**Cho X là không gian nền về kı́ch thước phòng họp căn cứvào số lượng bàn ghế đủ cho các chuyên**

**gia ngồi. Giả sử có 10 phòng với khả năng chứa là một phần tử của tập Y như sau Y =**

**{15,38,48,60,75,88,102,116,140}.**

**a) Xây dựng các tập mờ và hàm thuộc tương ứng (1,5 điểm)**

** A là tập mờ các phòng họp rộng**

** B là tập mờ các phòng có thể chứa 90 chuyên gia**

** C là tập các phòng họp không chứa quá 90 chuyên gia**

**b) Hãy vẽ biểu đồ cho các tập mờ trên (1,5 điểm)**

**A)**

A(x) = { 0 nếu x ≤ 15

(x - 15) / (38 - 15) nếu 15 < x ≤ 38

(60 - x) / (60 - 38) nếu 38 < x ≤ 60

0 nếu x > 60

}

B(x) = { 0 nếu x ≤ 60

(x - 60) / (88 - 60) nếu 60 < x ≤ 88

(102 - x) / (102 - 88) nếu 88 < x ≤ 102

0 nếu x > 102

}

C(x) = { 1 nếu x ≤ 60

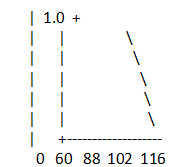
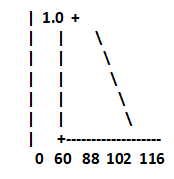
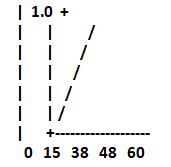
(88 - x) / (88 - 60) nếu 60 < x ≤ 88

(102 - x) / (102 - 88) nếu 88 < x ≤ 102

0 nếu x > 102

}

**B)**



**Cho tập nền X = {x1, x2, x3, x4}, Y = {y1, y2, y3, y4}, tập mờ A**

**A = {(0.3/ x1), (0.5/ x2), (0.7/ x3), (1/ x4)}**

**Và ma trận quan hệ mờ R như sau:**

**R = 0.5 1 0.6 0.4**

**0.5 0.8 0.7 0.5**

**0.4 0.6 0.5 0.4**

**0.2 0.4 0.3 0.1**

**Giả sử giá của sản phẩm được xác định bởi tập mờ B = A ∘ R**

**a) Hãy tập mờ B = A ∘ R (1 điểm)**

**b) Hãy mô phỏng một đoạn code python để tính giá trị B (1 điểm)**

**c) Hãy vẽ biểu đồ cho các tập mờ A, B (1 điểm)**

**a)** Để tính tập mờ B = A ∘ R, chúng ta thực hiện phép nhân các phần tử tương ứng của tập mờ A với ma trận quan hệ mờ R. Kết quả sẽ là tập mờ B. Dưới đây là tập mờ B:

Tập mờ B = {(0.3/ x1) \* 0.5, (0.5/ x2) \* 1, (0.7/ x3) \* 0.6, (1/ x4) \* 0.4}

**B)**

import numpy as np

# Định nghĩa tập mờ A và ma trận quan hệ mờ R

A = np.array([0.3, 0.5, 0.7, 1])

R = np.array([[0.5, 1, 0.6, 0.4],

[0.5, 0.8, 0.7, 0.5],

[0.4, 0.6, 0.5, 0.4],

[0.2, 0.4, 0.3, 0.1]])

# Tính tập mờ B = A ∘ R

B = A \* R

# In kết quả

print("Tập mờ B:")

print(B)

**C)**

import matplotlib.pyplot as plt

# Định nghĩa tên của các phần tử trong tập X

X = ['x1', 'x2', 'x3', 'x4']

# Giá trị của tập mờ A và B

A\_values = [0.3, 0.5, 0.7, 1]

B\_values = B.tolist()

# Vẽ biểu đồ

plt.figure(figsize=(10, 5))

plt.plot(X, A\_values, label='Tập mờ A', marker='o', linestyle='-')

plt.plot(X, B\_values, label='Tập mờ B', marker='x', linestyle='--')

plt.xlabel('Phần tử trong tập X')

plt.ylabel('Giá trị tập mờ')

plt.title('Biểu đồ tập mờ A và B')

plt.legend()

plt.grid(True)

plt.show()

**Xây dựng một hệ thống điều khiển máy lạnh tự động cho phòng máy tính căn cứ vào số lượng**

**người có trong phòng, nhiệt độ trong phòng học và nhiệt độ bên ngoài phòng.**

**Trong giai đoạn 1: Dữ liệu được đề xuất như sau**

**Giả sử dữ liệu nhập:**

**- Số người trong phòng N= [1,40] ∈ N**

**- Nhiệt độ trong phòng Ti= [16,32] ∈ N**

**- Nhiệt độ ngoài trời To = [13,42] ∈ N**

**Kết quả xuất và điều chỉnh:**

**- Mức điều chỉnh L = [18,30] ∈ N**

**Yêu cầu: Hãy xây dựng mức điều chỉnh nhiệt độ trong phòng theo quy trình 5 giai đoạn và thiện**

**thực bằng Python.**

import skfuzzy as fuzz

import numpy as np

import matplotlib.pyplot as plt

from skfuzzy import control as ctrl

# Khởi tạo các Antecedent và Consequent

N = ctrl.Antecedent(np.arange(1, 41), 'so\_nguoi')

Tin = ctrl.Antecedent(np.arange(16, 33), 'nhiet\_do\_trong\_phong')

Tout = ctrl.Antecedent(np.arange(13, 43), 'nhiet\_do\_ngoai\_troi')

L = ctrl.Consequent(np.arange(18, 31), 'Muc\_dieu\_chinh')

# Định nghĩa các hàm mục tiêu (membership functions)

N['It nguoi'] = fuzz.trapmf(N.universe, [1, 1, 2, 3])

N['Trung binh'] = fuzz.trapmf(N.universe, [2, 3, 5, 6])

N['Nhieu nguoi'] = fuzz.trapmf(N.universe, [5, 6, 7, 8])

N['Rat nhieu'] = fuzz.trapmf(N.universe, [7, 8, 10, 10])

Tin['Lanh'] = fuzz.trapmf(Tin.universe, [16, 16, 19, 21])

Tin['Vua'] = fuzz.trapmf(Tin.universe, [20, 22, 25, 27])

