Name: **GANESH SANJAY PANDHRE**

Roll Nos : **41** | Div: **D20B**

AIM: To design and implement a fuzzy control system using the scikit-fuzzy library in Python for controlling fan speed based on temperature input.

Experiment No: 8

Theory

A **fuzzy control system** is a rule-based decision-making system that applies **fuzzy logic** to handle imprecision and uncertainty in input data. Instead of using binary true/false logic, it uses **degrees of membership** between 0 and 1, allowing for smoother and more human-like decision-making.

Components of a Fuzzy Control System

1. **Fuzzification** Converts crisp input values (e.g., temperature = 30°C) into fuzzy values using **membership functions**. In this system:
    - `temperature` has fuzzy sets **low**, **medium**, and **high**.
    - `fan_speed` has fuzzy sets **slow**, **medium**, and **fast**.
2. **Rule Base** Contains IF–THEN rules derived from human reasoning:
    - IF temperature is low THEN fan speed is slow
    - IF temperature is medium THEN fan speed is medium
    - IF temperature is high THEN fan speed is fast
3. **Inference Engine** Applies the rules to the fuzzified inputs to determine fuzzy outputs.
4. **Defuzzification** Converts the fuzzy output back into a crisp numerical value (fan speed in %). The **centroid method** is used here by default.

Advantages

- Handles uncertainty and vagueness.
- No need for precise mathematical models.
- Mimics human reasoning.

```
!pip install scikit-fuzzy
```

```
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6 kB)
Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 920.8/920.8 kB 2.2 MB/s eta 0:00:00
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.5.0
```

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

```
temperature = ctrl.Antecedent(np.arange(0, 51, 1), 'temperature')  # 0 to 50°C
fan_speed = ctrl.Consequent(np.arange(0, 101, 1), 'fan_speed')     # 0 to 100%
```

```
temperature['low'] = fuzz.trimf(temperature.universe, [0, 0, 25])
temperature['medium'] = fuzz.trimf(temperature.universe, [15, 25, 35])
temperature['high'] = fuzz.trimf(temperature.universe, [25, 50, 50])

fan_speed['slow'] = fuzz.trimf(fan_speed.universe, [0, 0, 50])
fan_speed['medium'] = fuzz.trimf(fan_speed.universe, [25, 50, 75])
fan_speed['fast'] = fuzz.trimf(fan_speed.universe, [50, 100, 100])
```

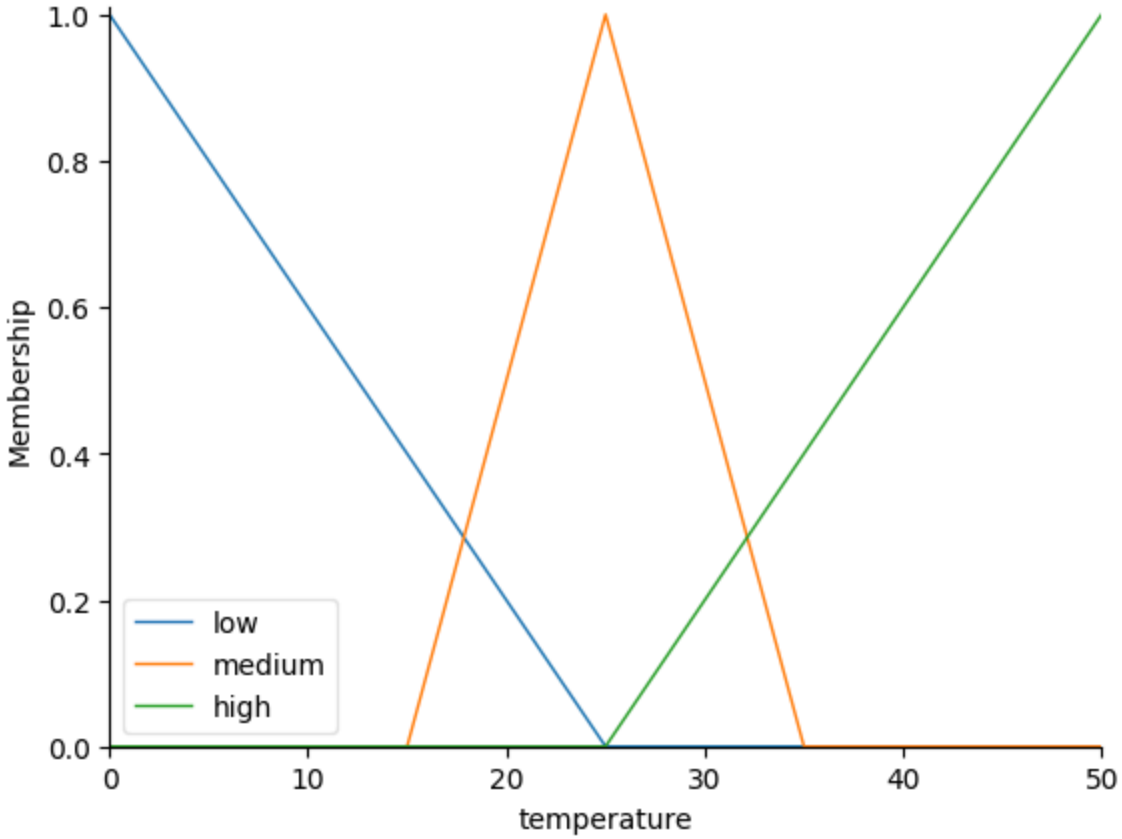
```
rule1 = ctrl.Rule(temperature['low'], fan_speed['slow'])
rule2 = ctrl.Rule(temperature['medium'], fan_speed['medium'])
rule3 = ctrl.Rule(temperature['high'], fan_speed['fast'])

# Step 4: Create control system
fan_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
fan_sim = ctrl.ControlSystemSimulation(fan_ctrl)
```

```
fan_sim.input['temperature'] = 30
fan_sim.compute()
```

