Dictionaries and Sets ¶

- Dictionary
    - which is used to store collection of data
    - Which represented by {}
    - {key1:value1, key2:value2----}
    - keys are unique idenfiers for values

In [ ]:

```
l = [1,2,'a']
t = (1,5,'t',6.54)
```

In [1]:

```
## empty dictionary

d = {}
type(d)
```

Out[1]:

```
dict
```

In [2]:

```
## dictionary
d = {1:'One',2:'Two',3:'Three'}
d
```

Out[2]:

```
{1: 'One', 2: 'Two', 3: 'Three'}
```

In [4]:

```
d[2]
```

Out[4]:

```
'Two'
```

In [5]:

```
d['Three']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-5-5c0b21b239f7> in <module>
----> 1 d['Three']

KeyError: 'Three'
```

In [6]:

```
d1 = dict(course='Python',branch ='ECE')
d1
```

Out[6]:

```
{'course': 'Python', 'branch': 'ECE'}
```

In [7]:

```
d1['course']
```

Out[7]:

```
'Python'
```

In [8]:
```
d1['Python']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-8-194260b5d27b> in <module>
----> 1 d1['Python']

KeyError: 'Python'
```

In [9]:
```
d[3] = 'Four'
```

In [10]:
```
d
```

Out[10]:
```
{1: 'One', 2: 'Two', 3: 'Four'}
```

In [11]:
```
print(dir(dict),end=' ')
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt_
_', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len_
_', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__reversed__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subcl
asshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popite
m', 'setdefault', 'update', 'values']
```

In [12]:
```
### fromkeys
# create a new dictionary with keys from sequence
# Syntax : fromkeys(seq,value)

keys = ('key1','key2','key3')
dic = dict.fromkeys(keys)
dicr = dict.fromkeys(keys,20)
print(dic)
print(dicr)
```

```
{'key1': None, 'key2': None, 'key3': None}
{'key1': 20, 'key2': 20, 'key3': 20}
```

In [13]:
```
dicr['key1']
```

Out[13]:
```
20
```

In [14]:
```
dicr.get('key1')
```

Out[14]:
```
20
```

In [15]:
```
dicr.keys()
```

Out[15]:
```
dict_keys(['key1', 'key2', 'key3'])
```

In [16]:
```
dicr.values()
```

Out[16]:
```
dict_values([20, 20, 20])
```

In [17]:
```
dicr.items()
```

Out[17]:
```
dict_items([('key1', 20), ('key2', 20), ('key3', 20)])
```

In [18]:
```
r = 'string'
for i in r:
    print(i)
```

```
s
t
r
i
n
g
```

In [19]:
```
d = {'Course':'Python','branch':'ECE','Org':'APSSDC'}
for i in d:
    print(i)
```

```
Course
branch
Org
```

In [20]:
```
for i in d.values():
    print(i)
```

```
Python
ECE
APSSDC
```

In [21]:
```
for i in d.keys():
    print(i)
```

```
Course
branch
Org
```

In [22]:
```
for i in d.items():
    print(i)
```

```
('Course', 'Python')
('branch', 'ECE')
('Org', 'APSSDC')
```

In [24]:

```
for key,value in d.items():
    print(key,value)
```

```
Course Python
branch ECE
Org APSSDC
```

In [27]:

```
## copy

info = d.copy()
print(info)
print(d)
```

```
{'Course': 'Python', 'branch': 'ECE', 'Org': 'APSSDC'}
{'Course': 'Python', 'branch': 'ECE', 'Org': 'APSSDC'}
```

In [32]:

```
d
```

Out[32]:

```
{'Course': 'Python', 'branch': 'ECE', 'Org': 'APSSDC'}
```

In [28]:

```
## popitem() -- which doesn't take any arguments

info.popitem()
```

Out[28]:

```
('Org', 'APSSDC')
```

In [29]:

```
info
```

Out[29]:

```
{'Course': 'Python', 'branch': 'ECE'}
```

In [30]:

```
## pop() takes key

info.pop('Course')
```

Out[30]:

```
'Python'
```

In [31]:

```
info
```

Out[31]:

```
{'branch': 'ECE'}
```

In [33]:

```
## Update -- update specific key value
## add new key value pair
d.update({'branch':'All'})
d
```

