SET manipulations

- Predefined class for SET is <class 'set>
- Heterogeneous Objects are allowed [dis-similar values]
- Insertion order is not maintained
- SET Collection is mutable
- Duplicate Objects are NOT allowed
- None type Objects are allowed
- All Object in the SET must be taken in between {}
- Every Object must be separated by using ,

Note:

- SET Collection is mostly meant for to achieve SET operations in math's such as intersection | union | minus …..

Creating The SET Collection Objects:

```
#App-1
s={10,"A",2.2,None,10}
print("Type is : ",type(s))
print("Data is : ",s)
print("-"*30)
```

```
#set( ) -> set object
s2=set()
print("Type is : ",type(s2))
print("Data is : ",s2)

#set(iterable) -> set Object
print("-"*30)
lst=["aaa","bbb","ccc","ddd"]
s3=set(lst)
print("type is : ",type(s3))
print("Data is : ",s3)
```

Reading The Values From SET Collection:

```
s={101,"ramesh","hyd"}

print("SET Collection ",s)



#indexing is not supported

#print("First ",s[0])

#TypeError: 'set' object is not subscriptable
```

#slicing is not supported

```
#print("First 3 Objects : ",s[0:3])

#TypeError: 'set' object is not subscriptable
```

#Unpacking is possibile but we don't know about order

```
a,b,c=s

print(a,b,c,sep='.....')
```

#Reading All The Values From SET Collection

```
import time

for i in s:

    time.sleep(1)

    print(i)
```

Set Methods :

add(item) -> None

- ➢ Used to add an item to an existed set Collection

Syn: S.add(item) -> None

```
s=set( )

print("Type is : ",type(s))

print("Data is : ",s)


#S.add(item)

itm=int(input("Enter an item "))

s.add(itm)

print("After Adding an Item : ",s)
```

2.update(iterable)

> Used to update an existed set with specified iterable

#S.update(iterable) -> None

Here iterable may be str | list | tuple | set | dict…

```
s1={'a','b','c'}

s2={1,2,3}


print("Before update ")

print("s1 : ",s1)

print("s2 : ",s2)


s1.update(s2)  # s1=s1+s2  s1={'a','b','c',1,2,3}

print("After update ")

print("s1 : ",s1)

print("s2 : ",s2)
```

3.copy()

> It is used to create shallow copy of set collection

Syn: S.copy( ) -> Set Object

```
#S.copy() -> set object

s1={"aaa","bbb","ccc"}

print("Data From s1 : ",s1)


s2=s1.copy()

print("After Copy is : ",s2)


s1.add("ddd")

print("Adding an object to s1 : ",s1)

print("Data From s2",s2)
```

Deleting the Object From SET Collection:

Pop()

- It is used remove and return any random object from set and it will return . PVM raise TypeError while using pop() on an empty set
  - Syn: S.pop() -> item

```
s1={"aaa","bbb","ccc"}

print("Data From s1 : ",s1)

itm=s1.pop()

print("Deleted item is : ",itm)
```

remove(item)

- It is used to remove the specified item from set collection, if the specified item is not existed then it will raise “KeyError”

```
#S.remove() -> None | KeyError

s1={"aaa","bbb","ccc"}

print("Data From s1 : ",s1)

s1.remove("bbb")

print("After Remove : ",s1)
```

discard()

- It will remove the specified item from set collection , if the specified item is not existed then it won’t raise an Error.
    - S.discard(item) ->None

```
s1={"aaa","bbb","ccc"}
print("Data From s1 : ",s1)
s1.discard("ccc")
print("After Remove : ",s1)
```

clear()

- It will erase all objects from the set Collection
    - Syn: s.clear( ) -> None

```
s1={"aaa","bbb","ccc"}
print("Data From s1 : ",s1)
s1.clear()
print("After Remove : ",s1)
```

Set Operations [union | intersection | difference ...]

Union

- It will return set collection by combining two set
    - Syn: S.update(iterable) -> set

```
s1={1,2,3}

s2={3,4,5}

print("S1 : ",s1)

print("S2 : ",s2)
```

```
s3=s1.union(s2)

print("After union ",s3)  #s3={1,2,3,4,5}
```

App: 2

```
s3=s1 | s2    # as same as S.union(iterable)  s3=s1+s2

print("After union ",s3)
```

intersection

- It will return common objects existed in the both Sets
  - Syn: S.intersection(iterable) → set Object

```
s1={1,2,3}
s2={3,4,5}
print("S1 : ",s1)
print("S2 : ",s2)
s3=s1.intersection(s2)
print("Intersection ",s3)
```

App2:

```
s3=s1 & s2
print("Intersection ",s3)
```

difference [minus]

- It will return objects which exited in SET-A and Which are not existed in the SET-B → SET-A.difference(SET-B) [A-B]

```
#S.difference(iterable) -> set object

s1={1,2,3}

s2={3,4,5}

print("S1 : ",s1)

print("S2 : ",s2)

s3=s1.difference(s2)

print("s1-s2 : ",s3)

s4=s2.difference(s1)

print("s2-s1 : ",s4)
```

App 2:

```
s3=s1-s2

print("s1-s2 : ",s3)
```

#S.symmetric_difference(iterable) -> set

- **It will return set Object with all the object which are unique from both sets**

```
s1={1,2,3}

s2={3,4,5}

print("S1 : ",s1)

print("S2 : ",s2)

s3=s1.intersection(s2)

print("Result is : ",s3)

s4=s1.symmetric_difference(s2)

print("Result is : ",s4)

#App-2:

s4=s1^s2

print("Result is : ",s4)
```

#S.issuperset(iterable) -> bool

- It will returns True if S is the super set of iterable collection

```
s1={1,2,3,4,5,6}

s2={3,4,5}

print("S1 : ",s1)

print("S2 : ",s2)

b=s1.issuperset(s2)

print("Is s1 is superset of s2 ? : ",b)
```

#S.issubset(iterable) -> bool

```
s1={1,2,3,4,5,6}

s2={3,4,5}

print("S1 : ",s1)
print("S2 : ",s2)

b=s1.issubset(s2)

print("Is s1 is subset of s2 ? : ",b)

b=s2.issubset(s1)

print("Is s2 is subset of s1 ? : ",b)
```

What is difference between set collection and frozenset

- Set collection is mutable collection where as frozenset collection is immutable collection
- To create set collection we have to use set( ) or set(iterable) lly to create frozenset collection we have to use frozenset( ) or frozenset(iterable)
- We can perform only Math related set operation on frozenset [union | intersection | difference | ….]

```
#frozenset( ) -> frozenset
#frozenset(iterable) -> frozenset

s={1,2,3,4,5}
print("Type is : ",type(s))
print("Data is : ",s)

fs=frozenset(s)
print("Type is : ",type(fs)) #<class 'frozenset'>
print("Data is : ",fs)

print("-"*30)
s.add("aaa")
print("After Adding into SET",s)
#fs.add("ccc")
#AttributeError: 'frozenset' object has no attribute 'add'
```