NAME : Krish Sukhani

Class : TE IT

Batch : D

UID : 2018140059

Soft Compution Pre 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

In [1]:
```
set1 = set()
```

In [2]:
```
set2 = set()
```

In [3]:
```
#case1 : set1_n = 4
#case2 : set1_n = 4
```

In [4]:
```
set1_n = int(input("Enter the number of elements in set 1 : "))
```

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

In [5]:
```
#case1 : set1_n = 3
#case2 : set1_n = 2
```

In [6]:
```
set2_n = int(input("Enter the number of elements in set 2 : "))
```

```
Enter the number of elements in set 2 : 3
```

In [7]:
```
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
```

In [8]:
```
for _ in range(set2_n):
  set2.add(input("Enter the element value in set2 : "))
```

```
Enter the element value in set2 : 5
Enter the element value in set2 : 6
Enter the element value in set2 : 7
```

In [9]:
```
#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '2', '3', '4'}
```

In [10]:
```
set1
```

Out[10]:
```
{'1', '2', '3', '4'}
```

In [11]:
```
#case1 : set1 = {'5', '6', '7'}
#case2 : set1_n = {'2', '5'}
```

In [12]:
```
set2
```

Out[12]:
```
{'5', '6', '7'}
```

UNION

In [13]:
```
union = set1.union(set2)
```

In [14]:
```
#case1 : set1 = {'1', '2', '3', '4', '5', '6', '7'}
#case2 : set1_n = {'1', '2', '3', '4', '5'}
```

In [15]:
```
union
```

Out[15]:
```
{'1', '2', '3', '4', '5', '6', '7'}
```

By code

In [16]:
```
set3 = set()
```

In [17]:
```
for element in set1:
  set3.add(element)
```

In [18]:
```
for element in set2:
  set3.add(element)
```

In [19]:
```
#case1 : set1 = {'1', '2', '3', '4', '5', '6', '7'}
#case2 : set1_n = {'1', '2', '3', '4', '5'}
```

In [20]:
```
set3
```

Out[20]:
```
{'1', '2', '3', '4', '5', '6', '7'}
```

INTERSECTION

In [21]:
```
intersection = set1.intersection(set2)
```

In [22]:
```
#case1 : set1 = {}
#case2 : set1_n = {'2'}
```

In [23]:
```
intersection
```

Out[23]:
```
set()
```

By code

In [24]:
```
int_by_code = set()
```

In [25]:
```
for  element in set1:
  for element1 in set2:
    if element == element1:
      int_by_code.add(element)
```

In [26]:
```
#case1 : set1 = {}
#case2 : set1_n = {'2'}
```

In [27]:
```
int_by_code
```

Out[27]:
```
set()
```

COMPLEMENT

In [28]:
```
complement = set2.difference(set1)
```

In [29]:
```
#case1 : set1 = {'5', '6', '7'}
#case2 : set1_n = {'5'}
```

In [30]:
```
complement
```

Out[30]:
```
{'5', '6', '7'}
```

In [31]:
```
complement2 = set1.difference(set2)
```

In [32]:
```
#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '3', '4'}
```

In [33]:
```
complement2
```

Out[33]:
```
{'1', '2', '3', '4'}
```

By Code

In [34]:
```
#case1 : set1 = {'5', '6', '7'}
#case2 : set1_n = {'5'}
```

In [35]:
```
comp1_bycode = set3 - set1
```

In [36]:
```
comp1_bycode
```

Out[36]:
```
{'5', '6', '7'}
```

In [37]:
```
#case1 : set1 = {'1', '2', '3', '4'}
#case2 : set1_n = {'1', '3', '4'}
```

In [38]:
```
comp2_bycode = set3 - set2
```

In [39]:
```
comp2_bycode
```

Out[39]:
```
{'1', '2', '3', '4'}
```

In [39]:

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.

In [40]:
```
A = {'2': 0.3, '3': 0.5}
```

In [41]:
```
B = {'1': 0.2, '2': 0.3, '3': 0.6, '4': 0.8}
```

In [42]:
```
universe = ['1', '2', '3', '4']
```

In [43]:
```
for i in universe:
  if i not in A.keys():
    A[i] = 0
```

In [44]:
```
A
```

Out[44]:
```
{'1': 0, '2': 0.3, '3': 0.5, '4': 0}
```

In [45]:
```
for i in universe:
  if i not in B.keys():
    B[i] = 0
```

In [46]:
```
B
```

Out[46]:
```
{'1': 0.2, '2': 0.3, '3': 0.6, '4': 0.8}
```

In [78]:
```
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)
```

In [79]:
```
union(A,B)
```

Out[79]:
```
{'1': 0.2, '2': 0.3, '3': 0.6, '4': 0.8}
```

In [80]:
```
def intersection(A,B):
  inter = {}
  for i in A:
    if i in B:
      inter[i]=min(A[i],B[i])
  return inter
```

