In [1]:
```
'''
 Set:

      ==> Sets are used to store multiple items in a single variable.

      ==> A set is a collection which is unordered, unchangeable, and unindexed.

      ==> Duplicates Value Not Allowed.

      ==> A set is an unordered collection of items.

          Every set element is unique (no duplicates) and must be immutable (cannot

      ==> set()

      ==> Like :

                      {1,2,4,56,}

'''
""
```

Out[1]:
```
''
```

How To Make A Set

In [1]:
```
# Use : {}
```

In [1]:
```
s1 = {}
```

In [2]:
```
s1
```

Out[2]:
```
{}
```

In [3]:
```
type(s1)
```

Out[3]:
```
dict
```

How To Make A Empty Set

In [4]:
```
## Use : set()
```

In [5]:
```
s1 = set()
print("Empty Set Is :",s1,"\nType Is :",set(s1))
```

```
Empty Set Is : set()
Type Is : set()
```

Unorderd

In [6]:
```
s1 = {10,100,-110,-25,20,25,225}
```

In [7]:
```
s1
```

Out[7]:
```
{-110, -25, 10, 20, 25, 100, 225}
```

In [8]:
```
data = {-10,100,5,55,"A","a"}
```

In [9]:
```
data
```

Out[9]:
```
{-10, 100, 5, 55, 'A', 'a'}
```

In [10]:
```
ord("A")
```

Out[10]:
```
65
```

In [11]:
```
ord("a")
```

Out[11]:
```
97
```

In [12]:
```
chr(12505)
```

Out[12]:
```
'ベ'
```

In [13]:
```
ord("😀")
```

Out[13]:
```
128512
```

In [20]:
```
chr(14500)
```

Out[20]:
```
'栄'
```

Unindexed

In [21]:
```
s1
```

Out[21]:
```
{-110, -25, 10, 20, 25, 100, 225}
```

In [22]:
```
s1[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [22], in <cell line: 1>()
----> 1 s1[0]

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

It Does Not Support Duplicate Values

In [24]:
```
s1 = {"Sean","Mike","Finn","Allen","Sean","Luke"}
```

In [25]:
```
s1
```

Out[25]:
```
{'Allen', 'Finn', 'Luke', 'Mike', 'Sean'}
```

In [26]:
```
s2 = {5,5,5,5,5,5,5,}
```

In [27]:
```
s2
```

Out[27]:
```
{5}
```

Remove The Duplicate Values From A List

In [28]:
```
l1 = ["Sean","Mike","Finn","Allen","Sean","Luke"]
```

In [29]:
```
l1
```

Out[29]:
```
['Sean', 'Mike', 'Finn', 'Allen', 'Sean', 'Luke']
```

In [19]:
```
l5 = [5,5,5,5,5,5,5,]
l5
```

Out[19]:
```
[5, 5, 5, 5, 5, 5, 5]
```

In [30]:
```
inp = input("Do You Want To Remove Duplicate Values :")
if inp=="Yes":
    l1= set(l1)
    print("\nSuccessFully..")
    l1 = list(l1)
    print("\nList Is :",l1)
else:
    print("\nOkay !!")
```

```
Do You Want To Remove Duplicate Values :Yes

SuccessFully..

List Is : ['Finn', 'Luke', 'Sean', 'Mike', 'Allen']
```

In [20]:
```
inp = input("Do You Want To Remove Duplicate Values :")
if inp=="Yes":
    l5 = set(l5)
    print("\nSuccessFully..")
    l5 = list(l5)
    print("\nList Is :",l5)
else:
    print("\nOkay !!")
```

```
Do You Want To Remove Duplicate Values :Yes

SuccessFully..

