1. Importing Necessary Libraries

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

- **numpy:** Used for numerical operations.
- **scikit-fuzzy (skfuzzy):** A fuzzy logic toolkit for Python.
- **matplotlib.pyplot:** Used for plotting graphs.

2. Defining Fuzzy Variables

```
price_trend = ctrl.Antecedent(np.arange(0,11,1), 'price_trend')#Stock
price trend
volume = ctrl.Antecedent(np.arange(0,11,1),'volume')# Volume of trade
sentiment = ctrl.Antecedent(np.arange(0,11,1),'sentiment')#Market
sentiment
decision = ctrl.Consequent(np.arange(0,101,1),'decision')# Investment
decision
```

- `price_trend`, `volume`, and `sentiment` are the input variables (antecedents).
- `decision` is the output variable (consequent).

3. Defining Membership Functions

Membership functions describe how each input is mapped to a degree of membership between 0 and 1.

For **price_trend**:

```
price_trend['downtrend']=fuzz.trimf(price_trend.universe,[0,0,5])
price_trend['stable'] = fuzz.trimf(price_trend.universe,[0,5,10])
price_trend['uptrend']=fuzz.trimf(price_trend.universe,[5,10,10])
```

- **downtrend:** Triangular membership function representing a decreasing trend.
- **stable**: Triangular membership function representing a stable trend.
- **uptrend:** Triangular membership function representing an increasing trend.

For volume:

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

- **low:** Triangular membership function for low trade volume.
- **medium:** Triangular membership function for medium trade volume.
- **high:** Triangular membership function for high trade volume.

For sentiment:

```
sentiment['negative'] = fuzz.trimf(sentiment.universe, [0, 0, 5])
sentiment['neutral'] = fuzz.trimf(sentiment.universe, [0, 5, 10])
sentiment['positive'] = fuzz.trimf(sentiment.universe, [5,0,10])
```

- **`negative`:** Triangular membership function for negative market sentiment.
- **`neutral`:** Triangular membership function for neutral market sentiment.
- **`positive`:** Triangular membership function for positive market sentiment.

For **`decision`:**

```
decision['sell'] = fuzz.trimf(decision.universe, [0, 0, 50])
decision['hold'] = fuzz.trimf(decision.universe, [0, 50, 100])
decision['buy'] = fuzz.trimf(decision.universe, [50, 100, 100])
```

- **`sell`:** Triangular membership function indicating a sell decision.
- **`hold`:** Triangular membership function indicating a hold decision.
- **`buy`:** Triangular membership function indicating a buy decision.

4. Defining Fuzzy Rules

Fuzzy rules define the logic of the decision-making process.

```
rule1 = ctrl.Rule(price_trend['uptrend'] & volume['high'] &
sentiment['positive'], decision['buy'])
rule2 = ctrl.Rule(price_trend['downtrend'] & volume['high'] &
sentiment['negative'], decision['sell'])
rule3 = ctrl.Rule(price_trend['stable'] | volume['medium'] |
sentiment['neutral'], decision['hold'])
```

- **`rule1`**: If the price trend is uptrend, the volume is high, and the sentiment is positive, then the decision is to buy.
- **`rule2`:** If the price trend is downtrend, the volume is high, and the sentiment is negative, then the decision is to sell.
- **`rule3`:** If the price trend is stable, or the volume is medium, or the sentiment is neutral, then the decision is to hold.

5. Creating Control System

```
investment_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
investment_sim = ctrl.ControlSystemSimulation(investment_ctrl)
```

- **`investment_ctrl`**: Combines the fuzzy rules into a control system.
- **`investment_sim`:** Creates a simulation environment for the control system.

6. Providing Input Values and Computing the Decision

```
investment_sim.input['price_trend'] = 7  # Uptrend
investment_sim.input['volume'] = 8  # High
investment_sim.input['sentiment'] = 9  # Positive

# Compute result
investment_sim.compute()
print(f"Investment Decision:
{investment_sim.output['decision']}")
```

- Sets input values for `price_trend`, `volume`, and `sentiment`.
- Computes the fuzzy logic output and prints the investment decision.

7. Plotting the Result

```
price_trend.view(sim=investment_sim)
plt.subplots_adjust(top=0.9, bottom=0.1, left=0.1, right=0.9,
hspace=1.5)
plt.show()

volume.view(sim=investment_sim)
plt.subplots_adjust(top=0.9, bottom=0.1, left=0.1, right=0.9,
hspace=1.5)
plt.show()

sentiment.view(sim=investment_sim)
plt.subplots_adjust(top=0.9, bottom=0.1, left=0.1, right=0.9,
hspace=1.5)
plt.show()

decision.view(sim=investment_sim)
plt.subplots_adjust(top=0.9, bottom=0.1, left=0.1, right=0.9,
hspace=1.5)
plt.show()
```

- Plots the membership functions and the decision for `price_trend`, `volume`, and `sentiment` based on the input values.

The code above creates a fuzzy logic system to make investment decisions based on stock price trends, trade volume, and market sentiment. The output is a recommendation to buy, hold, or sell based on the provided input values.