"HI":68,"DBMS":67})
print(lst,type(lst),id(lst))
```

```
[10, 'Mahaboob', 'MRIIRS'] <class 'list'> 2240098964288
```

In [39]:
```
lst.pop(-1)
```

Out[39]:
```
'MRIIRS'
```

In [6]:
```
d1={10:1.2,20:2.2}
d2={30:1.2,40:2.3}
print(d1,type(d1),id(d1))
```

```
{10: 1.2, 20: 2.2} <class 'dict'> 3005005806080
```

In [8]:
```
print(d2,type(d2),id(d2))
```

```
{30: 1.2, 40: 2.3} <class 'dict'> 3005005794752
```

In [10]:
```
d1.update(d2) # Merging
print(d1,type(d1),id(d1))
```

```
{10: 1.2, 20: 2.2, 30: 1.2, 40: 2.3} <class 'dict'> 3005005806080
```

In [12]:
```
d1={10:1.2,20:2.2}
d2={30:1.2,40:2.3}
print(d1,type(d1),id(d1))
print(d2,type(d2),id(d2))
```

```
{10: 1.2, 20: 2.2} <class 'dict'> 3005029484032
{30: 1.2, 40: 2.3} <class 'dict'> 3005029485632
```

In [16]:
```
for m,k in d2.items(): # Merging (model-2)
    d1[m]=k
print(d1,type(d1),id(d1))
```

```
{10: 1.2, 20: 2.2, 30: 1.2, 40: 2.3} <class 'dict'> 3005029484032
```

In [18]:
```
d1={10:1.2,20:2.2} # using update function
d2={30:1.2,10:12.3}
print(d1,type(d1),id(d1))
print(d2,type(d2),id(d2))
d1.update(d2)
print(d1,type(d1),id(d1))
```

```
{10: 1.2, 20: 2.2} <class 'dict'> 3005060336448
{30: 1.2, 10: 12.3} <class 'dict'> 3005060334976
{10: 12.3, 20: 2.2, 30: 1.2} <class 'dict'> 3005060336448
```

In [22]:
```
l1=[10,20,30]
l2=["Mahaboob","MRIIRS","CDOE"]
z=zip(l1,l2) # data in hexadecimal format
print(z)
```

```
<zip object at 0x000002BBAB913280>
```

In [24]:
```
for x in z:
    print(x)
```

```
(10, 'Mahaboob')
(20, 'MRIIRS')
(30, 'CDOE')
```

In [28]:
```
l1=[10,20,30]
l2=["Mahaboob","MRIIRS","CDOE"]
z=zip(l1,l2)
d=dict(z)
print(d,type(d))
```

```
{10: 'Mahaboob', 20: 'MRIIRS', 30: 'CDOE'} <class 'dict'>
```

In [30]:
```
for m,k in d.items():
    print(m,k)
```

```
10 Mahaboob
20 MRIIRS
30 CDOE
```

In [32]:
```
l1=[10,20,30]
l2=["Mahaboob","MRIIRS","CDOE"]
z=zip(l1,l2[::-1])
d=dict(z)
print(d,type(d))
```

```
{10: 'CDOE', 20: 'MRIIRS', 30: 'Mahaboob'} <class 'dict'>
```

In [34]:
```
# tuple to dict
lst=[(10,"Mahaboob"),(20,"MRIIRS"),(30,"CDOE")]
d=dict(lst)
print(d,type(d))
```

```
{10: 'Mahaboob', 20: 'MRIIRS', 30: 'CDOE'} <class 'dict'>
```

In [38]:
```
d1={10: 'CDOE', 20: 'MRIIRS', 30: 'Mahaboob'}
for x in d1.keys():
    print(x)
```

```
10
20
30
```

In [40]:
```
for x in d1.values():
    print(x)
```

```
CDOE
MRIIRS
Mahaboob
```

In [42]:
```
# elements are coming in tuple format
for x in d.items():
    print(x)
```

```
(10, 'Mahaboob')
(20, 'MRIIRS')
(30, 'CDOE')
```

In [44]:
```
for x in d1:
    print(x)
```

```
10
20
30
```

In [48]:
```
for x in d1:
    print(x, d1.get(x))
```

```
10 CDOE
20 MRIIRS
30 Mahaboob
```

In [50]:
```
for x in d1:
    print(x, d1[x])
```

```
10 CDOE
20 MRIIRS
30 Mahaboob
```

In [ ]:

Non Type:

In [ ]:
```
--> NoneType is pre-defined class
--> None is the keyword treated as a value Nonetype is data type
--> None is not False, Space, Zero and None is nothing
--> We can't creat an object of None type because it contain single value
```

In [53]:
```
None==None
```

Out[53]: True

In [57]:
```
None==True
```

Out[57]: False

In [59]:
```
None==False
```

Out[59]: False

In [63]:
```
None==""
```

Out[63]: False

In [85]:
```
print(list().clear())
```

```
None
```

In [87]:
```
print(set().clear())
```

```
None
```

In [89]:
```
print(dict().clear())
```

```
None
```

In [65]:
```
a=None
print(a,type(a),id(a))
```

```
None <class 'NoneType'> 140725369124816
```

In [79]:
```
import keyword as khan
khan.kwlist
```

Out[79]:
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [81]:
```
x=NoneType()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[81], line 1
----> 1 x=NoneType()

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

In [ ]:
```

```