Tin['Nong'] = fuzz.trapmf(Tin.universe, [26, 28, 30, 30])

Tout['Lanh'] = fuzz.trapmf(Tout.universe, [13, 13, 20, 22])

Tout['Vua'] = fuzz.trapmf(Tout.universe, [21, 23, 26, 28])

Tout['Nong'] = fuzz.trapmf(Tout.universe, [27, 29, 34, 36])

Tout['Rat nong'] = fuzz.trapmf(Tout.universe, [35, 38, 42, 42])

L['Rat thap'] = fuzz.trimf(L.universe, [18, 18, 20])

L['Thap'] = fuzz.trimf(L.universe, [19, 20, 22])

L['Trung binh'] = fuzz.trimf(L.universe, [21, 22, 23])

L['Cao'] = fuzz.trimf(L.universe, [22, 24, 26])

L['Rat cao'] = fuzz.trimf(L.universe, [25, 26, 30])

# Thêm các quy tắc vào rules theo yêu cầu của đề bài

# Khởi tạo hệ thống điều khiển

system = ctrl.ControlSystem(rules)

sim = ctrl.ControlSystemSimulation(system)

# Nhập dữ liệu

iN = int(input('Nhập số người trong phòng (1-40): '))

iTin = float(input('Nhập nhiệt độ trong phòng (16-32): '))

iTout = float(input('Nhập nhiệt độ ngoài trời (13-42): '))

# Gán dữ liệu đầu vào

sim.input['so\_nguoi'] = iN

sim.input['nhiet\_do\_trong\_phong'] = iTin

sim.input['nhiet\_do\_ngoai\_troi'] = iTout

# Tính toán

sim.compute()

# Lấy kết quả

muc\_dieu\_chinh = sim.output['Muc\_dieu\_chinh']

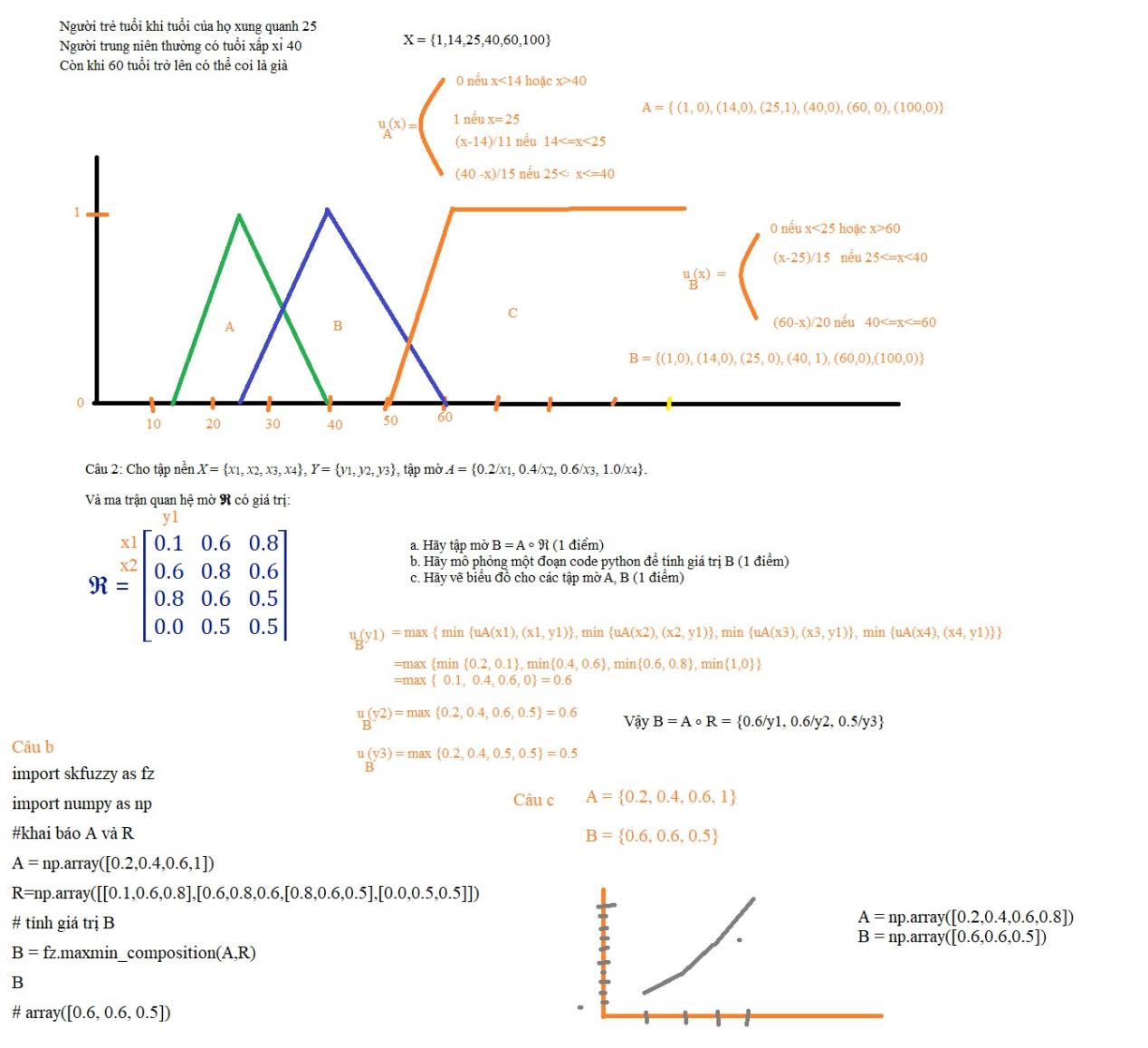
# Hiển thị kết quả

L.view(sim)

plt.show()

print(f'Với {iN} người trong phòng, nhiệt độ trong phòng là {iTin} độ C, và nhiệt độ ngoài trời là {iTout} độ C,')

print(f'mức điều chỉnh là {muc\_dieu\_chinh} độ C.')



Sự khác biệt giữa fuzzy logic và logic thông thường.

Xử lý không chắc chắn:

Logic thông thường chỉ xử lý các giá trị đúng hoặc sai (0 hoặc 1). Nó không thể xử lý thông tin mờ mịt hoặc không chắc chắn.

Fuzzy logic cho phép xử lý các giá trị mờ mịt và biểu đạt sự không chắc chắn bằng cách sử dụng tập mờ (fuzzy set) và các hàm thuộc tính mờ (fuzzy membership functions).

