**“DISCRETE MATHAMATICS”**

**PROJECT REPORT**

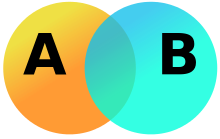
|  |
| --- |
| * **GROUP MEMBERS** **:** * **MHER RAFIQ 12117** * **AAHIL ALWANI 12072** * **PROJECT NAME :** “venn diagram” |

* **FEATURES**

1. It can shade according to user’s given instruction like if user choose to shade **A U B** then it will show the shading of **A U B** in Venn diagram.
2. User can also input any random values & implementation of functions will be apply on the given sets A & B which will be display in output.

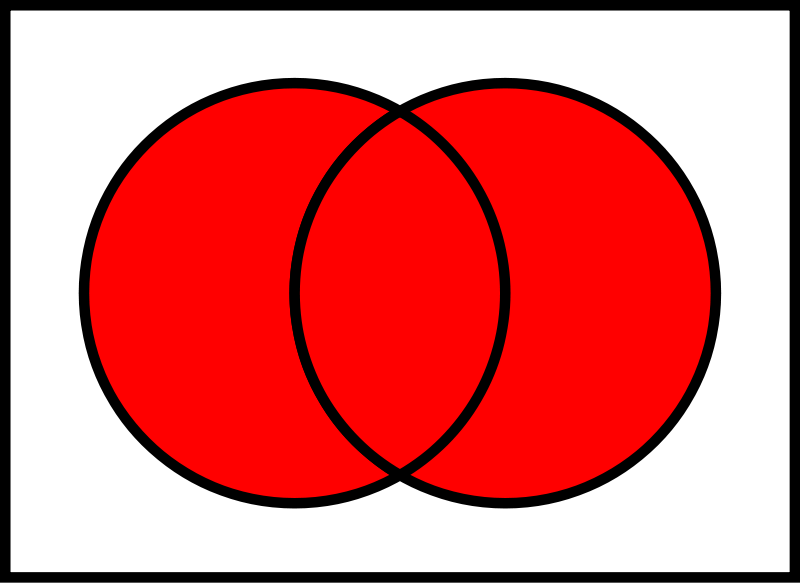
* **INTRODUCTION OF VENN DIAGRAM**

A Venn diagram in math is used in logic theory and set theory to show various sets or data and their relationship with each other. A Venn diagram is read by observing all of the circles that make up the entire diagram. Each circle is its own item or data set. The portions of the circles that overlap indicate the areas that are in common amongst the different items whereas the parts that do not overlap indicate unique traits among the item or data set represented by the circle.



* **IMPLEMENTATION OF FUNCTIONS:**

1. **UNION**



Let A and B be subsets of a universal set U. The union of sets A and B is the set of all elements in U that belong to A or to B or to both, and is denoted A ∪ B.

**Symbolically:**

A ∪ B = {x ∈U | x ∈A or x ∈ B}

**EMAMPLE:**

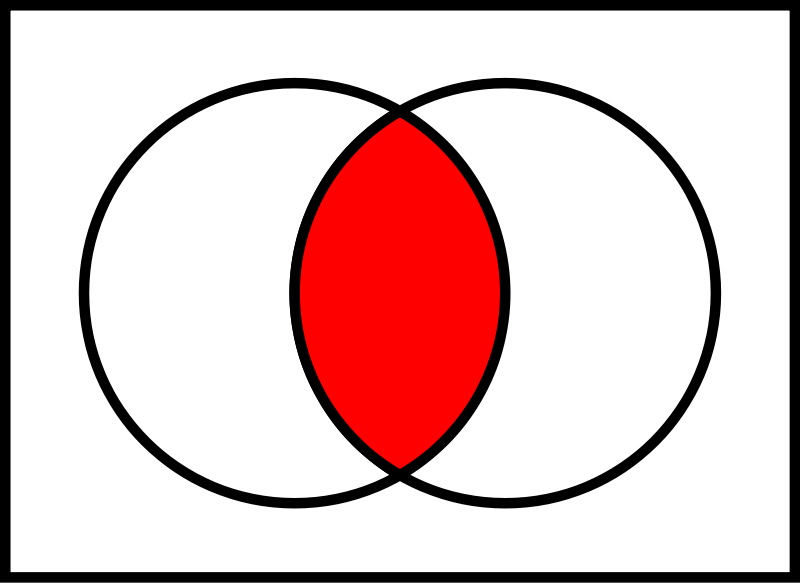
Let U = {a, b, c, d, e, f, g}

A = {a, c, e, g}, B = {d, e, f, g}

Then A ∪ B = {x ∈U | x ∈A or x ∈ B}

= {a, c, d, e, f, g}

1. **INTERSECTION**



Let A and B subsets of a universal set U. The intersection of sets A and B is the set of all elements in U that belong to both A and B and is denoted A ∩ B.

