```
#fuzzy sets operations
'''
Implement Union, Intersection, Complement and Difference operations on fuzzy sets.
product of any two fuzzy sets and perform max-min composition on any two fuzzy rel
'''
#Here we define two sets young(x) and middleAged(x)

def student(x):
    if x<=20:
        return 1
    elif 20<x<=30:
        return (30-x)/(30-20)
    elif x>30:
        return 0



def doJob(x):
    if x>30:
        return 1
    elif x<=20:
        return 0
    elif 20<x<=30:
        return (x-20)/(10)




class Fuzzy:
    def __init__(self,item,membership):
        self.val = item
        self.membership = membership

    def return_membership(self):
        return self.membership





import random

x1 = random.sample(range(15, 40), 20)
x2 = x1

set1 = [] #student
set2 = [] #doJob



print(x1)
print(x2)
```

```
[15, 25, 26, 39, 30, 21, 37, 32, 38, 17, 16, 36, 28, 20, 27, 31, 29, 35, 19,
[15, 25, 26, 39, 30, 21, 37, 32, 38, 17, 16, 36, 28, 20, 27, 31, 29, 35, 19,
```

```
    for member in x1:
        val = student(member)
        obj = Fuzzy(member,val)
        set1.append(obj)

    for member in x2:
        val = doJob(member)
        obj = Fuzzy(member,val)
        set2.append(obj)




    for obj in set1:
        print(obj.val,':',obj.membership)

    print('=================================')

    for obj in set2:
        print(obj.val,':',obj.membership)
```

```
15 : 1
25 : 0.5
26 : 0.4
39 : 0
30 : 0.0
21 : 0.9
37 : 0
32 : 0
38 : 0
17 : 1
16 : 1
36 : 0
28 : 0.2
20 : 1
27 : 0.3
31 : 0
29 : 0.1
35 : 0
19 : 1
22 : 0.8
=================================
15 : 0
25 : 0.5
26 : 0.6
39 : 1
30 : 1.0
21 : 0.1
37 : 1
32 : 1
38 : 1
17 : 0
16 : 0
36 : 1
28 : 0.8
20 : 0
27 : 0.7
```

```
31 : 1
29 : 0.9
35 : 1
19 : 0
22 : 0.2
```

```
student_x = []
student_y = []
for obj in set1:
    student_x.append(obj.val)
    student_y.append(obj.membership)
job_x=[]
job_y=[]
for obj in set2:
    job_x.append(obj.val)
    job_y.append(obj.membership)

print(student_x,student_y,job_x,job_y)
```

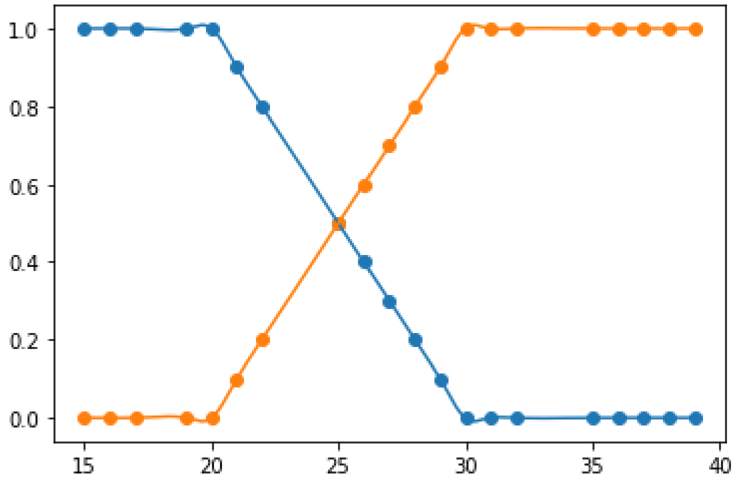
```
[15, 25, 26, 39, 30, 21, 37, 32, 38, 17, 16, 36, 28, 20, 27, 31, 29, 35, 19,
```

```
import numpy as np
import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)

#========================================

x1=np.array(job_x)
y1=np.array(job_y)

x_new1 = np.linspace(x1.min(), x1.max(),500)

f1 = interp1d(x1, y1, kind='quadratic')
y_smooth1=f1(x_new1)
```

```
plt.plot (x_new,y_smooth)
plt.scatter (x, y)
plt.plot (x_new1,y_smooth1)
plt.scatter (x1, y1)
```

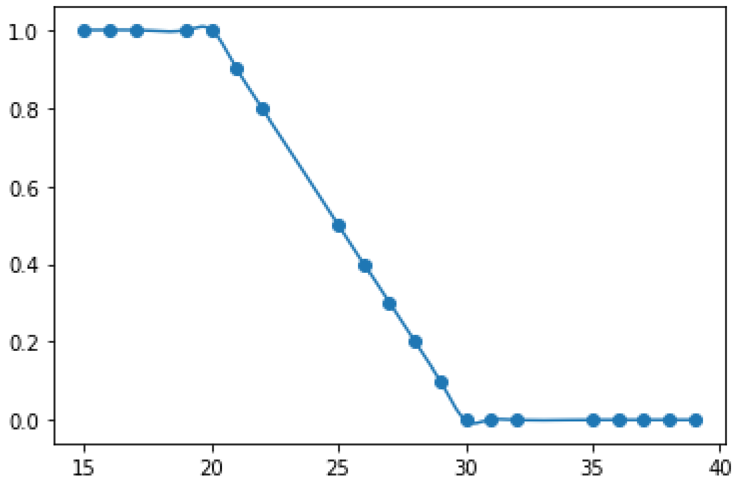
```
plt.scatter (x1, y1)
```

<matplotlib.collections.PathCollection at 0x7f69fe047e10>




```
#complement
#on set 1
import numpy as np
import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)

plt.plot (x_new,y_smooth)
plt.scatter (x, y)
```

<matplotlib.collections.PathCollection at 0x7f69fdfcdd90>



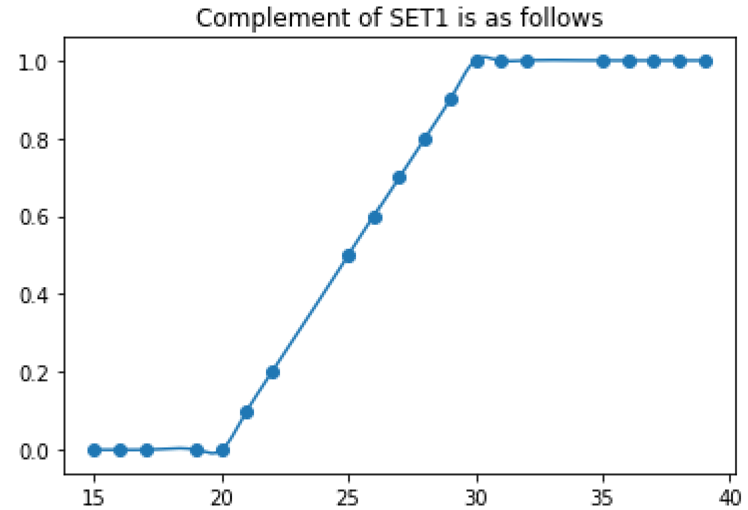

```
NOTset1 = []
for obj in set1:
```

```
for obj in set1:
    new_obj = Fuzzy(obj.val,1- obj.membership)
    NOTset1.append(new_obj)
```

```
student_x1 = []
student_y1 = []
for obj in NOTset1:
    student_x1.append(obj.val)
    student_y1.append(obj.membership)
```

```
x_compli=np.array(student_x1)
y_compli=np.array(student_y1)

x_new_compli = np.linspace(x_compli.min(), x_compli.max(),500)

f2 = interp1d(x_compli, y_compli, kind='quadratic')
y_smooth_compli=f2(x_new_compli)

plt.plot (x_new_compli,y_smooth_compli)
plt.title('Complement of SET1 is as follows')
plt.scatter (x_compli, y_compli)
```

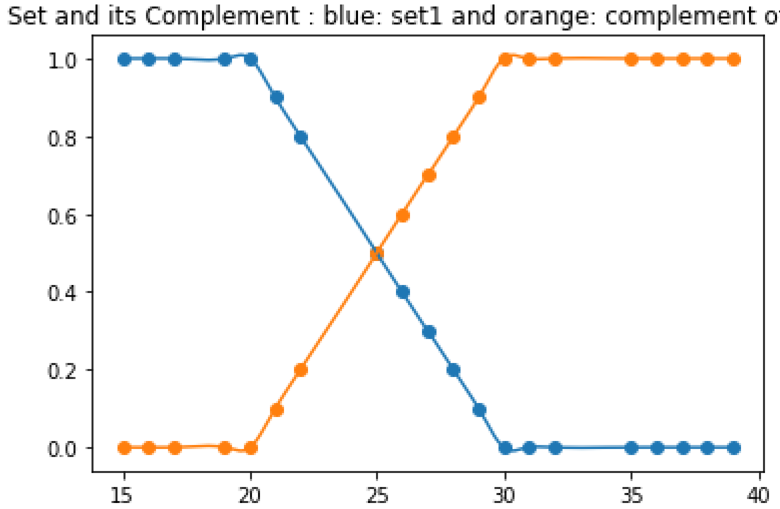
```
<matplotlib.collections.PathCollection at 0x7f69fdb32f90>
```




```
x=np.array(student_x)
y=np.array(student_y)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)

plt.plot (x_new,y_smooth)
plt.scatter (x, y)
```

```
x_compli=np.array(student_x1)
y_compli=np.array(student_y1)

x_new_compli = np.linspace(x_compli.min(), x_compli.max(),500)

f2 = interp1d(x_compli, y_compli, kind='quadratic')
y_smooth_compli=f2(x_new_compli)
plt.title('Set and its Complement : blue: set1 and orange: complement of set1')
plt.plot (x_new_compli,y_smooth_compli)
plt.scatter (x_compli, y_compli)
```

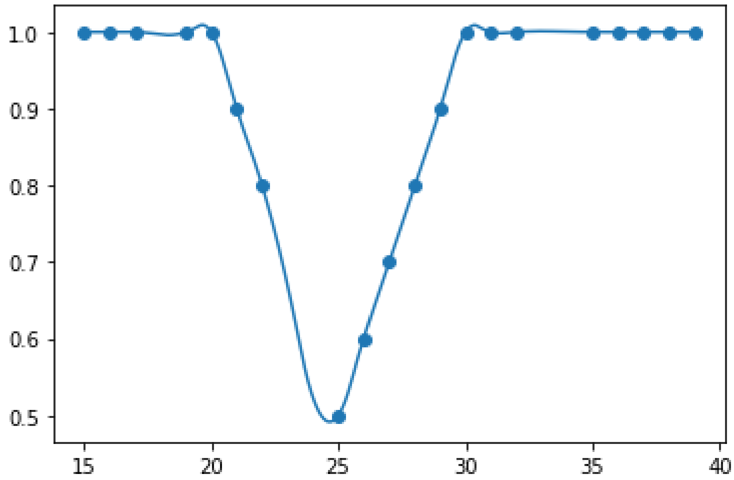
```
<matplotlib.collections.PathCollection at 0x7f69fda2aa10>
```




```
#union
import math
union = []
for obj1,obj2 in zip(set1,set2):
    new_obj = Fuzzy(obj1.val,max(obj1.membership,obj2.membership))
    union.append(new_obj)




Ux = []
Uy = []
for obj in union:
    Ux.append(obj.val)
    Uy.append(obj.membership)



x=np.array(Ux)
y=np.array(Uy)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)
```

```
plt.plot (x_new,y_smooth)
plt.scatter (x, y)
```

```
<matplotlib.collections.PathCollection at 0x7f69fda36450>
```

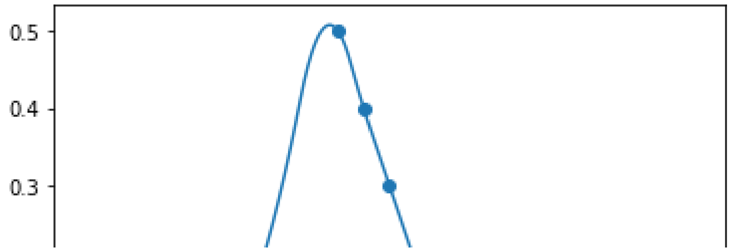



```
#intersection
intersection = []
for obj1,obj2 in zip(set1,set2):
    new_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership))
    intersection.append(new_obj)
```

```
Ix = []
Iy = []
for obj in intersection:
    Ix.append(obj.val)
    Iy.append(obj.membership)
```

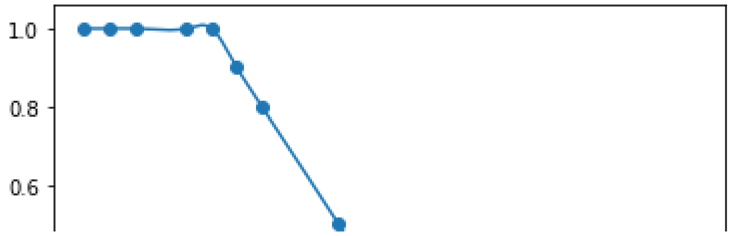
```
x=np.array(Ix)
y=np.array(Iy)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)


plt.plot (x_new,y_smooth)
plt.scatter (x, y)
```

```
<matplotlib.collections.PathCollection at 0x7f69fd9a7190>
```




```
#difference
#difference of 2 sets F1 and F2 is
# F1 - F2 = F1 intersect |-F2|


# set2 complement
NOTset2 = []
for obj in set2:
    new_obj = Fuzzy(obj.val,1- obj.membership)
    NOTset2.append(new_obj)



intersection1 = []
for obj1,obj2 in zip(set1,NOTset2):
    new_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership))
    intersection1.append(new_obj)



Ix1 = []
Iy1 = []
for obj in intersection1:
    Ix1.append(obj.val)
    Iy1.append(obj.membership)


x=np.array(Ix1)
y=np.array(Iy1)

x_new = np.linspace(x.min(), x.max(),500)

f = interp1d(x, y, kind='quadratic')
y_smooth=f(x_new)


plt.plot (x_new,y_smooth)
plt.scatter (x, y)
```

```
<matplotlib.collections.PathCollection at 0x7f69fd912910>
```




```
# MIN-MAX composition of 2 fuzzy relatons

'''
Let 2 relations be R and S and given as follows
'''

R = [
    [0.62,0.45,0.4],
    [0.12,0.1,0,34],
    [0.43,0.27,0.9]
]


S = [
    [0.11,0.33,0.98],
    [0.23,0.37,0.74],
    [0.6,0.5,0.19]
]

#min-max composition will be N*N matrix where N is dimension of both R and S
N = 3
min_max = []

for i in range(N):
    List = []
    for j in range(N):
        #ith row and jth column
        I = R[i]
        new = []
        for k in range(N):
            new.append(min(I[k],S[k][j]))
        List.append(max(new))
    min_max.append(List)




print('R : ')
for i in range(N):
    for j in range(N):
        print(R[i][j],' ',end='')
    print()

print('================================')

print('S:')
for i in range(N):
```

```
        for j in range(N):
            print(S[i][j],' ',end=' ')
        print()

    print('=========MIN-MAX COMPOSITION IS: ==========')
    for i in range(N):
        for j in range(N):
            print(min_max[i][j],' ',end='')
        print()
```

```
R :
0.62  0.45  0.4
0.12  0.1  0
0.43  0.27  0.9
================================
S:
0.11   0.33   0.98
0.23   0.37   0.74
0.6   0.5   0.19
=========MIN-MAX COMPOSITION IS: ==========
0.4  0.4  0.62
0.11  0.12  0.12
0.6  0.5  0.43
```

✓ 0s completed at 11:08 AM