Telkom
University

CII2M3-IF-44-INT - INTRODUCTION TO ARTIFICIAL INTELLIGENCE

EVEN SEMESTER SESSION 2021/2022

PROGRAMMING ASSIGNMENT 02- Reasoning

Group No : 5

Section: IF-44-INT

Lecturer Name: SIR EDWARD FERDIAN

Group Member:

NAME	STUDENT ID
MUHAMMAD FADLI RAMADHAN	1301203533
NUR FASIHAH AYUNI BINTI MOHD YAHYA	1301213670

Programming Assignment 2 - Reasoning

1.0 - Question

Give bengkel.xlsx file which contains data of 100 auto mechanics in Bandung with 2 attributes:

Service quality (real number 1-100; with 100 means best quality) and
Price (real number 1-10, with 10 means most expensive).

Develop a Fuzzy Logic-based system to choose 10 best auto mechanics in Bandung.

The system reads the bengkel.xlsx as input file and outputs a file ranking.xlsx consists of 10 best auto-mechanics using their ID and score (defuzzification output).

Points to design and analyze:

- Amount and Linguistic names for each input attribute
- Shape and limit of the input membership function
- Inference rules
- Defuzzification method
- Shape and limit of the output membership function (depending on your Defuzzification method)

2.0 - Problem Description

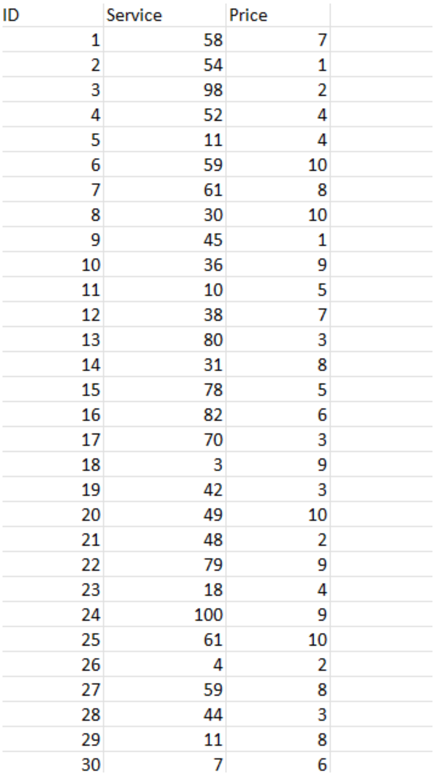
Artificial Intelligence (AI) is classified as machine-generated intelligence. Artificial Intelligence is generated and programmed into a system (computer) to perform a task that humans can perform.

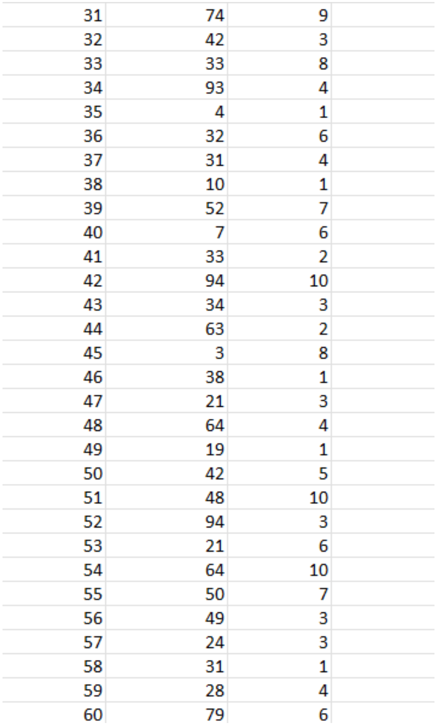
Fuzzy Logic is a logic operation method based on many-valued logic rather than binary (two-valued logic) and it has a similar value to make a computer imitate human intelligence with the hope that what humans do can be done by computers.

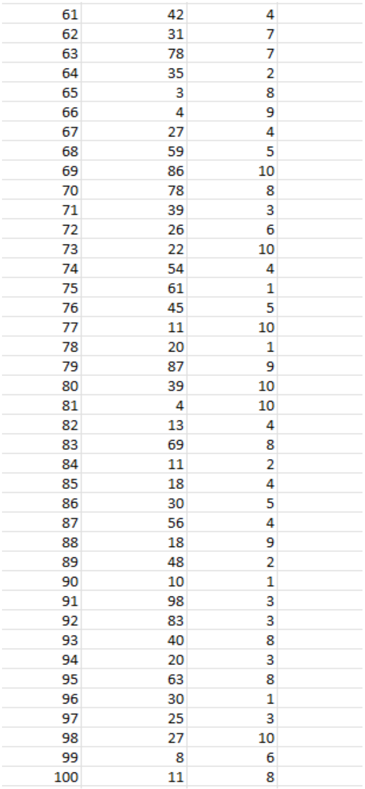
Purpose of this report is to learn and understand more about Fuzzy Logic and its application, understand the Fuzzy Logic linguistic name, understand the membership function of Fuzzy Logic and understand the stages in Fuzzy Logic.

We are given sample data in excel and asked to choose 10 best auto-mechanics using their ID and score. The **service quality** value given in range [0-100] and the **price** value given in range [0-10]. The determination to choose 10 best auto-mechanics done by using fuzzy logic method

Sample Data







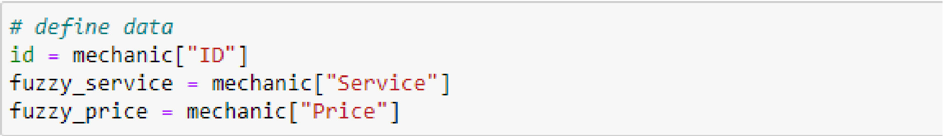
ID	Service	Price
1	58	7
2	54	1
3	98	2
4	52	4
5	11	4
6	59	10
7	61	8
8	30	10
9	45	1
10	36	9
11	10	5
12	38	7
13	80	3
14	31	8
15	78	5
16	82	6
17	70	3
18	3	9
19	42	3
20	49	10
21	48	2
22	79	9
23	18	4
24	100	9
25	61	10
26	4	2
27	59	8
28	44	3
29	11	8
30	7	6
31	74	9
32	42	3
33	33	8
34	93	4
35	4	1
36	32	6
37	31	4
38	10	1
39	52	7
40	7	6
41	33	2
42	94	10
43	34	3
44	63	2
45	3	8
46	38	1
47	21	3
48	64	4
49	19	1
50	42	5
51	48	10
52	94	3
53	21	6
54	64	10
55	50	7
56	49	3
57	24	3
58	31	1
59	28	4
60	79	6
61	42	4
62	31	7
63	78	7
64	35	2
65	3	8
66	4	9
67	27	4
68	59	5
69	86	10
70	78	8
71	39	3
72	26	6
73	22	10
74	54	4
75	61	1
76	45	5
77	11	10
78	20	1
79	87	9
80	39	10
81	4	10
82	13	4
83	69	8
84	11	2
85	18	4
86	30	5
87	56	4
88	18	9
89	48	2
90	10	1
91	98	3
92	83	3
93	40	8
94	20	3
95	63	8
96	30	1
97	25	3
98	27	10
99	8	6
100	11	8

3.0 Amount and Linguistic names for each input attribute

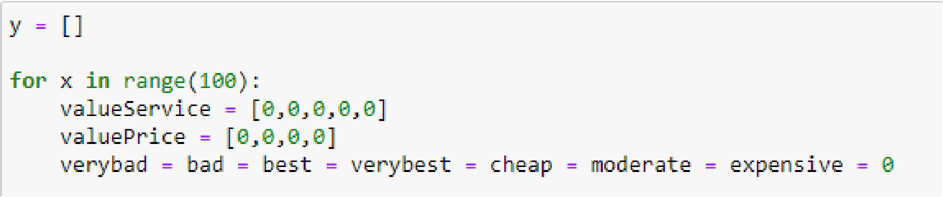
In determining the linguistic variable it depends on the level of difference. If we use more variables, then we have higher accuracy. There are 4 linguistic names for the input to this fuzzy logic. For **service quality** fuzzy variables including **very bad, bad, best and very best**. Meanwhile for the **price** fuzzy variable there are 3 linguistic names which are **cheap, moderate , expensive.** As for this fuzzy output, the variable name of fuzzy is ranked with 3 linguistic names which are **not recommended, recommended and very recommended**.