**Symbolically:**

A ∩ B = {x ∈U | x ∈ A and x ∈B

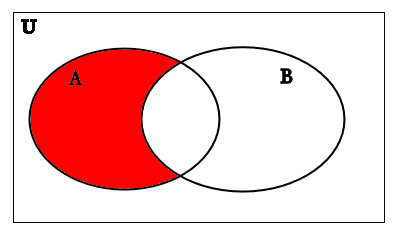
**EXAMPLE:**

Let U = {a, b, c, d, e, f, g}

A = {a, c, e, g}, B = {d, e, f, g}

Then A ∩ B = {e, g}

1. **DIFFERENCE**



Let A and B be subsets of a universal set U. The difference of “A and B” (or relative complement of B in A) is the set of all elements in U that belong to A but not to B, and is denoted A – B or A \ B.

**Symbolically:**

A – B = {x ∈U | x ∈ A and x∉B}

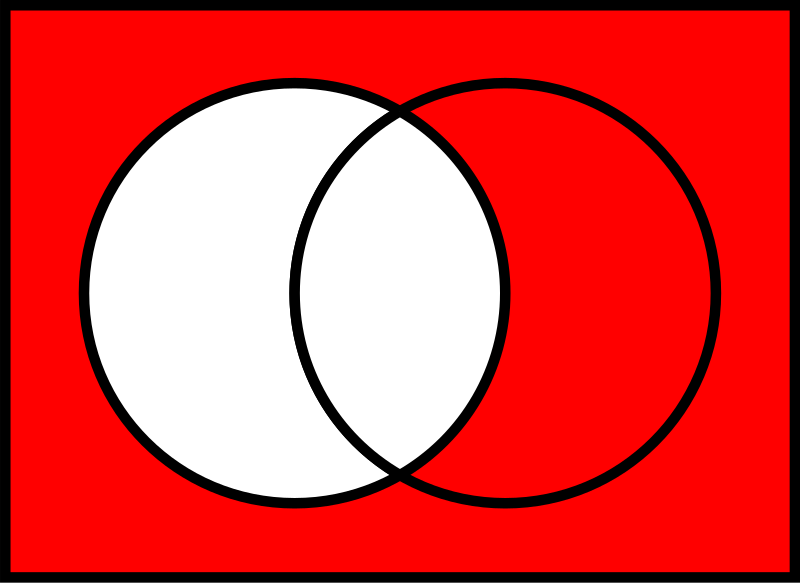
**EXAMPLE:**

Let U = {a, b, c, d, e, f, g}

A = {a, c, e, g}, B = {d, e, f, g}

Then A – B = {a, c}

1. **COMPLEMENT**



Let A be a subset of universal set U. The complement of A is the set of all element in U that do not belong to A, and is denoted AΝ, A or Ac

**Symbolically:**

Ac

= {x ∈U | x ∉A}

**EXAMPLE:**

Let U = {a, b, c, d, e, f, g]

A = {a, c, e, g}

Then Ac

= {b, d, f}

* **IMPLEMENTATION OF FUNCTION THROUGH CODE(PYTHON)**

1. **STARTING CODE (MAIN PART)**

from matplotlib\_venn import venn2, venn2\_circles

from numpy import array

import Element\_Input as el

import Venn\_Shading as vs

from matplotlib import pyplot as plt

for i in range(1):

for j in range(80):

print("-", end='')

print()

print("\t\tWELCOME TO SETT AND VENN DIAGRAM IMPLEMENTATION")

for i in range(1):

for j in range(80):

print("-", end='')

print()

print("Choose either one of the following: ")

print("1. Element Input")

print("2. Venn Shadng")

arr= array([1,2])

print()

mod= int(input("Enter choice here: "))

if(mod== arr[0]):

el.Element\_Input().User\_Inp()

1. **VENN DIAGRAM SHADING CODE**

from matplotlib\_venn import venn2, venn2\_circles, venn2\_unweighted

from numpy import array

from matplotlib import pyplot as plt

class Venn\_Shading:

def \_init\_(self):

self.A=1

self.B=2

self.C=3

self.AandB=4

self.AandC=5

self.BandC=6

def show(self, exp):

#if we get A and B in that string

if ("A" in exp) and ("B" in exp):

#if we found union symbol in the string

if 'u' in exp:

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('red')

v.get\_patch\_by\_id('01').set\_color('red')

v.get\_patch\_by\_id('11').set\_color('red')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('white')

plt.gca().set\_axis\_on()

if 'n' in exp:

#this is for interssection

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('white')

v.get\_patch\_by\_id('01').set\_color('white')

v.get\_patch\_by\_id('11').set\_color('red')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('white')

plt.gca().set\_axis\_on()