List Is : [5]
```

How To Make A Set Using Input Function

In [2]:
```
## Syntax : set(input().split())
```

In [31]:
```
s1 = set(input("Enter Data :").split())
print("\nEmp Data Is :",s1)
```

```
Enter Data :Mike Peter Luke Jack Marnus Glenn

Emp Data Is : {'Peter', 'Luke', 'Glenn', 'Jack', 'Mike', 'Marnus'}
```

In [32]:
```
type(s1)
```

Out[32]:
```
set
```

Task : Open A Text File And Remove Duplicate Value

In [33]:
```
## File ==> Open
## Function ==> open(file_name,mode)
## Mode ==> r , w , a , x
## By Default Mode ==> r (read mode)
```

In [34]:
```
file = open("emp.txt")
print(file.read())
```

```
Mike,Luke,I Am Learning Python,Peter,Mike,Jason,Luke
```

In [35]:
```
file = open("emp.txt")
data = file.read()
print(data)
print("\nType Is :",type(data))
```

```
Mike,Luke,I Am Learning Python,Peter,Mike,Jason,Luke

Type Is : <class 'str'>
```

In [38]:
```
data = list(data)
print(data)
```

```
['M', 'i', 'k', 'e', ',', 'L', 'u', 'k', 'e', ',', 'I', ' ', 'A', 'm', ' ', 'L', 'e
', 'a', 'r', 'n', 'i', 'n', 'g', ' ', 'P', 'y', 't', 'h', 'o', 'n', ',', 'P', 'e',
't', 'e', 'r', ',', 'M', 'i', 'k', 'e', ',', 'J', 'a', 's', 'o', 'n', ',', 'L', 'u
', 'k', 'e']
```

In [39]:
```
file = open("emp.txt")
data = file.readlines() ## readlines => Store => List
print(data)
print("\nType Is :",type(data))
```

```
['Mike,Luke,I Am Learning Python,Peter,Mike,Jason,Luke']

Type Is : <class 'list'>
```

In [40]:
```
data = data.split()
data
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [40], in <cell line: 1>()
----> 1 data = data.split()
      2 data

AttributeError: 'list' object has no attribute 'split'
```

In [54]:
```
file = open("emp.txt")
data = file.read()
print(data)
print("\nType Is :",type(data))
```

```
Mike,Luke,I Am Learning Python,Peter,Mike,Jason,Luke

Type Is : <class 'str'>
```

In [55]:
```
data = set(data.split())
print(data)
```

```
{'Learning', 'Mike,Luke,I', 'Am', 'Python,Peter,Mike,Jason,Luke'}
```

In [45]:
```
data = data.split()
data
```

Out[45]:
```
['Mike,Luke,I', 'Am', 'Learning', 'Python,Peter,Mike,Jason,Luke']
```

In [46]:
```
data = set(data)
```

In [47]:
```
data
```

Out[47]:
```
{'Am', 'Learning', 'Mike,Luke,I', 'Python,Peter,Mike,Jason,Luke'}
```

In [49]:
```
l1 = []
file = open("emp.txt")
for k in file.read():
    print(k,end="")
    l1.append(k)
print("\nList Data Is :",l1)
```

```
Mike,Luke,I Am Learning Python,Peter,Mike,Jason,Luke
List Data Is : ['M', 'i', 'k', 'e', ',', 'L', 'u', 'k', 'e', ',', 'I', ' ', 'A', 'm
', ' ', 'L', 'e', 'a', 'r', 'n', 'i', 'n', 'g', ' ', 'P', 'y', 't', 'h', 'o', 'n',
',', 'P', 'e', 't', 'e', 'r', ',', 'M', 'i', 'k', 'e', ',', 'J', 'a', 's', 'o', 'n
', ',', 'L', 'u', 'k', 'e']
```

In [ ]:

Methods of Set

In [56]:
```
dir(set)
```

Out[56]:
```
['__and__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__isub__',
 '__iter__',
 '__ixor__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__rand__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__ror__',
 '__rsub__',
 '__rxor__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__sub__',
 '__subclasshook__',
 '__xor__',
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