In [81]:
```
intersection(A,B)
```

Out[81]:
```
{'1': 0, '2': 0.3, '3': 0.5, '4': 0}
```

In [83]:
```
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
```

In [84]:
```
complement(A)
```

Out[84]:
```
{'1': 1, '2': 0.7, '3': 0.5, '4': 1}
```

In [85]:
```
complement(B)
```

Out[85]:
```
{'1': 0.8, '2': 0.7, '3': 0.4, '4': 0.2}
```

In [86]:
```
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')
```

In [87]:
```
one_sub_two(A,B)
```

```
Yes
```

In [88]:
```
one_sub_two(B,A)
```

```
No
```

DEMorgan Laws

In [89]:
```
complement(intersection(A,B)) == union(complement(A),complement(B))
```

Out[89]:
```
True
```

In [90]:
```
complement(union(A,B)) == intersection(complement(A),complement(B))
```

Out[90]:
```
True
```

In [ ]:
```

```

PS2

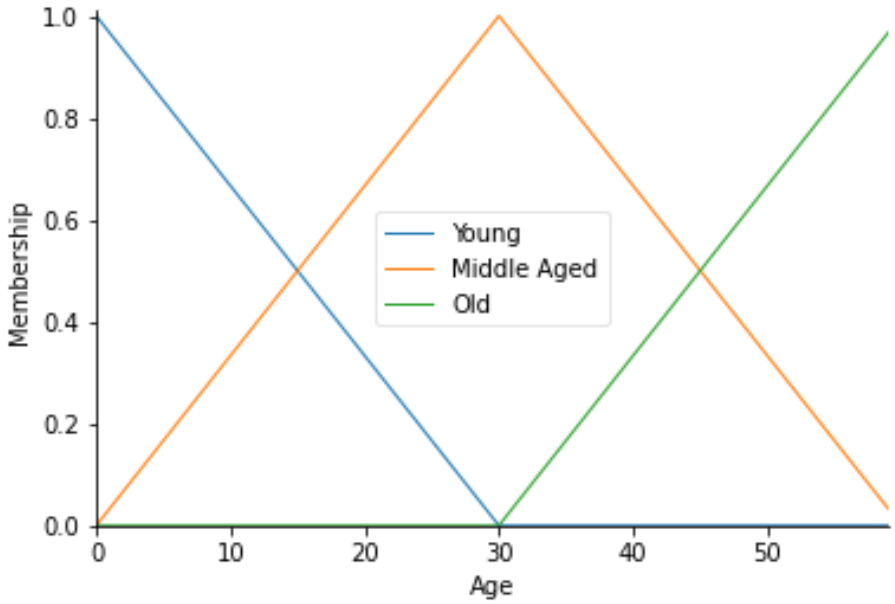
Design a fuzzy library for representing the standard membership functions of 1D Fuzzy Sets for Continuous Universe of Discourse

In [49]:
```
!pip install -U scikit-fuzzy
```

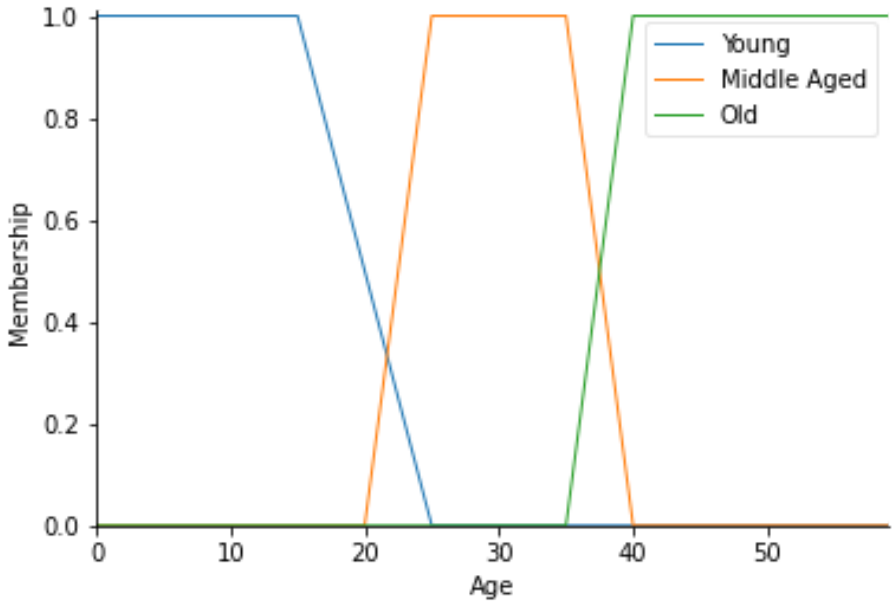
```
Collecting scikit-fuzzy
  Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
     |████████████████████████████████| 993 kB 4.1 MB/s
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from scik
it-fuzzy) (1.19.5)
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.7/dist-packages (from scik
it-fuzzy) (1.4.1)
Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from s
cikit-fuzzy) (2.6.2)
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-any.whl size=894089 sha25
6=0fc13fbfd9b1a1a21e18ef5700e84246b211e3ef79acf5baa4f040b30dac9f8c
  Stored in directory: /root/.cache/pip/wheels/d5/74/fc/38588a3d2e3f34f74588e6daa3aa5b0a322bd6f9
420a707131
Successfully built scikit-fuzzy
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.4.2
```