#this is for complements

# (A n B)'= A' u B'

if (("A\'" in exp) and ("B\'" in exp)) or(("(AnB)\'" in exp)) or (("(AuB)\'" in exp)):

if ("(AnB)\'" in exp) or ("A\'uB\'" in exp):

# (A n B)'= A' u B'

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('blue')

v.get\_patch\_by\_id('01').set\_color('blue')

v.get\_patch\_by\_id('11').set\_color('white')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('blue')

plt.gca().set\_axis\_on()

else:

# (A u B)'= A' n B'

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('white')

v.get\_patch\_by\_id('01').set\_color('white')

v.get\_patch\_by\_id('11').set\_color('white')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('blue')

plt.gca().set\_axis\_on()

if (("A\'" in exp) and not("B\'" in exp))or("B-A" in exp):

#A'nB= B-A

if ("n" in exp) or("B-A" in exp):

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('white')

v.get\_patch\_by\_id('01').set\_color('red')

v.get\_patch\_by\_id('11').set\_color('white')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('white')

plt.gca().set\_axis\_on()

else:

#A'U B

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('white')

v.get\_patch\_by\_id('01').set\_color('blue')

v.get\_patch\_by\_id('11').set\_color('blue')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('blue')

plt.gca().set\_axis\_on()

if (not("A\'" in exp) and ("B\'" in exp)) or("A-B" in exp):

#AnB'= A-B

if 'n' in exp or("A-B" in exp):

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('red')

v.get\_patch\_by\_id('01').set\_color('white')

v.get\_patch\_by\_id('11').set\_color('white')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('white')

plt.gca().set\_axis\_on()

else:

#AUB'

v=venn2(subsets=(self.A, self.B, self.AandB), set\_labels=('A','B'), alpha=0.8)

v.get\_patch\_by\_id('10').set\_color('blue')

v.get\_patch\_by\_id('01').set\_color('white')

v.get\_patch\_by\_id('11').set\_color('blue')

venn2\_circles(subsets=(self.A, self.B, self.AandB))

plt.gca().set\_facecolor('blue')

plt.gca().set\_axis\_on()

plt.title("VENN DIAGRAM")

plt.show()

1. **ELEMENT INPUT CODE**

from matplotlib\_venn import venn3, venn3\_circles

from numpy import array

from matplotlib import pyplot as plt

class Element\_Input:

"""description of class"""

def \_init\_(self):

self.universal= None

self.A= None

self.B= None

self.C= None

self.AnBnC= None

self.AnB= None

self.AnC= None

self.BnC= None

def User\_Inp(self):

self.universal=int(input("Enter Universal set: "))

self.A=int(input("Enter A value: "))

self.B=int(input("Enter B value: "))

self.C=int(input("Enter C value: "))

self.AnBnC=int(input("Enter A and B and C value: "))

self.AnB=int(input("Enter A and B value: "))

self.AnC=int(input("Enter A and C value: "))

self.BnC=int(input("Enter B and C value: "))

self.Operation()

def Operation(self):

#if any of the intersections is greater than universal set,

#generate an error

if((self.AnB>self.universal) or (self.AnBnC>self.universal) or (self.AnC> self.universal)):

print("Your intersection should not be greater than universal value!! ")

elif((self.A>self.universal)):

print("A set should not be greater than universal set")

elif((self.B>self.universal)):

print("B set should not be greater than universal set")

elif((self.C>self.universal)):

print("C set should not be greater than universal set")

elif((self.AnB>self.AnB) or (self.AnBnC>self.AnC) or (self.AnC> self.BnC)):

print("AnBnC should not be greater than either of the intersenctions")

else:

self.AnB-= self.AnBnC

self.BnC-= self.AnBnC

self.AnC-= self.AnBnC

self.A-=(self.AnB+self.AnBnC+self.AnC)

self.B-=(self.AnB+self.AnBnC+self.BnC)

self.C-=(self.AnC+self.AnBnC+self.BnC)

self.universal-=(self.A+self.B+self.C+self.AnB+self.BnC+self.AnC)

self.Value\_Put(self.A, self.B, self.C, self.AnB, self.BnC, self.AnC, self.AnBnC)

def Value\_Put(self, A, B, C, AnB, BnC, AnC, AnBnC):