In [4]:
```
'''
 'add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update'

'''
""
```

Out[4]:
```
''
```

add()

In [57]:
```
## add() :

     ## It Is Used To Add The Value .

    ## Syntax :

        ###  set_name.add(data)
```

In [59]:
```
s1 = {'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
s1
```

Out[59]:
```
{'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [60]:
```
s1.add("AB")
```

In [61]:
```
s1
```

Out[61]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [62]:
```
s1.add("Jenny","Ria")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [62], in <cell line: 1>()
----> 1 s1.add("Jenny","Ria")

TypeError: set.add() takes exactly one argument (2 given)
```

clear()

In [27]:
```
##### clear() :


   ###### it is used to remove all elements from a set.

    #### create a empty set.

     #### synatx :

            ##### set_name.clear()
```

In [63]:
```
s1
```

Out[63]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [64]:
```
data = s1.copy()
```

In [65]:
```
data
```

Out[65]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [66]:
```
data.clear()
```

In [67]:
```
print("Empty Set Is :",data)
```

```
Empty Set Is : set()
```

In [33]:
```
## del() :

 ## It Is Used To Delete The Object

    ## Syntax :   del(object)

    ## del --> It Is A Keyword.
```

In [68]:
```
data
```

Out[68]:
```
set()
```

In [69]:
```
inp = input("Do You Want To Delete The Data :")
if inp=="Yes":
    del(data)
else:
    print("Exit..")
```

```
Do You Want To Delete The Data :Yes
```

In [70]:
```
data
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [70], in <cell line: 1>()
----> 1 data

NameError: name 'data' is not defined
```

In [75]:
```
a1 = {'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
a1
```

Out[75]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [76]:
```
del(a1[0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [76], in <cell line: 1>()
----> 1 del(a1[0])

TypeError: 'set' object doesn't support item deletion
```

In [78]:
```
lst = list(a1)
lst
```

Out[78]:
```
['Finn', 'Peter', 'Luke', 'Mike', 'Allen', 'AB', 'Jack']
```

In [79]:
```
del(lst[-2])
```

In [80]:
```
a1 = set(lst)
a1
```

Out[80]:
```
{'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

copy()

In [37]:
```
'''
 ## copy() :

     => It Is USed To Copy All Elements of One Set To New Set.

     ### Syntax :  set_name.copy()

'''
""
```

Out[37]:
```
''
```

In [81]:
```
s1 = {'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [82]:
```
s1
```

Out[82]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [83]:
```
back_up = s1.copy()
```

In [84]:
```
back_up
```

Out[84]:
```
{'AB', 'Allen', 'Finn', 'Jack', 'Luke', 'Mike', 'Peter'}
```

In [43]:
```
s1 is back_up
```

Out[43]:
```
False
```

In [88]:
```
a1=int(input("Enter place value : "))
my_list = [1, 2, 3, 4, 5]
print(f"Original list: {my_list}")

# Using del to remove an element
a1=int(input("Enter place value of the value to be deleted : "))
del my_list[a1]
print(f"After using del: {my_list}")

# Using add to add an element
b1=int(input("Enter place value to add : "))
my_list.append(b1)
print(f"After using add: {my_list}")

# Using copy to create a new list
new_list = my_list.copy()
print(f"New list after using copy: {new_list}")
# Using clear to empty the list
my_list.clear()
print(f"After using clear: {my_list}")
```