Out[33]:

```
{'Course': 'Python', 'branch': 'All', 'Org': 'APSSDC'}
```

In [34]:

```
info
```

Out[34]:

```
{'branch': 'ECE'}
```

In [35]:
```
info.update({'year':2021})
```

In [36]:
```
info
```

Out[36]:
```
{'branch': 'ECE', 'year': 2021}
```

In [37]:
```
## setdefault(key,value)
## if key exists it returns the value
info.setdefault('year')
```

Out[37]:
```
2021
```

In [38]:
```
info.setdefault('course')
```

In [39]:
```
info
```

Out[39]:
```
{'branch': 'ECE', 'year': 2021, 'course': None}
```

In [42]:
```
info.setdefault('org','Apssdc')
```

Out[42]:
```
'Apssdc'
```

In [43]:
```
info
```

Out[43]:
```
{'branch': 'ECE', 'year': 2021, 'course': None, 'org': 'Apssdc'}
```

In [44]:
```
d.clear()
```

In [45]:
```
d
```

Out[45]:
```
{}
```

In [46]:
```
info.clear()
```

In [47]:
```
info
```

Out[47]:
```
{}
```

In [48]:
```
del info
```

In [49]:

```
info
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-49-886ead46232a> in <module>
----> 1 info

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

In [52]:

```
## Contact Application

## create contact
## update contact
## search contact
## delete contact

contacts = {}
def create_contact(name,num):
    if name in contacts:
        print(name,'Contact already exists....')
    else:
        contacts[name] = num
        print('contact',name,'is added....')

create_contact('Nandini',348576256)
```

```
contact Nandini is added....
```

In [53]:

```
contacts
```

Out[53]:

```
{'Nandini': 348576256}
```

In [54]:

```
create_contact('Suresh',45567367)
```

```
contact Suresh is added....
```

In [55]:

```
create_contact('Nandini',2345346958)
```

```
Nandini Contact already exists....
```

Set

- Set is also a one of the data structures
- A set is a collection which is unordered and unindexed
- which is collection of elements/data
- Immutable
    - we can't change the values once we assigned
- Advantage: set eliminate the duplicate data elements

- represented by {}

In [56]:
```
s = {23,12,78,'abc',34.78}
s
```

Out[56]:
```
{12, 23, 34.78, 78, 'abc'}
```

In [62]:
```
## predefined functions

s = {23,12,78,56,34}
print(min(s))
print(max(s))
print(sum(s))
print(sorted(s))
print(len(s))
```

```
12
78
203
[12, 23, 34, 56, 78]
5
```

In [63]:
```
print(dir(set),end=' ')
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__
iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '_
_ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand
__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor_
_', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__x
or__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'i
ntersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'p
op', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
'update']
```

In [64]:
```
s.add(67)
```

In [65]:
```
s
```

Out[65]:
```
{12, 23, 34, 56, 67, 78}
```

In [66]:
```
s.clear()
```

In [67]:
```
s
```

Out[67]:
```
set()
```

In [68]:
```
s1 = {1,2,3,4}
s2 = s1.copy()
```

In [69]:
```
s2
```

Out[69]:
```
{1, 2, 3, 4}
```

In [70]:
```
s2.add('abc')
```

In [71]:
```
s2
```

Out[71]:
```
{1, 2, 3, 4, 'abc'}
```

In [72]:
```
s2.discard(4)
```

In [73]:
```
s2
```

Out[73]:
```
{1, 2, 3, 'abc'}
```

- difference
  - removes the common elements in two sets
  - it returns the unique elements in the first set
  - Syntax:
    - set1.difference(set2)

In [74]:
```
s2
```

Out[74]:
```
{1, 2, 3, 'abc'}
```

In [75]:
```
s2 = {1, 2, 3, 'abc',2,3,6,7}
```

In [76]:
```
s2
```

Out[76]:
```
{1, 2, 3, 6, 7, 'abc'}
```

In [77]:
```
s1 = {10,1,5,3,9}
s2 = {1,10,4,7}
s3 = s1.difference(s2)
s4 = s2.difference(s1)
print(s3)
print(s4)
print(s1)
print(s2)
```

