LAPORAN UJI COBA PRAKTIKUM - KECERDASAN KOMPUTASIONAL \n

Tujuan Praktikum : Melakukan Implementasi Inferensi dengan Metode Fuzzy Logic Mamdani dan Tsugeno

Nama : Abdan Hafidz

NRP : 5054231021

In [3]:
```
import numpy as np
import skfuzzy as fuzz
from matplotlib import pyplot as plt
```

Problem Overview

This example models a system that controls the heating comfort level on two input factors:

- **Humidity**: How humid the environment inside the greenhouse is.
- **Temperature**: The temperature inside the greenhouse.

The output of the system is **Comfort Level**, which determines how much the heating system needs to operate to maintain optimal conditions for plant growth.

Variables

Input Variables:

- **Temperature (°C)**:
  - **Low**: 0°C - 20°C
  - **Medium**: 15°C - 30°C
  - **High**: 25°C - 40°C
- **Humidity (%)**:
  - **Low**: 0% - 40%
  - **Medium**: 30% - 70%
  - **High**: 60% - 100%

Output Variable:

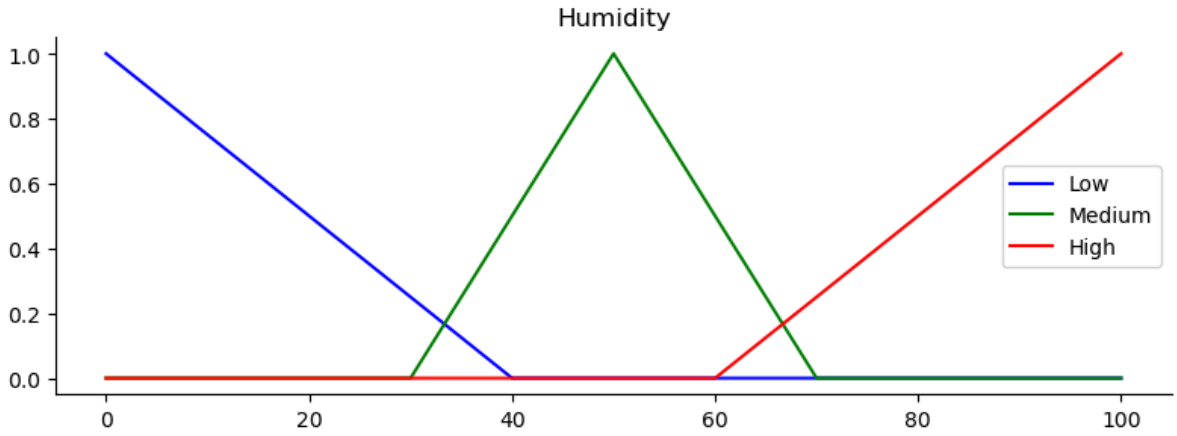
- **Comfort Level**:
  - **Low**: 0% - 40%
  - **Medium**: 30% - 70%
  - **High**: 60% - 100%

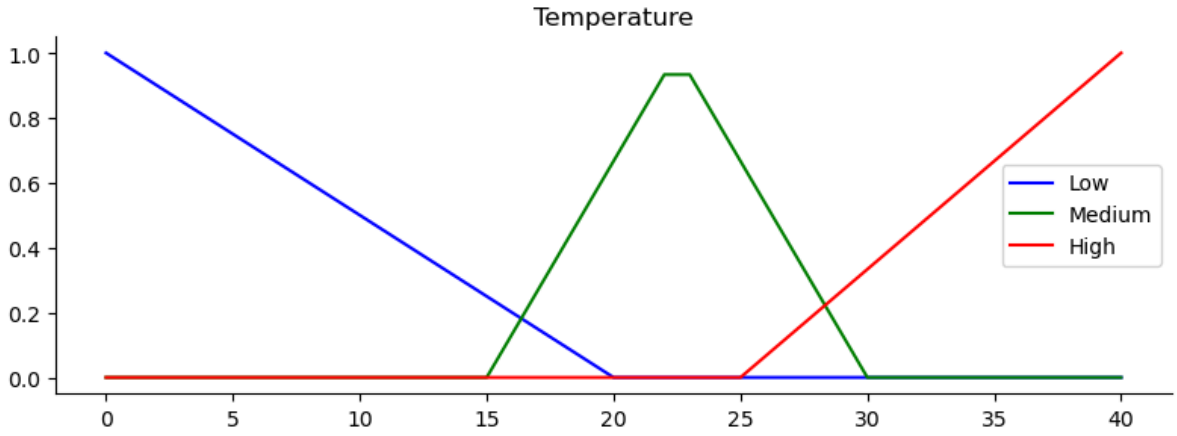
The fuzzy system uses triangular membership functions for the input and output variables.