```
Enter place value : 0
Original list: [1, 2, 3, 4, 5]
Enter place value of the value to be deleted : 2
After using del: [1, 2, 4, 5]
Enter place value to add : 2
After using add: [1, 2, 4, 5, 2]
New list after using copy: [1, 2, 4, 5, 2]
After using clear: []
```

In [90]:
```
s1={'Allen', 'Finn', 'Luke', 'Mike', 'Sean'}
l1=list(s1)
def add_():
    ele=input("enter element to add: ")
    s1.add(ele)
    print(s1)

def copy_():
    backup=s1.copy()
    print("\nbackup data: ",backup)

def clear_():
    s1.clear()
    print("\n Empty set: ",s1)
def del_():
    print("\n Elements: ",l1)
    ind=int(input("enter index value to remove element:"))
    del(l1[ind])
    s1=set(l1)
    print("\n Set after deletion: ",s1)

print("\nOptions:\n1. Add\n2.Copy\n3.Delete\n4.Clear data\n")
ch=int(input("Enter your choice: "))
if ch==1:
      add_()
elif ch==2:
      copy_()
elif ch==3:
      del_()
elif ch==4:
      clear_()
else:
      print("\nInvalid option")
```

```
Options:
1. Add
2.Copy
3.Delete
4.Clear data

Enter your choice: 2

backup data:  {'Finn', 'Mike', 'Luke', 'Sean', 'Allen'}
```

In [91]:
```
l1={"Hello","HYE",1,2,3,40}
print('''
            1)add()
            2)clear()
            3)copy()
            4)del()


      ''')
opt=input("Enter the option:")
if opt=="1":
    data=input("Enter a data:")
    l1.add(data)
    print(l1)
elif opt=="2":
    l1.clear()
    print(l1)
elif opt=="3":
    l2=l1.copy()
    print(l2)
elif opt=="4":
    l2=list(l1)
    index=int(input("Enter the index:"))
    del(l2[index])
    print(l2)
else:
    print("No such option found")
```

```
            1)add()
            2)clear()
            3)copy()
            4)del()


Enter the option:1
Enter a data:Mike Luke
{1, 2, 3, 'Hello', 40, 'Mike Luke', 'HYE'}
```

In [95]:
```
def add_element(s):
    ele = input("Enter element to add: ")
    s.add(ele)
    print("Set after adding:", s)
def copy_set(s):
    backup = s.copy()
    print("Backup data:", backup)
def clear_set(s):
    s.clear()
    print("Empty set:", s)
def delete_element(s):
    l = list(s)
    print("Elements:", l)
    ind = int(input("Enter index value to remove element: "))
    if 0 <= ind < len(l):
        l.pop(ind)
        s = set(l)
        print("Set after deletion:", s)
    else:
        print("Invalid index.")

if __name__ == "__main__":
    s1 = {'Allen', 'Finn', 'Luke', 'Mike', 'Sean'}
    print("Options:")
    print("1. Add")
    print("2. Copy")
    print("3. Delete")
    print("4. Clear data")

    choice = int(input("Enter your choice: "))
    if choice == 1:
        add_element(s1)
    elif choice == 2:
        copy_set(s1)
    elif choice == 3:
        delete_element(s1)
    elif choice == 4:
        clear_set(s1)
    else:
        print("Invalid option")
```

```
Options:
1. Add
2. Copy
3. Delete
4. Clear data
Enter your choice: 2
Backup data: {'Finn', 'Mike', 'Luke', 'Sean', 'Allen'}
```

difference()

In [5]:
```
##### difference() :

   ##### it methods returns a set that contains the difference b/w two sets.

        #### only returned set contains items that exist in the first set , not in


        ##### Syntax :

           ##### set_1.difference(set2,....)
```

In [96]:
```
k = {100,50,200,1000,2000,"Mike"}
a = {100,20,10,5000,500}
```

In [97]:
```
a.difference(k)
```

Out[97]:
```
{10, 20, 500, 5000}
```

In [98]:
```
k.difference(a)
```

Out[98]:
```
{1000, 200, 2000, 50, 'Mike'}
```

In [99]:
```
a
```

Out[99]:
```
{10, 20, 100, 500, 5000}
```

In [100…
```
k
```

Out[100]:
```
{100, 1000, 200, 2000, 50, 'Mike'}
```

In [101…
```
s1 = {100,"Mike",20,"Sean"}
s2 = {200,100,"Mike","Luke"}
s5 = {"Mike",100}
s1.difference(s2,s5)
## s1 <-- s2  :  20 , Sean
## 20 , Sean <-- s5  : 20 , Sean
```

Out[101]:
```
{20, 'Sean'}
```

In [102…
```
s1
```

Out[102]:
```
{100, 20, 'Mike', 'Sean'}
```

In [104…
```
s1 = {100,20,"Mike","Luke"}
s2 = {20,55,"Luke","Peter"}
s3 = {100,"Peter",55,20,"Finn"}
s4 = {56,"Jason"}
s5 = {20,"Nile"}
s5.difference(s1,s3,s2,s4)
## s5 <-- s1 : "Nile"
## "Nile" <-- s3 : "Nile"
## "Nile" <-- s2 : "Nile"
## "Nile" <-- s4 : "Nile"
```

Out[104]:
```
{'Nile'}
```

In [105…
```
s5
```

Out[105]:
```
{20, 'Nile'}
```

difference_update

In [106…
```
## Difference ==> Apply : Update
```

In [112…
```
k
```

Out[112]:
```
{1000, 200, 2000, 50, 'Mike'}
```

In [113…
```
a
```

Out[113]:
```
{10, 20, 100, 500, 5000}
```

In [114…
```
k.difference(a)
```

Out[114]:
```
{1000, 200, 2000, 50, 'Mike'}
```

In [115…
```
k
```

Out[115]:
```
{1000, 200, 2000, 50, 'Mike'}
```

In [116…
```
k.difference_update(a)
```

In [111…
```
k ## {50, 200, 1000, 2000} ==> Update => k
```

Out[111]:
```
{1000, 200, 2000, 50, 'Mike'}
```

In [117…
```
s1.difference(s2,s5)
```

Out[117]:
```
{100, 'Mike'}
```

In [118…
```
s1
```

Out[118]:
```
{100, 20, 'Luke', 'Mike'}
```

In [119…
```
s1.difference_update(s2,s5)
```

In [120…
```
s1
```

Out[120]:
```
{100, 'Mike'}
```

pop()

In [17]:
```
##### pop() :

   #### it is used to remove a random item or element from a set


    #### Syntax :

       #####  set_name.pop()
```

In [121…
```
s1 = {100,"Sean",500,"Mike"}
```

In [122…
```
s1
```

Out[122]:
```
{100, 500, 'Mike', 'Sean'}
```

In [123…
```
s1.pop()
```

Out[123]:
```
'Sean'
```

In [124…
```
s1
```

Out[124]:
```
{100, 500, 'Mike'}
```

In [125…
```
s1.pop()
```

Out[125]:
```
'Mike'
```

In [126…
```
s1
```

Out[126]:
```
{100, 500}
```

In [127…
```
a
```

Out[127]:
```
{10, 20, 100, 500, 5000}
```

In [128…
```
a.pop()
```

Out[128]:
```
100
```

remove()

In [29]:
```
##### remove() :

     #### it is used to remove specified element from  a set.

     ##### if elemnt is not here in set , it will raise an error


        ##### Syntax :

            ##### set_name.remove(elment)
```

In [129…
```
s1 = {100,"Sean",500,"Mike"}
```

In [130…
```
s1
```

Out[130]:
```
{100, 500, 'Mike', 'Sean'}
```

In [131…
```
s1.remove("Mike")
```

In [132…
```
s1
```

Out[132]:
```
{100, 500, 'Sean'}
```

In [134…
```
s1.discard("Sean")
```

Out[134]:
```
{100, 500}
```

In [135…
```
s1
```

Out[135]:
```
{100, 500}
```

In [136…
```
s1.remove("Sean")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [136], in <cell line: 1>()
----> 1 s1.remove("Sean")