```
{9, 3, 5}
{4, 7}
{1, 3, 5, 9, 10}
{1, 10, 4, 7}
```

- difference_update

    - returns the new updated set after making the difference between two sets
    - change the original set
    - Syntax:
        - set1.difference_update(set2)

```
In [80]: s3 = {1,2,3,4,5}
         s4 = {1,2,78,45,32}
         s3.difference_update(s4)
         print(s3)

         {3, 4, 5}
```

```
In [81]: s4

Out[81]: {1, 2, 32, 45, 78}
```

```
In [82]: s3

Out[82]: {3, 4, 5}
```

```
In [83]: d = {23,56,12,11}
         e = {67,22,66,16,11}
         d.union(e)

Out[83]: {11, 12, 16, 22, 23, 56, 66, 67}
```

```
In [85]: d.pop()

Out[85]: 56
```

```
In [86]: d

Out[86]: {11, 12, 23}
```

```
In [87]: d.discard(12)
```

```
In [88]: d

Out[88]: {11, 23}
```

- intersection
    - return the common elements in sets
- intersection_update
    - removes the elements in one set which are not present in other set
    - returns updated set

In [3]:

```
a = {12,84,67,12,67,89}
b = {12,56,78,89,33,77,67}
res = a.intersection(b)
print(res)
```

```
{89, 67, 12}
```

In [4]:

```
print(a)
print(b)
```

```
{89, 67, 12, 84}
{33, 67, 12, 77, 78, 56, 89}
```

In [5]:

```
s = {2,4,6,8,10}
s_1 = {1,3,5,7,9,2}
s.intersection_update(s_1)
print(s)
```

```
{2}
```

In [6]:

```
print(s)
```

```
{2}
```

In [7]:

```
s
```

Out[7]:

```
{2}
```

In [8]:

```
s_1
```

Out[8]:

```
{1, 2, 3, 5, 7, 9}
```

In [9]:

```
s = {2,4,6,8,10}
s_1.intersection_update(s)
```

In [10]:

```
s_1
```

Out[10]:

```
{2}
```

- isdisjoint
    - returns True if no items in set1 matches set2
    - else return False
- issubset()
    - returns True if all elements in set1 mathces set2
    - else return False
- issuperset()
    - returns True if all items in set1 matches set2
    - else return False

In [11]:

```
a = {1,2,3,4,5}
b = {6,7,8,9}
c = {1,2,3,4,5}
r1 = a.isdisjoint(b)
r2 = a.isdisjoint(c)
print(r1)
print(r2)
```

```
True
False
```

In [13]:

```
a = {1,2,3,4,5,7}
b = {1,2,3,4}
r3 = a.issubset(b)
r4 = b.issubset(a)
print(r3)
print(r4)
```

```
False
True
```

In [15]:

```
r5 = a.issuperset(b)
r6 = b.issuperset(a)
print(r5)
print(r6)
```

```
True
False
```

- symmetric difference
  - returns set which contain elments which are not same
  - syntax: set1.symmetric difference(set2)
- symmetric_difference_update
  - removes the common elements in both sets
  - inserts the items which are not present in both sets
  - changes the original

In [18]:

```
## common elements
a = {1,2,3,4,5,7}
c = {1,2,3,4}
d = a.symmetric_difference(c)
print(d)
print(a)
```

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

In [20]:

```
## non common elements

a = {1,2,3,'a','b'}
b = {5,6,'y','t'}
f = a.symmetric_difference(b)
print(f)
print(a)
```

```
{1, 2, 3, 'y', 5, 6, 't', 'b', 'a'}
{1, 2, 3, 'b', 'a'}
```

In [31]:

```
a = {1,2,3,4,5,7}
c = {1,2,3,4}
a.symmetric_difference_update(c)
a
```

Out[31]:

```
{5, 7}
```

In [26]:

```
print(c)
```

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

In [27]:

```
a = {1,2,3,4,5,7}
c.symmetric_difference_update(a)
```

In [28]:

```
print(c)
```

```
{5, 7}
```

In [ ]:

```
```