Variable	Fuzzy Variable	Domain	Explanation
Input	Service Quality	0-100	Value
	Price	0-10	Value
Output	Ranking	0-100	%

Source Code Amount and Linguistic names for each input attributes



```
# define data
id = mechanic["ID"]
fuzzy_service = mechanic["Service"]
fuzzy_price = mechanic["Price"]
```



```
y = []

for x in range(100):
    valueService = [0,0,0,0,0]
    valuePrice = [0,0,0,0]
    verybad = bad = best = verybest = cheap = moderate = expensive = 0
```

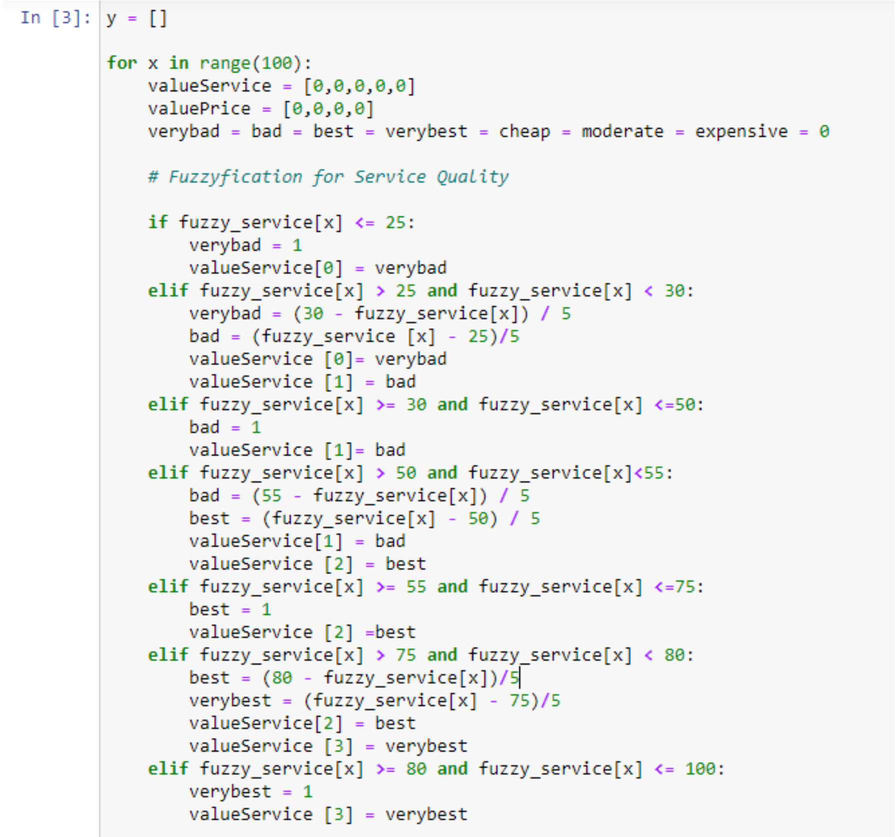
4.0 Shape and limit of the input membership function

The membership function is made in a linear representation, i.e. a surface represented as a straight line. This form is the simplest and becomes a good choice for approaching a concept that is less clear. the shape that is used in this fuzzy, namely trapezium . There are 2 possibilities: the state of a linear fuzzy set. First, the increment of the set starts at a dominant value that has zero membership degree [0] moves to the right towards the value of the domain that has a higher degree of membership.

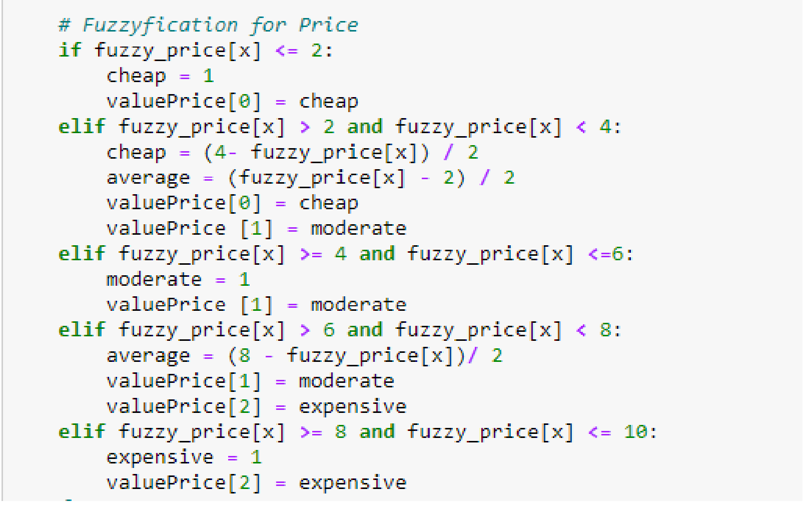
Fuzzy controller which is designed to use two inputs, namely: service and price as well as one output, namely ratings. Based on data in "**bengkel.xlsx**" you can specify a range for each linguistic name service and price as follows:

Variable	Fuzzy Variable	Linguistic name
Input	Service	Very Bad
		Bad
		Best
		Very Best
	Price	Cheap
		Moderate
		Expensive
Output	Ranking	Not Recommended
		Recommended
		Very Recommended

Source Code Shape and Limit of the input membership function

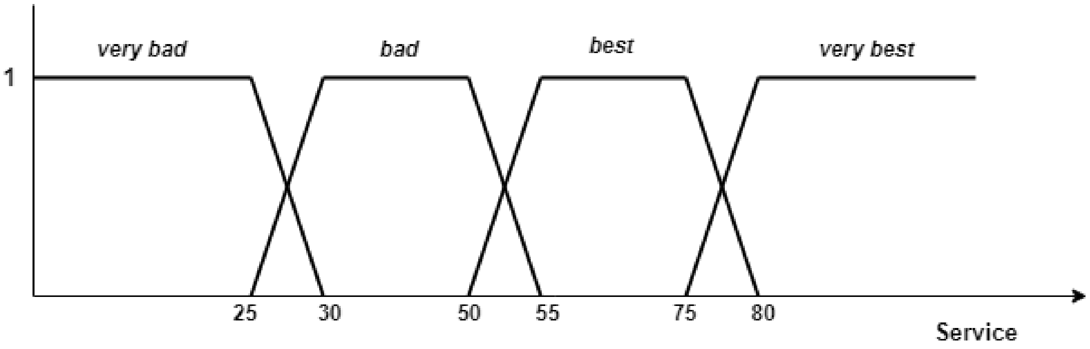


```
In [3]: y = []

        for x in range(100):
            valueService = [0,0,0,0,0]
            valuePrice = [0,0,0,0]
            verybad = bad = best = verybest = cheap = moderate = expensive = 0

            # Fuzzyfication for Service Quality

            if fuzzy_service[x] <= 25:
                verybad = 1
                valueService[0] = verybad
            elif fuzzy_service[x] > 25 and fuzzy_service[x] < 30:
                verybad = (30 - fuzzy_service[x]) / 5
                bad = (fuzzy_service [x] - 25)/5
                valueService [0]= verybad
                valueService [1] = bad
            elif fuzzy_service[x] >= 30 and fuzzy_service[x] <=50:
                bad = 1
                valueService [1]= bad
            elif fuzzy_service[x] > 50 and fuzzy_service[x]<55:
                bad = (55 - fuzzy_service[x]) / 5
                best = (fuzzy_service[x] - 50) / 5
                valueService[1] = bad
                valueService [2] = best
            elif fuzzy_service[x] >= 55 and fuzzy_service[x] <=75:
                best = 1
                valueService [2] =best
            elif fuzzy_service[x] > 75 and fuzzy_service[x] < 80:
                best = (80 - fuzzy_service[x])/5
                verybest = (fuzzy_service[x] - 75)/5
                valueService[2] = best
                valueService [3] = verybest
            elif fuzzy_service[x] >= 80 and fuzzy_service[x] <= 100:
                verybest = 1
                valueService [3] = verybest
```



```
    # Fuzzyfication for Price
    if fuzzy_price[x] <= 2:
        cheap = 1
        valuePrice[0] = cheap
    elif fuzzy_price[x] > 2 and fuzzy_price[x] < 4:
        cheap = (4- fuzzy_price[x]) / 2
        average = (fuzzy_price[x] - 2) / 2
        valuePrice[0] = cheap
        valuePrice [1] = moderate
    elif fuzzy_price[x] >= 4 and fuzzy_price[x] <=6:
        moderate = 1
        valuePrice [1] = moderate
    elif fuzzy_price[x] > 6 and fuzzy_price[x] < 8:
        average = (8 - fuzzy_price[x])/ 2
        valuePrice[1] = moderate
        valuePrice[2] = expensive
    elif fuzzy_price[x] >= 8 and fuzzy_price[x] <= 10:
        expensive = 1
        valuePrice[2] = expensive
```