KeyError: 'Sean'
```

In [137…
```
s1.discard("Sean")
```

In [36]:
```
s1
```

Out[36]:
```
{500, 'Sean'}
```

In [44]:
```
s1
```

Out[44]:
```
{500, 'Sean'}
```

In [45]:
```
s1.remove("Mike")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [45], in <cell line: 1>()
----> 1 s1.remove("Mike")

KeyError: 'Mike'
```

discard()

In [37]:
```
#### discard():

    #### it is used to remove specified element from  a set.

     ##### if elemnt is not here in set , it will not be raise an error..
```

In [138…
```
s1 = {100,"Sean",500,"Mike"}
```

In [139…
```
s1
```

Out[139]:
```
{100, 500, 'Mike', 'Sean'}
```

In [140…
```
s1.discard("Mike")
```

In [141…
```
s1
```

Out[141]:
```
{100, 500, 'Sean'}
```

In [142…
```
s1.discard(100)
```

In [143…
```
s1
```

Out[143]:
```
{500, 'Sean'}
```

In [46]:
```
s1
```

Out[46]:
```
{500, 'Sean'}
```

In [47]:
```
s1.discard("Mike")
```

Create A Program With Error Or Error Free Option

In [151…

```
data = set(input("Enter The Data :").split())
print("\nData Is :",data)
inp = input("\nDo You Want To Remove Data With Error Or Error Free :")
if inp=="With Error":
    print("\nWith Error..")
    value = input("\nEnter Value To Remove :")
    if value in data:
        data.remove(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print()
        print(value,"Is Already Removed Or Not In Data.")
elif inp=="Error Free":
    value = input("\nEnter Value To Remove :")
    if value in data:
        print("\nThis Method Passes Error.")
        data.discard(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print("\nPass Error..")
else:
    print("\nNo Method Is Here.")
```

```
Enter The Data :Mike Sean Jack Luke Finn Ria Jenny

Data Is : {'Finn', 'Luke', 'Ria', 'Sean', 'Jack', 'Mike', 'Jenny'}

Do You Want To Remove Data With Error Or Error Free :With Error

With Error..

Enter Value To Remove :Jenny

SuccessFully Removed.

Updated Data Is : {'Finn', 'Luke', 'Ria', 'Sean', 'Jack', 'Mike'}
```

In [152…

```
data = set(input("Enter The Data :").split())
print("\nData Is :",data)
inp = input("\nDo You Want To Remove Data With Error Or Error Free :")
if inp=="With Error":
    print("\nWith Error..")
    value = input("\nEnter Value To Remove :")
    if value in data:
        data.remove(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print()
        print(value,"Is Already Removed Or Not In Data.")
elif inp=="Error Free":
    value = input("\nEnter Value To Remove :")
    if value in data:
        print("\nThis Method Passes Error.")
        data.discard(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print("\nPass Error..")
else:
    print("\nNo Method Is Here.")
```

```
Enter The Data :With Error

Data Is : {'With', 'Error'}

Do You Want To Remove Data With Error Or Error Free :With Error

With Error..

Enter Value To Remove :10

10 Is Already Removed Or Not In Data.
```

In [153…
```
data = set(input("Enter The Data :").split())
print("\nData Is :",data)
inp = input("\nDo You Want To Remove Data With Error Or Error Free :")
if inp=="With Error":
    print("\nWith Error..")
    value = input("\nEnter Value To Remove :")
    if value in data:
        data.remove(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print()
        print(value,"Is Already Removed Or Not In Data.")
elif inp=="Error Free":
    value = input("\nEnter Value To Remove :")
    if value in data:
        print("\nThis Method Passes Error.")
        data.discard(value)
        print("\nSuccessFully Removed.")
        print("\nUpdated Data Is :",data)
    else:
        print("\nPass Error..")
else:
    print("\nNo Method Is Here.")
```

