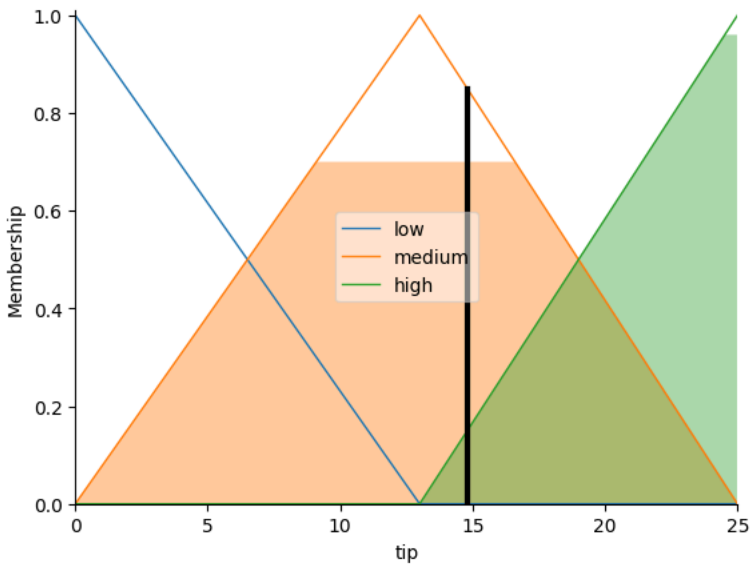
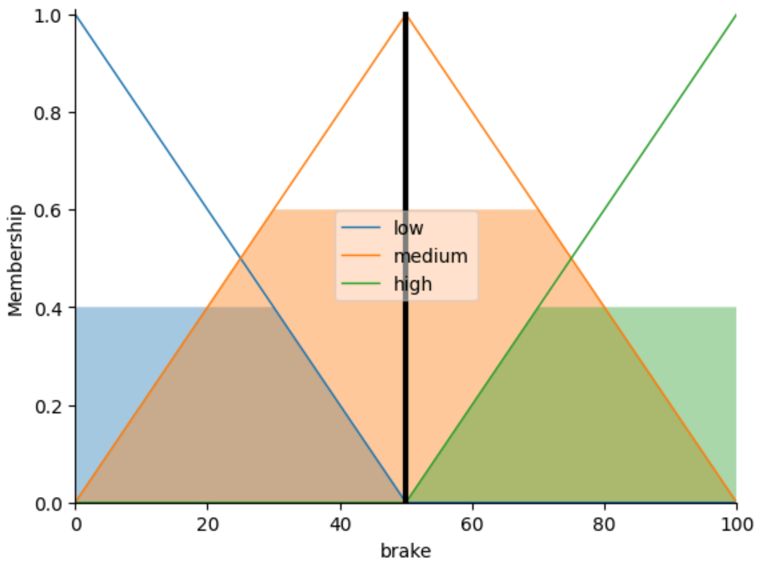
```
!pip install scikit-fuzzy
```

```
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.12/dist-packages (0.5.0)
```

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Restaurant Tipping System
def restaurant_tipping():
    # Define input and output variables
    service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
    food = ctrl.Antecedent(np.arange(0, 11, 1), 'food')
    tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

    # Define membership functions
    service.automf(3)
    food.automf(3)
    tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
    tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
    tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

    # Define fuzzy rules
    rule1 = ctrl.Rule(service['poor'] | food['poor'], tip['low'])
    rule2 = ctrl.Rule(service['average'], tip['medium'])
    rule3 = ctrl.Rule(service['good'] | food['good'], tip['high'])

    # Create control system
    tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
    tipping = ctrl.ControlSystemSimulation(tipping_ctrl)