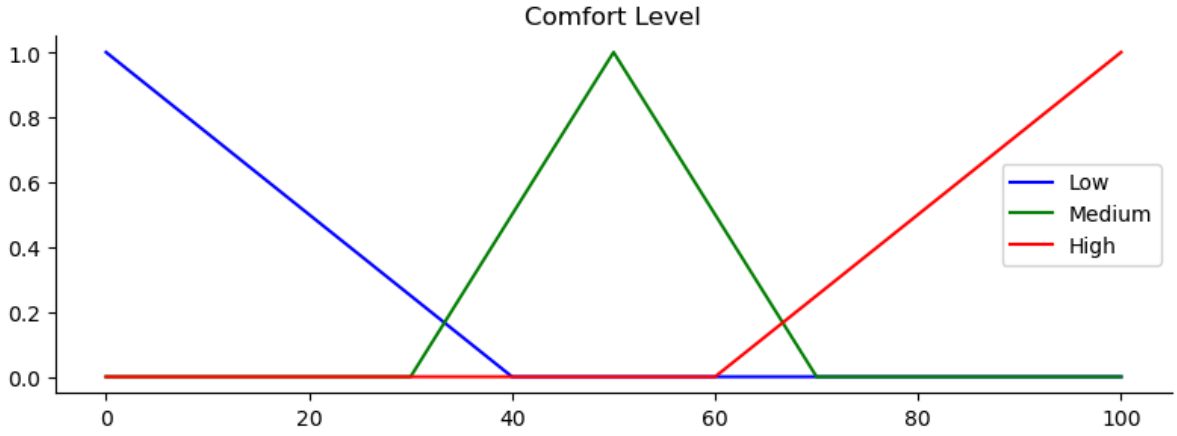
In [4]:
```
import numpy as np
import skfuzzy as fuzz
```

```
import matplotlib.pyplot as plt

# Define universe variables
#   * Humidity on the range [0, 100] percentage
#   * Temperature on the range [0, 40] degrees Celsius
#   * Comfort Level on the range [0, 100] percentage
x_humidity = np.arange(0, 101, 1)
x_temp = np.arange(0, 41, 1)
x_comfort = np.arange(0, 101, 1)

# Generate fuzzy membership functions for humidity
humidity_lo = fuzz.trimf(x_humidity, [0, 0, 40])
humidity_md = fuzz.trimf(x_humidity, [30, 50, 70])
humidity_hi = fuzz.trimf(x_humidity, [60, 100, 100])

# Generate fuzzy membership functions for temperature
temp_low = fuzz.trimf(x_temp, [0, 0, 20])
temp_medium = fuzz.trimf(x_temp, [15, 22.5, 30])
temp_high = fuzz.trimf(x_temp, [25, 40, 40])

# Generate fuzzy membership functions for comfort level (output)
comfort_low = fuzz.trimf(x_comfort, [0, 0, 40])
comfort_medium = fuzz.trimf(x_comfort, [30, 50, 70])
comfort_high = fuzz.trimf(x_comfort, [60, 100, 100])

# Visualize the membership functions for input and output variables
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))

# Humidity membership functions
ax0.plot(x_humidity, humidity_lo, 'b', linewidth=1.5, label='Low')
ax0.plot(x_humidity, humidity_md, 'g', linewidth=1.5, label='Medium')
ax0.plot(x_humidity, humidity_hi, 'r', linewidth=1.5, label='High')
ax0.set_title('Humidity')
ax0.legend()

# Temperature membership functions
ax1.plot(x_temp, temp_low, 'b', linewidth=1.5, label='Low')
ax1.plot(x_temp, temp_medium, 'g', linewidth=1.5, label='Medium')
ax1.plot(x_temp, temp_high, 'r', linewidth=1.5, label='High')
ax1.set_title('Temperature')
ax1.legend()
```

```
# Comfort Level membership functions
ax2.plot(x_comfort, comfort_low, 'b', linewidth=1.5, label='Low')
ax2.plot(x_comfort, comfort_medium, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_comfort, comfort_high, 'r', linewidth=1.5, label='High')
ax2.set_title('Comfort Level')
ax2.legend()

# Adjust subplot layout and remove top and right spines
for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

plt.tight_layout()
plt.show()
```







Fuzzy Rules

Now we define the rules that govern the behavior of the system. For this example, let's consider the following rules for the inference system:

1. If the **Temperature** is low and the **Humidity** is low, then the **Comfort Level** is low.
2. If the **Temperature** is low and the **Humidity** is medium, then the **Comfort Level** is medium.
3. If the **Temperature** is medium and the **Humidity** is low, then the **Comfort Level** is medium.
4. If the **Temperature** is medium and the **Humidity** is medium, then the **Comfort Level** is high.
5. If the **Temperature** is high and the **Humidity** is high, then the **Comfort Level** is high.

Rule Application

These rules will be applied based on the membership degrees of input values for **Temperature** and **Humidity**. Using the fuzzy logic system, we can infer the resulting **Comfort Level**.

Mamdani Inference System (Max-Min Aggregation)

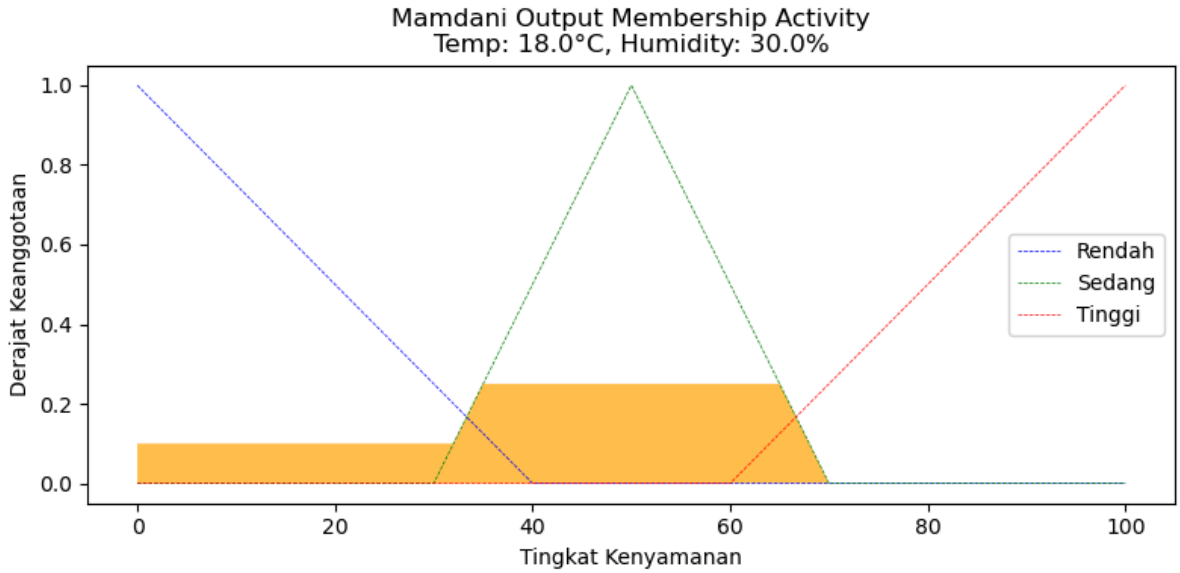
This method uses the min function for applying rules (i.e., determining how much each rule is active) and the max function for aggregating the output of multiple rules.