- **Graph Membership Function of Service**



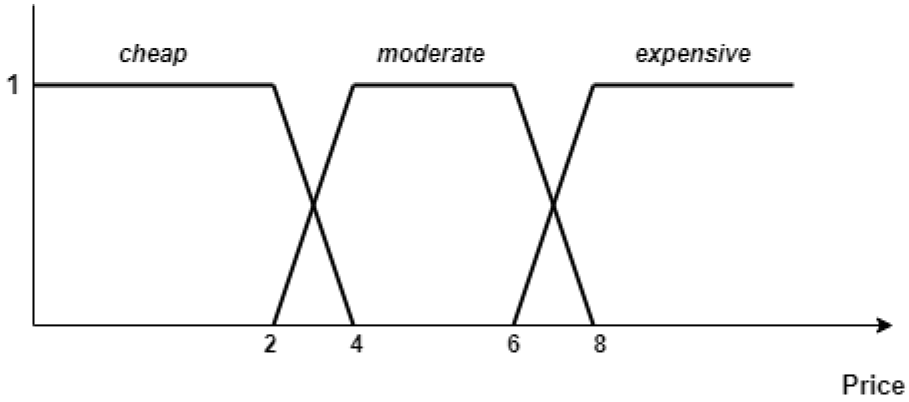
1) Membership Function for Service Quality (Very Best)
   - ➔ The range for Service Quality is [0,100]
   - ➔ We can determine that, Service > 80 is definitely very best, and Service <=75 is best
   - ➔ Therefore, the range [75,80] is the fuzzy area between best and very best

2) Membership Function for Service Quality (Best)
   - ➔ The best Service is considered between 55 and 75
   - ➔ While Service < 50 and Service> 80 is considered not best

3) Membership Function for Service Quality (Bad)
   - ➔ The bad Service is considered between 30 and 50
   - ➔ While Service < 25 and Service > 55 is considered not bad

4) Membership Function for Service Quality (Very Bad)
   - ➔ The range for Service Quality is [0,100]
   - ➔ We can determine that, Service <25 is definitely very bad, and Service >30 is bad
   - ➔ Therefore, the range [25,30] is the fuzzy area between very bad and bad

- **Graph Membership Function of Price**



5) Membership Function for Price(Expensive)
   - ➔ The range for Price is [0,10]
   - ➔ We can determine that, Price > 8 is definitely expensive, and Price <=6 is moderate
   - ➔ Therefore, the range [6,8] is the fuzzy area between moderate and expensive

6) Membership Function for Price (Moderate)
   - ➔ The Moderate Price is considered between 4 and 6
   - ➔ While Price < 2 and Price> 8 is considered not moderate

7) Membership Function for Price (Cheap)
   - ➔ The range for Price is [0,10]
   - ➔ We can determine that, Price <2 is definitely cheap, and Service >4 is moderate
   - ➔ Therefore, the range [2,4] is the fuzzy area between cheap and moderate

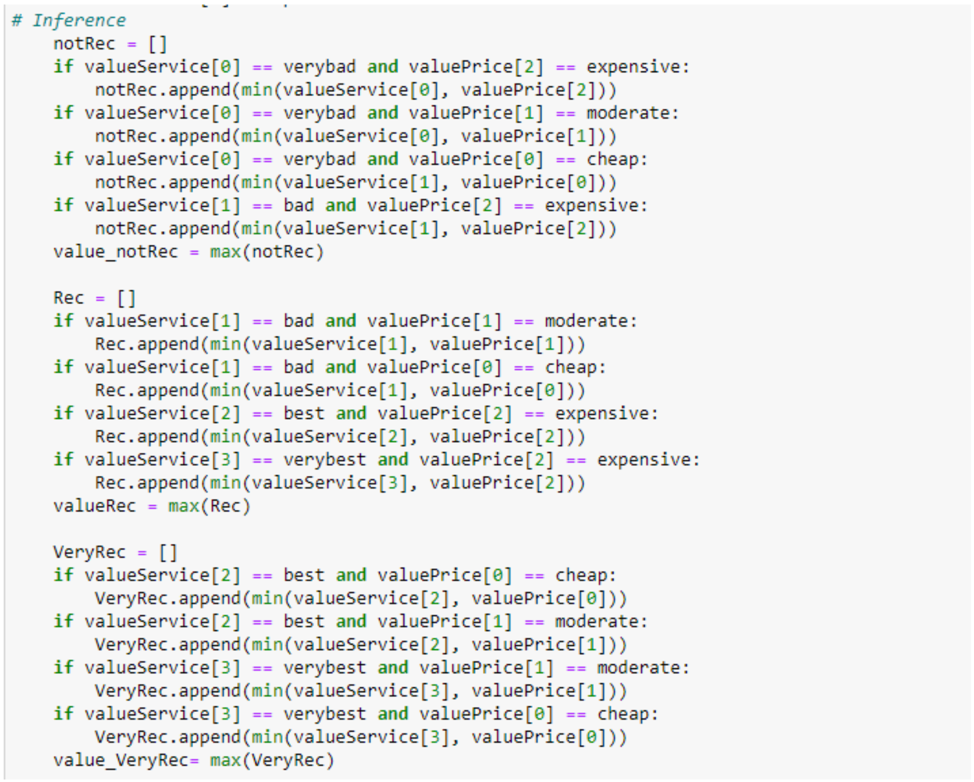
5.0 Inference Rule

Inference rules are created to control the output. This rule will follow logic and heuristic knowledge. The following inference rules are used in this fuzzy as follows:

- If the service is VERY BAD and the price is EXPENSIVE then the ranking is NOT RECOMMENDED
- If the service is VERY BAD and the price is MODERATE then the ranking is NOT RECOMMENDED
- If the service is VERY BAD and the price is CHEAP then the ranking is NOT RECOMMENDED
- If the service is BAD and the price is EXPENSIVE then the ranking is NOT RECOMMENDED
- If the service is BAD and the price is MODERATE then the ranking is RECOMMENDED
- If the service is BAD and the price is CHEAP then the ranking is RECOMMENDED
- If the service is BEST and the price is EXPENSIVE then the ranking is RECOMMENDED
- If the service is BEST and the price is MODERATE then the ranking is VERY RECOMMENDED
- If the service is BEST and the price is CHEAP then the ranking isVERY RECOMMENDED
- If the service is VERY BEST and the price is EXPENSIVE then the ranking is RECOMMENDED
- If the service is VERY BEST and the price is MODERATE then the ranking is VERY RECOMMENDED
- If the service is VERY BEST and the price is CHEAP then the ranking is VERY RECOMMENDED