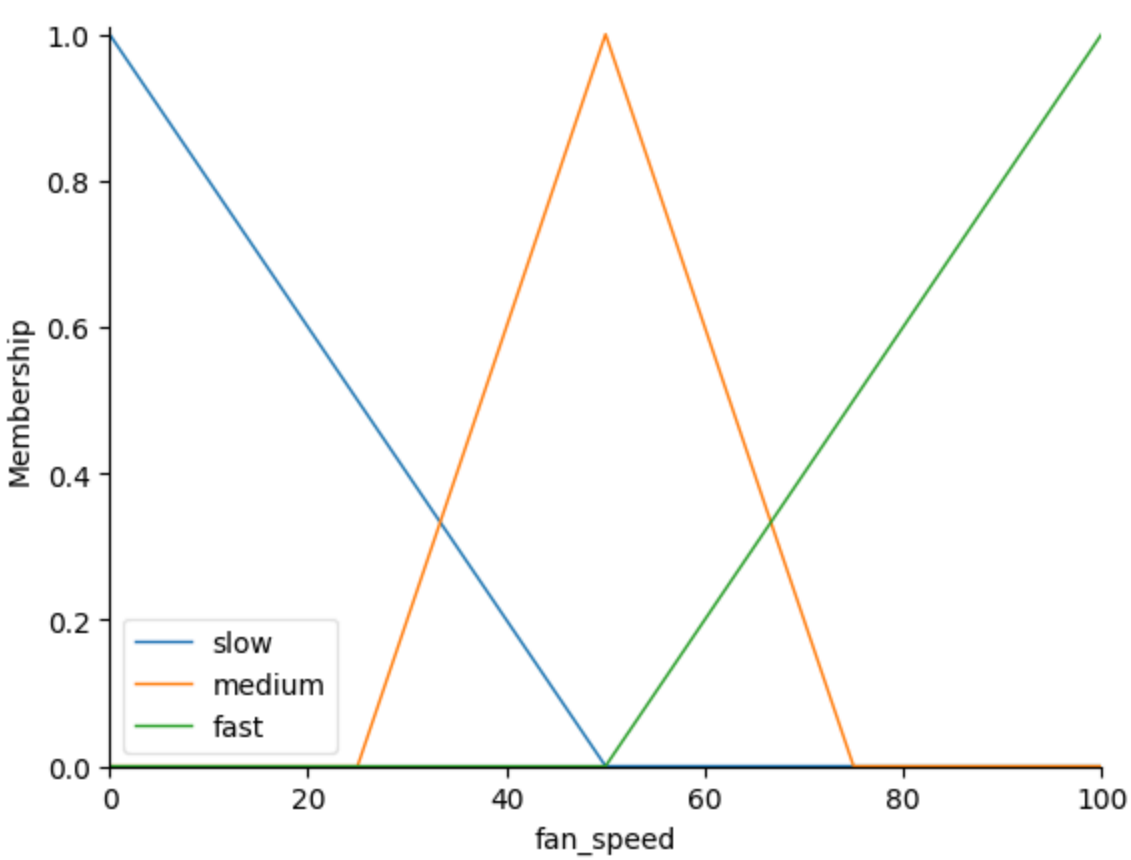
```
print("Input Temperature:", 30)
print("Output Fan Speed:", fan_sim.output['fan_speed'])
```

```
Input Temperature: 30
Output Fan Speed: 58.21305841924404
```

