**Q:** Implement fuzzy membership functions as discussed in the class. Your program should contain functions for R-function, L-function, triangular-function, and trapezoidal function. The program should ask required arguments for above functions and should generate fuzzy sets.

Solution:

**Code available at:** <https://colab.research.google.com/drive/1pLsICoPWl1xjBNVOzkTznbIGVDL0aIZe?usp=sharing>

class FuzzySet:

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

self.elements = []

self.memberships = []

def add\_element(self, element):

self.elements.append(element)

def r\_function(self, alpha\_value, beta\_value, set1):

mem = []

for element in set1.elements:

if element <= alpha\_value:

mem.append(0)

elif element >= beta\_value:

mem.append(1)

else:

mem\_value = (float(element) - float(alpha\_value)) / (float(beta\_value) - float(alpha\_value))

mem.append(round(mem\_value, 2))

for element, membership in zip(set1.elements, mem):

print(f"{element} - {membership}")

def l\_function(self, alpha\_value, beta\_value, set1):

mem = []

for element in set1.elements:

if element <= alpha\_value:

mem.append(1)

elif element >= beta\_value:

mem.append(0)

else:

mem\_value = (float(beta\_value) - float(element)) / (float(beta\_value) - float(alpha\_value))

mem.append(round(mem\_value, 2))

for element, membership in zip(set1.elements, mem):

print(f"{element} - {membership}")

def t\_function(self, alpha\_value, beta\_value, gamma\_value, set1):

mem = []

for element in set1.elements:

if element < alpha\_value or element > gamma\_value:

mem.append(0)

elif alpha\_value <= element <= beta\_value:

mem\_value = (float(element) - float(alpha\_value)) / (float(beta\_value) - float(alpha\_value))

mem.append(round(mem\_value, 2))

else:

mem\_value = (float(gamma\_value) - float(element)) / (float(gamma\_value) - float(beta\_value))

mem.append(round(mem\_value, 2))

for element, membership in zip(set1.elements, mem):

print(f"{element} - {membership}")

def tra\_function(self, alpha\_value, beta\_value, gamma\_value, sigma\_value, set1):

mem = []

for element in set1.elements:

if element < alpha\_value or element > sigma\_value:

mem.append(0)

elif alpha\_value <= element <= beta\_value:

mem\_value = (float(element) - float(alpha\_value)) / (float(beta\_value) - float(alpha\_value))

mem.append(round(mem\_value, 2))

elif beta\_value <= element <= gamma\_value:

mem.append(1)

else:

mem\_value =(float(sigma\_value) - float(element)) / (float(sigma\_value) - float(gamma\_value))

mem.append(round(mem\_value, 2))

for element, membership in zip(set1.elements, mem):

print(f"{element} - {membership}")

def print\_set(self):

for i in range(len(self.elements)):

print(self.elements[i])

set1 = FuzzySet()

n = int(input("Enter the number of elements in set: "))

for i in range(n):

element = input(f"Enter element {i+1} in set: ")

set1.add\_element(element)

print("\nSet:")

set1.print\_set()

rFunction = FuzzySet()

alpha\_value = input(f"\nEnter the alpha value : ")

beta\_value = input(f"\nEnter the beta value : ")

gamma\_value = input(f"\nEnter the gamma value : ")

sigma\_value = input(f"\nEnter the sigma value : ")

print("\nR-Function:")

rFunction.r\_function(alpha\_value, beta\_value, set1)

lFunction = FuzzySet()

print("\nL-Function:")

lFunction.l\_function(alpha\_value, beta\_value, set1)

tFunction = FuzzySet()

print("\nTriangular-Function:")

tFunction.t\_function(alpha\_value, beta\_value, gamma\_value, set1)

traFunction = FuzzySet()

print("\nTrapezoid-Function:")

traFunction.tra\_function(alpha\_value, beta\_value, gamma\_value,sigma\_value, set1)

**Output**

Enter the number of elements in set: 7

Enter element 1 in set: 20

Enter element 2 in set: 30

Enter element 3 in set: 40

Enter element 4 in set: 50

Enter element 5 in set: 60

Enter element 6 in set: 70

Enter element 7 in set: 80

Set:

20

30

40

50

60

70

80

Enter the alpha value : 40

Enter the beta value : 70

Enter the gamma value : 70

Enter the sigma value : 70

R-Function:

20 - 0

30 - 0

40 - 0

50 - 0.33

60 - 0.67

70 - 1

80 - 1

L-Function:

20 - 1

30 - 1

40 - 1

50 - 0.67

60 - 0.33

70 - 0

80 - 0

Triangular-Function:

20 - 0

30 - 0

40 - 0.0

50 - 0.33

60 - 0.67

70 - 1.0

80 - 0

Trapezoid-Function:

20 - 0

30 - 0

40 - 0.0

50 - 0.33

60 - 0.67

70 - 1.0

80 - 0