In [50]:
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

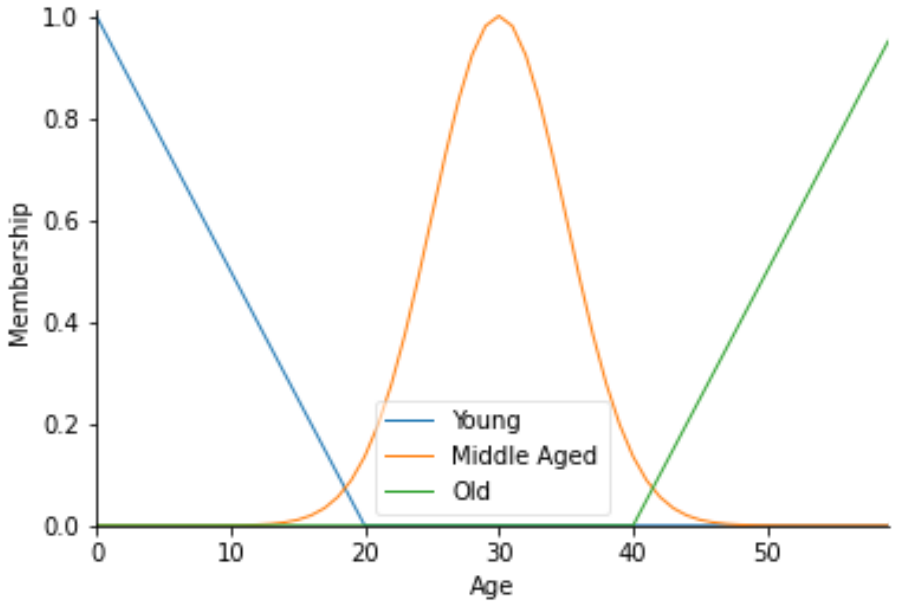
In [91]:
```
# INPUTS
Age= ctrl.Antecedent(np.arange(0,60, 1), 'Age')
##Using Triangular Membership Function
Age['Young'] = fuzz.trimf(Age.universe, [0, 0, 30])
Age['Middle Aged'] = fuzz.trimf(Age.universe, [0, 30, 60])
Age['Old'] = fuzz.trimf(Age.universe, [30,60,60])
Age.view()
```



In [95]:
```
##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()
```

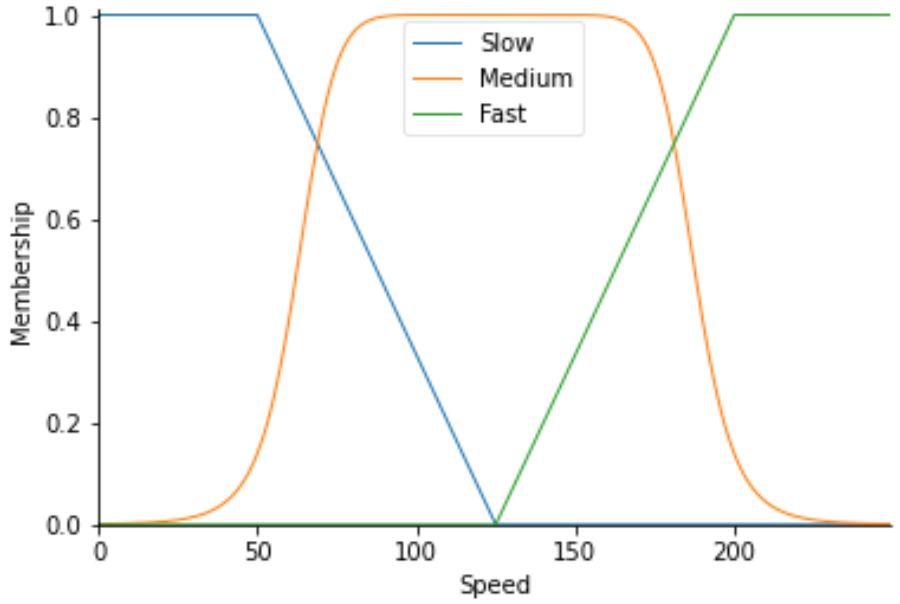


In [96]:
```
##Using Triangular and Gaussian Membership Function
Age['Young'] = fuzz.trimf(Age.universe, [0, 0, 20])
Age['Middle Aged'] = fuzz.gaussmf(Age.universe,30,5)
Age['Old'] = fuzz.trimf(Age.universe, [40,60,60])
Age.view()
```

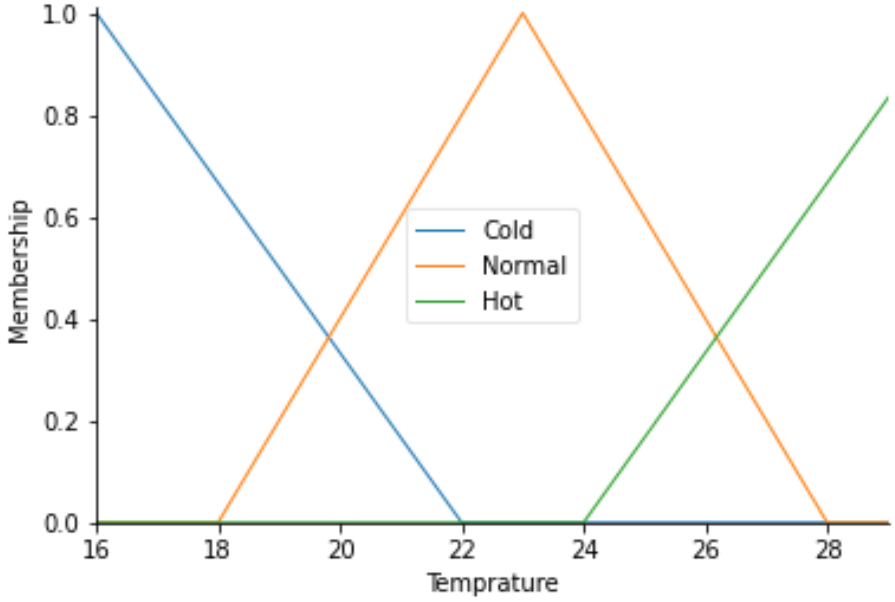


In [97]:

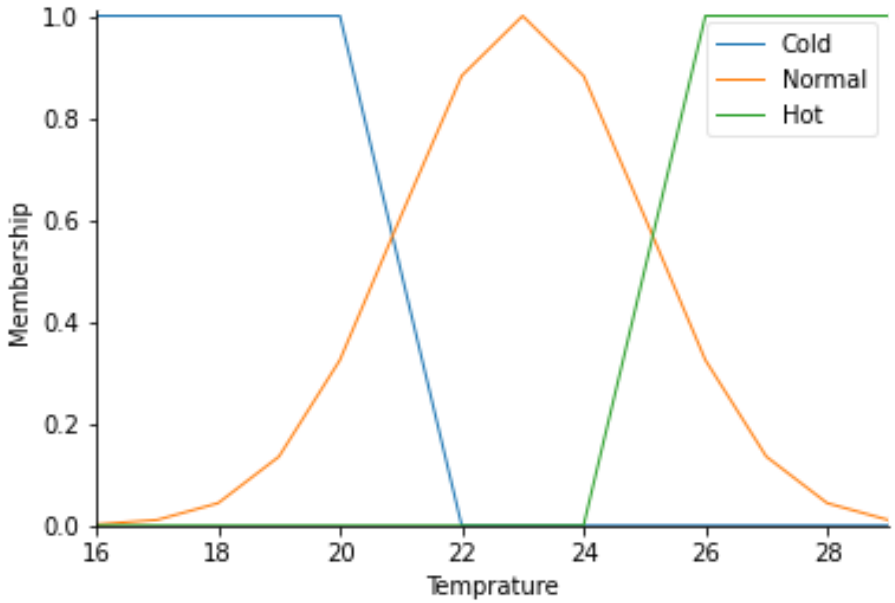
```
Speed= ctrl.Antecedent(np.arange(0,250, 1), 'Speed')
##Using Trapezoid and Generalized Bell Membership Function
Speed['Slow'] = fuzz.trapmf(Speed.universe, [0,0, 50, 125])
Speed['Medium'] = fuzz.gbellmf(Speed.universe,62.5,5,125 )
Speed['Fast'] = fuzz.trapmf(Speed.universe, [125, 200,250, 250])
Speed.view()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



In [98]:
```
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()
```



In [99]:
```
##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

In [99]: