EXP:1

**INPUT::**

**Write python program using ADT concept that implement different ‘set’ operations.**

class Set :

# Creates an empty set instance.

def \_\_init\_\_( self, initElementsCount ):

self.\_s = []

for i in range(initElementsCount) :

e = int(input("Enter Element {}: ".format(i+1)))

self.add(e)

def get\_set(self):

return self.\_s

def \_\_str\_\_(self):

string = "\n{ "

for i in range(len(self.get\_set())):

string = string + str(self.get\_set()[i])

if i != len(self.get\_set())-1:

string = string + " , "

string = string + " }\n"

return string

# Returns the number of items in the set.

def \_\_len\_\_( self ):

return len( self.\_s )

# Determines if an element is in the set.

def \_\_contains\_\_( self, e ):

return e in self.\_s

# Determines if the set is empty.

def isEmpty( self ):

return len(self.\_s) == 0

# Adds a new unique element to the set.

def add( self, e ):

if e not in self :

self.\_s.append( e )

# Removes an e from the set.

def remove( self, e ):

if e in self.get\_set():

self.get\_set().remove(e)

# Determines if this set is equal to setB.

def \_\_eq\_\_( self, setB ):

if len( self ) != len( setB ) :

return False

else :

return self.isSubsetOf( setB )

# Determines if this set is a subset of setB.

def isSubsetOf( self, setB ):

for e in setB.get\_set() :

if e not in self.get\_set() :

return False

return True

# Determines if this set is a proper subset of setB.

def isProperSubset( self, setB ):

if self.isSubsetOf(setB) and not setB.isSubsetOf(self):

return True

return False

# Creates a new set from the union of this set and setB.

def union( self, setB ):

newSet = self

for e in setB :

if e not in self.get\_set() :

newSet.add(e)

return newSet

# Creates a new set from the intersection: self set and setB.

def intersect( self, setB ):

newSet = Set(0)

for i in range(len(self.get\_set())) :

for j in range(len(setB.get\_set())) :

if self.get\_set()[i] == setB.get\_set()[j] :

newSet.add(self.get\_set()[i])

return newSet

# Creates a new set from the difference: self set and setB.

def difference( self, setB ):

newSet = Set(0)

for e in self.get\_set() :

if e not in setB.get\_set():

newSet.add(e)

return newSet

# Creates the iterator for traversing the list of items

def \_\_iter\_\_( self ):

return iter(self.\_s)

def createSet():

n=int(input("Enter number of Elements in set"))

s = Set(n)

return s

choice = 0

print("Create Set A")

s1 = createSet()

print(str(s1))

while choice != 10:

print("|-------------------|")

print("| Menu |")

print("| 1.Add |")

print("| 2.Remove |")

print("| 3.Contains |")

print("| 4.Size |")

print("| 5.Intersection |")

print("| 6.Union |")

print("| 7.Difference |")

print("| 8.Subset |")

print("| 9.Proper Subset |")

print("| 10.Exit |")

print("|-------------------|")

choice = int(input("Enter Choice"))

if choice==1:

e = int(input("Enter Number to Add"))

s1.add(e)

print(str(s1))

elif choice==2:

e = int(input("Enter Number to Remove"))

s1.remove(e)

print(str(s1))

elif choice==3:

e = int(input("Enter Number to Search"))

if e in s1:

print("Number Present in Set")

else:

print("Number is not Present in Set")

print(str(s1))

elif choice==4:

print("Set Contains {} elements".format(len(s1)))

elif choice==5:

print("Create a Set B for doing Intersection Operation")

s2 = createSet()

s3 = s1.intersect(s2)

print("Set A = "+str(s1))

print("Set B = "+str(s2))

print("Intersection = "+str(s3))

elif choice==6:

print("Create a Set B for doing Union Operation")

s2 = createSet()

s3 = s1.union(s2)

print("Set A = "+str(s1))

print("Set B = "+str(s2))

print("Union = "+str(s3))

elif choice==7:

print("Create a Set B for calculating Set Difference")

s2 = createSet()

s3 = s1.difference(s2)

print("Set A = "+str(s1))

print("Set B = "+str(s2))

print("Difference = "+str(s3))

elif choice==8:

print("Create a Set B for checking Subset or not")

s2 = createSet()

isSubset = s1.isSubsetOf(s2)

print("Set A = "+str(s1))

print("Set B = "+str(s2))

if isSubset:

print("Set B is the Subset of Set A")

else:

print("Set B is not a Subset of Set A")

elif choice==9:

print("Create a Set B for checking ProperSubset or not")

s2 = createSet()

isProperSubset = s1.isProperSubset(s2)

print("Set A = "+str(s1))

print("Set B = "+str(s2))

if isProperSubset:

print("Set B is the Proper Subset of Set A")

else:

print("Set B is not a Proper Subset of Set A")

elif choice==10:

break;

elif choice<1 or choice>10:

print("Please Enter Valid Choice")

**OUTPUT::**

Create Set A

Enter number of Elements in set3

Enter Element 1: 55

Enter Element 2: 25

Enter Element 3: 28

{ 55 , 25 , 28 }

|-------------------|

| Menu |

| 1.Add |

| 2.Remove |

| 3.Contains |

| 4.Size |

| 5.Intersection |

| 6.Union |

| 7.Difference |

| 8.Subset |

| 9.Proper Subset |

| 10.Exit |

|-------------------|

Enter Choice1

Enter Number to Add1

{ 55 , 25 , 28 , 1 }

|-------------------|

| Menu |

| 1.Add |

| 2.Remove |

| 3.Contains |

| 4.Size |

| 5.Intersection |

| 6.Union |

| 7.Difference |

| 8.Subset |

| 9.Proper Subset |

| 10.Exit |

|-------------------|

Enter Choice2

Enter Number to Remove3

{ 55 , 25 , 28 , 1 }

|-------------------|

| Menu |

| 1.Add |

| 2.Remove |

| 3.Contains |

| 4.Size |

| 5.Intersection |

| 6.Union |

| 7.Difference |

| 8.Subset |

| 9.Proper Subset |

| 10.Exit |

|-------------------|

Enter Choice4

Set Contains 4 elements

|-------------------|

| Menu |

| 1.Add |

| 2.Remove |

| 3.Contains |

| 4.Size |

| 5.Intersection |

| 6.Union |

| 7.Difference |

| 8.Subset |

| 9.Proper Subset |

| 10.Exit |

|-------------------|