Session - 10

clear() ⇒ removes everything in a dictionary. inplace operation

Ex ⇒
```
test = {1:1,2:4,3:9}
test.clear()
Print(test)
```
O/P ⇒ { } None

delete - del ⇒ removes an element from a dictionary. inplace operation

Ex ⇒
```
test={"a":[1,2,3,4],"b":[4,5,6,7]}
del test['a']
Print(test)
```
O/P ⇒ {'b':[4,5,6,7]}

zip() ⇒ method to access elements from two dicts or lists at the same time.

```
a={1:1,2:4,3:9}
for x,y in zip(a.keys(),a.values()):
    print(x,y)
```
O/P ⇒
1 1
2 4
3 9

```
a={'name':['neha','minnu','milky'],
   'id':[1,2,3]}
for k,v in zip(a.keys(),a.values()):
    print(k,v)
```
O/P ⇒ name ['neha','minnu','milky']
id [1,2,3]

copy()

```
d1={1:1,2:4,3:9}
d2=d1.copy()
print(d1,d2)
```
O/P ⇒ {1:1,2:4,3:9} {1:1,2:4,3:9}

```
d1={1:1,2:4,3:9}
d2=d1.copy() // deep copy
d2[4]=16
print(d1,d2)
```
O/P ⇒ {1:1,2:4,3:9}
{1:1,2:4,3:4,4:16}

```
d1={1:1,2:4,3:9}
d1=d2 (or) d2=d1 // shallow copy
d1[4]=16
print("d1:",d1)
print("d2:",d2)
```
O/P ⇒ d1:{1:1,2:4,3:9,4:16}
d2:{1:1,2:4,3:9,4:16}

Sets data type -

- represented as { }
- set is defined in { }
- All the elements that are present inside a set are unique.

```
a={1,1,2,3,3,4,5,5,5}
print(a)
```
O/P ⇒ {1,2,3,4,5}

- All elements inside a set should be of immutable datatype.

a={1,2,[3,4,5],(6,7,8)} ⇒ Error ✗

a={1,2,(3,4,5),(6,7,8)} ⇒ ✓

- Set is a mutable datatype. It can be performed add, update and delete.
- Unordered data type ⇒ It doesn't support indexing.

- slicing is also not supported in a set.

Union and Intersection

```
a = {1,2,3,4,5}
b = {2,3,6,7,8}
aUb = {1,2,3,4,5,6,7,8}  // union
anb = {2,3}              // Intersection
a difference b = a - anb
               = {1,4,5}
b difference a = b - anb
               = {6,7,8}
a symmetric difference b = {1,4,5,6,7,8}
a<b = False              // Subset
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s3 = s1.union(s2)
print(s3) Print("s3:",s3)
```

olp ⇒ s3: {70,40,10,80,50,20,60,30} (unordered)

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s3 = s1.intersection(s2)
print(s3) print("s3:",s3)
```

olp ⇒ s3: {40,50}

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s3 = s1.difference(s2)
print(s3) Print("s3:",s3)
```

olp ⇒ s3: {10,20,30}

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s3 = s2.difference(s1)
print(s3) Print("s3:",s3)
```

olp ⇒ s3: {80,60,70}

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s3 = s1.symmetric_difference(s2)
print(s3)
```

olp ⇒ {80,20,70,10,60,30}

Update()-

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s1.update(s2)
print(s1)
```

olp ⇒ {70,40,10,80,50,20,60,30}

intersection_update()-

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s1.intersection_update(s2)
print(s1)
```

olp ⇒ {40,50}

difference_update()-

```
s1 = {10,20,30,40,50}
s2 = {40,50,60,70,80}
s1.difference_update(s2)
print(s1)
```

olp ⇒ {10,20,30}

Deleting the elements-

- pop
- discard
- remove
- del
- clear.

- pop()

```
a = {1,2,3,4,5,6}
r = a.pop()
print(a,r)
```

compiler deletes an element itself

olp ⇒ {2,3,4,5,6} 1

- remove()

```
a = {10,20,30,40,50}
r = a.remove(30)
print(a,r)
```

r = a.remove(300) ⇒ KeyError

olp ⇒ {50,20,40,10} None

- discard()

```
a = {5,6,7,8,9}
r = a.discard(a)
print(a,r)
```

r = a.discard(300) ⇒ {5,6,7,8,9} None

olp ⇒ {5,6,7,8} None