Biểu đạt mức độ:

Logic thông thường biểu đạt mức độ của sự đúng sai bằng các giá trị rời rạc như 0 hoặc 1 (đúng hoặc sai).

Fuzzy logic biểu đạt mức độ của sự đúng sai bằng các giá trị liên tục trong khoảng từ 0 đến 1, cho phép biểu đạt mức độ của sự đúng sai một cách mờ mịt.

Quyết định:

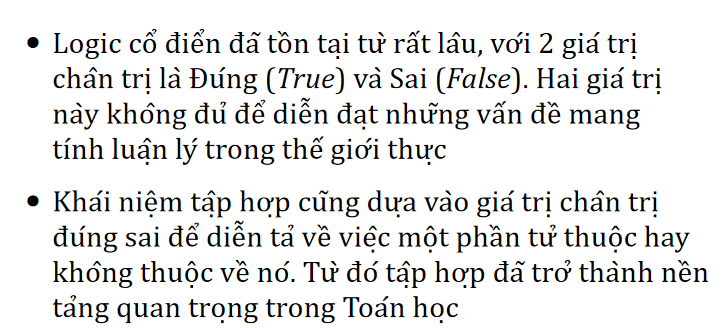
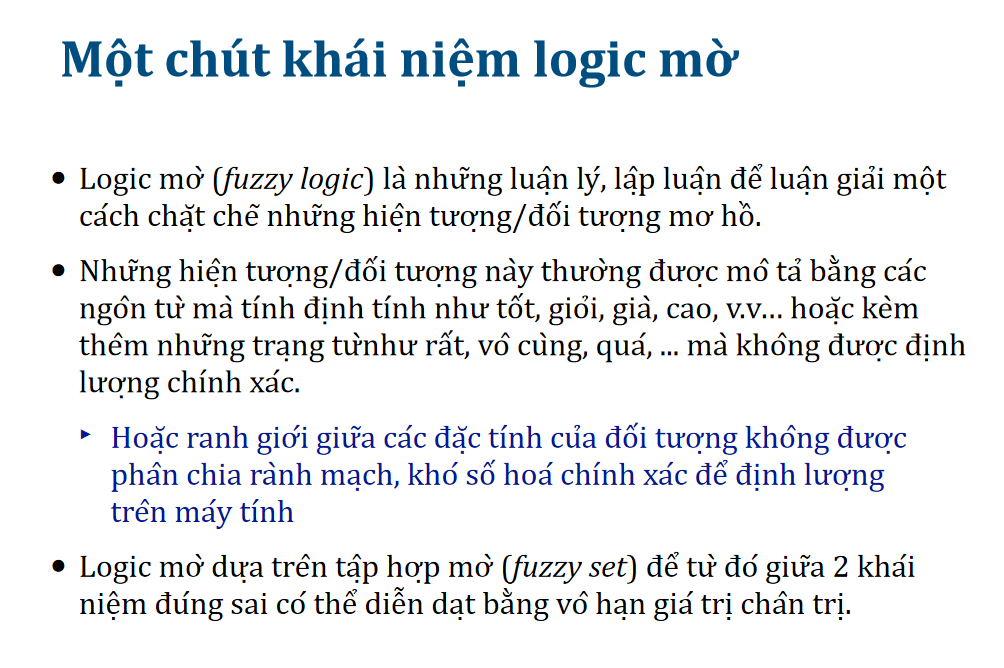
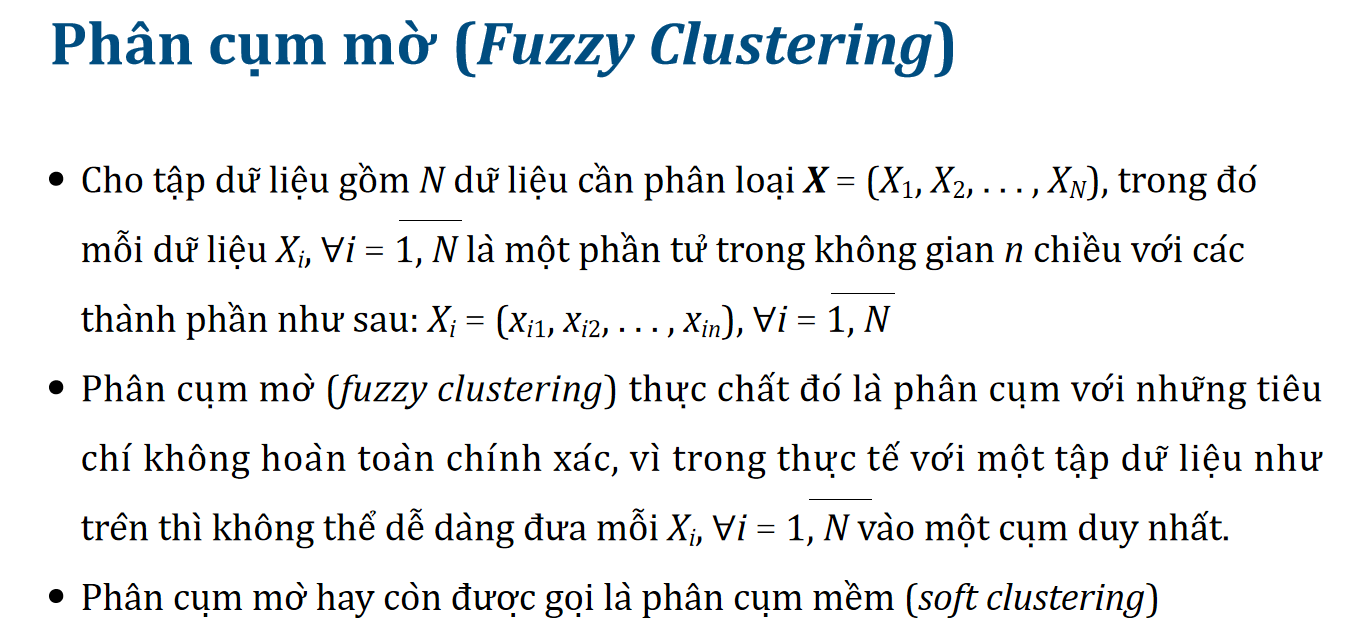
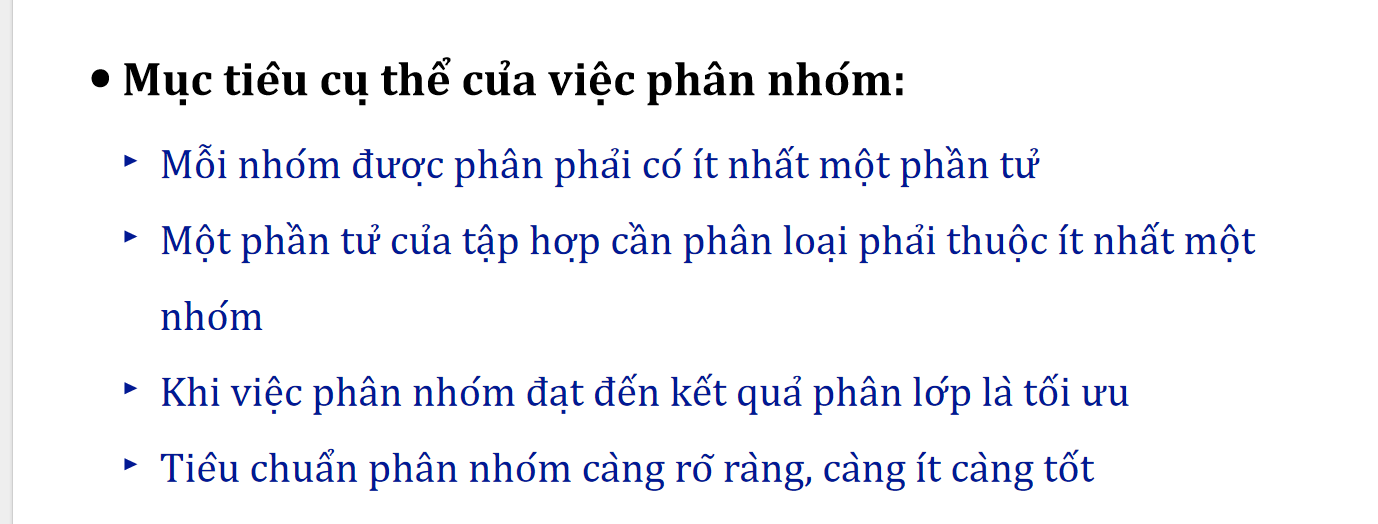
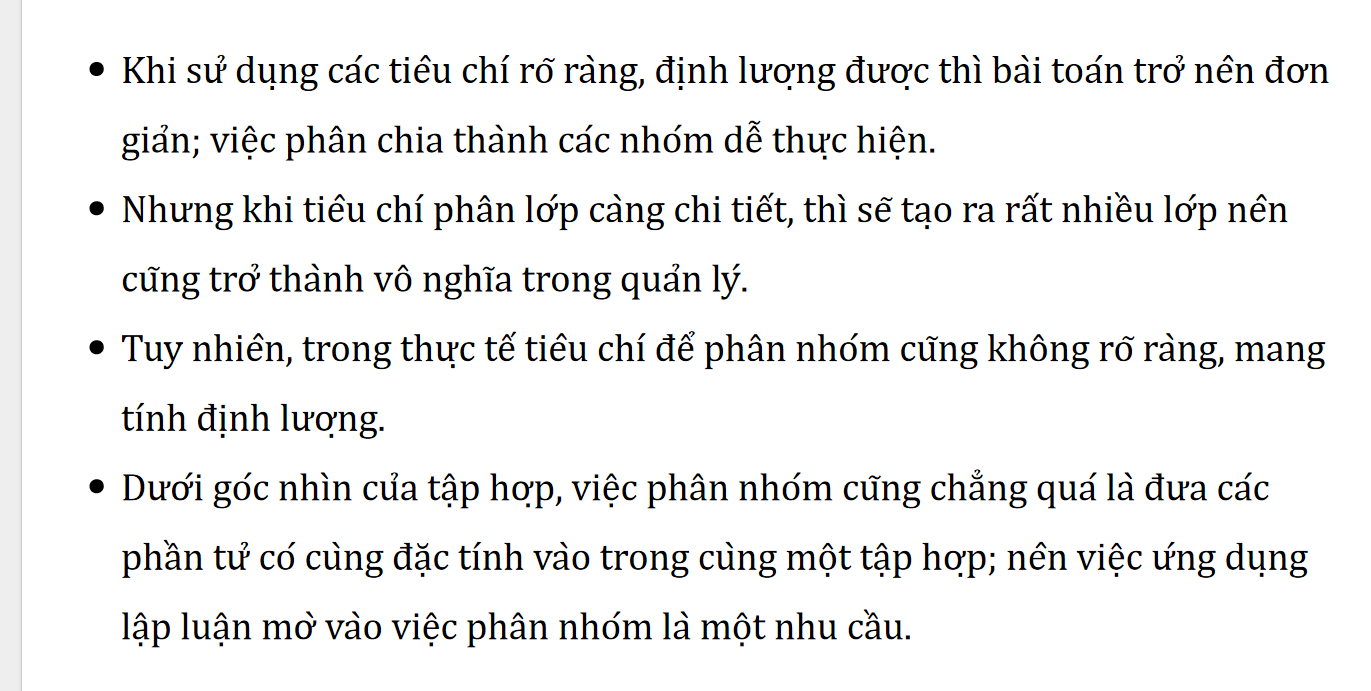
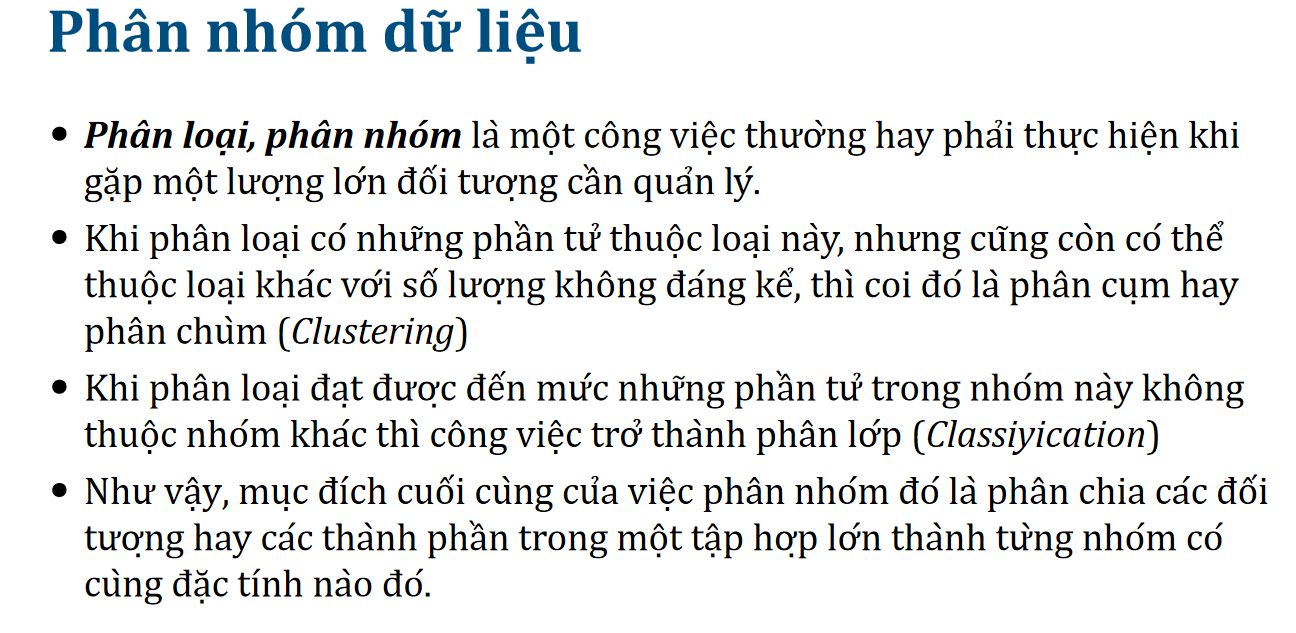
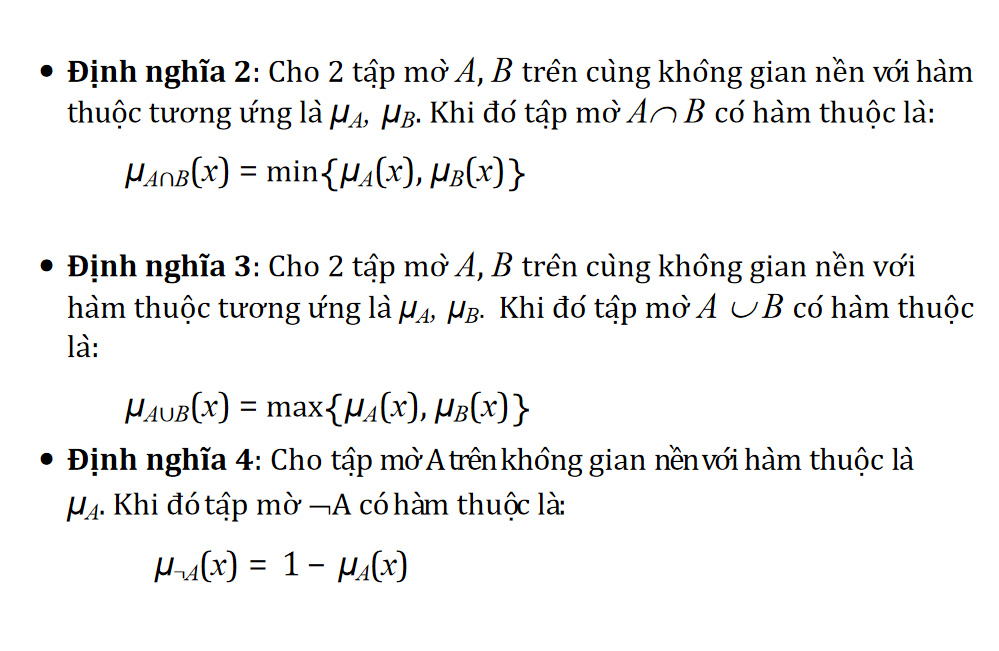
Logic thông thường thực hiện quyết định dựa trên các luật logic cụ thể và thường dựa vào giá trị đúng hoặc sai.