```
Enter The Data :Mike Sean Jack Luke Finn Ria Jenny

Data Is : {'Finn', 'Luke', 'Ria', 'Sean', 'Jack', 'Mike', 'Jenny'}

Do You Want To Remove Data With Error Or Error Free :Error Free

Enter Value To Remove :10

Pass Error..
```

intersection()

In [52]:
```
##### intersection() :

   #### it is used to get the common values from one or more than one set.

    #### this method return siminlarity values b/w two or more sets.

    #### Syntax :

        #####  set_name.initersection(set_name1,set_name2.....)
```

In [154…
```
s1 = {100,"Mike",20,"Sean"}
s2 = {200,100,"Mike","Luke",20}
s5 = {"Mike",100}
```

In [155…
```
s1.intersection(s2,s5)
```

Out[155]:
```
{100, 'Mike'}
```

In [156…
```
s1 = {100,"Mike",20,"Sean"}
s2 = {200,100,"Mike","Luke"}
s5 = {"Mike",100}
s6 = {20}
```

In [157…
```
s1.intersection(s2,s5,s6)
```

Out[157]:
```
set()
```

In [158…
```
s1
```

Out[158]:
```
{100, 20, 'Mike', 'Sean'}
```

intersetion_update()

In [58]:
```
#### intersetion_update() :

  #### it is used to get the common values from one or more than one set.

    #### this method return siminlarity values b/w two or more sets.

    #### at the end , update the set with updated common values...its means remove


        ###  that is not present in both sets.



    #### Syntax :

        #####  set_name.initersection(set_name1,set_name2.....)
```

In [159…
```
s1.intersection(s2,s5)
```

Out[159]:
```
{100, 'Mike'}
```

In [160…
```
s1
```

Out[160]:
```
{100, 20, 'Mike', 'Sean'}
```

In [161…
```
s1.intersection_update(s2,s6)
```

In [162…
```
s1
```

Out[162]:
```
set()
```

union()

In [63]:
```
##### union() :

   #### it return a set that contains all items from original set and all items fro


    #### Synatx :

       #####  set_name.union(set_name1,set_name2.....)
```

In [64]:
```
s1 = {100,"Mike",20,"Sean"}
s2 = {200,100,"Mike","Luke"}
s5 = {"Mike",100}
s4 = {"Jenny"}
```

In [163…
```
## Unique => 100,"Mike",20,"Sean",200,Luke,Jenny
```

In [164…
```
print(s1.union(s2,s4,s5))
```

```
{'Mike', 'Luke', 100, 'Jason', 200, 56}
```

issubset()

In [165…
```
s1 = {"Mike","Jackie"}
s2 = {"Mike","Jackie",100}
```

In [166…
```
s1
```

Out[166]:
```
{'Jackie', 'Mike'}
```

In [167…
```
s2
```

Out[167]:
```
{100, 'Jackie', 'Mike'}
```

In [168…
```
s1.issubset(s2)
```

Out[168]:
```
True
```

In [169…
```
s2.issubset(s1)
```

Out[169]:
```
False
```

issuperset()

In [170…
```
s1
```

Out[170]:
```
{'Jackie', 'Mike'}
```

In [171…
```
s2
```

Out[171]:
```
{100, 'Jackie', 'Mike'}
```

In [172…
```
s2.issuperset(s1)
```
Out[172]:
```
True
```

In [173…
```
s1.issuperset(s1)
```
Out[173]:
```
True
```

In [174…
```
A1 = {10,500}
B1 = {10,500}
```

In [175…
```
A1.issubset(B1)
```
Out[175]:
```
True
```

In [176…
```
B1.issubset(A1)
```
Out[176]:
```
True
```

In [177…
```
A1.issuperset(B1)
```
Out[177]:
```
True
```

In [178…
```
B1.issuperset(A1)
```
Out[178]:
```
True
```

isdisjoint()

In [82]:
```
## isdisjoint() :

