In [1]:
```
d={"K1":100, "k2":23.33, "k3":True, "k4":"Mr.Raju"}
```

In [4]:
```
len(d)
print(type(d))
help({}) # help(d) #help(dict)
```

```
<class 'dict'>
```

In [7]:
```
#How to fetch single data from dict?

# listname[index]----> emt

print(d['K1'])
print(d['k3'])
print(d['K4'])
```

```
100
True
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-7-bbae27994b0f> in <module>
      5 print(d['K1'])
      6 print(d['k3'])
----> 7 print(d['K4'])

KeyError: 'K4'
```

In [11]:
```
d["k5"]="10.20.30.40" # dictname["newkey"]=value
print(d)
d["k2"]="DATA" #dictname["oldkey"]=updated_value
print(d)
```

```
{'K1': 100, 'k2': 'DATA', 'k3': True, 'k4': 'Mr.Raju', 'k5': '10.20.30.40'}
{'K1': 100, 'k2': 'DATA', 'k3': True, 'k4': 'Mr.Raju', 'k5': '10.20.30.40'}
```

In [12]:
```
L=[1,2,3,4,5]
for var in L:
    print(var)
```

```
1
2
3
4
5
```

In [13]:
```
for var in d: # iterating over the list of keys
    print(var) # displays only keys
```

```
K1
k2
k3
k4
k5
```

In [17]:
```
for var in d:
    print(var,"/",d[var])
```

```
K1 / 100
k2 / DATA
k3 / True
k4 / Mr.Raju
k5 / 10.20.30.40
```

In [16]:
```
for var in d:
    print("{}/{}".format(var,d[var]))
```

```
K1/100
k2/DATA
k3/True
k4/Mr.Raju
k5/10.20.30.40
```

In [18]:
```
d.pop("k5")
```

Out[18]:
```
'10.20.30.40'
```

In [19]:
```
print(d)
```

```
{'K1': 100, 'k2': 'DATA', 'k3': True, 'k4': 'Mr.Raju'}
```

In [20]:
```
d.setdefault("k6","Oracle") # dictname.setdefault("key",value)
#- checks for key.inserts (k,v) if absent
```

Out[20]:
```
'Oracle'
```

In [21]:
```
print(d)
```

```
{'K1': 100, 'k2': 'DATA', 'k3': True, 'k4': 'Mr.Raju', 'k6': 'Oracle'}
```

In [22]:
```
d.setdefault("K1",3000)
```

Out[22]:
```
100
```

In [23]:
```
print(d)
```

```
{'K1': 100, 'k2': 'DATA', 'k3': True, 'k4': 'Mr.Raju', 'k6': 'Oracle'}
```

In [26]:
```
print(d.items())
print(d.keys())
print(d.values())
```

```
dict_items([('K1', 100), ('k2', 'DATA'), ('k3', True), ('k4', 'Mr.Raju'), ('k6', 'Oracl
e')])
dict_keys(['K1', 'k2', 'k3', 'k4', 'k6'])
dict_values([100, 'DATA', True, 'Mr.Raju', 'Oracle'])
```

In [27]:
```
for var in d.items():
    print(var)
```

```
('K1', 100)
('k2', 'DATA')
('k3', True)
('k4', 'Mr.Raju')
('k6', 'Oracle')
```

In [29]:
```
print(d.get("K1"))
print(d['K1'])
```

```
100
100
```

In [30]:
```
emp={"eid":[123,456,222,444], "ename":["Mr.Raj","Mr.Kamal","Ms.Puja"],

"edept":["Sales","HR","Prod"]}
```

In [34]:
```
print(type(emp))
```

```
<class 'dict'>
```

In [33]:
```
print(type(emp['edept']))
```

```
<class 'list'>
```

In [38]:
```
#dictname["existing_key"]---> value
emp["eid"]
#L.append("value")
emp["eid"].append(55555)
print(emp)
emp["eid"].insert(2,101)
print(emp)
```

```
{'eid': [123, 456, 222, 444, 55555, 55555, 55555], 'ename': ['Mr.Raj', 'Mr.Kamal', 'Ms.P
uja'], 'edept': ['Sales', 'HR', 'Prod']}
{'eid': [123, 456, 101, 222, 444, 55555, 55555, 55555], 'ename': ['Mr.Raj', 'Mr.Kamal',
'Ms.Puja'], 'edept': ['Sales', 'HR', 'Prod']}
```

In [41]:
```
emp={"eid":(123,456,222,111,123,444)} #  list & tuple allows duplicates

print(emp["eid"])
print(emp["eid"].count(123))
```

```
(123, 456, 222, 111, 123, 444)
2
```

In [43]:
```
d={"class":{"k1":"v1", "k2":"v2", "k3":"v3"}, "id":{"k1":100,"k2":200}}
```

In [44]:
```
d['class']
```

Out[44]:
```
{'k1': 'v1', 'k2': 'v2', 'k3': 'v3'}
```

In [45]:
```
d['class']['k1'] #dict['key1']['key2']
```

Out[45]:
```
'v1'
```

In [46]:
```
d['id']
```

Out[46]:
```
{'k1': 100, 'k2': 200}
```

In [47]:
```
d['id']['k2']
```

Out[47]:
```
200
```

In [49]:
```
emp={"ename":"Ms.Swedha", "dept":"Sales","eid":101, "ecost":589222.333}

print(emp["ename"]) # key based access
print(emp.get("ename")) # dictname.get("key")---> "value"
```

```
Ms.Swedha
Ms.Swedha
```

In [51]:
```
print(emp["ENAME"]) # key based access
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
```

```
<ipython-input-51-2b1dc88259d9> in <module>
----> 1 print(emp["ENAME"]) # key based access

KeyError: 'ENAME'
```

In [52]:
```
print(emp.get("ENAME")) # dictname.get("key")---> "value"
```

```
None
```

In [55]:
```
#to add new data to dict

#dictname['newkey']=value
emp['ename']="Bangalore"
emp
```

Out[55]:
```
{'ename': 'Bangalore',
 'dept': 'Sales',
 'eid': 101,
 'ecost': 589222.333,
 'ecity': 'Bangalore'}
```

In [58]:
```
#dictname.setdefault('newkey','newvalue')
emp.setdefault("ecountry","India")
print(emp)
emp.setdefault("dept","NODATA")
print(emp)
```

```
{'ename': 'Bangalore', 'dept': 'Sales', 'eid': 101, 'ecost': 589222.333, 'ecity': 'Banga
lore', 'ecountry': 'India'}
{'ename': 'Bangalore', 'dept': 'Sales', 'eid': 101, 'ecost': 589222.333, 'ecity': 'Banga
lore', 'ecountry': 'India'}
```

In [ ]:
```
#To get list of keys
#key based way
for var in d:
    print(var)
```

In [ ]:
```
# method based
d.key()
```

In [60]:
```
s={"k1","k2",10,"k3"}
print(type(s))
```

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

In [63]:
```
s={} # empty dict
print(type(s))
```

```
<class 'dict'>
```

In [64]:
```
s=set() # empty set
print(type(s))
```

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

In [66]:
```
s={"k1","k2",10,20,"k3"}
len(s)
"k1" in s # membership opr
```

Out[66]:
```
True
```

In [69]:
```
#add new data to existing set
s.add(1000) # add single data
print(s)
s.add(1000)
s.add(1000)
s.add(1000)
print(s)# set doesnot allow duplicates
```

```
{1000, 'k2', 10, 'k1', 'k3', 20}
{1000, 'k2', 10, 'k1', 'k3', 20}
```

In [70]:
```
L1=["d1","d2","d3","d4"]

s.update(L1) # adding a collection to set
print(s)
```

```
{'d4', 'd3', 1000, 'k2', 10, 'k1', 'd1', 'k3', 20, 'd2'}
```

In [73]:
```
L1=["d1","d2","d3","d4"]
L2=["d6","d7","d2","d3","d4"]
L3=L1+L2 # has duplicates
```

In [80]:
```
set(L3)
```

Out[80]:
```
{'d1', 'd2', 'd3', 'd4', 'd6'}
```

In [81]:
```
s={"d1","d2","d3","d4"}
s.remove("d4")
print(s)
```

```
{'d3', 'd2', 'd1'}
```

In [82]:
```
s.discard("d2")
print(s)
```

```
{'d3', 'd1'}
```

In [83]:
```
s.remove("d9")# d9 not available in set
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-83-85fcb8799954> in <module>
----> 1 s.remove("d9")# d9 not available in set

KeyError: 'd9'
```