    # Input values
    tipping.input['service'] = 6.5
    tipping.input['food'] = 9.8

    # Compute the result
    tipping.compute()

    # Print and visualize the result
    print(f"Tip: {tipping.output['tip']:.2f}")
    tip.view(sim=tipping)

# ABS Brake System
def abs_brake_system():
    # Define input and output variables
    speed = ctrl.Antecedent(np.arange(0, 101, 1), 'speed')
    slip = ctrl.Antecedent(np.arange(0, 11, 1), 'slip')
    brake = ctrl.Consequent(np.arange(0, 101, 1), 'brake')

    # Define membership functions
    speed.automf(3)
    slip.automf(3)
    brake['low'] = fuzz.trimf(brake.universe, [0, 0, 50])
    brake['medium'] = fuzz.trimf(brake.universe, [0, 50, 100])
    brake['high'] = fuzz.trimf(brake.universe, [50, 100, 100])

    # Define fuzzy rules
    rule1 = ctrl.Rule(speed['poor'] | slip['poor'], brake['high'])
    rule2 = ctrl.Rule(speed['average'], brake['medium'])
    rule3 = ctrl.Rule(speed['good'] | slip['good'], brake['low'])

    # Create control system
    braking_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
    braking = ctrl.ControlSystemSimulation(braking_ctrl)

    # Input values
    braking.input['speed'] = 70
    braking.input['slip'] = 3

    # Compute the result
    braking.compute()

    # Print and visualize the result
    print(f"Brake: {braking.output['brake']:.2f}")
    brake.view(sim=braking)

# Run the systems
restaurant_tipping()
abs_brake_system()
plt.show()
```

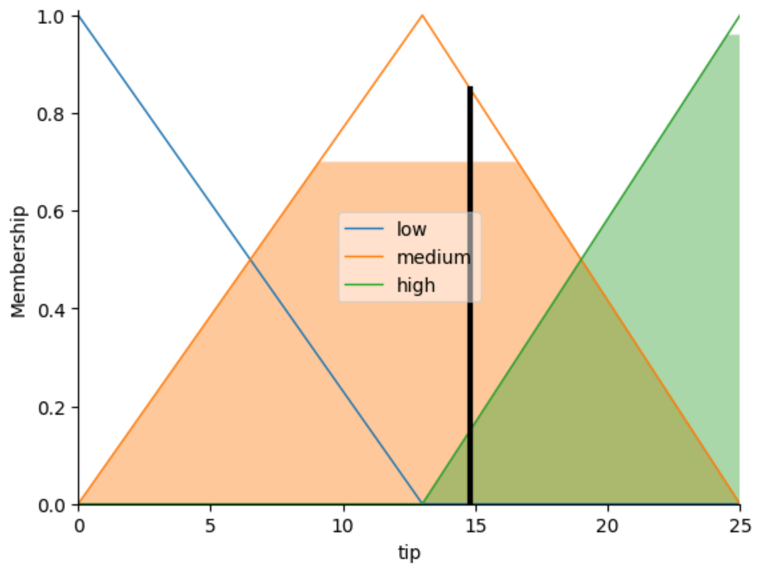
```
Tip: 14.80
Brake: 50.00
```

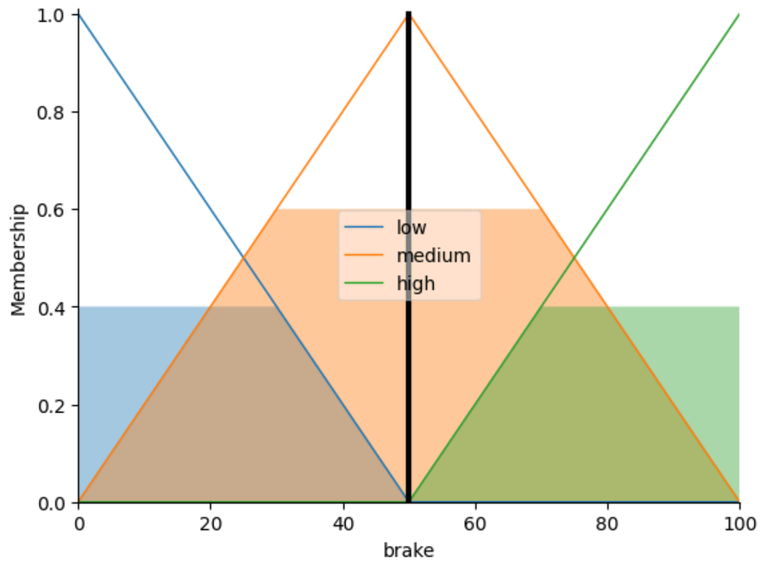




```
!pip install scikit-fuzzy

import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# Restaurant Tipping System
def restaurant_tipping():
    # Define input and output variables
    service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
    food = ctrl.Antecedent(np.arange(0, 11, 1), 'food')
    tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

    # Define membership functions
    service.automf(3)
    food.automf(3)
    tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
    tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
    tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])
    # Define fuzzy rules
    rule1 = ctrl.Rule(service['poor'] | food['poor'], tip['low'])
    rule2 = ctrl.Rule(service['average'], tip['medium'])
    rule3 = ctrl.Rule(service['good'] | food['good'], tip['high'])

    # Create control system
    tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
    tipping = ctrl.ControlSystemSimulation(tipping_ctrl)

    # Input values
    tipping.input['service'] = 6.5
    tipping.input['food'] = 9.8

    # Compute the result
    tipping.compute()

    # Print and visualize the result
    print(f"Tip: {tipping.output['tip']:.2f}")
    tip.view(sim=tipping)

# ABS Brake System
def abs_brake_system():
    # Define input and output variables
    speed = ctrl.Antecedent(np.arange(0, 101, 1), 'speed')
    slip = ctrl.Antecedent(np.arange(0, 11, 1), 'slip')
    brake = ctrl.Consequent(np.arange(0, 101, 1), 'brake')

    # Define membership functions
    speed.automf(3)
    slip.automf(3)
    brake['low'] = fuzz.trimf(brake.universe, [0, 0, 50])
    brake['medium'] = fuzz.trimf(brake.universe, [0, 50, 100])
    brake['high'] = fuzz.trimf(brake.universe, [50, 100, 100])

    # Define fuzzy rules
    rule1 = ctrl.Rule(speed['poor'] | slip['poor'], brake['high'])
    rule2 = ctrl.Rule(speed['average'], brake['medium'])
    rule3 = ctrl.Rule(speed['good'] | slip['good'], brake['low'])

    # Create control system
    braking_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
    braking = ctrl.ControlSystemSimulation(braking_ctrl)

    # Input values
    braking.input['speed'] = 70
    braking.input['slip'] = 3

    # Compute the result
    braking.compute()

    # Print and visualize the result
    print(f"Brake: {braking.output['brake']:.2f}")
    brake.view(sim=braking)

# Run the systems
restaurant_tipping()
abs_brake_system()
plt.show()
```

```
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.12/dist-packages (0.5.0)
Tip: 14.80
Brake: 50.00
```

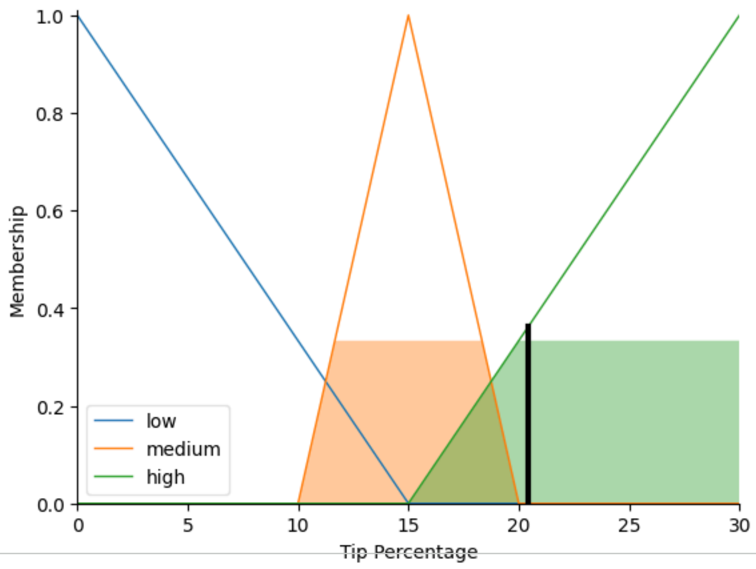




```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt

# --- 1. Restaurant Tipping System (Enhanced Rules and MFs) ---
def enhanced_restaurant_tipping():
    """
    Implements a fuzzy logic controller for calculating a restaurant tip
    based on service and food quality.
    """
    print("--- 1. Restaurant Tipping System (Enhanced) ---")

    # Define input and output variables
    # Service and Food are rated on a 0-10 scale
    service = ctrl.Antecedent(np.arange(0, 11, 1), 'Service Quality')
    food = ctrl.Antecedent(np.arange(0, 11, 1), 'Food Quality')
    # Tip is a percentage of the total bill, 0-30%
    tip = ctrl.Consequent(np.arange(0, 31, 1), 'Tip Percentage')

    # --- Define Custom Membership Functions (MFs) ---

    # Service MFs (Trapezoidal/Triangular for better definition)
    service['poor'] = fuzz.trapmf(service.universe, [0, 0, 2, 5])
    service['decent'] = fuzz.trimf(service.universe, [3, 6, 9])
    service['excellent'] = fuzz.trapmf(service.universe, [7, 10, 10, 10])
```

```
        # Food MFs
        food['bad'] = fuzz.trapmf(food.universe, [0, 0, 3, 5])
        food['good'] = fuzz.trapmf(food.universe, [4, 7, 10, 10])

        # Tip MFs
        tip['low'] = fuzz.trimf(tip.universe, [0, 0, 15])      # 0% - 15%
        tip['medium'] = fuzz.trimf(tip.universe, [10, 15, 20]) # 10% - 20%
        tip['high'] = fuzz.trimf(tip.universe, [15, 30, 30])    # 15% - 30%

        # --- Define Fuzzy Rules (Inference Engine) ---
        # We use more nuanced rules to cover edge cases.

        # R1: If service is poor OR food is bad, tip is low. (Base minimum)
        rule1 = ctrl.Rule(service['poor'] | food['bad'], tip['low'])

        # R2: If service is decent, tip is medium. (Standard expectation)
        rule2 = ctrl.Rule(service['decent'], tip['medium'])

        # R3: If service is excellent AND food is good, tip is high. (Reward)
        rule3 = ctrl.Rule(service['excellent'] & food['good'], tip['high'])

        # R4: If service is excellent BUT food is bad, tip is still medium (compensating server)
        rule4 = ctrl.Rule(service['excellent'] & food['bad'], tip['medium'])

        # R5: If service is poor BUT food is good, tip is medium (compensating kitchen)
        rule5 = ctrl.Rule(service['poor'] & food['good'], tip['medium'])

        # Create control system and simulation
        tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5])
        tipping_sim = ctrl.ControlSystemSimulation(tipping_ctrl)

        # --- Input Values and Computation ---

        # Example: Good service (8.0), Decent Food (7.5)
        tipping_sim.input['Service Quality'] = 8.0
        tipping_sim.input['Food Quality'] = 7.5

        # Compute the result (Defuzzification)
        try:
            tipping_sim.compute()

            # Print and visualize the result
            calculated_tip = tipping_sim.output['Tip Percentage']
            print(f"Service Score: 8.0, Food Score: 7.5")
            print(f"Recommended Tip: {calculated_tip:.2f}%\n")

            # Visualize the result
            tip.view(sim=tipping_sim)

        except Exception as e:
            print(f"Error during tipping calculation: {e}")