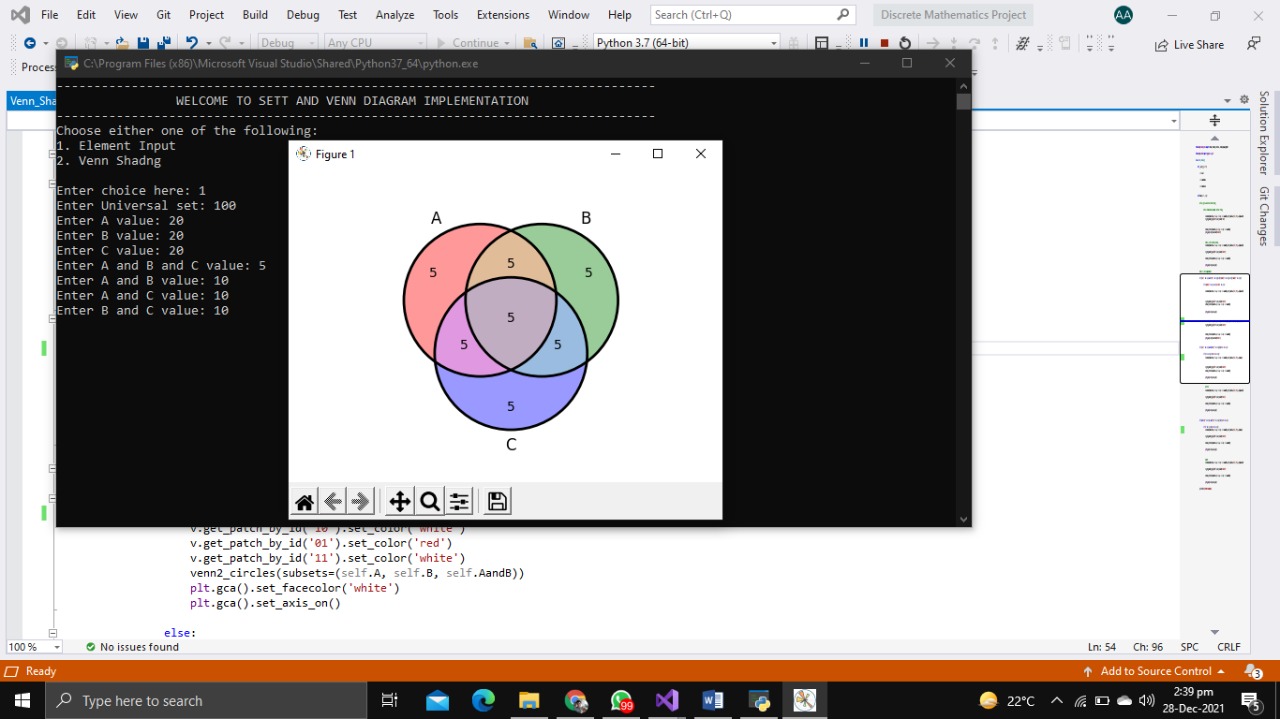
v=venn3(subsets=(A, B, AnB, C, AnC, BnC, AnBnC), set\_labels=('A','B','C'))

venn3\_circles(subsets=(A, B, AnB, C, AnC, BnC, AnBnC))

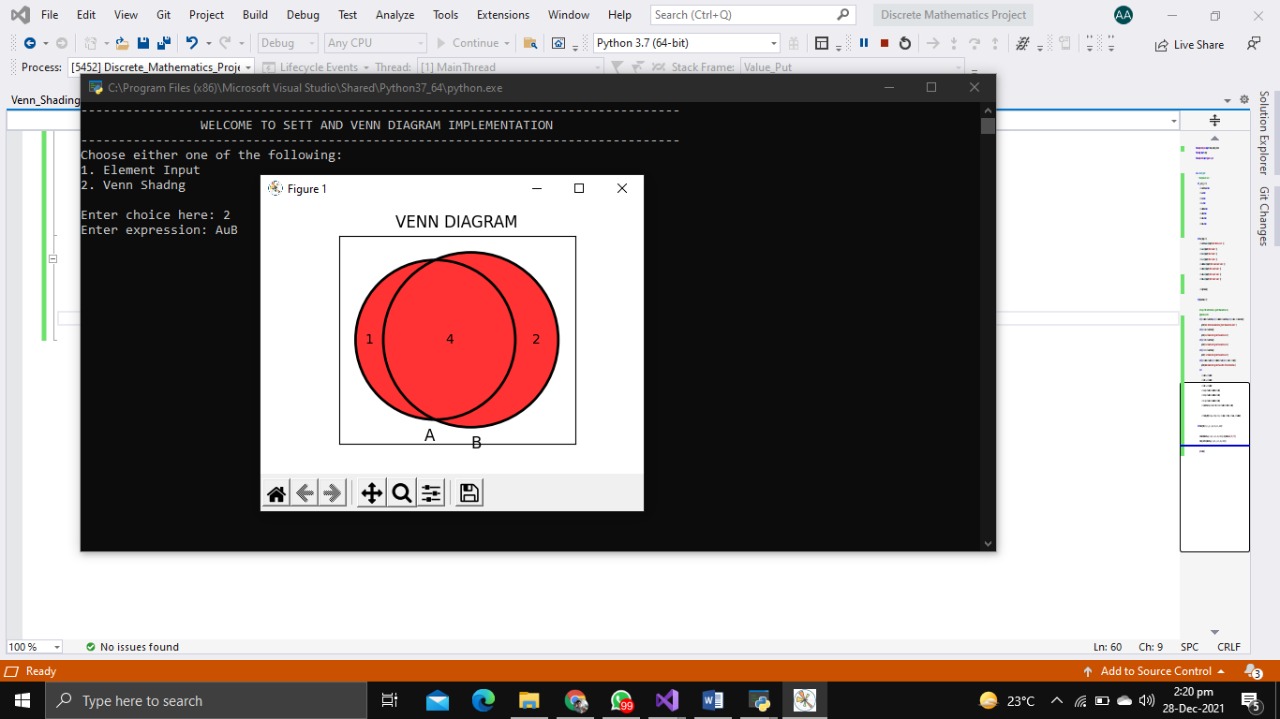
plt.show()