Service / Price	Cheap	Moderate	Expensive
Very Bad	NR	NR	NR
Bad	R	R	NR
Best	VR	VR	R
Very Best	VR	VR	R

Source Code of Inference Rule



```
# Inference
    notRec = []
    if valueService[0] == verybad and valuePrice[2] == expensive:
        notRec.append(min(valueService[0], valuePrice[2]))
    if valueService[0] == verybad and valuePrice[1] == moderate:
        notRec.append(min(valueService[0], valuePrice[1]))
    if valueService[0] == verybad and valuePrice[0] == cheap:
        notRec.append(min(valueService[1], valuePrice[0]))
    if valueService[1] == bad and valuePrice[2] == expensive:
        notRec.append(min(valueService[1], valuePrice[2]))
    value_notRec = max(notRec)

    Rec = []
    if valueService[1] == bad and valuePrice[1] == moderate:
        Rec.append(min(valueService[1], valuePrice[1]))
    if valueService[1] == bad and valuePrice[0] == cheap:
        Rec.append(min(valueService[1], valuePrice[0]))
    if valueService[2] == best and valuePrice[2] == expensive:
        Rec.append(min(valueService[2], valuePrice[2]))
    if valueService[3] == verybest and valuePrice[2] == expensive:
        Rec.append(min(valueService[3], valuePrice[2]))
    valueRec = max(Rec)

    VeryRec = []
    if valueService[2] == best and valuePrice[0] == cheap:
        VeryRec.append(min(valueService[2], valuePrice[0]))
    if valueService[2] == best and valuePrice[1] == moderate:
        VeryRec.append(min(valueService[2], valuePrice[1]))
    if valueService[3] == verybest and valuePrice[1] == moderate:
        VeryRec.append(min(valueService[3], valuePrice[1]))
    if valueService[3] == verybest and valuePrice[0] == cheap:
        VeryRec.append(min(valueService[3], valuePrice[0]))
    value_VeryRec= max(VeryRec)
```

6.0 Defuzzification method

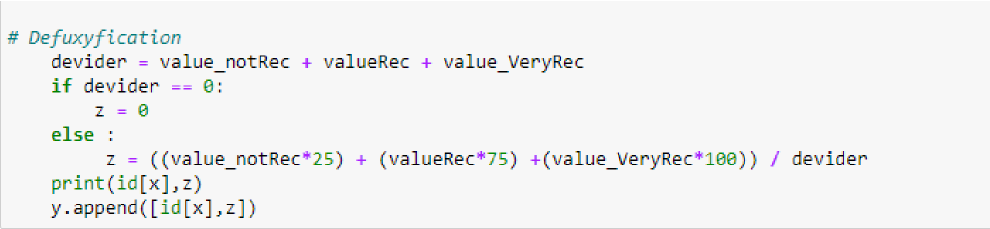
Defuzzification is the process of getting crisp values from a fuzzy set. We are using **Takagi-Sugeno-style** which is suitable for control systems and fast computation time which consumes less time to get the output or result. Why we choose this method is Takagi-Sugeno-style system is more accurate than Mamdani System.The defuzzification process for a Sugeno system is more computationally efficient compared to Mamdani system, since it uses a weighted average or weighted sum of a few data points rather than compute a centroid of a two-dimensional area.

Calculate crisp output using formula :

$$\boxed{z^* = \frac{\sum_{i=1}^{l} \mu B_i . c_i}{\sum_{i=1}^{l} \mu B_i}}$$

$$c_i = constant\ for\ i^{th}\ linguistic$$
$$\mu B_i = membership\ for\ i^{th}\ linguistic$$

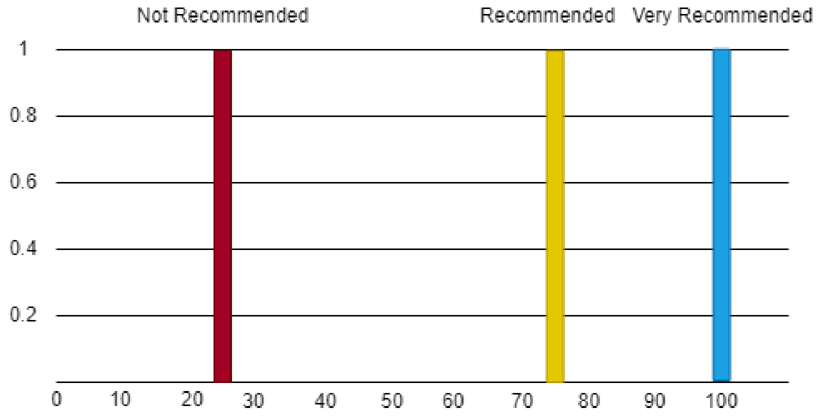
Source Code of Defuzzification Method



```
# Defuxyfication
    devider = value_notRec + valueRec + value_VeryRec
    if devider == 0:
        z = 0
    else :
         z = ((value_notRec*25) + (valueRec*75) +(value_VeryRec*100)) / devider
    print(id[x],z)
    y.append([id[x],z])
```

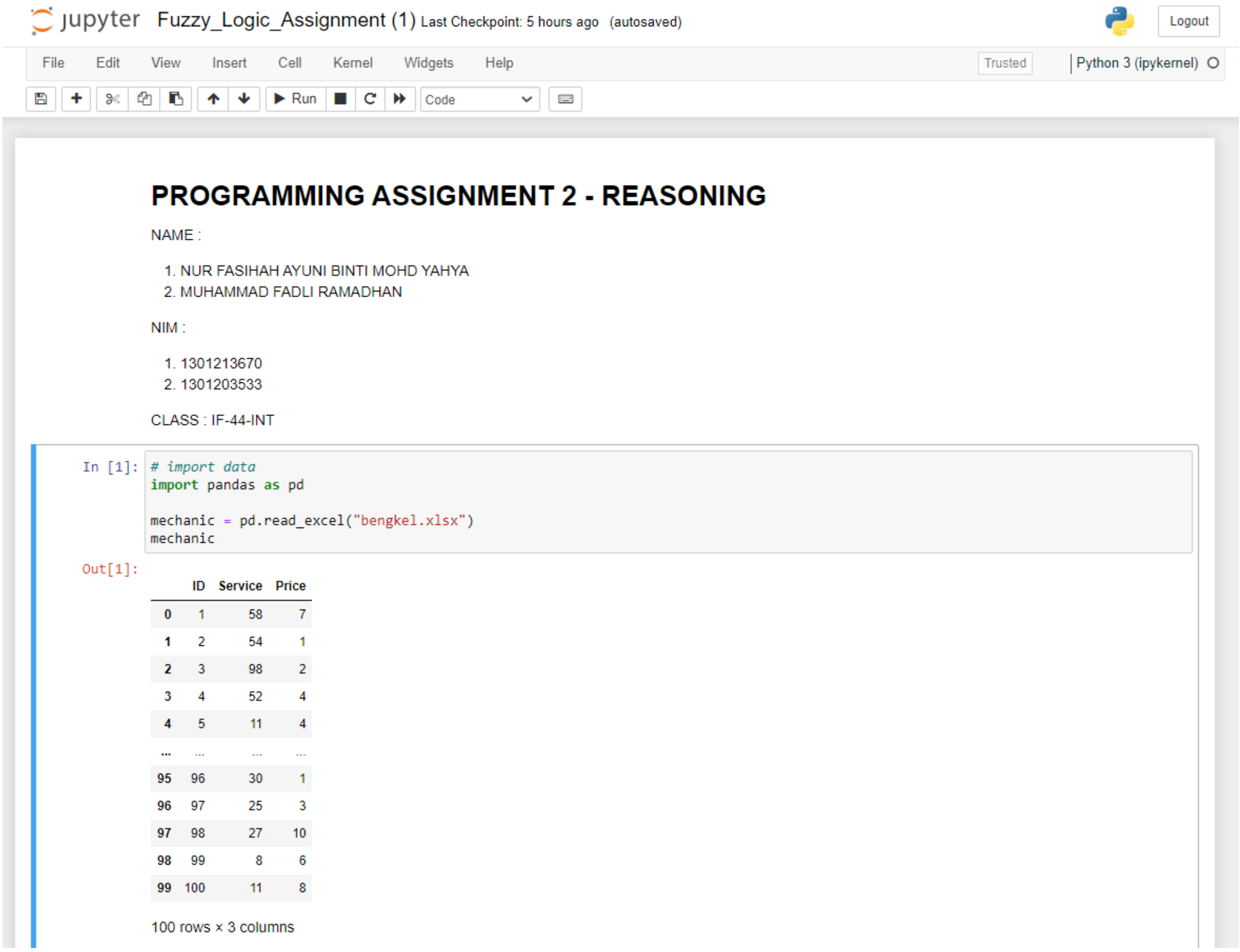
7.0 Shape and limit of the output membership function (depending on your Defuzzification method)

