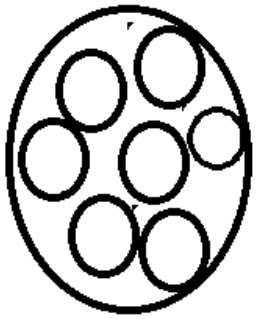
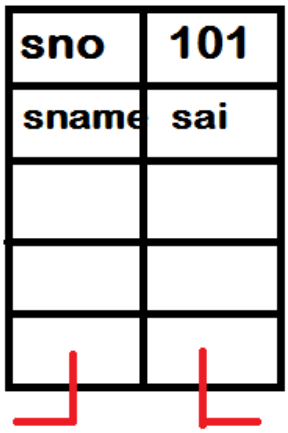
Dict manipulations



Other Collection

List | Set | tuple | frozenset str

VS



key : value
----------> item

dt = { item,item,item }

Key
duplicate(x) immutable objects only

value
duplicate (valid) values can be mutable or immutable

- Predefined class for dictionary <class 'dict'>
- In the dict collection values are organized in the form of key and value pairs
- Key and value should be separated by using :
- Each key and value is called an item
- Each item should be separated by using ,
- All item must be taken in between { }
- Key must be immutable object, values can of mutable or immutable
- Keys should be duplicate but values can be duplicate.
- Dict collection is mutable collection

- Insertion order is not maintain
- Dict items are manipulated by using keys but not using index | slicing

Creating dict Collection:

```
d1={}
print("Type is : ",type(d1))
print("Data is : ",d1)

#dict() -> dict object
d2=dict()
print("type is : ",type(d2))
print("Data is : ",d2)

d3={"sno":101,"sname":"ravi"}
print("Type is : ",type(d3))
print("Data is : ",d3)

#dict(iterable) -> dict | iterable is dict object
d4=dict(d3)
print("Type is : ",type(d4))
print("Data is : ",d4)
```

Reading Data From Dict Collection:

```
stu={"sno":101,"sname":"ravi","sage":34}
print("Data is : ",stu)

#D.keys() -> <class 'dict_keys'> | iterable
ky=stu.keys()
print("Type is : ",type(ky))
print("Keys : ",ky)  #ky=([sno,sname,sage])

import time

for k in ky:
    time.sleep(1)
    print(k)
print("-"*30)

#D.values() -> <class 'dict_values'>  | iterable
vs=stu.values()
print("Type is : ",type(vs))
print("values are : ",vs)

for v in vs:
    time.sleep(.1)
    print(v)
```

Reading Items From Dict:

```
stu={"sno":101,"sname":"ravi","sage":34}
print("Data is : ",stu)

import time
for item in stu.items():
    time.sleep(.2)
    print(item)

print("-"*30)

for item in stu.items():
    time.sleep(1)
    k,d=item
    print(k,d,sep='---->')

print("-"*30)
for k,d in stu.items():
    time.sleep(1)
    print(k,d,sep='---->')
```

```
stu={"sno":101,"sname":"ramesh","scity":"hyd"}
print("Type is : ",type(stu) )
print("Student : ",stu)

#Adding an item to dict collection
stu['spin']=500090
print("Student:")
print(stu)

#Reading Value from dict collection
name=stu['sname']
print("Sname is : ",name)

#a=stu['sage'] KeyError
#Update the value in the dict
stu['sno']=121
print("Student ")
print(stu)

#deleting an item from dict
del stu['sno']
print("Student ",stu)
```

```
#D.keys( ) ->  <class 'dict_keys'>   | iterable
stu={"sno":101,"sname":"ramesh","scity":"hyd"}
print(stu)

keys=stu.keys()
print("Keys : ",keys)

print(stu['sno'])  #101
print(stu['sname']) #ramesh
print(stu['scity']) #hyd

print("- "*30)
import time
for k in keys:    #keys([sno,sname,scity])
   time.sleep(1)
   print(k,"<<>>",stu[k])
```

```
#D.values( ) → <class 'dict_values'>  | iterable
stu={"sno":101,"sname":"ramesh","scity":"hyd"}
print(stu)

#D.values( ) -> <class 'dict_values'>
values=stu.values()
print("type is : ",type(values))
print("Values are : ",values) #values([101,ramesh,hyd])

#D.items() →  <class 'dict_items'>  | Iterable
stu={"sno":101,"sname":"ramesh","scity":"hyd"}
print(stu)

items=stu.items()
print("Type is : ",type(items)) #<class 'dict_items'>
print("items : ",items)
#dict_items[(sno,101),(sname,ranesh),(scity,hyd)]

import time
for item in items:
    time.sleep(1)
    k,d=item  #tuple unpacking
    print(k,"<<>>",d)
```

```
#App-2
print("*"*30)
for k,d in stu.items():
    time.sleep(1)
    print(k,"<<>>",d)
```