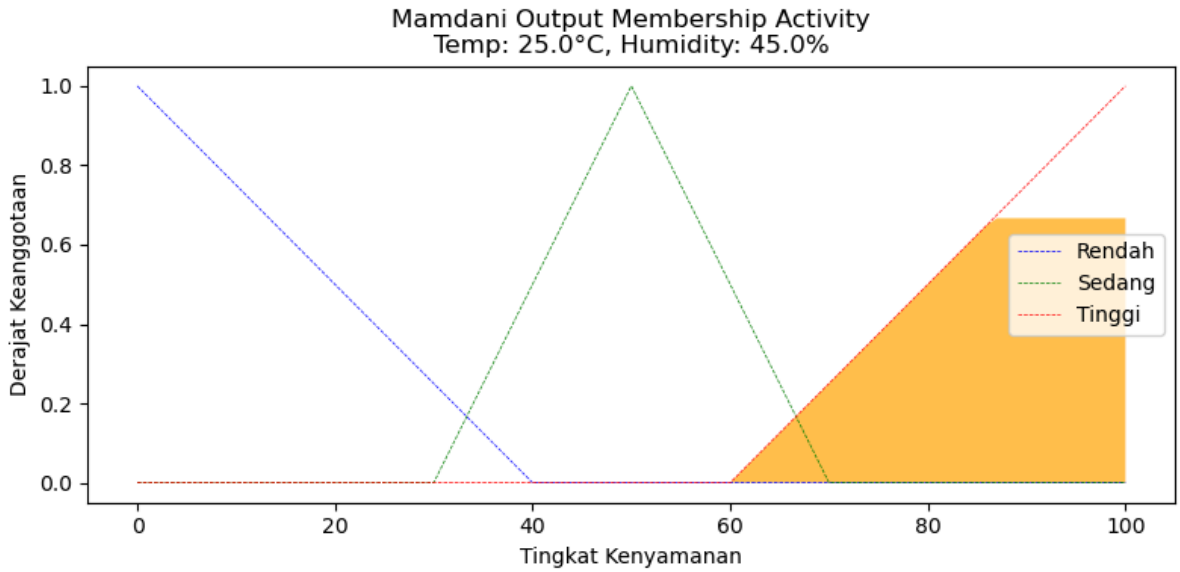
In [37]:

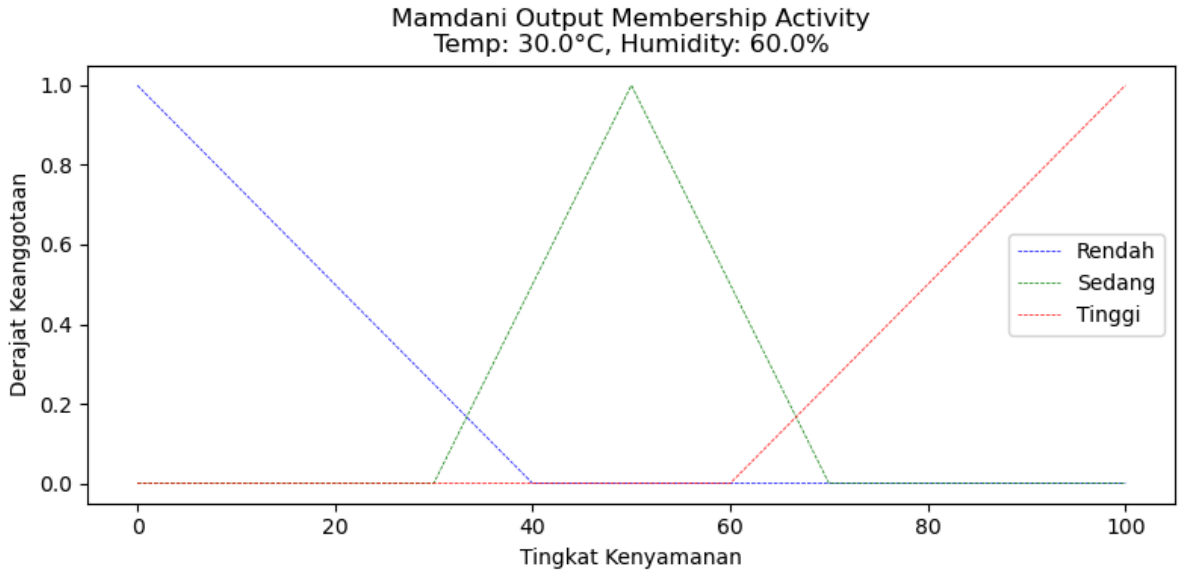
```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
import pandas as pd

# Define universe variables
x_humidity = np.arange(0, 101, 1)
x_temp = np.arange(0, 41, 1)
x_comfort = np.arange(0, 101, 1)

# Membership functions for humidity
humidity_lo = fuzz.trimf(x_humidity, [0, 0, 40])     # Rendah
humidity_md = fuzz.trimf(x_humidity, [30, 50, 70])    # Sedang
humidity_hi = fuzz.trimf(x_humidity, [60, 100, 100])  # Tinggi

# Membership functions for temperature
temp_low = fuzz.trimf(x_temp, [0, 0, 20])             # Rendah
temp_medium = fuzz.trimf(x_temp, [15, 22.5, 30])      # Sedang
temp_high = fuzz.trimf(x_temp, [25, 40, 40])          # Tinggi

# Membership functions for comfort level
comfort_lo = fuzz.trimf(x_comfort, [0, 0, 40])        # Rendah
comfort_md = fuzz.trimf(x_comfort, [30, 50, 70])       # Sedang
comfort_hi = fuzz.trimf(x_comfort, [60, 100, 100])     # Tinggi

# Define input values
input_values = [
    (18.0, 30.0),  # (Temperature, Humidity)
    (25.0, 45.0),
    (30.0, 60.0),
    (20.0, 80.0),
    (15.0, 50.0)
]
```

