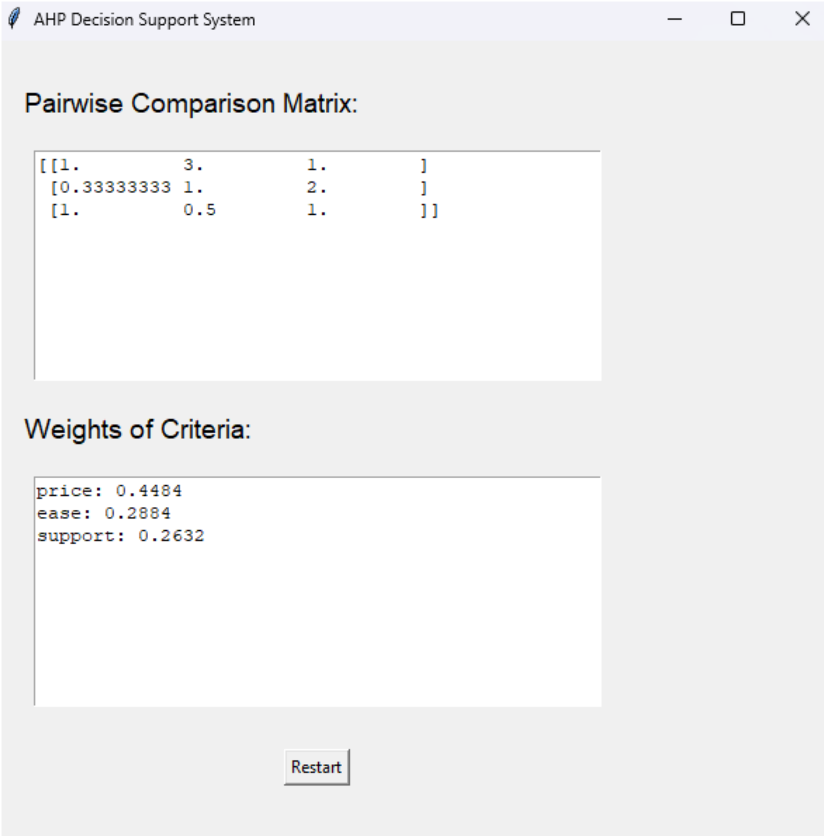
# Pairwise Comparison and AHP Algorithm




### Overview

This code implements the **Analytic Hierarchy Process (AHP)** method, which is a structured technique used to analyze complex decisions. The process breaks down a problem into a hierarchy, compares decision elements (criteria and options) in pairs, and calculates the relative importance of each element. The result is a set of normalized weights that can help decision-makers identify the best option based on the given criteria.

### Key Concepts

1. **Pairwise Comparison**: This is the process of comparing two elements (criteria or options) at a time to determine their relative importance. The user assigns a numerical value indicating how much more important one element is compared to another.

2. **Normalization**: After the pairwise comparison matrix is created, it is normalized so that the values in each column sum to 1. This ensures that the comparison matrix is balanced.

3. **Weight Calculation**: The weights of criteria are calculated by averaging the normalized values for each criterion.

4. **Consistency**: The AHP method includes a consistency check (though not implemented in this code) to ensure that the pairwise comparisons are logically consistent.

**Steps of the AHP Method in the Code**

**1. Input Criteria and Options**

The user is prompted to enter the decision criteria and the options to be compared.

Python code:

```
criteria = input("Enter the decision criteria separated by commas: ").split(',')

options = input("Enter the options separated by commas: ").split(',')
```

- **Criteria**: These are the factors by which the options will be evaluated. For example, "Cost", "Ease of Use", "Support".
- **Options**: These are the possible alternatives or solutions that need to be evaluated against the criteria. For example, "Option A", "Option B", "Option C".

**2. Pairwise Comparison Matrix for Criteria**

The user is asked to compare each pair of criteria by specifying how much more important one criterion is compared to the other. A matrix is then constructed, where each element represents the comparison of two criteria.

For example, if **Criterion 1** is more important than **Criterion 2**, the user would enter a number greater than 1 (e.g., 3). If **Criterion 1** is less important, the value would be less than 1 (e.g., 1/3).

Python code:

```
pairwise_matrix[i, j] = value

pairwise_matrix[j, i] = 1 / value
```

- pairwise_matrix[i, j] represents the importance of criterion i compared to criterion j.
- pairwise_matrix[j, i] = 1 / value ensures that the comparison matrix is reciprocal (if Criterion 1 is 3 times more important than Criterion 2, then Criterion 2 is 1/3 times as important as Criterion 1).

**3. Normalization of the Matrix**

Once the pairwise comparison matrix is created, it needs to be normalized. The normalization process ensures that the sum of each column in the matrix equals 1. This is done by dividing each element in the column by the sum of that column.