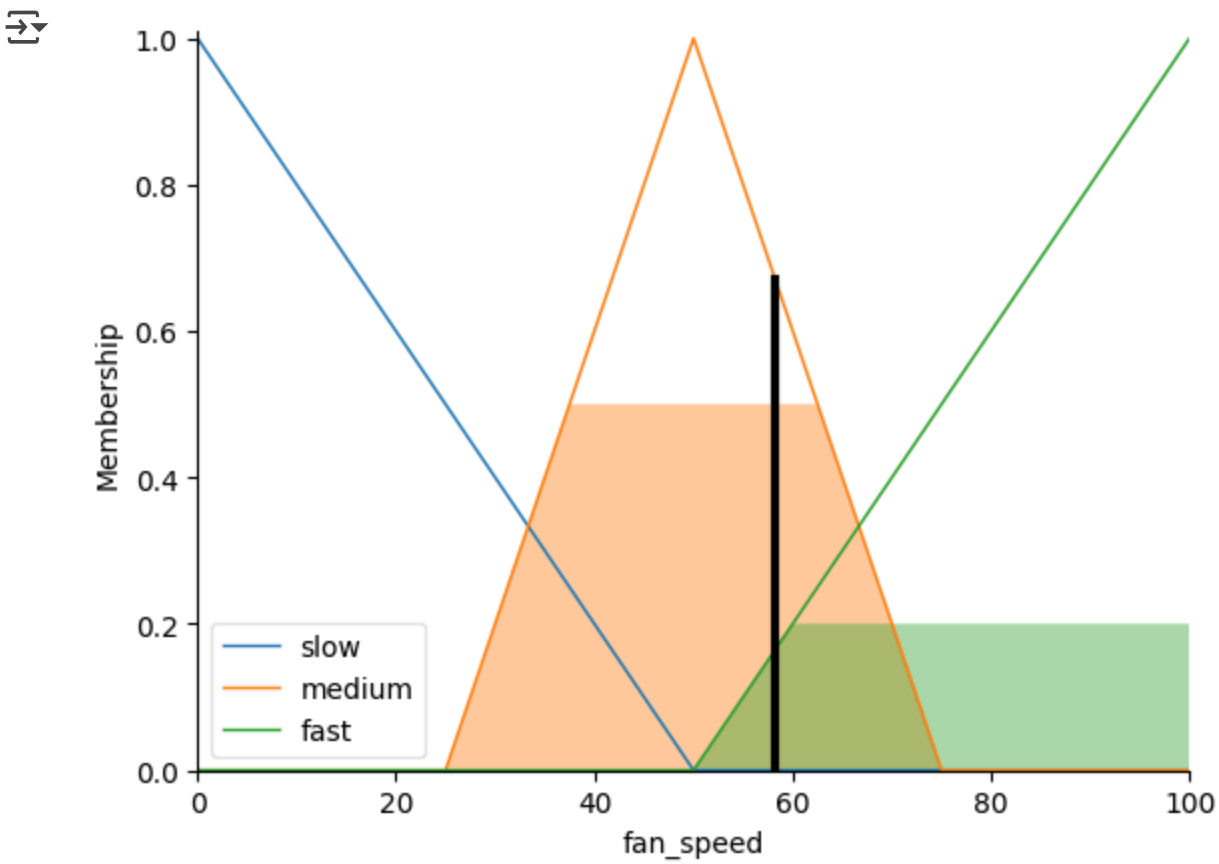
```
temperature.view()
```




```
fan_speed.view()
```



```
fan_speed.view(sim=fan_sim)
```



Conclusion

In this experiment, we successfully designed and implemented a fuzzy control system using the **scikit-fuzzy** Python library. The system:

- Accepted a crisp temperature input (30°C).
- Fuzzified it into degrees of **low**, **medium**, and **high** temperature.
- Applied the rule base to determine the output fuzzy set for fan speed.
- Defuzzified the result to produce a crisp fan speed value.

This demonstrates how fuzzy logic can be effectively applied for real-world control applications where inputs are imprecise, and decision-making needs to be smooth and adaptive.