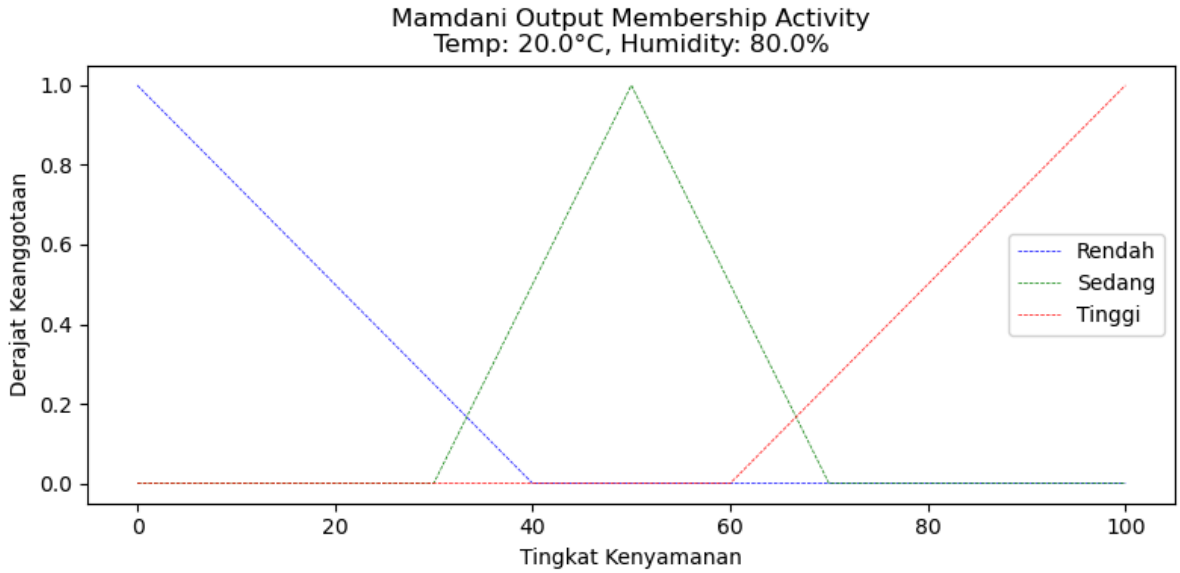
```
# Store results
results = []

# Calculate comfort level for each set of input values
for temp_value, humidity_value in input_values:
    # Calculate the degree of membership for each input value
    humidity_level_lo = fuzz.interp_membership(x_humidity, humidity_lo,
humidity_value)
    humidity_level_md = fuzz.interp_membership(x_humidity, humidity_md,
humidity_value)
    humidity_level_hi = fuzz.interp_membership(x_humidity, humidity_hi,
humidity_value)

    temp_level_low = fuzz.interp_membership(x_temp, temp_low,
temp_value)
    temp_level_medium = fuzz.interp_membership(x_temp, temp_medium,
temp_value)
    temp_level_high = fuzz.interp_membership(x_temp, temp_high,
temp_value)

    # Apply rules (Mamdani max-min inference):
    active_rule1 = np.fmin(temp_level_low, humidity_level_lo)
    comfort_activation_lo = np.fmin(active_rule1, comfort_lo)

    active_rule2 = np.fmin(temp_level_low, humidity_level_md)
    comfort_activation_md = np.fmin(active_rule2, comfort_md)

    active_rule3 = np.fmin(temp_level_medium, humidity_level_lo)
    comfort_activation_md2 = np.fmin(active_rule3, comfort_md)

    active_rule4 = np.fmin(temp_level_medium, humidity_level_md)
    comfort_activation_hi = np.fmin(active_rule4, comfort_hi)

    active_rule5 = np.fmin(temp_level_high, humidity_level_hi)
    comfort_activation_hi2 = np.fmin(active_rule5, comfort_hi)

    # Aggregating the results using max
    aggregated = np.fmax(comfort_activation_lo,
                         np.fmax(comfort_activation_md,
                                 np.fmax(comfort_activation_md2,
                                         np.fmax(comfort_activation_hi,
```

