Aim - To implement Fuzzy Set Properties in Python.

Theory -

What is fuzzy logic?

Fuzzy logic is a type of logic that allows for degrees of truth, rather than strict binary (true or false) values. It's designed to handle imprecise or uncertain information by assigning degrees of membership to different categories. This allows for more nuanced decision-making in situations where traditional binary logic might be inadequate.

Here's a simple **example**: Consider a thermostat controlling a room's temperature. In binary logic, the temperature is either "hot" or "cold" based on a fixed threshold. However, in reality, the room temperature can be warm but not excessively hot, or cool but not freezing cold. Fuzzy logic helps capture this by allowing terms like "warm," "slightly warm," "slightly cold," and "cold" to have varying degrees of truth.

For instance, if we use fuzzy logic to control the thermostat, the rules might look like this:

1. If the temperature is "very cold," then increase the heater's intensity significantly.
2. If the temperature is "cold," then increase the heater's intensity moderately.
3. If the temperature is "warm," then maintain the current heater intensity.
4. If the temperature is "hot," then decrease the heater's intensity moderately.
5. If the temperature is "very hot," then decrease the heater's intensity significantly.

This allows the thermostat to make more gradual and accurate adjustments based on the degree to which the temperature matches each category. So, if the room is slightly cold, the thermostat would increase the heater's intensity slightly, rather than abruptly switching it from off to full blast, as binary logic might do.

Explain the fuzzy set properties.

1. Commutativity: Fuzzy set operations are commutative, which means the order of operands doesn't affect the result. For example, if you're taking the union of fuzzy sets A and B (A ∪ B), it's the same as taking the union of B and A (B ∪ A).

2. Associativity: Fuzzy set operations are associative, meaning that the grouping of operands doesn't influence the outcome. For instance, when you're using intersection (∩) or union (∪) operations, it doesn't matter in which order you group the sets – the result remains the same.

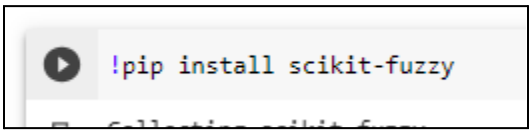
3. Distributivity: Fuzzy set operations follow distributive properties. This means that operations like union and intersection distribute over each other. For example, (A ∪ B) ∩ C is equal to (A ∩ C) ∪ (B ∩ C), and (A ∩ B) ∪ C is equal to (A ∪ C) ∩ (B ∪ C).

4. Idempotency: Fuzzy set operations are idempotent, which means that applying an operation multiple times doesn't change the result after the first application. For example, taking the union of a set with itself (A ∪ A) doesn't alter the set.

5. Transitivity: Transitivity is a property of relations between sets and elements. While not as directly applicable to fuzzy set operations, it's still relevant in terms of how fuzzy sets relate to each other in some contexts, like in fuzzy relations and fuzzy inference.

Code -

Installing fuzzy library



Code 1 - Commutativity, Associativity, Distributivity, Idempotency.

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

# Create a range of values for the x-axis
x = np.arange(0, 11, 1)

# Create membership functions
low = fuzz.trimf(x, [0, 0, 5])
medium = fuzz.trimf(x, [0, 5, 10])
```

```
high = fuzz.trimf(x, [5, 10, 10])

# Fuzzy logic operations
def fuzzy_commutativity(A, B):
    return np.minimum(A, B), np.minimum(B, A)

def fuzzy_associativity(A, B, C):
    left = np.minimum(np.minimum(A, B), C)
    right = np.minimum(A, np.minimum(B, C))
    return left, right

def fuzzy_distributivity(A, B, C):
    left = np.maximum(np.minimum(A, B), np.minimum(A, C))
    right = np.minimum(A, np.maximum(B, C))
    return left, right

def fuzzy_idempotency(A):
    return A, A

# Apply fuzzy logic operations to membership functions
result1, result2 = fuzzy_commutativity(low, medium)
result3, result4 = fuzzy_associativity(low, medium, high)
result5, result6 = fuzzy_distributivity(low, medium, high)
result7, result8 = fuzzy_idempotency(medium)

# Plot the membership functions and results
plt.figure(figsize=(10, 8))

plt.subplot(3, 3, 1)
plt.plot(x, low, 'b', linewidth=1.5, label='Low')
plt.plot(x, medium, 'g', linewidth=1.5, label='Medium')
plt.title('Membership Functions')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 3, 2)
plt.plot(x, result1, 'r', linewidth=1.5, label='A ∩ B')
plt.plot(x, result2, 'c', linewidth=1.5, label='B ∩ A')
plt.title('Commutativity')
```

