Aim:

To design and implement a fuzzy control system for an intelligent fan controller using a fuzzy logic library (Scikit-Fuzzy in Python).

Theory:

Fuzzy Logic is a soft computing technique that handles imprecision and uncertainty by allowing intermediate values between 0 and 1, unlike classical binary logic[1]. A Fuzzy Logic Controller (FLC) is an intelligent control system that makes decisions using linguistic rules instead of mathematical models[2].

It consists of the following main components[3]:

- **Identification of Variables:** Define the input variables (e.g., temperature, humidity) and output variables (e.g., fan speed)[4].
- **Fuzzy Subset Configuration:** Each variable is divided into fuzzy subsets such as Low, Medium, and High[5].
- **Obtaining Membership Functions:** Membership functions, often shaped as triangles or trapezoids, represent the degree to which a variable belongs to a fuzzy subset[6].
- **Fuzzy Rule Base Configuration:** A set of IF–THEN rules determines the control actions[7]. For example: IF temperature is Hot AND humidity is High THEN fan speed is VeryHigh.
- **Fuzzification:** Converts crisp numerical inputs (e.g., Temperature = 35°C) into fuzzy values using the membership functions[8].
- **Combining Fuzzy Outputs (Inference Engine):** Uses logical operators (AND, OR) to evaluate rules and combine their results, producing a fuzzy output[9].
- **Defuzzification:** Converts the fuzzy output into a single crisp control action (e.g., Fan Speed = 95%) using methods like Centroid or Mean of Maximum[10].

Thus, a Fuzzy Logic Controller works in seven systematic steps to map input conditions to desired output actions[11].

Design of Fuzzy Fan Controller:

Inputs:

1. **Temperature (0–40 °C)**
   - Cool: [0, 0, 10, 20]
   - Warm: [15, 22, 30]
   - Hot: [25, 30, 40, 40]
2. **Humidity (0–100 %)**
   - Low: [0, 0, 20, 40]
   - Medium: [30, 50, 70]
   - High: [60, 80, 100, 100]

Output:

1. **Fan Speed (0–100 %)**
   - VeryLow, Low, Medium, High, VeryHigh

Rule Base:

The following table defines the 9 rules for the fan controller:

Temperature \ Humidity	Low	Medium	High
Cool	VeryLow	Low	Medium
Warm	Low	Medium	High
Hot	Medium	High	VeryHigh

Methodology:

1. Identify input and output variables[12].
2. Configure fuzzy subsets for each variable[13].
3. Define membership functions for inputs and output[14].
4. Construct the fuzzy rule base[15].
5. Perform fuzzification of crisp inputs[16].
6. Combine fuzzy outputs using an inference mechanism[17].
7. Defuzzify the result to obtain the final fan speed[18].

Output:

Test Cases:

- **Inputs -> Temperature: 15°C, Humidity: 10%**
  - **Output -> Fan Speed: 8.81%**
- **Inputs -> Temperature: 22°C, Humidity: 50%**
  - **Output -> Fan Speed: 55.00%**
- **Inputs -> Temperature: 35°C, Humidity: 90%**
  - **Output -> Fan Speed: 94.58%**

Membership Function Plots:

Code:

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

# 1. Define fuzzy variables
temperature = ctrl.Antecedent(np.arange(0, 41, 1), 'temperature')
humidity = ctrl.Antecedent(np.arange(0, 101, 1), 'humidity')
fan_speed = ctrl.Consequent(np.arange(0, 101, 1), 'fan_speed')

# 2. Membership functions
# Temperature: Cool, Warm, Hot
```

```
temperature['Cool'] = fuzz.trapmf(temperature.universe, [0, 0, 10, 20])
temperature['Warm'] = fuzz.trimf(temperature.universe, [15, 22, 30])
temperature['Hot']  = fuzz.trapmf(temperature.universe, [25, 30, 40, 40])