The formula for normalization is:

$$\frac{\text{Element}}{\text{Sum of Column}} = \text{Normalized Value}$$

Python code :

```
normalized_matrix = pairwise_matrix / pairwise_matrix.sum(axis=0)
```

- The axis=0 argument means that the normalization is applied along the columns.

**4. Weight Calculation for Each Criterion**

After normalizing the matrix, the weight of each criterion is calculated by averaging the normalized values across each row.

The formula for calculating the weight of each criterion is:

$$_{i}\text{Normalized Value} \sum_{i=1}^{n} \frac{1}{n} = \text{Weight of Criterion}$$

where n is the number of criteria.

Python code:

```
weights = normalized_matrix.mean(axis=1)
```

This gives the relative weight or importance of each criterion, which is then used in the decision-making process.

**5. Output**

Finally, the program outputs the following:

- The pairwise comparison matrix.
- The normalized matrix.
- The calculated weights for each criterion.

**Mathematical Formulas**

1. **Pairwise Comparison Matrix**:
   The pairwise comparison matrix ***A*** is constructed by comparing each pair of criteria. If *aij* represents the importance of criterion *i* compared to criterion *j*, then:

$$\text{value entered by the user} = {}_{ij}a$$

and for the reverse comparison:

$$\frac{1}{{}_{ij}a} = {}_{ji}a$$

2. **Normalization of the Matrix**:
   Each element aija_{ij}aij in the matrix is normalized by dividing it by the sum of the elements in its column:

$$\frac{{}_{ij}a}{{}_{ij}a \, {}_{i=1}^{n}\sum} = {}'_{ij}a$$

3. **Weight Calculation**:
   The weight of each criterion WiW_iWi is calculated by averaging the normalized values in its row:

$${}'_{ij}a \sum_{j=1}^{n} \frac{1}{n} = {}_{i}W$$

4. **Consistency** (Not Implemented in the Code, but important for AHP):
   The consistency of the pairwise comparison matrix can be evaluated by calculating the **Consistency Ratio (CR)**. The consistency ratio checks if the matrix follows logical consistency in pairwise comparisons.

The **Consistency Index (CI)** is calculated as:

$$\frac{\lambda_{\max} - n}{n - 1} = CI$$

where $\lambda\text{max}$ is the largest eigenvalue of the matrix. The consistency ratio is then:

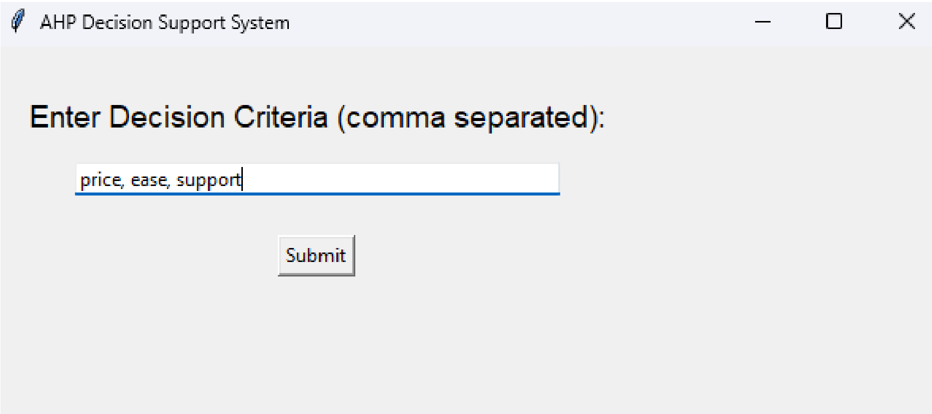
$$\frac{CI}{RI} = CR$$

where ***RI*** is a random consistency index based on the number of criteria.

**Example Scenario**

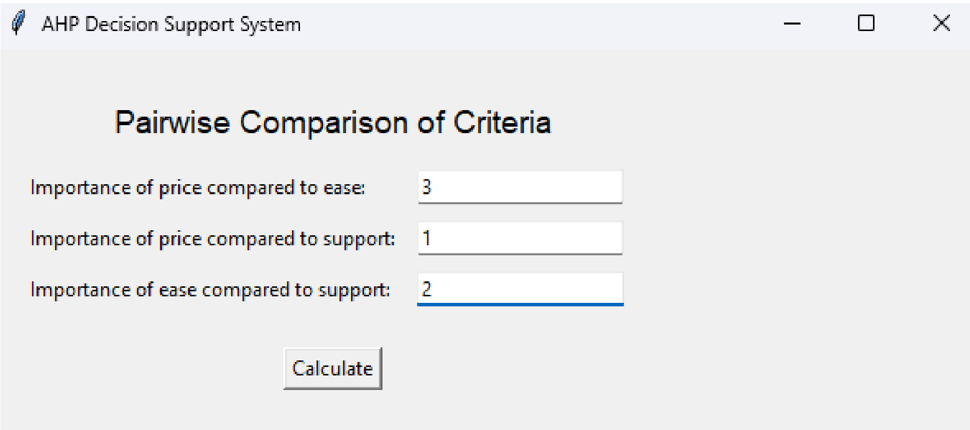
Assume you are selecting a project management software and have three criteria: **Cost**, **Ease of Use**, and **Support**. You are comparing three options: **Option A**, **Option B**, and **Option C**.