```
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 3, 3)
plt.plot(x, result3, 'r', linewidth=1.5, label='(A ∩ B) ∩ C')
plt.plot(x, result4, 'c', linewidth=1.5, label='A ∩ (B ∩ C)')
plt.title('Associativity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 3, 4)
plt.plot(x, result5, 'r', linewidth=1.5, label='A ∩ (B ∪ C)')
plt.plot(x, result6, 'c', linewidth=1.5, label='(A ∩ B) ∪ (A ∩ C)')
plt.title('Distributivity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 3, 5)
plt.plot(x, result7, 'r', linewidth=1.5, label='Medium')
plt.plot(x, result8, 'c', linewidth=1.5, label='Medium')
plt.title('Idempotency / Tautology')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

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

Code 2 - Commutativity, Associativity, Distributivity, Idempotency, Transitivity.

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

# Create a range of values for the x-axis
x = np.arange(0, 11, 1)

# Create membership functions
```

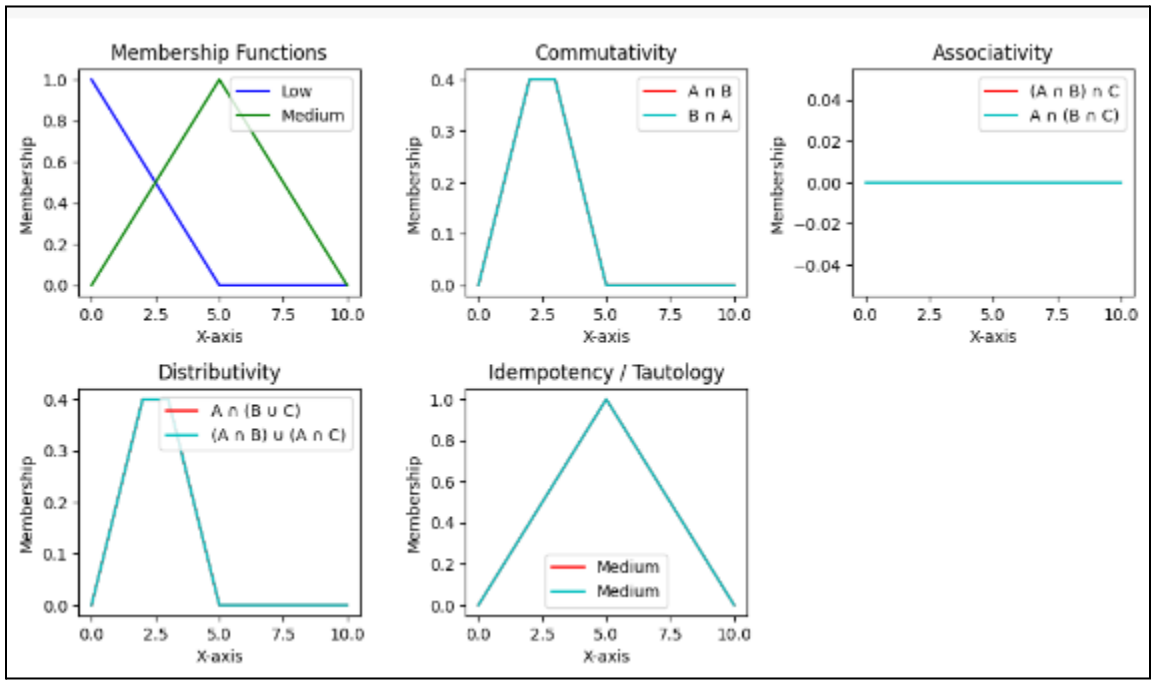
```
low = fuzz.trimf(x, [0, 0, 5])
medium = fuzz.trimf(x, [0, 5, 10])
high = fuzz.trimf(x, [5, 10, 10])

# Fuzzy logic operations
def fuzzy_commutativity(A, B):
    return np.minimum(A, B), np.minimum(B, A)

def fuzzy_associativity(A, B, C):
    left = np.minimum(np.minimum(A, B), C)
    right = np.minimum(A, np.minimum(B, C))
    return left, right

def fuzzy_distributivity(A, B, C):
    left = np.maximum(np.minimum(A, B), np.minimum(A, C))
    right = np.minimum(A, np.maximum(B, C))
    return left, right

def fuzzy_idempotency(A):
    return A, A

def fuzzy_transitivity(A, B, C):
    left = np.minimum(A, B)
    right = C
    return left, right

# Apply fuzzy logic operations to membership functions
result1, result2 = fuzzy_commutativity(low, medium)
result3, result4 = fuzzy_associativity(low, medium, high)
result5, result6 = fuzzy_distributivity(low, medium, high)
result7, result8 = fuzzy_idempotency(medium)
result9, result10 = fuzzy_transitivity(low, medium, high)