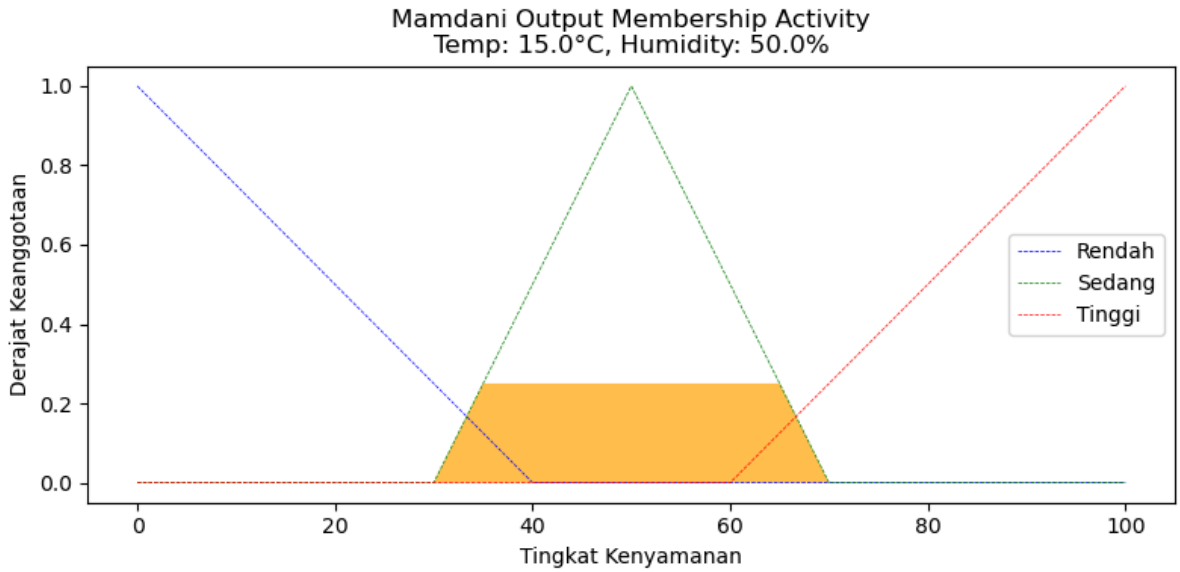
```
comfort_activation_hi2))))

    # Defuzzification (using centroid method)
    if aggregated.sum() > 0:  # Check if there is any activity
        comfort_result = fuzz.defuzz(x_comfort, aggregated, 'centroid')
    else:
        comfort_result = 0  # Default value when no activity

    results.append((temp_value, humidity_value, comfort_result))

    # Visualize the results of rule activation for the current input
    fig, ax0 = plt.subplots(figsize=(8, 4))
    ax0.fill_between(x_comfort, 0, aggregated, facecolor='orange',
alpha=0.7)
    ax0.plot(x_comfort, comfort_lo, 'b', linewidth=0.5, linestyle='--',
label='Rendah')
    ax0.plot(x_comfort, comfort_md, 'g', linewidth=0.5, linestyle='--',
label='Sedang')
    ax0.plot(x_comfort, comfort_hi, 'r', linewidth=0.5, linestyle='--',
label='Tinggi')

    ax0.set_title(f'Mamdani Output Membership Activity\nTemp:
{temp_value}°C, Humidity: {humidity_value}%')
    ax0.set_xlabel('Tingkat Kenyamanan')
    ax0.set_ylabel('Derajat Keanggotaan')
    ax0.legend()

    plt.tight_layout()
    plt.show()

# Create a DataFrame to display results
```


Mamdani Output Membership Activity
Temp: 18.0°C, Humidity: 30.0%
Derajat Keanggotaan
Tingkat Kenyamanan
Rendah
Sedang
Tinggi


Mamdani Output Membership Activity
Temp: 25.0°C, Humidity: 45.0%
Derajat Keanggotaan
Tingkat Kenyamanan
Rendah
Sedang
Tinggi


Mamdani Output Membership Activity
Temp: 30.0°C, Humidity: 60.0%
Derajat Keanggotaan
Tingkat Kenyamanan
Rendah
Sedang
Tinggi





	Temperature (°C)	Humidity (%)	Comfort Level
0	18.0	30.0	40.976090
1	25.0	45.0	85.555399
2	30.0	60.0	0.000000
3	20.0	80.0	0.000000
4	15.0	50.0	50.000000

Sugeno Inference System (Constant Output)

In Sugeno inference, we define constant outputs for each rule and use the degree of membership to compute a weighted average.

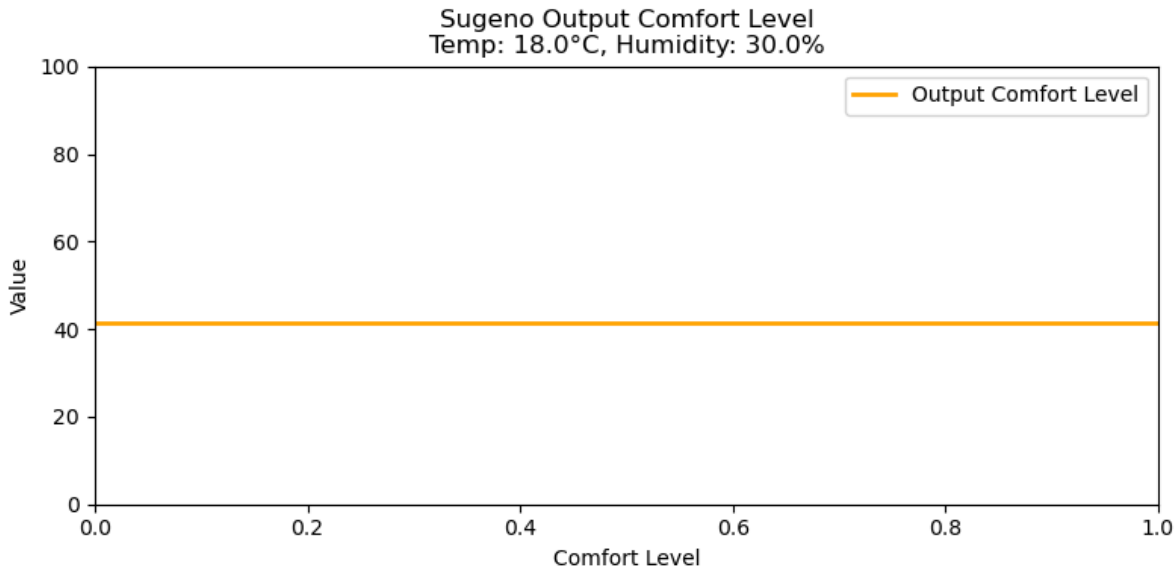
In [39]:

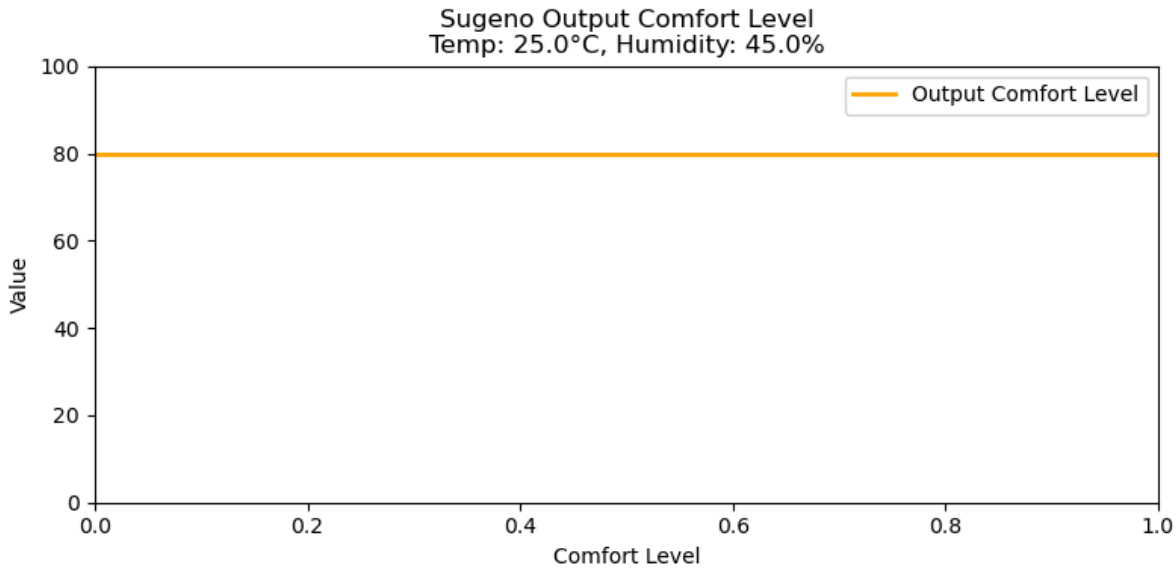
```
# Define universe variables
x_humidity = np.arange(0, 101, 1)
x_temp = np.arange(0, 41, 1)
x_comfort = np.arange(0, 101, 1)

# Membership functions for humidity
humidity_lo = fuzz.trimf(x_humidity, [0, 0, 40])      # Rendah
humidity_md = fuzz.trimf(x_humidity, [30, 50, 70])    # Sedang
```

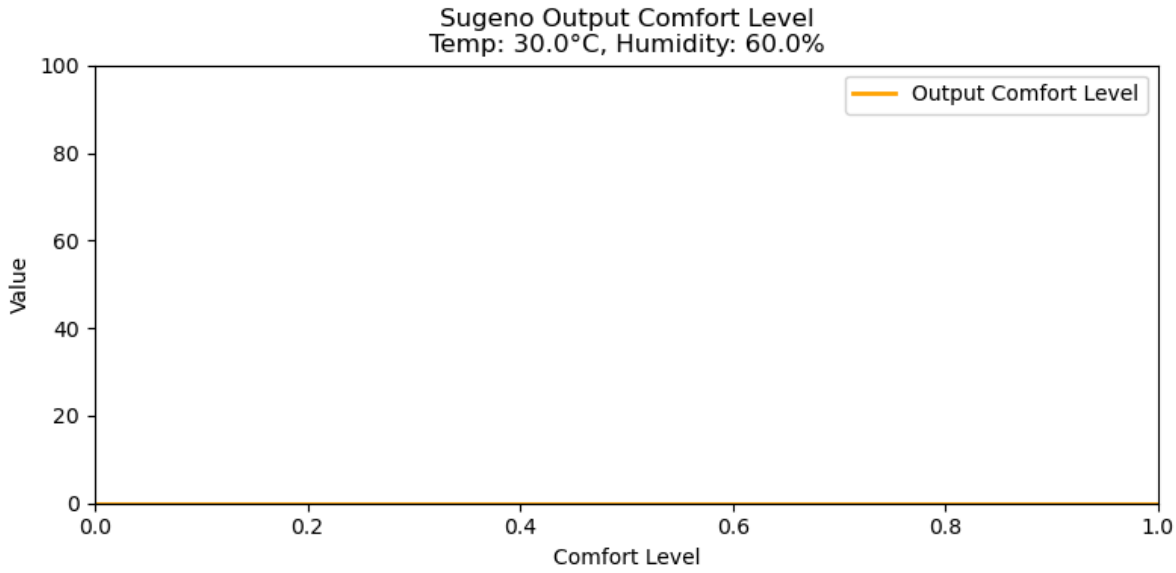
```
humidity_hi = fuzz.trimf(x_humidity, [60, 100, 100])  # Tinggi

# Membership functions for temperature
temp_low = fuzz.trimf(x_temp, [0, 0, 20])             # Rendah
temp_medium = fuzz.trimf(x_temp, [15, 22.5, 30])      # Sedang
temp_high = fuzz.trimf(x_temp, [25, 40, 40])          # Tinggi

# Define input values
input_values = [
    (18.0, 30.0),  # (Temperature, Humidity)
    (25.0, 45.0),
    (30.0, 60.0),
    (20.0, 80.0),
    (15.0, 50.0)
]

# Store results
sugeno_results = []

# Calculate comfort level for each set of input values
for temp_value, humidity_value in input_values:
    # Calculate the degree of membership for each input value
    humidity_level_lo = fuzz.interp_membership(x_humidity, humidity_lo,
humidity_value)
    humidity_level_md = fuzz.interp_membership(x_humidity, humidity_md,
humidity_value)
    humidity_level_hi = fuzz.interp_membership(x_humidity, humidity_hi,
humidity_value)

    temp_level_low = fuzz.interp_membership(x_temp, temp_low,
temp_value)
    temp_level_medium = fuzz.interp_membership(x_temp, temp_medium,
temp_value)
    temp_level_high = fuzz.interp_membership(x_temp, temp_high,
temp_value)

    # Sugeno inference
    # Sugeno constants for each rule's output
    comfort_lo_const = 20  # Low comfort level
    comfort_md_const = 50  # Medium comfort level
    comfort_hi_const = 80  # High comfort level
```

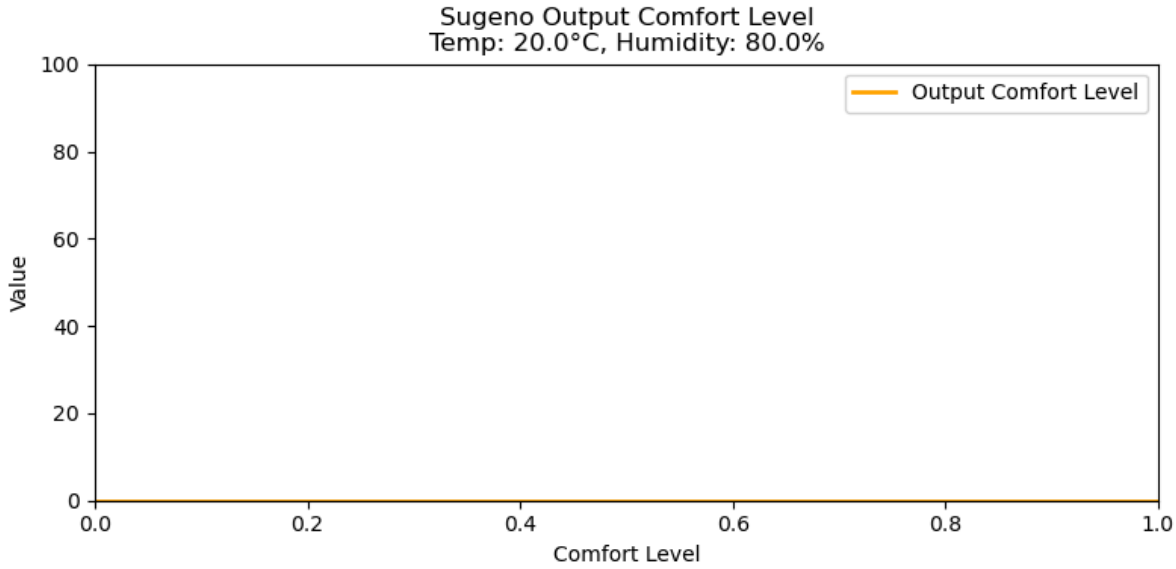
```
    # Apply rules (Sugeno inference with constant outputs):
    # Rule 1: If temperature is low and humidity is low, comfort is low
    rule1_activation = np.fmin(temp_level_low, humidity_level_lo)
    rule1_output = rule1_activation * comfort_lo_const

    # Rule 2: If temperature is low and humidity is medium, comfort is
medium
    rule2_activation = np.fmin(temp_level_low, humidity_level_md)
    rule2_output = rule2_activation * comfort_md_const

    # Rule 3: If temperature is medium and humidity is low, comfort is
medium
    rule3_activation = np.fmin(temp_level_medium, humidity_level_lo)
    rule3_output = rule3_activation * comfort_md_const

    # Rule 4: If temperature is medium and humidity is medium, comfort
is high
    rule4_activation = np.fmin(temp_level_medium, humidity_level_md)
    rule4_output = rule4_activation * comfort_hi_const

    # Rule 5: If temperature is high and humidity is high, comfort is
high
    rule5_activation = np.fmin(temp_level_high, humidity_level_hi)
    rule5_output = rule5_activation * comfort_hi_const

    # Aggregate the results (Sugeno method: weighted average of outputs)
    numerator = (rule1_output + rule2_output + rule3_output +
                 rule4_output + rule5_output)
    denominator = (rule1_activation + rule2_activation +
rule3_activation +
                   rule4_activation + rule5_activation)

    # Avoid division by zero
    if denominator == 0:
        comfort_result_sugeno = 0
    else:
        comfort_result_sugeno = numerator / denominator

    # Store results
    sugeno_results.append((temp_value, humidity_value,
comfort_result_sugeno))
```

