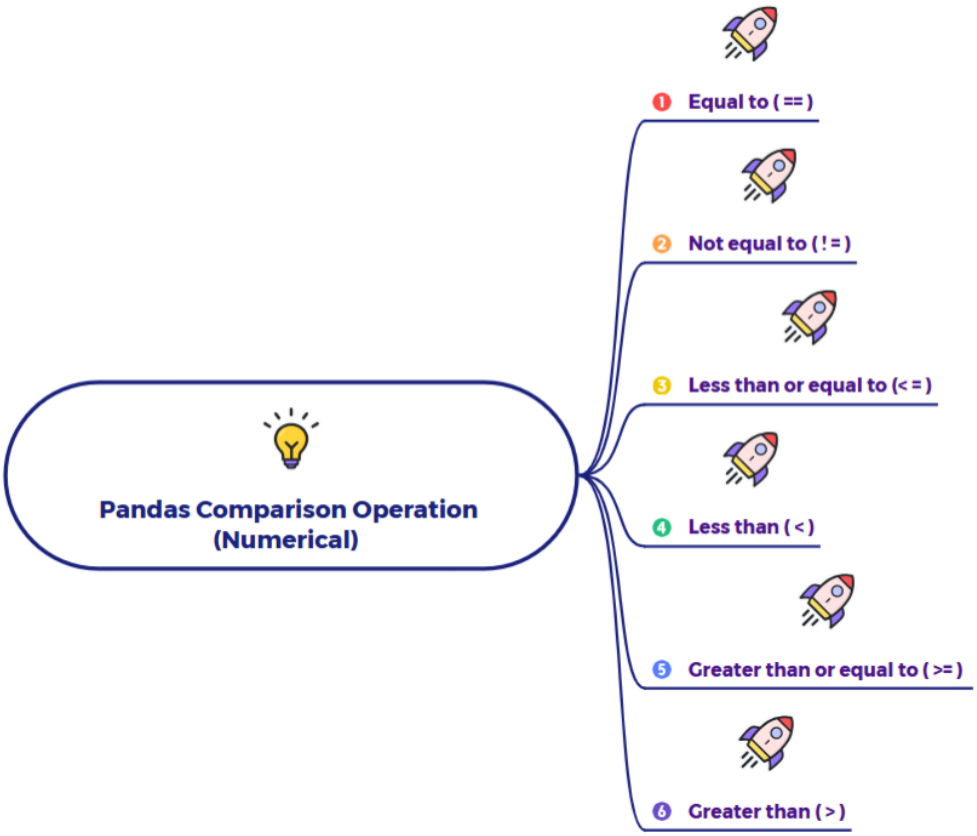
How to perform Comparison operation in Numerical Variable

Comparison operations in Pandas involve evaluating relationships between numerical values. These operations are fundamental for filtering data, identifying specific conditions, and performing conditional logic within your datasets. Instead of performing calculations that change the values, comparison operations ask questions about the values and return a True/False answer.



Purpose of Comparison Operations

The primary **purpose** of comparison operations is to **select subsets of data, validate conditions, or categorize data based on specific numerical criteria.** This allows you to:

- Filter rows that meet certain numerical thresholds (e.g., sales above a target).
- Identify outliers or values outside an expected range.
- Check for equality or inequality between values.

- Create boolean masks that can be used for advanced indexing and data manipulation.

How Comparison Operations are Handled and Why They Are Required

Pandas handles comparison operations element-wise, meaning it compares each corresponding value. The result of a comparison operation is always a **boolean Series or DataFrame** (containing True or False values).

The image highlights the standard comparison operators:

1. **Equal to (==):**
    - **What it does:** Checks if two numerical values are exactly the same.
    - **Why it's required:** Used to find exact matches (e.g., finding all products with a 'Price' exactly equal to 25.00).
2. **Not equal to (!=):**
    - **What it does:** Checks if two numerical values are different from each other.
    - **Why it's required:** Used to exclude specific values or find discrepancies (e.g., selecting all customers whose 'Age' is not 30).
3. **Less than or equal to (<=):**
    - **What it does:** Checks if the value on the left is less than or equal to the value on the right.
    - **Why it's required:** Useful for setting upper bounds or including a threshold (e.g., identifying all employees with 'Years of Experience' less than or equal to 5).
4. **Less than (<):**
    - **What it does:** Checks if the value on the left is strictly less than the value on the right.
    - **Why it's required:** Used for setting strict upper bounds (e.g., finding all orders with 'Quantity' less than 10).
5. **Greater than or equal to (>=):**

- **What it does:** Checks if the value on the left is greater than or equal to the value on the right.
- **Why it's required:** Useful for setting lower bounds or including a threshold (e.g., selecting all sales records where 'Revenue' is greater than or equal to 1000).

6. **Greater than (>):**
    - **What it does:** Checks if the value on the left is strictly greater than the value on the right.
    - **Why it's required:** Used for setting strict lower bounds (e.g., identifying all customers with 'Purchase Count' greater than 50).

How they are used (Boolean Indexing/Masking):

The boolean Series or DataFrame generated by these comparison operations is most commonly used for **boolean indexing (or masking)**. This means you can pass the True/False Series directly into your DataFrame's selection brackets to retrieve only the rows where the condition is True.

Example Scenario (Conceptual):

Imagine a DataFrame of customer data with an 'Age' column. If you want to find all customers who are 18 years or older, you would perform a comparison operation on the 'Age' column (df['Age'] >= 18). This would return a Series of True and False values. You then use this Series to filter your original DataFrame, effectively selecting only the rows (customers) where the age condition is met.

Why they are required:

Comparison operations are fundamental building blocks for:

- **Data Filtering:** The most common use case, allowing you to narrow down your dataset to specific observations of interest.
- **Data Validation:** Checking if data falls within expected ranges or meets certain criteria.
- **Conditional Logic:** Creating new columns based on conditions (e.g., a 'High Value Customer' column that is True if 'Total Spend' > 500).

- **Data Segmentation:** Dividing your data into different groups based on numerical properties.

In essence, comparison operations provide the means to ask precise questions about your numerical data and retrieve the exact subsets that answer those questions, making them indispensable for data exploration and preparation.