In [85]:
```
s.discard("d9")
print(s)
```

```
{'d3', 'd1'}
```

In [86]:
```
s1={"a","b","c","d"}
s2={"c","d","e","f","g"}
```

In [89]:
```
#union
#opr
print(s1|s2)
#method
print(s1.union(s2))
print(s2.union(s1))
```

```
{'d', 'g', 'b', 'e', 'f', 'c', 'a'}
{'d', 'g', 'b', 'e', 'f', 'c', 'a'}
{'d', 'g', 'b', 'e', 'f', 'c', 'a'}
```

In [90]:
```python
#to get redundant data
print(s1&s2)
print(s1.intersection(s2))
print(s2.intersection(s1))
```

```
{'d', 'c'}
{'d', 'c'}
{'d', 'c'}
```

In [93]:
```python
#Activity

s1={"p1.c","p2.c","p3.java","Demo"}
s2={"p1.java","p1.c","p2.c","p3.java","Demo","D1"}
# Filter common file
#intersection
print(s1&s2)
# combine all files without duplicates and typecast to list
print(list(s1.union(s2)))
```

```
{'p2.c', 'p1.c', 'Demo', 'p3.java'}
['p2.c', 'p3.java', 'p1.c', 'D1', 'Demo', 'p1.java']
```

In [96]:
```python
#set difference
s1={"p1.c","p2.c","p3.java","Demo"}
s2={"p1.java","p1.c","p2.c","p3.java","Demo","D1"}
# s1 only files
print(s1-s2)# returns empty set. bcaz all data available in s2
print(s2-s1) # s2 only files
print(s2.difference(s1)) # s2 only files
```

```
set()
{'D1', 'p1.java'}
{'D1', 'p1.java'}
```

In [99]:
```python
s1={1,2,3,4,5}
s2={4,5,6,7,8}
#unique only from A and B
#symmetric diffrence-> without common data in A & B
print(s1^s2)
print(s1.symmetric_difference(s2))
```

```
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
```

In [100…
```python
#task
#file content to dict
d={}
fobj=open("D:\\property.txt")
for var in fobj:
    var.strip() # remove \n
    L=var.split("=")
    k,v= L # multiple  initialization
    d[k]=v # adding data to dict

for v in d:
    print("{}:{}".format(v,d[v]))
```

```
interface:eth0
```

```
bootproto:static

Ip:10.20.30.40

DNS2:122.333.444.555
```

In [101…

```python
#task
#file content to dict
d={}
fobj=open("D:\\property.txt")
for var in fobj.readlines():# List of file lines
    var.strip() # remove \n
    L=var.split("=")
    k,v= L # multiple  initialization  # combine line8 & 9 as d[L[0]]=L[1]
    d[k]=v # adding data to dict

for v in d:
    print("{}:{}".format(v,d[v]))
```

```
interface:eth0

bootproto:static

Ip:10.20.30.40

DNS2:122.333.444.555
```

In [108…

```python
a=10
x,y=10,20
print(x)
print(y)
l=[10,20]
x,y=l
print(x,y)
```

```
10
20
10 20
```

In [113…

```python
#task
#file content to dict

fobj=open("D:\\property.txt")
print(fobj.readline())  # returns single str
print(fobj.readline())
print(fobj.readline())
print(fobj.readline())

print(fobj.readline())
```

```
interface=eth0

bootproto=static

Ip=10.20.30.40

DNS2=122.333.444.555
```

In [116…

```python
# updated task
#task
#file content to dict
d={}
```

```
fobj=open("D:\\property.txt")
for var in fobj:
    var.strip() # remove \n
    L=var.split("=")
    k,v= L # multiple  initialization
    d[k]=v # adding data to dict

for v in d:
    print("{}:{}".format(v,d[v]))


d["interface"]="eth1" # dict modification
d["bootproto"]="None" #dict modification
d["onboot"]="yes" # adding new entry to dict

print("Updated dict details")
for v in d:
    print("{}:{}".format(v,d[v]))

with open("D:\\newproperty.txt","w") as wobj:
    for v in d:
        wobj.write("{}={}\n".format(v,d[v]))
```

```
interface:eth0

bootproto:static

Ip:10.20.30.40

DNS2:122.333.444.555
Updated dict details
interface:eth1
bootproto:None
Ip:10.20.30.40

DNS2:122.333.444.555
onboot:yes
```