Constant Defuzzification (Takagi-Sugeno- Style). To choose a constant value to represent each output linguistic , we set the constant value to 25, 75, 100



NR= Not Recommended = 25
R = Recommended = 75
VR = Very Recommended = 100

8.0 Screenshot of the coding



PROGRAMMING ASSIGNMENT 2 - REASONING

NAME :

1. NUR FASIHAH AYUNI BINTI MOHD YAHYA
2. MUHAMMAD FADLI RAMADHAN

NIM :

1. 1301213670
2. 1301203533

CLASS : IF-44-INT

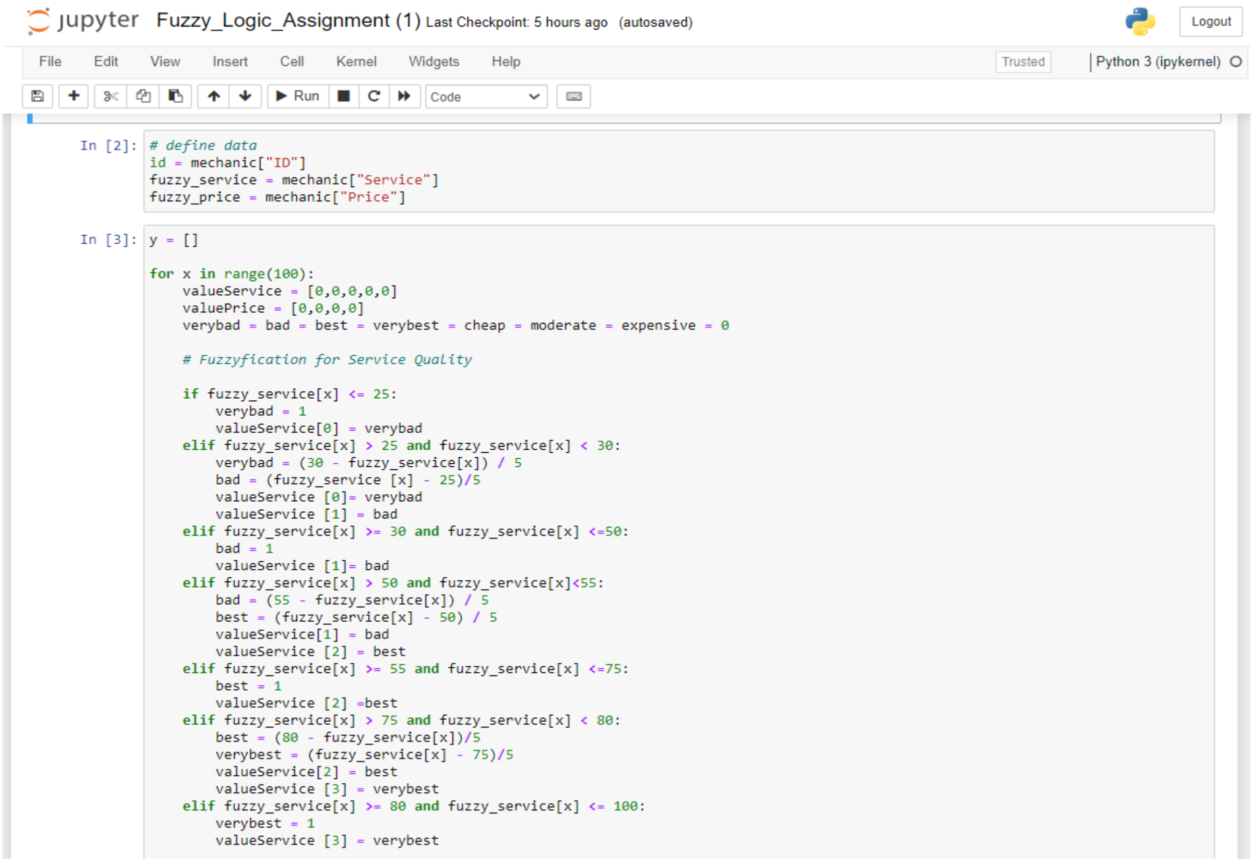
In [1]:

```
# import data
import pandas as pd

mechanic = pd.read_excel("bengkel.xlsx")
mechanic
```

Out[1]:

	ID	Service	Price
0	1	58	7
1	2	54	1
2	3	98	2
3	4	52	4
4	5	11	4
...	...	...	...
95	96	30	1
96	97	25	3
97	98	27	10
98	99	8	6
99	100	11	8

