SETS

Introduction: : A set is a collection of unique elements. A common use is to eliminate duplicate elements from a list. In addition, it supports set operations like union intersection and difference**.** A set is created by builtin function 'set', enclosed between braces and each element separated by comma

E.g.:
```
set_a = set((1,2,3,4,5,3,2))
print(set_a) # {1, 2, 3, 4, 5}
```

Note: Set contains unique elements only even though we put duplicates into it.

Accessing values in sets: To access values from the sets we must convert into a list and can access using index numbers. We can access an list by using index number, where index nukber starts with 0 and ends with length of the list-1.

E.g.:
```
set_a = set((1,2,3,4,5,3,2))
print(set_a) # {1, 2, 3, 4, 5}
a=list(s)
a[0] #1
```

Working with dictionaries: in sets we can perform uninion, intersection operations.

Union: A union of two or more sets is the set of all elements present in the set.

E.g.:
```
s=set((1,2,3,4,5,3,2))    # {1, 2, 3, 4, 5}
t=set((11,21,31,41,51,31,21))    # {11, 21, 31, 41, 51}
u=set((10,20,30,40,50,30,20))    # {10, 20, 30, 40, 50}
s.union(t) # {1, 2, 3, 4, 5, 11, 21, 31, 41, 51}
s.union(u)# {1, 2, 3, 4, 5, 10, 20, 30, 40, 50}
t.union(u)# {10, 11, 20, 21, 30, 31, 40, 41, 50, 51}
s.union(t,u)# {1, 2, 3, 4, 5, 10, 11, 20, 21, 30, 31, 40, 41, 50, 51}
```

Intersection: A intersection of two or more sets of all the common elements present in the sets.

E.g.:
```
s=set((1,2,3,4,5,3,2))    # {1, 2, 3, 4, 5}
t=set((11,21,31,41,51,31,21,1,2))    # {11, 21, 31, 41, 51,1,2}
u=set((10,20,30,40,50,30,20,1,2,11,21))    # {10, 20, 30, 40, 50,1,2,11,21}
intersection(t) # {1, 2}
s. intersection (u)# {1, 2}
t. intersection (u)# {1,2,11,21}
s. intersection (t,u)# {1,2,11,21}
```

Functions and methods: : Python contains functions and methods to perform operations on dictionaries.

Add: Add adds the element to the end of the set and returns the entire set.

Syntax: set.add(element)

E.g:

```
color=set(("blue","red"))
color.add("yellow")
color # {'blue', 'red', 'yellow'}
```

Clear: This clear methods clears the list, and returns the empty set.

Syntax: set.clear()

E.g:

```
color=set(("blue","red",”yellow”))
print(color) # {'blue', 'red', 'yellow'}
color.clear()
print(color) #set()
```

Copy: Copy methods copies the entire set and returns the set.

Syntax: set.copy()

E.g:

```
color=set(("blue","red",”yellow”))
print(color)= # {'blue', 'red', 'yellow'}
color.copy() #{'blue', 'red', 'yellow'}
```

Difference: Difference finds the difference between two sets and returns the values.

Syntax: set_a. difference(set.b)

E.g:

```
color=set(("blue","red",”yellow”,”pink”))
flag= set(("blue","red”,1,2,3))
color.difference(flag) #{“yellow”,”pink”}
```

Discard: This method discards the given element from the set if present, if the element is not present it returns the entire set.

Syntax: set.discard(a)

E.g:

```
color=set(("blue","red",”yellow”,”pink”))
color.discard(“blue”) #
color #{“red”,“yellow”,”pink”}
```

Remove: This method removes the given element from the set if present, if the element is not present it returns key value error.

Syntax: set.remove(a)

E.g:

```
color=set(("blue","red",”yellow”,”pink”))
color.remove(“blue”) #
color ##{“red”,“yellow”,”pink”}
```

Pop: Pop removes the last element from the entire set, and returns an error if the set is empty.

Syntax: set.pop()

E.g:

```
color=set(("blue","red",”yellow”,”pink”))
color.pop()
color ##{“ blue","red",”yellow”}
```

Issubset: This methods check weather one set is subset of another given set .If true it returns the True value and if it is not subset it returns the false value.

Syntax: set_1.issubset(set_2)

E.g:

```
s=set((1,2,3,4,5,3,2))    # {1, 2, 3, 4, 5}
t=set((1,2))    # {1,2}
s.issubset(t) #False
t.issubset(s) # True
```

Issuperset: This methods check weather one set is superset of another given set .If true it returns the True value and if it is not superset it returns the false value.

Syntax: set_1.issuperset(set_2)

E.g:

```
s=set((1,2,3,4,5,3,2))    # {1, 2, 3, 4, 5}
t=set((1,2))    # {1,2}
s.issuperset(t) #True
t.issuperset(s) # False
```