# Plot the membership functions and results
plt.figure(figsize=(10, 8))

plt.subplot(3, 4, 1)
plt.plot(x, low, 'b', linewidth=1.5, label='Low')
plt.plot(x, medium, 'g', linewidth=1.5, label='Medium')
plt.plot(x, high, 'r', linewidth=1.5, label='High')
```

```
plt.title('Membership Functions')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 4, 2)
plt.plot(x, result1, 'r', linewidth=1.5, label='A ∩ B')
plt.plot(x, result2, 'c', linewidth=1.5, label='B ∩ A')
plt.title('Commutativity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 4, 3)
plt.plot(x, result3, 'r', linewidth=1.5, label='(A ∩ B) ∩ C')
plt.plot(x, result4, 'c', linewidth=1.5, label='A ∩ (B ∩ C)')
plt.title('Associativity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 4, 4)
plt.plot(x, result5, 'r', linewidth=1.5, label='A ∩ (B ∪ C)')
plt.plot(x, result6, 'c', linewidth=1.5, label='(A ∩ B) ∪ (A ∩ C)')
plt.title('Distributivity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

plt.subplot(3, 4, 5)
plt.plot(x, result7, 'r', linewidth=1.5, label='Medium')
plt.plot(x, result8, 'c', linewidth=1.5, label='Medium')
plt.title('Idempotency / Tautology')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()


plt.subplot(3, 4, 6)
plt.plot(x, result9, 'r', linewidth=1.5, label='A ∩ B')
```

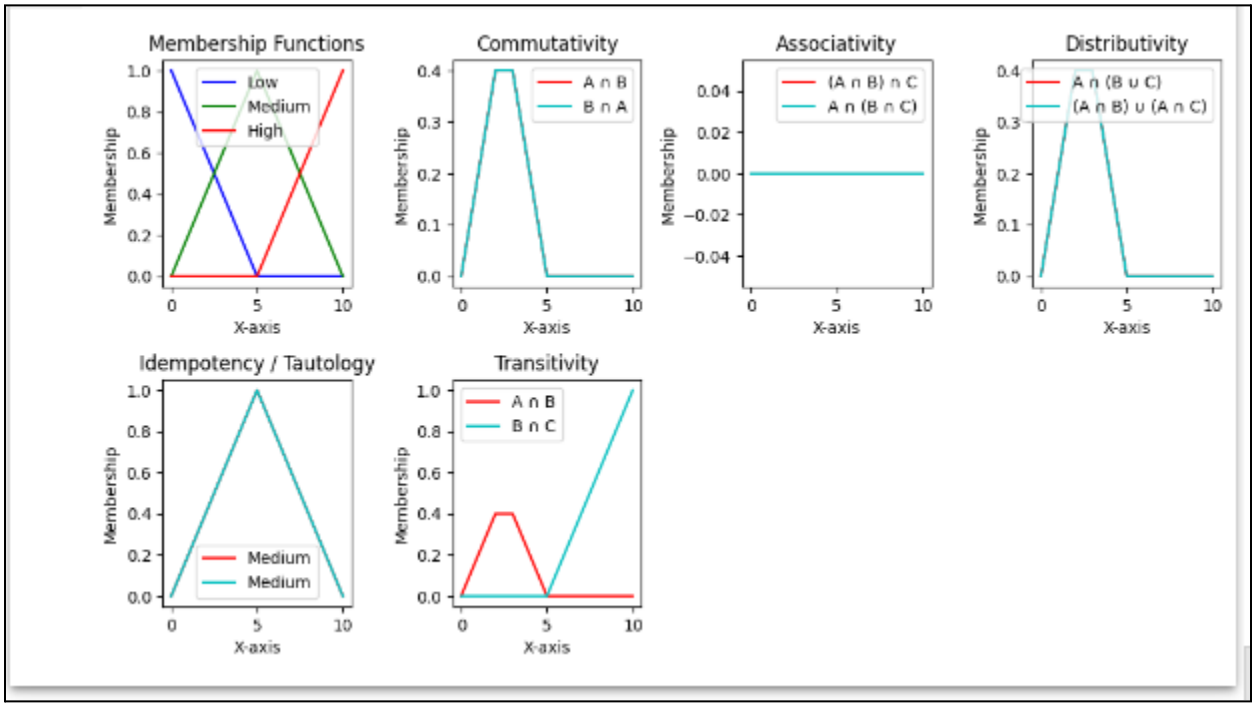
```
plt.plot(x, result10, 'c', linewidth=1.5, label='B ∩ C')
plt.title('Transitivity')
plt.xlabel('X-axis')
plt.ylabel('Membership')
plt.legend()

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

Output -

Output 1 -



Output 2 -



Conclusion - Thus, we can conclude that we have successfully studied the properties of fuzzy sets and implemented them in python.