100 rows × 3 columns

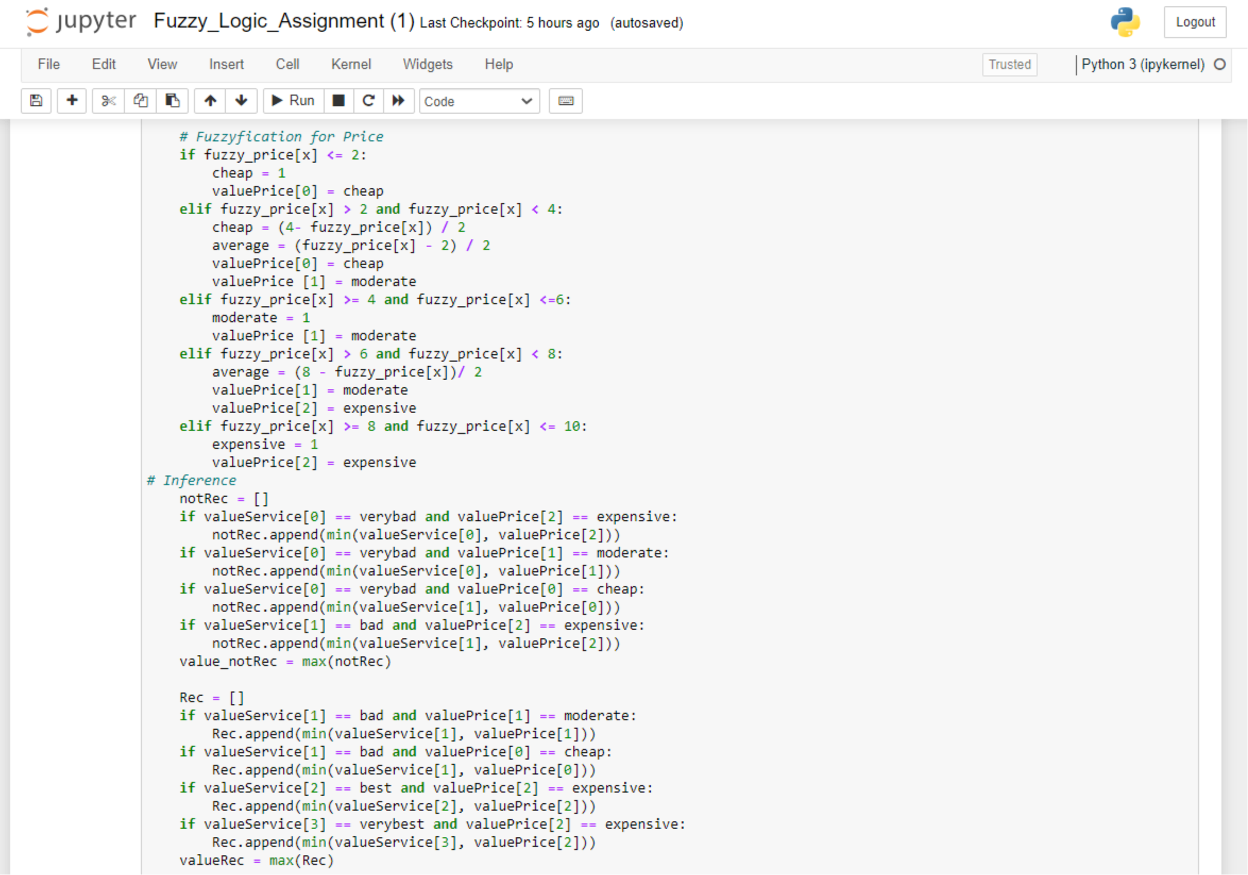


In [2]:

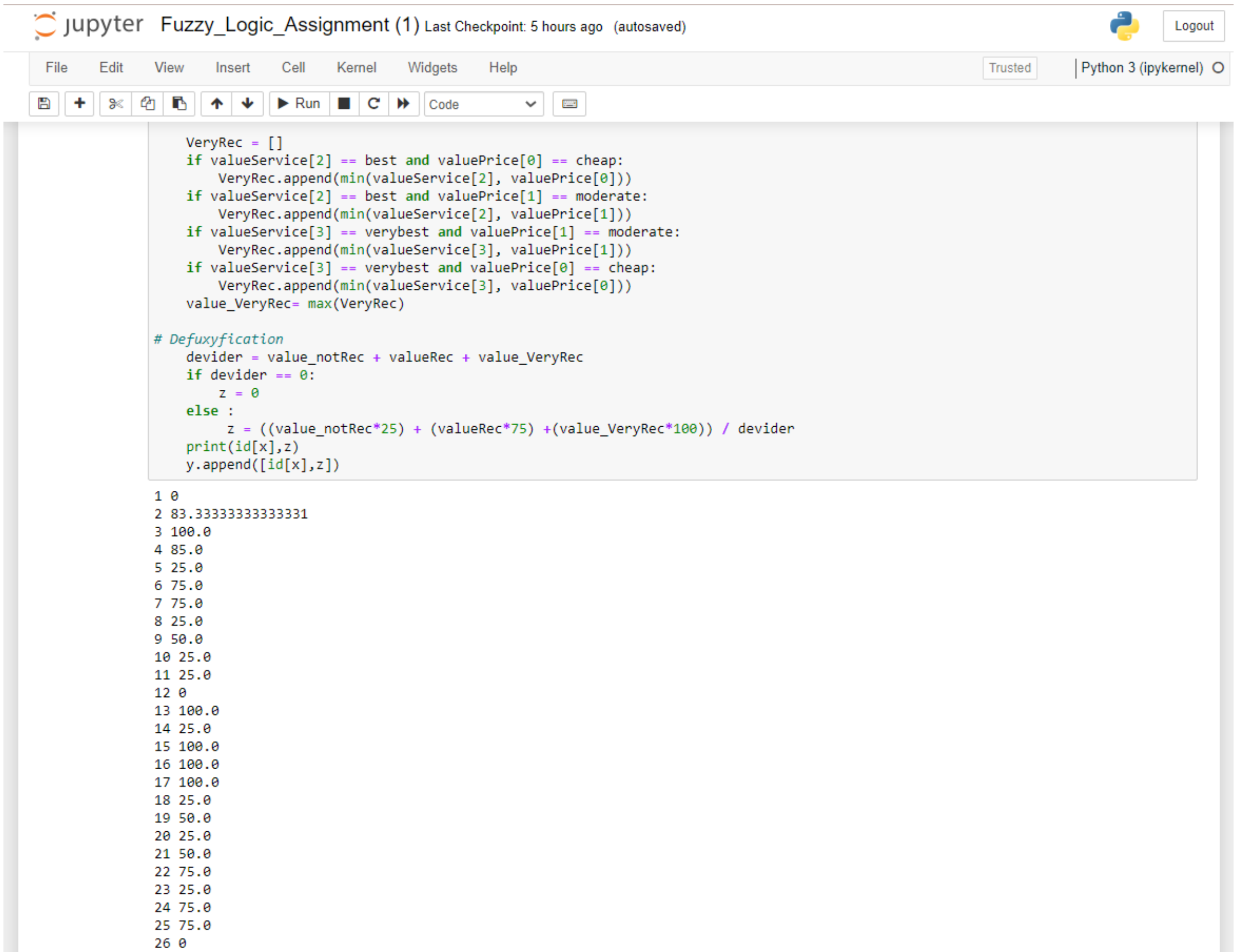
```
# define data
id = mechanic["ID"]
fuzzy_service = mechanic["Service"]
fuzzy_price = mechanic["Price"]
```

In [3]:

```
y = []

for x in range(100):
    valueService = [0,0,0,0,0]
    valuePrice = [0,0,0,0]
    verybad = bad = best = verybest = cheap = moderate = expensive = 0

    # Fuzzyfication for Service Quality

    if fuzzy_service[x] <= 25:
        verybad = 1
        valueService[0] = verybad
    elif fuzzy_service[x] > 25 and fuzzy_service[x] < 30:
        verybad = (30 - fuzzy_service[x]) / 5
        bad = (fuzzy_service [x] - 25)/5
        valueService [0]= verybad
        valueService [1] = bad
    elif fuzzy_service[x] >= 30 and fuzzy_service[x] <=50:
        bad = 1
        valueService [1]= bad
    elif fuzzy_service[x] > 50 and fuzzy_service[x]<55:
        bad = (55 - fuzzy_service[x]) / 5
        best = (fuzzy_service[x] - 50) / 5
        valueService[1] = bad
        valueService [2] = best
    elif fuzzy_service[x] >= 55 and fuzzy_service[x] <=75:
        best = 1
        valueService [2] =best
    elif fuzzy_service[x] > 75 and fuzzy_service[x] < 80:
        best = (80 - fuzzy_service[x])/5
        verybest = (fuzzy_service[x] - 75)/5
        valueService[2] = best
        valueService [3] = verybest
    elif fuzzy_service[x] >= 80 and fuzzy_service[x] <= 100:
        verybest = 1
        valueService [3] = verybest
```