## if Sets Are Different  --> True
## Single Comman --> False
```

In [179…
```
s1
```
Out[179]:
```
{'Jackie', 'Mike'}
```

In [180…
```
s2
```
Out[180]:
```
{100, 'Jackie', 'Mike'}
```

In [181…
```
s1.isdisjoint(s2)
```
Out[181]:
```
False
```

In [182…
```
a1 = {100,20}
b1 = {"Sean","Mike"}
```

In [87]:
```
a1
```

Out[87]:
```
{20, 100}
```

In [88]:
```
b1
```

Out[88]:
```
{'Mike', 'Sean'}
```

In [183…
```
a1.isdisjoint(b1)
```

Out[183]:
```
True
```

update()

In [184…
```
'''
## update() :

            ==> This Method Update/Add Elements From Another Set Or Other Iterable
'''
""
```

Out[184]:
```
''
```

In [185…
```
s1 = {"Mike","Jack"}
```

In [186…
```
s1
```

Out[186]:
```
{'Jack', 'Mike'}
```

In [187…
```
type(s1)
```

Out[187]:
```
set
```

In [188…
```
lst = ["A","B"]
```

In [6]:
```
lst
```

Out[6]:
```
['A', 'B']
```

In [7]:
```
type(lst)
```

Out[7]:
```
list
```

In [189…
```
s1.update(lst)
```

In [190…
```
s1
```

Out[190]:
```
{'A', 'B', 'Jack', 'Mike'}
```

In [191…
```
s2 = {20,25}
```

In [192…
```
s2
```

Out[192]:
```
{20, 25}
```

In [193…
```
type(s2)
```

Out[193]:
```
set
```

In [194…
```
s2.update(s1)
```

In [195…
```
s2
```

Out[195]:
```
{20, 25, 'A', 'B', 'Jack', 'Mike'}
```

symmetric_difference()

In [15]:
```
'''
 ## symmetric_difference() :

     ==> Returns A Set With Symmetric Difference of Sets.
'''
""
```

Out[15]:
```
''
```

In [196…
```
s1
```

Out[196]:
```
{'A', 'B', 'Jack', 'Mike'}
```

In [197…
```
s2
```

Out[197]:
```
{20, 25, 'A', 'B', 'Jack', 'Mike'}
```

In [198…
```
s1.symmetric_difference(s2)
```

Out[198]:
```
{20, 25}
```

In [199…
```
s2.symmetric_difference(s1)
```

Out[199]:
```
{20, 25}
```

In [200…
```
s5 = {True,False}
```

In [20]:
```
s5
```

Out[20]:
```
{False, True}
```

In [21]:
```
type(s5)
```

Out[21]:
```
set
```

In [201…
```
s1.symmetric_difference(s2,s5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [201], in <cell line: 1>()
----> 1 s1.symmetric_difference(s2,s5)

TypeError: set.symmetric_difference() takes exactly one argument (2 given)
```

In [202…
```
s1
```
Out[202]:
```
{'A', 'B', 'Jack', 'Mike'}
```

Note : Set.Symmetric_Difference() Takes Exactly One Argument

symmetric_difference_update()

In [203…
```
s1
```
Out[203]:
```
{'A', 'B', 'Jack', 'Mike'}
```

In [204…
```
s2
```
Out[204]:
```
{20, 25, 'A', 'B', 'Jack', 'Mike'}
```

In [205…
```
s1.symmetric_difference(s2)
```
Out[205]:
```
{20, 25}
```

In [206…
```
s1
```
Out[206]:
```
{'A', 'B', 'Jack', 'Mike'}
```

In [207…
```
s1.symmetric_difference_update(s2)
```

In [208…
```
s1
```
Out[208]:
```
{20, 25}
```

In [209…
```
s1
```
Out[209]:
```
{20, 25}
```

In [210…
```
s1.pop()
```
Out[210]:
```
20
```

Make A Fucntion of Set of All Methods

Open A Text File And Remove Duplicate Value

In [ ]: