Documentation:

This script implements a basic fuzzy logic controller using the skfuzzy library to determine the appropriate action based on a given speed input. The fuzzy logic system is designed to classify the speed into categories (slow, medium, fast) and then determine whether to increase, maintain, or decrease the speed. The controller uses triangular membership functions for both speed and action, applies fuzzy rules, and performs defuzzification to generate a crisp output.

Libraries Used

- `numpy`: For numerical operations and array manipulations.
- `skfuzzy`: For fuzzy logic operations including membership functions, rule application, and defuzzification.
- `matplotlib`: For plotting the membership functions and the resulting actions.

Import Libraries

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

- `numpy`: A fundamental package for scientific computing with Python.
- `skfuzzy`: A Python library for fuzzy logic.
- `matplotlib.pyplot`: A plotting library used for creating static, animated, and interactive visualizations in Python.

Define the Speed Range

```
x_speed = np.arange(0, 101, 1)
```

- `x_speed`: An array representing the speed range from 0 to 100 km/h with a step of 1.

Define the Membership Functions for Speed

```
slow = fuzz.trimf(x_speed, [0, 0, 40])
medium = fuzz.trimf(x_speed, [30, 50, 70])
fast = fuzz.trimf(x_speed, [50, 80, 100])
```

- `slow`, `medium`, `fast`: Triangular membership functions defining "Slow", "Medium", and "Fast" speeds.

Plot the Membership Functions for Speed

```
plt.figure()
plt.plot(x_speed, slow, 'b', label='Slow')
plt.plot(x_speed, medium, 'g', label='Medium')
plt.plot(x_speed, fast, 'r', label='Fast')
plt.title('Speed Membership Functions')
plt.xlabel('Speed (km/h)')
plt.ylabel('Membership Degree')
plt.legend()
plt.show()
```

- This section creates a plot for the speed membership functions and displays it.

Define the Output Actions

```
x_action = np.arange(0, 101, 1)
increase_speed = fuzz.trimf(x_action, [0, 0, 50])
maintain_speed = fuzz.trimf(x_action, [30, 50, 70])
decrease_speed = fuzz.trimf(x_action, [50, 100, 100])
```

- `x_action`: An array representing the action range from 0 to 100 with a step of 1.
- `increase_speed`, `maintain_speed`, `decrease_speed`: Triangular membership functions defining actions to "Increase Speed", "Maintain Speed", and "Decrease Speed".

Assume an Input Speed

```
input_speed = 55
```

- `input_speed`: An assumed input speed of 55 km/h.

Fuzzify the Input Speed

```
speed_level_slow = fuzz.interp_membership(x_speed, slow, input_speed)
speed_level_medium = fuzz.interp_membership(x_speed, medium,
input_speed)
speed_level_fast = fuzz.interp_membership(x_speed, fast, input_speed)
```

- `speed_level_slow`, `speed_level_medium`, `speed_level_fast`: The degree to which the input speed belongs to each of the "Slow", "Medium", and "Fast" categories.

Apply the Fuzzy Rules

```
rule1 = np.fmin(speed_level_slow, increase_speed)
rule2 = np.fmin(speed_level_medium, maintain_speed)
rule3 = np.fmin(speed_level_fast, decrease_speed)
```

- `rule1`, `rule2`, `rule3`: The output actions based on the fuzzy rules applied to the input speed.

Aggregate the Rules

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

- `aggregated`: The combined result of the fuzzy rules.

Defuzzify the Result

```
action = fuzz.defuzz(x_action, aggregated, 'centroid')
action_activation = fuzz.interp_membership(x_action, aggregated,
action)
```

- `action`: The defuzzified result representing the suggested action.
- `action_activation`: The degree to which the suggested action belongs to the aggregated output actions.

Print the Result

```
print(f"Input speed: {input_speed} km/h")print(f"Action:
{action:.2f} (0-50: Increase speed, 50-70: Maintain speed, 70-100:
Decrease speed)")
```

- This section prints the input speed and the suggested action.

Plot the Result

```
plt.figure()
plt.plot(x_action, increase_speed, 'b', label='Increase speed')
plt.plot(x_action, maintain_speed, 'g', label='Maintain speed')
plt.plot(x_action, decrease_speed, 'r', label='Decrease speed')
plt.fill_between(x_action, 0, aggregated, color='orange', alpha=0.7)
plt.plot([action, action], [0, action_activation], 'k', linestyle='-
-', linewidth=1.5)
plt.title('Output Action Membership Functions')
plt.xlabel('Action')
plt.ylabel('Membership Degree')
plt.legend()
plt.show()
```

- This section creates a plot for the action membership functions and the aggregated result, and displays it.

The code uses fuzzy logic to determine the appropriate action based on an input speed, and visualizes both the membership functions and the resulting action.