# Humidity: Low, Medium, High
humidity['Low']    = fuzz.trapmf(humidity.universe, [0, 0, 20, 40])
humidity['Medium'] = fuzz.trimf(humidity.universe, [30, 50, 70])
humidity['High']   = fuzz.trapmf(humidity.universe, [60, 80, 100, 100])

# Fan Speed: VeryLow, Low, Medium, High, VeryHigh
fan_speed['VeryLow']  = fuzz.trapmf(fan_speed.universe, [0, 0, 10, 20])
fan_speed['Low']      = fuzz.trimf(fan_speed.universe, [15, 30, 45])
fan_speed['Medium']   = fuzz.trimf(fan_speed.universe, [40, 55, 70])
fan_speed['High']     = fuzz.trimf(fan_speed.universe, [60, 75, 90])
fan_speed['VeryHigh'] = fuzz.trapmf(fan_speed.universe, [85, 95, 100, 100])

# 3. Rules
rules = [
    # Cool temperature
    ctrl.Rule(temperature['Cool'] & humidity['Low'], fan_speed['VeryLow']),
    ctrl.Rule(temperature['Cool'] & humidity['Medium'], fan_speed['Low']),
    ctrl.Rule(temperature['Cool'] & humidity['High'], fan_speed['Medium']),

    # Warm temperature
    ctrl.Rule(temperature['Warm'] & humidity['Low'], fan_speed['Low']),
    ctrl.Rule(temperature['Warm'] & humidity['Medium'], fan_speed['Medium']),
    ctrl.Rule(temperature['Warm'] & humidity['High'], fan_speed['High']),

    # Hot temperature
    ctrl.Rule(temperature['Hot'] & humidity['Low'], fan_speed['Medium']),
    ctrl.Rule(temperature['Hot'] & humidity['Medium'], fan_speed['High']),
    ctrl.Rule(temperature['Hot'] & humidity['High'], fan_speed['VeryHigh']),
]

# 4. Control system & simulation
system = ctrl.ControlSystem(rules)
sim = ctrl.ControlSystemSimulation(system)

# 5. Test Examples
sim.input['temperature'] = 35
sim.input['humidity'] = 90
sim.compute()
print(f"Output -> Fan Speed: {sim.output['fan_speed']:.2f}%")