* **OUTPUTS OF FUNCTIONS**

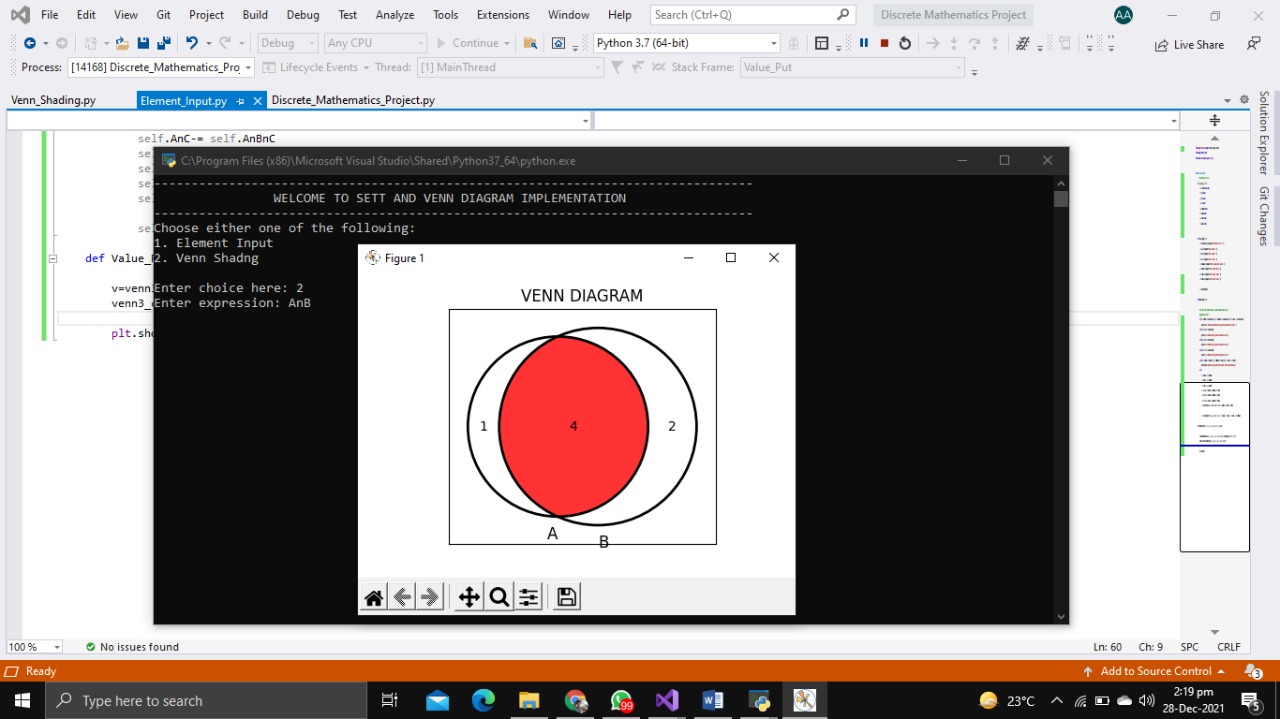
“ELEMENT INPUT”



