Name- Yash Gandhi BE IT Batch A 2017140014

Aim:

To implement a fuzzy library for 1D fuzzy sets

Problem Statements:

PS1

Design and implement a fuzzy library comprising the following Fuzzy set operations for discrete Universe 1D Fuzzy Sets

1. Containment, Union, Intersection, and Complement.
2. Verify the De-Morgan's law.

PS2

Design a fuzzy library for representing the standard membership functions of 1D Fuzzy Sets for Continuous Universe of Discourse

Tool/Language:

Programming language: Python (matplotlib,numpy)

Formulae/ Equations:

Complement:

Output= 1 - x

Union:

Output = max(1xA, 2xA)

Intersection:

Output= min(1xA, 2xA)

```
Gaussian(x,c,sigma):
   z = -0.5 * (((x-c)/sigma)^2)

triangular(x,lower,top,lower2):
      if x<=lower:
         return 0
      elif lower<=x and x<=top:
         return (x-lower)/(top-lower)
      elif top<=x and x<=lower2:
         return (lower2-x)/(lower2-top)
      else:
         return 0
```

Code:

PS1:

```
In [2]: class fuzzy_operations(object):

            def complement(self,set1):
                output = {}
                for i in range(len(set1)):
                    output[i+1] = round(1 - set1[i+1],3)

                return output

            def union(self,set1,set2):
                output = {}
                for i in range(len(set1)):
                    output[i+1] = max(set1[i+1], set2[i+1])

                return output
            def intersection(self,set1,set2):
                output = {}
                for i in range(len(set1)):
                    output[i+1] = min(set1[i+1], set2[i+1])

                return output

            def containment(self,set1,set2):
                c1 = 0
                c2 = 0
                for i in range(len(set1)):
                    if set1[i+1] >= set2[i+1]:
                        c1 = c1+1
                    if set1[i+1] <= set2[i+1]:
                        c2 = c2 + 1
                if c1 == len(set1):
                    print('set1 contains set2')
                    return
                if c2 == len(set2):
                    print('set2 contains set1')
                    return
                else:
                    print('No containment')
                    return
```

```
    def de_morgan(self,set1,set2):

        print('~(AnB) = ~A U ~B')
        part1 = self.complement(self.intersection(set1,set2))
        part2 = self.union(self.complement(set1),self.complement(set2))
        print(part1)
        print(part2)
        if(part1==part2):
            print('satisfies')
        print('~(AUB) = ~A n ~B')
        part1 = self.complement(self.union(set1,set2))
        part2 = self.intersection(self.complement(set1),self.complement(set2))
        print(part1)
        print(part2)
        if(part1==part2):
            print('satisfies')
        return
```

Results:

Test Cases:

PS 1:

Test your program for the following cases

1. A={(1,0.2), (2,0.3),(3,0.8),(4,1)} and B={(1,0.3),(2,0.2) ,(3,0.5),(4,0.8)}
2. A={(2,0.3),(3,0.5) } and B={(1,0.2),(2,0.3), (3,0.6),(4,0.8)}

```
In [3]: set1 = {1:0.2, 2:0.3, 3:0.8, 4:1}
        set2 = {1:0.3, 2:0.2, 3:0.5, 4:0.8}
        diff=set(set1)-set(set2)
        while diff:
            element=diff.pop()
            set2[element]=0
```

```
In [4]: obj = fuzzy_operations()
        complement = {}
        complement = obj.complement(set1)
        print('Complement')
        print(complement)
```

```
Complement
{1: 0.8, 2: 0.7, 3: 0.2, 4: 0}
```

```
In [5]: union = {}
        union = obj.union(set1,set2)
        print('Union')
        print(union)
```

```
Union
{1: 0.3, 2: 0.3, 3: 0.8, 4: 1}
```

```
In [6]: intersection = {}
        intersection = obj.intersection(set1,set2)
        print('Intersection')
        print(intersection)
```

```
Intersection
{1: 0.2, 2: 0.2, 3: 0.5, 4: 0.8}
```

```
{1: 0.2, 2: 0.2, 3: 0.5, 4: 0.8}
```

```
In [7]: print('Containment :')
        obj.containment(set1,set2)
```

```
Containment :
No containment
```

```
In [8]: print('Verifying Demorgans laws')
        obj.de_morgan(set1,set2)
```

```
Verifying Demorgans laws
~(AnB) = ~A U ~B
{1: 0.8, 2: 0.8, 3: 0.5, 4: 0.2}
{1: 0.8, 2: 0.8, 3: 0.5, 4: 0.2}
satisfies
~(AUB) = ~A n ~B
{1: 0.7, 2: 0.7, 3: 0.2, 4: 0}
{1: 0.7, 2: 0.7, 3: 0.2, 4: 0}
satisfies
```

```
In [13]: set1 = {2:0.3, 3:0.5}
         set2 = {1:0.2, 2:0.3, 3:0.6, 4:0.8}
         diff=set(set2)-set(set1)
         while diff:
             element=diff.pop()
             set1[element]=0
```

```
In [14]: obj = fuzzy_operations()
         complement = {}
         complement = obj.complement(set1)
         print('Complement')
         print(complement)
```

```
Complement
{1: 1, 2: 0.7, 3: 0.5, 4: 1}
```

In [15]:
```
union = {}
union = obj.union(set1,set2)
print('Union')
print(union)
```

```
Union
{1: 0.2, 2: 0.3, 3: 0.6, 4: 0.8}
```

In [16]:
```
intersection = {}
intersection = obj.intersection(set1,set2)
print('Intersection')
print(intersection)
```

```
Intersection
{1: 0, 2: 0.3, 3: 0.5, 4: 0}
```

In [17]:
```
print('Containment :')
obj.containment(set1,set2)
```

```
Containment :
set2 contains set1
```

In [18]:
```
print('Verifying Demorgans laws')
obj.de_morgan(set1,set2)
```

```
Verifying Demorgans laws
~(AnB) = ~A U ~B
{1: 1, 2: 0.7, 3: 0.5, 4: 1}
{1: 1, 2: 0.7, 3: 0.5, 4: 1}
satisfies
~(AUB) = ~A n ~B
{1: 0.8, 2: 0.7, 3: 0.4, 4: 0.2}
{1: 0.8, 2: 0.7, 3: 0.4, 4: 0.2}
satisfies
```

In [ ]:
```

```

PS 2:

Represent the following fuzzy sets using the MFs created

1. Age (Young, Middle Aged, Old) – Experiment with appropriate variants of MFs available
2. High speeds for Racing Cars
3. Temperature ranges of air-conditioners

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Fuzzy membership for Age using Gaussian function

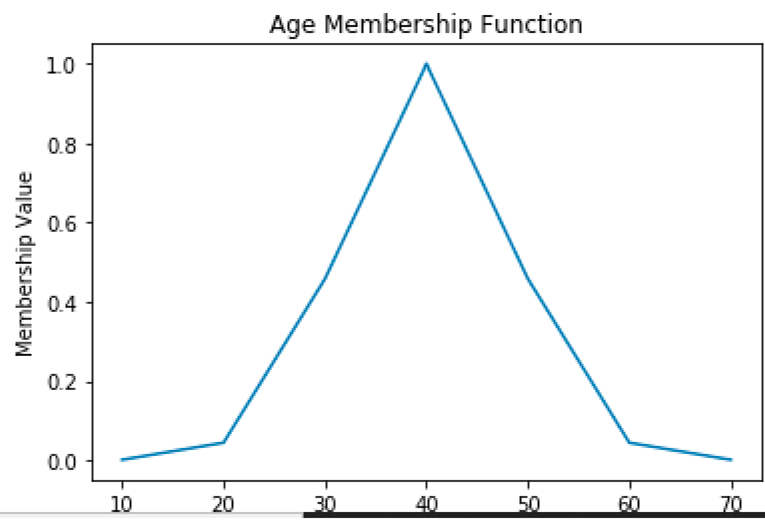
In [2]:
```
def gaussian(x,c,sigma):
    z = -0.5 * (((x-c)/sigma)**2)
    return np.exp(z)
```

In [3]:
```
x = [i for i in range(10,80,10)]
inputv = []
print("Age       Value")
for j in x:
    value = gaussian(j,40,8)
    print(str(j) + " = " + str(value))
    inputv.append(value)

plt.plot(x,inputv)
plt.title("Age Membership Function")
plt.xlabel("Age")
plt.ylabel("Membership Value")
```

```
Age       Value
10 = 0.00088382630693505
20 = 0.04393693362340742
30 = 0.45783336177161427
40 = 1.0
50 = 0.45783336177161427
60 = 0.04393693362340742
70 = 0.00088382630693505
```

Out[3]: `Text(0,0.5,'Membership Value')`



Fuzzy membership for Age using Triangular function

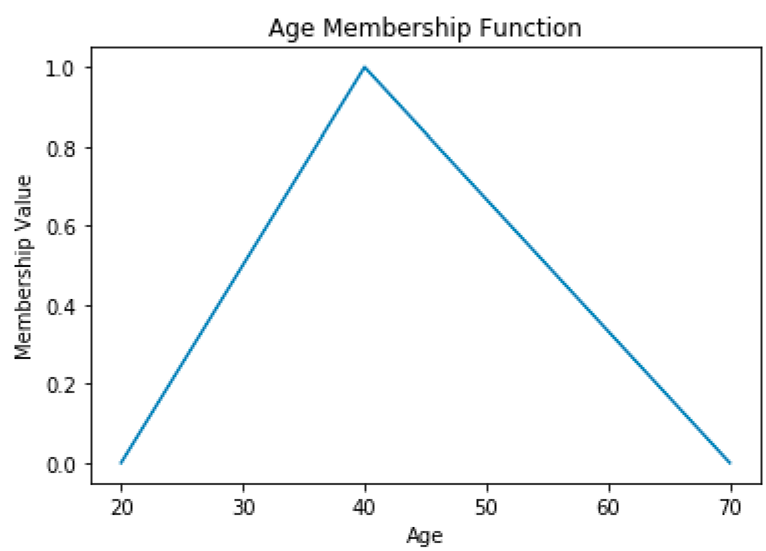
In [4]:
```
def triangular(x,lower,top,lower2):
        if x<=lower:
            return 0
        elif lower<=x and x<=top:
            return (x-lower)/(top-lower)
        elif top<=x and x<=lower2:
            return (lower2-x)/(lower2-top)
        else:
            return 0
```

In [5]:
```
x = [i for i in range(20,80,10)]
inputv = []
print("Age       Value")
for j in x:
    value = triangular(j,20,40,70)
    print(str(j) + " = " + str(value))
    inputv.append(value)

plt.plot(x,inputv)
plt.title("Age Membership Function")
plt.xlabel("Age")
plt.ylabel("Membership Value")
```

```
Age       Value
20 = 0
30 = 0.5
40 = 1.0
50 = 0.6666666666666666
60 = 0.3333333333333333
70 = 0.0
```

Out[5]: `Text(0,0.5,'Membership Value')`



Fuzzy membership for Speed Car using increasing function

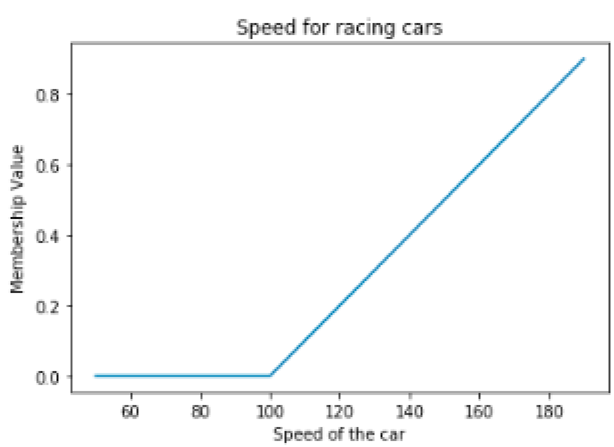
In [6]:
```
def increasing(value,lower,upper):
        if value<=lower:
            return 0
        elif value>=upper:
            return 1
        else:
            return (value-lower)/(upper-lower)
```

In [7]:
```
x = [i for i in range(50,200,10)]
inputv = []
print("Speed      Value")
for j in x:
    value = increasing(j,100,200)
    print(str(j) + " = " + str(value))
    inputv.append(value)

plt.plot(x,inputv)
plt.title("Speed for racing cars")
plt.xlabel("Speed of the car")
plt.ylabel("Membership Value")
```

```
Speed      Value
50 = 0
60 = 0
70 = 0
80 = 0
90 = 0
100 = 0
110 = 0.1
120 = 0.2
130 = 0.3
140 = 0.4
150 = 0.5
160 = 0.6
170 = 0.7
180 = 0.8
190 = 0.9
```

Out[7]: `Text(0,0.5,'Membership Value')`



Fuzzy membership for AC Temperature Range using Gaussian function

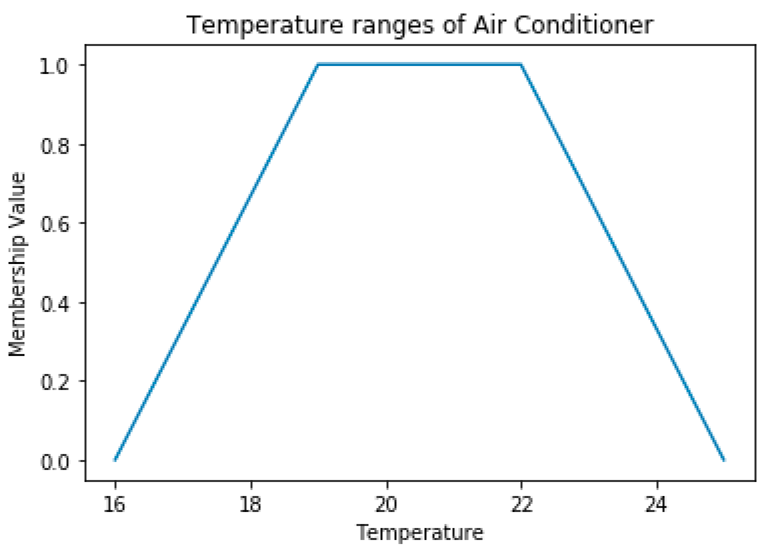
In [8]:
```
def trapezoidal(x,lower,top,top2,lower2):
        if x<=lower:
            return 0
        elif lower<=x and x<=top:
            return (x-lower)/(top-lower)
        elif top<=x and x<=top2:
            return 1
        elif top2<=x and x<=lower2:
            return (lower2-x)/(lower2-top2)
        else:
            return 0
```

In [9]:
```
x = [i for i in range(16,26,1)]
inputv = []
print("Temp     Value")
for j in x:
    value = trapezoidal(j,16,19,22,25)
    print(str(j) + " = " + str(value))
    inputv.append(value)

plt.plot(x,inputv)
plt.title("Temperature ranges of Air Conditioner")
plt.xlabel("Temperature")
plt.ylabel("Membership Value")
```

```
Temp     Value
16 = 0
17 = 0.3333333333333333
18 = 0.6666666666666666
19 = 1.0
20 = 1
21 = 1
22 = 1
23 = 0.6666666666666666
24 = 0.3333333333333333
25 = 0.0
```

Out[9]: `Text(0,0.5,'Membership Value')`



Conclusion:

Fuzzy library comprising the Fuzzy set operations Containment, Union, Intersection, and Complement like for discrete Universe 1D Fuzzy Sets were implemented The De-Morgan's law was also verified.

Also a fuzzy library for representing the standard membership functions of 1D Fuzzy Sets for Continuous Universe of Discourse was implemented for Age, Speed of Racing car and AC temperature using membership functions like Gaussian, increasing, triangular and trapezoidal.