# 6. Plot Membership Functions
temperature.view()
```

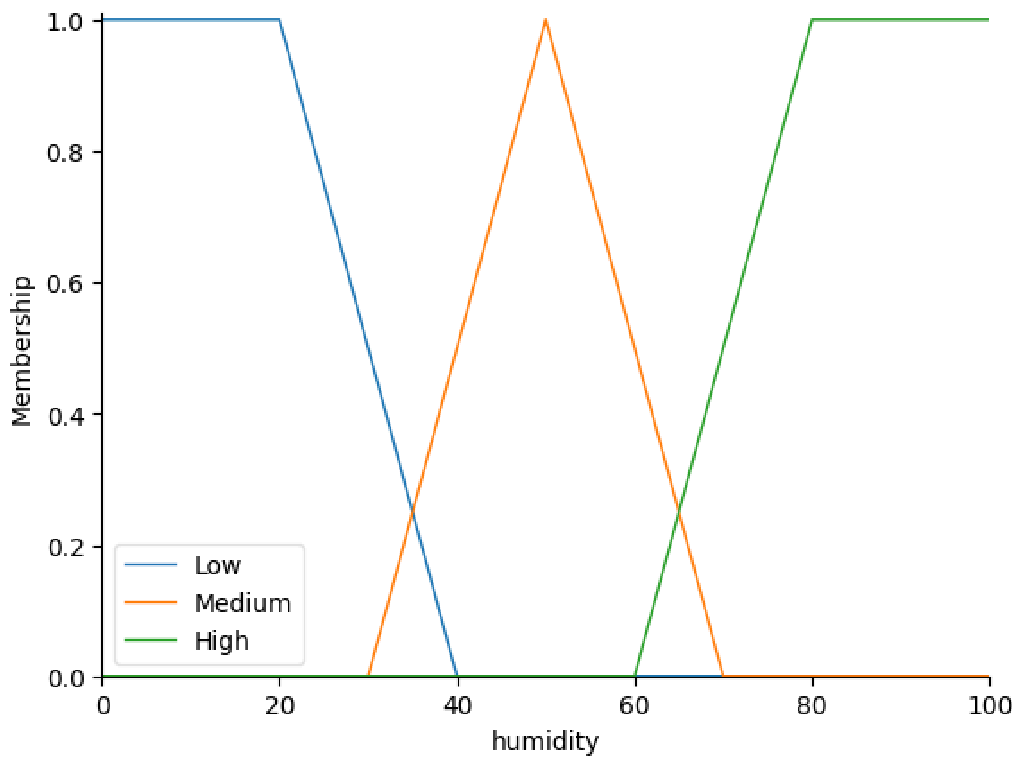
```
humidity.view()
fan_speed.view()
```

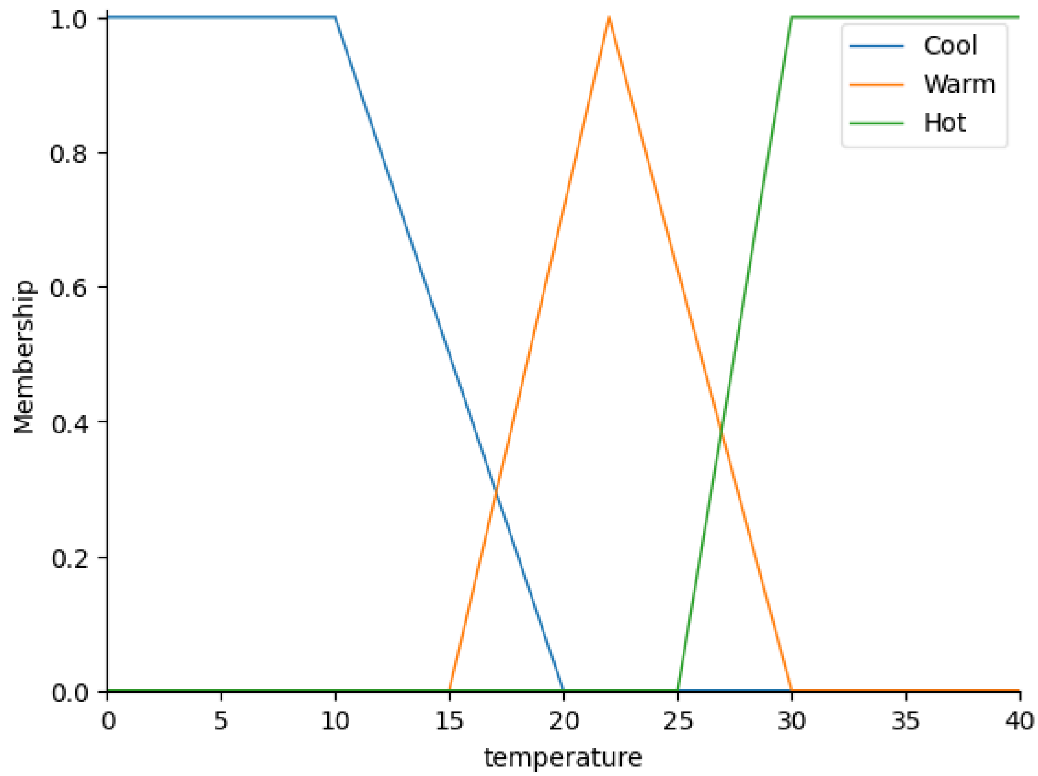
Output:

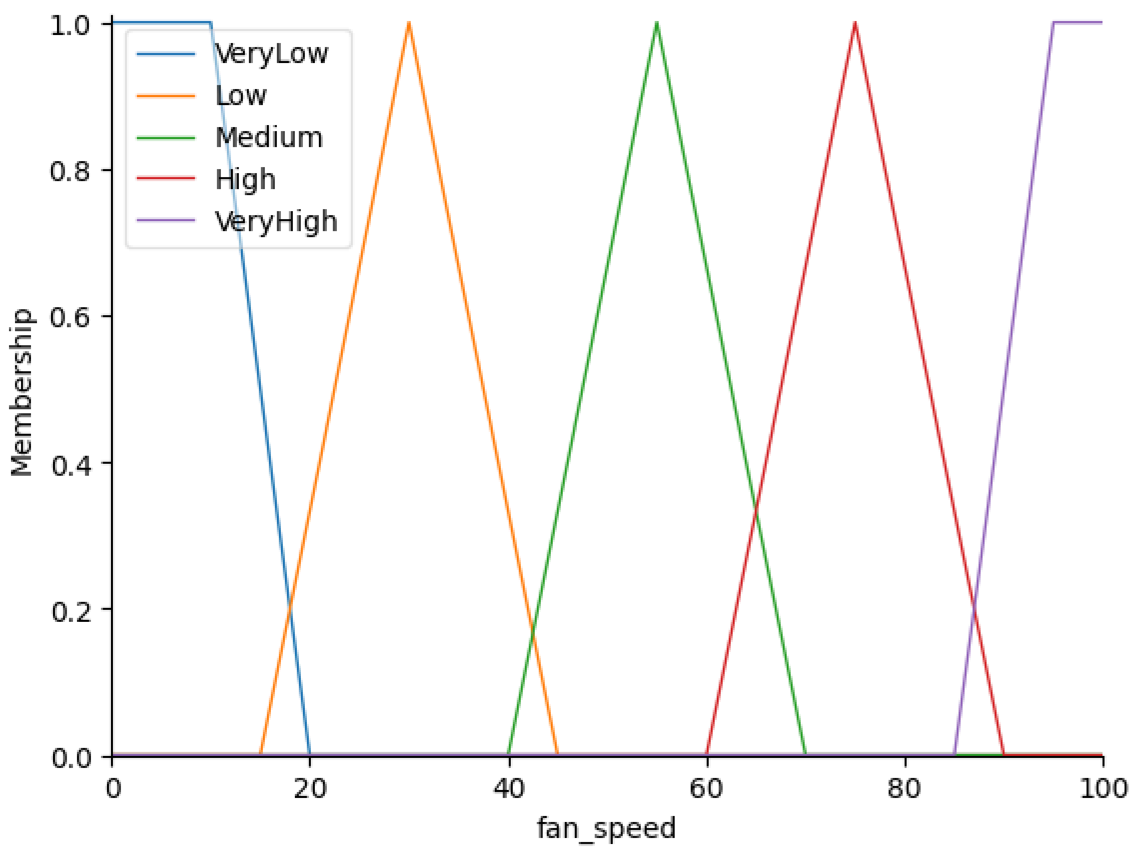
```
Inputs -> Temperature: 15°C, Humidity: 10%
Output -> Fan Speed: 8.81%

Inputs -> Temperature: 22°C, Humidity: 50%
Output -> Fan Speed: 55.00%

Inputs -> Temperature: 35°C, Humidity: 90%
Output -> Fan Speed: 94.58%
```




1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
5
10
15
20
25
30
35
40
temperature
Cool
Warm
Hot


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
20
40
60
80
100
fan_speed
VeryLow
Low
Medium
High
VeryHigh

Conclusion:

The fuzzy logic fan controller dynamically adjusts fan speed based on ambient temperature and humidity. It provides smooth, intelligent decision-making by using fuzzification, rule evaluation, and defuzzification[19]. Compared to manual switches or simple thermostats, this approach provides a more comfortable and energy-efficient environment by delivering a more nuanced response to environmental conditions.