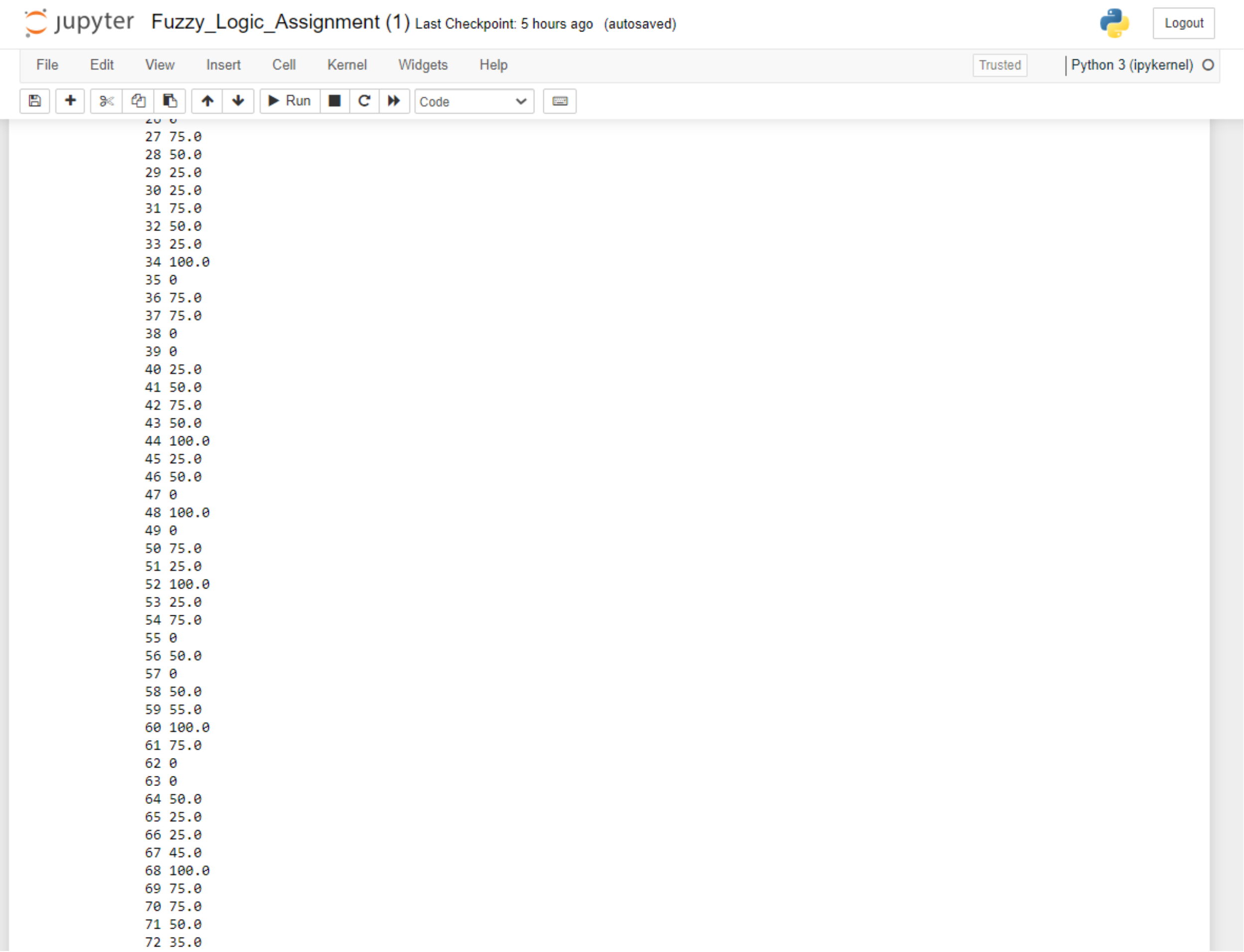


```
    # Fuzzyfication for Price
    if fuzzy_price[x] <= 2:
        cheap = 1
        valuePrice[0] = cheap
    elif fuzzy_price[x] > 2 and fuzzy_price[x] < 4:
        cheap = (4- fuzzy_price[x]) / 2
        average = (fuzzy_price[x] - 2) / 2
        valuePrice[0] = cheap
        valuePrice [1] = moderate
    elif fuzzy_price[x] >= 4 and fuzzy_price[x] <=6:
        moderate = 1
        valuePrice [1] = moderate
    elif fuzzy_price[x] > 6 and fuzzy_price[x] < 8:
        average = (8 - fuzzy_price[x])/ 2
        valuePrice[1] = moderate
        valuePrice[2] = expensive
    elif fuzzy_price[x] >= 8 and fuzzy_price[x] <= 10:
        expensive = 1
        valuePrice[2] = expensive
# Inference
    notRec = []
    if valueService[0] == verybad and valuePrice[2] == expensive:
        notRec.append(min(valueService[0], valuePrice[2]))
    if valueService[0] == verybad and valuePrice[1] == moderate:
        notRec.append(min(valueService[0], valuePrice[1]))
    if valueService[0] == verybad and valuePrice[0] == cheap:
        notRec.append(min(valueService[1], valuePrice[0]))
    if valueService[1] == bad and valuePrice[2] == expensive:
        notRec.append(min(valueService[1], valuePrice[2]))
    value_notRec = max(notRec)

    Rec = []
    if valueService[1] == bad and valuePrice[1] == moderate:
        Rec.append(min(valueService[1], valuePrice[1]))
    if valueService[1] == bad and valuePrice[0] == cheap:
        Rec.append(min(valueService[1], valuePrice[0]))
    if valueService[2] == best and valuePrice[2] == expensive:
        Rec.append(min(valueService[2], valuePrice[2]))
    if valueService[3] == verybest and valuePrice[2] == expensive:
        Rec.append(min(valueService[3], valuePrice[2]))
    valueRec = max(Rec)
```

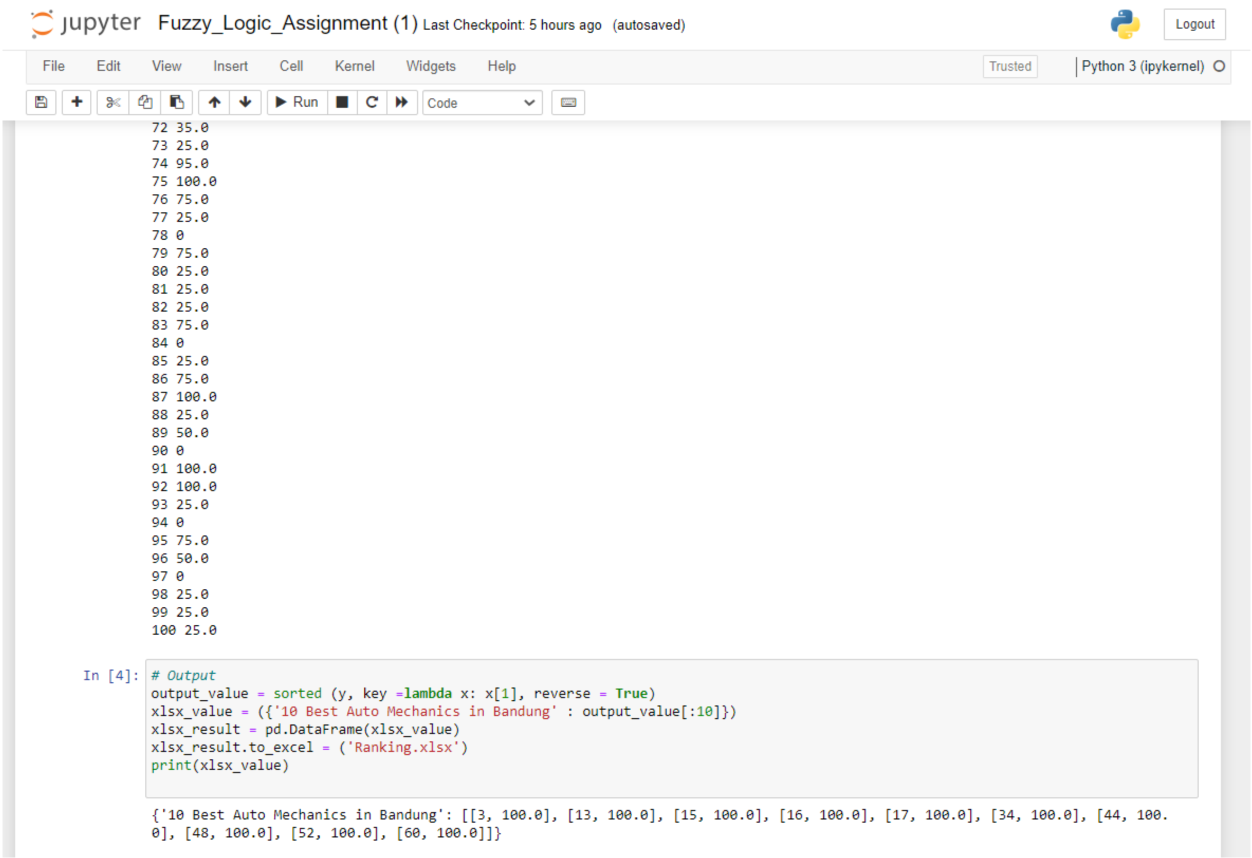


```
    VeryRec = []
    if valueService[2] == best and valuePrice[0] == cheap:
        VeryRec.append(min(valueService[2], valuePrice[0]))
    if valueService[2] == best and valuePrice[1] == moderate:
        VeryRec.append(min(valueService[2], valuePrice[1]))
    if valueService[3] == verybest and valuePrice[1] == moderate:
        VeryRec.append(min(valueService[3], valuePrice[1]))
    if valueService[3] == verybest and valuePrice[0] == cheap:
        VeryRec.append(min(valueService[3], valuePrice[0]))
    value_VeryRec= max(VeryRec)

# Defuxyfication
    devider = value_notRec + valueRec + value_VeryRec
    if devider == 0:
        z = 0
    else :
         z = ((value_notRec*25) + (valueRec*75) +(value_VeryRec*100)) / devider
    print(id[x],z)
    y.append([id[x],z])
```

```
1 0
2 83.33333333333331
3 100.0
4 85.0
5 25.0
6 75.0
7 75.0
8 25.0
9 50.0
10 25.0
11 25.0
12 0
13 100.0
14 25.0
15 100.0
16 100.0
17 100.0
18 25.0
19 50.0
20 25.0
21 50.0
22 75.0
23 25.0
24 75.0
25 75.0
26 0
```



```
26 0
27 75.0
28 50.0
29 25.0
30 25.0
31 75.0
32 50.0
33 25.0
34 100.0
35 0
36 75.0
37 75.0
38 0
39 0
40 25.0
41 50.0
42 75.0
43 50.0
44 100.0
45 25.0
46 50.0
47 0
48 100.0
49 0
50 75.0
51 25.0
52 100.0
53 25.0
54 75.0
55 0
56 50.0
57 0
58 50.0
59 55.0
60 100.0
61 75.0
62 0
63 0
64 50.0
65 25.0
66 25.0
67 45.0
68 100.0
69 75.0
70 75.0
71 50.0
72 35.0
```