    # --- 2. ABS Brake System (Enhanced Rules and MFs) ---
    def enhanced_abs_brake_system():
        """
        Implements a fuzzy logic controller for determining brake pressure
        based on vehicle speed and wheel slip, simulating an ABS system.
        """
        print("--- 2. ABS Brake System (Enhanced) ---")

        # Define input and output variables
        speed = ctrl.Antecedent(np.arange(0, 101, 1), 'Vehicle Speed (km/h)')
        slip = ctrl.Antecedent(np.arange(0, 11, 1), 'Wheel Slip Ratio (0-10)')
        brake = ctrl.Consequent(np.arange(0, 101, 1), 'Brake Pressure (%)')

        # --- Define Custom Membership Functions (MFs) ---

        # Speed MFs
        speed['slow'] = fuzz.trimf(speed.universe, [0, 0, 50])
        speed['medium'] = fuzz.trimf(speed.universe, [30, 60, 90])
        speed['fast'] = fuzz.trimf(speed.universe, [70, 100, 100])

        # Slip MFs (Critical for ABS: 0.2 is ideal, >0.2 is high risk)
        slip['low'] = fuzz.trimf(slip.universe, [0, 0, 2])       # Good grip, <20% slip
        slip['optimal'] = fuzz.trimf(slip.universe, [1, 4, 6])   # Ideal braking range
        slip['high'] = fuzz.trimf(slip.universe, [5, 10, 10])    # Skidding, >50% slip
        # Brake Pressure MFs
        brake['none'] = fuzz.trimf(brake.universe, [0, 0, 25])
        brake['moderate'] = fuzz.trimf(brake.universe, [20, 50, 80])
        brake['full'] = fuzz.trimf(brake.universe, [75, 100, 100])

        # --- Define Fuzzy Rules (Inference Engine) ---

        # R1: If speed is fast AND slip is low/optimal, apply full brake (Safe aggressive stop)
        rule1 = ctrl.Rule(speed['fast'] & (slip['low'] | slip['optimal']), brake['full'])

        # R2: If speed is medium AND slip is low/optimal, apply moderate brake (Normal stop)
        rule2 = ctrl.Rule(speed['medium'] & (slip['low'] | slip['optimal']), brake['moderate'])

        # R3: If slip is high (skidding), release brake (none) to regain traction. (ABS Core)
        rule3 = ctrl.Rule(slip['high'], brake['none'])

        # R4: If speed is slow, brake moderately regardless of minor slip.
        rule4 = ctrl.Rule(speed['slow'], brake['moderate'])

        # Create control system and simulation
        braking_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4])
        braking_sim = ctrl.ControlSystemSimulation(braking_ctrl)

        # --- Input Values and Computation ---

        # Example: High Speed (80 km/h), Medium Slip (5.0) -> Should result in lower pressure to recover
        braking_sim.input['Vehicle Speed (km/h)'] = 80
        braking_sim.input['Wheel Slip Ratio (0-10)'] = 5.0 # High slip

        # Compute the result (Defuzzification)
        try:
            braking_sim.compute()

            # Print and visualize the result
            calculated_brake = braking_sim.output['Brake Pressure (%)']
            print(f"Speed: 80 km/h, Slip: 5.0")
            print(f"Recommended Brake Pressure: {calculated_brake:.2f}%\n")

            # Visualize the result
            brake.view(sim=braking_sim)

        except Exception as e:
            print(f"Error during braking calculation: {e}")


    # --- Run the enhanced systems ---

    # Run the systems
    enhanced_restaurant_tipping()
    enhanced_abs_brake_system()

    # Display all plots
    plt.show()
```

```
--- 1. Restaurant Tipping System (Enhanced) ---
Service Score: 8.0, Food Score: 7.5
Recommended Tip: 20.42%

--- 2. ABS Brake System (Enhanced) ---
Speed: 80 km/h, Slip: 5.0
Recommended Brake Pressure: 61.45%
```


Membership
Tip Percentage
low
medium
high