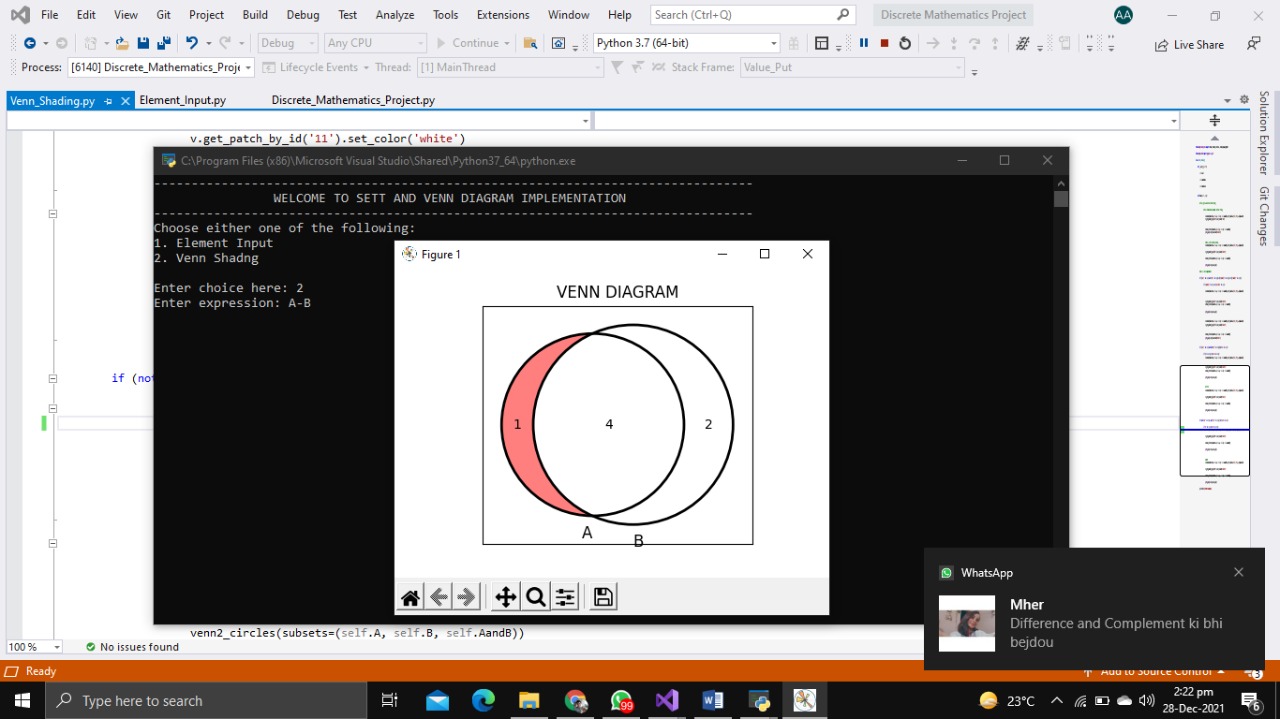
“A U B”



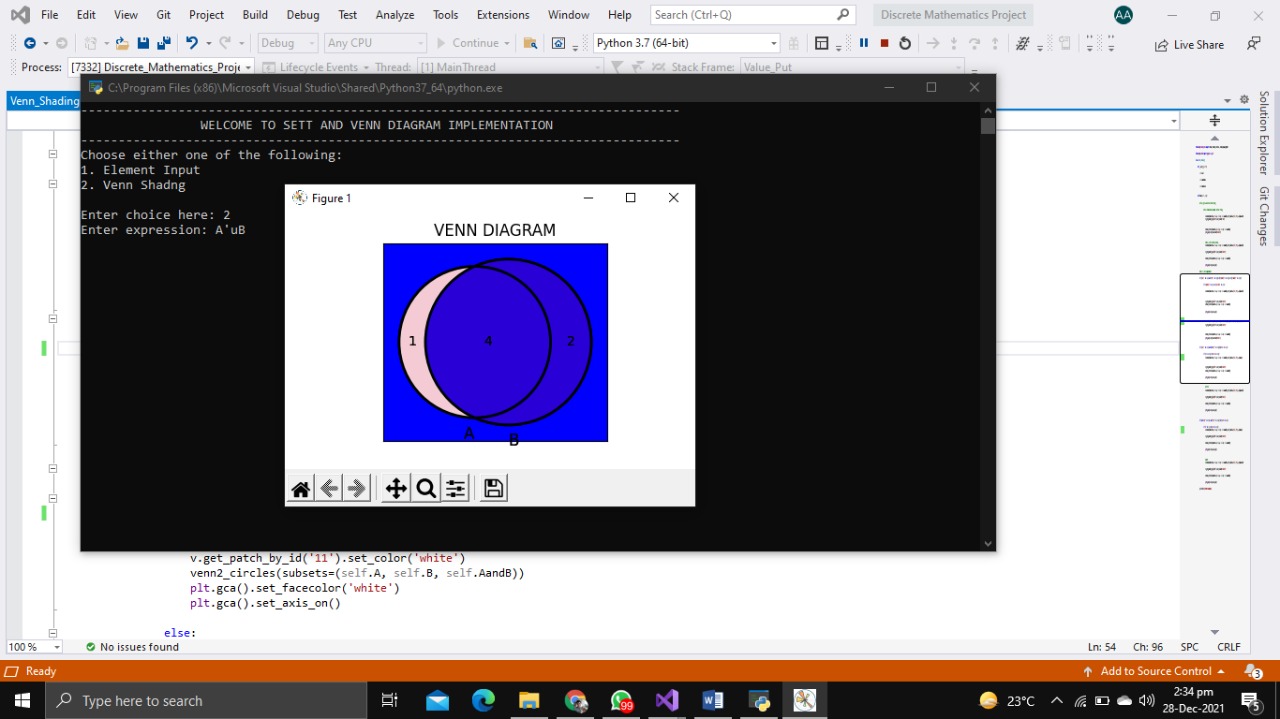
“A n B”