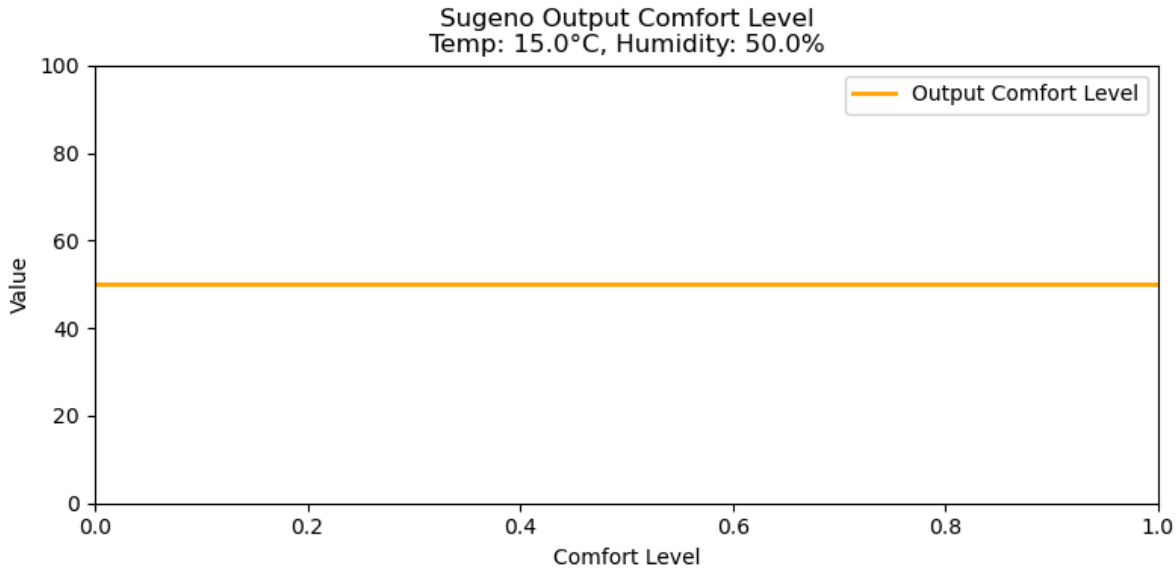
```
    # Visualize the results for the current input
    fig, ax0 = plt.subplots(figsize=(8, 4))
    ax0.axhline(y=comfort_result_sugeno, color='orange', linewidth=2,
label='Output Comfort Level')
    ax0.set_ylim(0, 100)
    ax0.set_title(f'Sugeno Output Comfort Level\nTemp: {temp_value}°C,
Humidity: {humidity_value}%')
    ax0.set_xlabel('Comfort Level')
    ax0.set_ylabel('Value')
    ax0.legend()

    plt.tight_layout()
    plt.show()

# Create a DataFrame to display Sugeno results
sugeno_results_df = pd.DataFrame(sugeno_results, columns=['Temperature
(°C)', 'Humidity (%)', 'Comfort Level'])
```











8. Comparison of Inference Systems

Input Test Cases

We will test the systems with the following input values:

- Case 1: Temperature = 18°C, Humidity = 30%

- Case 2: Temperature = 25°C, Humidity = 45%
- Case 3: Temperature = 30°C, Humidity = 60%
- Case 4: Temperature = 20°C, Humidity = 80%
- Case 5: Temperature = 15°C, Humidity = 50%

Mamdani Result

In [40]:
```
mamdani_results_df
```

Out[40]:

	Temperature (°C)	Humidity (%)	Comfort Level
0	18.0	30.0	40.976090
1	25.0	45.0	85.555399
2	30.0	60.0	0.000000
3	20.0	80.0	0.000000
4	15.0	50.0	50.000000

Sugeno Result

In [41]:
```
sugeno_results_df
```

Out[41]:

	Temperature (°C)	Humidity (%)	Comfort Level
0	18.0	30.0	41.428571
1	25.0	45.0	80.000000
2	30.0	60.0	0.000000
3	20.0	80.0	0.000000
4	15.0	50.0	50.000000

Output Comparison

The results obtained from the Mamdani and Sugeno inference systems for the given input cases are as follows:

Input Temperature (°C)	Input Humidity (%)	Mamdani Comfort Level	Sugeno Comfort Level
18.0	30.0	40.98	41.43
25.0	45.0	85.56	80.00
30.0	60.0	0.00	0.00
20.0	80.0	0.00	0.00
15.0	50.0	50.00	50.00

1. Handling Edge Cases

- **Mamdani Inference System**:

    - The Mamdani system handles edge cases effectively through its aggregation of fuzzy sets. For instance, in **Case 1** (18°C, 30% humidity), it produced a comfort level of **40.98**. This output indicates a reasonable assessment of comfort, even when the input values are on the lower end.
- **Sugeno Inference System**:
    - The Sugeno system, however, provides fixed outputs based on rule activations. For **Case 1**, it resulted in a comfort level of **41.43**. While this is similar to the Mamdani output, the constant nature of Sugeno may not capture the subtleties of the fuzzy logic as effectively.

2. Smoothness of Output Changes

- **Mamdani Inference System**:
    - The output changes in the Mamdani system are relatively gradual. For example, from **Case 1** to **Case 2**, the comfort level increased from **40.98** to **85.56**, demonstrating a smooth interpolation that aligns well with the increase in temperature and humidity.
- **Sugeno Inference System**:
    - In contrast, the Sugeno system shows more abrupt changes. The comfort level dropped to **0.00** in **Case 3** and **Case 4** (30°C, 60% humidity, and 20°C, 80% humidity, respectively). This indicates that the Sugeno system may struggle to provide nuanced outputs in certain conditions.

3. Interpretability of Rules

- **Mamdani Inference System**:
    - The Mamdani system is more interpretable due to its use of linguistic rules. The rule “If temperature is low and humidity is low, then comfort is low” translates easily into human-understandable language. The system outputs reflect natural language descriptions, making it easier for users to interpret results.
- **Sugeno Inference System**:
    - While Sugeno's rules are still interpretable, the fixed outputs can diminish clarity. For example, in **Case 2**, the output of **80.00** from a rule like "If temperature is medium and humidity is medium, then comfort is 80" is less descriptive than a fuzzy output, which can offer a range of comfort values based on varying inputs.

Conclusion

In conclusion, both the Mamdani and Sugeno inference systems exhibit distinct strengths and weaknesses based on the provided implementation results. The Mamdani system excels in handling edge cases, providing smoother output transitions, and offering high interpretability through its linguistic rules. In contrast, while the Sugeno system delivers straightforward outputs and might simplify computation, its abrupt changes and less nuanced responses may limit its applicability in certain scenarios. The choice between these

systems should be based on the specific requirements of the application, including the need for clarity, smooth output transitions, and effective handling of edge cases.