1. **Criteria**: Cost, Ease of Use, Support
2. **Options**: A, B, C

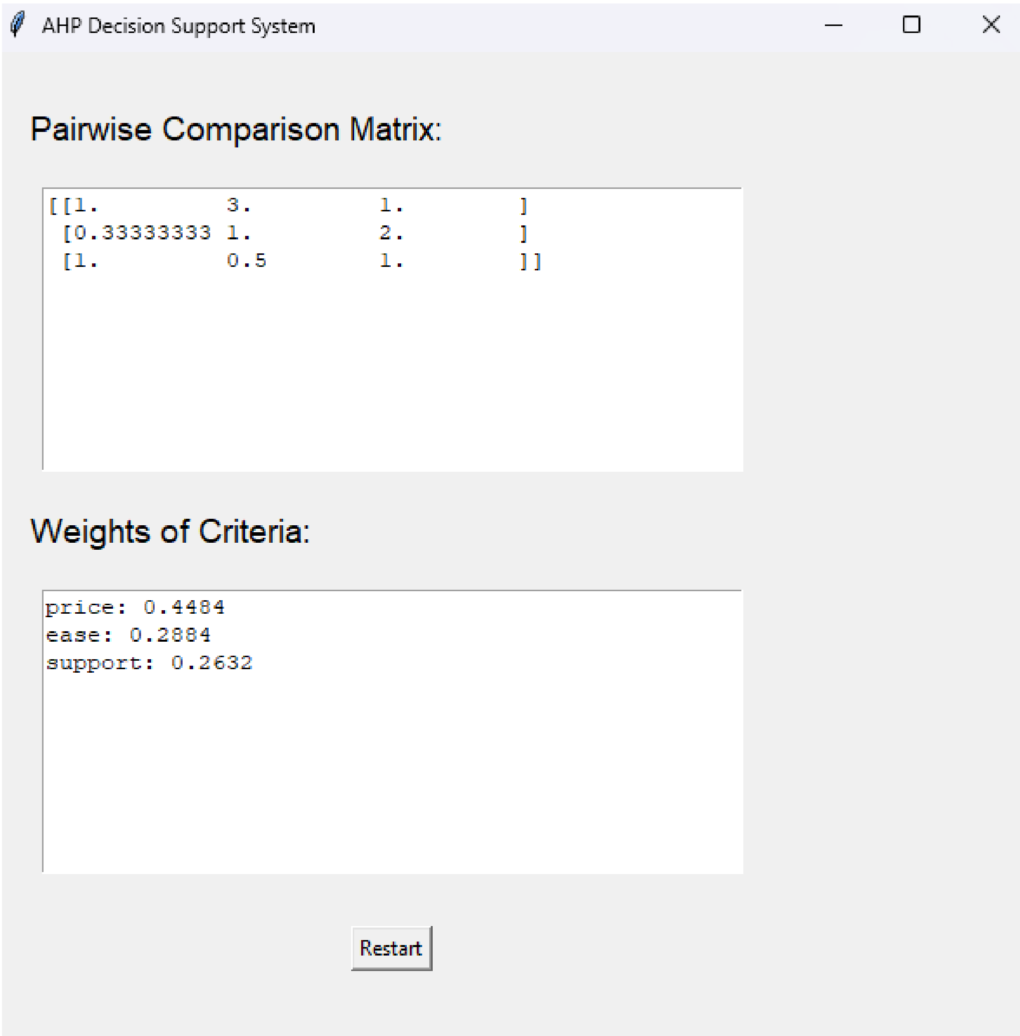



After running the program, the following happens:

- You compare **Price** to **Ease of Use** and decide that **Price** is 3 times more important, so you input 3.
- You then compare **Ease of Use** to **Support** and input 2 (because you think **Support** is less important than **Ease of Use**).



The program computes the weights and presents the pariwise comparison matrix and weights of criterias and user can identify the best option based on the computed importance of each criterion.




**Conclusion**

This code provides a basic implementation of the AHP method for decision-making. It involves gathering pairwise comparisons from the user, normalizing those comparisons, and then calculating the relative importance of each criterion. By applying these weights, the best decision can be identified based on the provided criteria and options.

The next step would typically involve performing consistency checks on the pairwise comparisons and further analysis of the options based on the calculated weights.