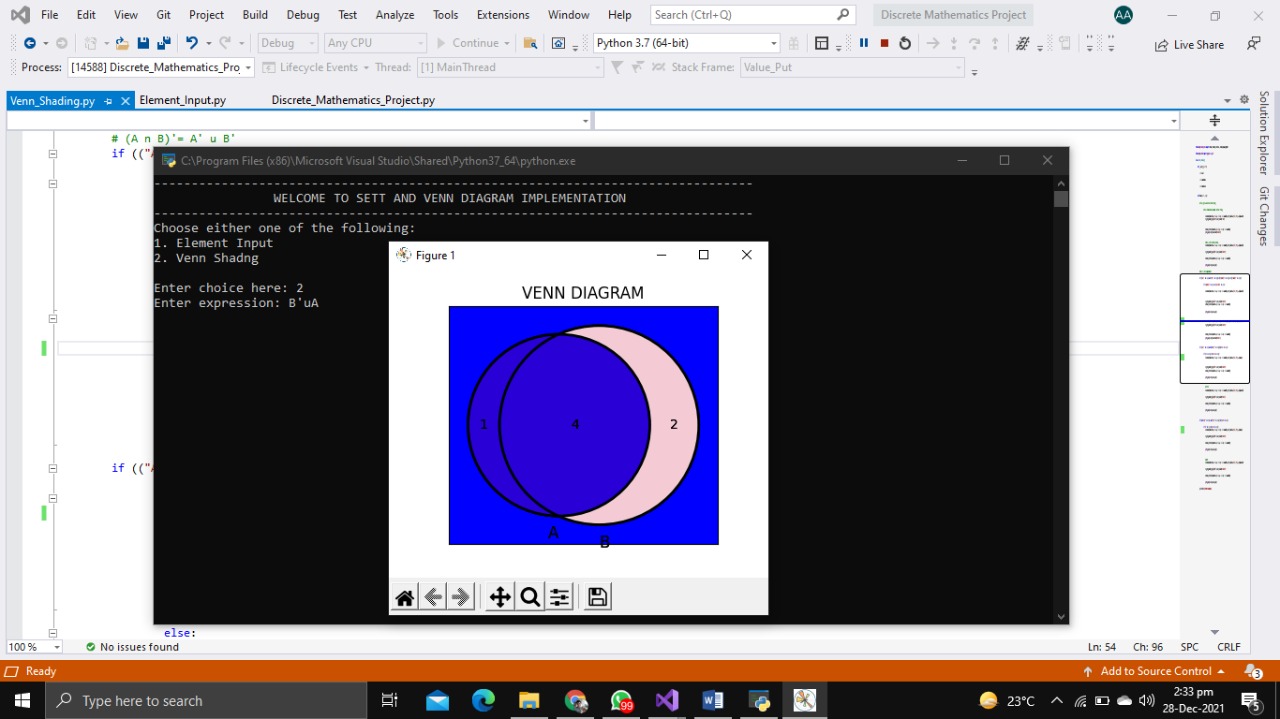
“A-B”



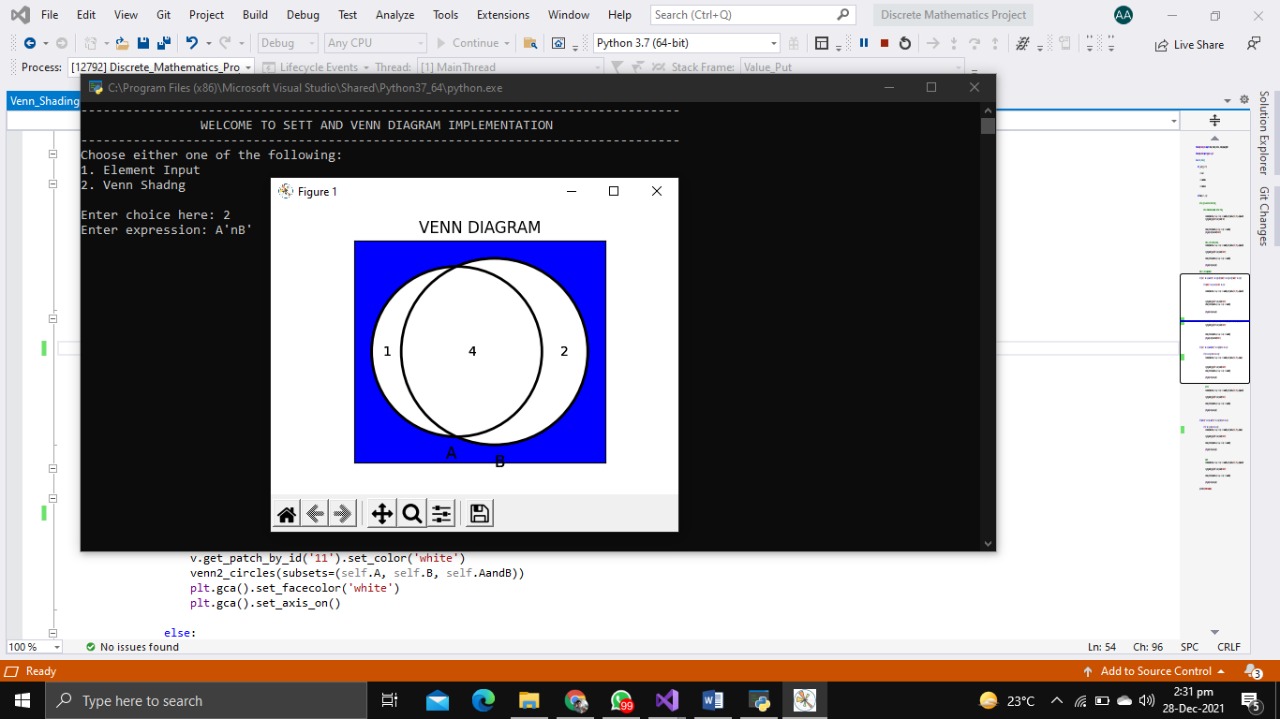
“A’ U B”



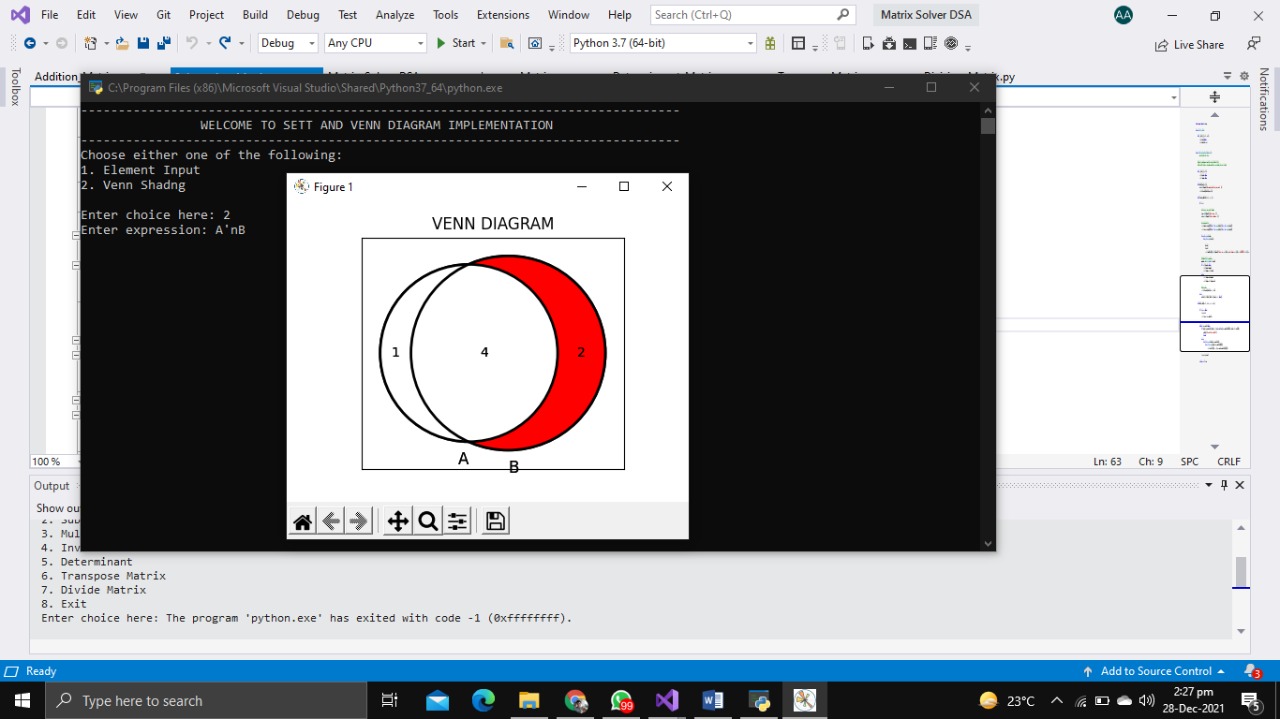
“ A U B’ ”



“A’ n B’ ”



“A’ n B”



“A n B’ ”