Fuzzy logic cho phép quyết định dựa trên mức độ của sự đúng sai và có khả năng xử lý tình huống mờ mịt và không chắc chắn.

Ứng dụng:

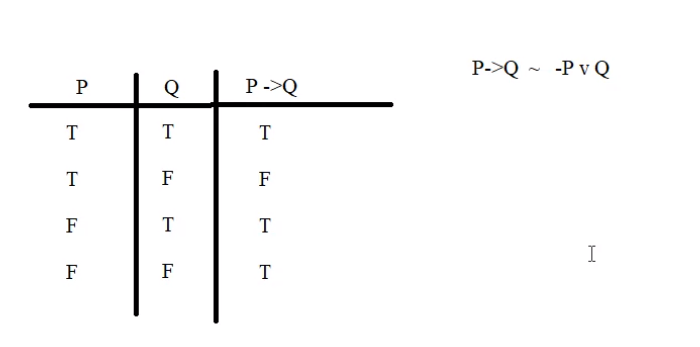
Logic thông thường thường được sử dụng trong các hệ thống đơn giản hoặc trong các tình huống mà thông tin đúng sai là đủ.

Fuzzy logic thường được sử dụng trong các hệ thống có sự không chắc chắn và mờ mịt, như trong điều khiển tự động, trí tuệ nhân tạo, và các lĩnh vực liên quan đến quyết định dựa trên đánh giá con người.



Tập mờ là một khái niệm trong lý thuyết tập học mờ (Fuzzy Set Theory). Trong lý thuyết này, tập mờ là một tập hợp các phần tử, trong đó mỗi phần tử được gán một giá trị từ 0 đến 1 để biểu thị mức độ thuộc tính của phần tử đó trong tập.

Ví dụ, trong tập mờ "cao", một phần tử như "1.75m" có thể được gán một giá trị thuộc tính là 0.8, biểu thị mức độ cao của chiều cao này trong tập mờ "cao". Từ đó, ta có thể đánh giá mức độ thuộc tính của một phần tử trong một tập mờ.