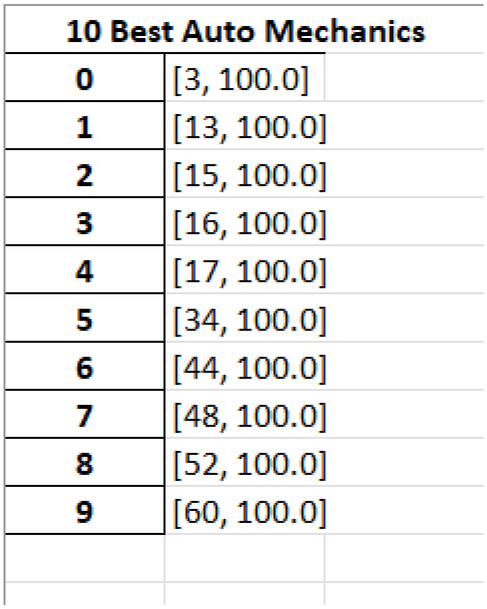
jupyter Fuzzy_Logic_Assignment (1) Last Checkpoint: 5 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
72 35.0
73 25.0
74 95.0
75 100.0
76 75.0
77 25.0
78 0
79 75.0
80 25.0
81 25.0
82 25.0
83 75.0
84 0
85 25.0
86 75.0
87 100.0
88 25.0
89 50.0
90 0
91 100.0
92 100.0
93 25.0
94 0
95 75.0
96 50.0
97 0
98 25.0
99 25.0
100 25.0
```

```
In [4]: # Output
        output_value = sorted (y, key =lambda x: x[1], reverse = True)
        xlsx_value = ({'10 Best Auto Mechanics in Bandung' : output_value[:10]})
        xlsx_result = pd.DataFrame(xlsx_value)
        xlsx_result.to_excel = ('Ranking.xlsx')
        print(xlsx_value)
```

```
{'10 Best Auto Mechanics in Bandung': [[3, 100.0], [13, 100.0], [15, 100.0], [16, 100.0], [17, 100.0], [34, 100.0], [44, 100.
0], [48, 100.0], [52, 100.0], [60, 100.0]]}
```

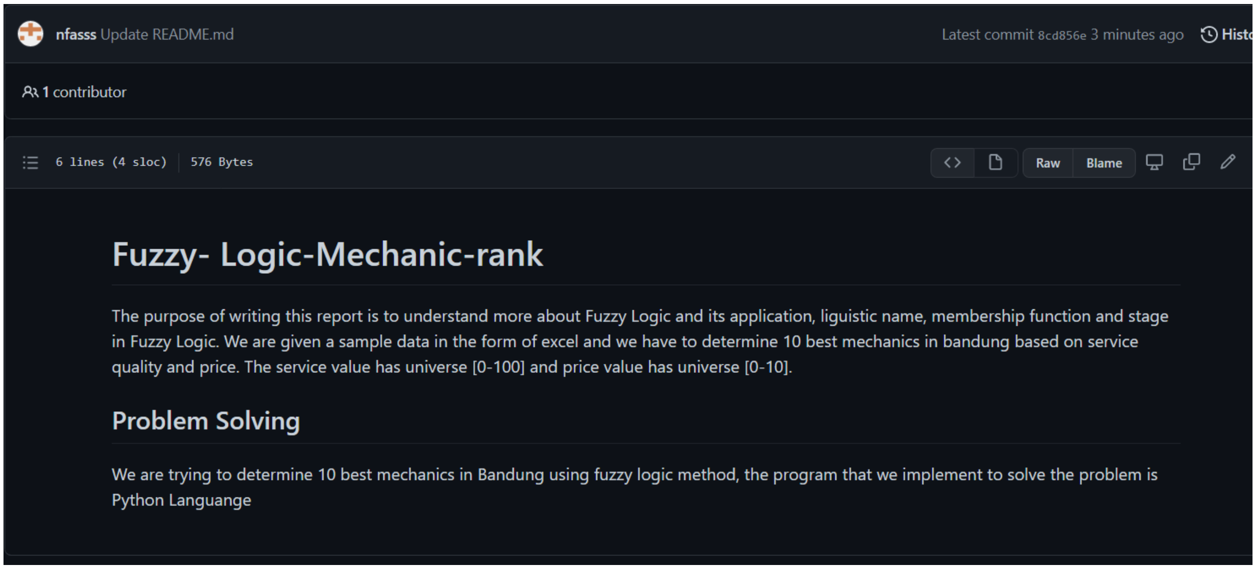
9.0 Output



	10 Best Auto Mechanics
0	[3, 100.0]
1	[13, 100.0]
2	[15, 100.0]
3	[16, 100.0]
4	[17, 100.0]
5	[34, 100.0]
6	[44, 100.0]
7	[48, 100.0]
8	[52, 100.0]
9	[60, 100.0]

10.0 Instruction on how to run the Readme.txt

Click this link to run the Readme.txt file
https://github.com/nfasss/Mechanic-rank/blob/main/README.md



Fuzzy- Logic-Mechanic-rank

The purpose of writing this report is to understand more about Fuzzy Logic and its application, liguistic name, membership function and stage in Fuzzy Logic. We are given a sample data in the form of excel and we have to determine 10 best mechanics in bandung based on service quality and price. The service value has universe [0-100] and price value has universe [0-10].

Problem Solving

We are trying to determine 10 best mechanics in Bandung using fuzzy logic method, the program that we implement to solve the problem is Python Languange

11.0 Presentation video

This is the link for our presentation video

https://youtu.be/ghrmxvJ5RcY

12.0 Task Distribution

Name	Task
MUHAMMAD FADLI RAMADHAN	- Report - Video Presentation
NUR FASIHAH AYUNI BINTI MOHD YAHYA	- Program Source Code - Report - Readme.txt - Video Presentation