NAME : Krish Sukhani

Class : TE IT

Batch : D

UID : 2018140059

Soft Compution Lab 1

Aim:

To implement a fuzzy library for 1D fuzzy sets

Problem Statements:

Pre-Lab

Design and implement a library with the following operations on Classical sets: Union, Intersection, Complement

```
set1 = set()
```

```
set2 = set()
```

```
#case1 : set1_n = 4
#case2 : set1_n = 4
```

```
set1_n = int(input("Enter the number of elements in set 1 : "))
```

```
    Enter the number of elements in set 1 : 4
```

```
#case1 : set1_n = 3
#case2 : set1_n = 2
```

```
set2_n = int(input("Enter the number of elements in set 2 : "))
```

```
Enter the number of elements in set 2 : 2
```

```
for _ in range(set1_n):
  set1.add(input("Enter the element value in set1 : "))
```

```
Enter the element value in set1 : 1
Enter the element value in set1 : 2
Enter the element value in set1 : 3
Enter the element value in set1 : 4
```

```
for _ in range(set2_n):
  set2.add(input("Enter the element value in set2 : "))
```

```
Enter the element value in set2 : 2
Enter the element value in set2 : 5
```

```
#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '2', '3', '4'}
```

```
set1
```

```
{'1', '2', '3', '4'}
```

```
#case1 : set1 = {'5', '6', '7'}
#case2 : set1_n = {'2', '5'}
```

```
set2
```

```
{'2', '5'}
```

UNION

```
union = set1.union(set2)
```

```
#case1 : set1 = {'1', '2', '3', '4', '5', '6', '7'}
#case2 : set1_n = {'1', '2', '3', '4', '5'}
```

```
union
```

```
{'1', '2', '3', '4', '5'}
```

By code

```
set3 = set()


for element in set1:
  set3.add(element)


for element in set2:
  set3.add(element)


#case1 : set1 = {'1', '2', '3', '4', '5', '6', '7'}
#case2 : set1_n = {'1', '2', '3', '4', '5'}


set3
```

```
{'1', '2', '3', '4', '5'}
```

INTERSECTION

```
intersection = set1.intersection(set2)


#case1 : set1 = {}
#case2 : set1_n = {'2'}


intersection
```

```
{'2'}
```

By code

```
int_by_code = set()
```

```
for  element in set1:
  for element1 in set2:
    if element == element1:
      int_by_code.add(element)


#case1 : set1 = {}
#case2 : set1_n = {'2'}


int_by_code
```

```
{'2'}
```

COMPLEMENT

```
complement = set2.difference(set1)


#case1 : set1 = {'5', '6', '7'}
#case2 : set1_n = {'5'}


complement
```

```
{'5'}
```

```
complement2 = set1.difference(set2)


#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '3', '4'}


complement2
```

```
{'1', '3', '4'}
```

By Code

```
#case1 : set1 = {'5', '6', '7'}
```

```
#case2 : set1_n = {'5'}
```

```
comp1_bycode = set3 - set1
```

```
comp1_bycode
```

```
{'5'}
```

```
#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '3', '4'}
```

```
comp2_bycode = set3 - set2
```

```
comp2_bycode
```

```
{'1', '3', '4'}
```

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.

```
A = {'2': 0.3, '3': 0.5}
```

```
B = {'1': 0.2, '2': 0.3, '3': 0.6, '4': 0.8}
```

```
universe = ['1', '2', '3', '4']
```

```
for i in universe:
  if i not in A.keys():
    A[i] = 0
```

```
A
```

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

```
for i in universe:
  if i not in B.keys():
    B[i] = 0
```

```
B
```

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

```
def union(A,B):
  u = {}
  for i in A:
    if i in B:
      u[i]=max(A[i],B[i])
    else:
      u[i]=A[i]
  for i in B:
    if i not in A:
      u[i]=B[i]
  return (u)
```

```
union(A,B)
```

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

```
def intersection(A,B):
  inter = {}
  for i in A:
    if i in B:
      inter[i]=min(A[i],B[i])
  return inter
```

```
intersection(A,B)
```

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

```
def complement(fuzzy_set):
```

```
def complement(fuzzy_set):
  complement_of_set={}
  for i in fuzzy_set:
    complement_of_set[i] = round((1-fuzzy_set[i]),1)
  return complement_of_set
```

```
complement(A)
```

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

```
complement(B)
```

```
{'1': 0.8, '2': 0.7, '3': 0.4, '4': 0.2}
```

```
def one_sub_two(one,two):
  flag = 0
  for i in one:
    if i in two:
      if (one[i] <= two[i]):
        flag += 1
    else:
      continue
  if (flag == len(one)):
    print('Yes')
  else:
    print('No')
```

```
one_sub_two(A,B)
```

```
Yes
```

```
one_sub_two(B,A)
```

```
No
```

DEMorgan Laws

```
complement(intersection(A,B)) == union(complement(A),complement(B))
```

```
True
```

```
complement(union(A,B)) == intersection(complement(A),complement(B))
```

```
True
```

PS1 - Case 2

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.

```
A = {'1': 0.2, '2': 0.3, '3': 0.8, '4': 1}
```

```
B = {'1': 0.3, '2': 0.2, '3': 0.5, '4': 0.8}
```

```
universe = ['1', '2', '3', '4']
```

```
for i in universe:
  if i not in A.keys():
    A[i] = 0
```

```
A
```

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

```
for i in universe:
  if i not in B.keys():
    B[i] = 0
```

```
B
```

```
{'1': 0.3, '2': 0.2, '3': 0.5, '4': 0.8}
```

```
def union(A,B):
  u = {}
  for i in A:
    if i in B:
      u[i]=max(A[i],B[i])
    else:
      u[i]=A[i]
  for i in B:
    if i not in A:
      u[i]=B[i]
  return (u)


union(A,B)
```

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

```
def intersection(A,B):
  inter = {}
  for i in A:
    if i in B:
      inter[i]=min(A[i],B[i])
  return inter


intersection(A,B)
```

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

```
def complement(fuzzy_set):
  complement_of_set={}
  for i in fuzzy_set:
    complement_of_set[i] = round((1-fuzzy_set[i]),1)
  return complement_of_set



complement(A)
```

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

```
complement(B)
```

```
{'1': 0.7, '2': 0.8, '3': 0.5, '4': 0.2}
```

```
def one_sub_two(one,two):
  flag = 0
  for i in one:
    if i in two:
      if (one[i] <= two[i]):
        flag += 1
    else:
      continue
  if (flag == len(one)):
    print('Yes')
  else:
    print('No')
```

```
one_sub_two(A,B)
```

```
No
```

```
one_sub_two(B,A)
```

```
No
```

DEMorgan Laws

```
complement(intersection(A,B)) == union(complement(A),complement(B))
```

```
True
```

```
complement(union(A,B)) == intersection(complement(A),complement(B))
```

```
True
```

PS2

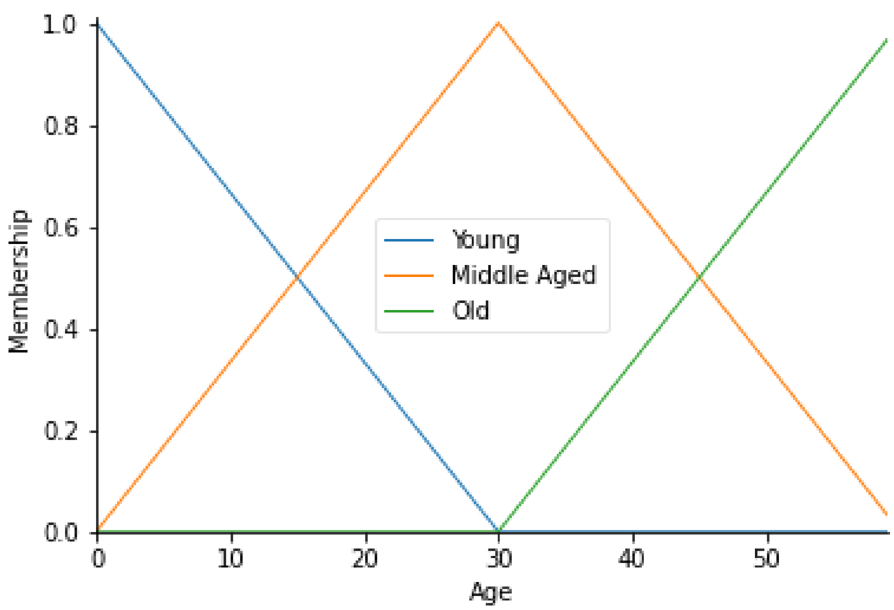
Design a fuzzy library for representing the standard membership functions of 1D Fuzzy Sets for Continuous Universe of Discourse

```
!pip install -U scikit-fuzzy
```

```
Collecting scikit-fuzzy
  Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
     |████████████████████████████████| 993 kB 4.9 MB/s
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.
Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/pytho
Building wheels for collected packages: scikit-fuzzy
  Building wheel for scikit-fuzzy (setup.py) ... done
  Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-py3-none
  Stored in directory: /root/.cache/pip/wheels/d5/74/fc/38588a3d2e3f34
Successfully built scikit-fuzzy
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.4.2
```

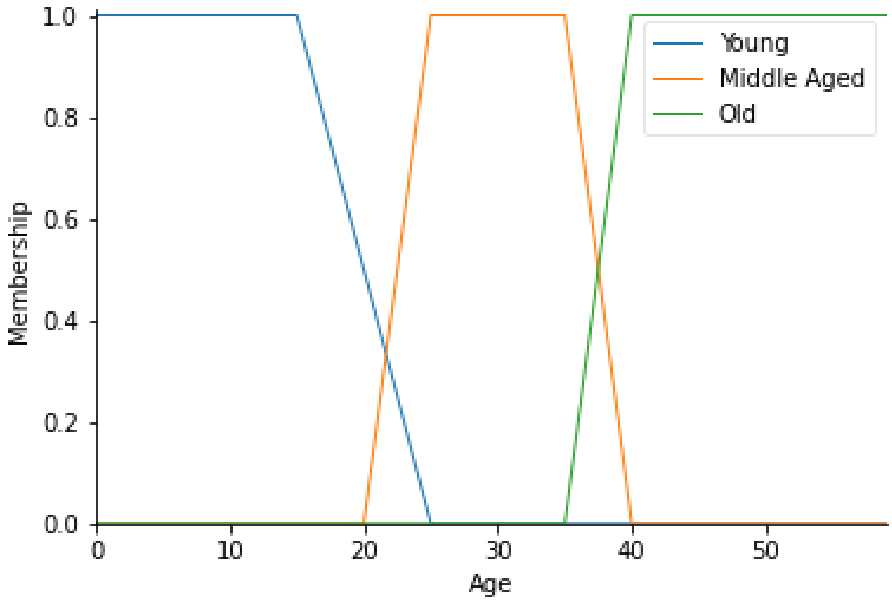
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl


# INPUTS
Age= ctrl.Antecedent(np.arange(0,60, 1), 'Age')
##Using Triangular Membership Function
Age['Young'] = fuzz.trimf(Age.universe, [0, 0, 30])
Age['Middle Aged'] = fuzz.trimf(Age.universe, [0, 30, 60])
Age['Old'] = fuzz.trimf(Age.universe, [30,60,60])
Age.view()
```

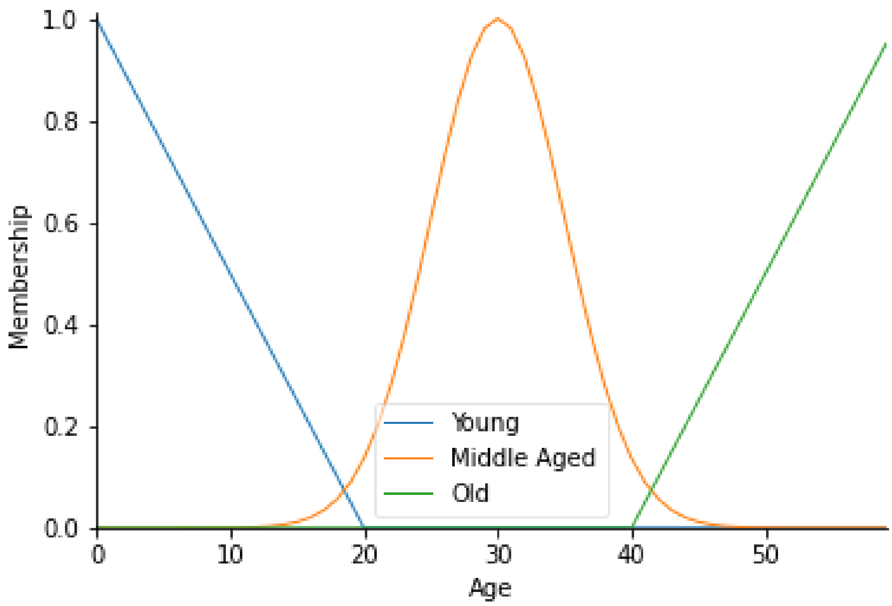



```
##Using Trapezoid Membership Function
Age['Young'] = fuzz.trapmf(Age.universe, [0, 0,15,25])
Age['Middle Aged'] = fuzz.trapmf(Age.universe, [20,25,35,40])
```

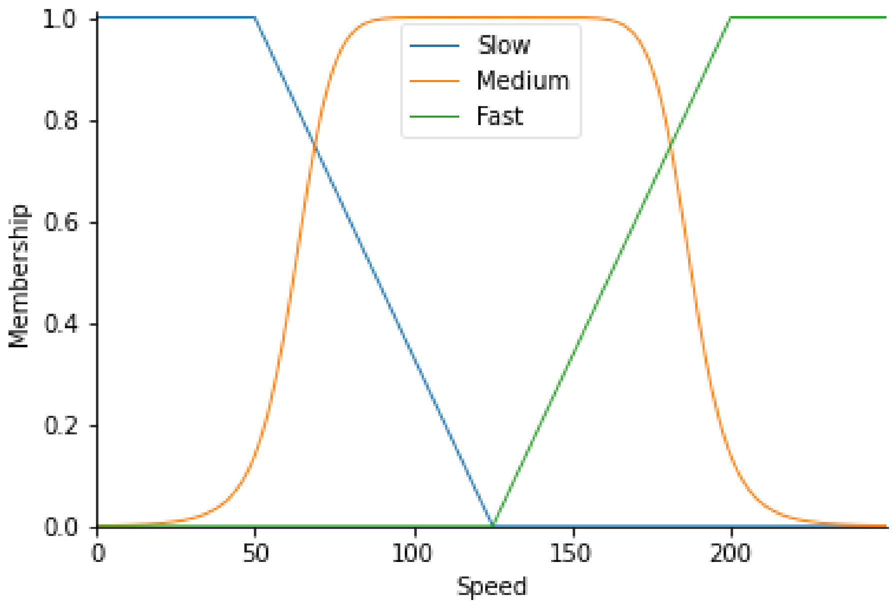
```
Age['Old'] = fuzz.trapmf(Age.universe, [35,40,60,60])
Age.view()
```



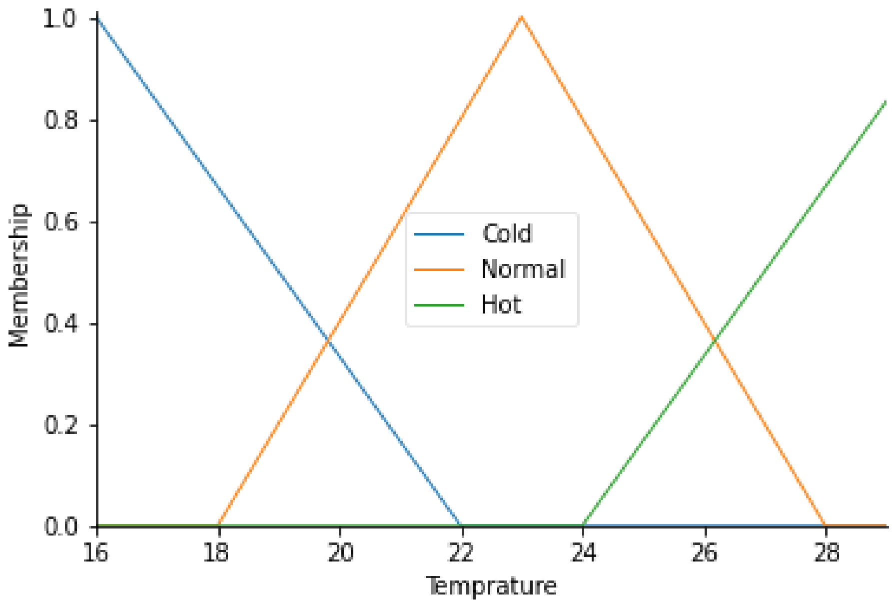

```
##Using Triangular and Gaussian Membership Function
Age['Young'] = fuzz.trimf(Age.universe, [0, 0, 20])
Age['Middle Aged'] = fuzz.gaussmf(Age.universe,30,5)
Age['Old'] = fuzz.trimf(Age.universe, [40,60,60])
Age.view()
```



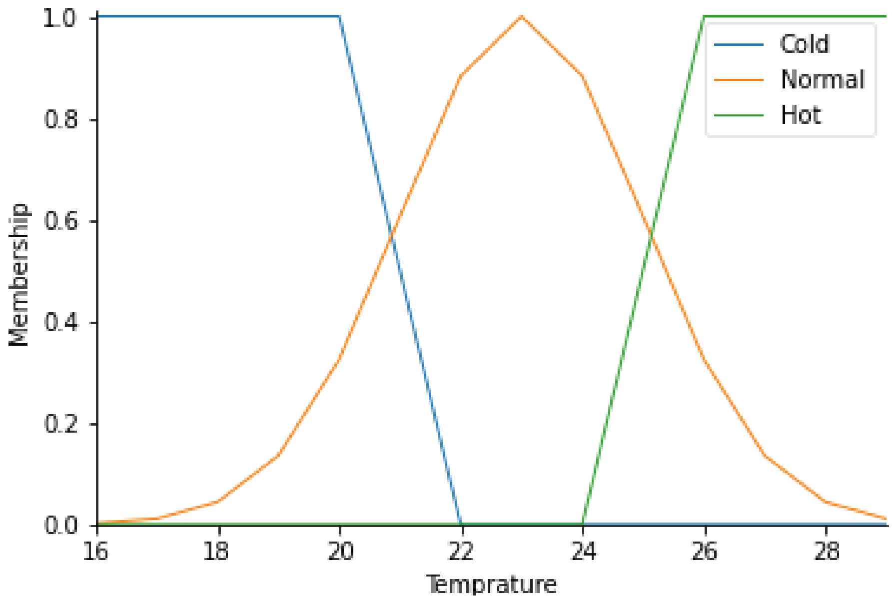

```
Speed= ctrl.Antecedent(np.arange(0,250, 1), 'Speed')
##Using Trapezoid and Generalized Bell Membership Function
Speed['Slow'] = fuzz.trapmf(Speed.universe, [0,0, 50, 125])
Speed['Medium'] = fuzz.gbellmf(Speed.universe,62.5,5,125 )
Speed['Fast'] = fuzz.trapmf(Speed.universe, [125, 200,250, 250])
Speed.view()
```




```
Temp = ctrl.Antecedent(np.arange(16, 30, 1), 'Temprature')
##Using Traingular Membership Function
Temp['Cold'] = fuzz.trimf(Temp.universe, [16, 16, 22])
Temp['Normal'] = fuzz.trimf(Temp.universe, [18, 23, 28])
Temp['Hot'] = fuzz.trimf(Temp.universe, [24, 30, 30])
Temp.view()
```




```
##Using Trapezoid and Gaussian Membership Function
Temp['Cold'] = fuzz.trapmf(Temp.universe, [16, 16, 20, 22])
Temp['Normal'] = fuzz.gaussmf(Temp.universe, 23, 2)
Temp['Hot'] = fuzz.trapmf(Temp.universe, [24, 26, 30, 30])
Temp.view()
```



Conclusion:

Pre-Lab : Understood about basic set operations

PS1 : Implemented various operations on fuzzy sets and understood the difference in the normal sets and fuzzy sets

PS2 : Designed standard membership functions for the given problem statements

0s completed at 9:53 PM