Giai đoạn 1: dữ liệu

Dữ liệu nhập:

Tổng số người nhiễm bệnh (dương tı́nh): XC = [1, 100000] ∈ ℕ

Số người đã tử vong: XD = [1, 10000] ∈ ℕ

Mức độ tiếp giáp: XB = {1, 2, 3}

Kết quả:

Mức độ lây lan: XS = [1, 10] ∈ ℕ

### Xây dựng 4 không gian nền tương ứng

XC = np.arange(1,100001)

XD = np.arange(1,10001)

XB = [1,2,3]

XS = np.arange(1,11)

### Định nghĩa tiền đề và kết luận dựa trên các biến ngôn ngữ

C = ctrl.Antecedent( XC,"Cases" )

D = ctrl.Antecedent( XD,"Deaths" )

B = ctrl.Antecedent( XB,"Border" )

S = ctrl.Consequent( XS,"Spread" )

Giai đoạn 2:

xây dựng các tập mờ

Số người nhiễm

Ít

Trung bı̀nh

Nhiều

Rất nhiều

C["It"] = fz.zmf( C.universe, 1000, 10000 )

C["Trung binh"] = fz.pimf( C.universe, 3000, 10000,2 0000, 60000 )

C["Nhieu"] = fz.gbellmf( C.universe, 8000, 3, 70000 )

C["Rat nhieu"] = fz.smf( C.universe, 65000, 90000 )

C.view()

Số người tử vong

Ít

Trung bı̀nh

Nhiều

Rất nhiều

D["It"] = fz.trapmf( D.universe,[1,1,200,500])

D["Trung binh"] = fz.trapmf( D.universe,[300,1000,2000,6000])

D["Nhieu"] = fz.trapmf( D.universe,[5000,6000,7000,8000])

D["Rat nhieu"] = fz.trapmf( D.universe,[7500,8000,10000,10000])

D.view()

Mức độ tiếp giáp

Không

Có

Gián tiếp

B["Khong"] = fz.trimf( B.universe,[1, 1, 2])

B["Co"] = fz.trimf( B.universe,[1, 2, 3])

B["Gian tiep"] = fz.trimf( B.universe,[2, 3, 3])

B.view()

Mức độ lây lan

Không

Ít

Vừa phải

Cao

Rất cao

S["Khong"] = fz.trimf( S.universe,[1, 1, 3] )

S["It"] = fz.trapmf( S.universe, [2, 3, 4, 5] )

S["Vua phai"] = fz.trapmf( S.universe, [4, 5, 6, 7] )

S["Cao"] = fz.trapmf( S.universe, [6, 7, 8, 9] )

S["Rat cao"] = fz.trimf( S.universe, [8, 10, 10] )

S.view()

Giai đoạn 3: Luật mờ

Số người nhiễm, hoặc số người tử vong càng nhiều thı̀ mức độ lây lan càng cao

Tiếp giáp với vùng dịch thı̀ dễ lây nhiễm

Để phủ khắp, số luật cần có: 4 × 4 × 3 = 48 luật

rules = [..]

model = ctrl.ControlSystemSimulation(ctrl.ControlSystem(rules))

iC = int( input("Số người bị nhiễm: "))

iD = int( input("Tổng số tử vong: "))

iB = int( input("Mức độ tiếp giáp (1:không, 2:có, 3:gián tiếp): "))

model.input["Cases"] = iC

model.input["Deaths"] = iD

model.input["Border"] = iB

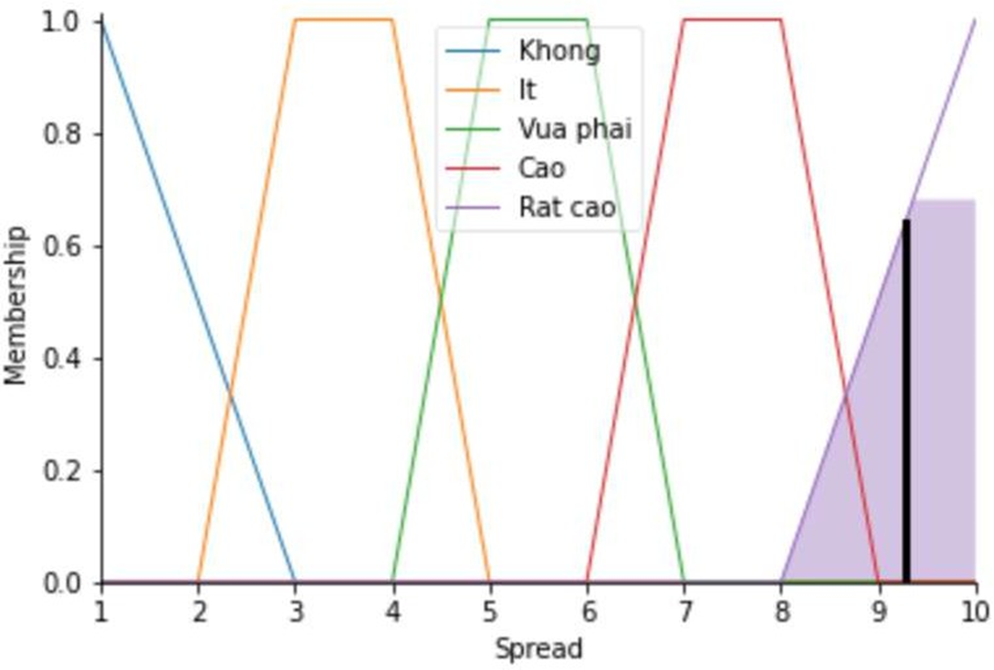
model.compute()

oS = model.output["Spread"]

print( "Số người nhiễm %d, tử vong %d; tiếp giáp cấp độ %d. Khả năng lây lan cấp độ %d" %

(iC,iD,iB,oS))

S.view( model )



### Xây dựng không gian nền tương ứng

X = np.arange(0,101,1)

Y = X

Z = np.array( [0, 4, 18, 32, 46, 60] )

### Xác định tiền đề và kết luận

D = ctrl.Antecedent(X, "Độ bẩn")

G = ctrl.Antecedent( Y, "Lượng dầu mở" )

T = ctrl.Consequent( Z, "Thời gian giặt" )

### Các tập mờ

D["Bẩn ít"] = fz.trimf(D.universe,[0, 0, 50])

D["Trung bình"] = fz.trimf(D.universe,[0, 50, 100])

D["Bẩn nhiều"] = fz.trimf(D.universe,[50, 100, 100])

G["Dầu mở ít"] = fz.trimf(G.universe,[0,0,50])

G["Dầu mở vừa phải"] = fz.trimf(G.universe,[0, 50, 100])

G["Dầu mở nhiều"] = fz.trimf(G.universe,[50, 100, 100])

T["Rất nhanh"] = fz.trapmf(T.universe,[0, 0, 4, 18])