D.update({k:d}) -> dict

It is used to update an existed dict collection with new dict

```
stu={"sno":101}
print(stu)

#Adding an Item
#App-1
stu['sname']='shashi'
print(stu)

#App-2
#D.update({k:d})
stu.update({"scity":"hyd"})
print(stu)
```

```
#App-3
k=input("Enter Key : ")
d=input("Enter Value : ")
stu.update({k:d})
print(stu)
```

get( )

D.get(key[,d]) -> value

Case 1: if the specified key is existed then it will return the value associated with specified key

Case 2: if the specified key is not existed then it will return None,when default [d] value is not given

Case 3: if the specified key is not existed then it will return [d] if [d] is given

```
stu={"sno":101}
print(stu)

#D.get(k[,d]) -> value | None
#case 1:
value=stu.get("sno")
```

```
print("value : ",value)

#case 2:
v=stu.get('sname')
print("Result is : ",v)

#case 3:
v=stu.get('scity','hyd')
print("Result is : ",v)

setdefault( )
stu={"sno":101}
print(stu)

#D.setdefault(k[,d])
#default value for d is None
'''
case 1: if the specified key is existed then it will return
the values associated with given key '''
v=stu.setdefault('sno')
print("Result is : ",v)

'''
```

```
case 2: if the specified key is not existed then it will
create new item with k-key and with value is None when
[d] is not given '''
stu.setdefault('sname')
print(stu)

'''
case 3: if the specified key is not existed then it will create
new item with k-key and with value is [d] when [d] is given
'''
stu.setdefault('scity','hyd')
print(stu)

fromkeys()
D.fromkeys(iterable[,value=None])
lst=['ramesh','sudha','madhu']

#D.fromkeys(iterable[,value=None]) -> dict
#case 1: it will create dict collection by taking each item
#from iterable collection as  keys with values None
#when value is not given

stu={}
stu1=stu.fromkeys(lst)
```

```
print(stu)
print(stu1)

#Case 2: it will create new dict collection by taking each
#item from iterable collection as keys with the value [value]
stu2=stu.fromkeys(lst,'Python')
print(stu2)

ZIP()
lst_n=['ramesh','sudha','madhu']
lst_c=['Python','Go-lan','DS']

#zip(iterable,iterable) -> zip object  | iterable
z=zip(lst_n,lst_c)
print("type is : ",type(z))

#dict(iterable) -> dict
stu=dict(z)
print(stu)
```

Can U Converting String to DICT:

```
#String to dict Demo
s="ramesh=34,sudha=28,radha=23"
print(s)

#Step-1 split each item at ,
lst=s.split(',')   #[ramesh=34,sudha=28,radha=23]
print(lst)

#Step-2
stu={}
import time
for item in lst:
    time.sleep(.2)
    k,d=item.split('=')
    stu.update({k:int(d)})

print(stu)
```

Copy( )

- It used to create shallow copy of dict collection
  - D.copy() -> dict object

```
stu={"sno":101,"sname":"sudha"}
print("stu : ",stu)

stu1=stu.copy()
print("stu1 :",stu1)

stu['sname']='chinni'
print("stu : ",stu)
print("stu1 : ",stu1)

#D.pop(k[,d]) -> item
stu={"sno":101,"sname":"sudha"}
print("stu : ",stu)

#D.pop(k[,d]) ->value | KeyError
#case 1: if the sepcified k is existed then it will remove
#the item assoicated with k and it will return the value
associate with k
value=stu.pop("sno")
print("Value is : ",value)
print("stu : ",stu)

#Case 2: if the sepcified k is not existed then it will return
#KeyError
```

```
#value=stu.pop('scity')
#print("value ",value)

#Case 3: if the specified k is not existed then it will return
#[d] if d is given otherwise it will raise "KeyError"
value=stu.pop("spin",500090)
print("Value is : ",value)
```

D.popitem( ) -> item

```
stu={"sno":101,"sname":"sudha","scit":"hyd"}
print("stu : ",stu)

#it will remove and return any random item from dict
collection upto python 3.7 in python 3.8 it will work LIFO
t=stu.popitem()
print("Deleted item is : ",t)
```

D.clear()

```
> it will erase all items from dict collection
stu={"sno":101,"sname":"sudha","scit":"hyd"}
print("stu : ",stu)

stu.clear()
print("stu : ",stu)
```