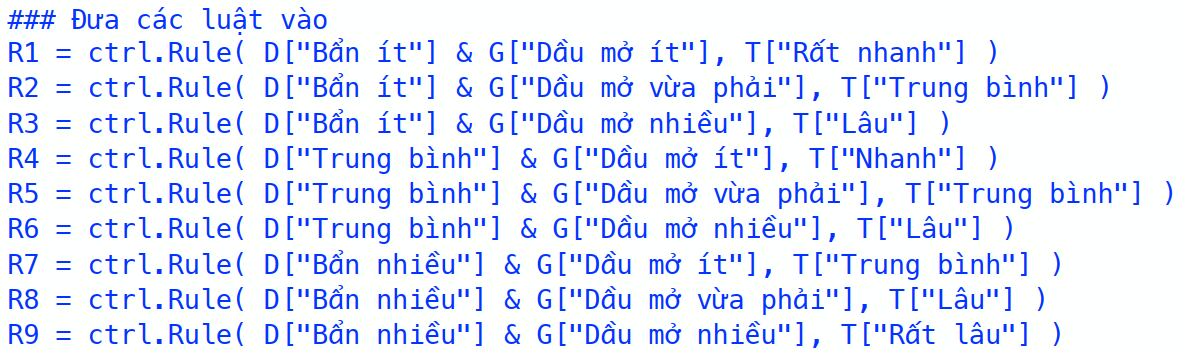
T["Nhanh"] = fz.trimf(T.universe,[4, 18, 32])

T["Trung bình"] = fz.trimf(T.universe,[18, 32, 46])

T["Lâu"] = fz.trimf(T.universe,[32, 46, 60])

T["Rất lâu"] = fz.trimf(T.universe,[46, 60, 60])

Đưa luật (Bước 4) và tạo mô hı̀nh (Bước 5)



### Tạo mô hình để suy diễn

model = ctrl.ControlSystemSimulation(ctrl.ControlSystem([R1,R2,R3,R4,R5,R6,R7,R8,R9]))

### Đưa dữ liệu vào để xử lý rồi xuất kết quả

B = float(input("Độ bẩn của áo quần:"))

C = float(input(“Lượng dầu mở dính:"))

model.input["Độ bẩn"] = B

model.input["Lượng dầu mở"] = C

model.compute()

D = model.output["Thời gian giặt"]

print( "Với %.2f độ bản và %.2f lượng dầu mở trên áo quần, thời gian giặt là %2.f phút" % (B,C,D)

### Tạo 2 không gian nền Tiền đề và 1 không gian nền Kết luận

X = np.arange( 1,11,1 )

Y = np.arange( 18,31,1 )

Z = np.arange( 20,41,1 )

W = np.arange( 18,28,1 )

N = ctrl.Antecedent( X,"So nguoi" )

Tin = ctrl.Antecedent( Y,"Nhiet do trong phong" )

Tout = ctrl.Antecedent( Z,"Nhiet do ngoai troi" )

L = ctrl.Consequent( W,"Muc dieu chinh" )

Giai đoạn 2: xây dựng các tập mờ

Số người: có 4 tập mờ

Ít người

Trung bı̀nh

Nhiều người

Rất nhiều

N["It nguoi"] = fz.trapmf( N.universe,[1,1,2,3] )

N["Trung binh"] = fz.trapmf( N.universe,[2,3,5,6] )

N["Nhieu nguoi"] = fz.trapmf( N.universe,[5,6,7,8] )

N["Rat nhieu"] = fz.trapmf( N.universe,[7,8,10,10] )

N.view()

Nhiệt độ trong phòng: có 3 tập mờ

Lạnh

Vừa

Nóng

Tin["Lanh"] = fz.trapmf( Tin.universe,[18,18,19,21] )

Tin["Vua"] = fz.trapmf( Tin.universe,[20,22,25,27] )

Tin["Nong"] = fz.trapmf( Tin.universe,[26,28,30,30] )

Tin.view()

Nhiệt độ ngoài trời: có 4 tập mờ:

Lạnh

Vừa

Nóng

Rất nóng

Tout["Lanh"] = fz.trapmf( Tout.universe,[20,20,21,22] )

Tout["Vua"] = fz.trapmf( Tout.universe,[21,23,26,28] )

Tout["Nong"] = fz.trapmf( Tout.universe,[27,29,34,36] )

Tout["Rat nong"] = fz.trapmf( Tout.universe,[35,38,40,40] )

Tout.view()

Mức điều chỉnh: có 5 tập mờ:

Rất thấp

Thấp

Trung bı̀nh

Cao

Rất cao

L["Rat thap"] = fz.trimf( L.universe,[18,18,20] )

L["Thap"] = fz.trimf( L.universe,[19,20,22] )

L["Trung binh"] = fz.trapmf( L.universe,[21,22,23,24] )

L["Cao"] = fz.trapmf( L.universe,[23,24,25,26] )

L["Rat cao"] = fz.trapmf( L.universe,[25,26,27,27] )

L.view()

Giai đoạn 3: Luật mờ dạng if then được rút ra theo quy tắc của "chuyên gia" như sau:

Nếu nhiệt độ càng nóng thı̀ mức điều chı̉nh phải thấp.

Nhiệt độ (cả bên ngoài lahn bên trong) tỷ lệthuận với số người có trong phòng

Bằng ngôn ngữ Python (không phải ngôn ngữ tự nhiên như bı̀nh thường) các luật này được viết dưới dạng như sau:

R = ctrl.Rule( N["..."] & Tin["..."] & Tout["..."], L["..."] )

Giai đoạn 4: xây dựng hệ thống mô phỏng dựa trên các luật đã có

rules = [...]

model = ctrl.ControlSystemSimulation(ctrl.ControlSystem(rules))

Giai đoạn 5: nhập dữ liệu đầu vào là 3 yếu tố: N, Tin, Tout sau đó giải mờ và xuất kết quả là L.

iN = int(input("Số người: "))

iTin = float(input("Nhiệt độ trong phòng: "))

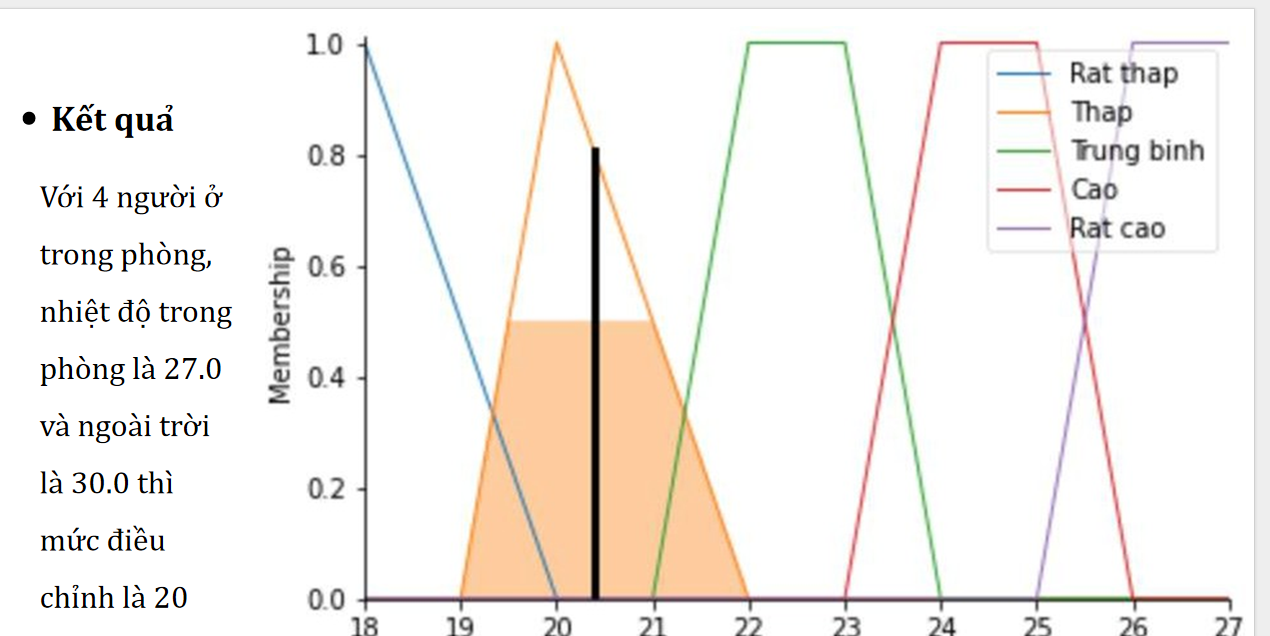
iTout = float(input("Nhiệt độ bên ngoài: "))

model.input["So nguoi"] = iN

model.input["Nhiet do trong phong"] = iTin

model.input["Nhiet do ngoai troi"] = iTout

model.compute()  
oL = model.output["Muc dieu chinh"]  
L.view(model)  
print("Với %d người ở trong phòng, nhiệt độ trong phòng là %.1f và ngoài trời là %.1f   
 thì mức điều chỉnh là %d" % (iN, iTin, iTout, oL))



import numpy as np

import skfuzzy as fuzz

import matplotlib.pyplot as plt

water\_level = np.arange(0,101,10)

control\_signal = np.arange(0,11,1)

water\_level\_low = fuzz.trimf(water\_level,[0,0,50])

water\_level\_medium=fuzz.trimf(water\_level,[0,50,100])

water\_level\_high=fuzz.trimf(water\_level,[50,100,100])

control\_signal\_low=fuzz.trimf(control\_signal,[0,0,5])

control\_signal\_medium=fuzz.trimf(control\_signal,[0,5,10])

control\_signal\_high=fuzz.trimf(control\_signal,[5,10,10])

rule1=np.fmin(water\_level\_low,control\_signal\_high)

rule2=np.fmin(water\_level\_medium,control\_signal\_medium)

rule3=np.fmin(water\_level\_high,control\_signal\_low)

aggregated = np.fmax(rule1,np.fmax(rule2,rule3))

control\_value=fuzz.defuzz(control\_signal,aggregated,'centroid')

fig,ax=plt.subplots()

ax.plot(control\_signal,control\_signal\_low,'b',linewidth=1.5,label='Low')

ax.plot(control\_signal,control\_signal\_medium,'g',linewidth=1.5,label='Medium')

ax.plot(control\_signal,control\_signal\_high,'r',linewidth=1.5,label='High')

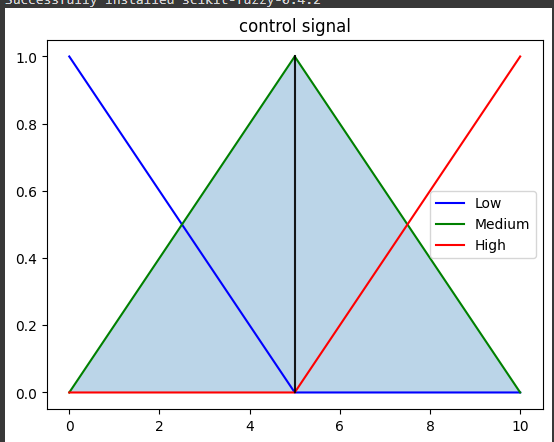
ax.fill\_between(control\_signal, 0, aggregated, alpha=0.3)

ax.plot([control\_value, control\_value],[0,fuzz.interp\_membership(control\_signal,aggregated, control\_value)],'k',linewidth=1.5,alpha=0.9)

ax.set\_title('control signal')

ax.legend()

plt.show()



**Bước 1**: Xây dựng không gian nền tương ứng (dùng mảng numpy.array([]) để lưu trữ).

**Bước 2:** Xác định đâu là tập tiền đề đâu là tập kết luận qua hàm trong module control:

fuzzy.control.Antecendent([],\_), fuzzy.control.Consequent([],\_)

**Bước 3:** Có thể dùng numpy.arange(\_,\_,\_) hay numpy.array([]) để tạo tập mờ, hoặc dùng một trong các hàm thành viên như: fuzzy.trimf([],[]), fuzzy.trapmf([],[]), fuzzy.gbellmf([],[]), v.v…

**Bước 4:** Dùng để đưa các luật rules vào bằng fuzzy.control.Rule(.,.)

**Bước 5:** Tạo mô hı̀nh (model) suy diễn với fuzzy.control.ControlSystemSimulation(fuzzy.control.ControlSystem(rules))

**Bước 6:** Nhập dữ liệu như là sự kiện cần suy đoán thông qua phương thức input() từ model ở trên, rồi dùng phương thức compute() để xử lý sau đó xuất kết quả bằng phương thức output()